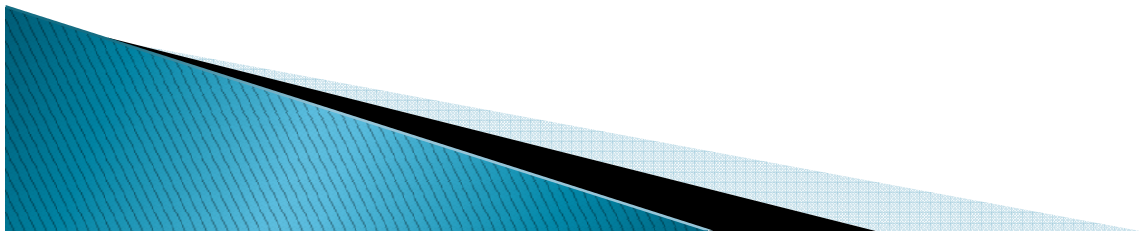
JAVA PROGRAMMING

This ppt is dedicated to my Supreme Friends "AMMA BHAGAVAN"
& for the Java Beginners.

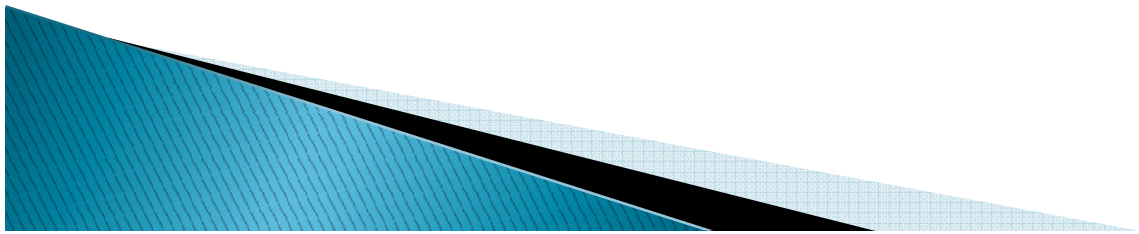
Developed by,

S.V.G.REDDY,
Associate Professor,
CSE Dept., GIT,
GITAM UNIVERSITY.



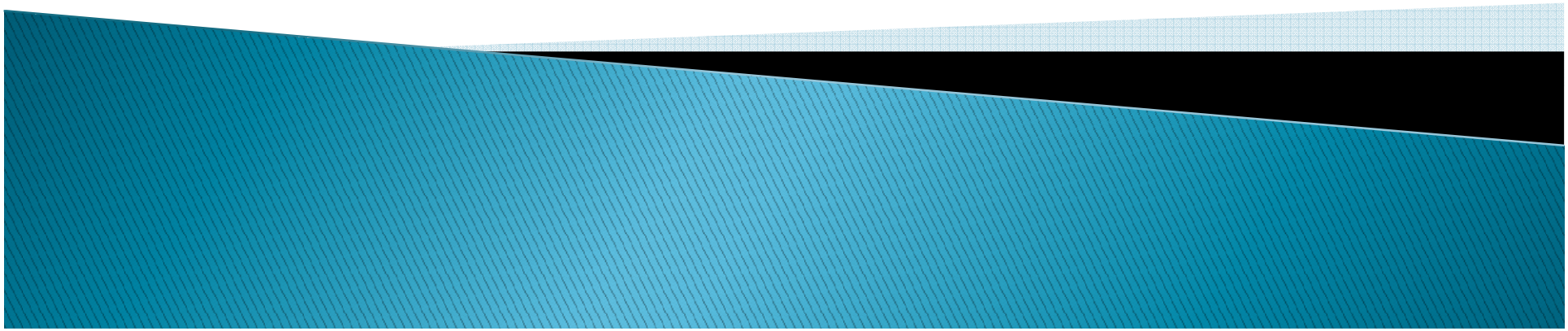
Chapter 1

Java Evolution & Overview of Java Language



What is java?

Java is a programming language originally developed by James Gosling at Sun Microsystems, which is a platform independent language which can run on any environment having java virtual machine.

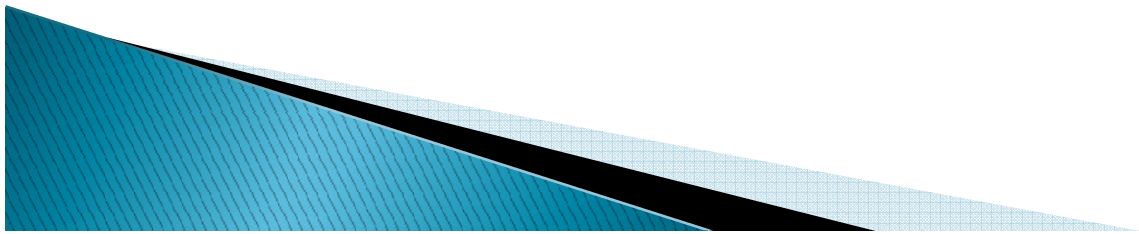


Features of Java

Object Oriented: Java is a complete object oriented language. Everything in java resides with in a class.

Robust : Java is intended for writing programs that must be reliable in a variety of ways. Java puts a lot of emphasis on early checking for possible problems, later dynamic (runtime) checking, and eliminating situations that are error prone.

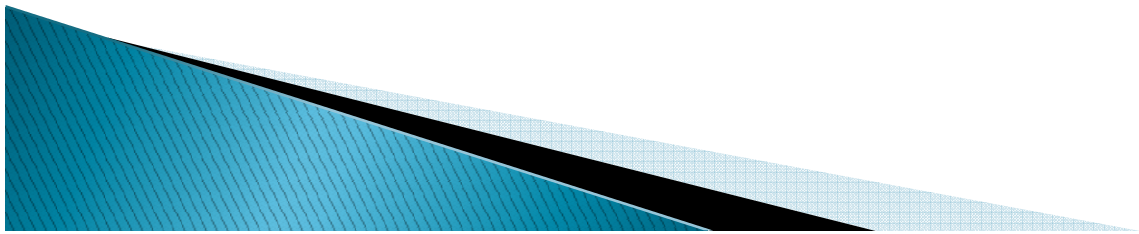
Secure: Java verifies all the memory accesses made in a program and it also ensures that no viruses are communicated with an applet. Abscence of pointers ensures that program cannot gain access to memory locations without proper authorizations.



Distributed: Java can share resources over the networks. Thus it enables many users to work on a single project from various locations.

Multithreaded: Java can handle multiple tasks simultaneously. This is achieved by dividing the program into small and independent executable units known as threads. There is a time lag between the C.P.U and the other devices. When one thread waits for a particular device, the C.P.U executes the other threads in the mean time.

Dynamic: Java is capable of linking a new class libraries, methods and objects at run time.

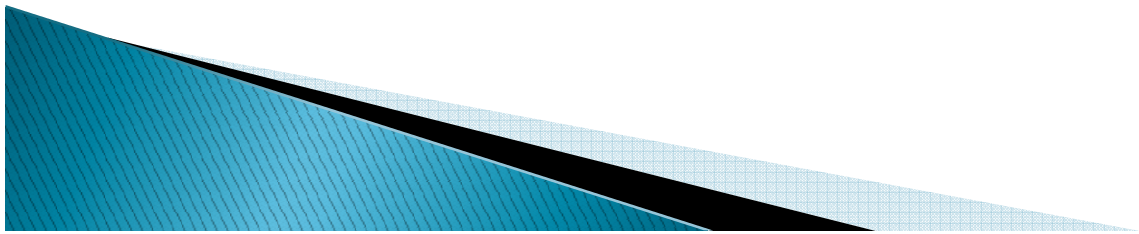


Compiled and Interpreted: Unlike any other programming language, the execution of the program is done in two stages

1.**Compilation**:When the program is compiled, a byte code is generated from the source code.

2.**Interpretation**:The generated byte code is interpreted by the interpreter(JVM). Due to this, the program once compiled can be run over and over again from the byte code. Hence its compile once and run forever scheme.

Platform independent and Portable:The byte code generated can run on any platform having a JVM.Even though the JVM(Interpreter) changes from machine to machine, all of them can execute the same byte code.



Java Development Kit (JDK)

Java Development kit contains with the following tools

appletviewer Enables us to run java applets(without actually using a java-compatible browser).

java : java interpreter, which runs applets and applications by reading and interpreting bytecode files.

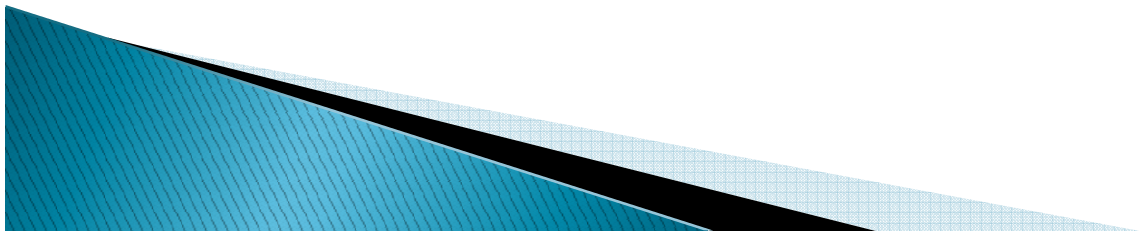
javac : The java compiler, which translates java source code to bytecode files that the interpreter can understand.

javadoc : Creates HTML format documentation from java source code files.

javah : Produces header files for use with native methods

javap : Java disassembler, which enables us to convert bytecode files into a program description.

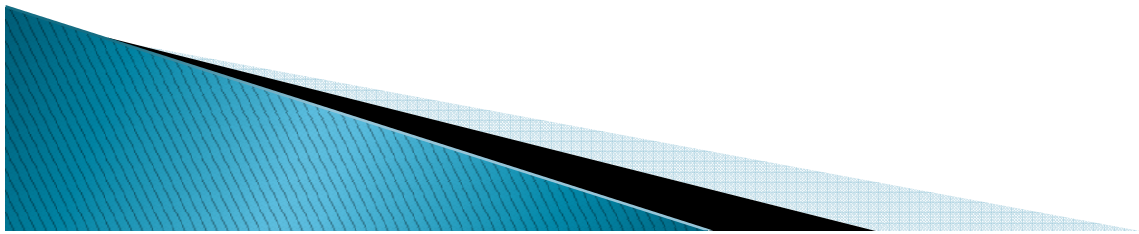
jdb : Java debugger, which helps us to find errors in our programs.



Application programming interface(API)

Java supports several packages which contains hundreds of classes & methods.

java.lang.*	–	language support package
Java.util.*	–	utilities package
java.io.*	–	Input/Output package
java.net.*	–	networking package
java.awt.*	–	AWT package
java.applet.*	–	Applet package



Java Virtual Machine(JVM)

The java compiler produces an intermediate code known as bytecode for a machine that does not exist, this machine is called the java virtual machine, it exist only inside the computer memory.

Source code

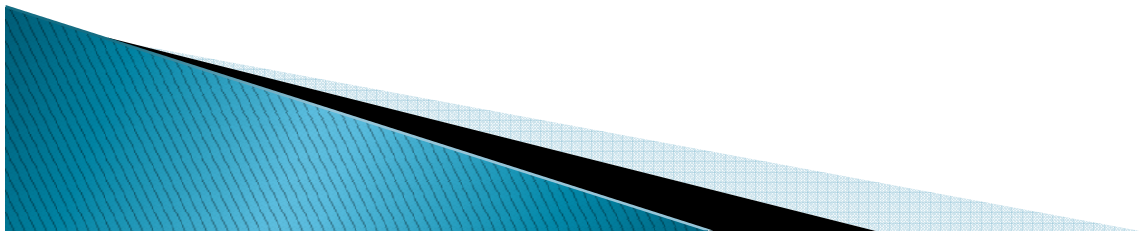
Byte code

The bytecode is executed by the interpreter in this step:

Java tokens

Java language includes five types of tokens

1. Reserved Keywords
2. Identifiers
3. Literals
4. Operators
5. Separators

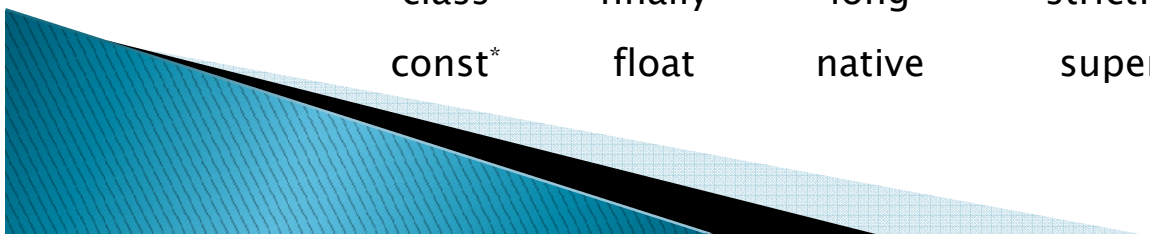


Keywords

Keywords are essential to define language. Specific features of language are implemented using them.

There are 50 Keywords in java language. They are,

abstract	continue	for	new	switch
assert	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const*	float	native	super	while



Identifiers

These are programmer-designed tokens.

They are used for naming classes, methods, variables, objects, labels, packages and interfaces in a program.

Java identifiers follow following **rules**:

- They can have alphabets, digits , and underscore and dollar sign characters.
- They must not begin with a digit.
- Uppercase and lowercase letters are distinct.
- They can be of any length.

Examples:

- Average
- dayTemperature
- HelloJava
- PRINCIPAL_AMOUNT

Literals

Sequence of characters representing a constant value to be stored in a variable.

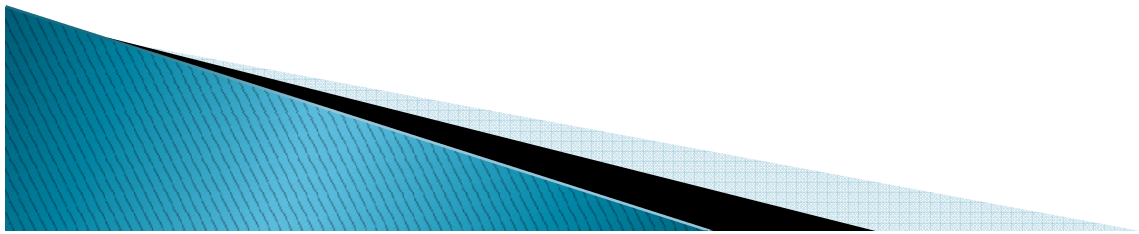
Eg- int a=10; // 10 is an integer literal

boolean b=true; // true is a Boolean literal

Operators

An operator is a symbol that takes one or more arguments and operates on them to produce a result.

Eg- 2+3, 5*6 etc..



Seperators

Symbols used to indicate where groups of code are divided and arranged.

Symbol	Name	Description
()	Parentheses	Used in: 1. method signatures to contain lists of arguments. 2. expressions to raise operator precedence. 3. narrowing conversions. 4. loops to contain expressions to be evaluated
{}	Braces	Used in: 1. declaration of types. 2. blocks of statements 3. array initialization.
[]	Brackets	Used in: 1. array declaration. 2. array value dereferencing
< >	Angle brackets	Used to pass parameter to parameterized types.
;	Semicolon	Used to terminate statements and in the for statement to separate the initialization code, the expression, and the update code.
:	Colon	Used in the for statement that iterates over an array or a collection.
,	Comma	Used to separate arguments in method declarations.
.	Period	Used to separate package names from subpackages and type names, and to separate a field or method from a reference variable.

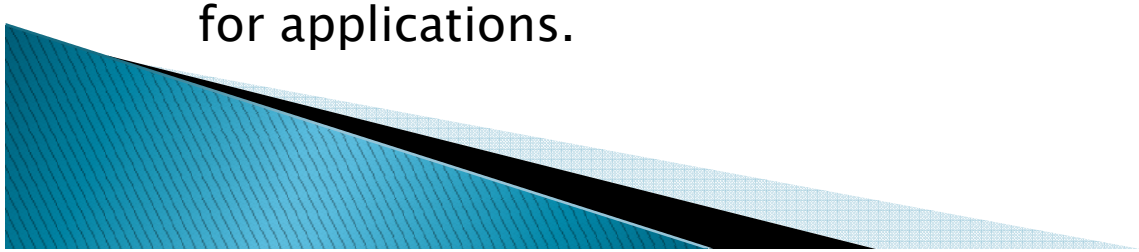
Applications

Since Java has inbuilt networking features so it is widely used for developing network related programs and softwares. Socket programming using JAVA is very simple and efficient and simple as compared to c.

It is widely used for creating web applications and it can also generate static HTML content like php and JavaScript so it is used in website designing too.

In application software development Java is widely used because it is a fully Object Oriented Programming language.

Creating Graphical User Interface(GUI) in Java is very simple and efficient as compared to c and c++.So it is also used in creating GUI for applications.

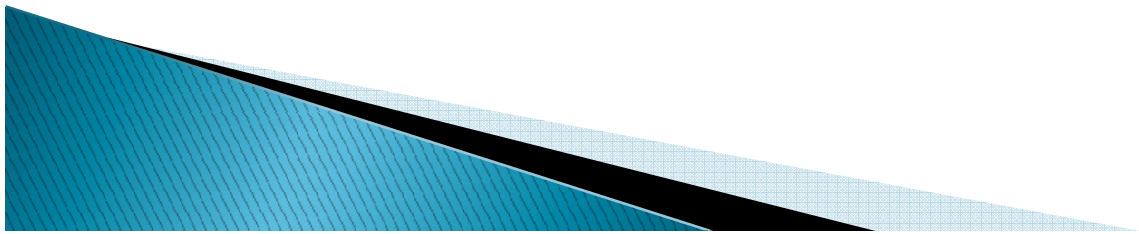


WWW

The World Wide Web, abbreviated as WWW and commonly known as the Web, is a system of interlinked hypertext documents accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks.

www comprises of

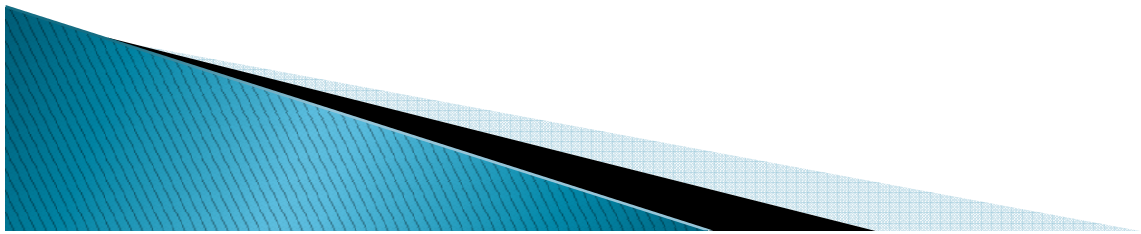
- Internet
- Browser
- Http
- Html
- URL



Web Browser

A software application that is meant for converting the HTML into the readable format as seen in webpage.

for eg: internet explorer, mozilla firefox, opera etc..



Java Basics

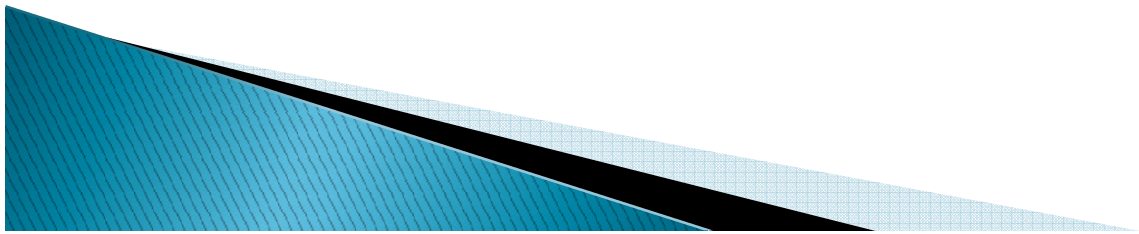
Before we start programming in java we need to set the path.

What is path setting and why to set the path-

It gives the location of the JDK. so when ever we compile using "javac", it goes to system variable called "path", and read the information of the location of "javac" in JDK & JRE tools.

What happens if we don't give the path-

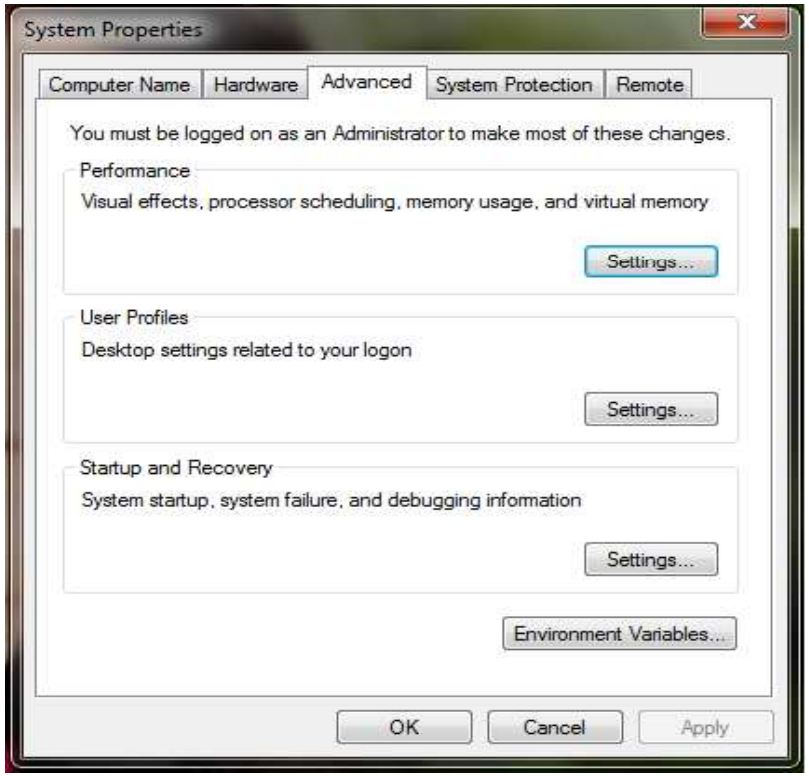
if we don't specify the path then we need to save the program in c:\jdk\bin, compile & execute at the same location.



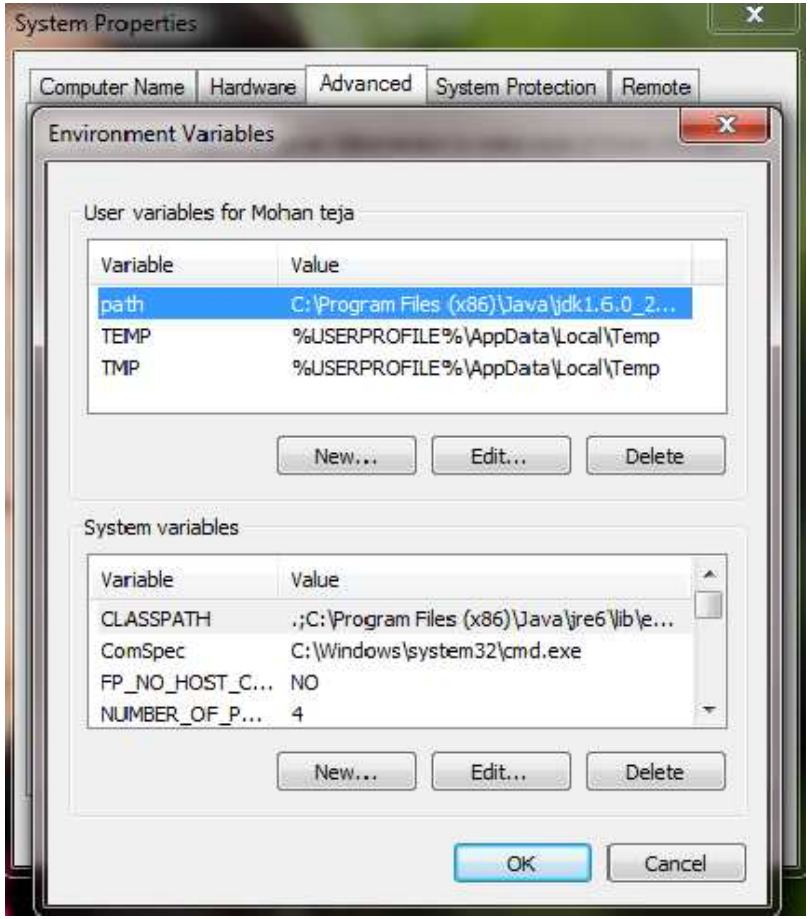
How to set the path

Right click on my computer->properties

Properties->Advanced



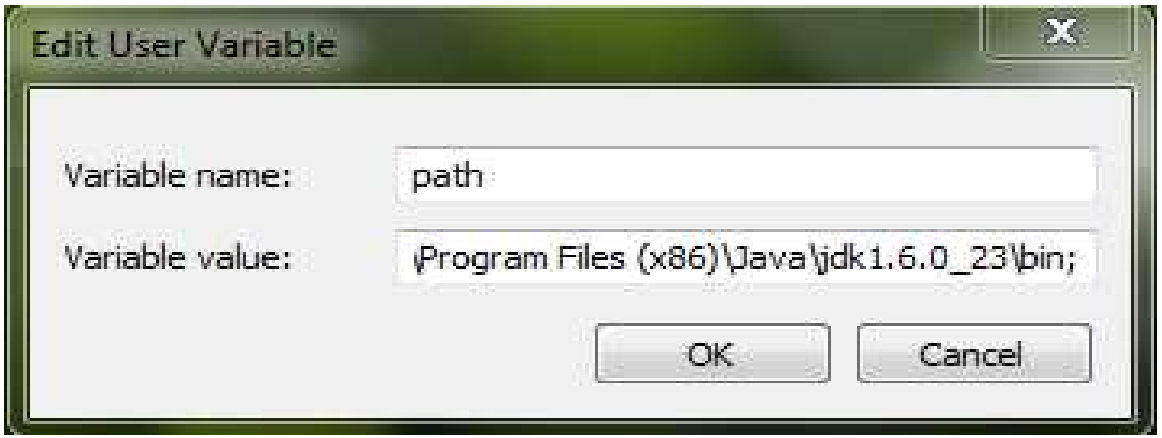
Advanced->Environment variables(at bottom)



Select path in user variables and click edit.

Type the variable name as path.

To set the “java bin” directory path to the path variable, append the directory path in the variable value text box, separated by a semi colon i.e. the location of the JDK in the directory.



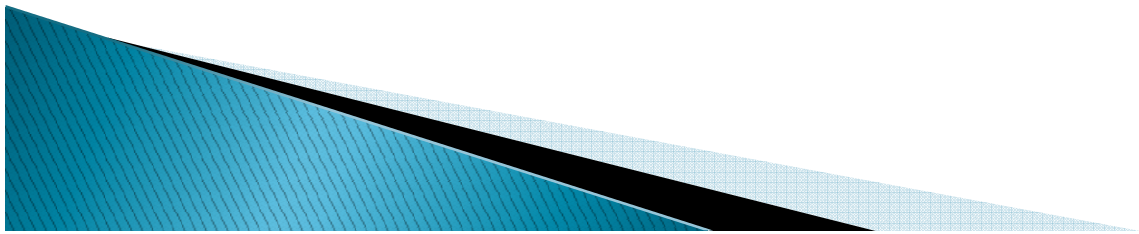
Click OK to close the Edit System variable box.

Click OK to close Environment variable box.

Click OK to close System properties box.

Structure of a java program

Documentation	• Suggested • It includes comments to increase understandability.
Package Statement	• Optional • Specify the packages to import.
Import Statement	• Optional • Importing the parts of other packages.
Interface Statements	• Optional • Specifies the interfaces to be implimented.
Class Definitions	• Optional • Defines the classes
Main class definition	• Essential • It holds the main method, the execution starts from here.

