

## Java Programmer Certification Notes

An *identifier* is an unlimited-length sequence of *Java letters* and *Java digits*, the first of which **must be a Java letter**. [4chevy, all/clear, get-lot-fred are illegal, not contain space or #]

The **\$ character** should be used only in mechanically generated Java code or, rarely, to access pre-existing names on legacy systems.

## Keywords

The following character sequences, formed from ASCII letters, are reserved for use as *keywords* and **cannot be used as identifiers (§3.8): (46 words)**

Modifier	Primitive Types	Loops	Access	Exception	Inheritance	Miscellaneous
abstract	boolean	if	private	throw	class	new
final	byte	else	protected	throws	interface	const
native	char	do	public	try	super	goto
volatile	double	while	package	catch	this	instanceof
static	float	for	import	finally	implements	
synchronized	int	switch			extends	
transient	long	case				
	short	default				
	void	break				
		continue				
		return				

Not used

The keywords `const` and `goto` are reserved by Java, even though they are not currently used in Java.

### Reserved Literals

While `true` and `false` might appear to be keywords, they are technically Boolean literals (§3.10.3). Similarly, while `null` might appear to be a keyword, it is technically the null literal (§3.10.7).

There `null`, `true`, `false` are reserved as **predefined literals not keywords**

	Datatype	width(bits)	Min value Max value	Wrapper class
1	boolean	NA	true, false	Boolean
2	byte	8	$-2^7, 2^7 - 1$	Byte
3	short	16	$-2^{15}, 2^{15} - 1$	Short
4	char	16	0x0, 0xffff [must be in single quotes & not double quotes] i.e from 0 to 65535	Character
5	int	32	$-2^{31}, 2^{31} - 1$	Integer
6	long	64	$-2^{63}, 2^{63} - 1$	Long
7	float	32	$\pm\text{MIN\_VALUE} = 1.4\text{e-}45\text{f};$ $\pm\text{MAX\_VALUE} = 3.4028235\text{e+}38\text{f};$	Float
8	double	64	$\pm\text{MIN\_VALUE} = 5\text{e-}324;$ $\pm\text{MAX\_VALUE} =$ $1.7976931348623157\text{e+}308;$	Double

0xY is hexadecimal literal, while 0Y is octal literal

/uxxx can be used anywhere in source code to represent Unicode char

e.g. `char a = '\u0061'`, `char \u0061 = 'a'`, `ch\u0061r a = 'a'` are valid

'0' to '9' is `\u0030` to `\u0039` | 'A' to 'Z' is `\u0041` to `\u005a` | 'a' to 'z' is `\u0061` to `\u007a`

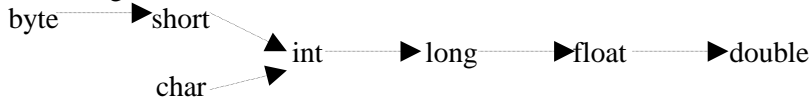
" = `\u0020` \b = `\u0008` \t = `\u0009` \n = `\u000a` \r = `\u000d`

A floating-point literal is of type `float` if it is suffixed with an ASCII letter `F` or `f`; **otherwise its type is double (default type of floating point literal is double)** and it can optionally be suffixed with an ASCII letter `D` or `d`.



## Type Casting

Widening



```
char c = 'a';
```

```
int i = c; //implicit widening
```

**Unary Numeric Promotion** converts operands **byte, short and char to int** by applying an implicit widening conversion (+, -, ~, array creation new int[5], indexing array arr['a'] evaluate to int value, <<, >>, >>>)

**Binary Numeric Promotion** If T is broader than int, both operands are converted to T otherwise both operands are **converted to int** (arithmetic, relational, equality, bitwise)

Assigning references does not copy the state of the source object on the RHS, only the reference value. (creates aliases)

### Type conversions on assignment

If destination is char, byte, short & source is an int whose value can be determined to be **in range of destination type**, implicit narrowing occurs. e.g.

*narrowing primitive conversions involving int literals*

```
short s = 10, char sym = 12 //int in range no cast required
```

```
byte t = (byte) 128 //int not in range (- 128 to 127) cast required
```

*narrowing primitive conversions involving int variable*

```
int i = -20;
```

```
final int j = 20; //constant
```

```
final int k = i; //value determinable only at runtime
```

```
byte b1 = j; //final value in range, no cast required
```

```
byte b2 = (byte)i; //value of i not determinable cast required
```

```
byte b3 = (byte)k; //value of k not determinable cast required
```

All other narrowing requires explicit cast

```
float f1 = (float)100.5D;
```

```
int in = (int) f1;
```

Narrowing between char, byte & short requires explicit cast

```
short val = (short)'a';
```

```
byte b = 32;
```

```
char c = (char) b;
```

-4.0/0.0 result -INF (NEGATIVE\_INFINITY)

0.0/0.0 result NaN (Not a Number)

### Numeric promotion in arithmetic expression

```
byte b = 3; //int in range, no cast required
```

```
b = (byte) -b; //explicit narrowing conversion required on assignment as -b becomes int
```

```
int i = ++b;
```

### For Extended Assignment Operators (opr=)

**x\*=a** **x=(T) (x \* a)** is implied

e.g. byte b = 2;

```
b += 10; //no cast required as it implies b = (byte) (b +10);
```

```
but b = b+10; //compile error. Explicit type cast required
```

## Strings

```

package testPackage;
class Test {
public static void main(String[] args) {
String hello = "Hello", lo = "lo";
System.out.print((hello == "Hello") + " "); //true string literals in same Cl & Pkg
System.out.print((Other.hello == hello) + " "); //true string literals in diff Cl &
same Pkg
System.out.print((other.Other.hello == hello) + " "); //true string literals in diff Cl
& diff Pkg
System.out.print((hello == ("Hel"+"lo")) + " "); //true string comptd at compile treated as literl
System.out.print((hello == ("Hel"+lo)) + " "); //false strings comptd at runtime r distinct
System.out.println(hello == ("Hel"+lo).intern()); //true result of explicitly interning
}
}

```

class Other { static String hello = "Hello"; }

and the compilation unit:

### package other;

```
public class Other { static String hello = "Hello"; }
```

produces the **output**:

```
true true true true false true
```

This example illustrates six points:

- Literal strings within the same class (§8) in the same package (§7) represent references to the same `String` object (§4.3.1).
- Literal strings within different classes in the same package represent references to the same `String` object.
- Literal strings within different classes in different packages likewise represent references to the same `String` object.
- Strings computed by constant expressions (§15.27) are computed at compile time and then treated as if they were literals.
- Strings computed at run time are newly created and therefore distinct.
- The result of explicitly interning a computed string is the same string as any pre-existing literal string with the same contents.

equals

<b>Equality for String objects means same character string</b> e.g. String m1 = "Hello" String m2 = "Hello" m1.equals(m2) is true	<b>Result of applying "==" operator to any two objects of any type:</b> String s1 = "hi"; String s2 = "hi"; System.out.println(s1==s2); //true string literals
<b>Equality for Boolean objects means same primitive value</b> e.g. Boolean b1 = new Boolean(true); Boolean b2 = new Boolean(true); then b1.equals(b2) is true	String s1 = new String("hi"); String s2 = new String("hi"); System.out.println(s1 == s2); //false different references  System.out.println(b1 == b2); //false different references
<b>a.equals(null)</b> //NullPointerException at runtime	

```

StringBuffer sb1 = new StringBuffer("hi");
StringBuffer sb2 = new StringBuffer("hi");
System.out.println(sb1 == sb2); //false different references
System.out.println(sb1.equals(sb2)); //false NOTE: StringBuffer does not override the equals()
method & hence compare the obj references

```

## Shift Operator Examples

<<

```
byte a = 32, b;
```

```
int j;
```

```
j = a<<3           //256    32*23
```

```
b = (byte) a<<3;   //0      only last 8bits are considered
```

```
a<<3
```

```
= 0000 0000 0000 0000 0000 0000 0010 0000 <<3
```

```
= 0000 0000 0000 0000 0000 0001 0000 0000
```

```
= 0x0  0  0  0  0  1  0  0
```

```
= 0x0000100
```

```
= 256
```

```
1098  7654 3210 9876 5432 1098 7654 3210
```

>>

```
byte a = -42;      //1101 0110
```

```
int result = b>>4; // -3
```

```
b>>4                //FILLS with whatever the MSB bit is
```

```
= 1111 1111 1111 1111 1111 1111 1101 0110 >>4
```

```
= 1111 1111 1111 1111 1111 1111 1111 1101 >>4
```

```
= 0xffffffa        (2's complement of 1101 = 3) 0010
```

```
= -3                0001  
                   0011
```

## Parameter Passing

In Java, **all parameters are passed by value**

For **primitives**, you pass a **copy of the actual value**.

For **references** to objects, you pass a **copy of the reference**

- **Passing Primitives**

If Actual Parameter is **Primitive datatype**, it is copied to the formal parameter. Formal parameter acts as local variable in the method & cannot change the actual parameter.

- **Passing References**

If the actual parameter is a reference to an object (instantiation of class or array of reference to object) then the **reference value is passed and not the object itself**.

i.e. actual & formal parameters are aliases to the same object during invocation of the method  
the **change is only to the state(members) of the object not to the reference itself**.

## Arrays

- In Java, Arrays are object. An array with length  $n$  can be indexed by the integers 0 to  $n-1$ . therefore  $i^{\text{th}}$  element has index  $i-1$
- Arrays with an interface type as the component type are allowed. (component = any class type that **implements the interface**)
- Arrays with an abstract class type as the component type are allowed. (component = any subclass of the **abstract class that is not itself abstract**.)
- Arrays **must be indexed by int values; short, byte, or char** values may also be used as index values
- All array accesses are **checked at run time**.
- The size of the array is never given during declaration of array reference. (implicit by init)

e.g. `int a[4] = {1,2,3,4}` //illegal  
`int a[] = {1,2,3,4}` //legal

declarations

```
int a[][] = new int [4][4];    //legal
int[][] a = new int[4][4];    //legal
int[] a[] = new int[4][4];    //legal
int []a[] = new int[4][];     //legal note [][][4] not allowed
```

- Array Type do not have a heirarchy & **cannot be cast to another primitive array type** e.g.  
`byte [] ba = {1,2,3,4};`  
`int [] ia = ba;` //COMPILE time error incompatible types found : byte[] | required: int[]

The members of an array type are all of the following:

- **length** - The `public final` field `length`, which contains the number of components of the array (**length may be positive or zero**)
- **clone()** - **All the members inherited from class Object**; the only method of `Object` that is not inherited is its `clone` method. The `public` method `clone`, which **overrides** the method of the same name in class `Object` and throws no checked exceptions

A `clone` of a multidimensional array is shallow, which is to say that it creates only a single new array.

**Subarrays are shared.**

- The string "`[]I`" is the run-time type signature for the class object "array with component type `int`"

## Scope & Accessibility

	UML notation	Members
public	+	Accessible everywhere
protected	#	any class in same pkg & subclass in other pkg
default		only classes in same pkg (Package accessibility)
private	-	Not accessible outside the class it is defined in

<b>PUBLIC</b>	Sub class (inherited)	Any class
Same Pkg	Y	Y
Outside Pkg	Y	Y

<b>PROTECTED (nested class)</b>	Sub class (inherited)	Any Class
Same Pkg	Y	Y
Outside Pkg	Y	No

A SC in another pkg can only access protected members in the superclass via reference of its **own type or subtype**

<b>DEFAULT</b>	Sub class (inherited)	Any class
Same Pkg	Y	Y
Outside Pkg	No	No

<b>PRIVATE (nested class)</b>	Sub class (inherited)	Any Class
Same Pkg	No	No
Outside Pkg	No	No

### Access Modifiers for class

Classes cannot be abstract & final at the same time

#### **abstract**

- class may contain abstract (no method body) / non-abstract methods(instance methods) / final methods
- cannot be instantiated (no 'new')
- interface is implied

#### **final**

- cannot be extended** ie. subclassed no inheritance
- all methods are implicitly final
- no interfaces

### Access Modifiers for members

	Variables	Methods
<b>static</b>	class variable (applicable to instance, static, local variables)	<ul style="list-style-type: none"> <li>Static methods can access only static members</li> <li>overridden method should also be static</li> <li>cannot use this.x in main() as it is static</li> </ul>
<b>final</b>	constants	methods cannot be overridden
<b>abstract</b>	N.A	no method body
<b>synchronized</b>	N.A	one thread at a time
<b>native</b>	N.A	method implemented in another language C/C++
<b>transient</b>	value will not be persistent if the object is serialized	N.A
<b>volatile</b>	value can change asynchronously	N.A.

#### **static**

- Known at Compile time- **Static methods does not come under runtime Polymorphism.** They are known at compile time itself. The CLASS owns the STATIC method and not the object of the class. So only the class type determines which static method is invoked and not the object at runtime which is determining the static method.
- There is NO WAY A REFERENCE of Superclass can call the static method of the subclass
- Static methods have an implicit object reference this and must **always supply an explicit object** reference when referring to **non-static members.**

```

class MyClass{
    static MyClass rf;
    String[] ag;
    public static void main(String args[]){    //static method
        rf = new MyClass();
        rf.func(args);
    }
    public void func(String[] args)        //non static method
    {
        rf.ag = args;        //explicit object reference needed
        System.out.println(rf.ag[1]);
    }
}

```

### *final*

- must be **initialized once** before it is used
- final when used as argument in a method
  - primitive - can not change the value
  - reference cannot change the reference
- reference cannot be changed but state/value can be changed
  - e.g. final Light aL = new Light();
    - aL .watts = 60; //state can be changed
    - aL = new Light() //NOT OK. no changing final reference
- Blank Finals
  1. final not initialized when declared are blank finals
  2. **blank finals must be initialized in the constructor**
  3. Each object can have different value (using random) & yet retain immutability quality

### *native*

e.g. generally use static initializer

```

static {
    System.loadLibrary("NativeLib");
}
native void nativeMethod();

```

### *transient*

- Transient modifier should not be specified for static variables as these do not belong to objects
- **Transient variable can be both static and final**
- The class need not be Serializable or Externalizable to declare its fields as "transient" [ though its meaningless to do so ]
- But to write the instance of the class ( serialize) into a stream, the class need to implement Serializable or Externalizable, else "**NotSerializableException**" will be thrown.
- Objects can be stored using serialization i.e. they can be later retrieved in the same state as when they were serialized. these are persistent objects.
- Transient indicates that the variable need not be saved when the object is put in persistent storage.

```

class x implements Serializable{ //Serializable saves the value of the non-transient instances
    transient int Temperature;
}

```

### *volatile*

Since thread can share variables, there can be inconsistency in value of the variable.  
 Volatile modifier informs the compiler that it should not attempt to perform optimization on the variable i.e. variable modification is allowed. e.g. volatile long clockReading (return correct current time)

## Flow Control

`{ }` is a valid statement block

if - else

condition statement can contain

- method calls
- only expr which evaluates to *boolean* value can be used as condition

`if(false); else; // is legal`

switch

- control falls through next statement unless appropriate action is taken
- all labels are optional
- at most one *default* label
- labels can be specified in any order
- type of the expression must be char, byte, short or int [**cannot be boolean /long /floating pt**] the type of case labels must be assignable to the type of switch expr
- can be nested & labels can be redefined in the nested blocks

for

- all expressions in header are optional (two semicolons are mandatory) `for(;;)` is infinite loop
- multiple variables can be given but must be of the **SAME** type

do | while

- condition must evaluate to *boolean* value
- `while(true)` is an infinite loop

break

- transfer control out of the current context (closest enclosing block)
  - **not possible to break out of if statement**
- e.g. `if(true) { break; }`
- **but if it is placed inside a labeled block, switch or loop usage of 'break' in if is valid.**
  - for loop : out of the loop body, **terminating the loop** & going to next statement after loop
  - break outer - comes out of the loop labeled as 'outer' e.g.  
`block: { break block; } //valid`

continue

- can be used with for , while, do-while loop
  - the rest of the loop is skipped with the execution continuing with the `<incr expr>`
- `block: { break continue; } //is invalid`

scope

```
{
  int x = 12;
  {
    int x = 96; //illegal use of x
  }
}
```

```
{
  int x = 12;
  {
    int q = 96; //scope of x & q
  }
} //only x
```

## Exceptions

checked exceptions

- **except RuntimeException (Arithmetic, Null Pointer..), Error (AWT, linkage, thread, VM)** and their subclass, all exceptions are checked exceptions
- must explicitly deal with it
- An overriding method can't throw more or broader checked exceptions - this is not true for overloading methods

try-catch-finally

- block notation is mandatory
- must be in try-catch-finally order
- for each try block zero or more catch block but only ONE finally block
- if **finally block is specified it is guaranteed to be executed** regardless of cause of exit (cause of exit can be a catch, normal exit, return)

catch block

- type of exception must be assignable to the type of argument in the catch block
- header of catch takes exactly one argument
- code of first match of catch block is executed and all other catch blocks are skipped & execute finally
- compiler **complains if the catch of the superclass exceptn shadows** the catch block of the subclass exceptn e.g. Exception & then ArithmeticException is specified //error then order must be changed.
- find the catch block that handles it, then do finally block else exception handled by default top-level exception handler will have to catch it.

e.g. RuntimeException re = null;

```
throw re; //NullPointerException occurs as throw requires a throwable object
```

throw

e.g. try{

```
if (n1 == 0) throw new DivbyZero("/ by 0")
} catch (DivbyZero e){
}
```

throws

- can be a superclass of actual expression thrown e.g.  
class Divbyzero extends Exception{ }  
public void div() throws DivbyZero{ } -----a  
public void div() throws Exception{ } -----b are valid
- overriding method in the subclass can only specify **none, all or a subset** of the exception classes
- The main method can declare that it throws checked exceptions just like any other method

Exception

ClassNotFoundException | ClassNotSupportedException,

IllegalAccessException | InstantiationException | InterruptedException | NoSuchMethodException

RuntimeException ->

EmptyStackException, NoSuchElementException, **ArithmeticException,**

ArrayStoreException, **ClassCastException, IllegalMonitorStateException,**

NegativeArraySizeException, **NullPointerException, SecurityException.**

IllegalArgumentException -> (IllegalThreadStateException, NumberFormatException)

**IndexOutOfBoundsException** -> (ArrayIndexOutOfBoundsException,

StringIndexOutOfBoundsException)

AWTException

IOException ->

EOFException, FileNotFoundException, InterruptedIOException,

UTFDataFormatException, MalformedURLException, ProtocolException, SocketException,

UnknownHostException, UnknownServiceException.

