INDEX

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Introduction to Java

What is Java?
Java is an object Oriented Programming Language.

What is a Language?
It is a medium which is used to interact with others.

What is Programming Language?
It is also a medium which is used to interact with the computer.
Ex: C, C++, Java, C#.net, vb.net etc...

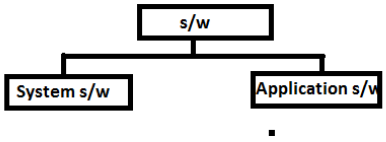
How we can interact with the computer?
By giving set of instructions/statements/commands.

What is a program?
a program is nothing but set of commands.

What is a software?
a software is nothing but collection of programs.

How many types of softwares are there?
2 types
1. System s/w 2. Application s/w.

What is the difference between system s/w and application s/w?



System s/w: It is a software which is used to manage the h/w components of a computer

Ex: O.S, Audio/Video Drivers, Network Drivers, Printer S/w.

Application S/w: It runs on system software.
1. Ms-office, Games, Billing S/w, Bank applications, photoshop, browsers etc..

Note: By using Java we can develop both system s/w and application s/w.

Categories of Programming Languages

These categories are divided based on the concepts they support, and way of writing. We have many no of categories some of them are given below.

1. Monolythic Programming Languages (Un-structured)
2. Structured Programming Languages
3. Procedural Programming Languages
4. Object Oriented Programming Language
5. Object Based Programming Languages

Monolythic Programming Languages: By using Monolythic Programming Languages we can write program that contain number of lines of code, which consist of only global data and sequential statements.

MSX BASIC language example:

```
key on
Ok
list
0  ' (c) 1987 by Andrew Nikitin
1 POKE &HFBB1,1:SCREEN0:WIDTH40:ONSTRIGG
OSUB3:STRIG(0)ON:KEYOFF:DEFUSR=&H156
2 FORI=1TO39:FORJ=0TO20:NEXTJ:VPOKEI-1,3
2:VPOKEI,62:NEXTI:GOTO2
3 H=104-I:VPOKEI-1,32:IFH>90THENRETURN2E
LSEVPOKEI,H:IFH=90THENFORC=14TO39:IFVPEE
K(C)=32THENRETURN2ELSENEXT:PRINT"Молодец
":H$=INPUT$(USR(1)):POKE&HFBB1,0:ENDELSE
RETURN2
Ok
```

DrawBacks:

- It is not subroutine concept
- The Program code is duplicated each time it is to be reused.
- Everything is global

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

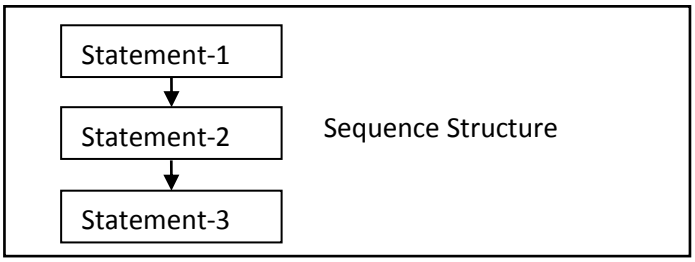
- There is no data abstraction
- Suitable for developing small and simple applications

Examples: These include early versions of BASIC (such as MSX BASIC and GW BASIC), JOSS, FOCAL, MUMPS, TELCOMP, machine-level code.

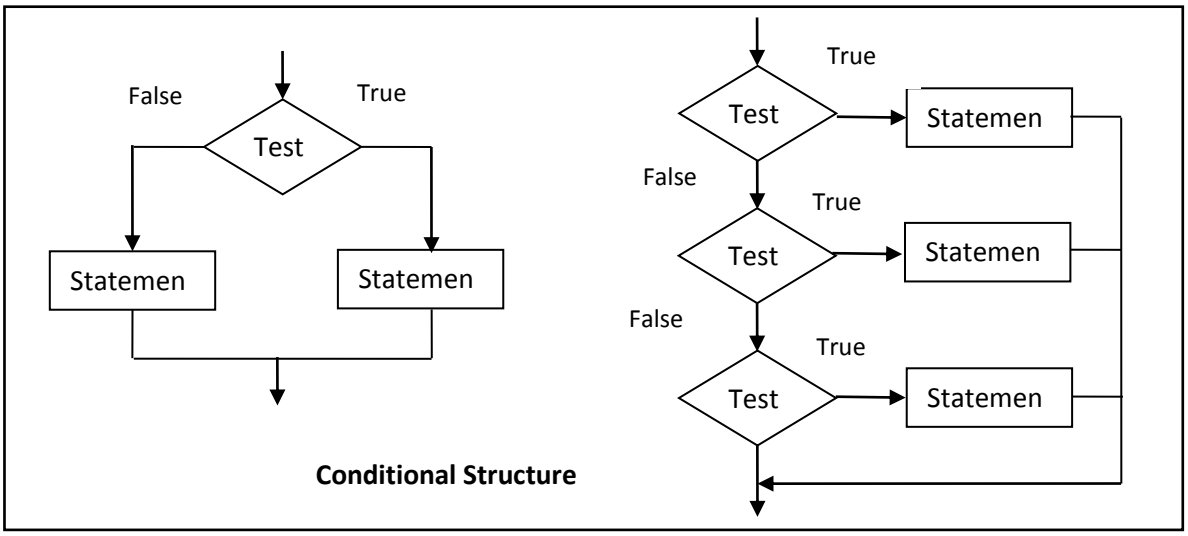
Structured Programming Language

Structured programming was first suggested by **Corrado Bohm** and **Guiseppe Jacopini**. The two mathematicians demonstrated that any computer program can be written with just three structures: **sequences, decisions and loops.**

Sequence Structure: "Sequence" refers to an ordered execution of statements. It contains program statements one after another.

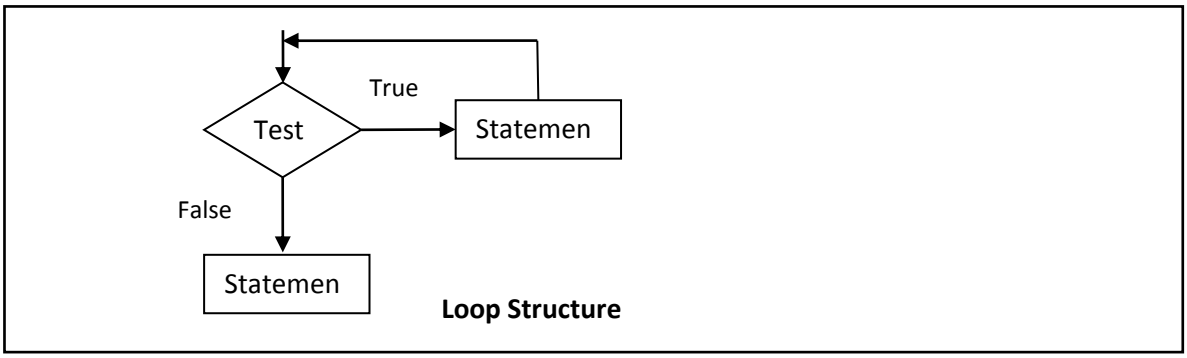


Selection or Conditional Structure: The Program has many conditions from which correct condition is selected to solve problems. These are (a) if-else (b) else-if, and (c) switch-case.



Repetition or loop Structure: The process of repetition or iteration repeats statements blocks several times when condition is matched, if condition is not matched, looping process is terminated.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.



Note:

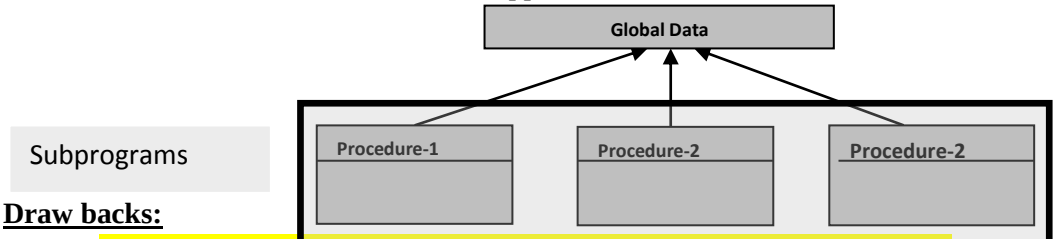
Almost all programming language are using the structured programming techniques to avoid common pitfalls of unstructured languages

Important point to remember: It is possible to do structured programming in any programming language, though it is preferable to use something like a procedural programming language.

Example: BCPL, CPL, COBOL, PASCAL, ALGOL.

III. Procedural Programming Language(POP): The important features of Procedural Programming Language are

- Programs are organized in the form of subroutine or functions or procedures
- In a POP method, emphasis is given to functions or subroutines or procedures.
- Functions are called repeatedly in a program to execute tasks performed by them.
- It follows the top down approach in program design.
- The state of a program is stored in variables. As the program execution goes on, the state changes. The actions (procedures or functions) will change the state (value of the variables).
- It is suitable for medium size s/w applications.



Draw backs:

- POP approach gives no importance to data. In C, a data member must be declared **GLOBAL** inorder to make it accessible by 2 or more functions in the program. What happens when 2 or more functions work on the same data member ? If there are 10 functions in a program, all these 10 functions can access a global data member. It is possible one function may accidentally change values of this global data member. If this data member is a key element of the program, any such accidental manipulation will affect the whole program.
- No data security.
- It will be too difficult to debug & identify which function is causing the problem if the program is really big(if the programming code goes beyond around 25,000 lines).
- In C we use a concept called structure, The problem with structures was that it handled only data. Structure do not allows to pack together associated functions inside it along with data. All the functions to manipulate data members inside structure has to be written separately inside a program. For this reason, a C program has an **over dependency on functions**.
- It doesn't model to real world problem very well.

Examples: C**, Ada, FORTRAN, Pascal.**

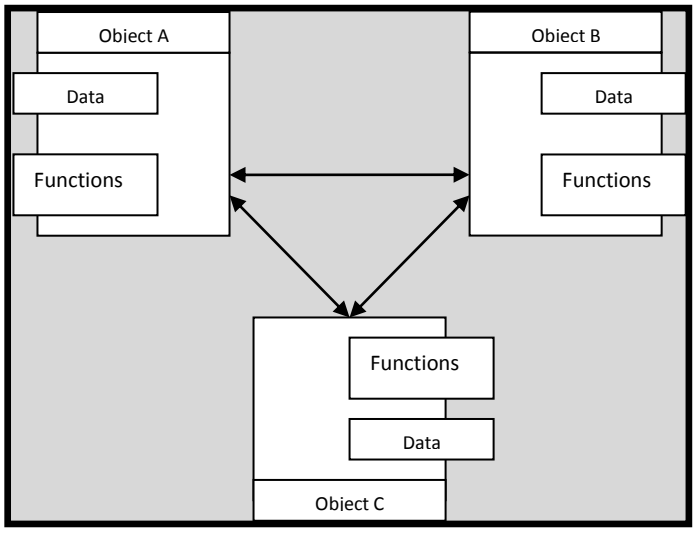
|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Note: C uses both Imperative (procedural), structured paradigms that is why C is called as both Structured and Procedural programming language. But if u asks me to say only one type, I prefer to say it is procedural programming language.

IV. Object Oriented Programming Language: OOP is developed by retaining all the best features of structured programming method/procedural method, to which they have added many concepts which facilitates efficient programming. Let's first keep in mind that OOP retains all best features of POP method like functions/sub routines, structure etc.

In Object oriented programming programs are divided into number of objects called modules which are easily manageable and provide many features. Some of those features are:

- programs are divided into number of modules(objects)
- Bind the data more closely to the functions
- It protects data from modifications from outside functions.
- Objects may communicate with each other through functions
- New data and functions can be easily added whenever necessary.
- Follows bottom-up approach in program design



Example: Simula, Smalltalk, C++, Objective-C, Eiffel, Python, Java, C#, Visual Basic .NET and REAL basic.

V. Object-based language: "object-based languages" supports all the Object Oriented Programming concepts except inheritance.

Example: Visual Basic (VB). VB supports both objects and classes, but not inheritance.

Top-Down and Bottom-Up Programming

What is Top-Down programming?

Top-Down programming is written from the top of a page down to the bottom, in logical order. It is a sequence that starts from very simple things and as it goes down, it gets more and more complicated. Characteristically, the things further down in the program depend on those above them.