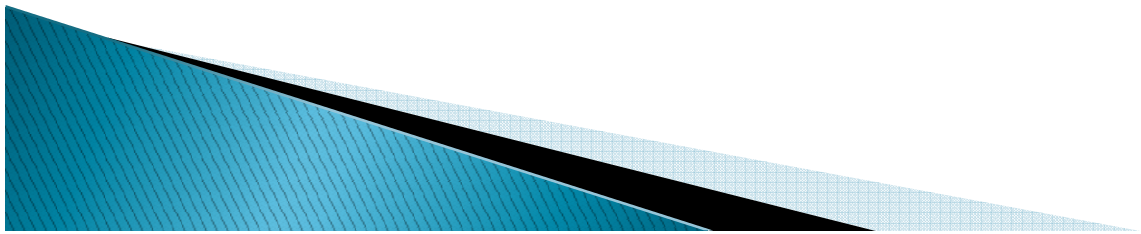


Sample Java Program

```
class Hello //class name
{
    public static void main(String[] args)//main function
        {
            System.out.println("Hello, Friends");
                                                //Using predefined functions
        }
 }
```

Output

Hello, Friends



Saving java file

Type the given code in notepad file.

You can use any text editor to write the code.

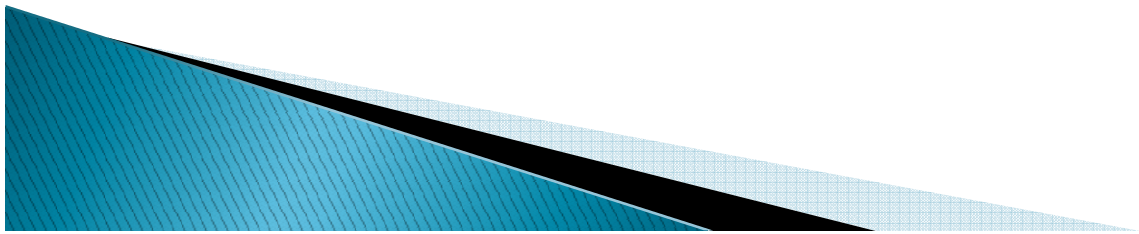
Save the file with the extension ".java"

Note: File name should be same as the class name which contains the main method.

EXAMPLE

The above program would be saved as "Hello.java"

Note: Hello.java is not equal to hello.java



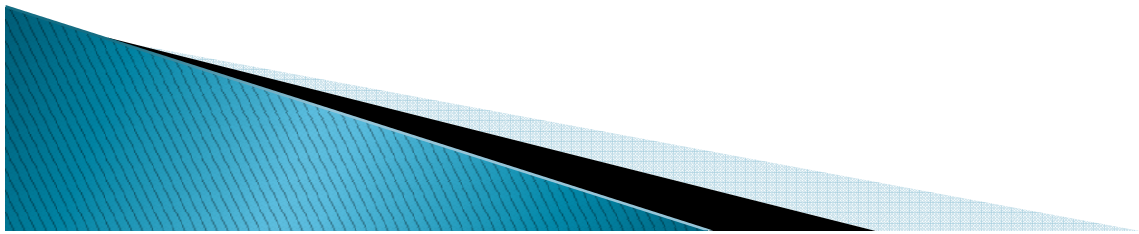
Compilation

To compile the program open run and type cmd.
Change the directory to location of the file.

Now type the command- javac Hello.java

The sample program was saved in “j” drive of my system.

A class file will be created with the name Hello.class this file contains the bytecode.

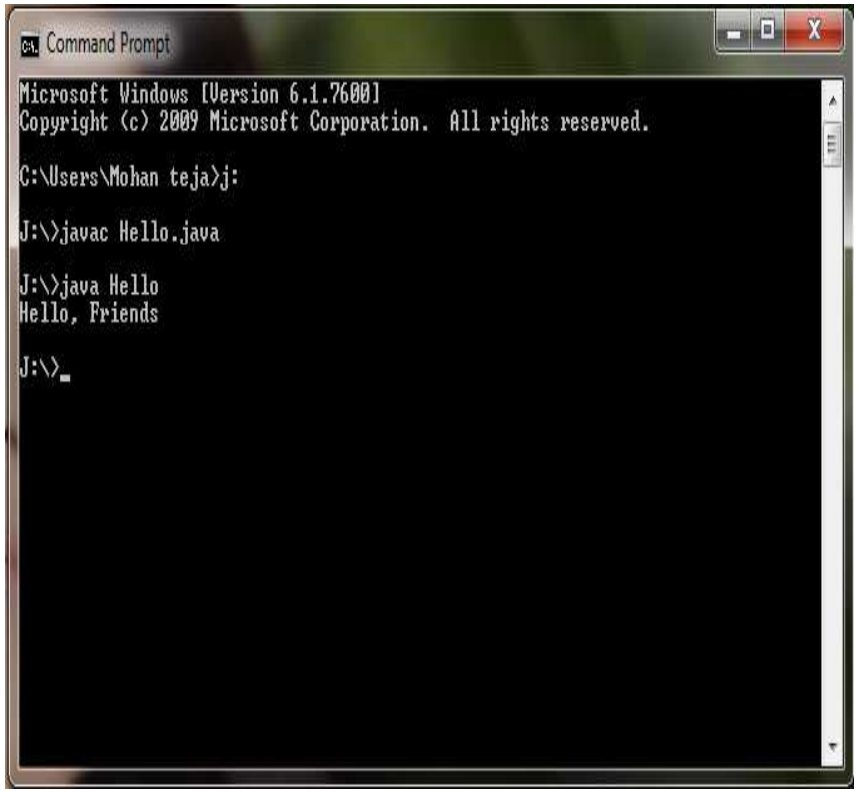


Compilation

Interpretation

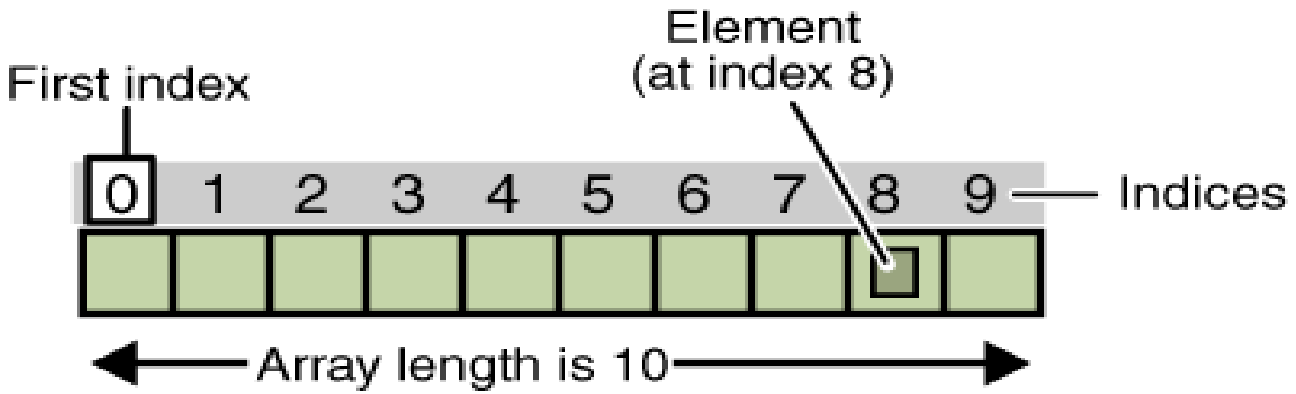
Now type the command – java Hello

Program will be executed and the output is as follows:



Arrays

An array is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created. After creation, its length is fixed

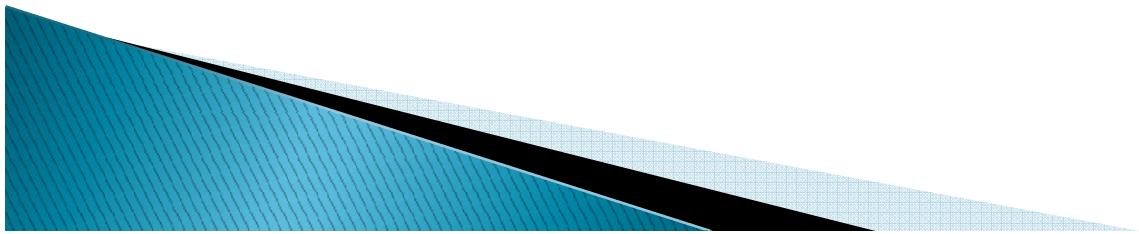


Example Program –

```
class ArrayDemo
{
    public static void main(String[] args)
     {
        int[] anArray; // declares an array of integers
        anArray = new int[4];//allocates memory for 4 integers
        anArray[0] = 100; // initialize first element
       anArray[1] = 200; // initialize second element
           anArray[2] = 300; // etc.
        anArray[3] = 400;
        System.out.println("Element at index 0: " + anArray[0]);
        System.out.println("Element at index 1: " + anArray[1]);
        System.out.println("Element at index 2: " + anArray[2]);
        System.out.println("Element at index 3: " + anArray[3]);
     }
}
```

Output:

```
Element at index 0: 100
Element at index 1: 200
Element at index 2: 300
Element at index 3: 400
```



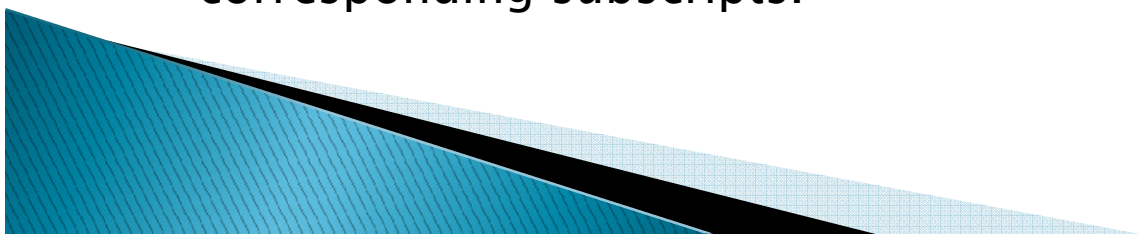
2D Arrays

2-dimensional arrays are usually represented with a row-column "spreadsheet" style. Assume we have an array, a, with two rows and four columns.

```
int[][] a = new int[2][4];  // Two rows and four columns.
```

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]

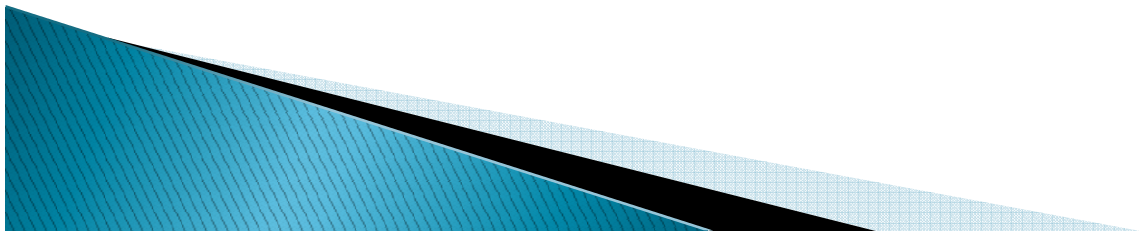
Two-dimensional arrays are usually visualized as a matrix, with rows and columns. This diagram shows the array a with its corresponding subscripts.



You can assign initial values to an array in a manner very similar to one-dimensional arrays, but with an extra level of braces. The dimension sizes are computed by the compiler from the number of values. This would allocate a 3x3 board.

```
class td
{
public static void main(String args[])
{
int t[][]={{3,6},{8,9}};
int i,j;
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
System.out.print(t[i][j]+"\t");
}
}
}
}
```

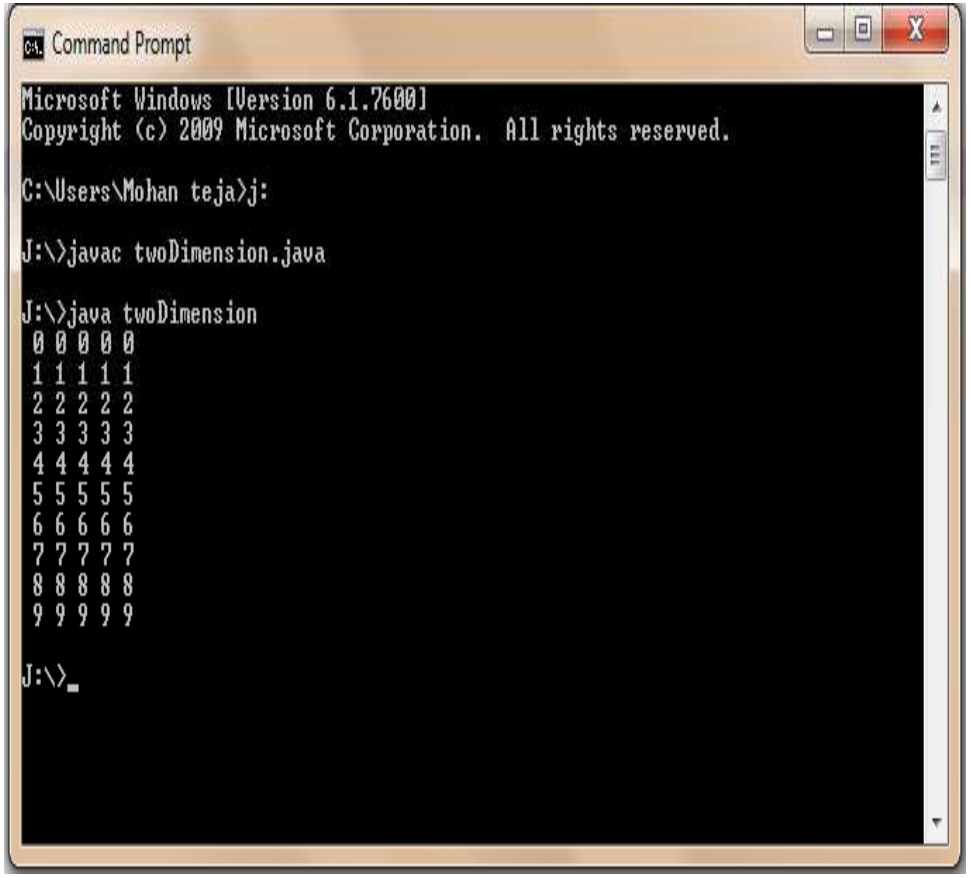
Output - 3 6 8 9



Program for 2D Arrays

```
public class twoDimension
{
  public static void main(String[] args)
{
    int[][] a2 = new int[10][5];
    for (int i=0; i<a2.length; i++)
{
      for (int j=0; j<a2[i].length; j++)
{
        a2[i][j] = i;
        System.out.print(" " + a2[i][j]);
      }
      System.out.println("");
   }
  }
}
```

Output


Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Mohan teja>j:
J:\>javac twoDimension.java
J:\>java twoDimension
0 0 0 0 0
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
6 6 6 6 6
7 7 7 7 7
8 8 8 8 8
9 9 9 9 9
J:\>

Command Line Arguments

The parameters which are provided to the program at the command line.

Eg: d:\> java a 10 20 30

Here, “java” is an interpreter, “a” is file name, 10,20,30 are arguments passed to the program while executing the program

And the notation of String args[ ] in main method will be cleared with the command line arguments.

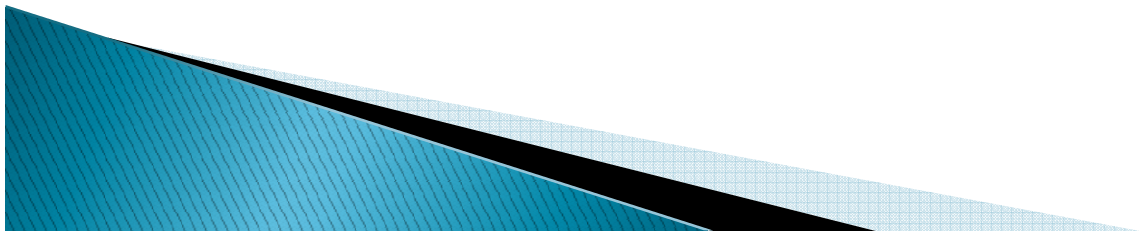
public static void main(String[ ] args)

The “String[ ] args” declares args as an array of strings.

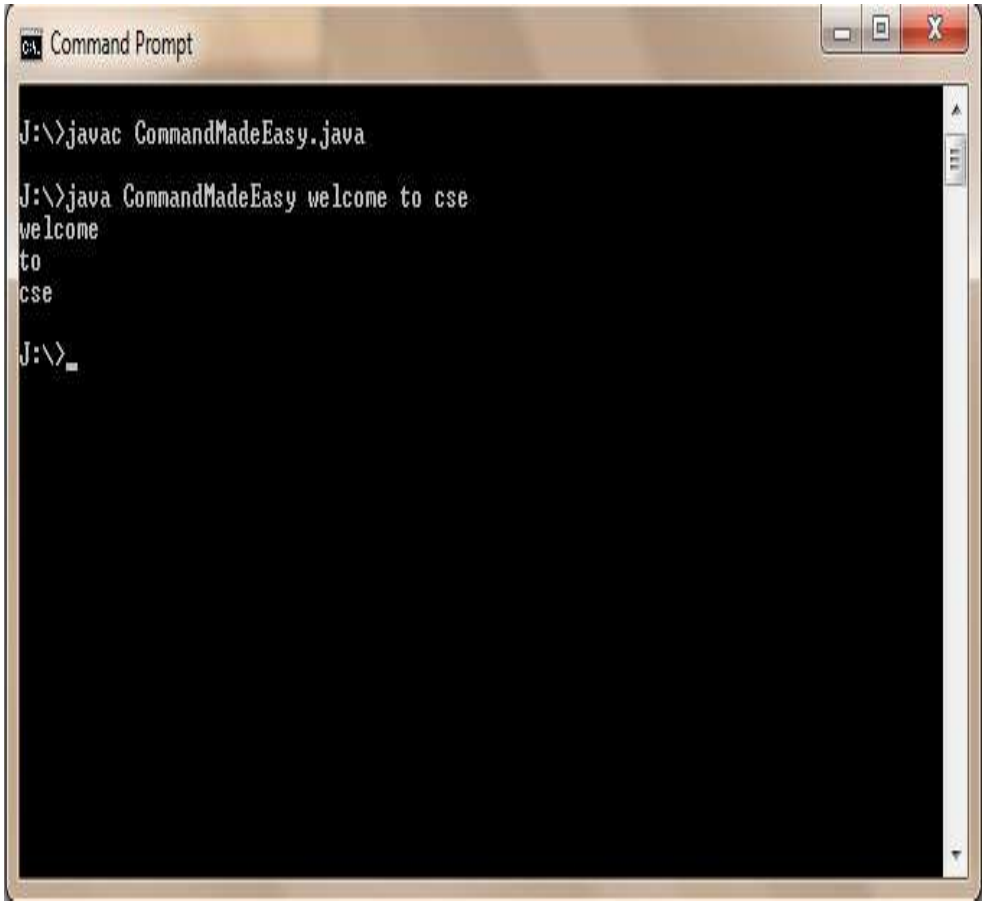
Args by default holds the strings given to the program while executing it.

Program

```
class CommandMadeEasy
 {
    public static void main(String args[])
   {
       System.out.println(args[0]);//holds the first string
       System.out.println(args[1]);//holds the second string
       System.out.println(args[2]);//holds the third string
   }
 }
```



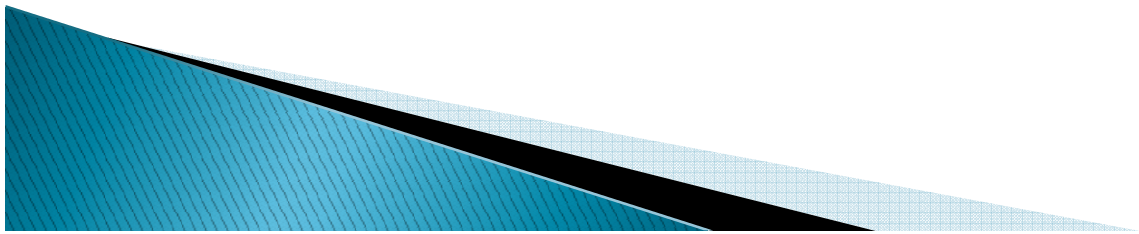
Output


Command Prompt
J:\>javac CommandMadeEasy.java
J:\>java CommandMadeEasy welcome to cse
welcome
to
cse
J:\>

Implementation of command line arguments

```
class applyCommand
{
  public static void main(String[] args)
  {
    String name;
    name=args[0];//first value given in command line
    int Roll;
    Roll=Integer.parseInt(args[1]);//converting string to integer
    float marks;
    marks=Float.parseFloat(args[2]);//converting string to float
    System.out.println("Name is:"+name);
    System.out.println("Roll no:"+Roll);
    System.out.println("Marks are:"+marks);
  }
}
```

Note : And thus the command line arguments makes the program interactive



Output


Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Mohan teja>j:
J:\>javac applyCommand.java
J:\>java applyCommand ragava 32 30.0
Name is:ragava
Roll no:32
Marks are:30.0
J:\>

Strings

Strings, which are widely used in Java programming, are a sequence of characters. In the Java programming language, strings are objects.

The Java platform provides the String class to create and manipulate strings.

Creating Strings –

The most direct way to create a string is to write:

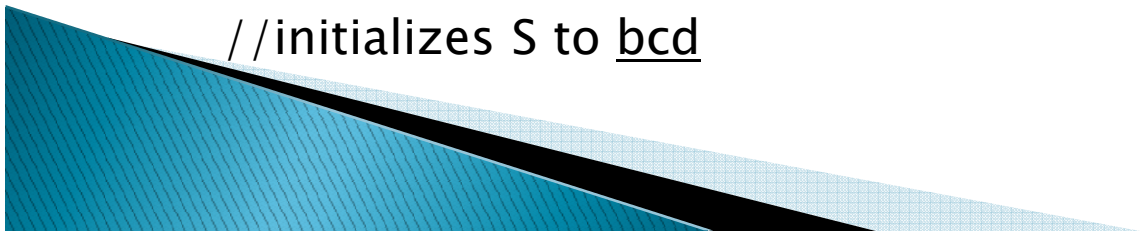
1. String greeting = "Hello CSE!"; // OR
2. String greeting=new String("Hello CSE!"); //OR
3. String(char chars[],int StartIndex,int noOfChars)

char chars[]={'a','b','c','d','e','f'};

String S=new String(chars,1,3);

//1 represents starting index and 3 is length of string

//initializes S to bcd



String methods

s1 = s.**toLowerCase**() –new String with all chars in lowercase

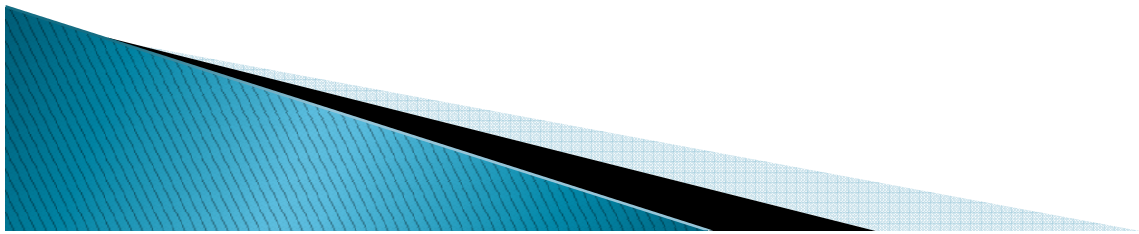
s1 = s.**toUpperCase**() – new String with all chars in uppercase

s1 = s.**trim**() – new String with whitespace deleted from front and back

s1 = s.**replace**(c1, c2) – new String with all c1 characters replaced by character c2.

i=s.**equals**(t) – true if the two strings have equal values

b = s.**equalsIgnoreCase**(t) – same as above ignoring case



i = s.**length**() – returns length of the string s.

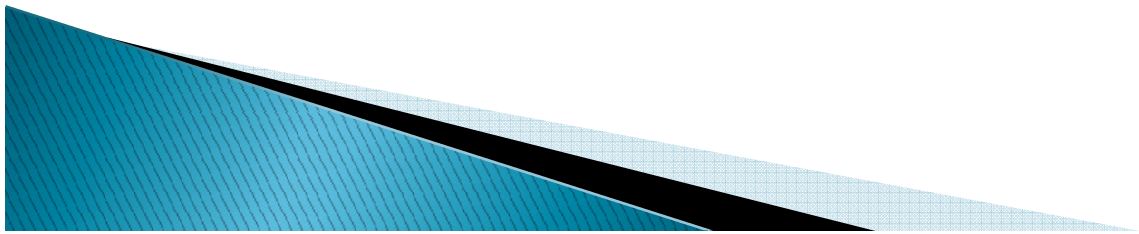
c=s.**charAt**(i)–returns char at position i in s.

s1 = s.**substring**(i) – substring from index i to the end of s.

s1 = s.**substring**(i,j) – substring from index i to BEFORE index j of s.

i = s.**indexOf**(t) – index of the first occurrence of String t in s.

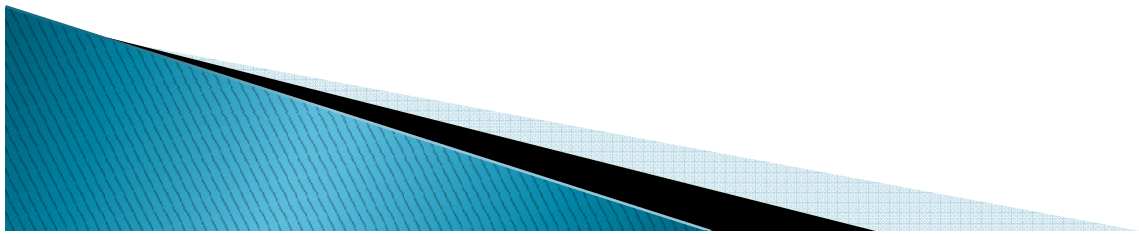
i = s.**indexOf**(t,i) – index of String t at or after position i in s.