## OOPs

- extends clause is used to specify inheritance
- In Java , classes can only extend a single class but there is no limit to how many classes can extend the same class
- **All members of the superclass are inherited by the subclass** (inheritance is different from accessibility)
- A subclass reference can be assigned to a superclass reference e.g. Object objRef = stringRef;
- cannot invoke the methods EXCLUSIVE to subclass via the superclass reference  
e.g. objRef.length() //compile error as length is a method of subclass String
- the actual method invoked is determined by **dynamic method lookup at runtime** e.g objRef.equals("Java")  
//calls the overridden method equals from String class & not Object class
- need to explicitly cast the value of superclass reference to a subclass type (down-casting)

The subclass of a non-abstract class can be declared abstract

**All arrays are objects.**

## Method Overriding / Variable Shadowing / Method Overloading

- must have the same method signature & same return type
- cannot "narrow" accessibility

this.x = p, this(3) ----calls constructor, this.dovalue() ---method call  
super() ---constructor call

### Overriding Methods

1. Only methods **that can be accessed** can be overridden.
  - **private methods cannot be overridden**
  - **final methods cannot be overridden.**
  - **overridden static method must also be static**
2. specify none, all or subset of exceptions specified in throws clause of overridden method (When you override a method, you can throw only the exceptions that have been specified in the base-class version of the method.)
3. Aliasing happens automatically during parameter passing (actual reference does not change)
4. parameters can be final in overridden method
5. subclass can use the keyword super to invoke the method in super class
6. **super.super.x() is not allowed** as super is a keyword not an attribute (1)
7. Variables are shadowed not overridden
8. cast only changes the type of reference not the class of the object see (2)
9. **casting the this** reference has no effect on which method is invoked (3)

```
class Superclass{
    protected String x = "Super variable";
    protected void methodA(){ System.out.println("Super method"); }
    public void banner(){ System.out.println("Let there be Light"); }
}
```

```
class Subclass extends Superclass{
    public String x = "Sub variable";
    //private void methodA(){ //ERROR attempting to assign weaker access privileges;
    protected void methodA(){ System.out.println("Sub method"); }
    public void banner(){ System.out.println("Let there be Tube Light"); }
}
```

```
class Childclass extends Subclass{
    public String x = "Child variable";
    public void methodA(){
        System.out.println("\nUse of super & this keyword");
        super.banner(); //call the method from immediate parent class if it exists
                        // else looks for the class above parent in heirarchy & so on
                        //super.super.banner() is not allowed to access top-level class method-(1)
        System.out.println("Child method");
        ((Superclass)this).banner(); //still prints the method from immed parent ---(2)
        System.out.println(((Superclass)this).x + "\n"); //prints the variable from super class --(3)
    }
}
```

```
public class test{
    boolean methodB(Superclass A){ return true; } //overload
    boolean methodB(Subclass A){ return false; }
    public static void main(String args[]){
        Subclass sc = new Subclass(); // sub -> sub
        Superclass sp = sc; // super ---> sub
        Superclass ssp = new Superclass(); // super -> super
        Childclass ch = new Childclass();
    }
}
```

```

//Overridden Method
sc.methodA();      // "Sub method"
sp.methodA();      // "Sub method" current object type method is invoked (use super)
ssp.methodA();     // "Super method"
ch.methodA();      // Let there be Light , Child method, Let there be Light, Super Variable

//Shadowed Variables
System.out.println(sc.x);      // "Sub variable"
System.out.println(sp.x);      // "Super variable" type of reference is used
System.out.println(ssp.x);     // "Super variable"
System.out.println(ch.x);      // "Child variable"
//Overloading
test t = new test();
System.out.println(t.methodB(sc)); // false sp has 2 methodB - superclass /subclass
// more specific one is chosen

System.out.println(t.methodB(sp)); // true
System.out.println(t.methodB(ssp)); // true
}
}

```

## Constructors

- Constructors have no return type i.e. void classname() is a method & not a constructor
  - Constructors can be private, protected or public
  - **Private Constructors** Suppose that you want to create a class but you don't want it to be instantiated as an object. Just declare the class's constructors as private.
  - Constructors cannot be final , static or abstract.
  - Constructors **cannot be overridden** but can be locally (same class) overloaded
  - Local chaining of constructors is possible with the use of *this* reference
- e.g. A(){this(0,true) } calls the appropriate constructor  
A(int,boolean) {this(a,b,"X")} and so on  
A(int,boolean,char) {.....}
- super() is used in subclass to invoke constructors of the immediate superclass
  - Java specifies that when using this() or super() it must occur as the first statement in a constructor (i.e both can not occur in the same constructor)
  - if constructor does not have this() or super() then a super() call to the default constructor of super class is inserted (e.g. first constructor called is Object class) provided the subclass does not define non-default constructors in which case call super() constructor with right args e.g. super(0,true,"X") or it will look for default constructor in superclass
- e.g. class A{  
A(){this("1","2")}; A(x,y) {this(x,y,"Z")}; A(x,y,z){...}  
}  
class B extends A{ B(string s) //will automatically call default constructor from a super()  
{ print s; }  
e.g. Mysub(int x, int y){super(num); count = x};  
Mysub(int x) { this(x,x);

## Interfaces

- interface is abstract by definition & cannot be instantiated , all declarations are public
- interface methods
  - must be public
  - cannot be static
  - all methods are implicitly abstract
- constants (variables) in interfaces
  - are public, static, & final
  - access can be done without using dot(.) notation e.g.  
interface Constants{ ... }  
LENGTH\_UNITS or Constants.LENGTH\_UNITS
- a class can implement an interface partly or wholly
  - wholly then all the methods must be implemented
  - if partly then declare the class as abstract
- multiple interface inheritance is allowed
- interface can extend several interfaces

<i><b>Abstract Class</b></i>	<i><b>Interface</b></i>
must not be instantiated	must not be instantiated
may contain static and final data	variables are implicitly static and final
abstract class can have non-abstract methods but, abstract method should be inside an abstract class	methods are implicitly abstract. Therefore, all methods should be implemented in the subclass which implements it. no method implementation strictly in the interface.
abstract method should not contain any of these keywords - private, final, static, native, synchronized [ <b>pfnsns</b> ]	methods in interface should not contain any of these keywords - protected, private, final, static, native, synchronized [ <b>ppfsns</b> ]
methods are not implicitly public	methods are implicitly public even if not specified (be careful when overriding)
can have constructors (should contain body)	interfaces can't have constructors

Normally you cannot put any code inside an interface, but a static inner class can be part of an interface

```
class Interface{  
    static class inner{  
        int i,j;  
        public inner() {}  
        void f(){ }  
    }  
}
```

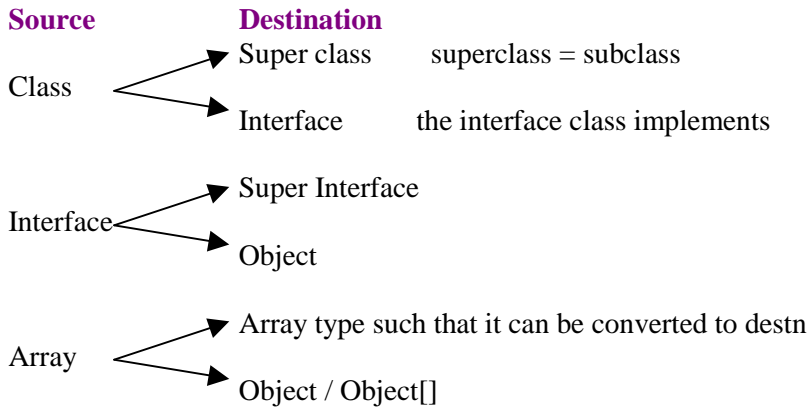
Rules for Reference assignments takes place at compile time [not runtime]

Always draw hierarchy diagram subclass → superclass, child - - > super interface

Sourcetype srcRef;

DestinationType dstRef = srcRef;

dstRef = srcRef when



instanceof operator

- If instanceof operator is false then the cast involving the operands will throw a ClassCastException at runtime i.e. It is the actual object reference at runtime is compared with the type specified on the RHS and not the type of the reference
- literal null is not an instance of any reference type  
boolean t2 = "String" instanceof Object //is true as always creates a string object for it ie extends Obj  
boolean t1 = null instanceof Pizza //Always false null not an instance
- An instance of superclass is not an instance of subclass
- An instance of a class is not an instance of an unrelated (or Peer class) class.
- An instance of a class is not an instance of an interface that it does not implement (unrelated)
- An instance of a array of non-primitive type is an instance of both Object & Object[] types

```

class A {}
class B extends A {}
class C extends B {}
class B1 extends A {}
class C1 extends B1 {}
A objA = new B(); //subclass to superclass
//String s = (String) L1; compile time error inconvertible types
//r1 = L1 instanceof String; compile time error inconvertible types
System.out.println( objA instanceof B1); //false Peer Class
//B1 bB1 = (B1) aA; //runtime error java.lang.ClassCastException: B
System.out.println( objA instanceof C); //false Superclass not instance of subclass
//C objC = (C) objA; //runtime error java.lang.ClassCastException: B
A objC1 = new C1();
System.out.println( objC1 instanceof B1); //true SubClass instance of super class
B1 objB1 = (B1) objC1; //cast required from type A to B1
A[] arrA;
B[] arrB;
arrA = new A[10];
arrB = new B[20];
System.out.println( arrB instanceof A[]); //true SubClass instance of super class
//arrB = arrA; //compile time error incompatible types (sub=super)
arrB = (B[])arrA; //runtime java.lang.ClassCastException:
arrA = arrB; //change reference A to B
arrB = (B[])arrA; //explicit cast required super to sub
  
```

Top-Level Nested Class (DEFINES BOTH static & non-static members)

- top-level class can have only default or public modifier but its **nested class can have any modifier**
- nested class defined within the enclosing class with the static keyword
- The **static** nested class is tied only to the outer class, **not an instance** of the outer class. e.g. Outer.Inner i = new Outer.Inner(); // i instanceof Outer = false
- A static nested top-level class can be instantiated w/o any reference to any instance of the enclosing context / nesting. Inner n = new Inner(); // **outer instance is not a must for static class**

static nested class	Enclosing context	OWN
Static methods (no <i>this</i> reference)	only static	only static
Non-static methods	only static	all

**Inner Classes - All inner classes can DEFINE ONLY non-static members**

**Inner classes can have STATIC FINAL members.** Outer class has a distinct type from enclosing class & instanceof does not take the outer class type into consideration

Types of inner class		Modifier for class & local variables	Outer Instance exist	Access to Enclosing context members
Non-Static Inner Class		all	Yes	all members
Local Class (as member variables - in blocks class within a <b>method, constructor</b> )	Non-Static	none	Yes	all members - enclosing class local final variables - enclosing method
	implicitly Static if context is static		No	<ul style="list-style-type: none"> <li>▪ static members (encl class)</li> <li>▪ local final variables (encl method)</li> </ul>
Anonymous Inner class (as <b>expressions</b> )	Non-Static	none	Yes	all members local final variables
	implicitly Static if context is static		No	<ul style="list-style-type: none"> <li>▪ static members</li> <li>▪ local final variables</li> </ul>

Non-static inner class

- nested class defined within the enclosing class without the static keyword
  - an instance of a non-static inner class can **ONLY** exist with an instance of its enclosing class.  
**<enclosing object reference>.new** e.g topref.new NonStaticclass();
  - **multiple objects** of the inner classes can be associated with an object of an enclosing class at runtime
- ```

Toplevelclass t
├─t.InRef1
└─t.InRef2
    
```
- can refer to members of enclosing class including private members
  - special this to access members of enclosing class **<enclosing class name>.this.<member>** [shadow]

Local classes

- A local class cannot be specified the keyword static
- getClass() returns <enclosing classname>.\$1\$localclassname

| only non-static members can be defined in local class                                                                                                                         | Enclosing Context method | Enclosing class | Super class of Enclosing class | Super class of local class |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|-----------------|--------------------------------|----------------------------|
| Non Static local class<br>1.requires an instance of enclosing class<br>2.return type of enclosing context is generally a <b>supertype</b> of the local class                  | final                    | all             | all                            | all                        |
| Static Local class<br>1.cannot be declared static, implicitly static if enclosing context is static<br>2.can be invoked either through the class name or an instance of class | final                    | only static     | only static                    | all                        |

## Anonymous classes

- **Access rules and Instantiation rules** for anonymous class **are same** as for local classes
- no name, combines the process of definition & instantiation into a single step
- context decides if the class is static or non-static
- anonymous class cannot define constructors no extends clause is used in the construct e.g.  
( **new <superclass classname>** (<optional argument list>)  
    { <class declaration> }  
);

for interface

```
(new <interface name> ()  
    {<class declarations> }  
); -> no implements is used
```

getClass() on anonymous class returns \$<nos> i.e. Painter\$1 , Painter\$2 for the 2 anonymous classes

- **anonymous class implicitly extends the Object class**

Declaration: 1>Outer o = new Outer();

```
    // Inner i = new Inner(); // NO! Can't instantiate Inner by itself!
```

```
    Outer.Inner i = o.new Inner();
```

2> Inner i = new Outer().new Inner(); //instance of outerclass a must if non-static inner class

- **Either override or implement abstract methods of the superclass in an anonymous class.** Any other member cannot be accessed.

e.g.

```
class Painter{  
    public Shape createShape () {                      //non static anonymous class  
        return new Shape(){.....  
            public void draw() {}  
        }; // -> no extends is used for a super class Shape  
    }  
  
    public static IDraw createIDraw () {              //static method  
        return new IDraw(){.....                    //hence static anonymous class  
            public void draw() {}  
        }; // -> no implement is used for a implementing interface  
    }  
}
```

```
new Painter().new createShape();                    //enclosing instance context a must for non-static
```

```
Painter.new createIDraw();                         //enclosing instance context not mandatory static
```

|                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Nested Top-Level class [not an inner class]</p> | <ul style="list-style-type: none"> <li>• Nested Top-Level class is a nested class that is declared "static"</li> <li>• Nested top-level classes are used as a convenient way to group related classes.</li> <li>• This is, but a new kind of top-level class</li> <li>• Since static doesn't have a "this" pointer to an instance of the enclosing class, a nested class has no access to the instance data of objects for its enclosing class.</li> <li>• Any class outside the declaring class accesses the nested class with the declaring class name acting similarly to a package.(eg, If class "top" has nested top-level class called "myNested", this would be referred to as top.myNested)</li> <li>• Top-level inner classes implicitly have access only to static variables.</li> </ul> | <pre>class outer {     int a, b;     static class myInner {         int c, d;         void innerMethod() {....}     }     void outerMethod() {         myInner mi = new myInner();     } } class other {     outer.myInner outInner = new outer.myInner(); }</pre>                                                                                                                                                                                              |
| <p>Member class</p>                                | <ul style="list-style-type: none"> <li>• A nested class cannot define any "static" variables</li> <li>• Scope of the inner class is the entire parent in which it is directly nested. ie, the inner class can reference any members in its parent.</li> <li>• The parent must declare an instance of an inner class before it can invoke the inner class methods, assign to data fields and so on (including private ones)</li> <li>• Unlike nested top-level classes, inner classes are not directly part of a package and are not visible outside the class in which they are nested.</li> </ul>                                                                                                                                                                                                 | <pre>class outer {     int a=5;     class myInner {         int c=a; //access to all members of encls.         void innerMethod(){...}     }     void outerMethod() {         myInner mi1 = new myInner();         myInner mi2 = new myInner();         mi1.c = mi2.c + 30;     } }</pre>                                                                                                                                                                       |
| <p>Local class</p>                                 | <ul style="list-style-type: none"> <li>• This is an inner class declared within a block, typically within a method</li> <li>• It is not a member of the enclosing class.</li> <li>• Their visibility is only within the block of their declaration.</li> <li>• In order for the class to be useful beyond the declaration block, it would need to implement a more publicly available interface.</li> <li>• Because local classes are not members, the modifiers public, protected, private, and static are not usable.</li> <li>• Local classes can access only final variables or parameters.</li> </ul>                                                                                                                                                                                         | <pre>class MyClass {     public static void main(String[] args) {         final String s = "some local results";         class Local {             Local()                 {System.out.println(s);}         }         new Local();     } }</pre>                                                                                                                                                                                                                |
| <p>Anonymous class</p>                             | <ul style="list-style-type: none"> <li>• A variation on a local class. The class is declared and instantiated within a single expression</li> <li>• These classes are simply inner classes that are not given a specific name.</li> <li>• When you don't even need a name because you really just want to pass a method that does something, then you can get away with creating an anonymous inner class.</li> <li>• Typically, a class is not named when it is used only once.</li> <li>• We are declaring an object of an anonymous class type that either implements the interface or extends the class. If it extends the class, then any methods we define may override the corresponding methods of the base class.</li> </ul>                                                              | <pre>public class A extends JApplet {     JButton b = new JButton("click");     public void init() {         b.addActionListener(new ActionListener() {             public void actionPerformed(ActionEvent e)                 {System.out.println("Action Performed");}         });     } } Extending a class : new SuperClassName(arguments) {     // Class body } Implementing an interface new InterfaceName() {     // Implement interface methods }</pre> |

## Garbage Collection

- Circular references does not prevent GC.
  - Finalizer is called only once on the object before being gc, an object can be resurrected once
  - *Object class contains finalize()* method therefore all objects have finalize method
- protected void finalize() throws Throwable{ ... }
- Normal exception handling occurs even in the finalize() method
  - It is not mandatory to call the overridden method finalize() from superclass
  - Finalizers are not implicitly chained like a constructors for subclass
  - Overloading the finalize() method is allowed but only the original method will be called by GC
  - Overridden definitions of finalize method in subclass will not be able to throw checked exception

*Operands must be defined before they are used (no forward reference)*

### Initializer

Instance initializer & Static initializer expressions

- instance variable initialized when object is created
- are executed in the order they are defined
- Initializer expressions cannot pass on **checked exceptions - it must be caught & handled**

### Static Initializer Block

- executed just once when the class is initialized
- usually used to *initialize static variable*, load external libraries for native methods
- *static{;}* is valid block
- a class can have more than one static block
- Static Initializer Block cannot pass on **checked exceptions - it must be caught & handled as no constructor is involved in the class init**

### Instance Initializer blocks

- similar to static blocks but act as constructors during object creation
- used to
  - a> factor out code common to all the constructors of the class e.g. final instance variables, in anonymous class which does not have constructors
  - b> initialize *any final blank variables*
- a class can have more than one instance instance init block
- exception handling is similar as above except that
- if an instance initializer block does **not catch a checked** exception that can occur during its execution then the exception **must be declared in the throws clause of every constructor** in the class
- *Instance initializer in anonymous class can throw any exception*

## Order of initialization

For a class

1. Final
2. Static expr & blocks in order they r specified in class
3. Object creation -> Super class initialization (static variable , constructors)
4. Instance variable & expr in order they r specified in class
5. Local chaining of Constructors of object

Note: The method invoked can *access the state* (default contents) of the object *before* this has been completely initialized (overriding)

For a interface

1. static initializer expressions

## Threads

- Start a thread - start()
  - When extending the Thread class to provide a thread's behavior run() method must be overridden
  - extend Thread class or implement Runnable interface
    - Thread class implements Runnable interface
    - Thread(Runnable threadTarget)
    - Thread(Runnable threadTarget, String threadName) , getName() returns name of the thread
  - All Java objects have a monitor and **each object can be used as a MUTUALLY EXCLUSIVE Lock**
  - Program **terminates when the last non-daemon** thread ends (daemon thread can continue to run in the b/g)
  - Daemon threads run in the background and do not prevent a program from terminating. For example, the garbage collector is a daemon thread
    1. setDaemon(boolean) makes it a daemon thread before the thread is started (else IllegalStateException) and
    2. isDaemon() to check type of thread
  - All threads are spawned from the main thread. The main() method can finish but the program will continue until all user threads are done
- If a parent thread [e.g.main()]creates some child threads and starts, them eventhough the parent thread finishes its work**