Example: In our program if we have a statement called "x+y", before that statement you must declare x and y and their types and then assign with some values.

What is Bottom-Up programming?

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

It is the opposite to top-down approach. In this programming you start with a result and then declare what each part is.

Exmaple: in the below example i have used a, b, c variables in functions (methods) but I have declared those variables after the usage.

```
class  Caliculation
{          void add()
           {  c=a+b;
           }
           void sub()
           {c=a+b;
           }
           int a,b,c;
}
```

Java Platforms

Java technology is both a programming language, and a platform. Here platform means an environment on which java applications are run.

Note:

All Java platforms consist of a Java Virtual Machine (JVM) and an application programming interface (API).

Java is divided into 4 platforms

- **Java SE (JavaPlatform Standard Edition):** It provides CoreAPI, virtual machin, development tools, deployment technologies, and other class libraries and toolkits commonly used in Java technology applications.
- **Java EE (JavaPlatform Enterprise Edition):** Java EE platform is built on top of the Java SE platform It provides an API and runtime environment for developing and running large-scale, multi-tiered,scalable, reliable and secure network application.
- **Java ME (Java Platform Micro Edition):** It provides an API and small –footprint virtual machine for running java applications on small devices, like mobile phones. The API is a subset of JavaSE API, along with special class libraries usefull for small device application development.
- **Java FX:** JavaFX is a software platform for creating and delivering desktop applications, as well as rich internet applications that can run across a wide variety of devices.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

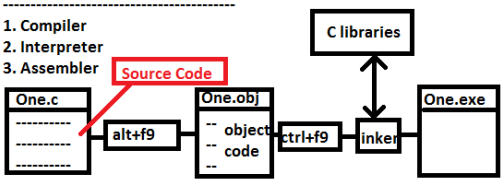
Categories of Programming languages, which are divided based on the applications we can build

1. Highlevel: We can develop application software
   Ex: COBOL
2. Low level: We can develop only system s/w.
   Ex: Machine Language, Assembly Language
3. Middle level: We can develop both Application and System s/w
   Ex: C, C++, Java etc..

If we write a program other than machine language, then we have to convert into machine language. to translate we have to use any one of the following translators.

There are 3 types of translators

1. Compiler
2. Interpreter
3. Assembler



alt+f9: This command invokes the compiler(translator), and this compiler only performs compilation, it means it checks whether the source code contains any syntax errors or not. if it contains errors then the compiler displays all the errors, otherwise then it translates the source code into object code. And this object code will be places in One.obj file.

ctrl+f9: it also invokes the compiler but it translates the object code into machine code, and that code will be placed in One.exe file.

Note: whatever the predefined libraries, which are used in our 'c' program are loaded before program execution at the time of .exe file creation, not at the time of program execution.

Note: c and c++ are also called as compiled programming languages. because, both these programming languages are using compiler translator to compile and run the program.

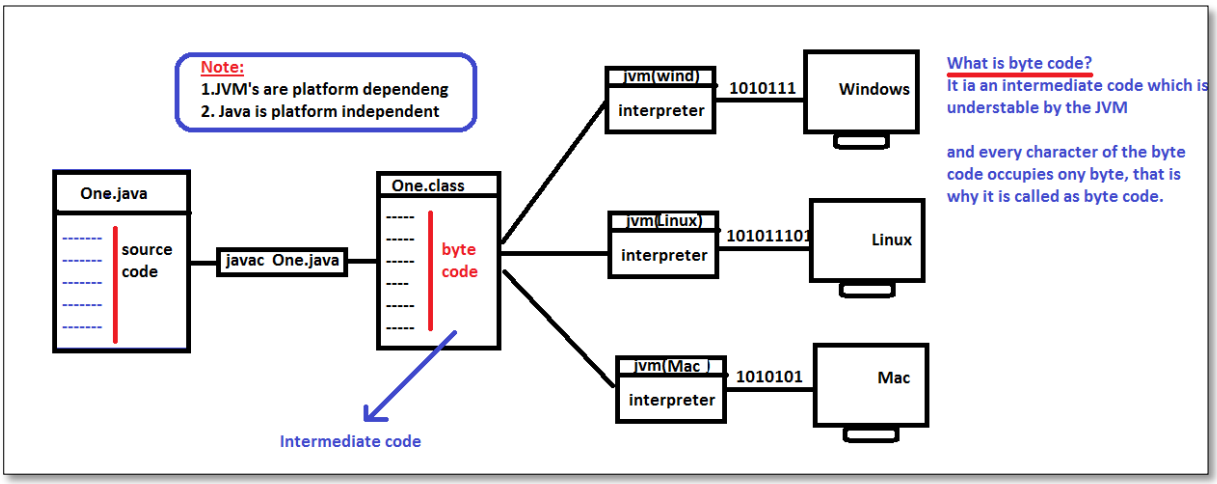
What is the difference between compiler and interpreter?

1. compiler	2. interpreter
1. compiler trnslates the total code at a time 2. translation is fast 3. by using compiler we can compile and run the programs 4. always puts the translated code in a seperate file. 5. it can generate .exe file 6. .exe files contains machine code so if we execute .exe files. because no need of translation execution will be fast.	1. It translates line by line 2. It is slow 3. only used to run the programs, by using it we can't compile the program 4. it directly gives the translated line to the processor. 5. It can't generate .exe file 6. at the time of program execution the source will be translated into line by line and executed.

Note:
1. c and c++ are called as compiled programming languages, because we use compiler (translator) to convert source code to machine code.

2. JavaScript, Html, Css are called as interpreted programming languages

3. Java is called as copiled and interpreted programming language.



What is byte code?
It ia an intermediate code which is understable by the JVM

and every character of the byte code occupies ony byte, that is why it is called as byte code.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

History and Features

Java History

In 1990's SunMicroSystem of USA started a secret project called "green", with a team lead by James Goslig, the team name is "green team".

Why the started?

To Invent a programming languages, which is used to develop s/w for electronic consumer devices.

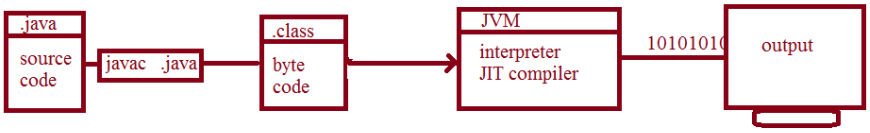
1. Java was invented in the year of 1991, the initial name of java is "oak".
2. in 1992 they invented a remote controller called "*7".
3. in 1995 first version of Java was introduced with a feature called "WORA"(Write Once Run Anywhere)
4. In 1995 "oak" was renamed as Java because of some legal problems.
5. 2010 jan 27th Java was purchased by Oracle company.

Oak: Tree name
Java: Iceland name(famous coffe beans)

Java: Just Another Virtual Approach

Java Features:

1. Java is compiled and interpreted Programming language.




```
for(int i=1;i<=100;i++)
{ out.println("Rajaaa...");
}
```

What is JIT(JustIntimeCompiler)?

It is a partial compiler, which translates only iterative statements into machine code. Especially introduced to make the java program execution faster.

2. Java is a platform independent programming language.

3. Java is an Object Oriented Programming Language

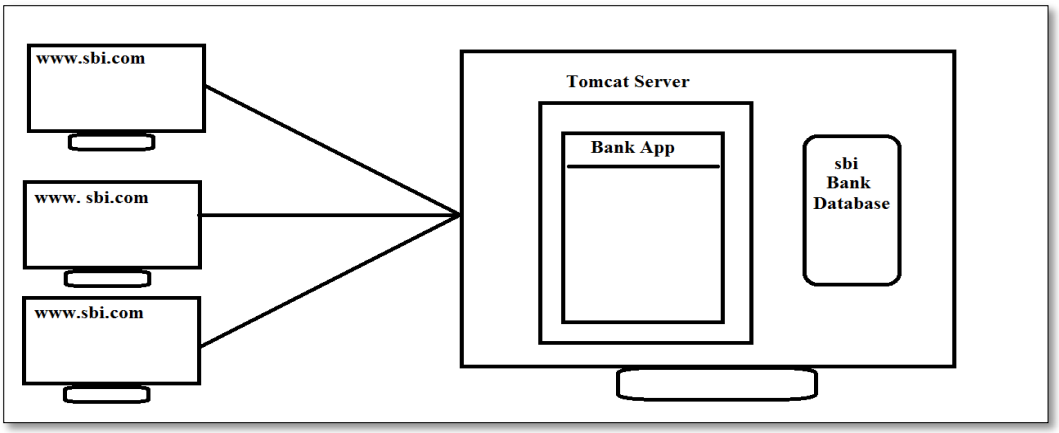
if a language supports the following 3 oop's concepts then it is called as Object Oriented Programming Language.

1. Data abstraction & Encapsulation 2. Polymorphism 3. Inheritance

4. Java is distributed and network supported programming language.

Java is called as distributed programming language, because we can develop distributed applications very easily in java.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.



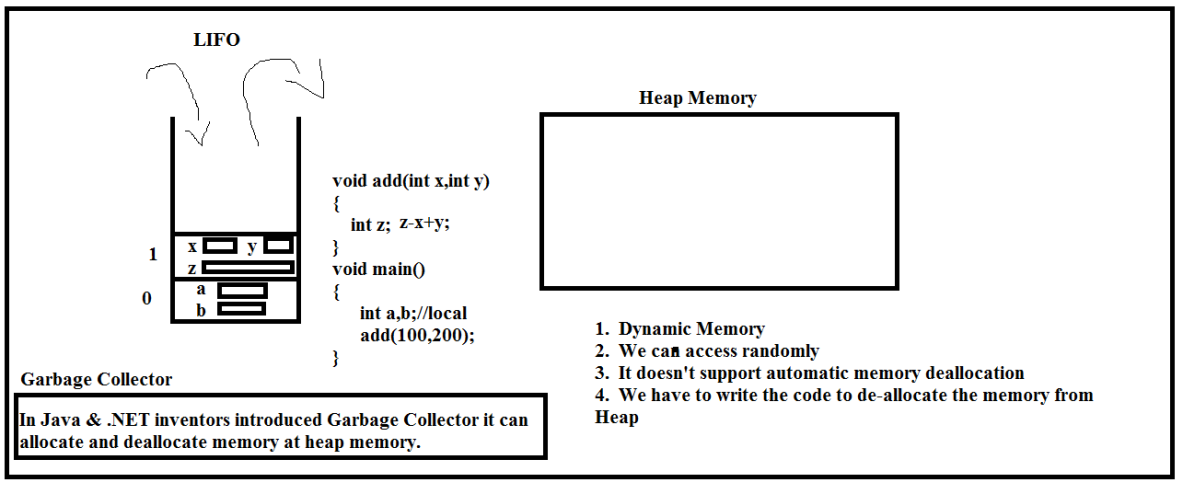
What is a distributed application?

It is an application through which we can distribute the data. See the above example

5. Java is robust and secure programming language.

Java is said to be a strongest programming language because the applications which are developed in java are not terminated or hang in middle of execution.

- It supports strict compile time checking
- It supports strict runtime checking
- Exception Handling: It is used to handle run time errors, if we handle runtime errors the program will not be terminated in middle of execution.
- Garbage Collector allocates and de allocates memory automatically.



6. Java is a multithreaded programming language: In Java we can perform more than one task at a time by using threads.

7. Java is simple and familiar language:

8. Dynamic and extensible: Java is said to be a dynamic programming language because whatever the libraries (functions, classes, etc...) Which are used in our program are loaded during program execution.

Java is an extendable programming language, because we can use the methods (functions) which are developed in other languages like 'C' and these methods are called as native methods.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Objecte Oriented Programming Concepts

What is Java?
Java is an Object Oriented Programming Language.

Java supports the following 3 oops concepts.
1. Data abstraction & encapsulation
2. Polymorphism
3. Inheritance

Before knowing about the above three oops concepts we have to know about what is a class and an object.

class:

1. class is a user defined or an abstract data type which contains, data(fields/variables) and methods(functions).
2. blueprint for objects
3. collection of objects
4. class is a template or a prototype for objects.