Basic I/O streams

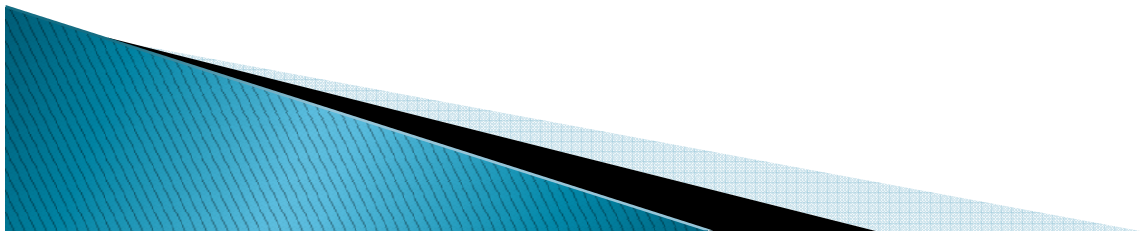
To Read **Integer**, **float** number from keyboard while Running the program

```
import java.util.*;
class a
{
public static void main(String args[])
{
System.out.println("Enter the integer number : ");
Scanner ob=new Scanner(System.in);
int i=ob.nextInt();
System.out.println("Enter the float number : ");
Scanner ob1=new Scanner(System.in);
float j=ob1.nextFloat();
System.out.println("Sum is : "+(i+j));
} }
```



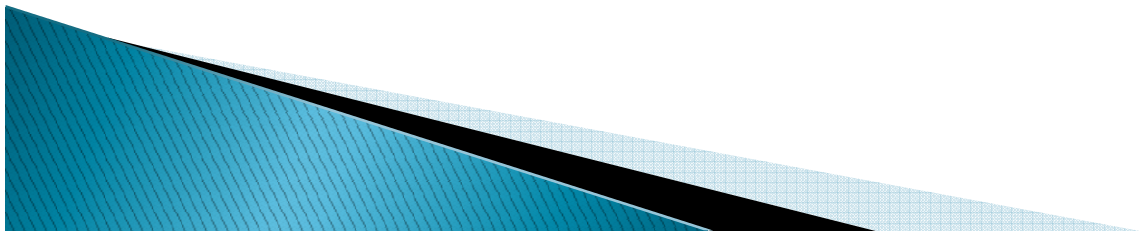
Read a character

```
import java.io.*;
class BRRead
{
public static void main(String args[])throws IOException
{
char c;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter characters ('q' to quit. ");

do
{
c=(char) br.read();
System.out.println(c);
}while(c!='q');
} }
```

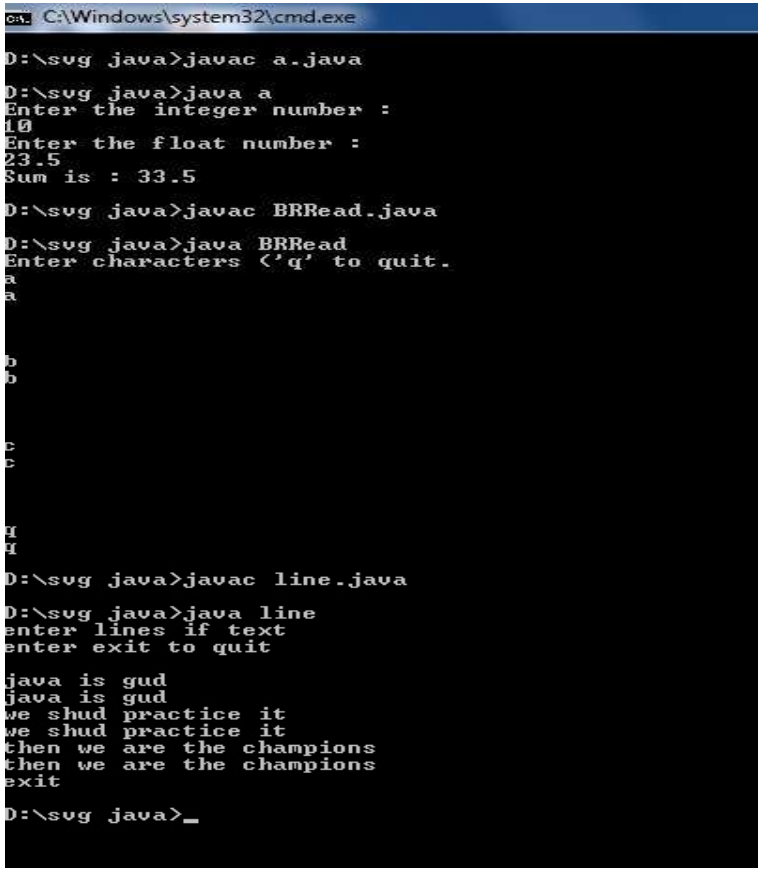


Read a String

```
import java.io.*;
class line
{
public static void main(String args[])throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
String str="  ";
System.out.println("enter lines if text");
System.out.println("enter exit to quit");
do{
System.out.println(str);
str=br.readLine();
}while(!str.equals("exit"));
} }
```



Output – Basic I/O Streams

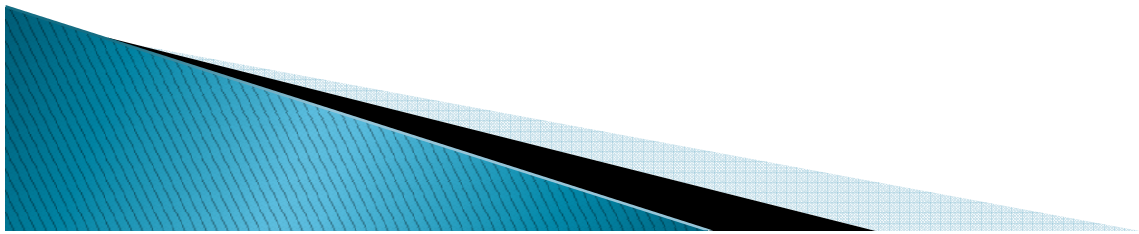


Chapter 2

Classes, Objects and Methods

Inheritance

Managing Errors And Exceptions



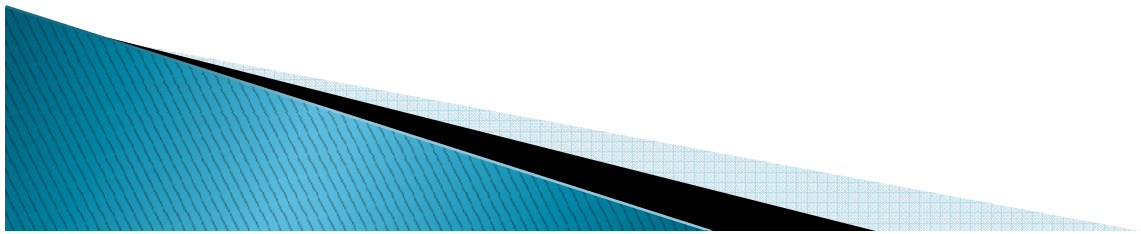
Defining a class

A class is a user defined data type that serves to define its properties. The variables created are termed as instances of classes.

Syntax:

```
class classname
{
fields declaration;
methods declaration;
}
```

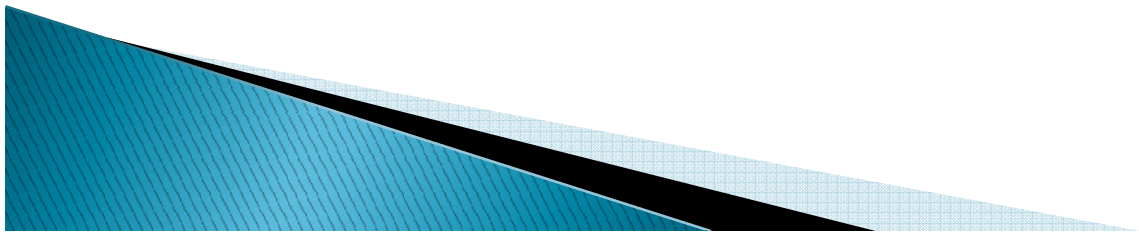
Note - The method



Fields declaration

Data is encapsulated in a class by placing data fields inside the body of the class definition. These variables are also called as instance variables(member variables).

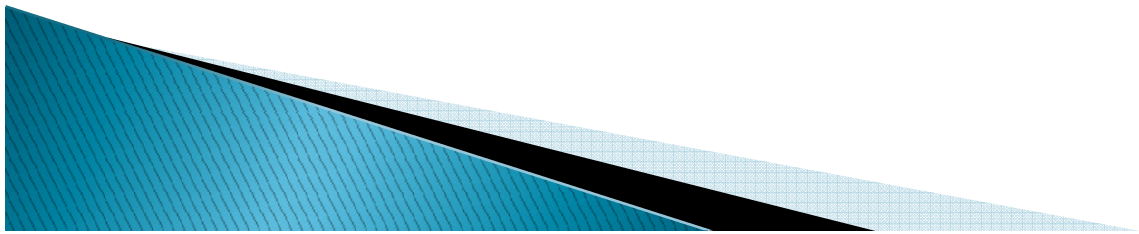
```
class Rectangle
{
int length, width;
}
```



Method declaration

Methods are necessary for manipulating the data contained in the class which are declared inside the body of the class after the declaration of instance variables.

```
type methodname(parameter list)
{
method body;
}
```



Creating Objects

An object in java is a block of memory that contains space to store all the instance variables

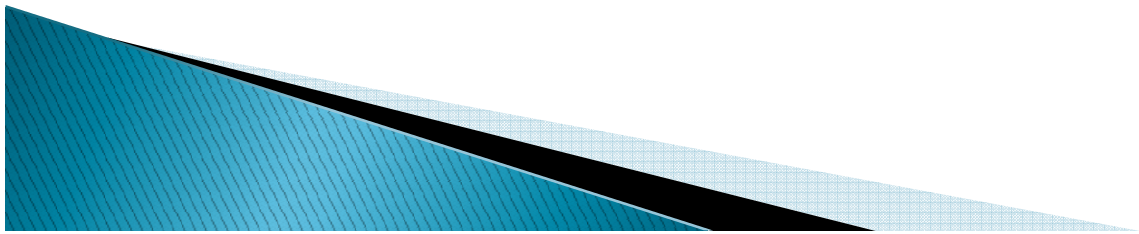
Polygon pol; //declare the object

Pol=new Polygon( );//instantiate the object

Both statements can be combined into one shown

Polygon pol=new Polygon( );

User can create any number of objects.



Rules in java

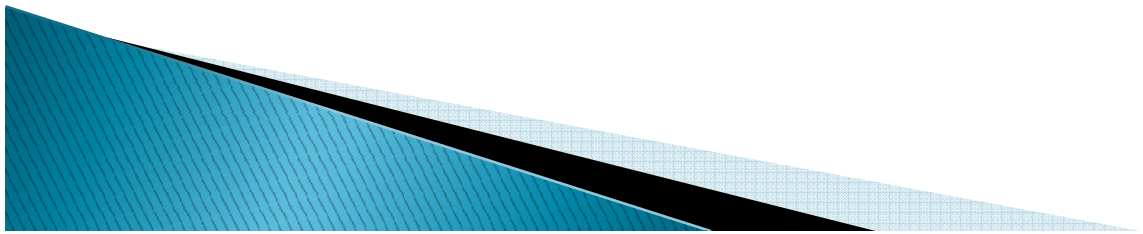
- The method will have the first character of first word in lower case, next every word's first character will be upper case.
  Eg – itemStateChanged();

- The package always starts with lower case.
  Eg –import java.awt.*;

- Java System Classes will start with upper case.
  Eg – Thread, Applet, Checkbox, Button etc.

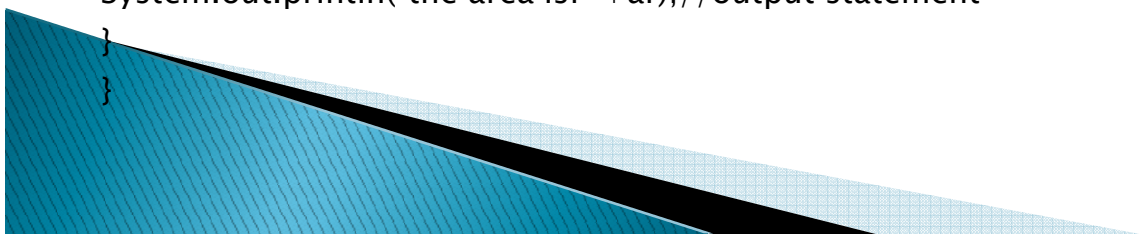
- The file name should be the name of the main class with ".java" extension.

- Float variables are dealt seperately.
  float a=23.5F; // then it works.

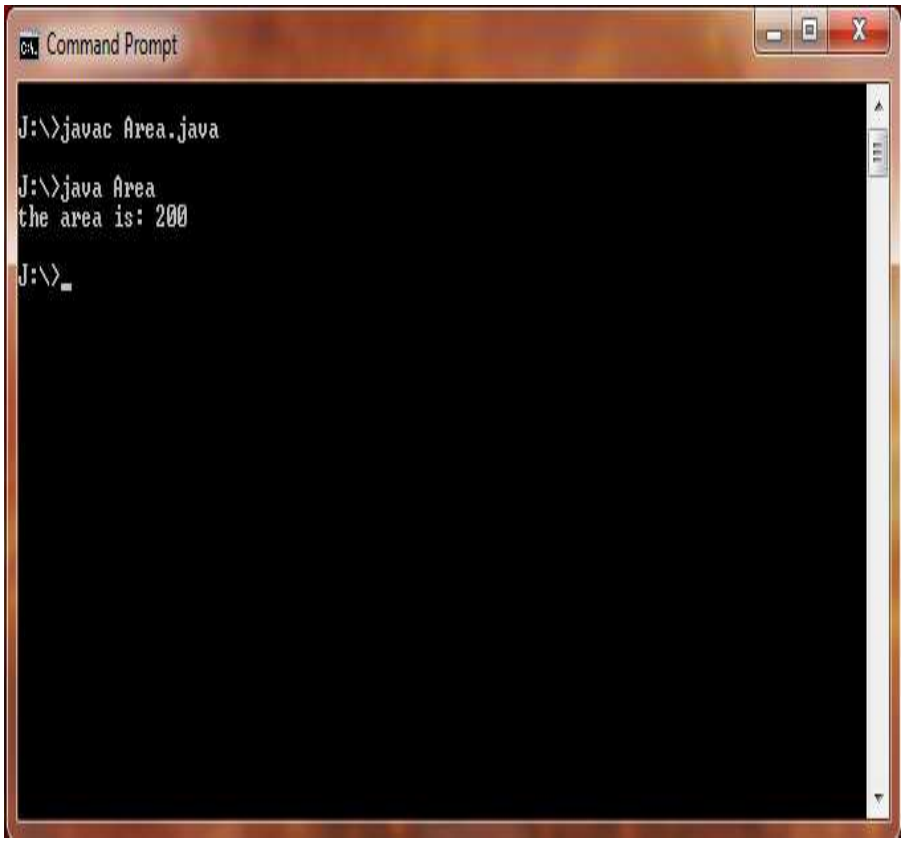


Sample program

```
class Polygon
{
int length,width;//fields declaration
Polygon(int x,int y)//constructor with two parameters
{
length=x;
width=y;
}
int showArea( )//method definition
{
return(length*width);
}
}
class Area//main class
{
public static void main(String args[])
{
Polygon p;//creating an object
p=new Polygon(10,20);//allocating memory to the object
int ar=p.showArea();//accessing the method with the object
System.out.println("the area is: "+ar);//output statement
}
}
```



Output



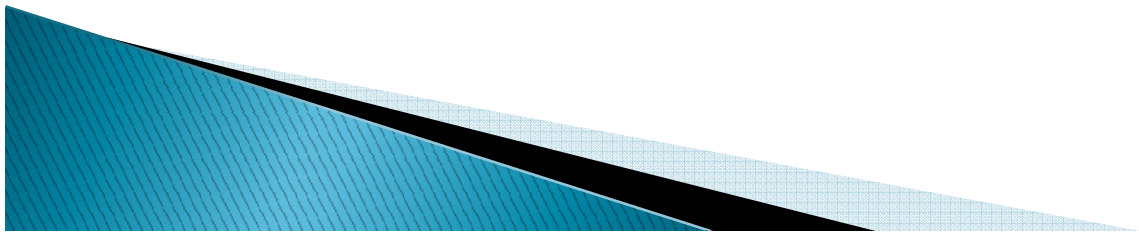
Concept of over loading

Overloading is the ability to define more than one method with the same name in a class. The compiler is able to distinguish between the methods because of their method signatures.

There are two types of over loading:

1.Method over loading

2.Constructor over loading

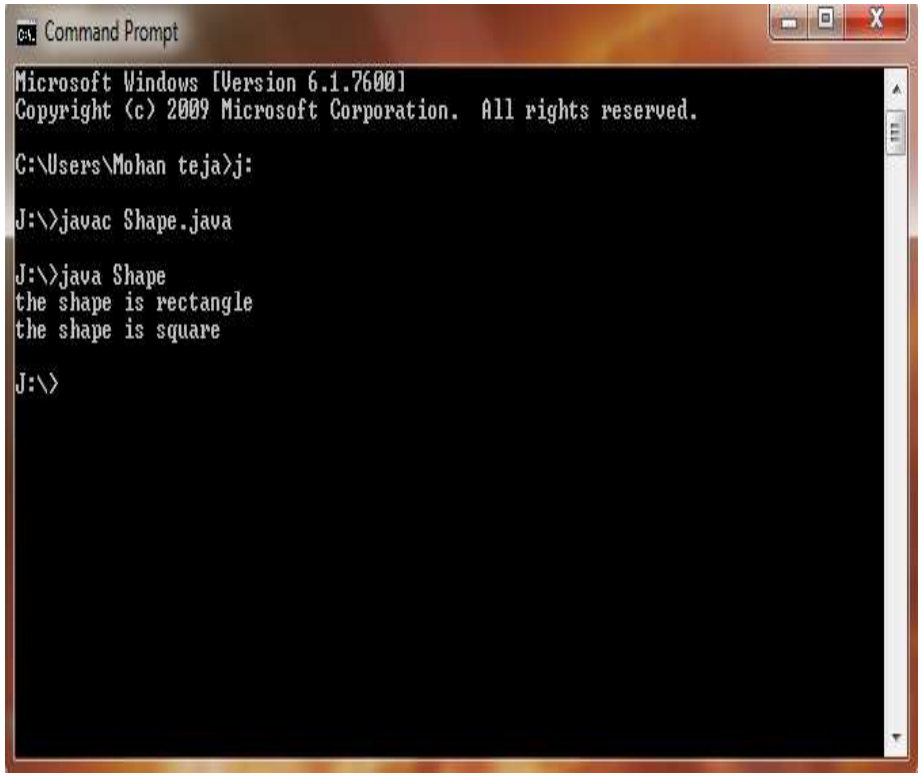


Constructor over loading

Having more than one constructor is called constructor over loading.

```
class Polygon
{
int length,width;//fields declaration
Polygon(int x,int y)//constructor with two parameters
{
length=x;
width=y;
System.out.println("the shape is rectangle");
}
Polygon(int x)
{
length=width=x;
System.out.println("the shape is square");
}
}
class Shape//main class
{
public static void main(String args[])
{
Polygon p;//creating an object
p=new Polygon(10,20);//this invokes the constructor with two parameters
Polygon q;//creating one more object
q=new Polygon(15);//this invokes the constructor with one parameter
}}
```

Output


Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Mohan teja>j:
J:\>javac Shape.java
J:\>java Shape
the shape is rectangle
the shape is square
J:\>

Method over loading

Having more than one method with the same name is called method over loading.

```
class Area
{
int length,width;//fields declaration
void showArea(int x,int y)//method definition with two parameters
{
length=x;
width=y;
System.out.println("the shape is rectangle with area: "+(length*width));
}
void showArea(int x)//method definition with one parameter
{
length=width=x;
System.out.println("the shape is square with area: "+(length*width));
}
}
class areaShape//main class
{
public static void main(String args[])
{
Area p;//creating an object
p=new Area( );
p.showArea(5);//this invokes the function with one parameter
p.showArea(4,5);//this invokes the function with two parameters
}
}
```

Output


Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Mohan teja>j:
J:\>javac areaShape.java
J:\>java areaShape
the shape is square with area: 25
the shape is rectangle with area: 20
J:\>

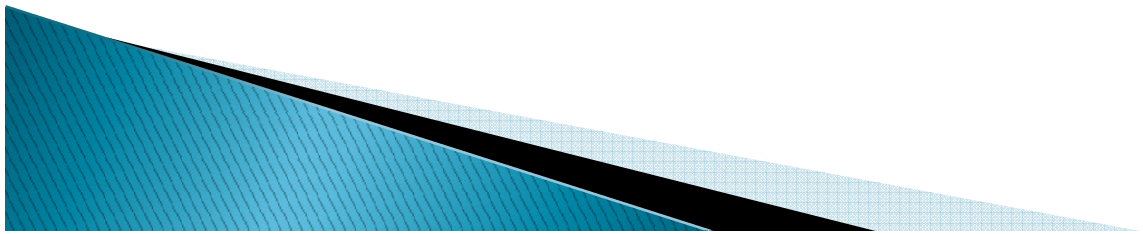
Static members

Static methods –

The methods which can be invoked with the class name only, they don't need an object.

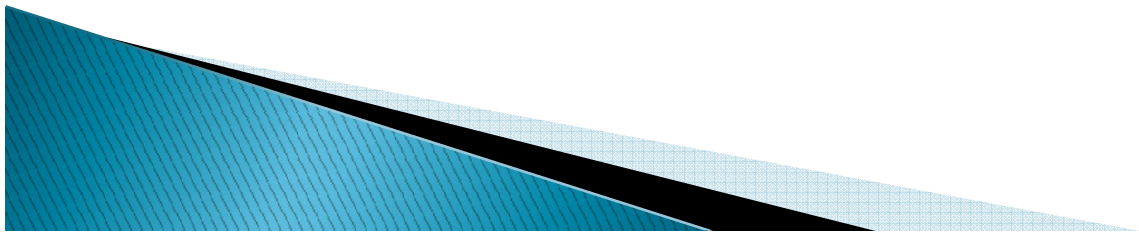
Static variables –

The variables which can be used with the class name only ,they don't need an object for their reference.

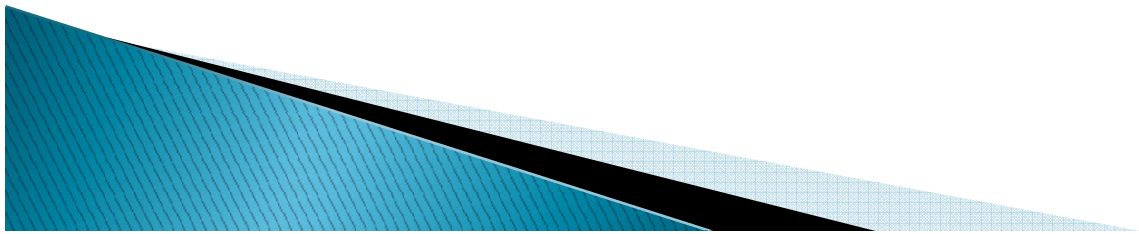


Sample program on static members

```
class myclass
{
static int x;
static void show()
{
System.out.println("i am a static function look carefully i dont need a
   object to be called, my class is enough to call me ");
}

}
```



```
class mainclass//main class
{
 public static void main(String args[])
 {
  myclass.show();// no need of object to call the static    method
  myclass.x=100;//even the data can be acessed with the class
    name
  System.out.println("static value"+myclass.x);
 }
 }
```



Output


Command Prompt
J:\>javac mainclass.java
J:\>java mainclass
i am a static function look carefully i dont need a object to be called, my clas
s is enough to call me
static value100
J:\>

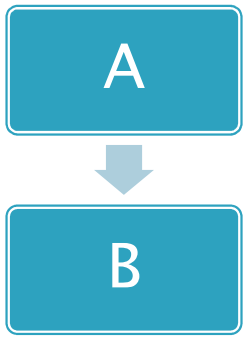
Inheritance

The mechanism of deriving the properties(methods, variables) to a new class from an old one is called inheritance.

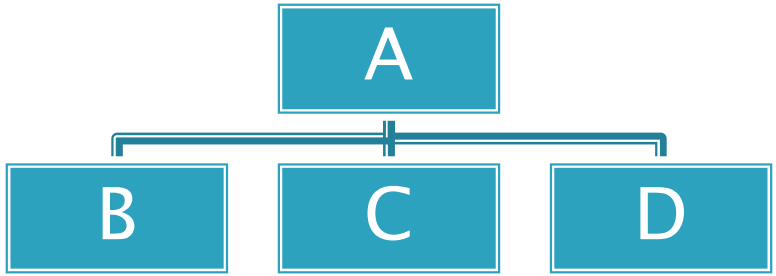
The old class is known as the base class or super class or parent class and the new one is called the sub class or derived class or child class.

Inheritance has so many forms:

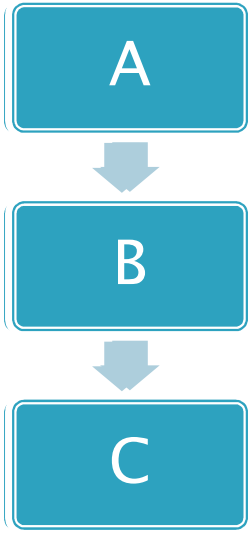
1.Single inheritance



Hierarchial inheritance:



Multilevel inheritance:



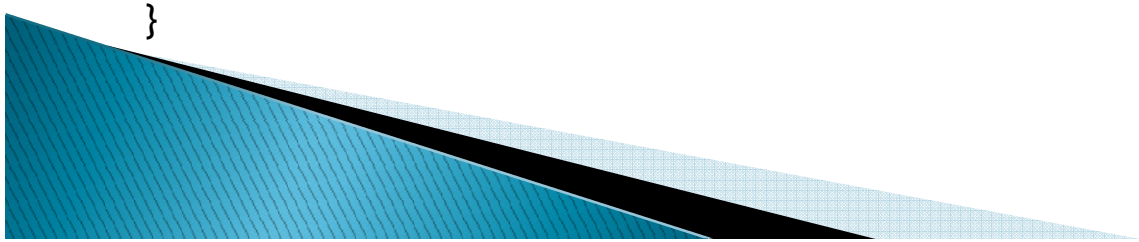
Syntax to define a sub class

class subclassname **extends** superclassname
{
Variable declaration;
Method declaration;
}

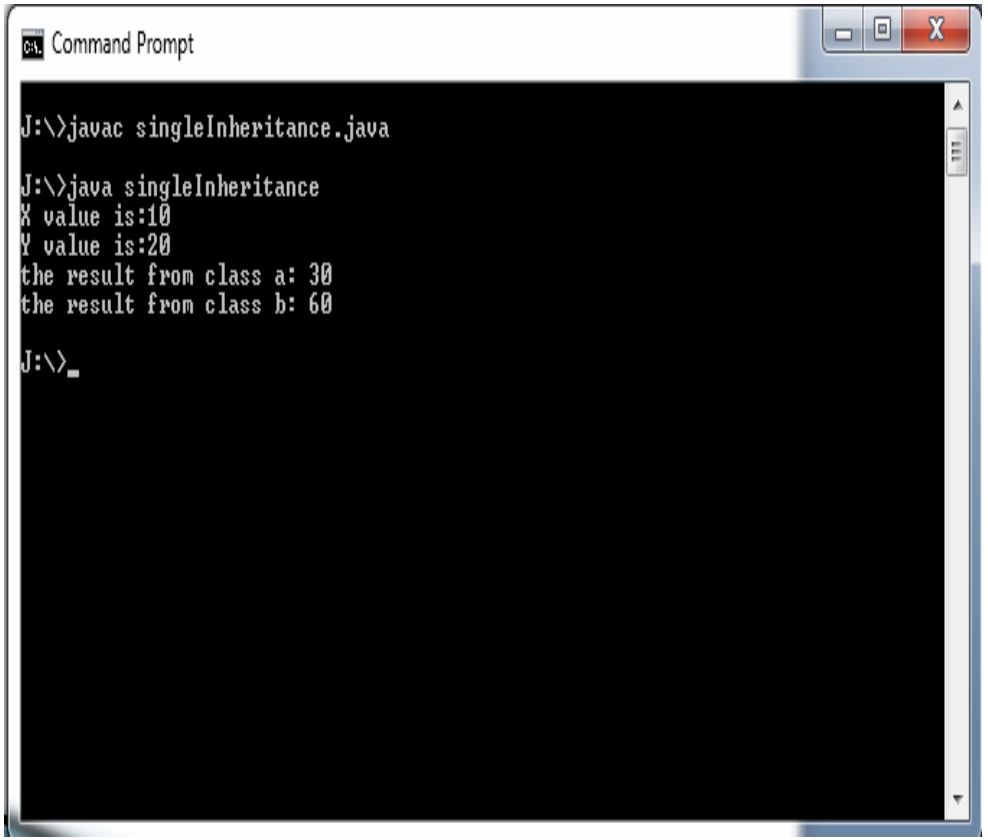
Sample Program on single inheritance

```
class A
{
  int x,y;
  A(int p,int q)
  {
  x=p;
  y=q;
  }
  int add2()
  {
    System.out.println("X value is:"+x+"\nY value is:"+y);
    return (x+y);
  }
}
```

```
class B extends A
{
  int z;
  B(int p,int q,int r)
  {
  super(p,q);
  z=r;
  }
  int add3()
  {
    return(x+y+z);
  }
}
class singleInheritance
{
  public static void main(String args[])
  {
  B ob=new B(10,20,30);
  int b=ob.add3();
  int a=ob.add2();
  System.out.println("the result from class a: "+a);
  System.out.println("the result from class b: "+b);
  }
}
```