### Synchronized Methods

- methods of an object executed by one thread at a time (push , pop)

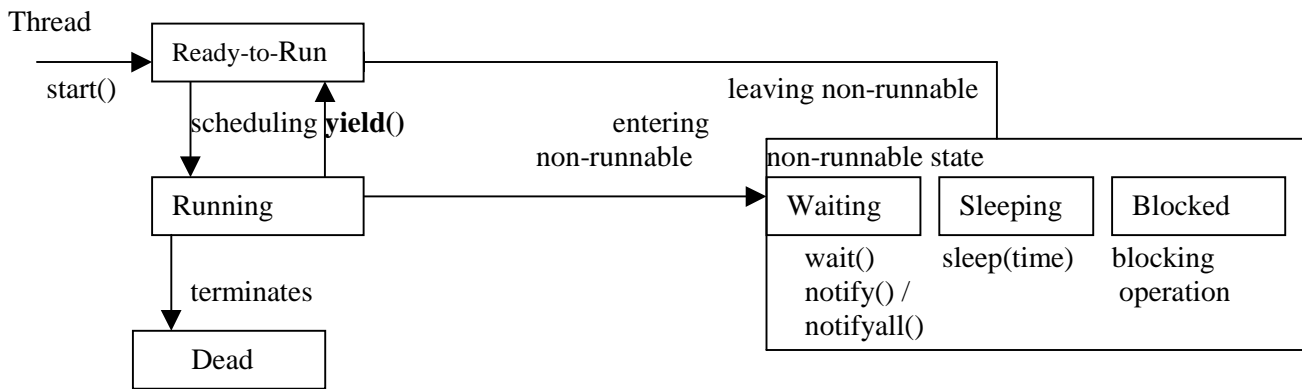
```
public synchronized Object pop(){...}
public synchronized Object push(){...}
```
- Steps
  - thread must enter the **object's monitor** (gain ownership of the monitor - by call to method)
  - any other thread wishing to execute the same method has to wait
  - in the meantime it can execute any other synchronized
- **Non-synchronized method** of the object can be called at **any time** by any thread i.e can run **concurrently with synchronized** methods
- **Synchronized methods can be static** - a thread acquires the class monitor before executing the static synchronized methods
- Synchronization of static methods in a **class is independent from the synchronization of instance** methods on objects of the class
- **subclass can decide whether** the inherited definition of the sync method will remain sync

### Synchronized Blocks

- the sync block allows arbitrary code to be sync on the **monitor of an arbitrary object**

```
synchronized(<Object reference>) { ... }
```
- Once a thread has entered the code block after acquiring the monitor of the specified object, no other thread will be able to execute the code block or another code requiring monitor until the monitor is released by the object
- **Object specification & { } are mandatory**
- Synchronizing on an **inner object and on its associated outer object are independent** of each other unless enforced as in the code
  - special form of **this operator** can be used to sync on the outer object associated with an object of inner class

```
class Outer{
    class Inner{
        synchronized(Outer.this){...}
    }
}
```



- A thread in waiting [Wait()] state **must be notified** by another thread in order to move to R-to-R
- The blocking **operation must complete** before the thread can move to Ready-to-Run state
- When a thread performs an I/O operation, it enters the waiting state It remains in the waiting state until the I/O operation is completed. The thread is said to be **blocked on I/O** or blocking on I/O
- A call to static **yield()** in the Thread class will cause the current running object **to move to R-to-R** state wait for its turn to get the CPU time. The thread scheduler decides which thread gets to run. If there are no threads waiting in the R-to-R state then the thread continues execution else priority decides which thread should be executed - highest priority, equal priority
- A thread once in Dead state cannot be resurrected.
- **JVM also dies when the System.exit or exit method of Runtime is called**
- Thread priority **Thread.MIN\_PRIORITY = 1 to highest Thread.MAX\_PRIORITY = 10 default is Thread.NORM\_PRIORITY = 5**
- inherits priority of parent thread & can be explicitly **set using setPriority() & getPriority to read (in Thread class)**

Thread Scheduler are implementation and platform dependent therefor unpredictable & usually employ

- Preemptive scheduling
- Time-sliced / Round-robin

### Waiting & Notifying

void wait(long timeout) throws InterruptedException // comes out of this waiting when the time

void wait(long timeout,int nanos) throws InterruptedException // elapses then goes directly to  
//Ready State with notify() call

void wait() throws InterruptedException

void notify() //is a method of Object class

void notifyAll()

- Threads are required to **own the object monitor when calling the wait() method.**
- In waiting state the thread release/*relinquishes the monitor/lock* of the object.
- **notify() is a method of Object class**
- wait(), notify() & notifyAll() methods **must be executed in synchronized code**, otherwise the call will result in **IllegalMonitorState Exception**

```
public void pop() {
```

```
    try{
```

```
        wait(); //causes wait to execute in a loop
```

```
//the loop ensures that the condition is always tested after notification moving thread back to wait state
```

```
// incase the condition is not met
```

```
    }catch(Interrupt e){}
```

```
    stakarr[top--] = null
```

```
    notify()...;
```

```
}
```

**A call to notify() has no consequence if there are not thread waiting**

**boolean isAlive()** method is used to find if the thread is alive or dead.

e.g. Parent thread finds if any child threads are alive before terminating itself

isAlive() will return *true* at all states (including suspended) except when the thread is in *new state* or *dead state*

**void join()** throws InterruptedException

A call to this method invoked on a thread will wait & not return until the thread has completed

```
try{
    cA.join();
    if(!cA.isAlive()) println....
}catch(InterruptedException e){ ... }
```

- *interrupt()* method does not stop the thread from executing. The thread which called this interrupt() continues as usual. The thread on which the interrupt() is invoked may/may not respond to this interruption i.e. InterruptedException is thrown if the current thread is interrupted by another thread
- *interrupt()* invokes *InterruptedException*
- Uncaught exceptions for threads are handled by **ThreadGroup.uncaughtException()**

java.lang Package

## Class

The 'Class' | 'ClassLoader' class can be used to load other classes

There is a 'Class' object for each class that is a part of your program

e.g. `Class.forName("Gum")` loads class named Gum

`Gum.class` returns handle to the class (checked at compile time)

`Object o = pets.elementAt[j];`

`petType[I].isInstance (o);` checks if it is instance of the given type of pet

The Class class provides over 30 methods that support the runtime processing of an object's class and interface information. This class does not have a constructor. Objects of this class, referred to as *class descriptors*, are automatically created as classes are loaded by the Java virtual machine. Despite their name, class descriptors are used for interfaces as well as classes.

|                                                    |                                                                                                                                                                                                                                                                                                     |
|----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>getName()</code> and <code>toString()</code> | return the String containing the name of a class or interface <code>toString()</code> method differs in that it prepends the string class or interface, depending on whether the class descriptor is a class or an interface                                                                        |
| <code>static forName()</code>                      | the class specified by a String object and returns a class descriptor for that class.                                                                                                                                                                                                               |
| <code>getSuperclass()</code>                       | method returns the class descriptor of the superclass of a class                                                                                                                                                                                                                                    |
| <code>isInterface()</code>                         | identifies whether a class descriptor applies to a class or an interface                                                                                                                                                                                                                            |
| <code>getInterfaces()</code>                       | method returns an array of Class objects that specify the interfaces of a class, if any                                                                                                                                                                                                             |
| <code>newInstance()</code>                         | method creates an object that is a new instance of the specified class. It can be used in lieu of a class's constructor, although it is generally safer and clearer to use a constructor rather than <code>newInstance()</code><br>useful to create instances of classes not known at compile time. |
| <code>getClassLoader()</code>                      | returns the class loader of a class, if one exists<br>Classes are not usually loaded by a class loader. However, if a class is loaded from outside the CLASSPATH, such as over a network, a class loader is used to convert the class byte stream into a class descriptor                           |

## System class

|                                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>in</code> , <code>out</code> , and <code>err</code> variables<br><code>setIn()</code> , <code>setOut()</code> , and<br><code>setErr()</code>                                                                                                                        | <ul style="list-style-type: none"><li>are, by default, assigned to the standard input, output, and error streams, which are used to support console I/O.</li><li>methods can be used to reassign these variables to other streams.</li></ul> |
| <code>getProperties()</code> /<br><code>getProperty</code>                                                                                                                                                                                                                | gets all the system properties and stores them in an object of class <code>Properties</code>                                                                                                                                                 |
| <code>setProperty()</code> /<br><code>setProperties()</code>                                                                                                                                                                                                              | sets the system properties to the values of a <code>Properties</code> object                                                                                                                                                                 |
| <code>identityClone()</code>                                                                                                                                                                                                                                              | returns the hash code associated with an object.                                                                                                                                                                                             |
| <code>getSecurityManager()</code> and<br><code>setSecurityManager()</code>                                                                                                                                                                                                |                                                                                                                                                                                                                                              |
| <code>exit()</code> ,<br><code>gc()</code> ,<br><code>load()</code> , <code>loadLibrary()</code> ,<br><code>runFinalizersOnExit()</code> , and<br><code>runFinalization()</code><br><code>arraycopy()</code><br><code>currentTimeMillis()</code><br><code>getenv()</code> |                                                                                                                                                                                                                                              |

java.lang Package

## Throwable Class

The Throwable class is at the top of the Java error-and-exception hierarchy. It is extended by the Error and Exception classes and provides methods that are common to both classes.

|                                                        |                                                                                                                               |
|--------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| getMessage()                                           | retrieve any messages that are supplied in the creation of Throwable objects.                                                 |
| fillInStackTrace() and<br>printStackTrace(PrintStream) | supply and print information that is used to trace the propagation of exceptions and errors throughout a program's execution. |

## Error Class

- superclass to define abnormal and fatal events that should not occur. It provides two constructors and no other methods.
- Four major classes of errors extend the Error class: **AWTError**, **LinkageError**, **ThreadDeath**, and **VirtualMachineError**.
- The AWTError class identifies fatal errors that occur in the Abstract Window Toolkit packages. It is a single identifier for all AWT errors and is not subclassed.
- The LinkageError class is used to define errors that occur as the result of incompatibilities between dependent classes. These incompatibilities result when class Y depends on class X, which is changed before class Y can be recompiled. The LinkageError class is extensively subclassed to identify specific manifestations of this type of error.
- The ThreadDeath error class is used to indicate that a thread has been stopped. Instances of this class can be caught and then rethrown to ensure that a thread is gracefully terminated, although this is not recommended. The ThreadDeath class is not subclassed.
- The VirtualMachineError class is used to identify fatal errors occurring in the operation of the Java Virtual Machine. It has four subclasses: **InternalError**, **OutOfMemoryError**, **StackOverflowError**, and **UnknownError**.

## Exception Class

provides a common superclass for the exceptions that can be defined for Java programs and applets.

## Object class

Object is a **root class** of EVERY inheritance heirarchy. **All arrays are genuine objects**

|                                                                        |                                                                                                                                                  |
|------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>int</b> hashCode()                                                  |                                                                                                                                                  |
| <b>Class</b> getClass()                                                |                                                                                                                                                  |
| boolean equals(Object obj)                                             | object value equality (contents are compared not reference)<br>obj.equals(null) is always false                                                  |
| String toString()                                                      | <b>if a subclass does not override</b> this method it returns a textual representation of the object <classname>@<hashcode>                      |
| <i>protected</i> <b>Object clone</b> throws CloneNotSupportedException |                                                                                                                                                  |
| <b>protected</b> void finalize() throws Throwable                      | - does nothing & can be overridden for garbage collections                                                                                       |
| <b>all wait() methods, notify() &amp; notifyAll()</b>                  | are methods defined in Object class. If the current thread is not the owner of the object's monitor an<br>IllegalMonitorStateException is thrown |

## final Wrapper Class

In order to **manipulate primitive values as objects**, java.lang package provides a final wrapper class for each.

- The objects of wrapper class that can be **instantiated are immutable**.
- **Void class is not instantiable**

|                                                   |                                                                                                                                                                                                                                                                   |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Constructors two type</b>                      | This constructor throws <b>NumberFormatException</b> if the parameter is not valid                                                                                                                                                                                |
| takes <i>primitive value</i>                      | Character cObj = new Character('\n');<br>Integer iObj = new Integer(2000);<br>Byte bt = new Byte((byte)16); //cast mandatory                                                                                                                                      |
| takes a <i>string</i>                             | & returns object of the corresponding wrapper class <b>except Character class</b><br>Boolean b = new Boolean("TrUe");//case ignored : true<br>Boolean b1 = new Boolean("XX");//false default value is assigned<br>Integer ix = <b>new Integer</b> ("2001");//2001 |
| <b>valueOf(String s)</b>                          | static method<br>every wrapper class <b>except Character class</b><br>• <b>String to Wrapper Object</b><br>e.g Integer intObj = Integer.valueOf("2010");//converts string to type                                                                                 |
| overrides <b>toString(), equals(), hashCode()</b> | methods. hashCode() return hashcode value based on primitive value of wrapper object<br>e.g. String intStr = intObj.toString(); // "2010"<br>boolean itest = intObj1.equals(intObj2); //false obj value equality                                                  |
| <b>typeValue()</b>                                | • <b>Wrapper objects to primitive value</b><br>e.g. char c = charobj.charValue();<br>intValue(), booleanValue(),doubleValue()                                                                                                                                     |

**Boolean Class** :\_Objects for boolean values : Boolean.TRUE | Boolean.FALSE

**Void Class** is a wrapper class , denotes the Class object representing the primitive type void

**Character Class** According to Unicode value Character.MIN\_VALUE & Character.MAX\_VALUE constants.

static methods

1. **static boolean isTitleCase(char c)**, static boolean **isLowerCase**(char c) , static boolean isUpperCase(char c)
2. **isLetterOrDigit**(char c) , isDigit(char c), isLetter(char c)
3. **static char toTitleCase(char c)**, static char toLowerCase(char c), static char toUpperCase(char c)  
e.g. if(Character.isLowerCase(ch)) ch = Character.toUpperCase(ch);

Abstract Number Class

- wrapper classes **Byte, Short, Integer, Long, Float, Double** are subclass of abstract Numeric class

- constants <wrapperclass>.MIN\_VALUE , <wrapperclass>.MAX\_VALUE

|                                           |                                                                                                                                          |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| static method <b>parseType</b> (String s) | <ul style="list-style-type: none"> <li>• <b>String object argument to primitive numeric value</b></li> </ul>                             |
|                                           | e.g. byte v = Byte.parseByte("16")<br>int v1 = Integer.parseInt("7UP") //NumberFormatException<br>double d1 = Double.parseDouble("3.14") |

final Math class

Constants: Math.E Math.PI

Static Methods:

|                           |                                                                                         |                                                          |
|---------------------------|-----------------------------------------------------------------------------------------|----------------------------------------------------------|
| <b>Random generator</b>   | static double <b>random</b> ()                                                          | <b>between 0.0 to 1.0</b>                                |
| <b>Rounding Functions</b> | static <type> <b>abs</b> (<type> x)<br>overloaded method <b>int, long float, double</b> | e.g. static long abs(long x)<br>long l = Math.abs(2010L) |
|                           | static <type> <b>max</b> (<type> m, <type> n) /<br><b>min</b> (type m,type n)           | e.g. long m =<br>Math.max(1984L,2010L)                   |
|                           | static <b>double ceil</b> (double d)<br>static double <b>floor</b> (double d)           |                                                          |
|                           | static <b>int round</b> (float f)<br>static <b>long round</b> (double d)                |                                                          |
| <b>Exponential</b>        | static double <b>pow</b> (double d1, double d2)                                         | e.g double r = Math.pow(2.0,4.0)                         |
|                           | static double <b>exp</b> (double d)                                                     | e.g. double r = Math.exp(2.0)                            |
|                           | static double <b>log</b> (double d)                                                     | e.g. double r = Math.log(2.0)                            |
|                           | static double <b>sqrt</b> (double d)                                                    | e.g. double r = Math.sqrt(2.0)                           |
| <b>Trigonometry</b>       | static double <b>sin</b> (double d)                                                     | //d - is angle in radians                                |
|                           | static double <b>cos</b> (double d)                                                     |                                                          |
|                           | static double <b>tan</b> (double d)                                                     |                                                          |
|                           | static double <b>asin</b> (double d)                                                    |                                                          |

String class

- **final String class - implements immutable character strings Read-Only once created & initialized**
- **StringBuffer class - dynamic character strings**
- **both are thread-safe**

final String class

|                  |                                                                                                                                                                   |                                                                                                                                                                 |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Constructors     | Only one anonymous String object is shared by all string literals with the same content                                                                           | String str1 = "Hello"<br>String str2 = "Hello" //both denote the same anonymous String object (reference same)                                                  |
|                  | creates a new String object <b>String(String s)</b>                                                                                                               | String str3 = new String("Hello");<br>//new <b>String()</b> is an empty string                                                                                  |
|                  | can also be done from <b>arrays of bytes, characters or StringBuffers</b>                                                                                         | byte[] b = {97, 98, 98, 97} ;<br>String s = new String(b); // stores "abba"<br>StringBuffer strBuf = new StringBuffer("axe");<br>String s = new String(strBuf); |
| Reading methods  | int <b>length</b> ()                                                                                                                                              |                                                                                                                                                                 |
|                  | <b>char charAt</b> (int index)<br>first character is at index 0 & last at index -1.<br>if index is not valid<br><i>StringIndexOutOfBoundsException</i> is thrown. | str.charAt(j)                                                                                                                                                   |
| Searching String | int <b>indexOf</b> (int ch)                                                                                                                                       | <b>//first occurrence of character from beginning</b>                                                                                                           |
|                  | int <b>indexOf</b> (int ch,int fromIndex)                                                                                                                         |                                                                                                                                                                 |

|                                            |                                                                                                                                                                                                                                                                              |                                                                                                                                                                          |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                            | int indexOf(String s)                                                                                                                                                                                                                                                        | //first occurrence of string                                                                                                                                             |
|                                            | int indexOf(String s,int fromIndex)                                                                                                                                                                                                                                          |                                                                                                                                                                          |
|                                            | int <b>lastIndexOf</b> (int ch)                                                                                                                                                                                                                                              | & other overloaded methods //first occurrence from end of string                                                                                                         |
|                                            | ▪ String <b>replace</b> (char old,char new)                                                                                                                                                                                                                                  |                                                                                                                                                                          |
|                                            | e.g. String s = "Java Jives"      String p = "One man, one vote"<br>String x = s.replace('J','W')      // x = "Wava Wives"<br>int t1 = s.indexOf('J')              // 0<br>int t2 = s.lastIndexOf('J')         //5<br>int t3 = p.lastIndexOf("One",8)     //0                |                                                                                                                                                                          |
| Comparing methods                          | boolean test = 'a' < 'b'                                                                                                                                                                                                                                                     | //unicode are tested i.e.true as 0x61 < 0x62                                                                                                                             |
|                                            | boolean equals(Object o)                                                                                                                                                                                                                                                     |                                                                                                                                                                          |
|                                            | boolean <b>equalsIgnoreCase</b> (String s)                                                                                                                                                                                                                                   |                                                                                                                                                                          |
|                                            | int compareTo(Object o)                                                                                                                                                                                                                                                      | //object is a String object else <b>ClassCastException</b>                                                                                                               |
|                                            | int compareTo(String s)<br>0 strings are equals<br>> 0 string1 is lexicographically greater than the argument<br>< 0 string1 is lexicographically less than the argument                                                                                                     | //string literal<br>String str1 = new String("abba");<br>String str2 = new String("aha");<br>int compval = str1.compareTo(str2)<br>//negative str1<str2                  |
| Case conversion                            | toUpperCase(), toLowerCase()                                                                                                                                                                                                                                                 | if <b>no change same object is returned</b> else a new object is created                                                                                                 |
|                                            | locale = Locale.getDefault()                                                                                                                                                                                                                                                 | (Locale locale) denotes to specific geographical region                                                                                                                  |
| Concatenation                              | String <b>concat</b> (String s)                                                                                                                                                                                                                                              |                                                                                                                                                                          |
| Extracting substring                       | String <b>trim</b> ()                                                                                                                                                                                                                                                        | //remove leading & trailing spaces)                                                                                                                                      |
|                                            | String <b>substring</b> (int startindex)                                                                                                                                                                                                                                     |                                                                                                                                                                          |
|                                            | String substring( <b>int startindex,int endindex</b> )                                                                                                                                                                                                                       | e.g. String s = "kakapo"<br>s = s.substring(2,5)                 //return kap                                                                                            |
| Convert objects & primitive type to String | ▪ static String <b>valueOf</b> (Object obj)<br>▪ static String valueOf( <b>char[] character</b> )<br>▪ static String valueOf( <b>&lt;type&gt; x</b> )<br>where type boolean, char, int, float, long, double                                                                  | same as obj.toString()<br>String astr = String.valueOf("make a str");<br>String cstr = String.valueOf(new char[]{'a','b','c'});<br>String sstr = String.valueOf(Math.PI) |
| Miscellaneous methods                      | toCharArray()                         //string characters into an array of characters<br>getBytes()                            //string to array of bytes<br>startsWith()                          //prefix<br>endsWith()                             //suffix<br>hashCode() |                                                                                                                                                                          |