How to write a class?

we can write/define a class by using a keyword called class.

class
```
class Emp
{
}
```

interface
```
interface Emp
{
}
```

annotation
```
@interface Emp
{
}
```

enum
```
enum Emp
{
}
```

```
class Emp
{       int eno;
        String ename;
        float sal;
        void display()
        {       -----
                ----
                ----
        }
        void setEmp()
        {
        }
}
```

Why we write a class?

We write a class
1. to bind the data and methods into a block
2. to provide security to our data and metods.
3. to bind data with methods.

```
class NN
{       private String byke="Scooty";
        public String van="TataVan";
      public String nikhila()
     {  --------
        --------
        if(ok)
        {return byke;
        }else
        return null;
     }
}

class PN
{
      private String byke=null;
        void poojitha()
        {
                byke=nn.nikhila();
                 nn.van;
        }
}
```

What is an Object?

1. Object is nothing but instance of a class
2. Runtime entity.

What is the meaning of object creation?

Object creation is nothing but memory allocation for the instance fields of a class.

```
class Emp
{       int eno;
        String ename;
        float sal;
        void display()
        {}
        void setEmp()
        {
        }
}
```

We can create an object in 4 ways

1. by using new operator and constructor calling
2. by using newInstance() method
3. by using clone() method
4. by using factory() method.

1. Object creation by using new operator and constructor calling

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

What is an Object?

1. Object is nothing but instance of a class
2. Runtime entity.

What is the meaning of object creation?

Object creation is nothing but memory allocation for the instance fields of a class.

```
class Emp
{       int eno;
        String ename;
        float sal;
        void display()
        {}
        void setEmp()
        {
        }
}
```

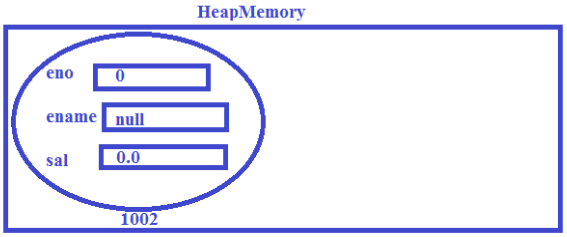
during object creation, these steps are executed.

1. new operator allocates memory.
2. constructor will be invoked, and it assigns default values to the fields
3. constructor returns the address of the object.

We can create an object in 4 ways

1. by using new operator and constructor calling
2. by using newInstance() method
3. by using clone() method
4. by using factory() method.

1. Object creation by using new operator and constructor calling




```
1002
Emp e1= new  Emp();
```

constructor calling statement.

new operator allocates memory dynamically

Reference variables

c lang ...code
int a=100;
int *p=&a;

Variables are divided into two types, based on the value they can store.

1. value type variables
2. reference variables.

Value type variables:

*. in these variables we can store only values not addresses.
*. to declare value type variables we can use the primitive/primary data types.
1. byte 2. short 3.int 4.long 5.float 6.double, 7. char 8. boolean

```
byte b=10;
short s=200;
char ch='h';
```

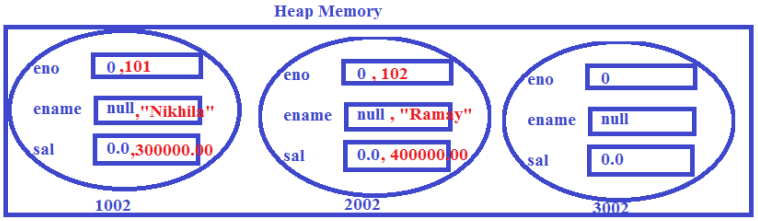
Refernece/Object variables.

*. It is a variable where we can store the address
*. We can declare a reference variable, by using

1. class names
2. interface names
3. annotation names
4. enum names
5. class name with combination of one or more subscripts, Ex: String []sarr;
6. primitive data type with combination of one or more subscripts, Ex: int []iarr; long sal[]; char ch[][];

```
class Emp
{
        int eno;
        String ename;
        float sal;
        void display()
        {------
        ------
        }
        void setEmp()
        {------
        ------
        }
}
```

1. new operator allocates memory
2. constructor will be invoked, and assigns default values.
3. constructor returns the address of the object




```
1002
Emp e1=new Emp():
e1.eno=101;
e1. ename="Nikhila";
e1. sal= 300000.00f;
```

e1 object initialization

```
Emp e2= new  Emp();
e2.eno=102;
e2.ename="Ramya";
e2.sal=400000.00f;
```

e2 object initialization

```
Emp e3=new Emp();
```

Object state: the data existed in the object
Object behaviour: the methods of the objects.
Object identity: The ref. Var, which is pointing to the object, is called Object identify.

Objects are independent: every object is independent to other, if u do modifications in one object it will not reflect to other objects

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Object Oriented Programming Concepts
1.Data Abstraction & Encapsulation
2. Polymorphism
3. Inheritance.

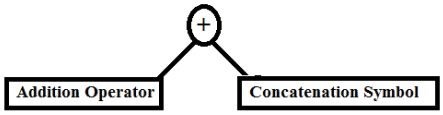
Data Abstraction: processof hiding unnecessary details from the end user.

```
public class Calc
{     private double mean()
      {     double mean;
            -------
            -------
            return mean;
      }
      public double getStandardDeviation()
      {     double sdn;
            mean();
            ---------
            ---------
            return sdn;
      }
}
```

```
class Usage
{
      void main()
      {
            Calc c=new Calc();
            c.getStanrdardDeviation();
      }
}
```

Encapsulation: Wrapping up of data and methods into a single unit(block) is called as encapsulation.
(or)
process of binding data with methods is called as encapsulation.

Polymorphism: existing in onr or more forms.
It is a greek word where poly means many morphs means forms.




```
int a=100,b=200;
int c=a+b;
```

```
String str="Madhu";
int n=420;
String res=str+n;
res="Madhu420";
```

```
class  MyMath
{
      int add(int a,int b)
      {
            int c=a+b;
            return c;
      }           10   20   30
      int add(int a,int b,int c)
      {
            int r=a+b+c;
            return r;
      }
      float add(float a,int b)
      {
            float r=a+b;
            return r;
      }
}
```

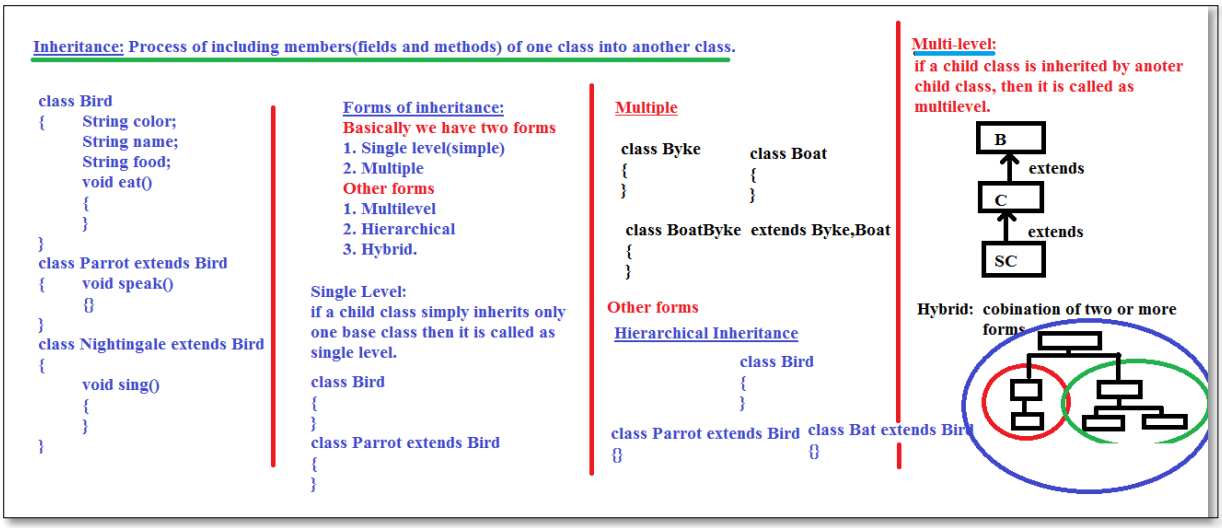
```
class Usage
{
      void main()
      {     MyMath mm=new MyMath();
                              60
            int  s1=  mm.add(10,20,30);
            float s2=mm.add(20.50f,30);
            int s3= mm.add(100,200);

      }
}
```

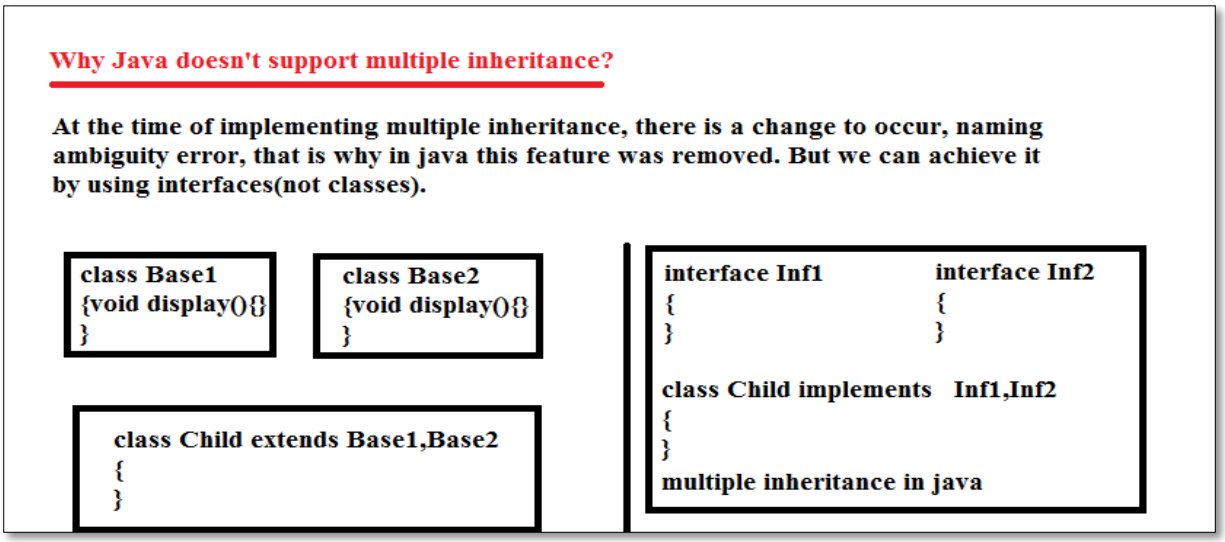
MethodOverloading:
We can write morethan one method with different signature in a class. It is called as method overloading.

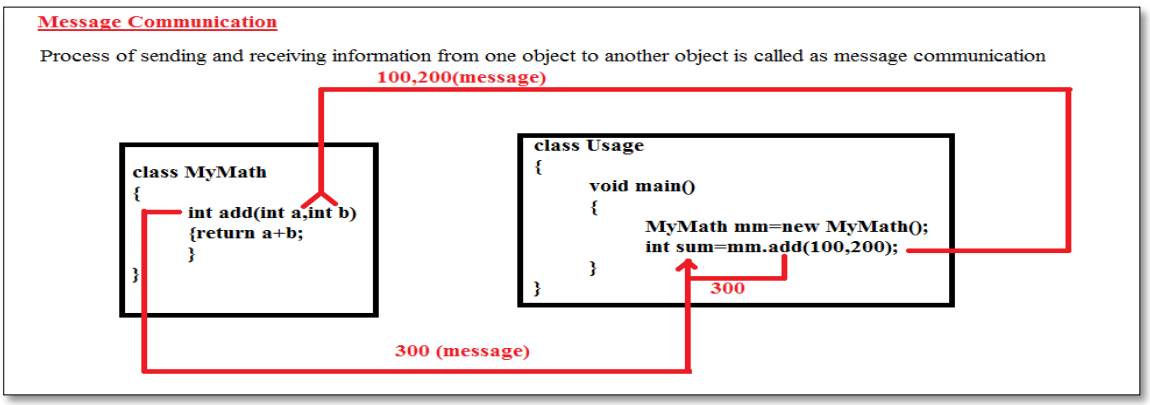
MethodSignature:
The method name and the count,type and order of arguments cobindly said to be a method signature.

These are calling
1. Method Overloading
2. Method Overriding
3. Operator Overloading

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.