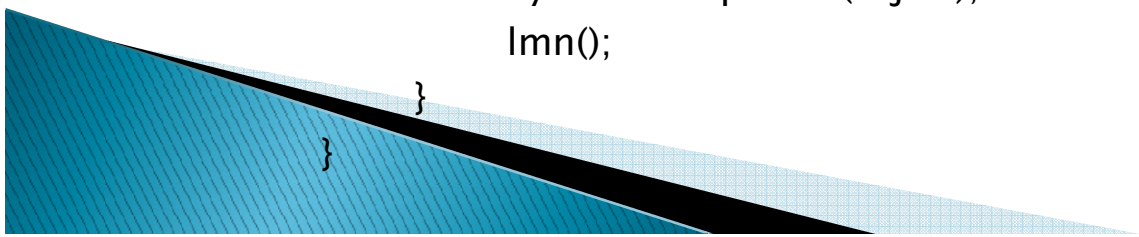


Output


Command Prompt
J:\>javac singleInheritance.java
J:\>java singleInheritance
X value is:10
Y value is:20
the result from class a: 30
the result from class b: 60
J:\>

Sample program on Multilevel Inheritance

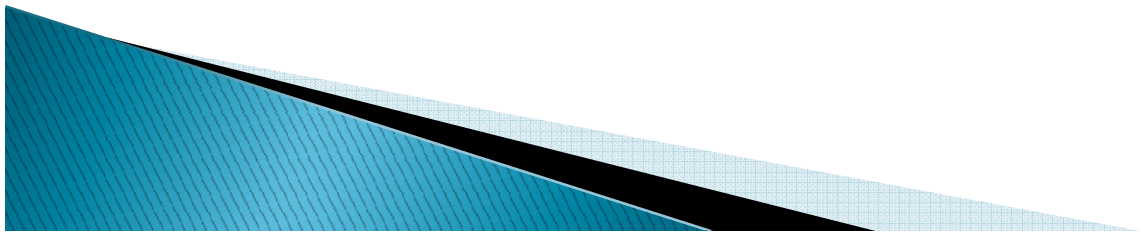
```
class A
{
        int i=10,j=20;
        void xyz()
        {
                System.out.println(i+j);
        }
        void lmn()
        {
                System.out.println("lmn");
        }
}
class B extends A
{
        int k=40,l=45;
        void abc()
        {
                System.out.println(i+j+k);
                lmn();
        }
}
```



```
class C extends B
{
}
class Inherit
{
        public static void main(String args[])
        {
        C a1= new C();
        a1.abc();
        }
}
```

Output:

```
J:\>javac Inherit.java
J:\>java Inherit
70
lmn
```

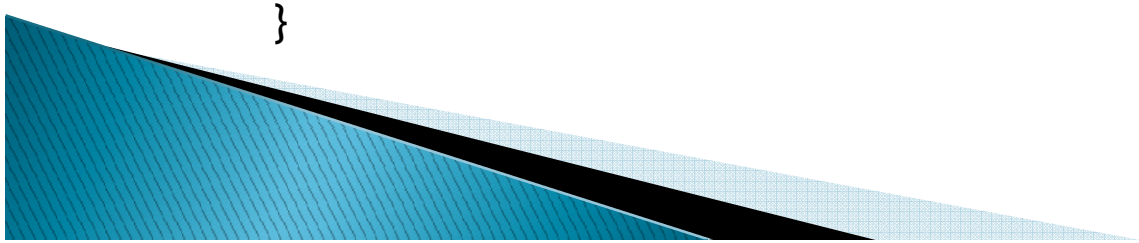


Method Overriding

A method is said to be overridden when one is in parent class and another is in child class with the same name, same return type, same parameter.

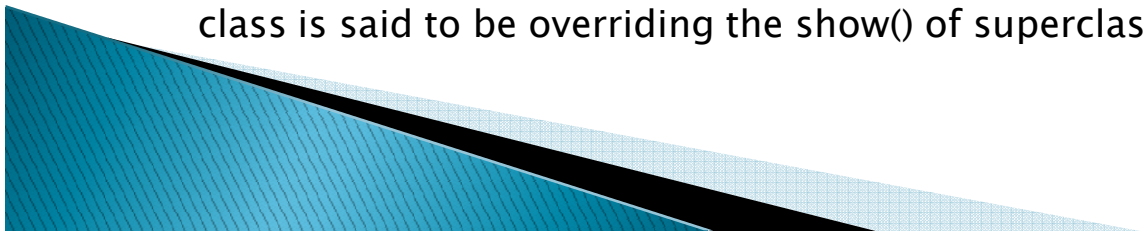
Sample Program:

```
class A
{
int i, j;
A(int a, int b)
{
i = a;
j = b;
}
void show()  // display i and j
{
System.out.println("i and j: " + i + " " + j);
}
}
```

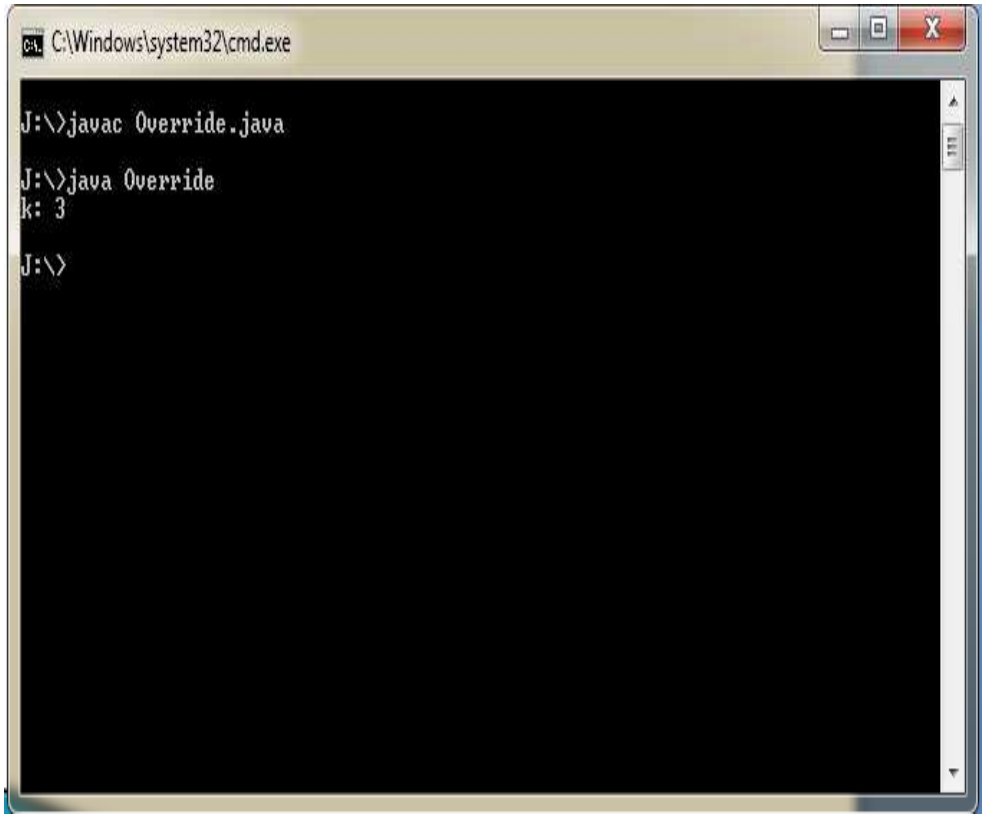


```
class B extends A
{
int k;
B(int a, int b, int c)
{
super(a, b);
k = c;
}
void show()  // display k – this overrides show() in A
{
System.out.println("k: " + k);
}
}
class Override {
public static void main(String args[]) {
B subOb = new B(1, 2, 3);
subOb.show(); // this calls show() in B
} }
```

Note - The show() from the subclass is invoked instead of superclass. Hence show() of sub class is said to be overriding the show() of superclass.



Output


C:\Windows\system32\cmd.exe
J:\>javac Override.java
J:\>java Override
k: 3
J:\>

Importance of Final

Final Variable – It acts as a constant value.

eg – static final int a=10; //this works as a constant

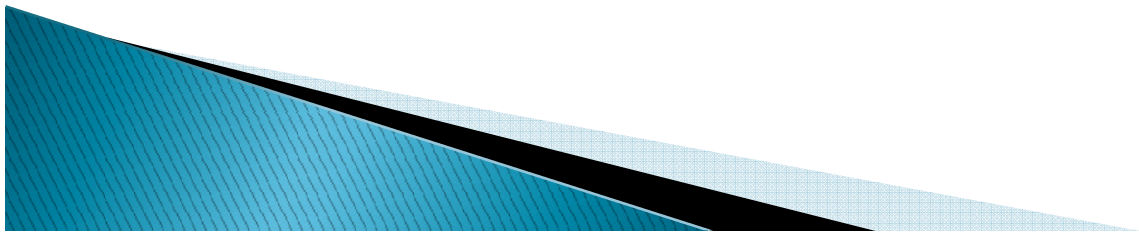
Final method – The final method prevents Method Overriding.

eg – final void sum();// will prevent Overriding

Final class – It prevents inheritance further.

Eg – final class A { --} // then no class can inherit further from A.

Finalize() – This method performs the process of Garbage collection. The objects creates Memory and that addresses will be DeReferenced by this method.

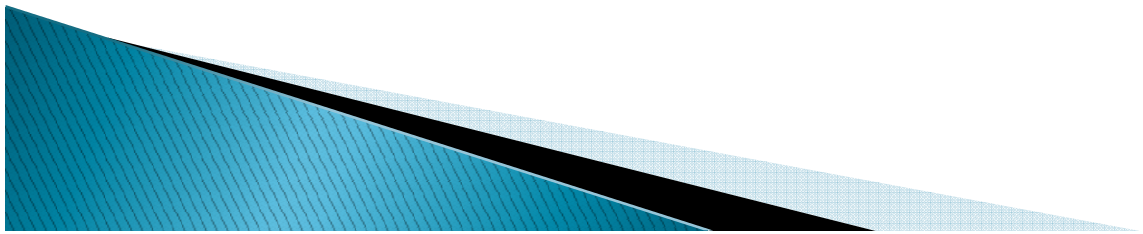


Abstract Methods and classes

An abstract class is a class that is declared abstract it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed. Overriding is done by using modifier keyword abstract in the method definition.

Syntax:

```
abstract class Shape
{
  ……….
  ……….
  abstract void draw();
  ……….
  ……….
}
```



While using abstract classes, the user must satisfy the following conditions. We cannot use abstract classes to instantiate objects directly. For example,

Shape s=new Shape( )

Is illegal because shape is an abstract class.

The abstract methods of an abstract class must be defined in its subclass.

We cannot declare abstract constructors or abstract static methods.

Note : The Abstract methods will be generally Declared and its definition will be in its subclass and the class having Abstract methods are Abstract classes.

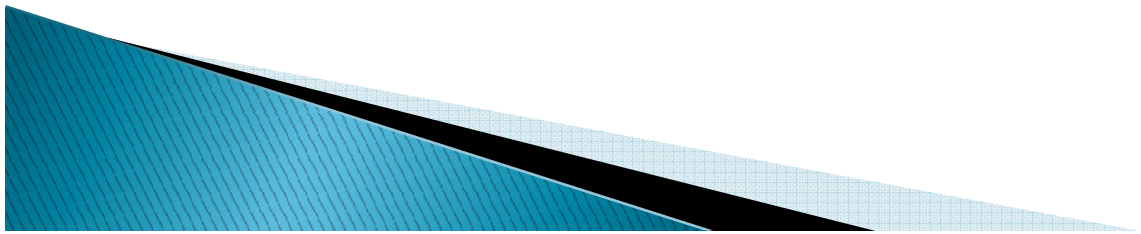
Sample program:

```
abstract class A {
abstract void callme();
// concrete methods are still allowed in abstract classes
void callmetoo() {
System.out.println("This is a concrete method.");
}
}
```

```
class B extends A {
void callme() {
System.out.println("B's implementation of callme.");
}
}
class AbstractDemo {
public static void main(String args[]) {
B b = new B();
b.callme();
b.callmetoo();
}
}
```

Output:

```
B's implementation of callme.
This is a concrete method.
```



Visibility Control

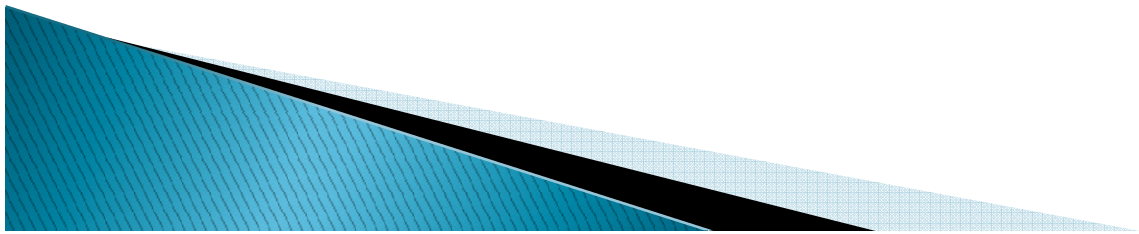
It is necessary to restrict the access to certain variables and methods from outside the class.

This is achieved by applying visibility modifiers to the instance variables and methods, they are also called as access modifiers. Java provides three types of visibility modifiers:

public

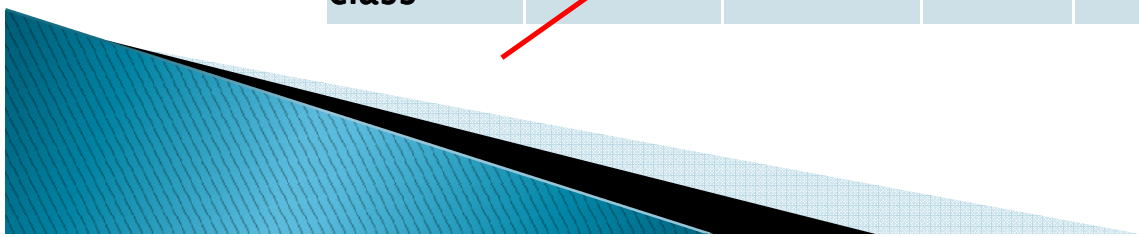
private

protected



Visibility of field in a class

	public	protected	default	private protected	private
Same class	Yes	Yes	Yes	Yes	Yes
Same package sub class	Yes	Yes	Yes	Yes	No
Same package Non-sub class	Yes	Yes	Yes	No	No
Different package sub class	Yes	Yes	No	Yes	No
Different package Non-sub class	Yes	No	No	No	No

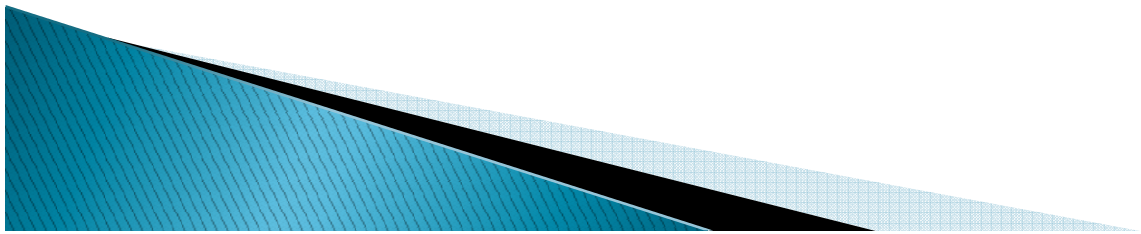


Exception Handling

Basically there are two types of errors.

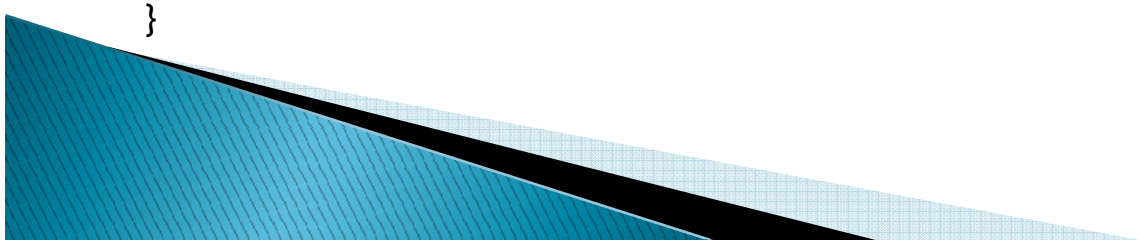
Compile Errors – Error due to the syntax
Eg- missing ; or { or (etc..

Runtime Error – Error due to the logic,.
Eg – 1) a/b where a=10, b=0;
2) stack overflow
3) Assigning one type of value to another type of variable.



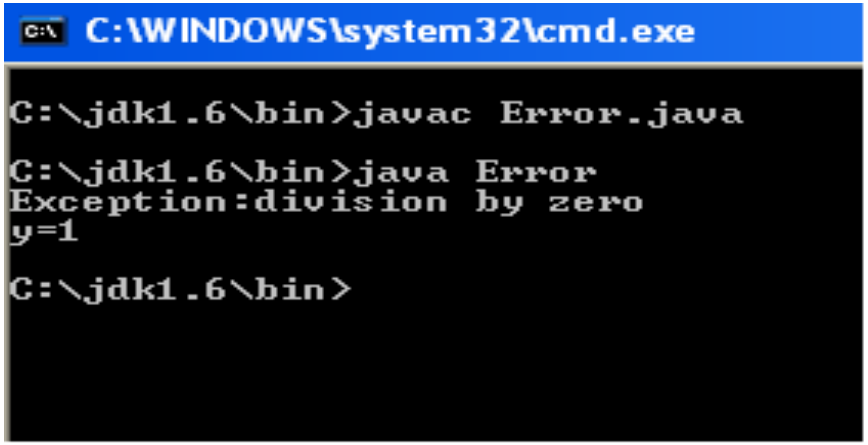
Sample program

```
class Error
{
public static void main(String args[])
{
int a=10;
int b=5,c=5;
int x,y;
try
{
x=a/(b-c);
}
catch(ArithmeticException e)
{
System.out.println("Exception:division by zero");
}
y=a/(b+c);
System.out.println("y="+y);
}
}
```



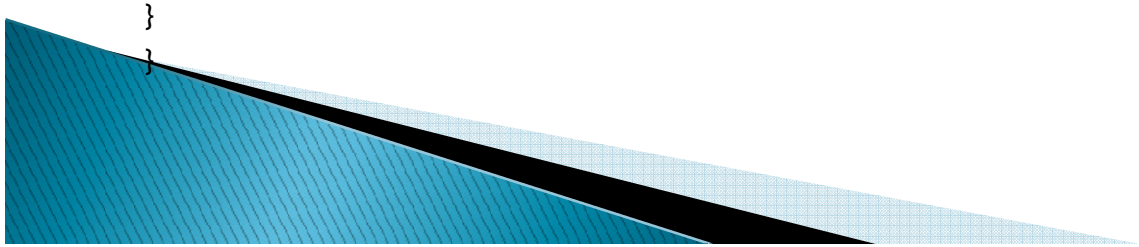
output

At the time of Execution, it goes to “try” block. Then as it encountered “1/0”, then the Java System Raises an Arithmetic Exception, and it throws the Exception to “Catch” block. And it executes that catch block and then it Procedes to the Next statement after catch block.



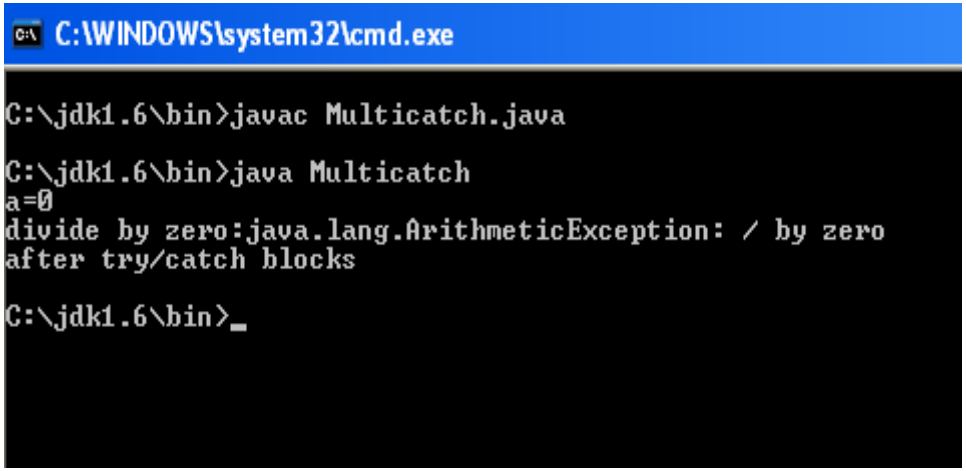
Multiple catch statements

```
class Multicatch
{
public static void main(String args[])
{
try
{
int a=args.length;
System.out.println("a="+a);
int b=42/a;
int c[]={1};
c[42]=99;
}
catch(ArithmeticException e)
{
System.out.println("divide by zero:"+e);
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("Array index oob:"+e);
}
System.out.println("after try/catch blocks");
}
}
```



output

Here from the try block, it encountered “1/0” and goes to Catch block of “Arithmetic Exception” and execute that block and from there the control goes to the next statement after all the catch blocks.



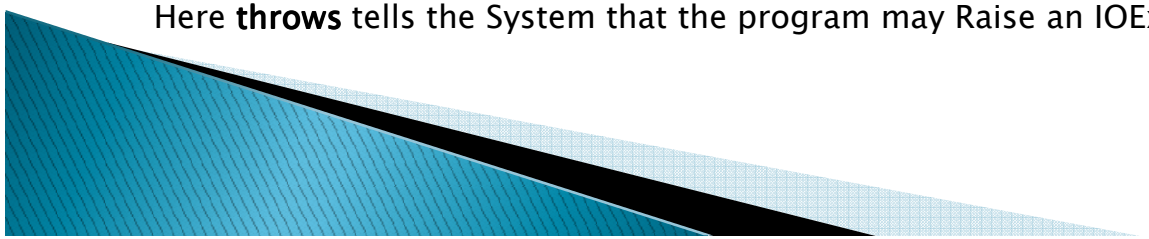
Throw, finally, Throws

```
class Finallydemo
{
static void procA()
{
try
{
System.out.println("inside procA's finally");
throw new RuntimeException("demo");
}
finally
{
System.out.println("procA's finally");
}
Public static void main(String args[]) throws IOException
{  }
}
```

Here **Throw** is to throw the Exception Manually by the User/ programmer

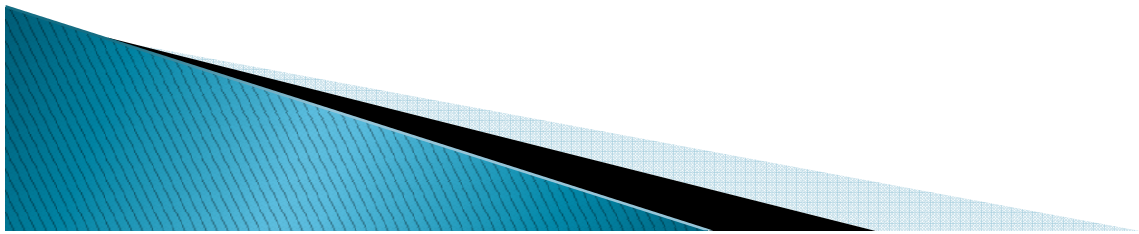
When **"Finally"** block is used, the control should come to the Finally block and then terminate from the program.

Here **throws** tells the System that the program may Raise an IOException.



Chapter 3

Interfaces, Package & Multithreaded programming



Interface

Using class concept, Multiple Inheritance is not possible in java. i.e.

Class A { --}

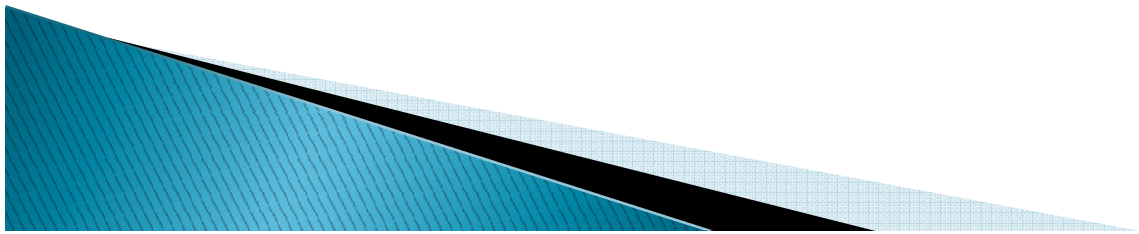
Class B { -- }

Class C extends A,B { -- } // this gives an Error which is not possible in java. But, with interface it works..

interface A { --}

interface B { -- }

interface C extends A,B { -- } // this is possible in java.



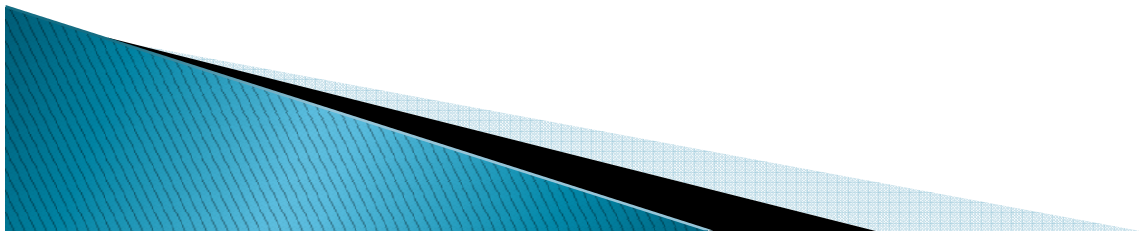
Interface definition

An interface is a group of related methods with empty bodies. A bicycle's behavior, if specified as an interface, might appear as follows:

Syntax:

```
interface Bicycle
{
void changeCadence(int newValue);
 void changeGear(int newValue);
void speedUp(int increment);
void applyBrakes(int decrement);
}
```

And the above method definitions will be in its immediate subclass.