## final StringBuffer Class

characters & the capacity of the buffer can be changed dynamically

|                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Constructors                  | <ul style="list-style-type: none"> <li>▪ <b>StringBuffer(String s)</b> //capacity = length of string + 16</li> <li>▪ <b>StringBuffer(int capacity)</b> //capacity &gt; 0</li> <li>▪ <b>StringBuffer()</b> //capacity = 16 characters</li> </ul>                                                                                                                                                   |
| Changing & Reading Characters | <ul style="list-style-type: none"> <li>▪ int length()</li> <li>▪ char charAt(int index)</li> <li>▪ void setCharAt(int index, char ch)</li> </ul>                                                                                                                                                                                                                                                  |
| Manipulating StringBuffer     |                                                                                                                                                                                                                                                                                                                                                                                                   |
| Append                        | <ul style="list-style-type: none"> <li>▪ StringBuffer append(Object obj)</li> <li>▪ StringBuffer append(String s)</li> <li>▪ StringBuffer append(char c) // similarly boolean, int, long, float, double converts</li> <li>▪ // primitive directly applying String.valueOf()</li> <li>▪ StringBuffer append(char[] str)</li> <li>▪ StringBuffer append(char[] str, int offset, int len)</li> </ul> |
| Insert                        | <ul style="list-style-type: none"> <li>▪ StringBuffer insert(int offset, Object obj)</li> <li>▪ StringBuffer insert(int offset, String s)</li> <li>▪ StringBuffer insert(int offset, char[] str)</li> <li>▪ StringBuffer insert(int offset, char c) // similarly boolean, int, long, float, double</li> </ul>                                                                                     |
| Delete                        | <ul style="list-style-type: none"> <li>▪ StringBuffer deleteCharAt(int index) //delete a character</li> <li>▪ StringBuffer delete(int start, int end) //deletes a substring</li> </ul>                                                                                                                                                                                                            |
| Reverse                       | <p>StringBuffer reverse()<br/> Concatenation in String is implemented using string buffers<br/> e.g. String s = "4" + "U" + "only" is same as<br/> String s = new StringBuffer().append(4).append("U").append("Only").toString();</p>                                                                                                                                                             |
| Controlling Capacity          | <ul style="list-style-type: none"> <li>▪ <b>int capacity()</b> //nos of char buffer can occupy</li> <li>▪ <b>void ensureCapacity(int minCapacity)</b></li> <li>▪ <b>void setLength(int newLength)</b> //newLength &gt;= 0</li> </ul> <p>to compact a string buffer &amp; remove any additional capacity<br/> buffer.setLength(buffer.length())</p>                                                |

String == StringBuffer //error illegal expression as neither of them is a superclass of the other

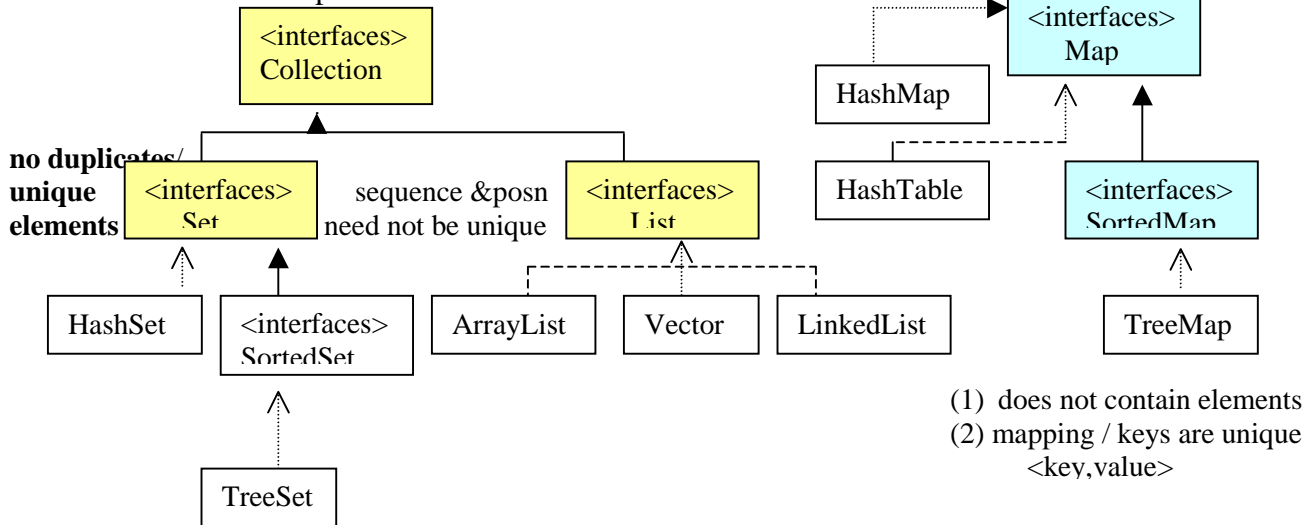
e.g. String s = "hello"

StringBuffer sb = new StringBuffer("hello")

if(s == sb) print "A" //error

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                         |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>public class trial { public void method1(StringBuffer s1, StringBuffer s2){     s1.append("There");     s2 = s1; //create a new s2 sb2 is unchanged } public static void main(String[] args){ StringBuffer sb1 = new StringBuffer("Hello"); StringBuffer sb2 = new StringBuffer("Hello"); trial sbt = new trial(); <b>sbt.method1(sb1, sb2);</b> System.out.println("sb1 is " + sb1 + "\nsb2 is " + sb2); } } Output: sb1 is HelloThere sb2 is Hello</pre> | <pre>public class trial { public static void main(String[] args){ StringBuffer sb1 = new StringBuffer("Hello"); StringBuffer sb2 = new StringBuffer("Hello"); sb1.append("There"); sb2 = sb1; System.out.println("sb1 is "+sb1+"\nsb2 is " +sb2); } } Output: sb1 is HelloThere sb2 is HelloThere</pre> |
| <pre>String s1 = "Hello1"; int v = 6; System.out.println(s1+8); System.out.println(s1+v);</pre>                                                                                                                                                                                                                                                                                                                                                                 | <pre>StringBuffer sb1 = new StringBuffer("Hello2"); //System.out.println(sb1+v); //COMPILE Error operator + cannot <b>be applied to java.lang.StringBuffer,int</b> System.out.println(sb1.append(v));</pre>                                                                                             |

## Collections & Maps



| Interfaces      | Implementation (java.util)                                                                            |
|-----------------|-------------------------------------------------------------------------------------------------------|
| Collection      | Set                                                                                                   |
| Set → SortedSet | <ul style="list-style-type: none"> <li>▪ Hash Set</li> <li>▪ TreeSet</li> </ul>                       |
| List            | List                                                                                                  |
| Maps            | <ul style="list-style-type: none"> <li>▪ ArrayList</li> <li>▪ Vector</li> <li>▪ LinkedList</li> </ul> |
| SortedMap       | Map                                                                                                   |
|                 | <ul style="list-style-type: none"> <li>▪ HashTable, HashMap</li> <li>▪ TreeMap</li> </ul>             |

- no direct implementation of Collection interface
- allows data to be passed from one collection to another [same for map]
- Collections & Maps are not interchangeable

java.util.Collections has static methods

- static int **binarySearch**(List l, Object key)
- static void **fill**(List l, Object o) e.g. Arrays class : char[] bar= new char[5]; Arrays.fill(bar, '\*') //fill 5 \*
- static void **shuffle**(List l)
- static void **sort**(List l) e.g. Collections.sort(keys1)

### Decorators for thread-safety & data immutability

Decorators for Thread-safety

**Collection** synchronizedCollection(Collection c)

**Set** synchronizedSet(Set s)

**List** synchronizedList(List l)

**Map** synchronizedMap(Map m)

**SortedSet** synchronizedSortedSet(SortedSet s)

**SortedMap** synchronizedSortedMap(SortedMap m)

e.g

```
Collection syncDecorator = Collections.synchronizedSortedSet(someCollection)
```

```
synchronized(syncDecorator){
    for(Iterator it=syncDecorator.iterator();it.hasNext();)
        doSomething(it.next);
}
```

Decorator for Unmodifiable /Read-only access

**Collection unmodifiableCollection(Collection c)** similarly for Set,List,Map,SortedSet,SortedMap

Create a immutable list: **List nCopies(int n, Object o)**

Single Set list: Collections.singleton(Object o)

Collections

Constants: EMPTY\_SET, EMPTY\_LIST

Some operations are optional (UnsupportedOperationException is thrown). Methods common to Sets & Lists

| Basic operations                                                                        | Bulk operations                                                                            |
|-----------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| int size                                                                                | void clear() <span style="float:right">empty</span>                                        |
| boolean isEmpty()                                                                       |                                                                                            |
| boolean contains(Object element) - <b>no change</b>                                     | boolean containsAll(Collection c) → b subset of a                                          |
| boolean add(Object element) //optional<br>append at end - changes collection            | boolean addAll(Collection c) → a U b                                                       |
| boolean remove(Object element) //optional<br>delete 1st occurrence - changes collection | boolean removeAll(Collection c) → a - b<br>boolean retainAll(Collection c) → a intersect b |

|                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                            |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| Bridge between Arrays & Colletion<br><ul style="list-style-type: none"> <li>▪ Arrays class has <b>asList()</b> to createlist view of arrays</li> </ul> Collection class Conversion to Arrays<br><ul style="list-style-type: none"> <li>▪ Object <b>toArray()</b></li> </ul> Object toArray(Object a[]) <span style="float:right">type of array can be specified</span> | <pre>interface Iterator(){     boolean hasNext();     Object next();     void remove(); //optional }</pre> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|

HashSet

Constructors

- HashSet()
- HashSet(Collection c)
- HashSet(int initialCapacity) //nos of buckets in hash table
- HashSet(int initialCapacity, float loadFactor) // (nos elements/total capacity)

List

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                                                   |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| Reading | <b>Object get(int index)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | range 0 to size()- 1                                                                              |
| Replace | <b>Object set(int index, Object element)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | //optional - replace                                                                              |
| Insert  | void add(int index, Object element)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | //optional -insert                                                                                |
|         | void addAll(int index, Collection c)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | //optional                                                                                        |
| Delete  | <b>Object remove(int index)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | //optional -delete                                                                                |
| Search  | <b>int indexOf(Object o)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | //if found the 1st & last occurrence else -1                                                      |
|         | <b>int lastIndexOf(Object o)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                                                                   |
|         | <b>List subList(int fromIndex, int toIndex)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | //view can be manipulated & will reflect in the actual list<br>//from - inclusive, to - exclusive |
| Looping | <b>ListIterator listIterator()</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | //start from 1st                                                                                  |
|         | <b>ListIterator listIterator(int index)</b><br>interface ListIterator extends Iterator{<br>boolean hasNext();<br>boolean hasPrevious();<br><br>Object next(); <span style="float:right">//element after the cursor</span><br>Object previous();<br><br>int nextIndex();<br>int previousIndex();<br>void remove(); <span style="float:right">//optional</span><br>void set(Object o); <span style="float:right">//optional</span><br>void add(Object o); <span style="float:right">//optional</span><br>}<br>//start from the index position |                                                                                                   |

ArrayList, Vector, LinkedList

- ArrayList primary implementation of the List interface as it has better performance
- All list have constructor to create empty list
- ArrayList & Vector have additional constructor with an existing collection
- Vector is Thread-Safe meaning any concurrent calls to the vector will not compromise its integrity i.e. uses synchronization

Note: All methods defined in Set are also defined in Collections

List defines additional methods

Vector, BitSet, Stack, HashTable **always STORE elements as OBJECTS** hence explicit cast required when retrieving elements. Therefore primitives cannot be added directly. They must be converted to object

Vector methods - all methods are synchronized

- addElement( Object),
- removeElement(Object),
- elementAt(i),
- size()

HashTable does not have any final methods

Array Class

- equals()
- sort()
- fill()
- binarySearch()
- asList() - converts to a list

java.util.Collection is the root interface

java.util.**Collections**

- is not an interface or abstract base class
- All methods are static

**Stack Class (LIFO)** extends the Vector class

a single default constructor, Stack(), that is used to create an empty stack.

|                                  |                                                                                               |
|----------------------------------|-----------------------------------------------------------------------------------------------|
| push()                           | placed on the stack using the method                                                          |
| pop() throw EmptyStackException  | retrieved from the stack                                                                      |
| peek() throw EmptyStackException | returns the top element of the stack without popping it off                                   |
| search()                         | enables you to search through a stack to see if a particular object is contained on the stack |
| empty()                          | to determine whether a stack is empty                                                         |

### BitSet Class (On/Off)

- The BitSet class is used to represent and manipulate a set of bits. Each individual bit is represented by a boolean value, and can be indexed much like an array or Vector. A bit set is a growable set, the capacity of which is increased as needed. The bits of a bit set are sometimes referred to as *flags*.
- Two BitSet constructors are provided. One enables the initial capacity of a BitSet object to be specified. The other is a default constructor that initializes a BitSet to a default size.
- The BitSet access methods provide and, or, and exclusive or logical operations on bit sets, enable specific bits to be set and cleared, and override general methods declared for the Object class.
- minimum size is 64 bits
- set(i), clear(i)

**Linked List** - addFirst(),addLast(),getFirst(),getLast(),removeFirst(),removeLast()

### Properties Class

- *Properties* are extensions of the Dictionary and Hashtable classes and are defined in the java.util package
- The Properties class is a subclass of Hashtable that can be read from or written to a stream.
- It also has a 2nd HashTable to hold default properties.
- It also provides the capability to specify a set of default values to be used if a specified key is not found in the table.
- Properties supports two constructors: a default constructor with no parameters and a constructor that accepts the default properties to be associated with the Properties object being constructed.
- The Properties class declares several new access methods.

|                 |                                                                                                                                                                                                                        |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| getProperty()   | enables a property to be retrieved using a String object as a key. A second overloaded getProperty() method allows a value string to be used as the default, in case the key is not contained in the Properties object |
| load()          | are used to load a Properties object from an input stream                                                                                                                                                              |
| save()          | save it to an output stream. The save() method enables an optional header comment to be saved at the beginning of the saved object's position in the output stream.                                                    |
| propertyNames() | provides an enumeration of all the property keys                                                                                                                                                                       |
| list()          | provides a convenient way to print a Properties object on a PrintStream object                                                                                                                                         |

### Observer Interface and Observable Class

- used to implement an abstract system by which observable objects can be observed by objects that implement the Observer interface.
- These objects maintain a list of observers. When an observable object is updated, it invokes the update() method of its observers to notify them that it has changed state.
- The *update() method* is the *only method* that is specified in the *Observer interface*. The method takes the Observable object and a second notification message Object as its parameters.
- The Observable class is an **abstract class** that must be subclassed by Observable objects. It provides several methods for adding, deleting, and notifying observers and for manipulating change status.
- Not a part of Collections API
- Not related to GUI components

## Map

| Basic operation                                 | Bulk Operation                  |
|-------------------------------------------------|---------------------------------|
| int size()                                      |                                 |
| boolean isEmpty()                               |                                 |
| Object get(Object key)                          |                                 |
| Object put(Object key, Object value) //optional | Object putAll(Map t) //optional |
| Object remove(Object key) //optional            | void clear() //optional         |
| boolean <b>containsKey</b> (Object key)         |                                 |
| boolean <b>containsValue</b> (Object value)     |                                 |

### Views

- **Set keySet()**
- **Collection value() //several keys can map to same value/duplicate value**
- **Set entrySet**

Map.Entry interface

```
interface Entry{
    Object getKey();
    Object getValue();
    Object setValue();
}
```

**Map subMap(int fromKey, int toKey)**

### HashMap and HashTable

- HashMap provide primary implementation
- HashTable is threadSafe
- Constructors : empty, existing , (capacity,load factor)

Implement Comparable interface **[natural order]**

**int compareTo**(Object o1) //implemented by String, Date & File class

Implement Comparator interface **[total order]**

**int compare**(Object o1, Object o2) // 0, >0, <0 works like compareTo of string class

| SortedSet                                              | SortedMap                                      |
|--------------------------------------------------------|------------------------------------------------|
| SortedSet headSet(Object toElement) //less than        | SortedMap headMap(Object toKey)                |
| SortedSet tailSet(Object fromElement) //greater or =   | SortedMap tailMap(Object fromKey)              |
| SortedSet subSet(Object fromElement, Object toElement) | SortedMap subMap(Object fromKey, Object toKey) |
| Object first()                                         | Object firstKey()                              |
| Object last()                                          | Object lastKey()                               |
| Comparator compare()                                   | Comparator compare()                           |