First Program

1. Before writing first program we have to install java software.

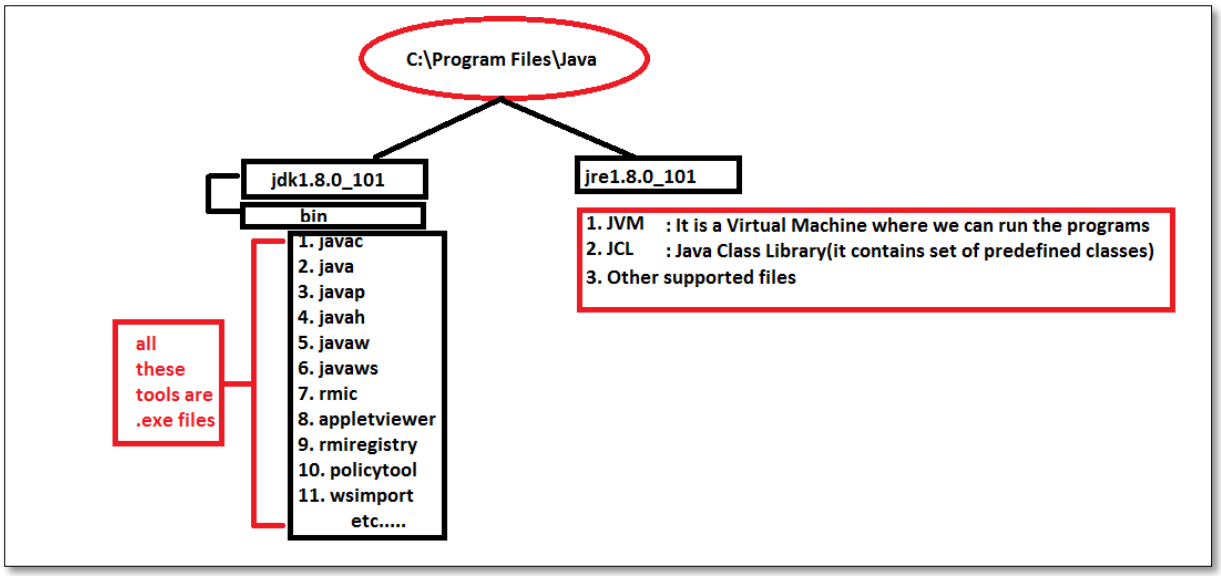
Steps to install

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

1. Download from http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Then double click the downloaded software and click on next button automatically the software will be installed in our computer.

Where the software will be installed?

1. It will be installed in the "C:\Program Files" folder



Interview Point of view question?
What is the difference between jdk and jre?

1. If we install JDK software then the set of tools will be installed in out computer and also JVM and JCL.
2. If we install JRE in our computer only JVM and JCL will be installed.

What is path?

Path is an environment variable, which is used to set the path for .exe files. If we set the path for .exe files then we can access them from anywhere within the computer.

In How many ways we can set the path?

1. There are two ways
    a. Temporary path setting
    b. Permanent path setting

Temporary path setting

If we set the path in command prompt then it is called as temporary. Because when we close the command prompt the path will be destroyed.

To see the already existed path in command prompt do the following

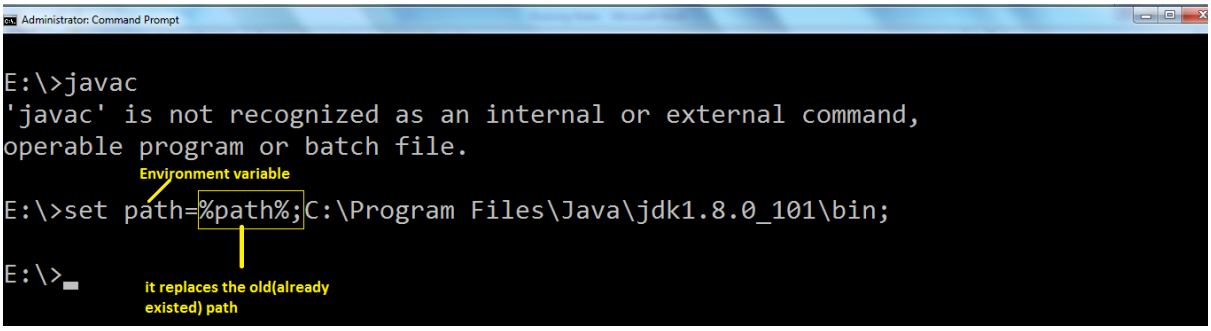
|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
E:\>notepad.exe

E:\>path
PATH=C:\ProgramData\Oracle\Java\javapath;c:\MinGW\bin;c:\MinGW\libexec\gcc\x86_6
4-pc-mingw32\6.1.0;C:\Users\NEW\gcc\bin;C:\Users\NEW\gcc\libexec\gcc\x86_64-pc-m
ingw32\6.1.0;F:\app\NEW\product\11.2.0\dbhome_2\bin;C:\Python34\;C:\Python34\Scr
ipts;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32
\WindowsPowerShell\v1.0\
```

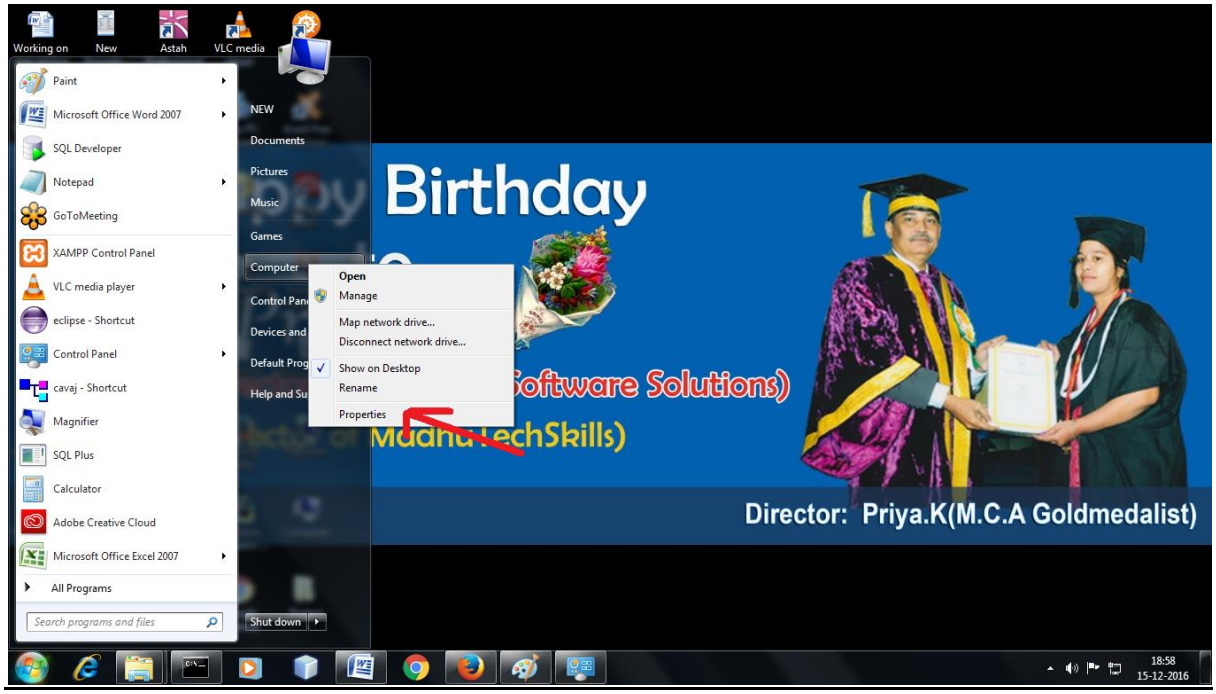
type the path press the enter

Temporary path setting

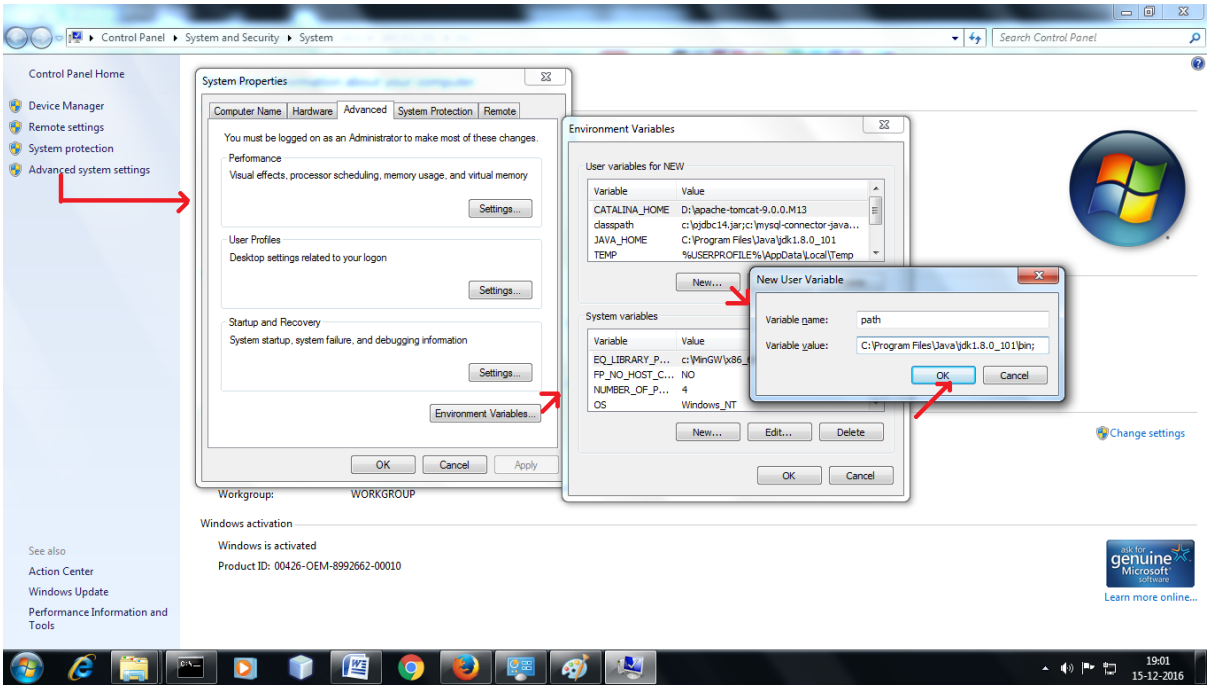
1. Open the command prompt set the path like below



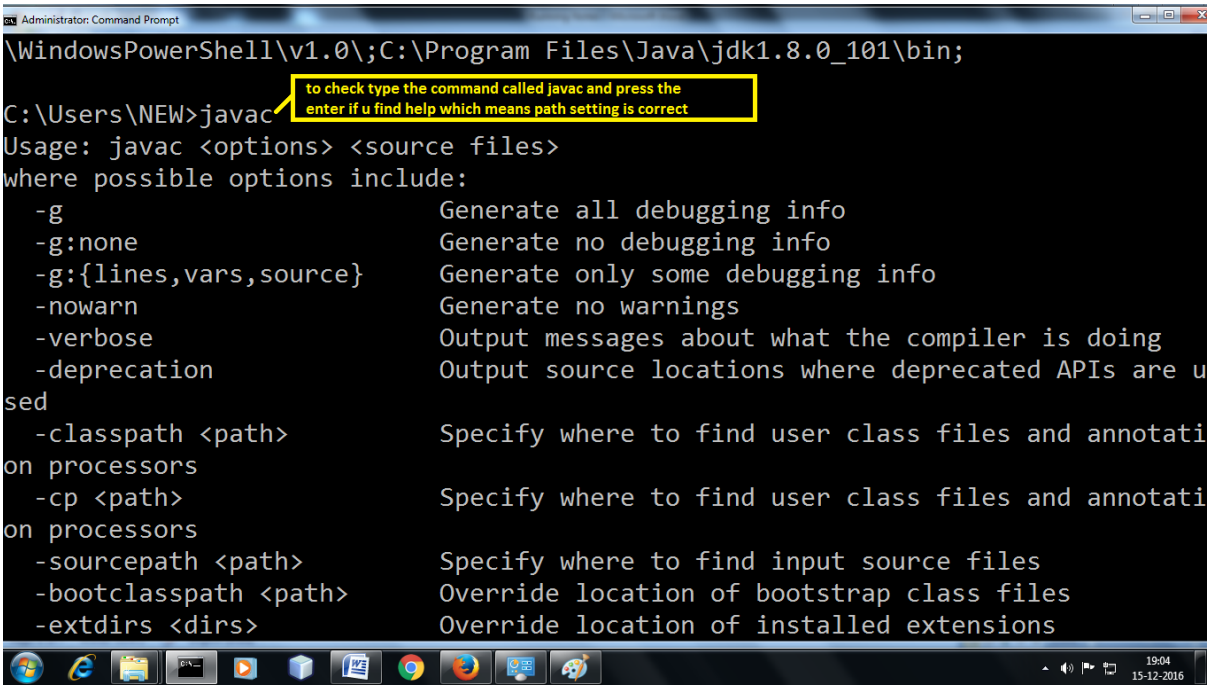
Permanent path setting



|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.



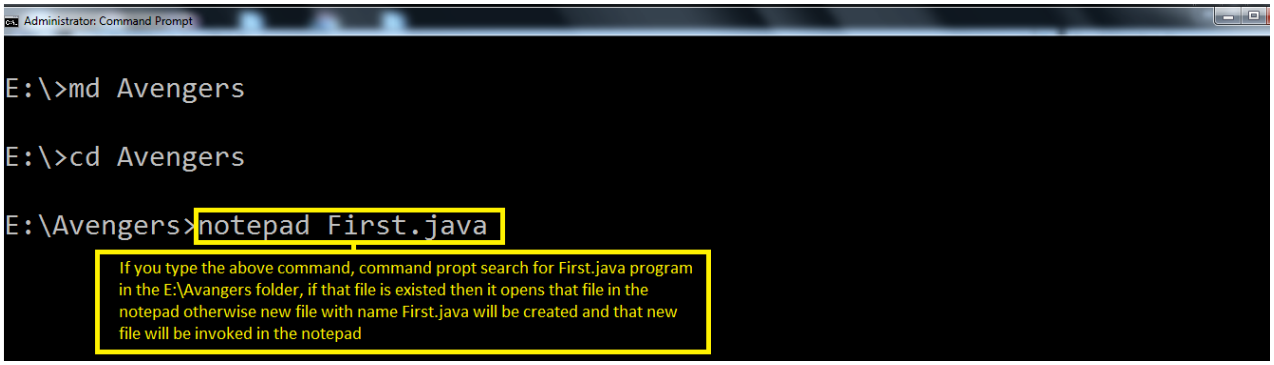
Note: close the already opened command prompt and open the new command prompt and check whether the path is perfectly set or not.



Writing First Java program

1. Create a separate folder with name “Avengers” in the E drive and give the below command to open the file (First.java).

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.



First.java