Interface

1)Class A
{ --}
Class B
{ -- }
Class C extends A,B
{ -- }

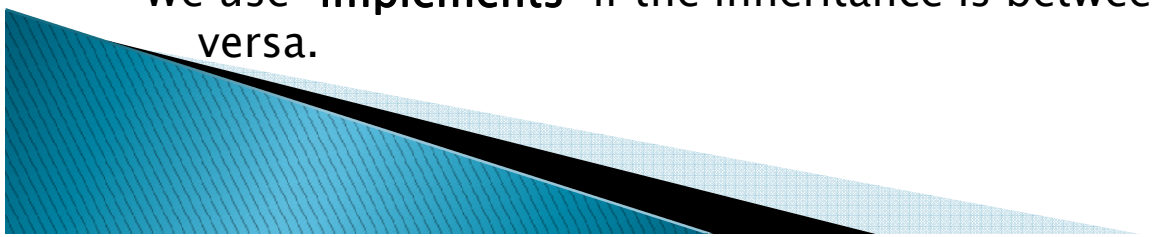
2)interface A
{ --}
interface B
{ -- }
interface C extends A,B
{ -- }

3)class A
{ --}
interface C implements A
{ --}

4) interface A
{ --}
Class B implements A
{ -- }

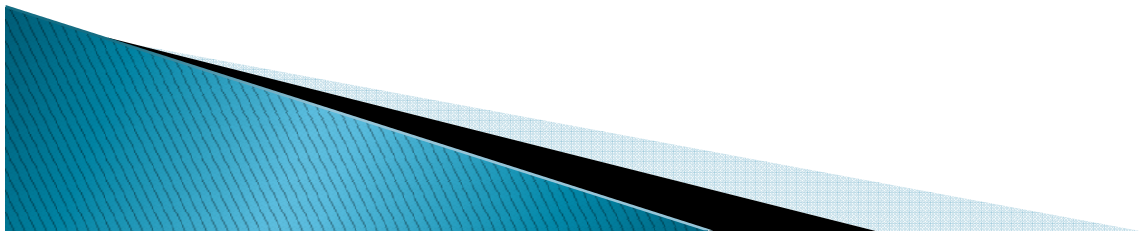
Note – we use "**Extends**" if all are classes/interfaces

We use "**implements**" if the inheritance is between class & interface or vice-versa.

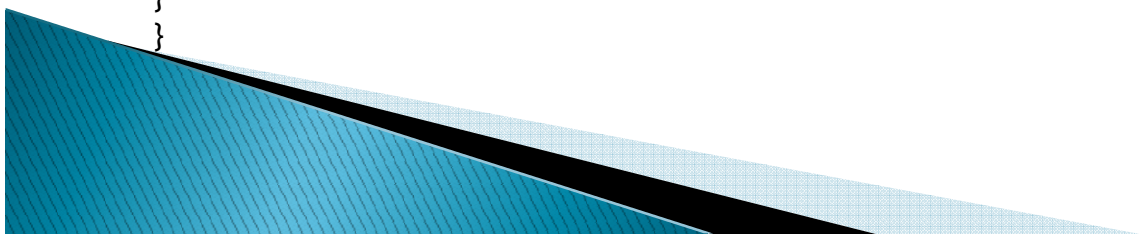


Example

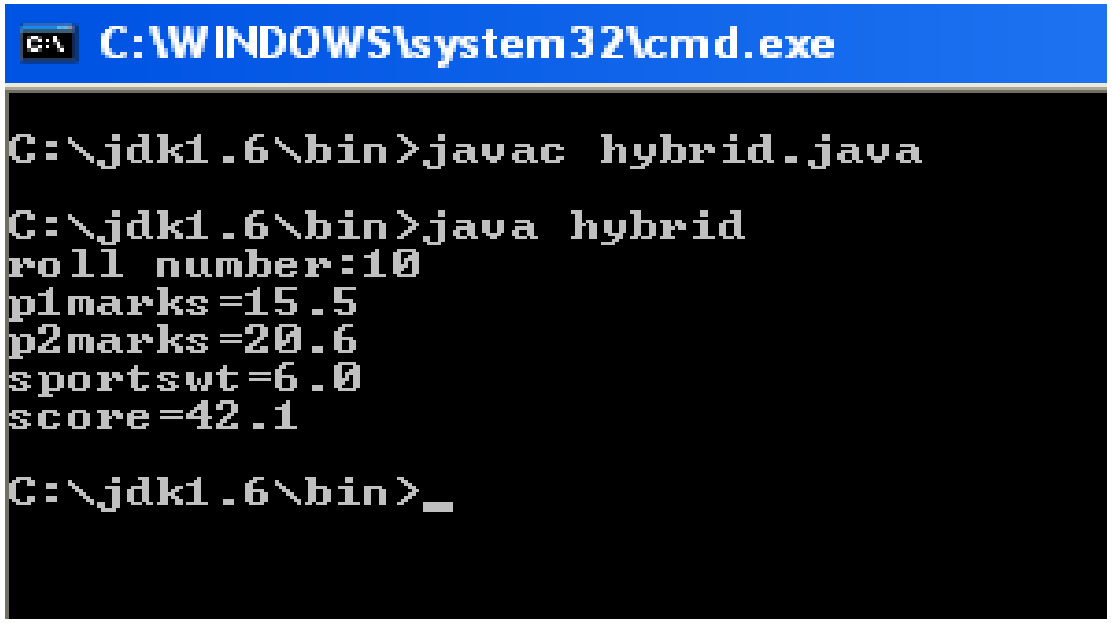
```
class student
{
int rno;
void getno(int n)
{
rno=n;
}
void putno()
{
System.out.println("roll number:"+rno);
}
}
class test extends student
{
float p1,p2;
void getmarks(float m1,float m2)
{
p1=m1;
p2=m2;
}
void putmarks()
{
System.out.println("p1marks="+p1);
System.out.println("p2marks="+p2);
}
```



```
interface sports
{
float sportwt=6.0f;
void putwt();
}
class result extends test implements sports
{
float total;
public void putwt()
{
System.out.println("sportswt="+sportwt);
}
void display()
{
total=p1+p2+sportwt;
putno();
putmarks();
putwt();
System.out.println("score="+total);
}
}
class hybrid
{
public static void main(String args[])
{
result r=new result();
r.getno(10);
r.getmarks(15.5f,20.6f);
r.display();
}
}
```



output

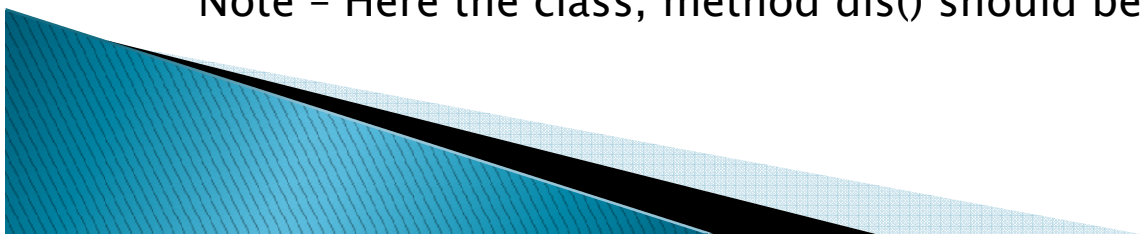

C:\WINDOWS\system32\cmd.exe
C:\jdk1.6\bin>javac hybrid.java
C:\jdk1.6\bin>java hybrid
roll number:10
p1marks=15.5
p2marks=20.6
sportswt=6.0
score=42.1
C:\jdk1.6\bin>_

Package

Creating a package – Package contains classes, classes contains methods. we can create our own package. All the Packages inherit from the Root package called “java”. That’s why we use import java.awt.*; Now we can create a package in the below way.

```
package testpackage;
public class tp
{
public void dis()
{
System.out.println("welcome to Applets");
}
}
```

Save it in a folder – D:\svg java\testpackage\tp.java
Compile the file – D:\svg java\testpackage> javac tp.java
Note – Here the class, method dis() should be public.



Importing a Package

```
import testpackage.*;
class tpm
{
public static void main(String args[])
{
tp ob=new tp();
ob.dis();
}
}
```

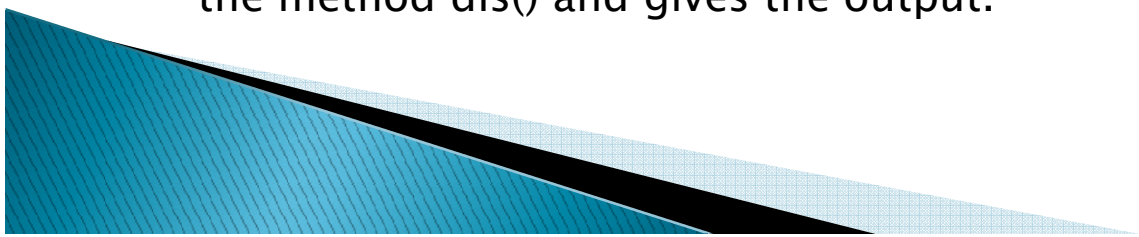
Save it in the folder - D:\svg java

i.e. one directory above D:\svg java\testpackage.

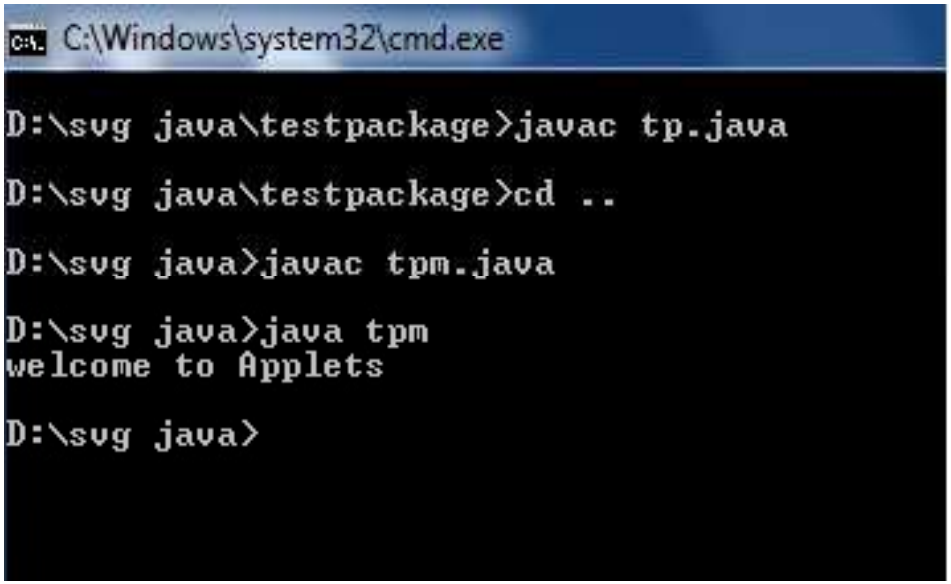
Compile the file - D:\svg java> javac tpm.java

Execute the file - D:\svg java> java tpm

From this location, it goes to D:\svg java\testpackage and import it, and it calls the method dis() and gives the output.



Output


C:\Windows\system32\cmd.exe
D:\svg java\testpackage>javac tp.java
D:\svg java\testpackage>cd ..
D:\svg java>javac tpm.java
D:\svg java>java tpm
welcome to Applets
D:\svg java>

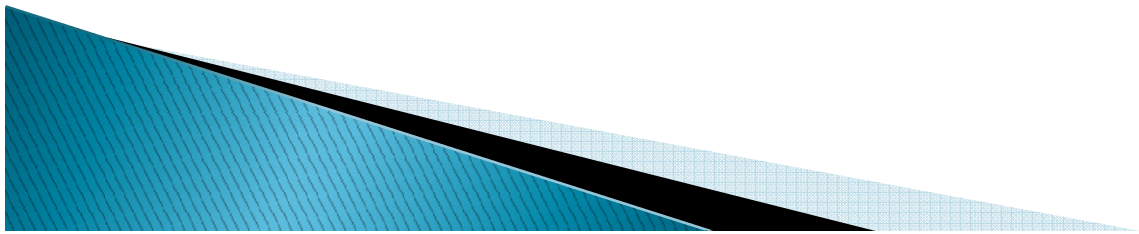
MultiThreading

Thread is a light weight process.

Running Multiple/more than one threads at a time is called Multi Threading.

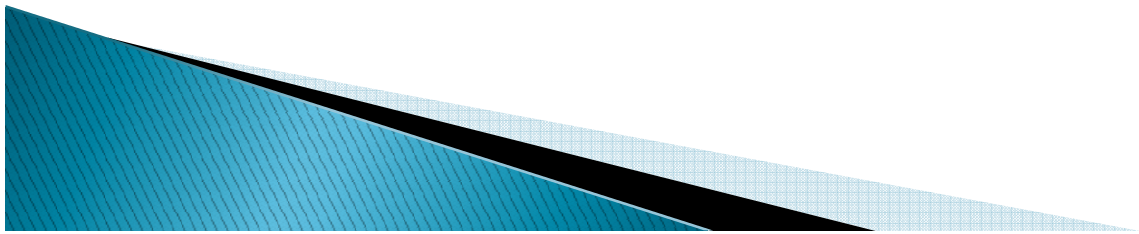
When the Processor is Running a Thread, it could switch to another Thread as it encounters I/O statements in the first Thread.

i.e. Processor works with GHz(10^9 instructions/sec) and I/O devices works with nearly 100 char/sec. So, when Processor encounters I/O statement, it will not wait, it switches to next Thread.



Sample program

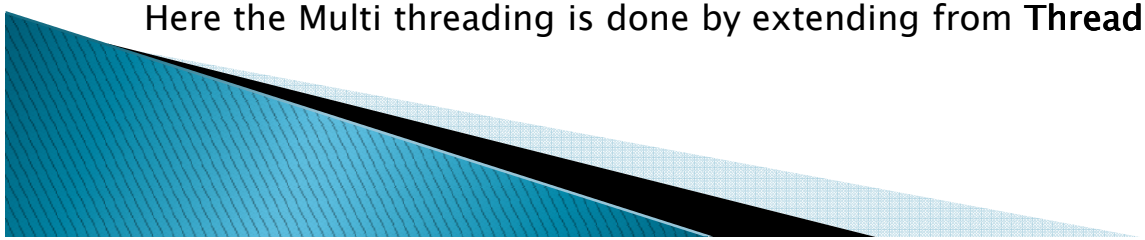
```
class A extends Thread
{
   public void run()
    {
       for(int i=1;i<5;i++)
        {
           System.out.println("\t from Thread A: i= "+i);
    }
              System.out.println("exit  from A");
            }
}
 class B extends Thread
  {
      public void run()
          {
             for(int j=1;j<=5;j++)
                {
                        System.out.println("\tfrom Thread B: j= "+j);
                }
                           System.out.println("exit from B");
              }
             }
```



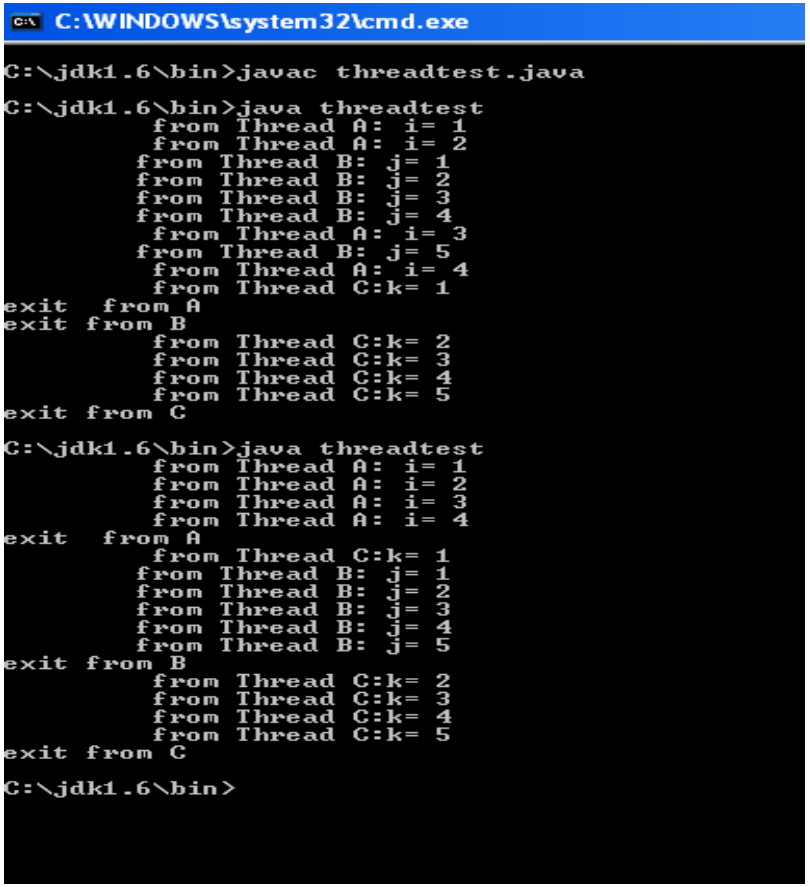
```
class C extends Thread
    {
          public void run()
             {
                for(int k=1;k<=5;k++)
                 {
                System.out.println("\t from Thread C:k= "+k);
                   }
                        System.out.println("exit from C");
                }
    }
  class threadtest
  {
  public static void main(String args[])
      {
          new A().start();
          new  B().start();
          new  C().start();
      }
   }
```

Note – if we Observe the Output of the program, we can find everytime the threads are executed differently. Because, it depends on the System conditions, and the Readiness of Threads .

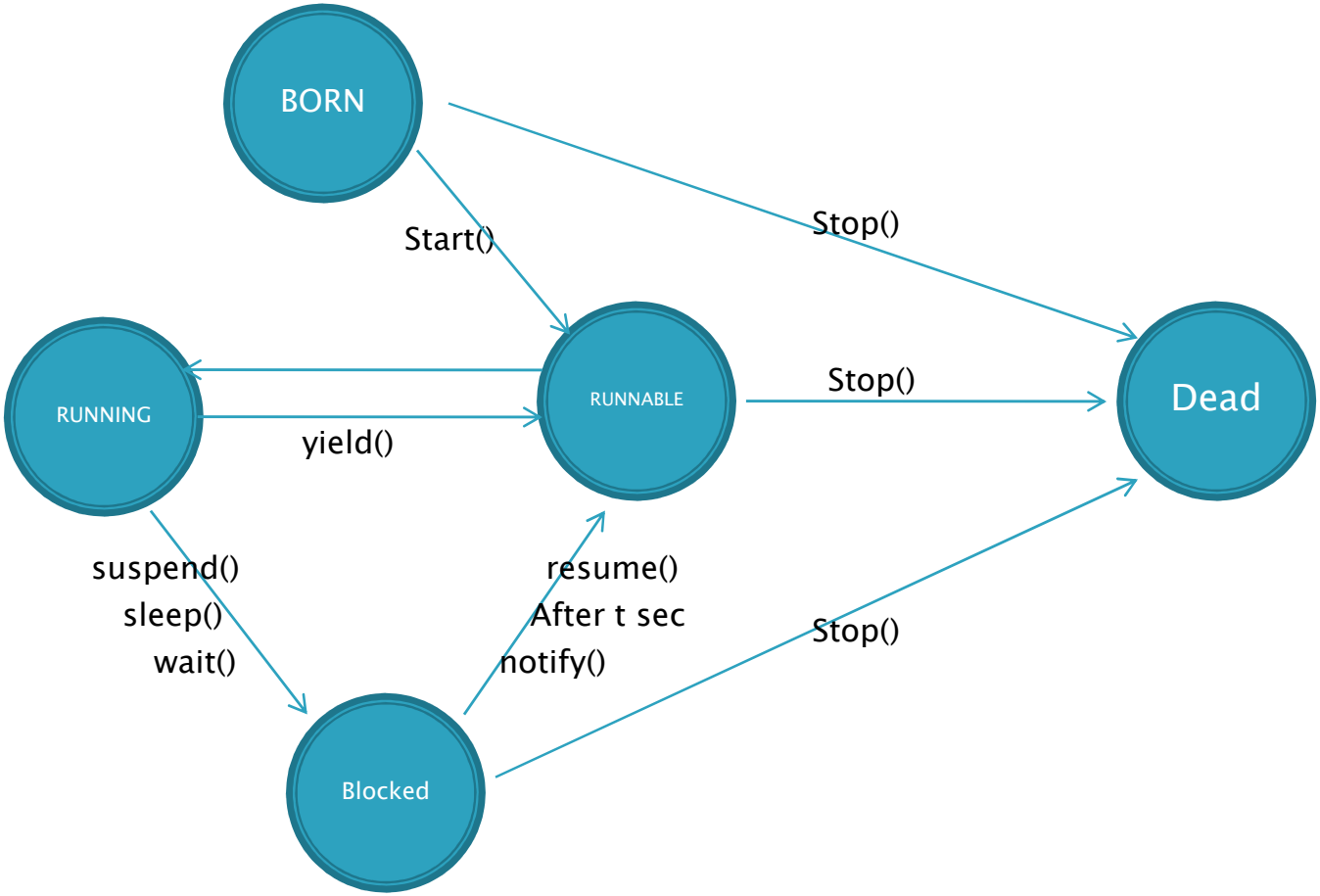
Here the Multi threading is done by extending from **Thread** class.



Output


C:\WINDOWS\system32\cmd.exe
C:\jdk1.6\bin>javac threadtest.java
C:\jdk1.6\bin>java threadtest
from Thread A: i= 1
from Thread A: i= 2
from Thread B: j= 1
from Thread B: j= 2
from Thread B: j= 3
from Thread B: j= 4
from Thread A: i= 3
from Thread B: j= 5
from Thread A: i= 4
from Thread C:k= 1
exit from A
exit from B
from Thread C:k= 2
from Thread C:k= 3
from Thread C:k= 4
from Thread C:k= 5
exit from C
C:\jdk1.6\bin>java threadtest
from Thread A: i= 1
from Thread A: i= 2
from Thread A: i= 3
from Thread A: i= 4
exit from A
from Thread C:k= 1
from Thread B: j= 1
from Thread B: j= 2
from Thread B: j= 3
from Thread B: j= 4
from Thread B: j= 5
exit from B
from Thread C:k= 2
from Thread C:k= 3
from Thread C:k= 4
from Thread C:k= 5
exit from C
C:\jdk1.6\bin>

Thread life cycle



Thread life cycle

When the Thread is created, Thread goes to Born state.

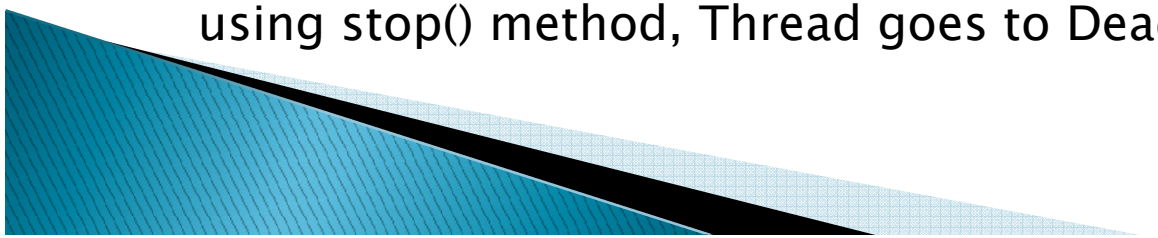
All the Born Threads will be Ready to have the Processor and will be in Runnable state.

From Runnable, it goes to Running state.

From Running state, if yield() method is called it goes to Runnable state.

The Running Thread may go to Blocked state if the methods suspend(), sleep(), wait() is called. And the Blocked Threads will go to Runnable state if the methods Resume(), after T sec, notify() is called Respectively.

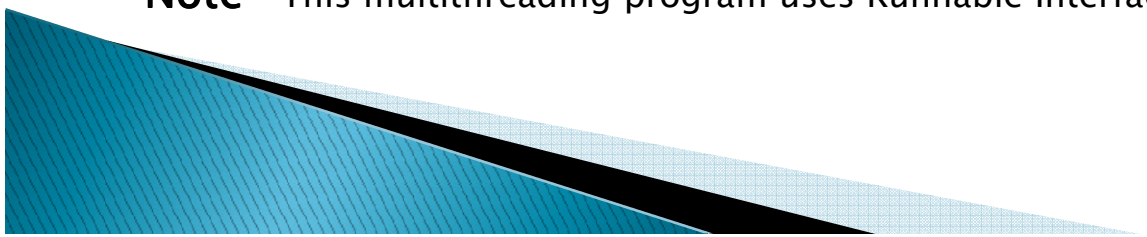
using stop() method, Thread goes to Dead state.



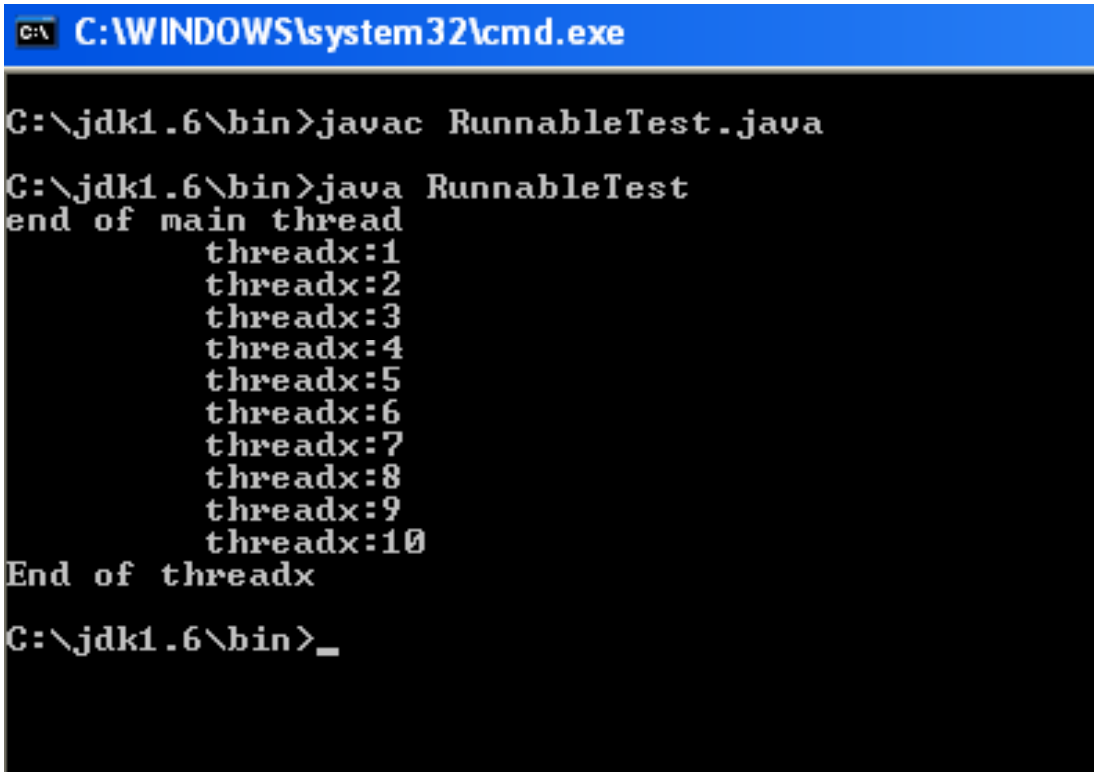
Multithreading using Runnable interface

```
class x implements Runnable
{
public void run()
{
for(int i=1;i<=10;i++)
{
System.out.println("\t threadx:"+i);
}
System.out.println("End of threadx");
}
}
class RunnableTest
{
 public static void main(String args[])
{
x runnable=new x();
Thread threadx=new Thread(runnable);
threadx.start();
System.out.println("end of main thread");
}
}
```

Note – This multithreading program uses Runnable Interface.