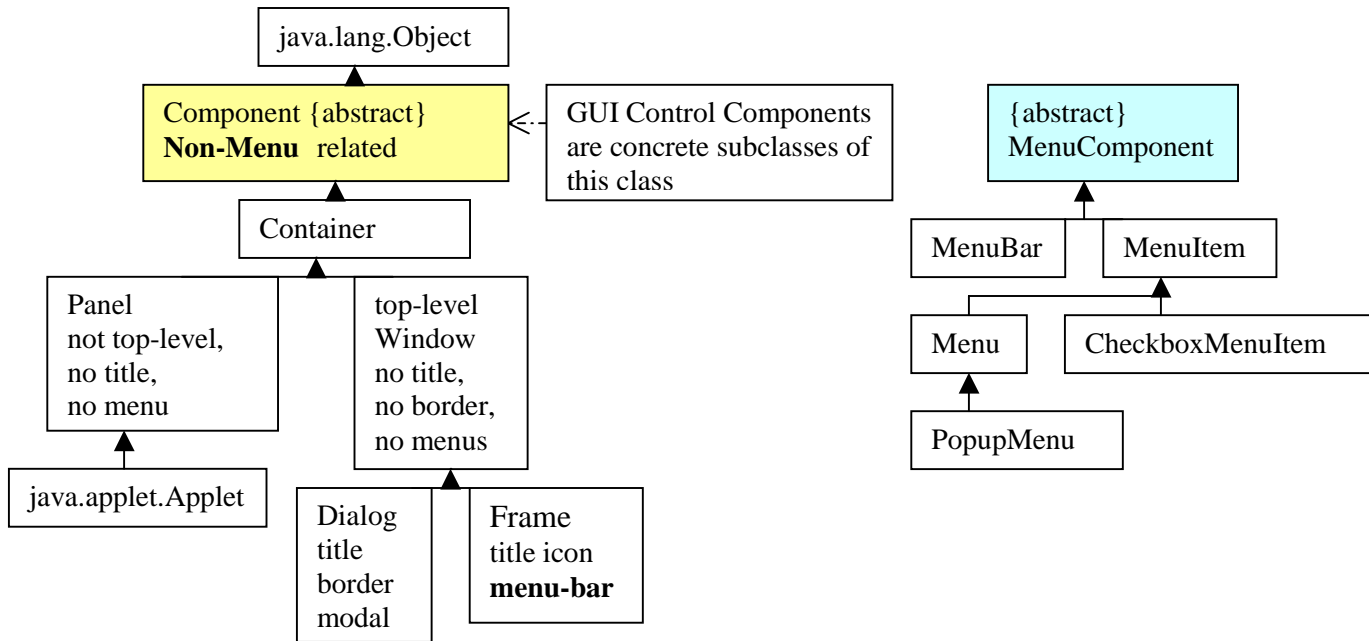
| TreeSet               | TreeMap               |
|-----------------------|-----------------------|
| TreeSet()             | TreeMap()             |
| TreeSet(Comparator c) | TreeMap(Comparator c) |
| TreeSet(Collection c) | TreeMap(Map m)        |
| TreeSet(SortedSet s)  | TreeMap(SortedMap m)  |

# GUI

JFC has 2 frameworks AWT & SWING

AWT - dependent on underlying windowing system

Swing - light-weight that is not dependent on the underlying windowing system, pluggable look & feel



## Component

|                                                                                                                       |                                                                                                  |                                                                                                                                |
|-----------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Size<br>Dimension getSize()<br>void setSize(int width,int height)<br>void setSize(Dimension d)                        | Coordinates<br>Point getLocation()<br>void setLocation(int x,int y)<br>void setLocation(Point p) | Bounds<br>Rectangle getBounds()<br>void setBounds(int x, int y, int width,int height)<br>void setBounds(Rectangle r)           |
| Color<br>void setForeground(Color c)<br>void setBackground(Color c)<br>Color getBackground()<br>Color getForeground() | Font<br>void setFont(Font f)<br>Font getFont()                                                   | void setEnabled(boolean b)<br>void setVisible(boolean b)<br>visible default is T for all components except Window,Frame&Dialog |
| add(PopupMenu popup)                                                                                                  |                                                                                                  |                                                                                                                                |

### Top-Level window cannot be incorporated into other components

Window Class

- void pack() //initiates layout management - window sizing
- void show()
- void dispose() //free window resources but does not delete the window

|                                                                                                             |                                                                                                                                                                                                                        |
|-------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Frame Class<br>Constructors<br>Frame()<br>Frame(String title)<br><br>methods<br>void setMenuBar(MenuBar mb) | Dialog Class (non-modal by default)<br>Constructors<br>Dialog( <b>Frame parent</b> )<br>Dialog(Frame parent, boolean modal)<br>Dialog(Frame parent, String title)<br>Dialog(Frame parent, String title, boolean modal) |
|-------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

GUI Components

1. Button
2. Canvas

3. Checkbox (an be used as radio buttons with CheckboxGroup). **CheckboxGroup is not a subclass of Component** & does not have graphical representation
4. Choice (pop up /drop-down menu)
5. Label
6. List (scrollable list - single or multiselect)
7. Scrollbar
8. TextComponent (does not provide any public constructors therefore not instantiable), 2 subclasses
  - TextField
  - TextArea

| GUI Component            | Constructor                                                                                                                                                                                                                | Methods                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Button                   | Button()<br>Button(String label)                                                                                                                                                                                           | String getLabel()<br>void setLabel(String label)                                                                                                                                                                                                                                                                                                                                                                     |
| Canvas                   | no default graphical representation<br>is subclassed to drawings, images<br>e.g.<br>class DrawRectRegion extends Canvas<br>{...}                                                                                           | the paint() method is usually overridden in subclass<br>public void paint(Graphic g){<br>g.drawRect(0,0,150,150)<br>}                                                                                                                                                                                                                                                                                                |
| Checkbox & CheckboxGroup | Checkbox()<br>Checkbox(String label)<br>Checkbox(String label, boolean state)<br>Checkbox(String label, boolean state, CheckboxGoup group)<br>Checkbox(String label, CheckboxGoup group, boolean state)<br>CheckboxGroup() | <ul style="list-style-type: none"> <li>▪ <b>boolean getState()</b></li> <li>void setState(boolean b)</li> <li><b>//default state is unchecked</b></li> <li>▪ String getLabel()</li> <li>void setLabel(String label)</li> <li>▪ Checkbox getSelectedCheckbox()</li> <li>void setSelectedCheckbox(Checkbox box)</li> <li>▪ CheckboxGroup getCheckboxGroup()</li> <li>void setCheckboxGroup(CheckboxGroup g)</li> </ul> |
| Choice                   | Choice()<br><br>e.g. Choice c = new Choice()<br>c.add("One"); //add choices to compon<br>c.add("Two");<br>add(c); //add comp to container<br>c.select("Two");                                                              | void add(String item)<br><br>int getItemCount()<br>String getItem(int index)<br>String getSelectedItem()<br>int getSelectedIndex()<br>void select(int pos)<br>void select(String str)                                                                                                                                                                                                                                |
| Label                    | Label()<br>Label(String text)<br>Label(String text,int alignment)<br>where integer constants LEFT, RIGHT, <b>CENTER default is left</b>                                                                                    | String getText()<br>void setText(String text)<br>int getAlignment()<br>void setAlignment(int alignment)                                                                                                                                                                                                                                                                                                              |
| List                     | List()<br>List(int rows)<br>List(int rows,int multipleMode)<br><br>e.g. String[] fruits = {"Mango","Apple"};<br>List fList = new List(fruit.length-1,true)<br>for(j=0...)<br>fList.add(fruit[j]);                          | void add(String item)<br>void add(String item,int index)<br><br>int getItemCount()<br>String getSelectedItem()<br>int getSelectedIndex()<br>void select(int pos)<br>void select(String str)<br><br>void getRows()<br><b>boolean isMultipleMode()</b><br>String[] getItems();<br>String[] getSelectedItems();                                                                                                         |

|           |                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |                                                                                                                                                                                                                                                                                  | void deselect(int index)                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Scrollbar | Scrollbar()<br>Scrollbar(int orientation)<br><b>Scrollbar(int orientation, int value, int visibleAmount, int min, int maximum)</b><br><br>HORIZONTAL, VERTICAL<br><b>vertical scrollbar is default</b>                                                                           | <ul style="list-style-type: none"> <li>▪ int getValue()<br/>void setValue(int value)</li> <li>▪ int getMinimum()<br/>void setMinimum(int newMin)</li> <li>▪ int getMaximum()<br/>void setMaximum(int newMax)</li> <br/> <li>▪ int <b>getVisibleAmount()</b> //scrollbar width<br/>void setVisibleAmount(int newAmount)</li> <li>▪ int <b>getUnitIncrement()</b><br/>void setUnitIncrement(int v)</li> <li>▪ <b>int getBlockIncrement()</b><br/>void setBlockIncrement(int b)</li> </ul> |
| TextField | TextField()<br>TextField(String text)<br>TextField(int columns) //col width<br>TextField(String text, int columns)                                                                                                                                                               | String getText()<br>void setText(String s)<br><br><b>int getColumnns()</b> //width of textfield/textare<br>void setColumnns(int columnns)                                                                                                                                                                                                                                                                                                                                               |
| TextArea  | TextArea()<br>TextArea(String text)<br>TextArea(int rows, int columns)<br>TextArea(String text, int rows, int columns)<br>TextArea(String text, int rows, int columns, int scrollbars)<br><br>SCROLLBARS_BOTH<br>SCROLLBARS_HORIZONTAL<br>SCROLLBARS_VERTICAL<br>SCROLLBARS_NONE | String getSelectedText()<br><br><b>boolean isEditable()</b><br>void setEditable(boolean b)<br><br>nos of columns is a measure of the size of the text line according to the particular font used for rendering the text                                                                                                                                                                                                                                                                 |

```
e.g. CheckboxGroup cbgroup1 = new CheckboxGroup(), cbgroup2 = new CheckboxGroup();
    CheckboxGroup[] groups = {cbgroup1,cbgroup1,cbgroup2,cbgroup2,null};
    setLayout(new FlowLayout());
    for(int i = 0;i<groups.length;i++)
        add(new Checkbox("box: "+i,true,groups[i]));
    pack();
```

Output shows box1, box3 & box4 selected

Note here: All checkboxes are selected when they are created. But within a checkboxgroup only one can be selected Therefore

between box 0 & box 1 of group1- most recent one box1 is selected

between box 2 & box 3 of group2- most recent one box3 is selected

box4 is regular checkbox - box4 is selected

### abstract MenuComponent class

- `getFont()` and `setFont()`—Gets and sets the font associated with the menu component.
- `getName()` and `setName()`—Gets and sets the name of the menu component.
- `getParent()`—Returns the parent container of the menu component
- The `MenuComponent` class has two direct superclasses, `MenuBar` and `MenuItem`, which provide most of the methods for creating and using menus

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>MenuBar class</b> <ul style="list-style-type: none"><li>▪ displayed at the top of an application window by a <code>Frame</code> object</li><li>▪ assigned a set of <code>Menu</code> objects</li><li>▪ <code>add()/remove()</code> methods adds or removes menus to menu-bar</li><li>▪ <b>A <code>Frame</code> object can have one and only one <code>MenuBar</code> object</b>, which is set using the <code>setMenuBar()</code> method of the <code>Frame</code> class</li><li>▪ The <code>MenuBar</code> class lets a special menu be designated as a Help menu. It is set using the <code>setHelpMenu()</code> method.</li><li>▪ The <code>getMenu()</code> and <code>getHelpMenu()</code> methods are used to retrieve the <code>Menu</code> objects that have been added to a <code>MenuBar</code>.</li></ul> | <b>MenuItem class</b><br>define menu item with a textual label<br><b>Menu Class</b> <ul style="list-style-type: none"><li>▪ implements a pull-down menu</li><li>▪ nested to <b>create submenus</b></li><li>▪ <code>add()</code>, <b><code>addSeparator()</code></b>,<code>insertSeparator()</code> method</li></ul> <b>Menu instance</b> <ol style="list-style-type: none"><li>1. <code>Instances of MenuBar</code></li><li>2. <code>MenuItem</code></li><li>3. <code>CheckboxMenuItem</code></li></ol> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### PopupMenu Class

- `PopupMenu` is not a subclass of `Component` but instances can be added the panel as pop-up menu associated with the component. (see `add` method in component)
- **Popup menu cannot be contained in a menu-bar**

### FileDialog

- The `FileDialog` class is used to construct dialog boxes that support the selection of files for input and output operations.
- It is a subclass of the `Dialog` class and provides three constructors.
- These constructors take as arguments the `Frame` window that contains the dialog box, the title to be used at the top of the dialog box, and a mode parameter that can be set to the `LOAD` or `SAVE` constants defined by `FileDialog`.
- `FileDialog` provides methods that are used to access the directory `getDirectory()` and filename `getFile()` of the user-selected file and to specify an object that implements the `FileNameFilter` interface.
- The `FileNameFilter` interface is defined in the `java.io` package. It defines the `accept()` method, which is used to determine the filenames that should be included in a file list.

### ScrollPane

- The `ScrollPane` class simplifies the development of scrollable applications. The `ScrollPane` class is like a combination of a panel and vertical and horizontal scrollbars.
- The great thing about it is that it performs all the scrollbar event handling and screen redrawing internally & hence is significantly faster.
- The `ScrollPane` class extends the `Container` class and, therefore, can contain other components.
- It is designed to automate scrolling for a single, contained component, such as a `Canvas` object.
- It provides two constructors—a single parameterless constructor and a constructor that takes an `int` argument. The parameterless constructor creates a `ScrollPane` object that displays scrollbars only when they are needed.
- The other constructor takes one of the three constants: **SCROLLBARS ALWAYS**, **SCROLLBARS AS NEEDED**, **and SCROLLBARS NEVER**. These constants determine if and when scrollbars are displayed by the `ScrollPane` object.
- The initial size of the `ScrollPane` object is 100×100 pixels. The `setSize()` method can be used to resize it.
- The `ScrollPane` class provides methods for accessing and updating its internal scrollbars, but in most cases this is both unnecessary and ill-advised. Other methods are provided to get and set the current scrollbar positions.

## Layout Manager

**Absolute Positioning and Sizing:** The AWT will also let you use absolute positioning and sizing. This is accomplished through a **null layout manager**.

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FlowLayout    | direction - left-right<br>maintain preferred size of component(comp <i>size never</i> change evenif cont resize)<br>default for Panel - Applet                                                                                                                                                                                                                                                                                        |
| BorderLayout  | 5 components<br>directions - north, south, east, west, center (not all regions need to be occupied)<br>comp can be explicitly added to a region using constraint<br>if North   South - prefer to honor <i>preferred height</i> but width is stretched/changed<br>if East   West - prefer to honor <i>preferred width</i> but height is stretched/changed<br>Center takes up whatever space is left<br>default for Window-Frame-Dialog |
| GridLayout    | rectangular grid(rows,col) all cell are of same size<br>one comp in each cell , <i>comp resized</i> to fit in the cell<br>direction - top-to bottom & left to right                                                                                                                                                                                                                                                                   |
| CardLayout    | direction - stack of indexed cards<br>only top component visible                                                                                                                                                                                                                                                                                                                                                                      |
| GridBagLayout | rectangular grid but flexible layout<br>a component can occupy multiple cells of grid but region it occupies is always rect<br>comp can be of different sizes                                                                                                                                                                                                                                                                         |

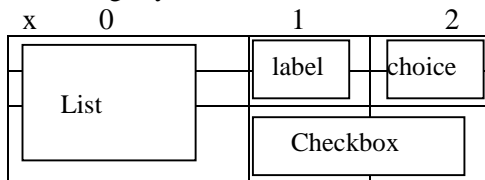
- LayoutManager getLayout()/ void setLayout(LayoutManager mgr) of the Container Class
- The LayoutManager2 interface extends the LayoutManager interface to deal with constraint-based layouts
- container calls the appropriate layout manager
- The preferred size of a component is returned by its getPreferredSize() method
- Size of checkbox is unaffected e
- Panel & Applet are containers that must be attached to a parent container
- Top-level (Window, Frame & Dialog) containers that are independent & cannot be put in another container

|                                                                                                                                                                                                                                            |                                                                        |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------|
| Component add(Component c)<br>Component add(Component c,int index)<br>Component add(Component c,Object constraints)<br>Component add(Component c,Object constraints,int index)<br>index = -1 is default placement placing component at end | void remove(int index)<br>void remove(Component c)<br>void removeAll() |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------|

- validate(), invalidate() and doLayout() methods can be used to cause a container to be laid out again.
- **validate()** method is used by the AWT to cause a container to lay out its subcomponents after the components it contains have been added to or modified.
- **invalidate()** method causes a component and all of its containers to be marked as needing to be laid out.
- **doLayout()** method is used to tell a layout manager to layout a component.

|              | Constructors                                                                                                                                                                                                                                      |                                                                                                                                                                          |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Flow Layout  | FlowLayout()<br>FlowLayout(int alignment)<br>FlowLayout(int alignment, int horizontalgap, int verticalgap)<br>LEFT,CENTER(default)RIGHT<br>default gap = 5 pixel                                                                                  |                                                                                                                                                                          |
| BorderLayout | BorderLayout()<br>BorderLayout(int horizontalgap, int verticalgap)<br>NORTH,SOUTH,EAST,WEST,CENTER(default)<br>use constraints in add to specify direction as "North", "South", "East", "West", "Center"<br>(case-sensitive)                      |                                                                                                                                                                          |
| GridLayout   | GridLayout()<br>GridLayout(int rows,int columns)<br>GridLayout(int rows,int columns, int horizontalgap, int verticalgap)<br>either row or cols can be zero but not both e.g <b>GridLayout(1,0) 1 row any nos of cols</b><br>default gap = 0 pixel |                                                                                                                                                                          |
| CardLayout   | CardLayout()<br>CardLayout(int horizontalgap, int verticalgap)<br>default gap = 0 pixel                                                                                                                                                           | void first(Container parent)<br>void next(Container parent)<br>void previous(Container parent)<br>void last(Container parent)<br>void show(Container parent,String name) |