```
import java.lang.*;
class First
{       public static void main(String args[])
        {System.out.println("Welcome To Java");
        }
}
```

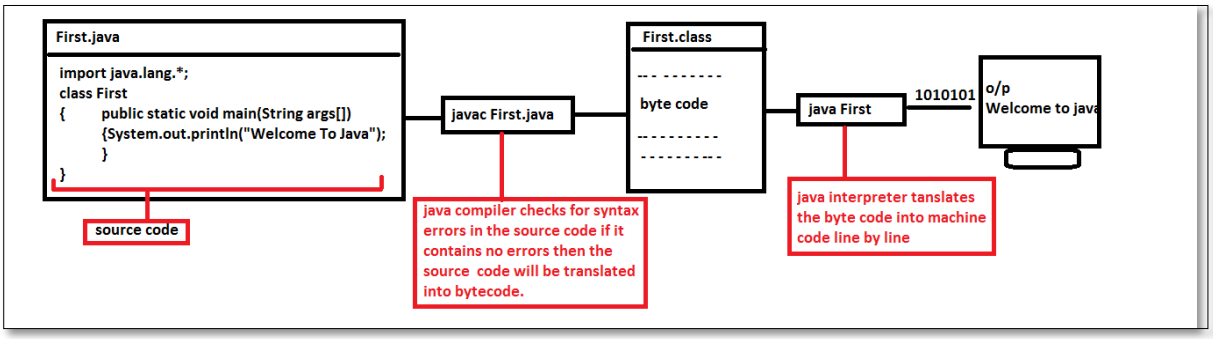
Compile the program

```
E:\Avengers>javac First.java
```

Run the program

```
E:\Avengers>java First
Welcome To Java
```

First Program Explanation:



|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Import:

1. import is a keyword which is used to import a package into our program, if we import a package into our program then we can use the classe and interfaces and annotations and enums of that package.
2. it is also used to import static members(stativ fields and methods) of a class into our program

What is a package?

It is a container for classes.
It is a container which contains collection of classes
a package is nothing but collection of classes
it is a container(folder) which contains collection of classes, interfaces, enums, and annotations.

Collection of predefined packages

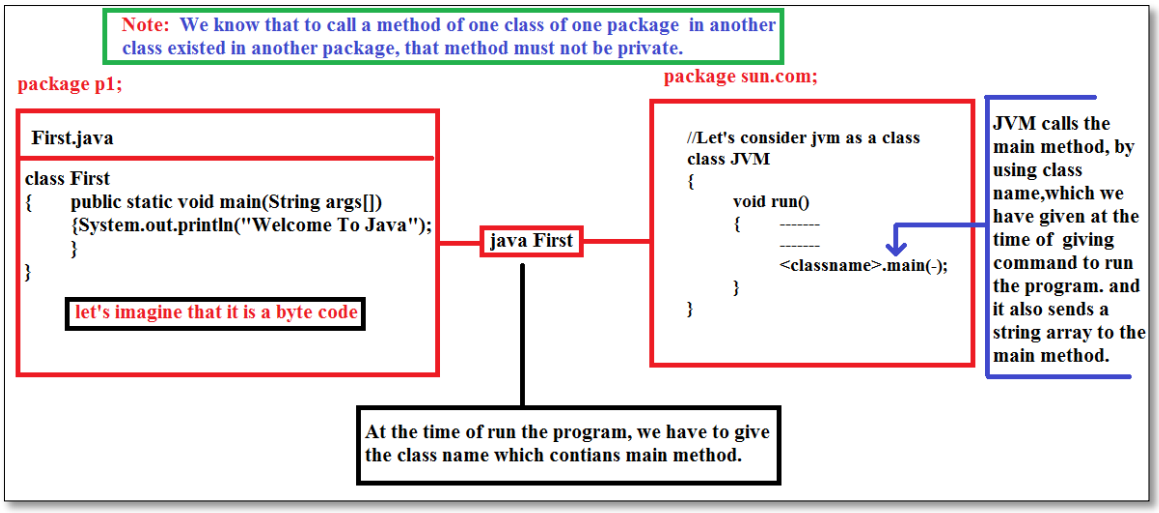
1. java.lang : it contains classes,interfaces,enums and annotatins which are usefull in every program
2. java.util
3. java.io
4. java.net
5. java.text
6. java.sql
7. java.applet

java.lang:
It is the default package, so we can use the classes, interfaces, enums and annotations of this package without importing it

String ,System:
These two classes are predefined, and existed in the java.lang package.

1. **Why the main method is public?**
    1. Public is an access modifier, which makes the main method unprotected.
    2. We have to declare the main method as public then only JVM can call the main method.

See the below image:



Modifiers: It is a keyword, which is used at the time of declaring fields, methods, classes etc... Modifiers fall into two categories:

1. **Access modifiers:**
    a. public
    b. protected
    c. private.
2. **Non-access modifiers**
    a. Static
    b. Abstract

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

c. strictfp,
d. final
e. volatile
f. native
g. synchronized
h. transient
i. assert

There are 4 four access control levels

1. **private:** Within the class
2. **package private (default)**: Within the package
3. **protected:** Within the package and child or sub child classes of other package
4. **Public:** in any package.

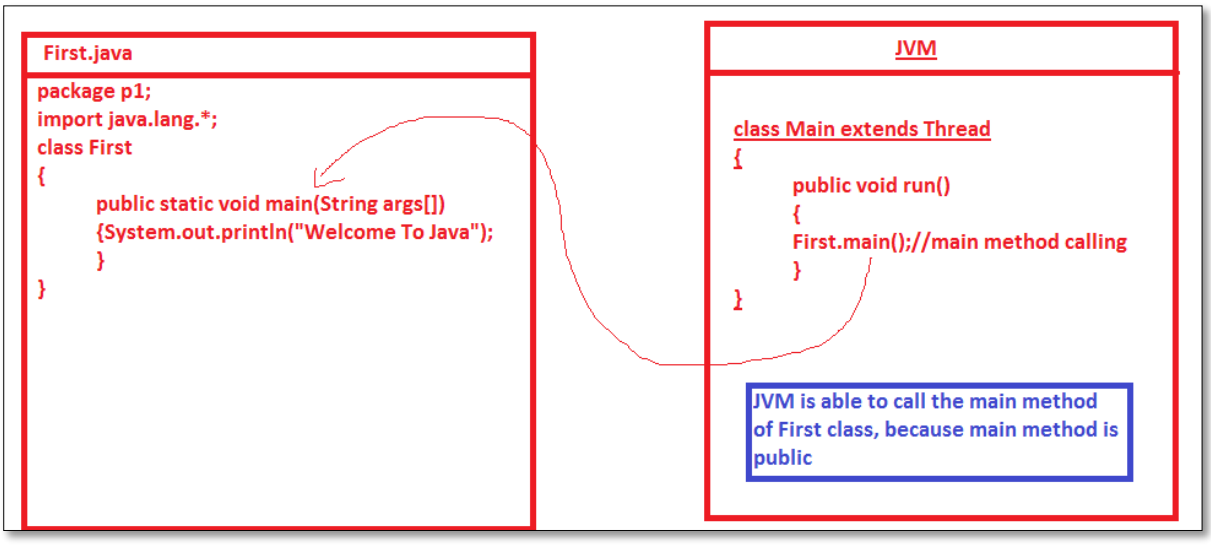
Access Controll Levels

```
package p1;
public class One
{       private int pri=100;
        int def=200;
        protected int pro=300;
        public int pub=400;
        public void display()
        {   pri   //(yes)
            def   //(yes)
            pro   //(yes)
            pub   //(yes)
        }
}
class Two extneds One
{   //def,pro,pub
    //display()
    public void display()
    {   pri   //no
        def   //yes
        pro   //yes
        pub   //yes
    }
}
```

```
package p1;
public class Three
{   One o1=new One();
    public void display()
    {o1.pri     //(no)
     o1.def     //(yes)
     o1.pro     //(yes)
     o1.pub     //(yes)
    }
}
```

```
package p2;
class Four extends p1.One
{   //pro, pub
    public void display()
    {   pri   //(no)
        def   //(no)
        pro   //(yes)
        pub   //(yes)
    }
}
class Five extends Four
{   //pro, pub
    public void display()
    {   pri   //(no)
        def   //(no)
        pro   //(yes)
        pub   //(yes)
    }
}
```

```
package p2;
class Sixth
{   p1.One o1=new p1.One();
    public void display()
    {   o1.pri     //(no)
        o1.def     //(no)
        o1.pro     //(no)
        o1.pub     //(yes)
    }
}
```



Why the main method is static?

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

1. Static is a non access modifier. We have to declare the main method as static then only JVM can call the main method directly by using class name, without any object creation.
2. JVM always calls the main method by using class name. So we have to declare the main method as static.

Example: usage of static non access modifier

```
class One
{     //non-static field/instance field/object
      int a;
      //static field/class field
      static int s;
      //non-static/instance/object method
      void display()
      {
      }
      //static method
      static void get()
      {
      }
}
```

```
class Two
{
      void call()
      {
            One obj=new One();
            System.out.println(obj.a)
            System.out.println(One.s);