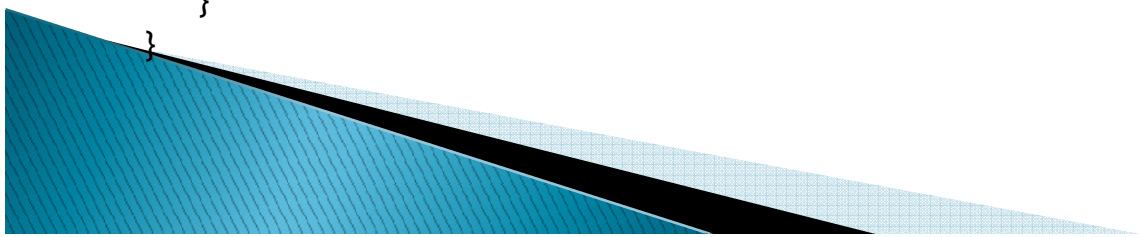


Output

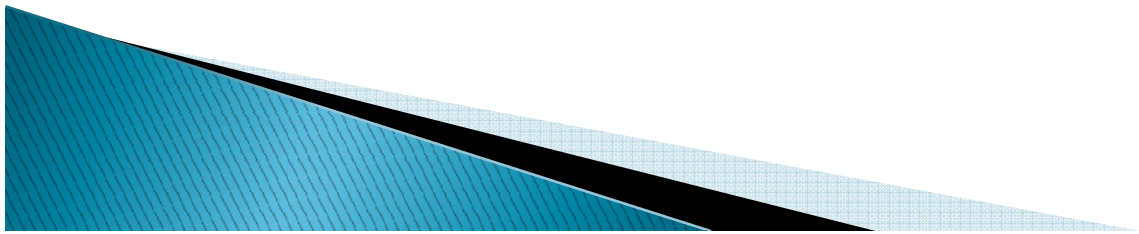

C:\WINDOWS\system32\cmd.exe
C:\jdk1.6\bin>javac RunnableTest.java
C:\jdk1.6\bin>java RunnableTest
end of main thread
threadx:1
threadx:2
threadx:3
threadx:4
threadx:5
threadx:6
threadx:7
threadx:8
threadx:9
threadx:10
End of threadx
C:\jdk1.6\bin>

Thread Priority

```
class A extends Thread
{
    public void run()
    {
            System.out.println("ThreadA Started");
            for(int i=0;i<5;i++)
            {
                    System.out.println("\tFrom Thread A:i="+i);
            }
            System.out.println("Exit from A");
    }
}
class B extends Thread
{
    public void run()
    {
            System.out.println("ThreadB Started");
            for(int j=0;j<5;j++)
            {
                    System.out.println("\tFrom Thread B:j="+j);
            }
            System.out.println("Exit from B");
    }
}
```



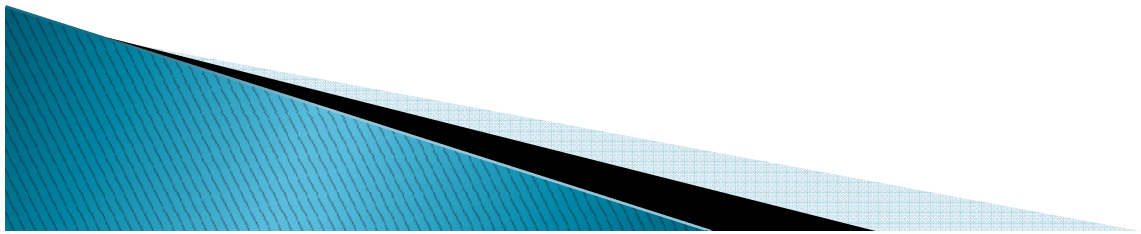
```
class C extends Thread
{
    public void run()
    {
            System.out.println("ThreadC Started");
            for(int k=0;k<5;k++)
            {
                    System.out.println("\tFrom Thread C:k="+k);
            }
            System.out.println("Exit from C");
    }
}
```



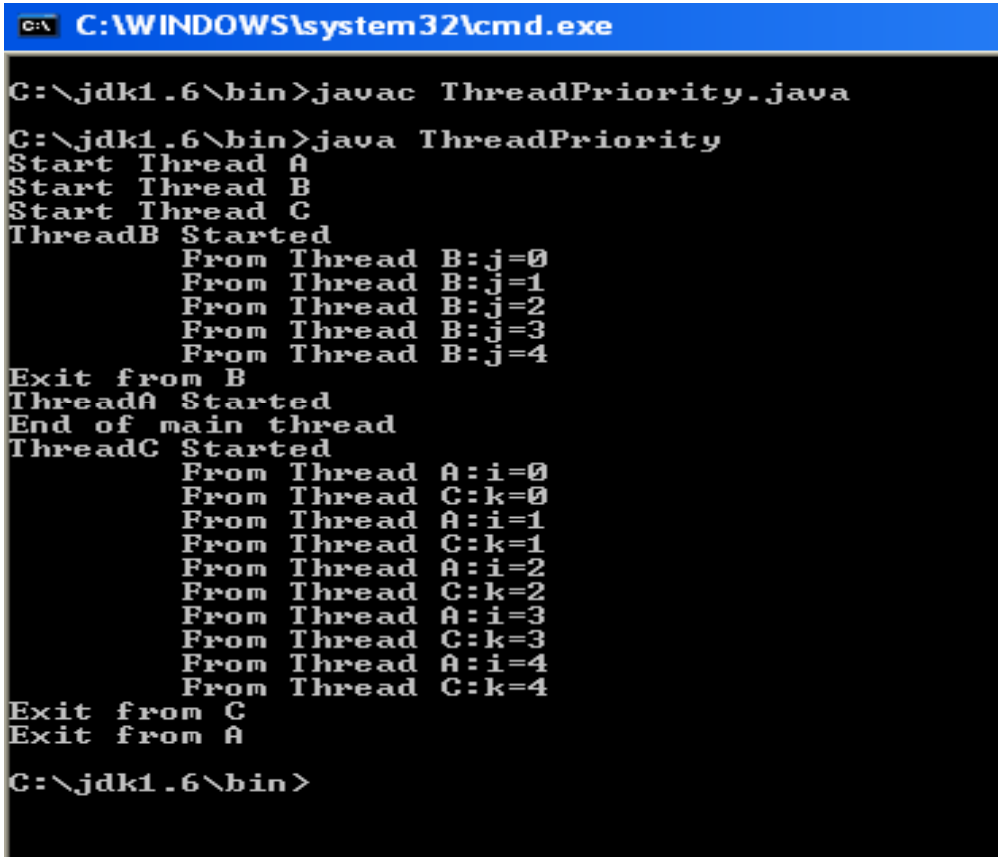
```
class ThreadPriority
{
    public static void main(String args[])
    {
            A threadA=new A();
            B threadB=new B();
            C threadC=new C();
            threadC.setPriority(Thread.MAX_PRIORITY);
            threadB.setPriority(threadA.getPriority()+1);
            threadA.setPriority(Thread.MIN_PRIORITY);
            System.out.println("Start Thread A");
            threadA.start();
            System.out.println("Start Thread B");
            threadB.start();
            System.out.println("Start Thread C");
            threadC.start();
            System.out.println("End of main thread");
    }
}
```

Note – if we observe in the Program, Thread C got the Max priority as 10, Thread A with 1, Thread B with priority equal to Thread A's priority + 1 .. i.e. 2.

We have started the Threads A,B,C Respectively. Initially A starts. But B will come and Preempt (stop) A. So, Threads will be executed according to their priorities.

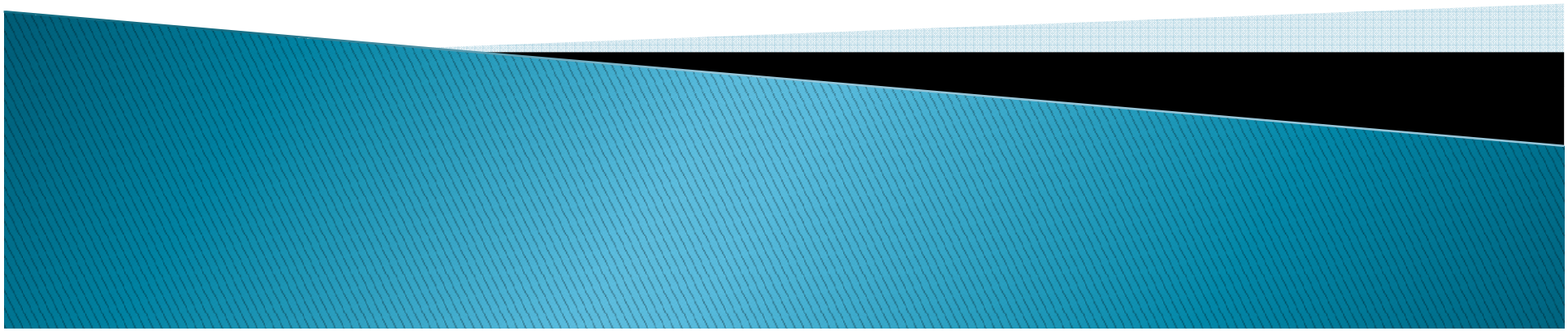


Output


C:\WINDOWS\system32\cmd.exe
C:\jdk1.6\bin>javac ThreadPriority.java
C:\jdk1.6\bin>java ThreadPriority
Start Thread A
Start Thread B
Start Thread C
ThreadB Started
From Thread B:j=0
From Thread B:j=1
From Thread B:j=2
From Thread B:j=3
From Thread B:j=4
Exit from B
ThreadA Started
End of main thread
ThreadC Started
From Thread A:i=0
From Thread C:k=0
From Thread A:i=1
From Thread C:k=1
From Thread A:i=2
From Thread C:k=2
From Thread A:i=3
From Thread C:k=3
From Thread A:i=4
From Thread C:k=4
Exit from C
Exit from A
C:\jdk1.6\bin>

Chapter 4

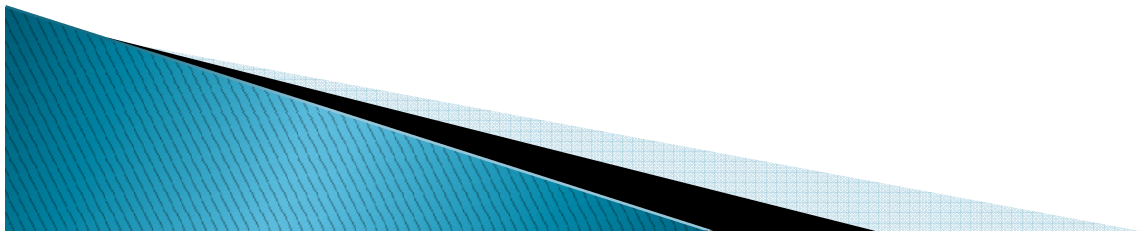
Applet Programming



Introduction

Applets are small java programs that are primarily used in internet computing.

Applets can perform Arithmetic operations, display graphics, play sounds, accept user input and play Games etc..

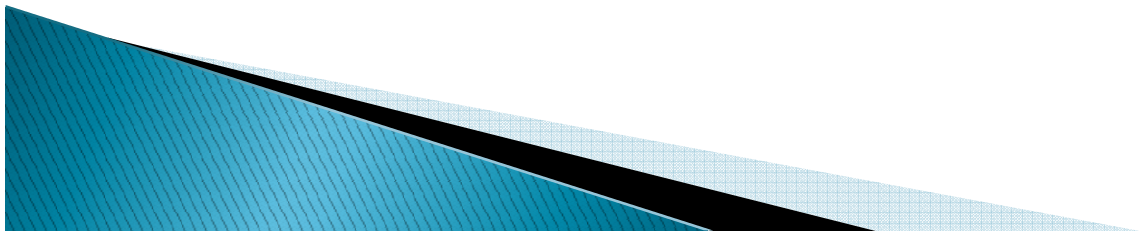


Applet source code

```
import java.awt.*;
import java.applet.*;
public class ap extends Applet
{
public void paint(Graphics g)
{
g.drawString("welcome to applets",10,100);
}
}
```

1. Save this file at some location say d:\ as- “ap.java”.
2. Compile the file – javac ap.java

Now create the Applet html file .



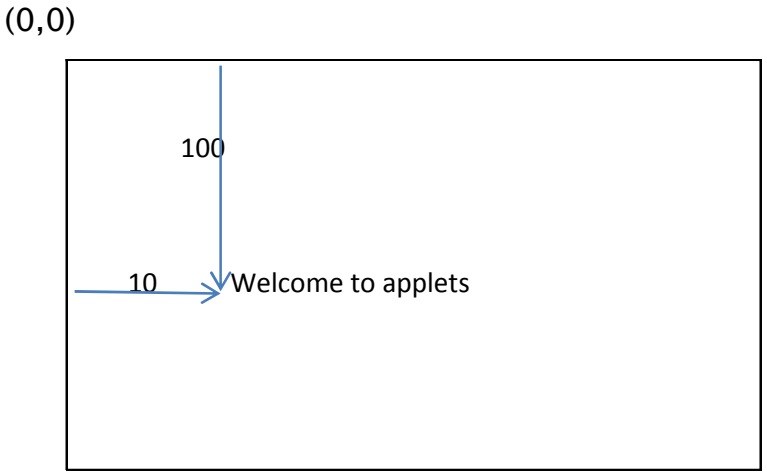
Explanation

First, we import the “awt” package i.e. “abstract window toolkit” and then the “applet” package.

Then the Applet source file class extends from Applet class.

It uses a method called paint() with Graphics class and object g as parameter.

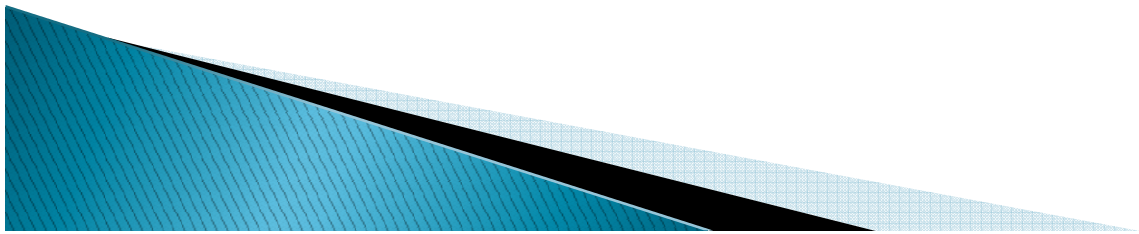
It uses a method drawString() which will have three parameters. The actual string to be drawn, and the location of (10,100) as follows.



Applet html file

```
<applet
code=ap.class
width=400
height=200>
</applet>
```

1. Save it at the same location d:\ as – ap.html
2. Now we can Execute the file as
   - appletviewer ap.html
   - double click on ap.html file



Output

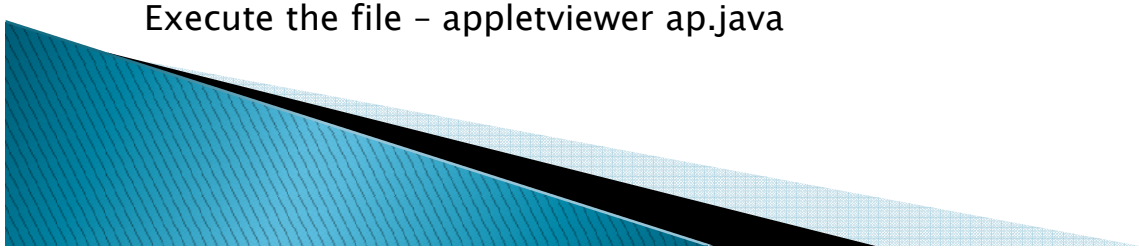
welcome to applets

Applet –Execution as a single file

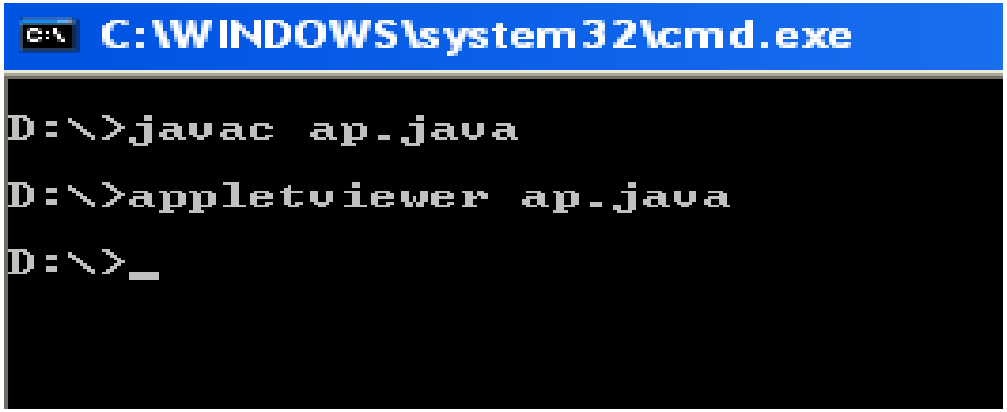
Now the Applet source code and the html file can be created in a single file.

```
import java.awt.*;
import java.applet.*;
/*<applet
code=ap.class
width=400
height=200>
</applet>
*/
public class ap extends Applet
{
public void paint(Graphics g)
{
g.drawString("welcome to applets",10,100);
}
}
```

Now save the file at any location say d:\ – ap.java
Compile the file – Javac ap.java
Execute the file – appletviewer ap.java



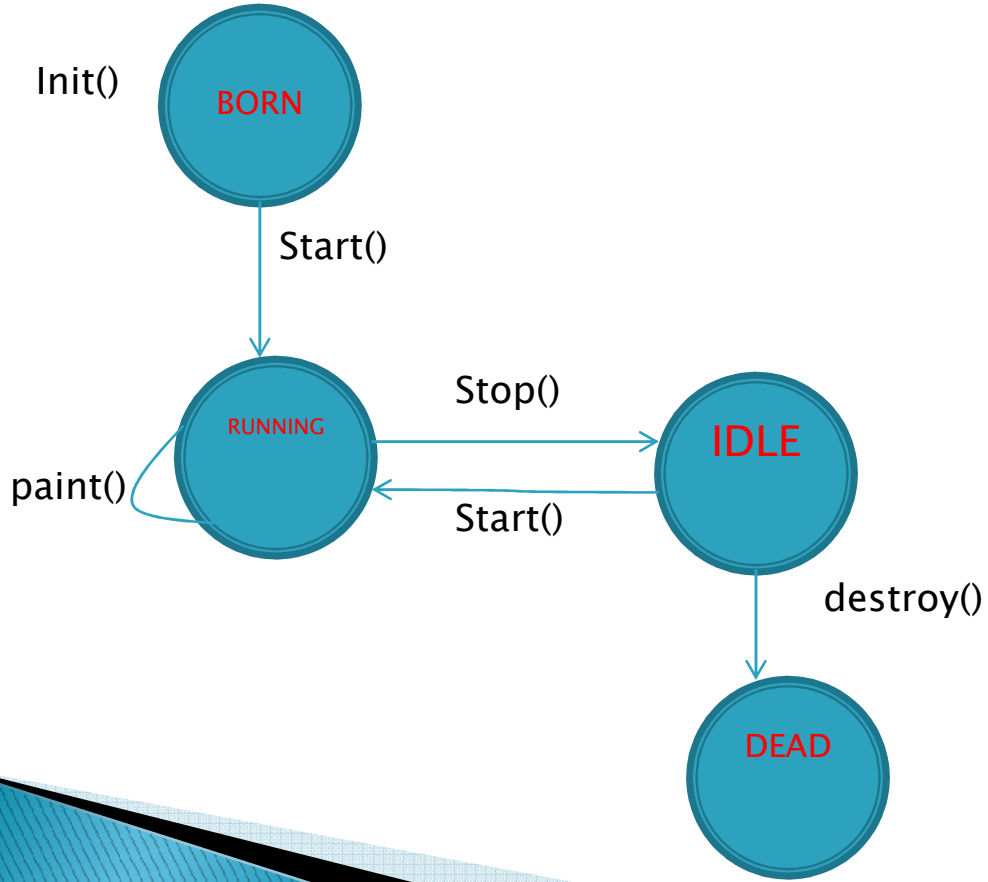
Output



welcome to applets

Applet life cycle

Here the Thread is said to be BORN when the init() is called.
When the Start() is called, Thread goes to RUNNING state.
Thread goes to IDLE state, when stop() is called.
Destroy() is called to update the Parameters in the Server before DEAD state.
Note – All the methods are inherited from Parent class "Applet".



Passing parameters to Applets

```
import java.awt.*;
import java.applet.*;
/*<applet
      code=hellojavaparam.class
      width=400
      height=200>
     <param name="string"
     value="applet">
   </applet>
*/
public class hellojavaparam extends Applet
{
 String str;
 public void init()
 {
  str=getParameter("string");
  if(str==null)
   str="java";
  str="hello"+str;
 }
public void paint(Graphics g)
 {
  g.drawString(str,10,100);
 }
}
```

Output

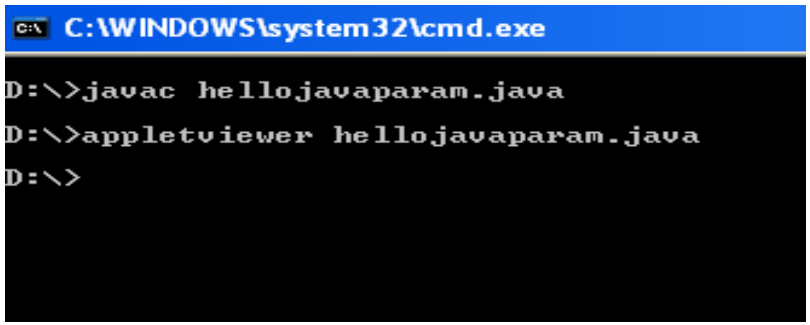
Now save the file – hellojavaparam.java

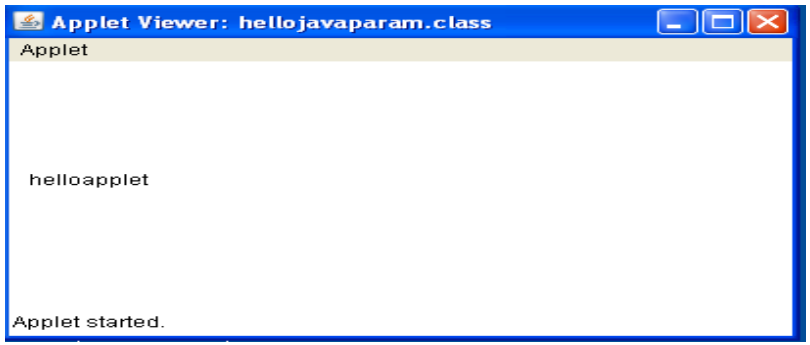
Compile the file.

Now execute the file –

- appletviewer hellojavaparam.java

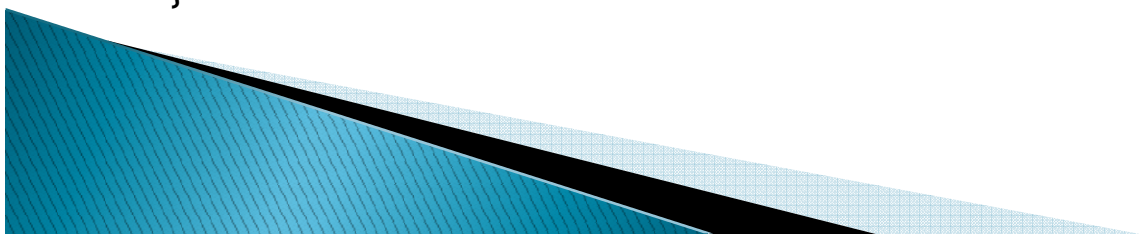
Here the parameter is passed from Applet tag to the Applet source code.



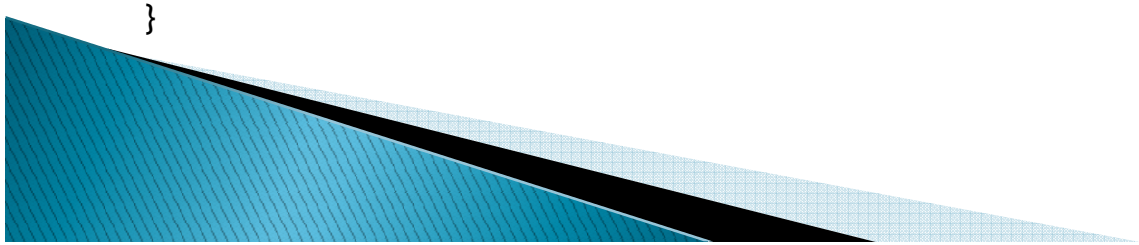


Getting input from the user

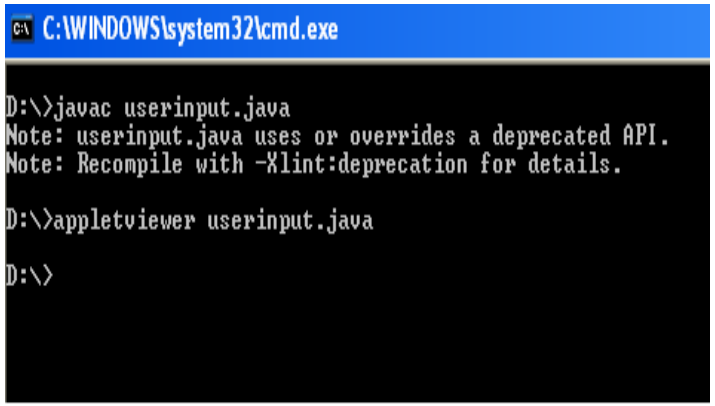
```
import java.awt.*;
import java.applet.*;
/*<applet code=userinput.class
 width=300
 height=300>
</applet>
*/
public class userinput extends Applet
{
  TextField text1,text2;
  public void init()
  {
    text1=new TextField(8);
    text2=new TextField(8);
    add(text1);
    add(text2);
    text1.setText("0");
    text2.setText("0");
  }
```



```
public void paint(Graphics g)
 {
  int x=0,y=0,z=0;
  String s1,s2,s;
  g.drawString("input number in each box",10,50);
  try
   {
     s1=text1.getText();
     x=Integer.parseInt(s1);
     s2=text2.getText();
     y=Integer.parseInt(s2);
   }
  catch(Exception e){}
  z=x+y;
  s=String.valueOf(z);
  g.drawString("the sum is",10,75);
  g.drawString(s,100,75);
 }
 public boolean action(Event event,Object object)
 {
  repaint();
  return true;
 }
}
```



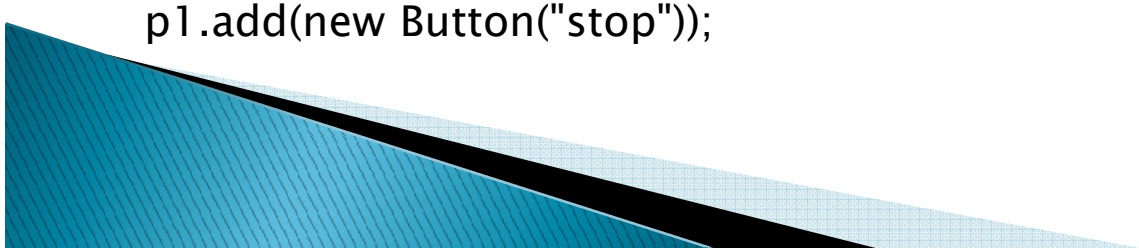
output



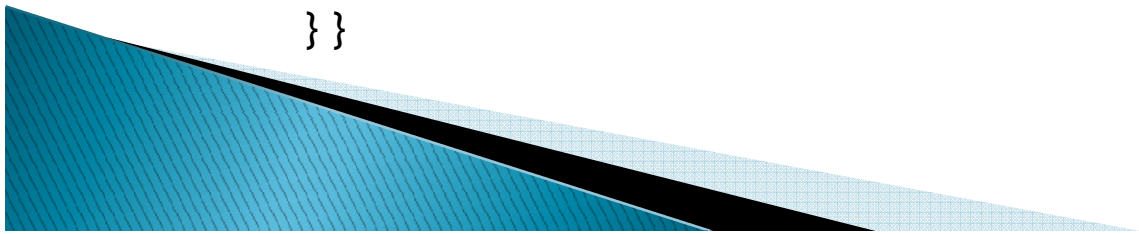
Applet started.