### GridBagLayout

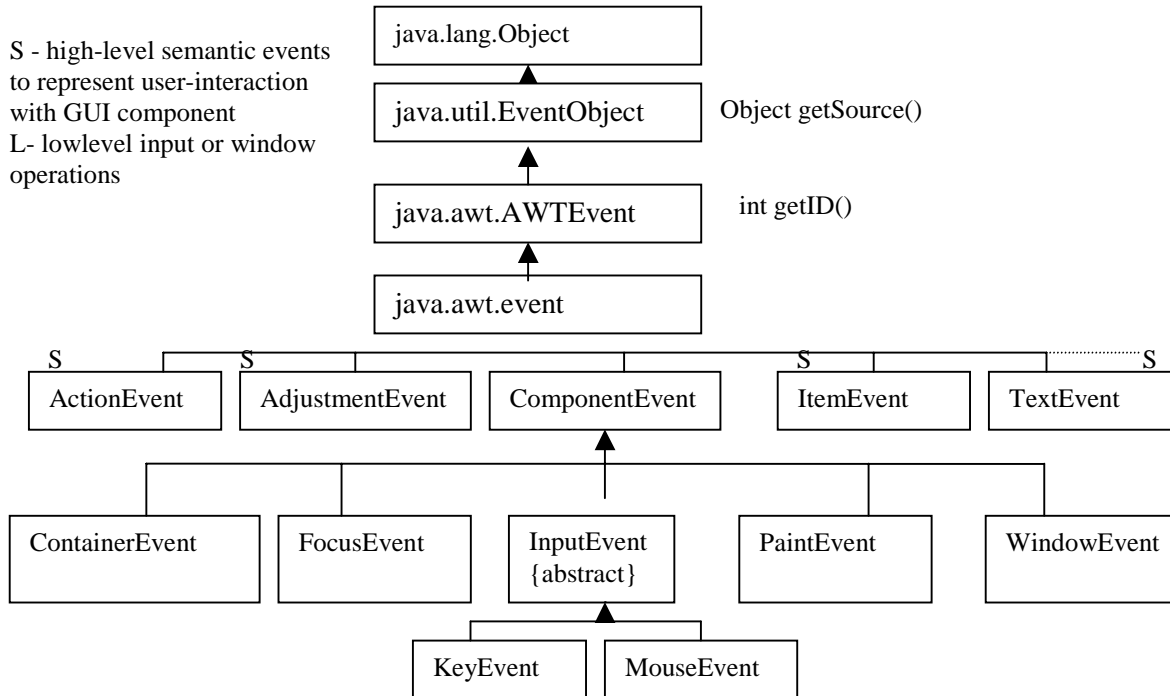


### Constructor

- GridBagLayout() - note dimensions are not specified
- GridBagConstraints() - these constraints can be reused for adding other components
- GridBagConstraints(
  - int gridx, int gridy, -Location Row,Col on upper left corner [**GridBagConstraints.RELATIVE**]
  - int gridwidth,int gridheight, - Dimension or nos of cells occupied
    - GridBagConstraints.RELATIVE - same as previous component
    - GridBagConstraints.REMAINDER - end of row [**1 cell**]
  - double weightx,double weighty, - Growth Factor/slack i.e. comp can be resized [**zero**]
  - int anchor, - Where to place it - **CENTER**(default),NORTH,SOUTH,EAST,WEST, NORTHEAST,NORTHWEST,SOUTHEAST,SOUTHWEST
  - int fill, - stretch & fill **NONE** (default) ,HORIZONTAL, VERTICAL, BOTH
  - Insets insets, - external padding (top,left,bottom,right) [ (0,0,0,0)]
  - int ipadx,int ipady) - internal padding to each side of the component(2\*ipadx,2\*ipady) [**0,0**]

## GUI Event Handling

Events are represented by objects in Java



- A GUI component may handle its own events by simply implementing the associated **event-handling interfaces**.
- However, it is also possible to extend the component's class and override its **event dispatching methods**.
- AWT components have methods of the form **processXEvent()**, where *X* refers to the event generated by the component. These methods dispatch events to the appropriate event listeners. You can override these methods to control the way in which events are dispatched. For example, the Button class uses the processActionEvent() method to dispatch ActionEvents to event listeners. You can override processActionEvent() in a subclass of Button to control how event dispatching takes place.
- In addition to *overriding the component's event-dispatching method*, you may also need to **enable the event**. This automatically takes place when an event listener is added to a component. However, if you do not add an event listener for the component, you'll have to invoke its **enableEvents()** method, passing the event identifier (defined in the AWTEvent class) as an argument. For example, to enable the ActionEvent in an object that is a subclass of Button, you invoke the following method for that object:  
**object.enableEvents(AWTEvent.ACTION\_EVENT\_MASK);**

|                 |                                                |                                                                                                                                         |
|-----------------|------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
|                 | Component                                      |                                                                                                                                         |
| ActionEvent     | Button<br>List<br>MenuItem<br>TextField        | String getActionCommand()<br>int getModifiers() retruns sum of modifier constants -<br>SHIFT_MASK   CTRL_MASK   META_MASK  <br>ALT_MASK |
| AdjustmentEvent | Scrollbar                                      | int getValue()                                                                                                                          |
| ItemEvent       | Checkbox<br>CheckboxMenuItem<br>Choice<br>List | Object getItem()<br>int getStateChange() [SELECTED   DESELECTED]                                                                        |
| TextEvent       | TextField<br>TextArea                          |                                                                                                                                         |

|                                                   |                                                                               |                                     |                                                                                                                                                                                                                                                                  |
|---------------------------------------------------|-------------------------------------------------------------------------------|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ComponentEvent<br>(handled <i>internally</i> AWT) | when comp is<br>hidden,show,moved,resized                                     | Component &<br>its subclasses       | Component getComponent()                                                                                                                                                                                                                                         |
| ContainerEvent<br>(handled internally AWT)        | when added or removed                                                         | Container                           |                                                                                                                                                                                                                                                                  |
| FocusEvent                                        | when gains or loses focus                                                     | Component                           | boolean isTemporary() focus                                                                                                                                                                                                                                      |
| InputEvent                                        |                                                                               | abstract class                      | getwhen() returns when the<br>key or mouse event occurred                                                                                                                                                                                                        |
| KeyEvent                                          | when presses or releases<br>key or both                                       | Component &<br>its subclasses       | KEY_PRESSED<br>KEY_RELEASED<br>KEY_TYPED<br>(keypress+keyrelease)<br><br>int getKeyCode()<br>char getKeyChar()                                                                                                                                                   |
| MouseEvent                                        | when moves or presses the<br>mouse                                            | Component &<br>its subclasses       | MOUSE_PRESSED<br>MOUSE_RELEASED<br>MOUSE_CLICKED<br>MOUSE_DRAGGED<br>MOUSE_MOVED<br>MOUSE_ENTERED<br>(boundary of component)<br>MOUSE_EXITED<br><br>int getX()<br>int getY()<br>Point getPoint()<br>void translatePoint(int dx,int<br>dy)<br>int getClickCount() |
| PaintEvent<br>handled internally                  | WHEN paint()/update<br>methods are invoked                                    |                                     |                                                                                                                                                                                                                                                                  |
| WindowEvent                                       | when an important<br>operation is performed on a<br>window given by constants | Window class<br>& its<br>subclasses | WINDOW_OPENED<br>WINDOW_CLOSING<br>WINDOW_CLOSED<br>WINDOW_ICONIFIED<br>WINDOW_DEICONIFIED<br>WINDOW_ACTIVATED<br>WINDOW_DEACTIVATED<br>Window getWindow()                                                                                                       |

## Principle of Event Delegation Model

Authority for event handling is delegated by implementing event listener *interfaces to provide implementations* that do the event handling

All events (XXX) listeners can be registered or removed

- addXXXListener e.g. <source>.addActionListener(<listener>)
- removeXXXListener

Notification of all listeners is *not guaranteed to occur in the same thread*.

*PaintEvent class is never handled in the event listener model*

**addActionListener & addItemListener are not part of java.awt.Component**

addMouseListener,addMouseMotionListener,addKeyListener are part of java.awt.Component

Since implementing interfaces means you have to implement all methods (unused as stubs)

**Adapters implements stubs for all methods for the corresponding interface.** (Applicable for low-level listeners only)

| Event Type      | Listener Interfaces<br><b>java.util.EventListener</b> | Listener methods                                                                                                                                                                                                                       | Adapter Implementation |
|-----------------|-------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| ActionEvent     | ActionListener                                        | actionPerformed(ActionEvent e)                                                                                                                                                                                                         | N.A.                   |
| AdjustmentEvent | AdjustmentListener                                    | AdjustmentValueChanged(AdjustmentEvent e)                                                                                                                                                                                              | N.A                    |
| ItemEvent       | ItemListener                                          | itemStateChanged(ItemEvent e)                                                                                                                                                                                                          | N.A                    |
| TextEvent       | TextListener                                          | textValueChanged(TextEvent e)                                                                                                                                                                                                          | N.A                    |
| ComponentEvent  | ComponentListener                                     | componentHidden(ComponentEvent e)<br>componentMoved(ComponentEvent e)<br>componentResized(ComponentEvent e)<br>componentShown(ComponentEvent e)                                                                                        | ComponentAdapter       |
| ContainerEvent  | <b>ContainerListener</b>                              | <b>componentAdded(ContainerEvent e)</b><br><b>componentRemoved(ContainerEvent e)</b>                                                                                                                                                   | ContainerAdapter       |
| FocusEvent      | FocusListener                                         | focusGained(FocusEvent e)<br>focusLost(FocusEvent e)                                                                                                                                                                                   | FocusAdapter           |
| KeyEvent        | KeyListener                                           | keyPressed(KeyEvent e)<br>keyRelease(KeyEvent e)<br>keyTyped(KeyEvent e)                                                                                                                                                               | KeyAdapter             |
| MouseEvent      | MouseListener                                         | mousePressed(MouseEvent e)<br>mouseReleased(MouseEvent e)<br>mouseClicked(MouseEvent e)<br>mouseEntered(MouseEvent e)<br>mouseExited(MouseEvent e)                                                                                     | MouseAdapter           |
|                 | <b>MouseMotionListener</b>                            | <b>mouseDragged(MouseEvent e)</b><br><b>mouseMoved(MouseEvent e)</b>                                                                                                                                                                   | MouseMotionAdapter     |
| WindowEvent     | WindowListener                                        | windowOpened(WindowEvent e)<br>windowClosing(WindowEvent e)<br>windowClosed(WindowEvent e)<br>windowIconified(WindowEvent e)<br>windowDeiconified(WindowEvent e)<br>windowActivated(WindowEvent e)<br>windowDeActivated(WindowEvent e) | WindowAdapter          |

### Explicit Event Handling

When an XEvent is received by a component, it is dispatched by the processEvent() method to a corresponding processXEvent() method of the component.

To enable all events of interest enableEvents() is passed a bit mask formed by OR'ing X\_EVENT\_MASK

e.g. **enableEvents(AWTEvent.WINDOW\_EVENT\_MASK| AWTEvent.KEY\_EVENT\_MASK)**

```
public void processWindowEvent(WindowEvent e){
    if(e.getID() == WindowEvent.WINDOW_CLOSING) <methodname(>;
    super.processWindowEvent(e);}
```

## Graphics / Painting

Abstract class `java.awt.Graphics` provides a **device-independent interface** for rendering graphics.

An instance of the `Graphics` class or its subclasses **cannot be created directly using a constructor** as abstract class.

All graphics object are instance of `java.awt.Graphics2D` but the signature of `paint()` uses `graphics` for compatibility with older version

It provides `graphics context` that can render the following targets

|            |                                                                                                            |
|------------|------------------------------------------------------------------------------------------------------------|
| Components | <code>void repaint()</code><br><code>void update(Graphics g)</code><br><code>void paint(Graphics g)</code> |
| Images     |                                                                                                            |
| Printers   |                                                                                                            |

**User Thread** (i.e. application) indirect call to `update` through `repaint()`

**AWT Thread** (i.e. AWT event handlers) direct call to `paint()` when the size of the component is changed

- calling `repaint()` eventually leads to invocation of **update()**
- `repaint()` clears the component screen-area , `update()` fills **current** b/g & foreground colors, it invokes `paint()`
- Note `repaint()` **schedules painting & method ends immediately**
- Most GUI control components are drawn using the **underlying windowing system** therefore **graphics should not** be used for drawing them (do not override `paint()` method for these)

Components that **do not have external graphical representation can be rendered** namely

- `Canvas` class
- `Container` class & it subclass : `Window, Frame, Dialog, Panel, Applet`
- Subclasses of `Component` which are not part of AWT

## Graphics class

To obtain the `graphics context`

**create()** method for a new object

**getGraphics()** method from existing

The creator of the `graphics context` should ensure that **dispose() method** is called to free resources

Properties encapsulated in `Graphics` class

- `Drawing Color`
- `Font` `Font getFont() | void setFont(Font c)` similar one from `Component` class can be used
- `Clip region`
- `Paint Mode`

Components class methods used for a component are

sizing `Dimension getSize() | Insets getInsets()`

color `Color getBackground()` `Color getForeground()` `Color getColor()`  
`void setBackground(Color c)` `void setForeground(Color c)` `void setColor(Color c)`

**constructors** **Color(int r,int g,int b)** 0-255  
**Color(float r,float g,float b)** 0.0-1.0  
**Color(int rgb)** bits r = 16-23, g=8 -15, b = 0-7

**13 predefined colors** [CASE-SENSITIVE]

`Color.black` `white` `red` `blue` `green` `yellow`

`darkGray` `gray` `lightGray`

`magenta` `orange` `pink` `cyan`

**class SystemColor provides the desktop color scheme** for current platform (constants -

`SystemColor.desktop` for background for desktop

`SystemColor.ontrol` for controls

`SystemColor.menuText` for text color for menus

e.g. class `A` extends `Applet`{  
    public void init(){ setFore...() }  
    public void paint(`Graphics g`){`g.drawString("Hi there",75,50);`}  
}

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Text Rendering | <pre>void drawString(String str,int x,int y) //baseline at the specified coord void drawChars(char[] data,int offset,int length,int x,int y) //start at offset void drawBytes(byte[] data, int offset,int length,int x,int y)</pre>                                                                                                                                                                                                                                                                                                                                                            |
| Line           | <pre>void drawLine(int x1,int y1,int x2,int y2)</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Rectangle      | <pre>void drawRect(int x,int y,int width,int height)outline : [area = (width+1)(height+1) pxl] void fillRect(int x,int y,int width,int height) fill : [area = width X height]  void drawRoundRect(int x,int y,int width,int height,int arcWidth,int arcHeight) void fillRoundRect(int x,int y,int width,int height,int arcWidth,int arcHeight)  void draw3DRect(int x,int y,int width,int height,int arcWidth,int arcHeight,boolean raised) void fill3DRect(int x,int y,int width,int height,int arcWidth,int arcHeight,boolean raised) void clearRect(int x,int y,int width,int height)</pre> |
| Ovals          | <pre>oval is bounded by an invisible rectangle void drawOval(int x,int y,int width,int height) void fillOval(int x,int y,int width,int height)</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Arc            | <pre>circular or elliptical starting point = starting angle in degrees (+ve angle counterclockwise direction) arc is bounded by a rectangle  void drawArc(int x, int y, int width,int height, int startAngle, int arcAngle) void fillArc(int x, int y, int width,int height, int startAngle, int arcAngle)</pre>                                                                                                                                                                                                                                                                               |
| Polygons       | <pre>constructor : Polygon(int[] xpoints,int[] ypoints,int npoints)  void drawPolygon(int[] xPoints,int[] yPoints,int nPoints) void drawPolygon(Polygon p) void fillPolygon(int[] xPoints,int[] yPoints,int nPoints) void fillPolygon(Polygon p)  void drawPolyline(int[] xPoints,int[] yPoints,int nPoints) //last vertex is not connected to the first vertex</pre>                                                                                                                                                                                                                          |
| Image          | <pre>boolean drawImage(Image img,int x,int y,ImageObserver observer)  call getGraphics() on the Image object  Component class implements the ImageObserver interface first call the image is loaded. If loading is incomplete the ImageObserver object is notified when more image is available &amp; call imageUpdate() method -&gt; repaint() -&gt; paint() Thus paint() is repeatedly called until the loading process is completed createImage(int x,int y) //Component class - creates image from scratch  getImage() addImage(Image img, int i)</pre>                                    |

## Font class

### **Font availability is platform dependent**

**canglyph(char c)** can be used to find if a font has a glyph for the given character

translate(int x,int y) to translate the origin

**constructor: Font(String name,int style,int size)**

Common fonts on all platforms

"Serif"(variable pitch) "SansSerif" w/o serif "Monospaced"(fixed pitch)  
"Dialog" "DialogInput" "Symbol"

Style

Font.BOLD Font.ITALIC Font.PLAIN (Font.BOLD | Font.ITALIC) //both

Size typographic point where 1 point = 1/72 inch

Properties of a font are accessed by using **FontMetrics class**. All measurements are in **pixels**

int getAscent() [baseline to top of char e.g. char 't']

int getDescent() [baseline to bottom e.g. char 'p']

int getMaxAscent() [baseline to ascentline e.g. char 'M' taking into consideration all characters for a font]

int getMaxDescent() [baseline to descentline]

int getLeading() [space between between 2 adjacent lines : descent of first & ascent of second]

int getHeight() [between baseline of adjacent lines]

int getMaxAdvance() [distance between one character & next in same line]

int charWidth(int ch)

int charHeight(int ch)

int stringWidth() [advance width of chars in the specified string]

To obtain fontmetric

use a component

```
Font f1 = new Font("Dialog",Font.ITALIC,12);  
FontMetrics m1 = component.getFontMetrics(f1);
```

graphics context

```
FontMetrics m2 = graphicsContext.getFontMetrics();  
FontMetrics m3 = graphicsContext.getFontMetrics(f2);
```

## Clipping

A Graphics object maintains a clip region.

The clip region of the graphic context defines area in which all drawing will be done.