      }
}
```

Note: we can access static fields/method in other classes directly by using class name.

Void: it is an empty data type. In Java main method always returns nothing so main method return type must be always declared as void.

What is String args [] in main method?

In Java, main method always takes single dimensional String array as an argument and it is used to handle command line arguments.

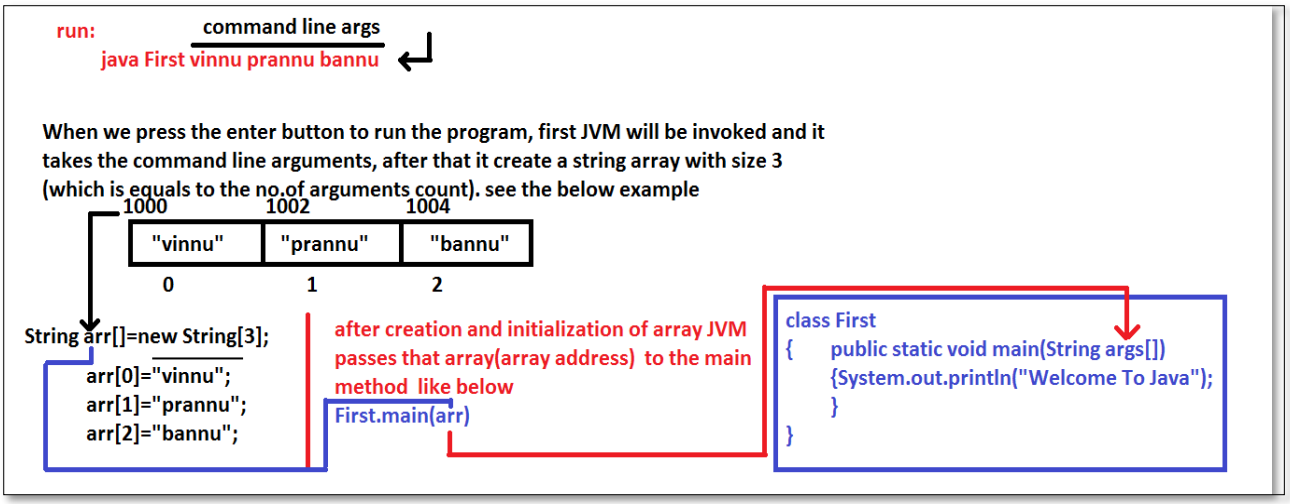
What are Command line arguments?

We can give some arguments at the time of giving command to run the program to the main method; through the command prompt those arguments are called as command line arguments.

What happens when we give command line arguments?

JVM takes those arguments, and creates a string array with the size which is equal to the no.of command line arguments.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.



CmdArgs.java

```
import java.lang.*;
class CmdArgs
{	public static void main(String args[])
	{	if(args.length==3)
		{
		System.out.println(args[0]);
		System.out.println(args[1]);
		System.out.println(args[2]);
		}else
		System.out.println("Enter 3 commad line arguements");
	}
}
```

Compile: E:\NUBatch>javac CmdArgs.java

Run: E:\NUBatch>java CmdArgs vinnu prannu bannu

```
vinnu
prannu
bannu
```

Length: It is an instance variable existed in every array in java. It returns the size of array.

CmdArgs.java

```
import java.lang.*;
class CmdArgs2
{	public static void main(String args[])
	{	System.out.println(args.length);
		System.out.println("Commang Line args.....\n----------");
		for(int i=0;i<args.length;i++)
		{	System.out.println(args[i]);
			//i++;
		}
	}
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Can a main method return any value?

No, it can't return any value

Can we write a return statement in main () method?

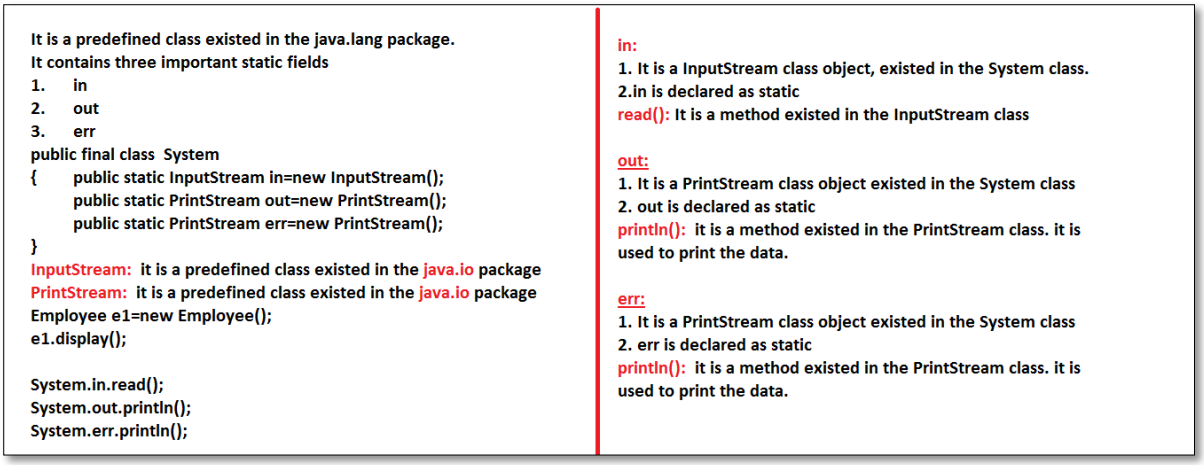
Yes, we can write a return statement which returns nothing. It only moves the cursor out of a method.

Example: Third.java

```
import  java.lang.*;
class Third
{       public static void main(String jilani[])
        {       System.out.println("Welcome to Java");
                if(true)
                return;
                System.out.println("Welcome to Java");
                /*      ----
                        ----
                        --
                */
        }
}
```

What is System?

It is a predefined class existed in the java.lang package. See the below image.



What is a stream?

1. Flow of data
2. It is a pipe, through which we can read/write the data from source to destination.

Note: Streams are unidirectional

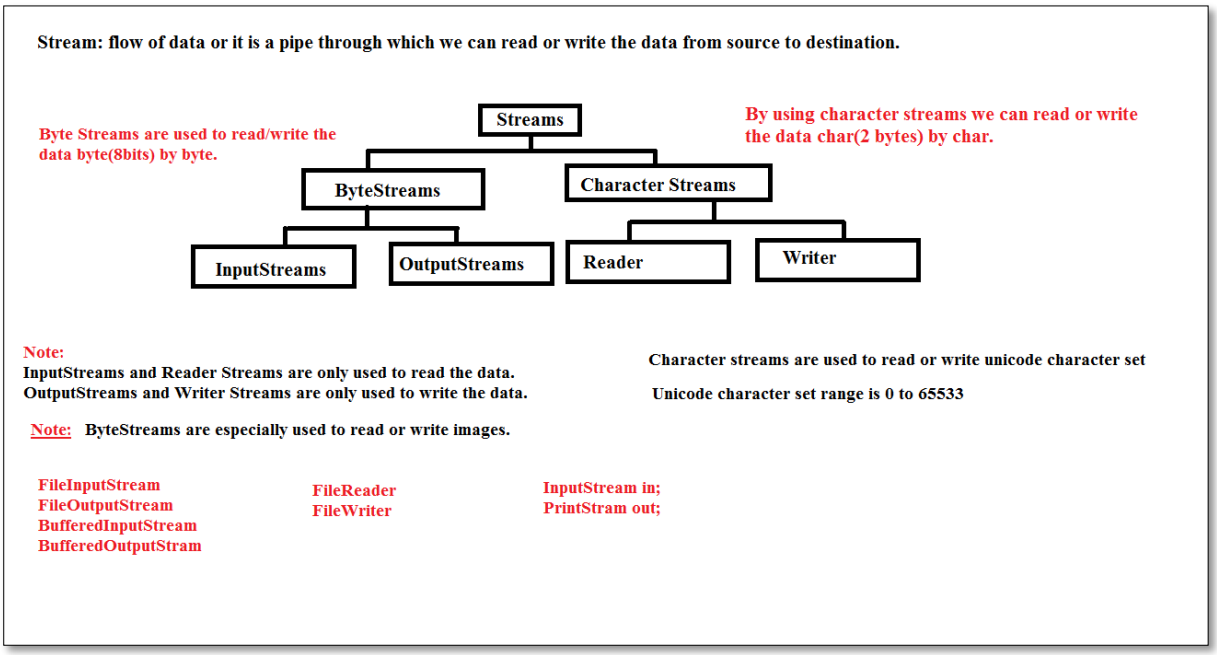
Note: In Java to read or write the data we have to use streams.

Where we can use streams:

We use stream

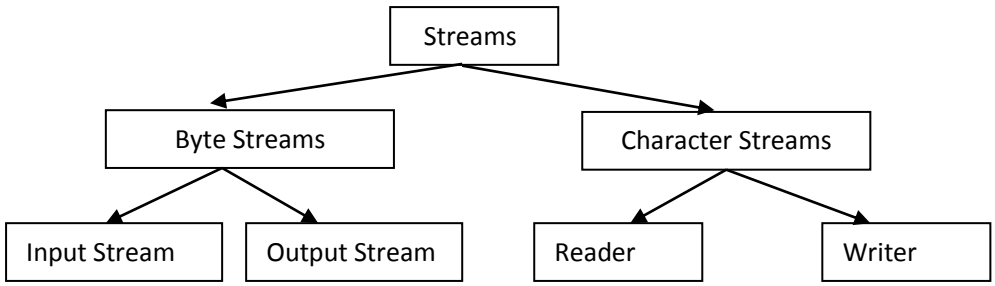
|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

1. To read the data from file
2. To write the data int to a file
3. To read the data from keyboard.
4. To Write the data to the console monitor etc....



Types of streams

There are two types of streams available in java. One is **ByteStreams** and **Character Streams**.



Byte Streams

Byte streams are used to read or write a non-unicode byte at a time. So we can't use these streams to read or write Unicode characters.

Note:

These streams are especially used to read or write images.

Character Streams

Character streams are used to read or write a Unicode character at a time. We can't use these streams to read or write images.

Note:

InputStreams and Reader Streams are used to read the data and OutputStreams and Writer streams are used to write the data.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

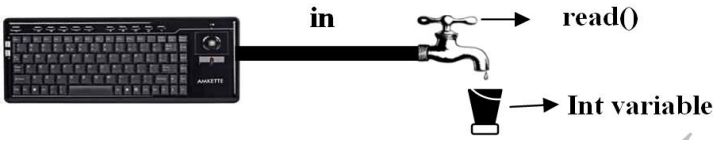
Read the data from keyboard:

To read the data from keyboard we need a stream which is logically connected to the standard input device (keyboard). Do we have that stream? Yes we have that stream in System class.

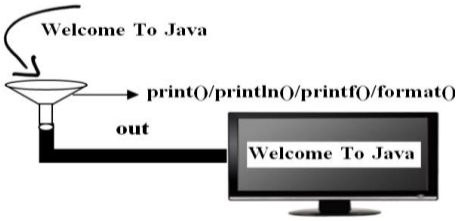
Do you know about System class?

System is a predefined class existed in the **java.lang** package, and it has three important fields. All these three fields are declared as static, those are given below:

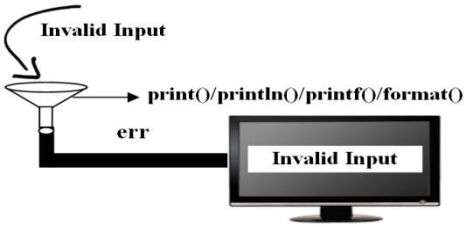
- **The in field:** It is an **InputStream** class object, which is logically connected to the standard input device, i.e, and keyboard. So by using this stream we can read the data given through the keyboard.



- **The out field:** It is a **PrintStream class** object, which is logically connected to the standard output device, i.e, and monitor. So by using this stream we can write the output to the standard output device.



- **The err field:** It is a **PrintStream class** object, which is logically connected to the standard output device, i.e, and monitor. So by using this stream we can write the error-messages to the standard output device.



Now I will read data using the stream called **"in"** observe the below example

Concatenation Symbol:

It is used to append anything to the string.