Graphics Programming- Panels

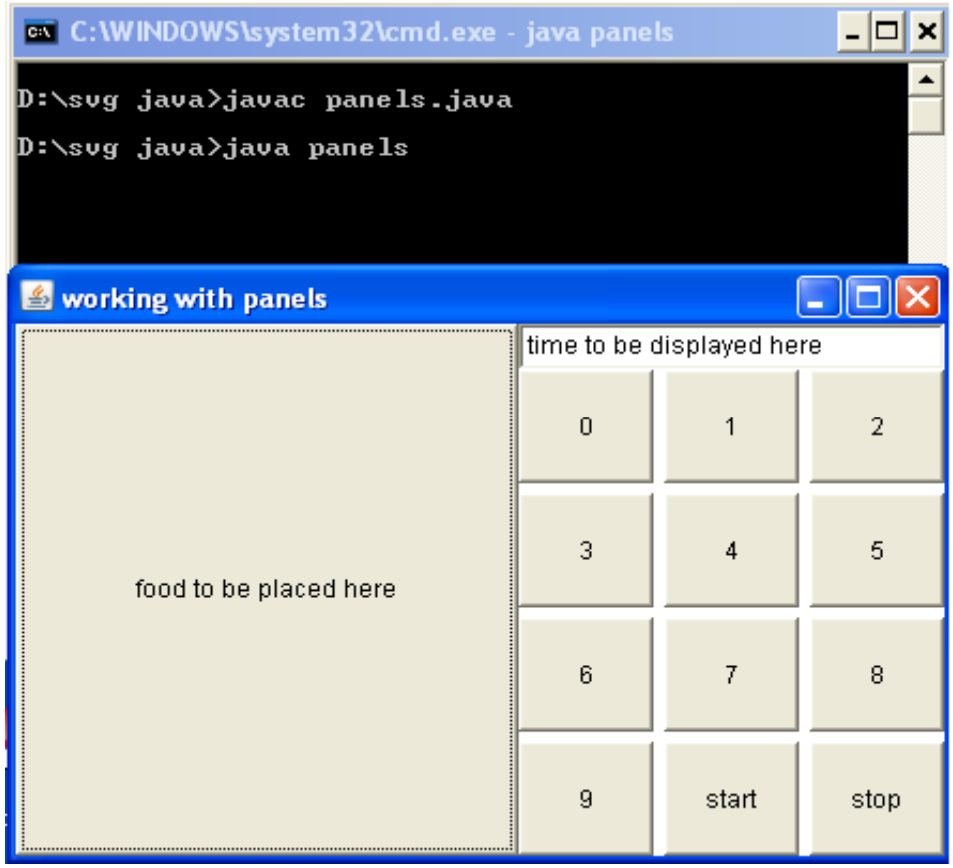
```
import java.awt.*;
import java.awt.event.*;
public class panels extends Frame
{
public panels()
{
//create panel p1 for the buttons and set grid layout

Panel p1=new Panel();
p1.setLayout(new GridLayout(4,3,5,5));

//add buttons to the panel

for(int i=0;i<=9;i++)
p1.add(new Button(""+i));
p1.add(new Button("start"));
p1.add(new Button("stop"));
```



```
//create panel p2 to hold a textfield and p1
Panel p2=new Panel(new BorderLayout());
p2.add(new TextField("time to be displayed here"),BorderLayout.NORTH);
//add contents to the frame
p2.add(p1,BorderLayout.CENTER);
add(p2,BorderLayout.EAST);
add(new Button("food to be placed here"),BorderLayout.CENTER);
}
public static void main(String atgs[])
{
panels f=new panels();
f.setTitle("working with panels");f.setSize(400,300);f.setVisible(true);
//closing a frame will exit from the application
    f.addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent evt)
    {
    System.exit(0);
    }});
} }
```



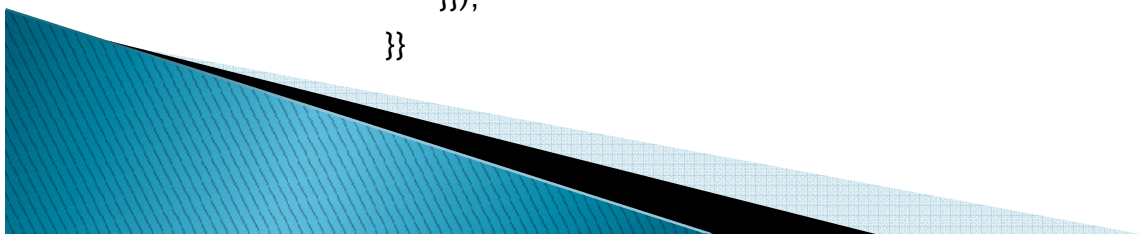
Output


C:\WINDOWS\system32\cmd.exe - java panels
D:\svg java>javac panels.java
D:\svg java>java panels
working with panels
food to be placed here
time to be displayed here
0
1
2
3
4
5
6
7
8
9
start
stop

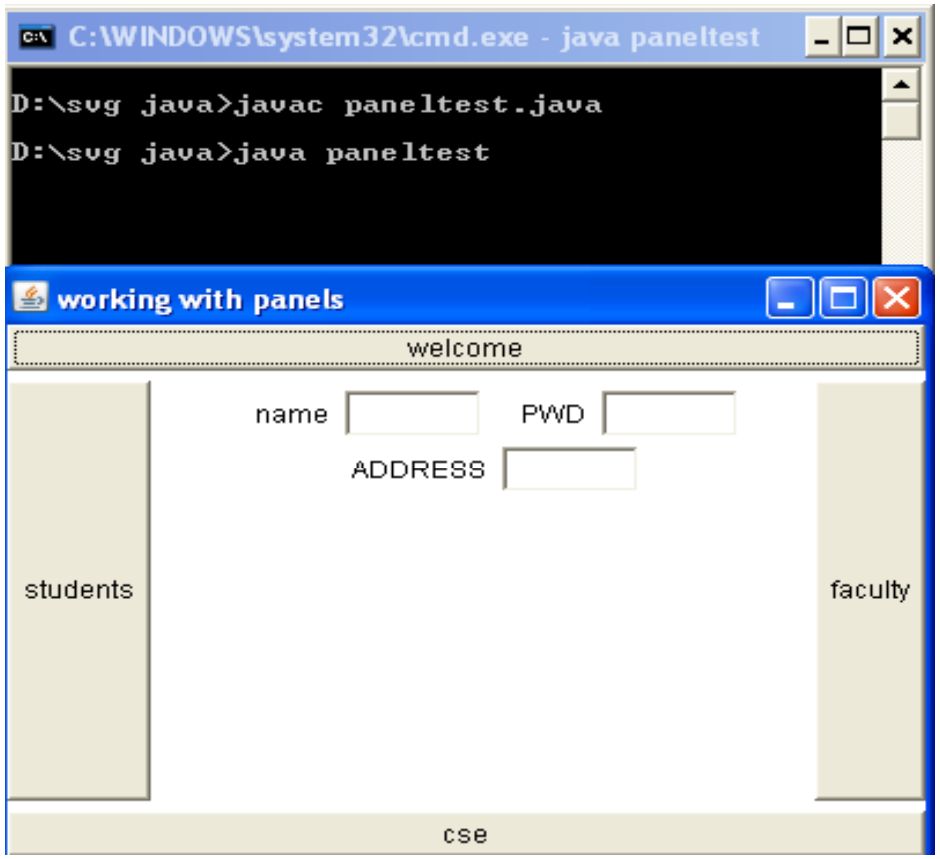
Panels

```
import java.awt.*;
import java.awt.event.*;
public class paneltest extends Frame
{
public paneltest()
{

//create panel p1 for the buttons and set Border layout

Panel p1=new Panel();
p1.setLayout(new BorderLayout(5,5));

//add buttons to the panel

      Button b1=new Button("welcome");       Button b2=new Button("cse");
      Button b3=new Button("students");      Button b4=new Button("faculty");
       p1.add(b1,BorderLayout.NORTH);       p1.add(b2,BorderLayout.SOUTH);
       p1.add(b3,BorderLayout.WEST);        p1.add(b4,BorderLayout.EAST);

//create panel p2 for labels & TextFields

Panel p2=new Panel(new FlowLayout());
      Label l1,l2,l3;
      l1=new Label("name",Label.RIGHT);
      l2=new Label("PWD",Label.RIGHT);
      l3=new Label("ADDRESS",Label.RIGHT);
```

```
TextField t1,t2,t3;
    t1=new TextField(5);
    t2=new TextField(5);
    t3=new TextField(5);
    p2.add(l1);p2.add(t1);              p2.add(l2);p2.add(t2); p2.add(l3);
                                p2.add(t3);
//add contents to the frame
p1.add(p2,BorderLayout.CENTER);
add(p1);
}
public static void main(String args[])
{
paneltest f=new paneltest();
f.setTitle("working with panels");
f.setSize(400,300);
f.setVisible(true);
//closing a frame will exit from the application

    f.addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent evt)
    {
    System.exit(0);
    }});
}}
```

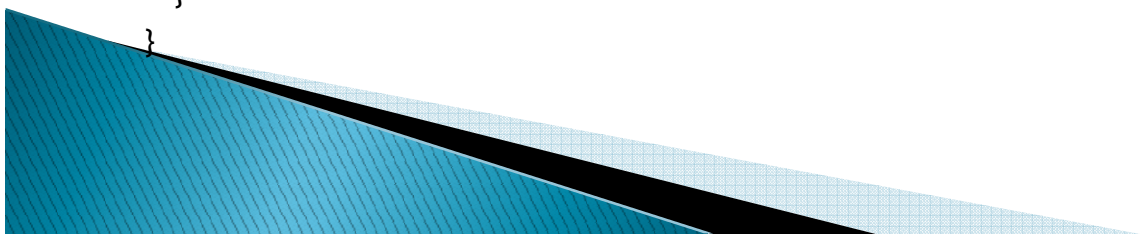


Output


C:\WINDOWS\system32\cmd.exe - java paneltest
D:\svg java>javac paneltest.java
D:\svg java>java paneltest
working with panels
welcome
name
PWD
ADDRESS
students
faculty
cse

Graphics Programming –Geometric Shapes

```
import java.awt.*;
import java.applet.*;
/*<applet code=shapes width=300  height=300>
</applet>*/
public class shapes extends Applet
{
public void paint(Graphics g)
 {
   g.drawOval(30,200,80,40);                    g.drawLine(30,220,70,50);
   g.drawLine(110,220,70,50);                   g.drawString("cone",50,260);
   g.drawOval(200,200,80,40);                   g.drawOval(200,50,80,40);
   g.drawLine(200,70,200,220);                  g.drawLine(280,70,280,220);
   g.drawString("cylinder",210,260);            g.drawRect(360,50,80,80);
   g.drawRect(400,90,80,80);                    g.drawLine(360,50,400,90);
   g.drawLine(440,50,480,90);                   g.drawLine(360,130,400,170);
   g.drawLine(440,130,480,170);                 g.drawString("cube",420,200);
   g.drawRect(30,350,100,100);                  g.drawOval(30,350,100,100);
   g.drawString("circle in square",30,470);     g.drawOval(220,350,100,100);
   g.drawRect(235,365,70,70);                   g.drawString("square in circle",220,470);
  }
}
```



Explanation

g.drawLine(110,220,70,50);

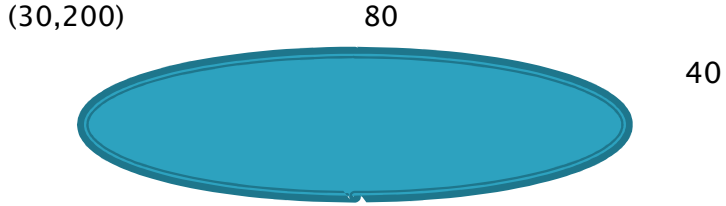
(110,220) is the start point & (70,50) is the end point.

g.drawRect(360,50,80,80);

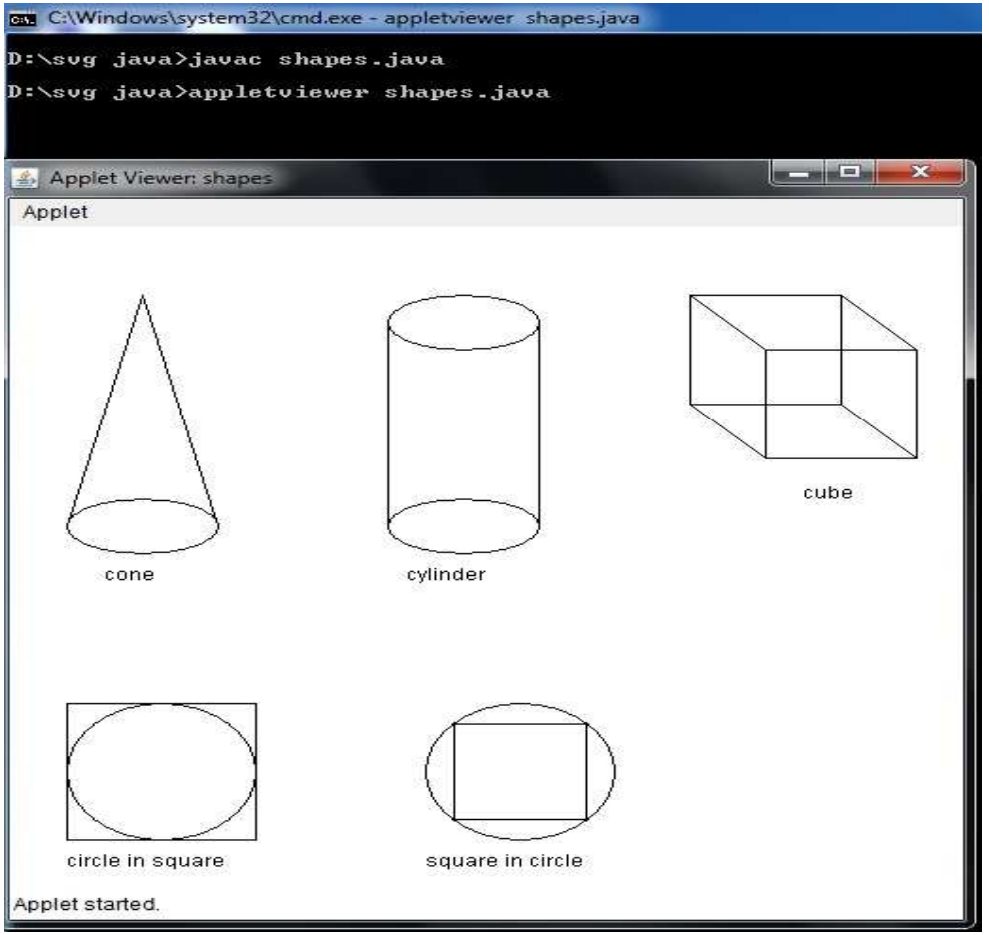
(360,50) is the start point, 80 is width, 80 is depth of Rectangle

g.drawOval(30,200,80,40);

(30,200) is the start point of imaginary rectangle, 80 is width,40 is depth of oval.



Output


C:\Windows\system32\cmd.exe - appletviewer shapes.java
D:\svg java>javac shapes.java
D:\svg java>appletviewer shapes.java
Applet Viewer: shapes
Applet
cone
cylinder
cube
circle in square
square in circle
Applet started.

Layout Manager – Border Layout

```
import java.awt.*;
import java.awt.event.*;
public class ShowBorderLayout extends Frame {
  public ShowBorderLayout()
  {
     setTitle("ShowBorderLayout");
     setLayout(new BorderLayout(5,10));
     add("East",new Button("East"));
     add("South",new Button("South"));
     add("West",new Button("West"));
     add("North",new Button("North"));
     add("Center",new Button("Center"));
  }
  public static void main(String args[])
   {
      Frame f=new ShowBorderLayout(); f.setSize(400,300);
      f.setVisible(true);
    //closing a frame will exit from the application
    f.addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent evt)
    {
    System.exit(0);
    }});
  } }
```

Note – Here the Frame concept is used to demonstrate Border Layout.

Output

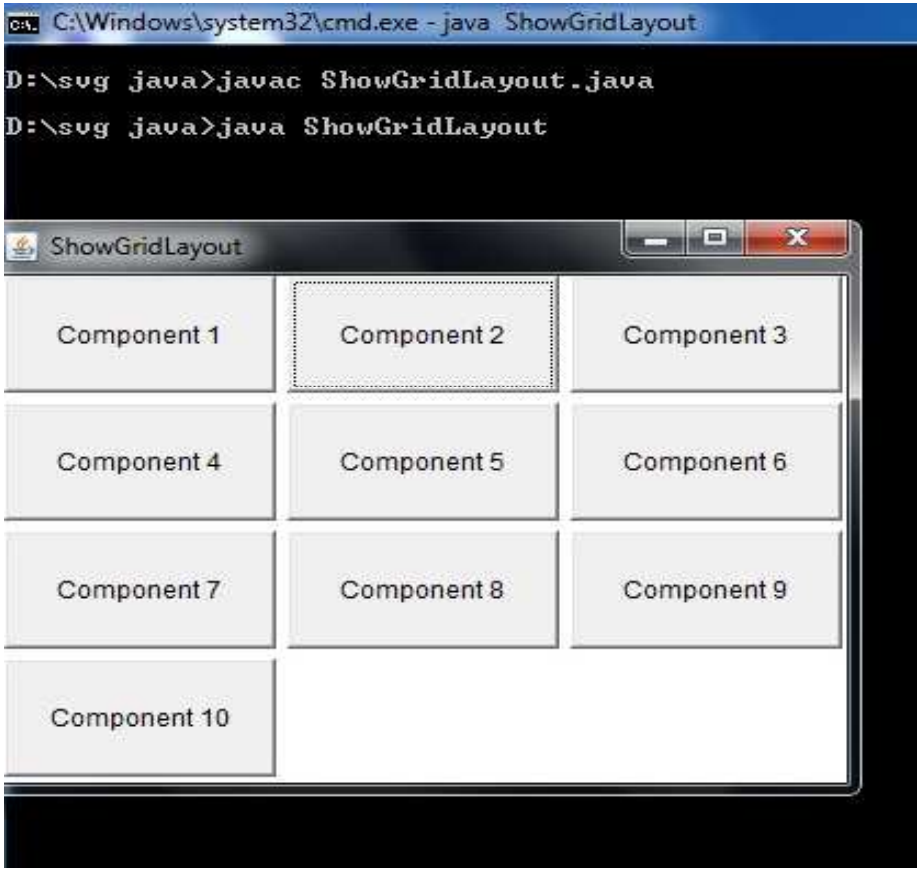

C:\Windows\system32\cmd.exe - java ShowBorderLayout
D:\svg java>javac ShowBorderLayout.java
D:\svg java>java ShowBorderLayout
ShowBorderLayout
North
West
Center
East
South

Grid Layout

```
import java.awt.*;
import java.awt.event.*;
public class ShowGridLayout extends Frame {
  public ShowGridLayout()
  {
     setTitle("ShowGridLayout");
     setLayout(new GridLayout(4,3,5,5));
     for(int i=1;i<=10;i++)
       add(new Button("Component "+i));
  }
  public static void main(String args[])
   {
      Frame f=new ShowGridLayout(); f.setSize(400,300);
      f.setVisible(true);
    //closing a frame will exit from the application
    f.addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent evt)
    {
    System.exit(0);
    }});
  } }
```

Note – Here the Frame concept is used to demonstrate Grid Layout.

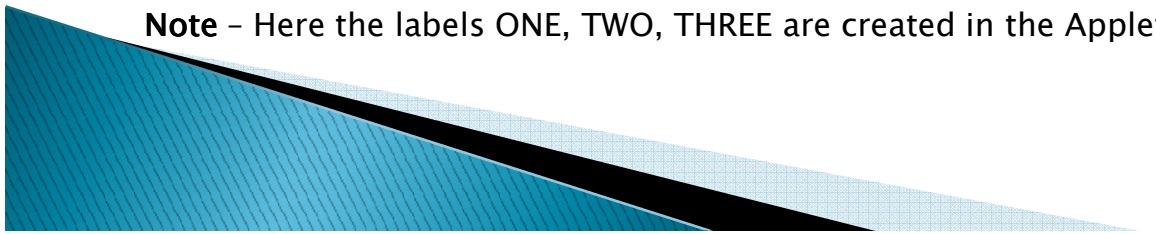
Output


C:\Windows\system32\cmd.exe - java ShowGridLayout
D:\svg java>javac ShowGridLayout.java
D:\svg java>java ShowGridLayout
ShowGridLayout
Component 1
Component 2
Component 3
Component 4
Component 5
Component 6
Component 7
Component 8
Component 9
Component 10

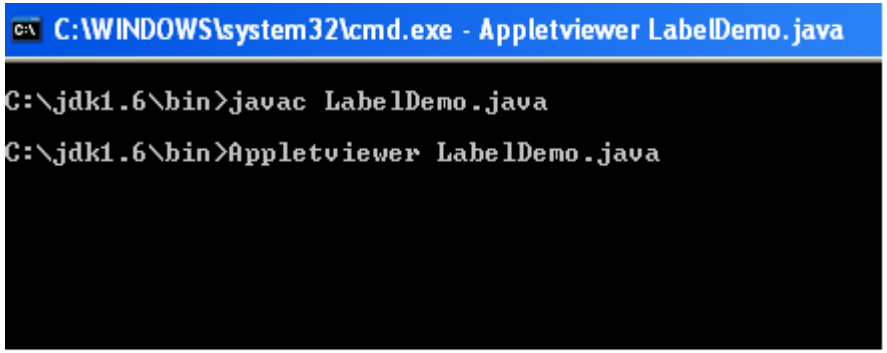
Graphics Programming – Label Demo

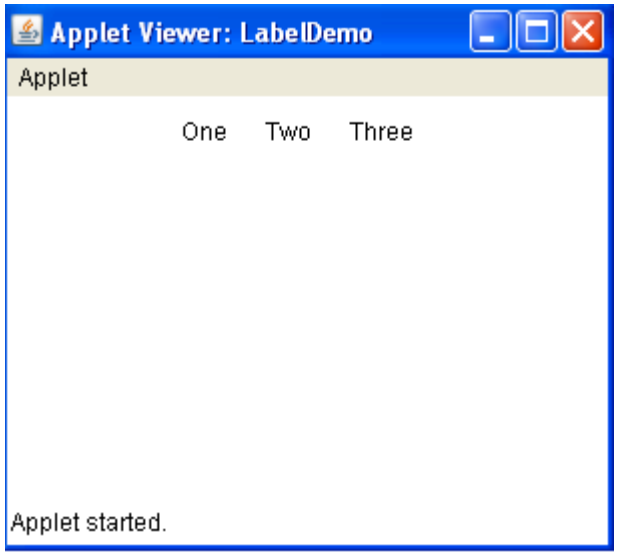
```
//Demostrate Labels
import java.awt.*;
import java.applet.*;
/*
<applet code="LabelDemo" width=300 height=200>
</applet>
*/

public class LabelDemo extends Applet {
    public void init() {
            Label one = new Label("One");
            Label two = new Label("Two");
            Label three = new Label("Three");

//add labels to applet window
    add(one);
    add(two);
    add(three);
}}
```

Note – Here the labels ONE, TWO, THREE are created in the Applet.



output


C:\WINDOWS\system32\cmd.exe - Appletviewer LabelDemo.java
C:\jdk1.6\bin>javac LabelDemo.java
C:\jdk1.6\bin>Appletviewer LabelDemo.java

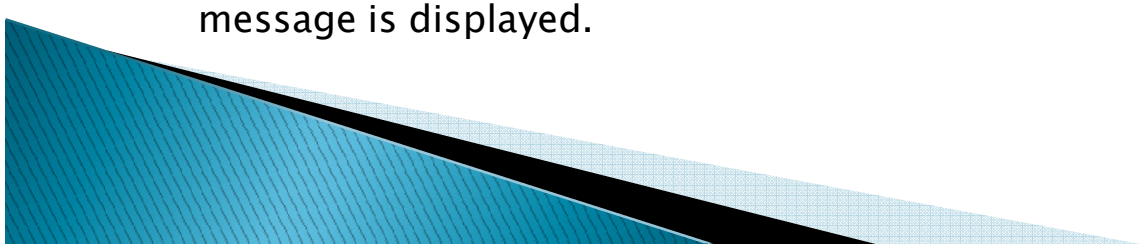

Applet Viewer: LabelDemo
Applet
One Two Three
Applet started.

Button Demo

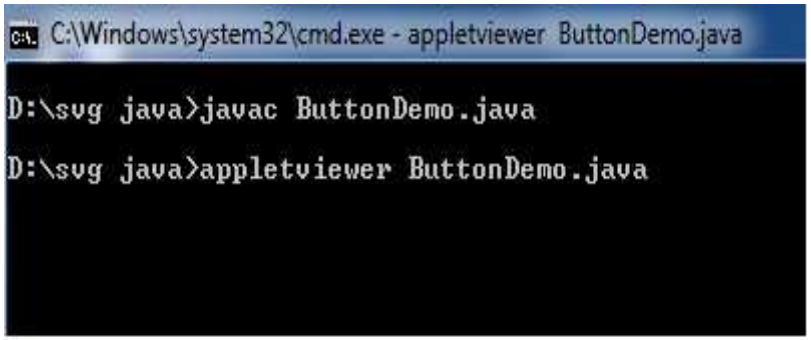
```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="ButtonDemo" width =250 height=150>
</applet>
*/

public class ButtonDemo extends Applet implements ActionListener {
    String msg = " ";
    Button yes,no,maybe;
    public void init() {
        yes=new Button("Yes");
        no=new Button("No");
        maybe=new Button("Undecided");

        add(yes);
        add(no);
        add(maybe);

        yes.addActionListener(this);
        no.addActionListener(this);
        maybe.addActionListener(this);
    }
```

```
public void actionPerformed(ActionEvent ae)
    {
            String str = ae.getActionCommand();
            if(str.equals("Yes")) {
                    msg="You pressed Yes.";
            }
            else if (str.equals("No")) {
                    msg="You pressed No.";
            }
            else {
                    msg="You pressed Undecided.";
            }
            repaint();
    }

    public  void paint(Graphics g) {
            g.drawString(msg,6,100);
    }}
```

Note – Here the Buttons YES, NO, UNDECIDED are created. When we click a button, a message is displayed.



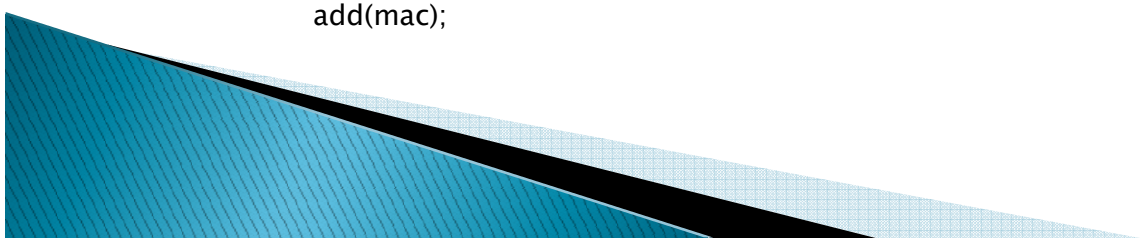
Output


C:\Windows\system32\cmd.exe - appletviewer ButtonDemo.java
D:\svg java>javac ButtonDemo.java
D:\svg java>appletviewer ButtonDemo.java


Applet Viewer: Button...
Applet
Yes
No
Undecided
You pressed Yes.
Applet started.