Rectangle getClipBounds()

Shape getClip()

void setClip(int x,int y,int width,int height)

void setClip(Shape clip)

Every GUI has an AWT thread that monitors the user input & event handling

Painting Modes

void setPaintMode() //overwrite paint mode

void setXORMode(Color c) //alternates pixels between current color & specified color

## Images class

platform-independent interface to handle images

Toolkit class provides methods for loading images

```
Toolkit currentTk = Toolkit.getDefaultToolkit();
```

```
Image i1 = currentTK.getImage("cover.gif");
```

```
URL url = new URL("http://www....") //Applet class getImage(URL url)
```

```
Image i2 = currentTK.getImage(url); // getImage(URL url,String name)
```

## Applets

```
<APPLET
CODE or OBJECT    Bob.class    //classif in <OBJECT>.... </OBJECT> is used
HEIGHT            height
WIDTH             width
[CODEBASE]        url
[ALT]             show alt message if runtime JVM is deactivated
[ARCHIVE]         jars
[ALIGN]           alignment
[NAME]            name
[HSPACE]          space
[VSPACE]          space
<PARAM NAME=ident VALUE=value>          for a parameter NAME is must / VALUE is optional
alternate text    java illiterate browser
</APPLET>
```

Applets have 4 methods:

- init() - called once prior to the applet being painted.  
followed by start() & paint() methods  
override like a constructor use for e.g. create threads, construct GUI, process PARAM-VALUE parameters
- start() - called when applet is first shown or becomes visible again
- stop() - called when applet is hidden or no longer visible
- destroy() - called when applet object is about to be destroyed to relinquish any system resources

Other methods is applet class

```
URL getDocumentBase()
URL getCodeBase()
void showStatus(String msg)
String getAppletInfo() //textual description of the applet
String getParameter(String parameterName)
String[][] getParameterInfo() //each element of the array should be a set of 3 strings- name, desc, type
AppletContext getAppletContext()
```

Interface AudioClip is used to handle audio

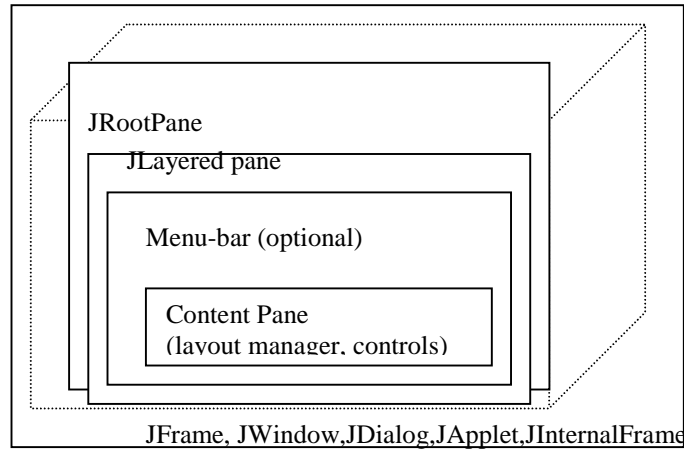
```
AudioClip getAudioClip(URL url)
AudioClip getAudioClip(URL url, String fileName)
void loop()
void play()
void play(URL url)
void play(URL url, String fileName)
void stop()
```

Applets in Web Browsers

- reading, writing, deleting files on local host no allowed
- running other applications from within the applet is prohibited
- calling System.exit(0) method to terminate the applet is not allowed
- no access to user, file or system information except to locale-specific information like Java version, OS name version, text encoding standard, file path line separator
- connecting to other hosts is not allowed

# SWING

## Containment Model for Root Components

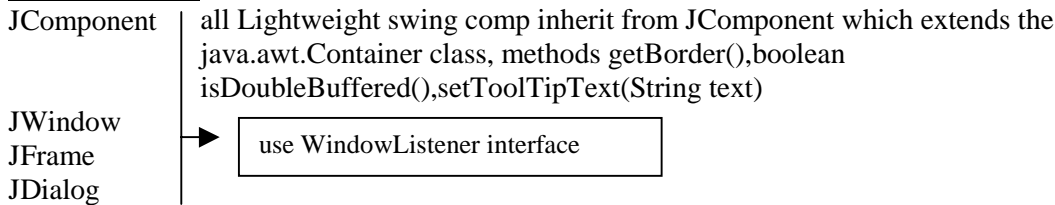


### RootPaneContainer Interface

- |                               |                                               |
|-------------------------------|-----------------------------------------------|
| Container getContentPane()    | void setContentPane(Container contentPane)    |
| Component getGlassPane()      | void setGlassPane(Component glassPane)        |
| JLayeredPane getLayeredPane() | void setLayeredPane(JlayeredPane layeredPane) |
| JRootPane getRootPane()       |                                               |
- ➔ create a RootPane
  - ➔ Create a container(i.e.contentpane) for RootPane
  - ➔ set layout for contentpane (default for contentpane is BorderLayout)
  - ➔ add components to contentpane

For JPanel which does not use the model traditional containment model is used  
create a new panel -> set its layout -> add components

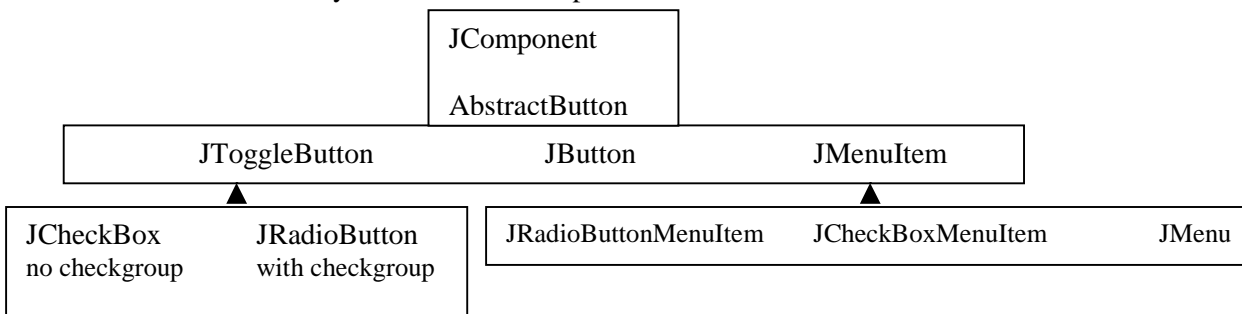
### Root Containers



JInternalFrame (lightweight) is not top-level container & does **not use WindowListener** interface  
uses the special **InternalFrameListener interface** usually placed inside a JDesktopPane container

### Button

the common functionality of button-like components is in AbstractButton class



AWT - Label & Buttons can contain only short text string  
 SWING- Label & Buttons can contain short text string, image or both  
 AWT - Menu Components are not part of main heirarchy  
 SWING - Menu components are part of the main inheritance hierarchy

ButtonModel Interface (Labels & Buttons)

- ButtonModel Interface provides methods to register listeners
- Method implemented by JLabel & AbstractButton class  
 Icon getIcon()                      void setIcon(Icon defaultIcon)  
 String getText()                      void setText(String text)  
 setBorder() from JComponent class
- AbstractButton class methods  
 void doClick()                      //key press/click  
 void doClick(int pressTime)  
 int getMnemonic()                      void setMnemonic(char mnemonic)    //hot key  
**ButtonModel getModel()                      void setModel(ButtonModel newModel)**

JMenuBar	can contain JMenu components - horizontal bar with pull-down menus it is attached to root component using setJMenuBar() method in JRootPane
JToolBar	A container that typically lets the user invoke commonly used actions can be undocked by user & left floating as a top-level window
JScrollBar	scrolling panes
JSlider	similar to scroll lets user select a value by sliding a knob
JProgressBar	tracks the progress of some process by displaying degree of completion
JComboBox	similar to choice editable text field with associated popup list
JSeparator	visual separator line
JList	<b>no scrolling</b> facilities for scrolling it should be decorated with jscrollpane <b>Default list model is DefaultListModel</b> Uses for content ListModel interface -> ListDataListener item rendering ListCellRenderer   <b>DefaultListCellRenderer</b> selection ListSelectionModel interface -> ListSelectionListener

Bounded Range Model (JScrollBar, JSlider, JProgressBar)

**BoundedRangeModel getModel()                      void setModel(BoundedRangeModel newModel)**

**ChangeListener** interface is used by buttons, scrollbar, slider, progressbar

Document Interface Model (text components)

- both the document interface & the JTextComponent class reside in javax.swing.text package
- uses DocumentListener

JTextComponent class {abstract}

JTextField	single line
JTextArea	multiline
JEditorPane	edit various kinds of content
JPasswordField	unreadable text
JTextPane	edit text that can be styled in various ways

Document getDocument()                      void setDocument(Document newdocument)  
 String getText()                      String getText(int off,int len)                      void setText(String t)  
 boolean isEditable()                      void setEditable(boolean b)

## Space-Saving components

JScrollPane	similar to the AWT component to view a portion of a larger component Jviewport class is used JScrollPane simply decorates a Jviewport component with scrollbars Each Jviewport component can have only one component
JTabbedPane	multiple pages using tabs addTab() insertTab() removeTabAt()
JSplitPane	user-adjustable divider between components

## Complex Models

JTable	▪ javax.swing.table Data Model - TableModel interface & Listener Column Model - TableColumnModel interface & Listener Row Selection model - ListSelectionModel interface & listener Column Header - JTableHeader class extends Jcomponent TableCellRenderer interface TableCellEditor interface
JTree	heirarchical data as tree nodes - leaf or branch TreeModel interface & Listener TreeSelectionModel interface & PropertyChangeListener (java.beans pkgg) TreeCellRenderer interface TreeCellEditor interface

## Compound Components

JFileChooser	similar to FileDialog is AWT select files or directories
JColorChooser	select a color using color model
JOptionPane	can be used to create a dialogbox with choices

## Additional Layout

### **borderLayout is default for JApplet**

#### Box Layout

BoxLayout(Container target, int axis) X\_AXIS | Y\_AXIS

row & col does not wrap regardless of resizing

attempt is made to make all component to the height of the highest component

*Box Class* creation of invisible separators

Slack (glue) - takes on left over space if available

Fixed size (strut & rigid areas) - use fixed amount of space

static Component createHorizontalGlue()

static Component createVerticalGlue()

static Component createHorizontalStrut(int width)

static Component createVerticalStrut(int height)

static Component createRigidArea(Dimension d)

Look & Feel

## Files & Streams

File class defines platform independent interface

public static final char separatorChar            pathname is absolute if it starts with a separator character  
 public static final **String** separator            / = **Unix** \ = **Windows**    := **Mac**

public static final char pathSeparatorChar    := Unix    ; = Windows e.g. c:\book;c:\windows;

public static final String pathSeparator

Constructors

File(String pathname)    which cannot be changed once file object is created, empty string mean current directory

e.g. File f1 = new File(File.separator+"book"+File.separator+"ch1");    /book/ch1    --absolute path

File f2 = new File(book"+File.separator+"ch1");                            book/ch1            ---relative path

File(String directoryPathname, String filename)

File(String directory, String filename)    directory = null means current directory

many methods throw a SecurityException e.g. if R / W access denied

Query

e.g. c:\document\...\book\ch1

String getName()	ch1
String getPath()	..\book\ch1
String getAbsolutePath()	c:\document\..\book\ch1
String getCanonicalPath() throws IOException	c:\book1\ch1
String getParent()	Unix "/book" then parent is /, windows c:\ Mac - HD:
boolean isAbsolute()	
long lastModified()	
long length()	file size in bytes
boolean equals(Object obj)	
boolean exists()	
boolean isFile()	
boolean isDirectory()	distinguish between a file & directory
boolean canWrite()	
boolean canRead()	
Listing Directory entries String[] list() String[] list(FilenameFilter filter) File listFiles() File listFiles(FilenameFilter filter) File listFiles(FileFilter filter)	filter implements either interface FilenameFilter{ boolean accept(File currentDirectory,String entryName); } interface FileFilter{ boolean accept(File pathName); }

New Files / renaming / deleting

boolean createNew() throws IOException

boolean mkdir()

boolean mkdirs()

boolean renameTo(**File** dest)

boolean delete()            a directory must be empty before u delete it

dir.list() returns the files contained in the instance of the File called dir

File f = new File("\*.\*) will obtain the contents of the current directory & populate the inst of the File class

## BYTE Streams: Input & Output Streams

int read() throws IOException int read(byte[] b) throws IOException int read(byte[] b,int off,int len) throws IOException reads() a byte but returns an int zeroing out the remaining bits returns a -1 when end of stream is reached	void write(int b) throws IOException void write(byte[] b) throws IOException void write(byte[] b,int off,int len) throws IOException
void close() throws IOException    does both close & flush void flush() throws IOException	

### *File Streams*

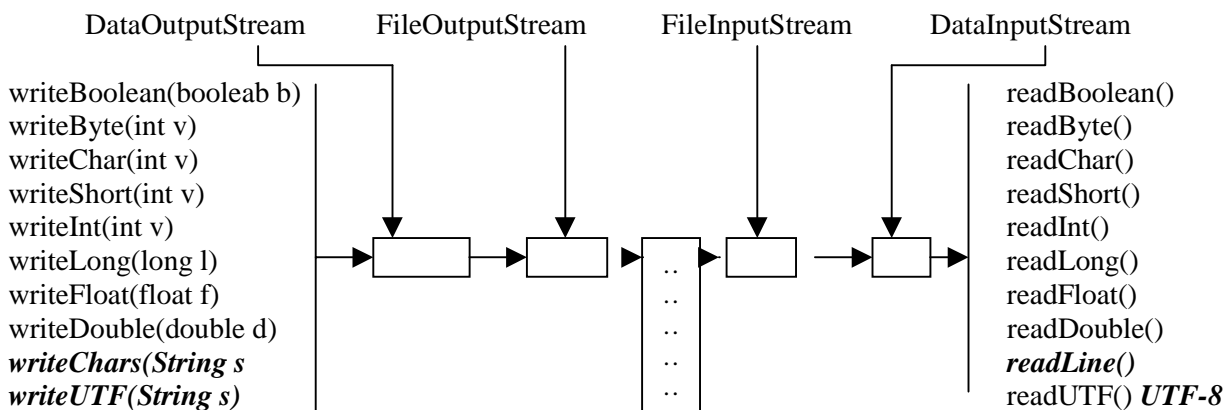
FileInputStream(String name) throws FileNotFoundException	FileOutputStream(String name) throws FileNotFoundException
FileInputStream(File file) throws FileNotFoundException	FileOutputStream(File file) throws FileNotFoundException contents reset unless append is indicated
FileInputStream(FileDescriptor fdObj)	FileOutputStream(FileDescriptor fdObj)
	FileOutputStream(String name, <b>boolean append</b> ) throws FileNotFoundException

### *Filter Streams (input - Filter,Buffered, Data, Pushback, output -Filter,Buffered,Data,Print)*

- FilterInputStream & FilterOutputStream classes & it subclasses are used
- subclasses BufferedInputStream & BufferedOutputStream for buffered input & output
- DataInputStream & DataOutputStream for java primitives
- DataOutputStream & ObjectOutputStream classes provide methods for writing binary representation of primitive data

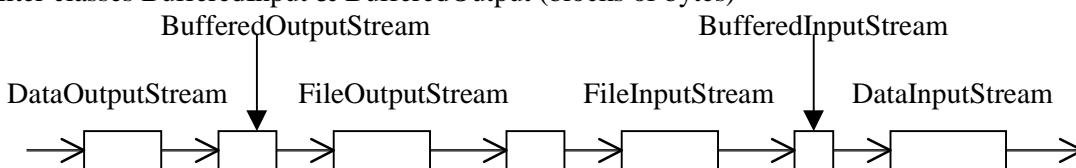
### *Primitives*

- DataOutputStream & DataInputStream implement (DataInput & DataOutput interfaces in java.io pkg)
- bytes can be skipped from DataInput stream using **skipBytes(int n)**



### *Buffered Byte Streams*

uses filter classes BufferedInput & BufferedOutput (blocks of bytes)



BufferedInputStream(InputStream i)	BufferedOutputStream(OutputStream o)
------------------------------------	--------------------------------------

## Character Streams : Readers & Writers

read & write Unicode characters using a specific character encoding (8859\_1 (default ie.ISO-Latin-1), 8859\_2, 8859\_3,8859\_4,UTF8 which is multi-byte encoding format).

has a LineNumberReader, Pushback Reader , PrinterWriter

int read() throws IOException //in range 0 - 65535 i.e. Unicode char

int read(char cbuf[]) throws IOException //returns a -1 at end of file

int read(char cbuf[],int off,int len) throws IOException

void write(int c) throws IOException //take int as argument but only writes out 16 bits

void write(char[] cbuf) throws IOException //note **no write() method for char**

void write(char[] cbuf,int off,int len) throws IOException

void write(String str) throws IOException

void write(String str,int off,int len) throws IOException

void close() throws IOException

void flush() throws IOException

long skip(long n) throws IOException //nos of characters to skip

### Unicode to Bytes

Input

- InputStreamReader(InputStream in)
- InputStreamReader(InputStream in,**String encodingName**) throws UnsupportedEncodingException

Output

- OutputStreamWriter(OutputStream out)
- OutputStreamWriter(OutputStream out,String encodingName) throws UnsupportedEncodingException
- String getEncoding()

### Print Writers

PrintWriter(Writer out)

PrintWriter(Writer out,boolean autoFlush) //flush if any println of PrintWriter class is called

PrintWriter(OutputStream out)

PrintWriter(OutputStream out,autoFlush)

print(<type> b) println(<type> t) [**Unix = \n, Windows = \r\n Mac = \r**]

**<type> is char[], String, Object** and all primitive types **except byte, short**

**checkError()** method is used to check for errors

### V.IMP - Reading & Writing Text Files

Primitives & Objects cannot be read directly from their textual representation. Characters must be read & converted to relevant values explicitly

- FileInputStream → InputStreamReader
- FileReader //uses default encoding format automatically

Three procedures to set up the print writer

- PrintWriter → OutputStreamWriter →FileOutputStream //OutputStream supplies the encoding sch
- PrintWriter → FileOutputStream //uses **default encoding format automatically**
- PrintWriter → FileWriter

### Buffered Character streams

BufferedReader(Reader in) | BufferedReader(Reader in, int size)

BufferedWriter(Writer out) | BufferedWriter(Writer out, int size)

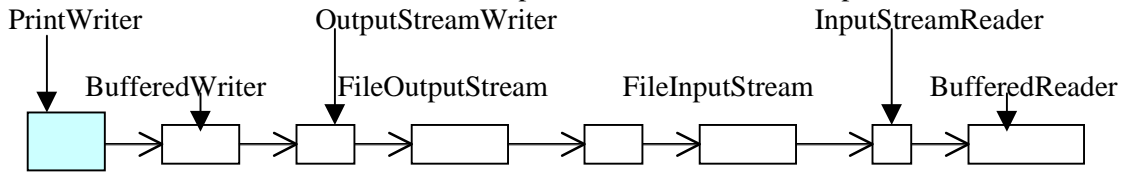
String **readLine()** throws IOException is used to read line text from the reader NULL is returned on EOF

**newLine()** in writer to provide platform dependent line-separated

e.g. byte b = (**byte**) **Integer.parseInt(bufferedReader.readLine())**

FileInputStream → InputStreamReader → BufferedReader (FileInputStream+InputStream = FileReader)

PrintWriter → **BufferedWriter** → OutputStreamWriter → FileOutputStream



```

FileInputStream fin;
FileOutputStream fout;
int i = fin.read();
fout.write(i);
fin.close();
fout.close();
  
```

```

FileOutputStream outfile = new FileOutputStream("abc.txt");
// BufferedOutputStream outbuff = new BufferedOutputStream(outfile);
DataOutputStream ostream = new DataOutputStream(outfile);
//DataOutputStream ostream = new DataOutputStream(outbuff);
ostream.writeInt(Integer.MAX_VALUE);
ostream.close();
  
```

```

FileInputStream infile = new FileInputStream("abc.txt");
DataInputStream instream = new DataInputStream(infile);
int i = instream.readInt();
instream.close();
  
```

Byte Stream	Character Streams
OutputStream InputStream	Writer Reader
ByteArrayOutputStream ByteArrayInputStream	CharArrayWriter CharArrayReader
	OutputStreamWriter OutputStreamReader
FileOutputStream FileInputStream	FileWriter FileReader
FilterOutputStream FilterInputStream	FilterWriter FilterReader
BufferedOutputStream BufferedInputStream	BufferedWriter BufferedReader
PrintStream	PrintWriter
DataOutputStream DataInputStream	
ObjectOutputStream ObjectInputStream	
PipedOutputStream PipedInputStream	PipedWriter PipedReader
	StringWriter StringReader
	LineNumberReader
PushbackInputStream	PushbackReader
SequenceInputStream	

Random Access (java.io.RandomAccessFile)

- inherits from Object class
- Allows to move forward & backward w/o reopening file
- objects of this class cannot be chained with streams
  - RandomAccessFile(**String** name,String mode) throws IOException //r | rw are only mode allowed
  - RandomAccessFile(**File** file,String mode) throws IOException
- IllegalArgumentException - if 'r' file not exist IllegalException error | 'rw' new file is created
- open file does not reset the contents of the file

```

long getFilePointer() throws IOException
long length() throws IOException //nos of bytes in the file
void seek(long offset) throws IOException //seek(length()) points to last byte of file
void close() throws IOException
  