ConcatDemo.java

```
import  java.lang.*;
class ConcatDemo
{
        public static void main(String args[])
        {       String name="Madhu";
                int n=420;
                String res=name+n;
                //"Madhu420"
                System.out.println("Res:\t"+res);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                //"Res:\t"+"Madhu420"
                //"Res: Madhu420"
        }
}
```

ConcatDemo2.java

```
import java.lang.*;
class ConcatDemo2
{       public static void main(String args[])
        {
                int eno=101;
                String ename="BabyKiranmayee";
                String str="insert into emp values("+eno+",'"+ename+"')";
                        //"insert into emp values("+101
                        //"insert into emp values(101"+",'"
                        //"insert into emp values(101,'"+"BabyKiranmayee";
                        //"insert into emp values(101,'BabyKiranmayee"+"')";
                        //"insert into emp values(101,'BabyKiranmayee')";
                System.out.println("Str:\t"+str);

        }
}
```

CmdArgsAddition.java

```
import java.lang.*;
class CmdArgsAddition
{       public static void main(String[] args)
        {       if(args.length==2)
                {       int a,b;
                        a=Integer.parseInt(args[0]);
                        b=Integer.parseInt(args[1]);
                        int c=a+b;
                        System.out.println("c:\t"+c);
                }else
                System.out.println("Enter only two arguments");
        }
}
```

parseInt(): It is a predefined static method existed in the Integer class, it takes the string as an argument, and converts it into int and returns that int to us.

Integer: **It is a predefined class existed in the java.lang package**

CmdArgsAddition.java

```
import java.lang.*;
class CmdArgsAddition2
{
        public static void main(String[] args)
        {       if(args.length==2)
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                {       float a,b;
                        a=Float.parseFloat(args[0]);
                        b=Float.parseFloat(args[1]);
                        float c=a+b;
                        System.out.println("c:\t"+c);
                }else
                System.out.println("Enter only two arguments");
        }
}
E:\LocalNonLocal>javac CmdArgsAddition2.java
E:\LocalNonLocal>java CmdArgsAddition2 10.50 20.30
c:      30.8
```

CmdArgsAddition3.java

```
import java.lang.*;
class CmdArgsAddition3
{
        public static void main(String[] args)
        {       if(args.length==2)
                {       double a,b;
                        a=Double.parseDouble(args[0]);
                        b=Double.parseDouble(args[1]);
                        double c=a+b;
                        System.out.println("c:\t"+c);
                }else
                System.out.println("Enter only two arguments");
        }
}

E:\LocalNonLocal>javac CmdArgsAddition3.java
E:\LocalNonLocal>java CmdArgsAddition3  123.23 345.45
c:      468.68
```

Escape Sequences/Escape Character / Back slash codes: we can use escape sequences in the strings.

BackSlashCodes.java

```
import static java.lang.System.*;
class BackSlashCodes
{       public static void main(String args[])
        {       out.println("Madhu\tTech\tSkills");
                out.println("Madhu\nTech\nSkills");
                out.println("abcde\b\bxyz");
                //(\r)carriage return(it moves the cursor to the first char of the line)
                out.println("abcde\rxyz");
                out.println("\"Hai\"");
                out.println("\'Hai\'");
                out.println("'Hai'");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
            out.println("27\\10\\1980");
        }
}
```

Static import: static import is used to import static members of a class into our program, if we import static members then we can used them directly without using class name.

What is the difference between print() and println() methods.

1. Print method takes only one argument
2. Print method doesn't move the cursor to the new line, after printing the data
3. Println() method takes 0 or 1 argument
4. Println () method moves the cursor to the new line, after printing the data.

PrintPrintln.java

```
import static java.lang.System.*;
class PrintPrintln
{       static public void main(String args[])
        {       out.print("MadhuTechSkills");
                out.print(",Vijayawada");
                out.println();
                out.println("MadhuTechSkills");
                out.print(",Vijayawada");
                out.println(10,20);
        }
}
```

What is Naming Convention?

A naming convention is a rule to follow at the time of writing the names of package, classes, interfaces, enums, methods, variables. Let us see the below table.

Java Element	Convention	Example
package	Packages are written in small letters	java.lang,java.io,java.util.
class,interface ,enum & annotation	Each word first letter must be a capital letter.	String, DataInputStream.
Method	The first word of the method name must be in small letters, then from the second word onwards each word first letter must be capital	readLine(), print(), getInputStream().
Variables	Same as method naming convention except method names ends with pair of paranthesis.	empName, empNo.
Constants	Constants are always in capital letters only	MAX_PRIORITY, MIN_PRIORITY.
Keywords	In Java keyword is always in single word, and is in small letters only.	void,int,abstract.

Other Important Points To rember at the time of writing a program:

- In a program we can write any number of classes.
- We can write only one public class in a proram(.java file).

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- If we write a public class in a program, public class name file name must be same.
- We can write any number of main methods in a program.
- At the time of program compilation give the program name with extension.
- At the time of run the program give the .class file name which contains main method.

Many.java

```
import static java.lang.System.*;
public  class Many
{       public static void main(String[] args)
        {out.println("Many main method..");
        }
}
class First
{       public static void main(String[] args)
        {out.println("First main method..");
        }
}
class Second
{       public static void main(String[] args)
        {out.println("Second main method..");
        }
}
Run:
E:\LocalNonLocal\examples>javac  Many.java

E:\LocalNonLocal\examples>java First
First main method..

E:\LocalNonLocal\examples>java Many
Many main method..

E:\LocalNonLocal\examples>java Second
Second main method..
```

SingleClassMoreMains.java

```
import static java.lang.System.*;
class SingleClassMoreMains
{
        public static void main(String args[])
        {       out.println("Main method...");
                main();
        }
        public static void main()
        {       out.println("Hai...");
        }
}
```

What are formatting() methods:

In jdk1.5 version two new methods are introduced, to replace the print() and println() methods, and these methods are called as formatting methods.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Those are:

1. Printf()
2. Format()

These methods are existed in the java.io.PrintStream class and these methods, format and printf, are equivalent to one another.

What is the difference between printf() and format() method?

Actually **printf()** method is internally calls the **format()** method, that is actual code is existed in the format method. The printf() method is just provided for our convenience.

Some of the converters which are used in format method

Converter	Flag	Explanation
d		A decimal integer
f		A float
n		A new line character appropriate to the platform running the application. You shold always use %n rather than \n
tB		Full name of month
tb		Short name of month
td		day of month, if it is single digit, it puts zero before that number
te		Day of month, it will not put zero before single digit.
ty		2 digit year
tY		4 digit year
tl		Hour in 12 hours clock
tM		Minutes in 2 digits, with leading zeros as necessary
tS		Seconds
tp		Am/pm
tm		Minutes in 2 digits, with leading zeros as necessary
tD		Date as %tm%td%ty
	08	8 characters in width, with leading zeros as necessary
	+	Includes sign, whether positive or negative
	,	Includes Locale-specific grouping characters.
	-	Left-justified
	.3	Three places after decimal part
	10.3	Ten characters in width, right justified, with three places, after decimal point

Printf():

It is existed in the PrintStream class, it takes 1 or more arguments.

FormattingMethods1.java

```
import static java.lang.System.*;
class FormattingMethods1
{
	public static void main(String args[])
	{	int a=10,b=20;
		int c=a+b;
		out.printf("%d+%d=%d %n",a,b,c);
		out.printf("%d*%d=%d",a,b,(a*b));
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        }
}
```

FormattingMethods2.java

```
import static java.lang.System.*;
class FormattingMethods1
{       public static void main(String args[])
        {       int a=10,b=20;
                int c=a+b;
                out.printf("%d+%d=%d %n",a,b,c);
                out.printf("%d*%d=%d",a,b,(a*b));
        }
}
```

FormatDemo3.java

```
import static java.lang.System.*;
class FormatDemo3
{       public static void main(String args[])
        {       long n = 261011;
                out.format("%d%n",n);      //  -->  "261011"
                out.format("%09d%n",n);    //  -->  "000261011"
                out.format("%9d%n",n);     //  -->  "   261011"
                out.format("%,9d%n",n);    //  -->  "  261,011"
                out.format("%+,9d%n%n",n); //  -->  " +261,011"
        }
}
```

FormatDemo4.java

```
import static java.lang.System.*;
class FormatDemo4
{       public static void main(String args[])
        {       double pi =3.141593;
                out.format("%f%n", pi);       // -->  "3.141593"
                out.format("%.3f%n", pi);     // -->  "3.142"
                out.format("%10.3f%n", pi);   // -->  "     3.142"
                out.format("%-10.3f", pi);  // -->  "3.142     "// left justified
                out.printf("Madhu Tech Skills...");
        }
}
```

FormatDemo5.java

```
import static java.lang.System.*;
import java.util.*;
class FormatDemo5
{
        public static void main(String args[])
        {       Calendar c = Calendar.getInstance();
                out.format("%tB %td, %tY%n", c, c, c); // -->  "July 09, 2013"
                out.format("%tb %te, %tY%n", c, c, c); // -->  "July 9, 2013"
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.format("%tl:%tM %tp%n", c, c, c);  // -->  "2:34 am"
                out.format("%tD%n", c);    // -->  "07/09/13"//month/day/year
        }
}
```

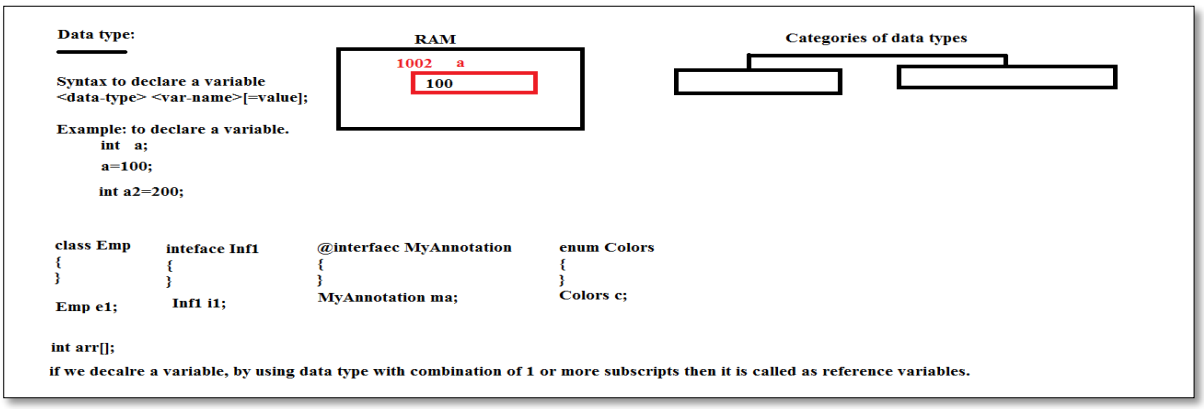
FormatDemo6.java

```
import static java.lang.System.*;
import java.util.*;
class FormatDemo6
{       public static void main(String args[])
        {       Date sd=new Date();
                out.println("SystemDate:\t"+sd);
                out.format("%td-%tB-%tY%n",sd,sd,sd);
                out.format("%te-%tb-%ty%n",sd,sd,sd);
                out.printf("%tl:%tM:%tS    %tp",sd,sd,sd,sd);
        }
}
```

Data Types

What is Data Type?

Data type is a keyword or a class name or an interface name or enum name or annotation name which is used to tell the JVM that how much memory has to allocate for a variable, and what type of data that variable can store.