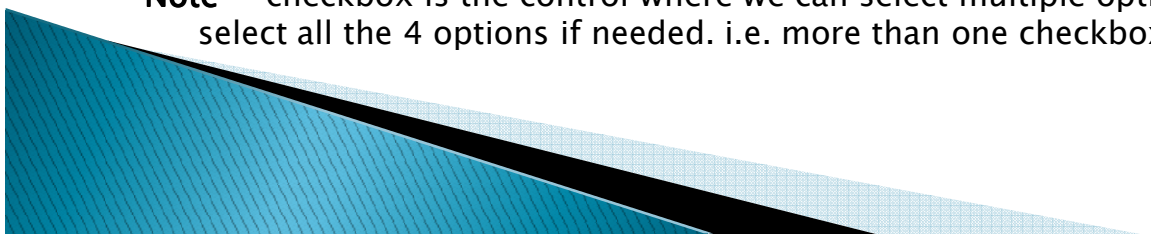
Checkbox Demo

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
    <applet code = "CheckBoxDemo" width=250 height=200>
    </applet>
*/

public class CheckBoxDemo extends Applet implements ItemListener {
    String msg = " ";
    Checkbox Win98, winNT,solaris,mac;

    public void init()
    {
            Win98 = new Checkbox("Windows 98/XP",null,true);
            winNT = new Checkbox("Windows NT/2000");
            solaris = new Checkbox("Solaris");
            mac = new Checkbox("MacOS");

            add(Win98);
            add(winNT);
            add(solaris);
            add(mac);
```

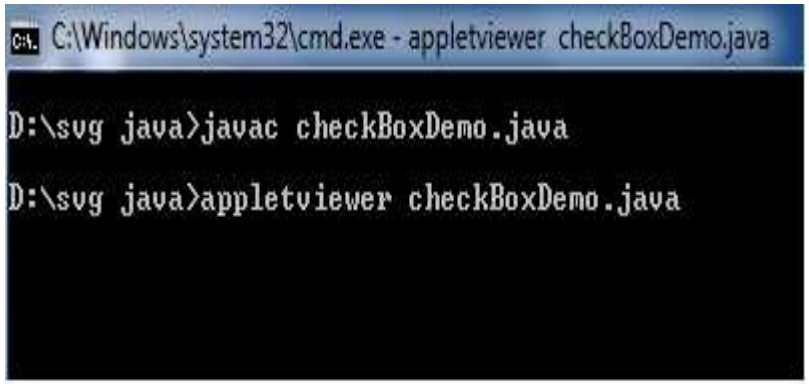


```
        Win98.addItemListener(this);
        winNT.addItemListener(this);
        solaris.addItemListener(this);
        mac.addItemListener(this);
}
        public void itemStateChanged(ItemEvent ie) {
                repaint();
        }

        //Display current state of the check boxes.
        public void paint(Graphics g){
                msg=" Current state :";
                g.drawString(msg,6,80);
                msg=" Windows 98/XP:"+Win98.getState();
                g.drawString(msg,6,100);
                msg=" Windows NT/2000:"+winNT.getState();
                g.drawString(msg,6,120);
                msg=" Solaris :"+solaris.getState();
                g.drawString(msg,6,140);
                msg=" MacOS: "+mac.getState();
                g.drawString(msg,6,160);
        } }
```

Note - checkbox is the control where we can select multiple options. i.e. If we have 4 checkboxes, we can select all the 4 options if needed. i.e. more than one checkbox can be selected at a time.



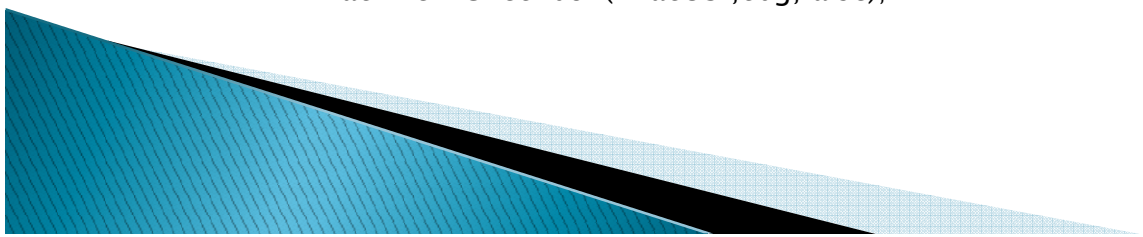
Output


C:\Windows\system32\cmd.exe - appletviewer checkBoxDemo.java
D:\svg java>javac checkBoxDemo.java
D:\svg java>appletviewer checkBoxDemo.java


Applet Viewer: Check...
Applet
Windows 98/XP
Windows NT/2000
Solaris
MacOS
Current state :
Windows 98/XP:true
Windows NT/2000:true
Solaris :true
MacOS: false
Applet started.

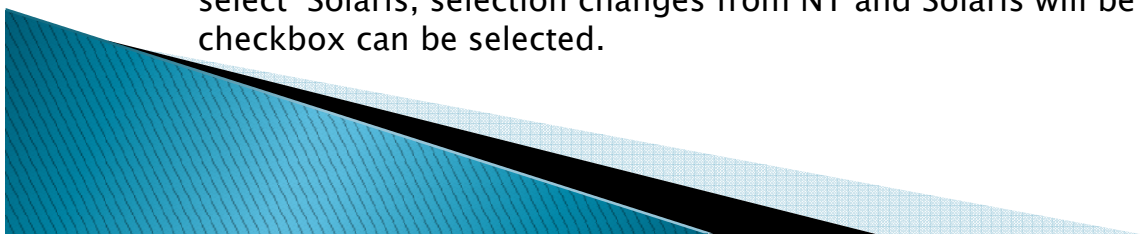
CheckboxGroup Demo

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

/* <applet code = "CBGroup" width=250 height=200>
    </applet>
*/


public class CBGroup extends Applet implements ItemListener {
    String msg = " ";
    Checkbox Win98,winNT,solaris,mac;
    CheckboxGroup cbg;

    public void init()
    {
            cbg=new CheckboxGroup();
            Win98= new Checkbox("Windows 98/XP",cbg,true);
            winNT= new Checkbox("Windows NT/2000",cbg,false);
            solaris=new Checkbox("Solaris",cbg,false);
            mac=new Checkbox("MacOS",cbg,false);
```

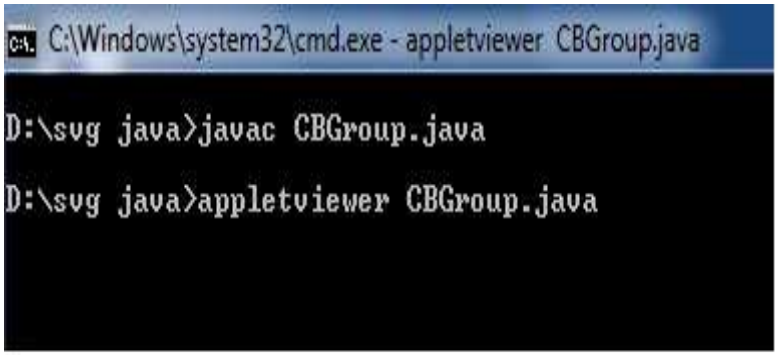


```
add(Win98);
            add(winNT);
            add(solaris);
            add(mac);
            Win98.addItemListener(this);
            winNT.addItemListener(this);
            solaris.addItemListener(this);
            mac.addItemListener(this);
    }
    public void itemStateChanged(ItemEvent ie)
    {
                        repaint();
    }
    public void paint(Graphics g)
    {
            msg="Current selection: ";
            msg+=cbg.getSelectedCheckbox().getLabel();
            g.drawString(msg,6,100);
    } }
```

Note - Here in the CheckboxGroup, we can select only one option. i.e. if we have 4 checkboxes in a checkboxgroup, we can select only one checkbox. if we select windows NT, NT will be selected. If we select Solaris, selection changes from NT and Solaris will be selected. So, At a time, only one checkbox can be selected.



Output

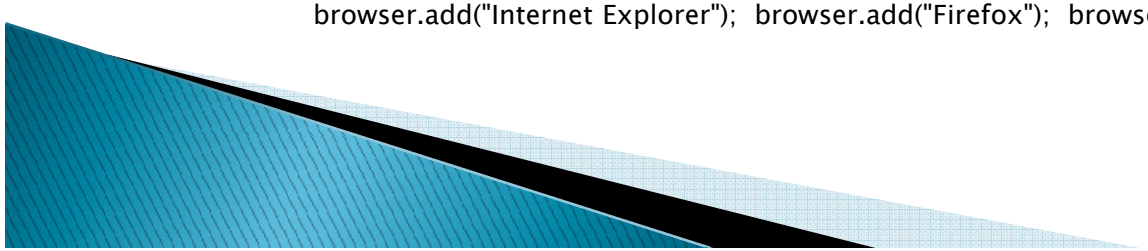

C:\Windows\system32\cmd.exe - appletviewer CBGroup.java
D:\svg java>javac CBGroup.java
D:\svg java>appletviewer CBGroup.java


Applet Viewer: CBGro...
Applet
Windows 98/XP
Windows NT/2000
Solaris
MacOS
Current selection: Windows NT/2000
Applet started.


Applet Viewer: CBGro...
Applet
Windows 98/XP
Windows NT/2000
Solaris
MacOS
Current selection: Solaris
Applet started.

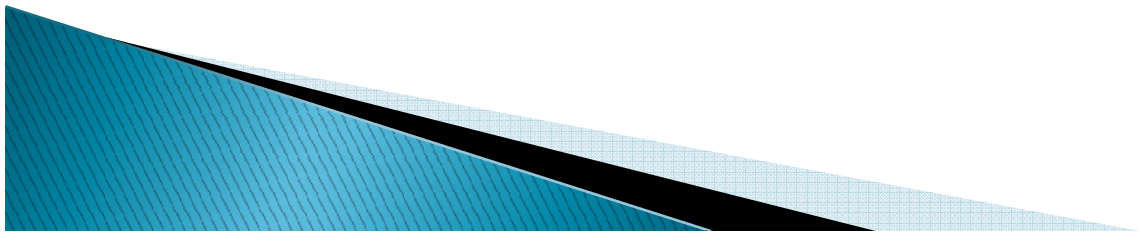
Choice Demo

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
    <applet code = "ChoiceDemo" width=300 height=180>
    </applet>
*/

public class ChoiceDemo extends Applet implements ItemListener {
    Choice os,browser;
    String msg=" ";

    public void init() {
            os=new Choice();
            browser= new Choice();

            //add items to os list
            os.add("Windows 98/XP");
            os.add("Windows NT/2000");
            os.add("Solaris");
            os.add("MacOS");

            //add items to browser list
            browser.add("Internet Explorer");  browser.add("Firefox");  browser.add("Opera");
```

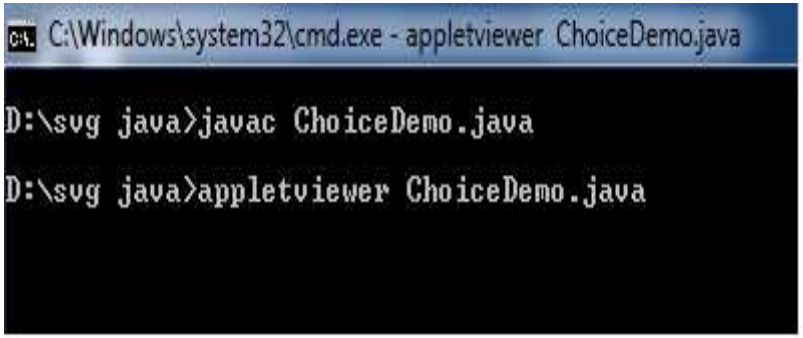


```
//add choice lists to window
        add(os);
        add(browser);

        //register to receive item events
        os.addItemListener(this);
        browser.addItemListener(this);
    }
        public void itemStateChanged(ItemEvent ie) {
                repaint();
    }

    public void paint(Graphics g) {
        msg="Current OS:";
        msg+=os.getSelectedItem();
        g.drawString(msg,6,120);
        msg="Current Browser: ";
        msg+=browser.getSelectedItem();
        g.drawString(msg,6,140);
    } }
```

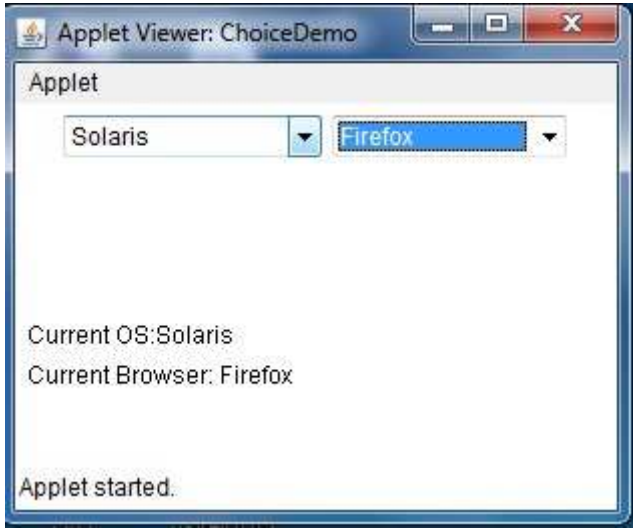
Note – Here we have one of the choice control OS. Under this we have windows 98, NT, solaris, Mac. It works like a DropDownList. We can select one from the list.



Output

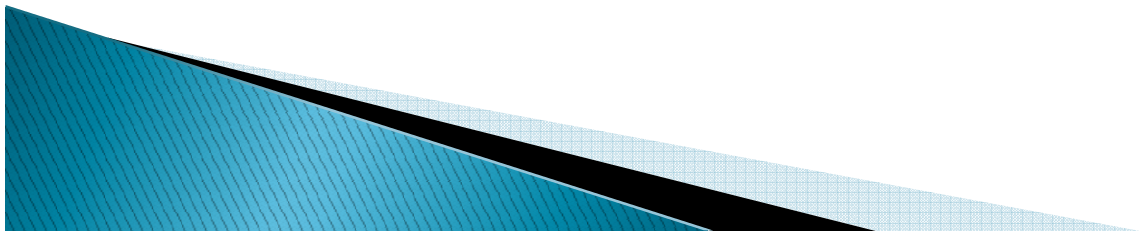

C:\Windows\system32\cmd.exe - appletviewer ChoiceDemo.java
D:\svg java>javac ChoiceDemo.java
D:\svg java>appletviewer ChoiceDemo.java


Applet Viewer: ChoiceDemo
Applet
Windows 98/XP
Internet Explorer
Windows 98/XP
Windows NT/2000
Solaris
MacOS
Current OS:Windows 98/XP
Current Browser: Internet Explorer
Applet started.


Applet Viewer: ChoiceDemo
Applet
Solaris
Firefox
Current OS:Solaris
Current Browser: Firefox
Applet started.

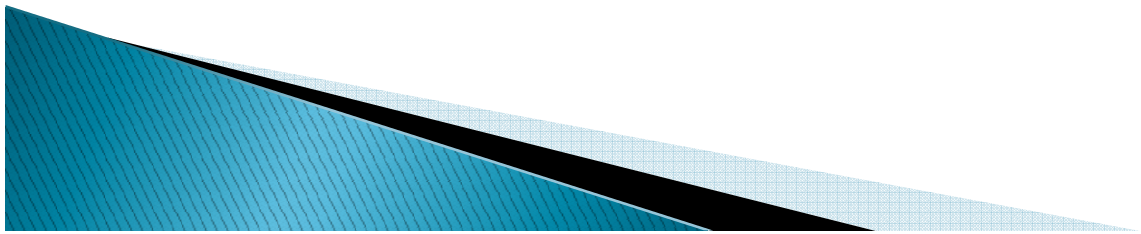
Textfield Demo

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
    <applet code="TextFieldDemo" width=380 height=150>
    </applet>
*/

    public class TextFieldDemo extends Applet
            implements ActionListener {
                    TextField name,pass;
                    public void init() {
                            Label namep=new Label("Name: ",Label.RIGHT); // Label Right Alignment.
                            Label passp=new Label("Password: ",Label.RIGHT);
                            name= new TextField(12);
                            pass= new TextField(8);
                            pass.setEchoChar('?');// it will display/echo '?' when password is typed.
                            add(namep);
                            add(name);
                            add(passp);
                            add(pass);
```

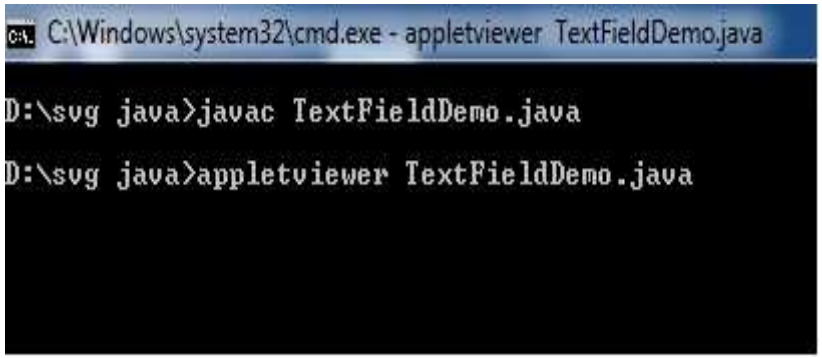


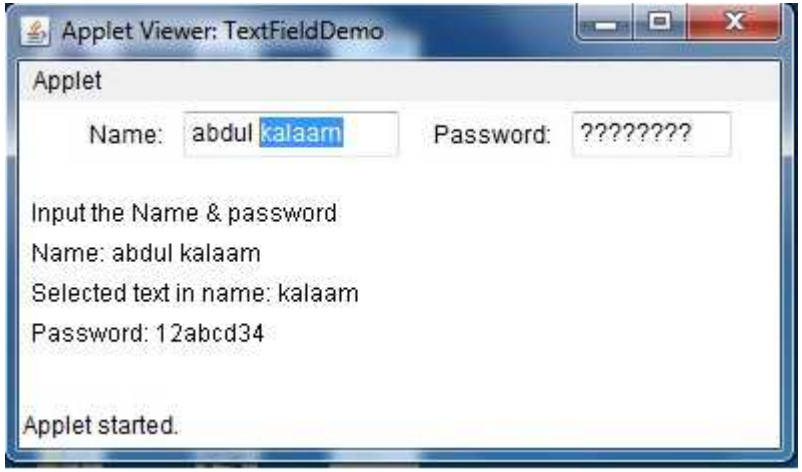
```
//register to receive action events
                    name.addActionListener(this);
                    pass.addActionListener(this);
                }

                //User pressed Enter.
                public void actionPerformed(ActionEvent ae) {
                repaint();
        }

                public void paint(Graphics g) {
                    g.drawString("Input the Name & password",6,60);
                    g.drawString("Name: "+name.getText(),6,80);
                    g.drawString("Selected text in name: "+name.getSelectedText(),6,100);
                    g.drawString("Password: "+ pass.getText(),6,120);
                }
        }
```

Note – Here we Input the Name & password. Password will be displayed with '?' character for security. Then, select the text in the Name TextField and press enter. Then it will display the name, Selected text in the name, password.



Output


C:\Windows\system32\cmd.exe - appletviewer TextFieldDemo.java
D:\svg java>javac TextFieldDemo.java
D:\svg java>appletviewer TextFieldDemo.java


Applet Viewer: TextFieldDemo
Applet
Name: abdul kalaam
Password: ????????
Input the Name & password
Name: abdul kalaam
Selected text in name: kalaam
Password: 12abcd34
Applet started.

Mini Project

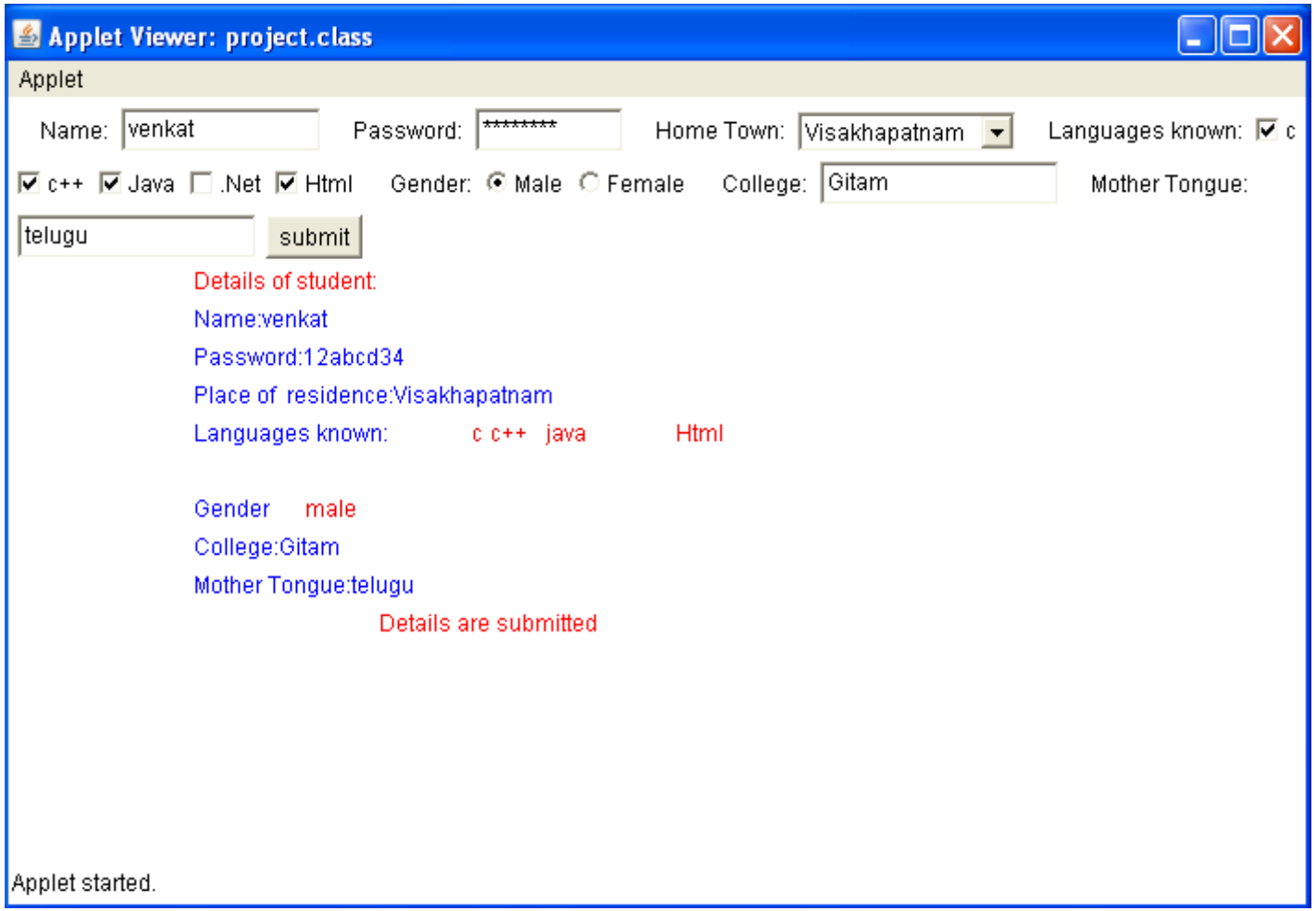
```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
  <applet code=project.class
        WIDTH=700
        HEIGHT=400>
  </applet>
*/
public class project extends Applet implements ActionListener,ItemListener
{

   TextField name,pass,co,mt;
   Choice res;
   CheckboxGroup cbg;
   Checkbox  c,cop,java,Net,Html,male,female;
   String msg=" ";
   Button submit;
   public void init()
   {
       setLayout(new FlowLayout(FlowLayout.LEFT));
       Label namep=new Label("Name:",Label.RIGHT);
       name=new TextField(12);
       Label passp=new Label("Password:",Label.RIGHT);
       pass=new TextField(8);
       pass.setEchoChar('*');
       Label place=new Label("Home Town:",Label.RIGHT);
       res=new Choice();
```

```
res.add("Hyderabad");res.add("Kakinada");res.add("Mahaboobnagar");
res.add("Nalgonda");res.add("Ongole");res.add("Visakhapatnam");
res.add("Warangal");Label lang=new Label("Languages known:",Label.RIGHT);
cbg=new CheckboxGroup();c=new Checkbox("c");
cop=new Checkbox("c++");java=new Checkbox("Java");
Net=new Checkbox(".Net");Html=new Checkbox("Html");
Label gender=new Label ("Gender:",Label.RIGHT);
male=new Checkbox("Male",cbg,true);
female=new Checkbox("Female",cbg,true);
Label col=new Label("College:",Label.RIGHT);
co=new TextField(15);Label mtl=new Label("Mother Tongue:",Label.RIGHT);
mt=new TextField(15);submit=new Button("submit");
add(namep);add(name);add(passp);add(pass);
add(place);add(res);add(lang);add(c);
add(cop);add(java);add(Net);add(Html);
add(gender);add(male);add(female);add(col);
add(co);add(mtl);add(mt);add(submit);
name.addActionListener(this);
pass.addActionListener(this);
res.addItemListener(this);
c.addItemListener(this);
cop.addItemListener(this);
java.addItemListener(this);
Net.addItemListener(this);
Html.addItemListener(this);
male.addItemListener(this);
female.addItemListener(this);
co.addActionListener(this);
mt.addActionListener(this);
submit.addActionListener(this); }
```

```
public void itemStateChanged(ItemEvent ie)
  {
     repaint();
  }
  public void actionPerformed(ActionEvent ae)
  {
     String str=ae.getActionCommand();
     if(str.equals("submit"))
     {
        msg="Details are submitted";
     }
     repaint();
  }
  public void paint(Graphics g)
  {
            String mesg="  ";
            Color c1=new Color(255,0,0);
             g.setColor(c1);
             g.drawString(msg,200,280);
              mesg=   "Details of student:";
             g.drawString(mesg,100,100);
              Color c2=new Color(0,0,255);
              g.setColor(c2);
              g.drawString("Name:"+name.getText(),100,120);
              g.drawString("Password:"+pass.getText(),100,140);
              g.drawString("Place of  residence:"+res.getSelectedItem(),100,160);
       g.drawString("Languages known:", 100, 180);
       Color c4 = new Color(236, 0,0);
       g.setColor(c4);
       boolean ct = c.getState();
       if (ct == true)
       {
                  g.drawString("c", 250, 180);
       }
       boolean coc = cop.getState();
```

```
if (coc == true)
        {
                g.drawString("c++", 260, 180);
        }
        boolean j = java.getState();
                if (j == true)
                {
                  g.drawString("java", 290, 180);
        }
                boolean n = Net.getState();
        if (n == true)
        {
                g.drawString(".Net", 330, 180);
        }
        boolean h = Html.getState();
        if (h == true)
        {
                g.drawString("Html", 360, 180);
        }
                g.setColor(c2);      g.drawString("Gender", 100, 200);    g.setColor(c4);
        boolean gm = male.getState();
        boolean gf = female.getState();
        if (gm == true)
        {
                  g.drawString("male", 160, 220);

                }
        else if (gf == true)
        {
                g.drawString("female", 160, 220);
       }
                g.setColor(c2);    g.drawString("College:"+co.getText(),100,240);
                g.drawString("Mother Tongue:"+mt.getText(),100,260);   } }
```

output


Applet Viewer: project.class
Applet
Name: venkat
Password: ********
Home Town: Visakhapatnam
Languages known: c
c++ Java .Net Html
Gender: Male Female
College: Gitam
Mother Tongue:
telugu
submit
Details of student:
Name:venkat
Password:12abcd34
Place of residence:Visakhapatnam
Languages known: c c++ java Html
Gender male
College:Gitam
Mother Tongue:telugu
Details are submitted
Applet started.

Share the Knowledge

&

Gain the Knowledge

Thank you