```

```

e.g. RandomAccessFile rmfile = new RandomAccessFile(new File("ab.txt"),"rw");
long l = rmfile.length();
rmfile.seek(i);
rmfile.close();
  
```

### Object Serialization

**Serializable interface has no method** defined it is just used to indicate that the class is serializable

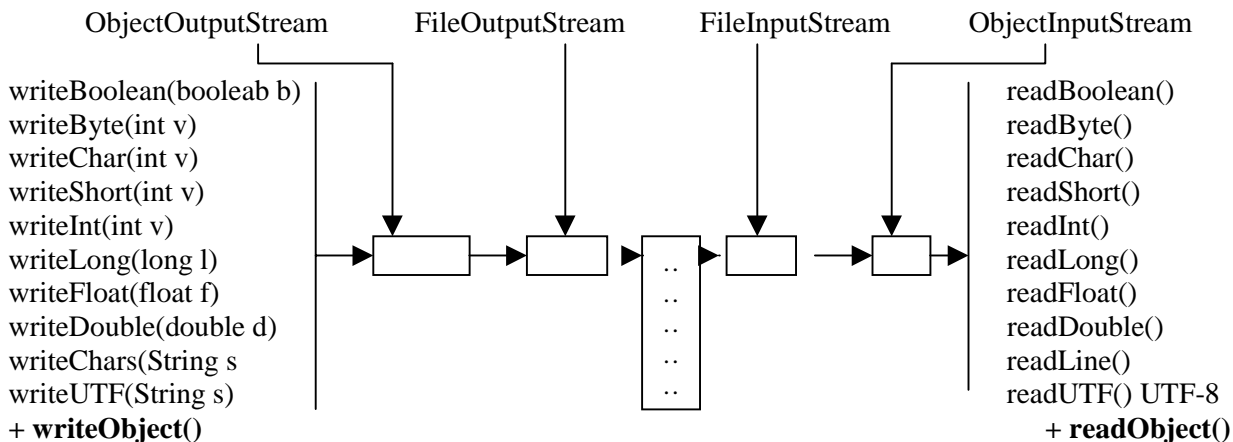
- sequence of bytes that can be recreated. ObjectInput & ObjectOutput - **read & write object** to & from streams
- Objects & values must be read in the same order they are serialized
- New Objects are created during deserialization so that no existing objects are overwritten

1. ObjectOutputStream must be chained to an output stream as follows -

- ObjectOutputStream(OutputStream out) throws IOException
- final void writeObject(**Object** obj) throws IOException
- ObjectInputStream(InputStream in) throws IOException, *StreamCorruptedException*
- final **Object** readObject(**Object** obj) throws *OptionalDataException, ClassNotFoundException, IOException*

```

FileOutputStream fout = new FileOutputStream("abc.dat");
ObjectOutputStream objstream = new ObjectOutputStream(fout);
String[] strarray = {"S7","S8","S6"};
String cstr = strarray[2]; //avoids duplication when same object is serialized - S6
objstream.writeObject(strarray);
objstream.writeObject(cstr);
String s = (String) objstream.readObject(); //An explicit cast is needed to convert the reference of
a deserialized object to the right type
  
```



### PipedInputStream/ PipedOutputStream class

The PipedInputStream class allows a pipe to be constructed between two threads and supports input through the pipe. It is used in conjunction with the PipedOutputStream class.

### SequenceInputStream class

enables two or more streams to be concatenated into a single stream

### Filter

- *Filters* are objects that read from one stream and write to another, usually altering the data in some way as they pass it from one stream to another.
- Filters can be used to buffer data, read and write objects, keep track of line numbers, and perform other operations on the data they move.
- Filters can be combined, with one filter using the output of another as its input. You can create custom filters by combining existing filters.

### FilterInputStream

- The BufferedInputStream class maintains a buffer of the input data that it receives. This eliminates the need to read from the stream's source every time an input byte is needed.
- The DataInputStream class implements the DataInput interface, a set of methods that enable String objects and primitive data types to be read from a stream.
- The **LineNumberInputStream** is used to keep track of input line numbers does not actually add line numbers to the data.
- The **PushbackInputStream** provides the capability to push data back onto the stream that it is read from so that it can be read again.

The java.io package declares ten interfaces.

DataInput and DataOutput interfaces	provide methods that support machine-independent I/O.
ObjectInput and ObjectOutput interfaces	extend DataInput and DataOutput to work with objects.
ObjectInputValidation interface	supports the validation of objects that are read from a stream.
Serializable and Externalizable interfaces	support the serialized writing of objects to streams.
FilenameFilter and FileFilter interfaces	are used to select filenames from a list.
ObjectStreamConstants interface	defines constants that serialize objects to and from streams

## JAVADOC

Documentation comment : starts with `/**` end with `*/` and is placed preceding

e.g. `/**`

```
 * com.foo.bar was invented by prof.Freddy. <1998>
```

```
 *
```

```
 */
```

- class & interface definitions
- member method definition including constructors
- member variable definitions

it is treated as regular comment if placed anywhere else.

("." " ") (period,space) in the text indicates end of sentence (e.g. after the prof.)Freddy.

For Javadoc to generate documentation each interface and class must be placed in their own compilation unit ie separate source file

hyperlinks can be specified using html tags in the comment

special tags - `@<tagname> <text>`

classnames if specified must be fully qualified

<code>@author</code>	Class & Interface	
<code>@version</code>	"	//version of class or interface
<code>@since</code>	Class, Interface & Members	//when the code was first introduced
<code>@deprecated</code>	"	// explanation
<code>@see</code>	"	// @see <fullpath>#<member name> // @see #topstack() member method
<code>{ @link }</code>	"	//creates an inline link
<code>@param</code>	Methods	// @param <parmname> <description>
<code>@return</code>	"	
<code>@throws</code>	"	<exception name> <explanation>
<code>@exception</code>	"	

CODE element of HTML is typically used for formatting keywords

Options for javadoc utility

Package names are specified using dot-notation(.)

`-d <directory>` destination directory for output files

`-author` to include author &

`-version` version nos

`-public` for public classes , interfaces & members only

`-protected` for protected/public classes , interfaces & members only `-DEFAULT` option

`-package` for package/protected/public classes , interfaces & members only

`-private` for ALL

`-nodeprecated` @deprecated paragraphs are not shown in the documentation

Compiled by Veena Iyer  
reference book:Mughal  
Thinking in Java, Bruce Eckel

Sno	Exam Name	Total qstns	Maha's Comments	URL
1	Javaranch - RulesRoundup game	144	This is a very <b>simple applet</b> which basically checks the Java Language rules by asking mostly yes/no qstns. It is good if we get 100% in this game before we take our final Sun SCJP2.	<a href="http://www.javaranch.com/">http://www.javaranch.com/</a>
2	Marcus green - Exam1	60	Marcus Green has composed 3 Mock exams in .html format , and the questions wording is discussed by many people in Javaranch, Marcus also participates in those and respects users comments, and if there is any ambiguity felt by many people , he changes the wording of those questions promptly. So the mock exams are refined by now and considered to be very good and <b>its toughness is close to the real exam</b> . It is good if we get atleast 90 to 95% in all the 3 exams to get a high score in Sun's SCJP2 Exam. Many people said the first time when we take Marcus's sample test, the % we get is close to the % we get from the real Exam. So save them for the last 3 weeks of your real exam and see your % on them.	<a href="http://www.jchq.net/mockexams/exam1.htm">http://www.jchq.net/mockexams/exam1.htm</a>
3	Marcus green - Exam2	60	- as above-	<a href="http://www.jchq.net/mockexams/exam2.htm">http://www.jchq.net/mockexams/exam2.htm</a>
4	Marcus green - Exam3	60	- as above -	<a href="http://www.jchq.net/mockexams/exam3.htm">http://www.jchq.net/mockexams/exam3.htm</a>
5	Applied Reasoning	77	Questions seem to be good and <b>slightly harder than the real Exam</b> . Some of the questions in this Applet seem to test our memory, asking which method in which class, like that. The GUI of the Exam simulator is also good	<a href="http://209.242.122.83/JavaCertification.html">http://209.242.122.83/JavaCertification.html</a>
6	Sun's sample papers	6	Sun has some official sample questions for SCJP2 exam in it site. <b>Don't miss them</b> .	<a href="http://suned.sun.com/USA/certification/pro_gsgwa.html">http://suned.sun.com/USA/certification/pro_gsgwa.html</a>
7	John Hunt	65	This is considered to be a standard and easy sample qstn paper with <b>questions scattered with all objectives</b> . This is the first sample test I took after I finished the 1st round of reading of R&H book.	<a href="#">View them</a>
8	Bill brogden (18 out of 32)	32	Bill Brogden is the author of well known book, Java2 Exam Cram. This is his official web site where he has an Exam Simulation Applet. Qstns considered to be <b>good</b> .	<a href="http://www.lanw.com/java/javacert/TestApplet6.htm">http://www.lanw.com/java/javacert/TestApplet6.htm</a>
9	Bill Brogden's Hardest questions	20	Since the name itself implies, that sample test is going to be <b>hard, I saved it for the last week</b> . I got 19 questions correct out of 20 asked.	<a href="http://www.lanw.com/java/javacert/HardestTest.htm">http://www.lanw.com/java/javacert/HardestTest.htm</a>
10	Poddar	42	This person had composed a set of <b>questions which mostly checks with String class</b> . To be through with the == and equals concept we should get all qstns correct in this test.	<a href="http://members.theglobe.com/apoddar/questions.html">http://members.theglobe.com/apoddar/questions.html</a>
11	MindQ	60	This is another <b>good set of question paper</b> . But it has 3 questions with wrong answers. Don't forget to write down your answers in a paper while taking this exam as the .html file has got some javascript errors.	<a href="#">Get MindQ Mock Exam</a> <a href="#">View MindQ Answers</a>
12	Majji - Exam1	10	This person also has composed a very <b>good collection</b> of questions.I came to know about him through <a href="http://www.javacert.com">www.javacert.com</a>	<a href="http://www.geocities.com/SiliconValley/Cable/1025/exam1.html">http://www.geocities.com/SiliconValley/Cable/1025/exam1.html</a>
13	Majji - Exam2	10	- as above -	<a href="http://www.geocities.com/SiliconValley/Cable/1025/exam2.html">http://www.geocities.com/SiliconValley/Cable/1025/exam2.html</a>
14	Majji - Exam3	10	- as above -	<a href="http://www.geocities.com/SiliconValley/Cable/1025/exam3.html">http://www.geocities.com/SiliconValley/Cable/1025/exam3.html</a>

				<a href="http://www.geocities.com/SiliconValley/Orchard/9362/java/javacert/newboone1-19.html">ble/1025/exam3.html</a>
15	Barry Boone exam	20	He is author of the another well known Java Certification book. His book is also known as 'BB' book among SCJP people. R&H and BB are those who started writing SCJP Certification books initially. Questions are good. <b>Some qstns were brought to Javaranch for ambiguity.</b>	<a href="http://www.geocities.com/SiliconValley/Orchard/9362/java/javacert/newboone1-19.html">http://www.geocities.com/SiliconValley/Orchard/9362/java/javacert/newboone1-19.html</a>
16	Barry Boone exam	20	- as above-	<a href="http://www.geocities.com/SiliconValley/Orchard/9362/java/javacert/newboone20-39.html">http://www.geocities.com/SiliconValley/Orchard/9362/java/javacert/newboone20-39.html</a>
17	Barry Boone exam	30	- as above -	<a href="http://www.geocities.com/SiliconValley/Orchard/9362/java/javacert/newboone40-70.html">http://www.geocities.com/SiliconValley/Orchard/9362/java/javacert/newboone40-70.html</a>
18	JDCert - applet to download	200 (I think so)	Questions are <b>simple and good</b> . It has an option of selecting SCJP1.1 and SCJP2.	<a href="http://www.geocities.com/SiliconValley/Orchard/9362/java/javacert/JDCert.html">http://www.geocities.com/SiliconValley/Orchard/9362/java/javacert/JDCert.html</a>
19	Jargon - applet to download	140	Another good applet. Has 'short answers' type questions. But <b>doesn't have explanation for the answers.</b>	<a href="http://www.sarga.com/java/jac.htm">http://www.sarga.com/java/jac.htm</a>
20	Jxam - Java Exam simulator applet to download	200	This Java application is <b>the best as accepted by many people</b> . Author has given the source code of the applet also. He has given explanation for all his questions. I like it a lot.	<a href="http://eddiemcnally.hypermart.net/jxam.htm">http://eddiemcnally.hypermart.net/jxam.htm</a>   OR <a href="#">download from here</a>
21	IBM online exam simulator	30	Questions seem to be good. But <b>few of the questions got wrong answers and they are discussed in javaranch.com</b> . You can use the 'search' facility in the Programmer Certification Forum with the keyword 'IBM' to get all IBM related discussion threads. You have to register first to get the mock Exam paper.	<a href="http://certify.torolab.ibm.com/">http://certify.torolab.ibm.com/</a> <a href="http://www.javaranch.com/ubb/Forum24/HTML/000497.html">http://www.javaranch.com/ubb/Forum24/HTML/000497.html</a>
22	Online Exam simulator	64	40 out of 64 questions are asked. <b>Worth to take</b> . The author seems to be an SCJP as well as an SCJD. (Sun Certified Java Developer)	<a href="http://bmil.freesevers.com/tester/test.html">http://bmil.freesevers.com/tester/test.html</a>
23	Another mock exam	56	Upto 36 qstns are written properly. All answers are not given. <b>Can be used as a good learning tool</b>	<a href="http://apollo.netservers.com/%7Efemibyte/javaquestions.html">http://apollo.netservers.com/%7Efemibyte/javaquestions.html</a>
24	Valiveru's exam applet		Questions are good. <b>I found 5 questions with wrong wording/wrong answers.</b>	<a href="http://valiveru.tripod.com/java/jvaltest.html">http://valiveru.tripod.com/java/jvaltest.html</a>
25	Mughal Khalid's exam applet	59	Questions are felt to be <b>HARDER THAN the real SCJP2 Exam</b> from Sun. So if you take this test and get low scores don't be upset. But their 'Java Certification Book' seems to be really good. I hadn't bought this book. But from others comment this book is more than just a Certification book. The author does not give answers in this applet. <b>I did a trick to get the answers</b> . I just answered only 1 question at a time and end the exam and saw the results. If I get 1/59 correct means, my answer is correct. Otherwise, if I get 0/59 correct means, my answer was wrong.	<a href="http://www.ii.uib.no/~khalid/pgjc/jcbook/engine.html">http://www.ii.uib.no/~khalid/pgjc/jcbook/engine.html</a>
26	Robert and Heller's self test	10	R&H has put the sample 10 questions on Chapter 6: Objects and Classes from their Java Certification Book, on the internet. <b>Good.</b>	<a href="http://developer.java.sun.com/developer/books/certification/testyourself.html">http://developer.java.sun.com/developer/books/certification/testyourself.html</a>
27	Jammi's applet	172	Questions are really <b>good and little harder</b> . I used to download the applet and take the test offline. It was a mystery for most of us, since this applet does not say how many questions are going to be asked. I went upto	<a href="http://www.jaworski.com/java/certification/">http://www.jaworski.com/java/certification/</a>

			150 qstns at a stretch. Finally from another Javaranch.com member we came to know the applet ends with 172 questions. The author also has got SCJP review applet which is also good.	
29	JQuest applet	20	The author of this applet is an young Engineering College student from Mumbai, India. <b>Some of his questions are taken from other Mock Exams.</b> Any how, the Exam Simulator GUI seems to be good.	<a href="http://jquest.webjump.com/">http://jquest.webjump.com/</a>
30	Java Exam applet1	158	This applet also <b>seems to be ok.</b> It pulls out 30 random questions from a database of 158 questions. What I used to do was I just download the applet and go offline (disconnect the internet connection) and take the test	<a href="http://www.geocities.com/SiliconValley/Screen/5046/test1.html">http://www.geocities.com/SiliconValley/Screen/5046/test1.html</a>
31	Java Exam applet2	158	.same as above	<a href="http://www.geocities.com/SiliconValley/Screen/5046/test2.html">http://www.geocities.com/SiliconValley/Screen/5046/test2.html</a>
32	Java Exam applet3	158	same as above	<a href="http://www.geocities.com/SiliconValley/Screen/5046/test3.html">http://www.geocities.com/SiliconValley/Screen/5046/test3.html</a>
33	Abhilash's site	80	I think this person is a JLS (Java Language Specification) favorite. Most of his questions are really very interesting and <b>test us really on our Java fundamentals explained in JLS.</b> Take it. It is really worth. You can get the answers from the author through email.	<a href="http://www.angelfire.com/or/abhilash/Main.html">http://www.angelfire.com/or/abhilash/Main.html</a>
35	Java tidbits	6	Not a test actually. Has some <b>handy charts</b> and some important notes. We can test ourselves on this.	<a href="http://www.absolutejava.com/articles/java-tidbits.html">http://www.absolutejava.com/articles/java-tidbits.html</a>
36	Mock test on JavaFundamentals.	13	This Sample Test, has questions on Java Fundamentals. For each question , the answer as well as <b>explanation are given. Good.</b>	<a href="http://www.absolutejava.com/articles/test-your-java-knowledge.html">http://www.absolutejava.com/articles/test-your-java-knowledge.html</a>
37	Naveen's SCJP site	59	Recently put up on the net. Questions are <b>good.</b> Some of his questions are discussed in <a href="http://www.javaranch.com">www.javaranch.com</a> in Programmer Certification Forum.	<a href="http://www.javaprep.com/questions/test.html">http://www.javaprep.com/questions/test.html</a>
39	Deepak's Sample Java Mock Exam Application	250	Mr.Deepak is an SCJP2 himself has his own homepage which has more SCJP related info. I did not get a chance to use his application since it was not available at the time when I was preparing for the exam.	<a href="http://deepak.htmlplanet.com/feed.html">http://deepak.htmlplanet.com/feed.html</a>
40	Free Brainbench Certification Exam	60	This is <b>not a must to take for the SCJP2</b> felt by many people. BrainBench offers their own Java Certification and their objectives are little different from Sun's. So you may get some questions which are not needed for Sun's SCJP exam.However,we can use this as a learning tool. For a restricted time Brain <b>Bench's Java Certification Exam is available free</b> to take online.	<a href="http://www.brainbench.com/testcenter/brainbench/cert.jsp?core=%2fvtc%2fcert%2fr egisterfast.jsp&amp;vid=losby">http://www.brainbench.com/testcenter/brainbench/cert.jsp?core=%2fvtc%2fcert%2fr egisterfast.jsp&amp;vid=losby</a>
41	JavaCaps	60	This exam also looks good. I haven't gone through each and every question in the exam. I thought of letting you all know about JavaCaps resources and get the feedback from you all. Please do send your comments. I will evaluate and add the comments and send the same to the author also. This site's author has got some java related tutorials also.Thank you.	<a href="http://www.javacaps.com/sjpc_mockexam.html">http://www.javacaps.com/sjpc_mockexam.html</a>