Categories of Data types

- Primary data types
- Reference or Object data types

Primary data types

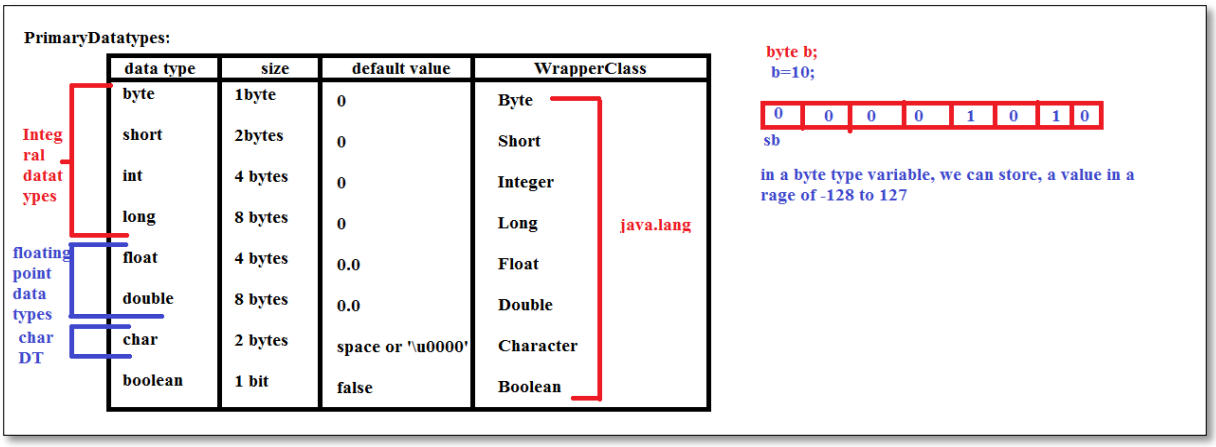
There are 8 primary data types available in java, those are divided into 4 categories

1. Integer(integral) data types
    1. Byte
    2. Short

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

3. Int
4. long

2. Floating point data types
   1. Float
   2. double
3. Char
   1. char
4. Boolean
   1. boolean



Signed data types: variables of these types can store either negative (-ve) values or positive values

- **byte**
- **short**
- **int**
- **long**
- **float**
- **double**

Unsigned data types: variables of these types can store only positive values

- **char**
- **boolean**

SignedUnsigned.java

```
import static java.lang.System.*;
class SignedUnsigned
{       public static void main(String args[])
        {
                //singned data types
                byte b1=100,b2=-100;
                short s1=100,s2=-100;
                int i1=100,i2=-100;
                long l1=100l,l2=-100l;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                float f1=10.20f,f2=-10.20f;
                double d1=10.20,d2=-10.20;

                //Unsigned data types
                char ch1='a';
                boolean bl1=false;
                boolean bl2=true;
                out.println("Signed.....");
                out.println("b1="+b1+", b2="+b2);
                out.println("s1="+s1+", s2="+s2);
                out.println("i1="+i1+", i2="+i2);
                out.println("l1="+l1+", l2="+l2);
                out.println("f1="+f1+", f2="+f2);
                out.println("UnSigned.....");
                out.println("ch1="+ch1);
                out.println("bl1="+bl1+", bl2="+bl2);
        }
}
```

Output

```
E:\LocalNonLocal>java SignedUnsigned
Signed.....
b1=100, b2=-100
s1=100, s2=-100
i1=100, i2=-100
l1=100, l2=-100
f1=10.2, f2=-10.2
UnSigned.....
ch1=a
bl1=false, bl2=true
```

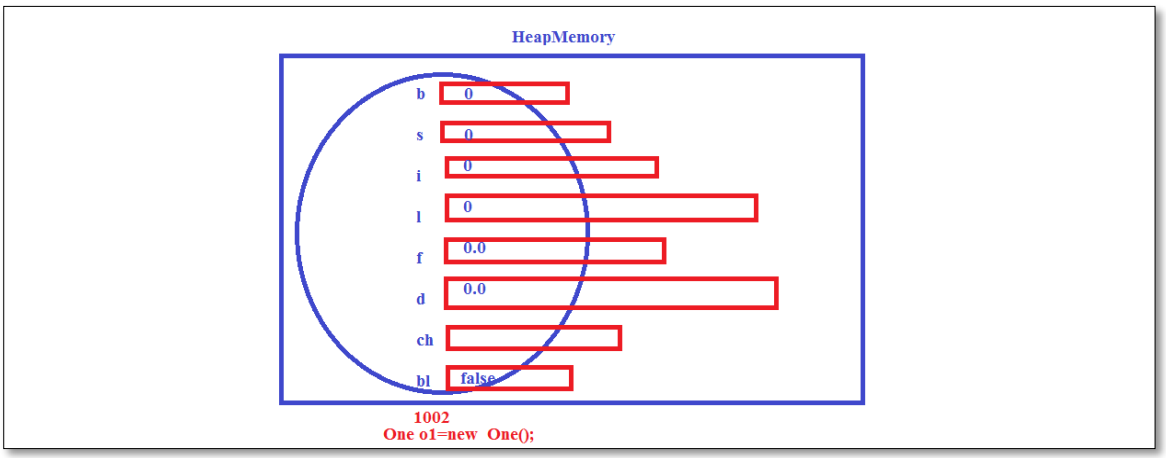
DataTypes.java

```
import static java.lang.System.*;
class One
{       byte b;
        short s;
        int i;
        long l;
        float f;
        double d;
        char ch;
        boolean bl;
        void display()
        {       out.println("Object state...");
                out.println("b:\t"+b);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("s:\t"+s);
                out.println("i:\t"+i);
                out.println("l:\t"+l);
                out.println("f:\t"+f);
                out.println("d:\t"+d);
                out.println("ch:\t"+ch);
                out.println("bl:\t"+bl);
        }
}
class DataTypes
{       public static void main(String args[])
        {       One o1=new One();
                o1.display();
        }
}
```

Object o1 view in heap memory



CharDemo.java

```
import static java.lang.System.*;
class CharDemo
{       public static void main(String args[])
        {       char ch1='a';
                char ch2=97;
                char ch3=0b1100001;
                char ch4=0x61;
                char ch5=0141;
                //max value we can give '\uffff'=65535
                char ch6='\u0061';
                out.println("ch1:\t"+ch1);
                out.println("ch2:\t"+ch2);
                out.println("ch3:\t"+ch3);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("ch4:\t"+ch4);
                out.println("ch5:\t"+ch5);
                out.println("ch6:\t"+ch6);
        }
}
```

Reference/Object data types

- class name.
- interface name.
- annotation name.
- enum name.
- primitive data type with combination of one or more subscripts.
- class name with combination of one or more subscripts.

RefDataTypes.java

```
import static java.lang.System.*;
class Emp
{       int eno;
        String ename;
        float sal;
        void display()
        {out.println("Emp details.....");
        out.println("Eno:\t"+eno);
        out.println("Ename:\t"+ename);
        out.println("Sal:\t"+sal);
        }
}
interface Inf1
{}
@interface MyAnnotation
{}
enum Colors{RED,GREEN,BLUE}
class RefDataTypes
{       public static void main(String args[])
        {       Emp e1=new Emp();
                e1.eno=101;
                e1.ename="Ramakanth";
                e1.sal=50000.00f;
                e1.display();
                Inf1 i1;
                MyAnnotation ma;
                Colors c;
                Emp  empArr1[];
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        Emp  empArr2[][];
        int []arr1;
        int [][]arr2;
    }
}
```

Tokens: Each and every individual unit in a program is called as token.

- **Keywords**
- **Identifiers**
- **Seperators**
- **Literals**
- **Operator**
- **Comments**

Keywords: keyword is a predefined word which has an implicit meaning. In java we have 50 keywords.

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

Is true,false and null are keywords?

No true, false, and null might seem like keywords,but not keywords, they are actually literals; you cannot use them as identifiers in your programs.

What is Identifier?

Identifier is a name given to variables, methods, classes, interfaces, packages and Enums.

Rules to follow at the time of define an Identifier

- Can't start with number
- It can start with any letter of the alphabet, it can even start with underscore or dollar sign.
- It should not be a keyword
- It is case sensitive
- Java identifier is unlimited in terms of length, it also says that a Java identifier is any combination of Unicode chars that means if your editor supports Telugu or Cyrillic or any other Unicode character set, you're good to go too.
- Don't use special characters in an identifier
- Don't use space in an identifier

Example: Identifier.java

```
import static java.lang.System.*;
class Identifier
{       public static void main(String args[])
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
{       int a;
        int a1;
        //int 1a; error
        int $a;
        int _a;
        //int goto;         error
        int A;
        int Veeeraaaaaabhadraaaanaaareeeeesssssshhhhh=20;
        //int a.b; error
        //int emp no; error
}
}
```

What is a variable?

A variable is a name which represents a storage location, where we can store a value. The value of variables can be changed.

How to declare a variable?

To declare a variable in Java, all that we need is the data type followed by the variable name. see the below example.

Ex: **int n;**

There are three types of variables we have in Java

Instance variables (Non-static variables):

It is a variable(field) which is not declared as static in a class, Objects store their individual states in instance fields. Instance variables are created each time a new object is created. If We want to access instance variable of an object in other classes directly , we have to use the object(instance).

```
Ex: class Demo
    {       int a,b;//instance variables
    }
```

Static variables (Class variables):

It is a static variable which is declared in a class. To declare a class variable we have to use static modifier, a static variable can be accessed by all the objects of the class. That is, there will be only one copy of the variable that is shared or accessed by all the objects. Static variable are created(memory allocation) at the time of class loads into the memory.

```
Ex: class Demo
    {       static int a,b;//static or class variables
    }
```

Local variables:

It is a variable which is declared within a block which is existed in a class. a method stores its temporary state in local variables. The scope of the local variable is within the method or block in which it is declared. That is, other methods cannot use. Local variables are only visible to the methods or block in which they are declared.

```
Ex: class Demo
    {    void display(){
             int a,b;//local variables
         }
    }
```

Variables.java

```
import static java.lang.System.*;
class One
{       //instance variable
        int a;
        //static variable
        static int s=100;
        int add(int a,int b)
        {int c=a+b;
        return c;
        }
}
class Variables
{       public static void main(String args[])
        {       One o1=new One();
                int sum=o1.add(10,20);
                out.println("Sum:\t"+sum);
        }
}
```

What is method state?

The values stored in the variable declared within the method are called as method state.

MethodState.java

```
import static java.lang.System.*;
class One
{       int add(int a,int b)
        {int c=a+b;
        return c;
        }
}
class MethodState
{       public static void main(String args[])
        {       One o1=new One();
                //the add method state is 100,200,300
                out.println("Sum:\t"+o1.add(100,200));
        }
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

What is a constant?

A constant is an identifier whose value will not be changed through the program execution. In java we can declare a constant by using **"final"** keyword.

How to declare a constant variable?

```
Ex:    final  int n;
```

Example: Constant.java

```
import static java.lang.System.*;
class Constant
{       public static void main(String args[])
        {       final int a=100;
                out.println("a:\t"+a);
                a=20;
        }
}
```

Compile: javac Constant.java

```
D:\Hi-flyers>javac Constant.java
Constant.java:7: error: cannot assign a value to final variable a
            a=20;
            ^
1 error
```

When we assign a value to the final instance variable?

Either at the time of declaration or at the time of constructor calling

Note: if we declare the static and local variables as final, we have to initialize them at the time of declaration only.

What is Literal?

A literal is a constant value used in a program. There are various literals in Java, Those are explained below.

Usage:

```
Byte b=10; int i=20; short s=30;Boolean b=false;
```

In the above example 10, 20, 30, and false are literals. There are different types of literals available in java those are

- **Integer literals:** integer literals is nothing but fixed integer values like 100, 234, -34, etc. all these numbers belong to decimal system, which uses 10 digits (from 0 to 9) to represent any number. Suppose, we want to write an integer in octal number system (octal number system uses 8 digits, from 0 to 7), then we should prefix o before the number. To represent hexadecimal number (hexadecimal number system uses 16 digits from 0 to 9 and from A to F), we should prefix ox before the value

IntLit.java

```
import static  java.lang.System.*;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
class IntLit
{       final public static void main(String arg[])
        {
         int dval=26;   // it is decimal
         int oval=032; // it is octal for the number 26(zero is the prefix for octal)
         int hval=0x1a; // it is hexadecimal of 26(zero and x is the prefix for hexadecimal)
         int bval=0b00001010; // it is binary value of 10(zero and b is the prefix for binary)
         out.println("dval:\t"+dval);
         out.println("oval:\t"+oval);
         out.println("hval:\t"+hval);
         out.println("bval:\t"+bval);
        }
}
```

- **Float literals:** Float literals represent fractional numbers. These are the numbers with decimal points like 12.23, 2.3, -1.222, etc. which used with float or double type variables. While writing these literals, we can use E or e for scientific notation, F or f for float literal, and D or d for double literal.

FloatLit.java

```
import static  java.lang.System.*;
class FloatLit
{       final public static void main(String arg[])
        {double d1=123.4D;
        double d2=0.1234e3;  //  same value as d1, but in scientific notation
        float f1=123.4F;
        out.println("d1:\t"+d1);
        out.println("d2:\t"+d2);
        out.println("float:\t"+f1);
        }
}
```

- **Character literals:** character literals must be enclosed in single quotation marks.Character literals indicate the following:

```
General characters, like A, b, 9, etc.
Special characters, like ?, @, etc.
Unicode characters, like "\u0042" (\u hexa decimal value) (this represents a in ISO latin 1 character set).
Escape sequence \n, \t, etc.
```

- **String literals:** String literals must be enclosed in double quotations for example "Madhu Tech Skills", "Madhu", "Priya", "James Bond007" etc.
- **Boolean literals:** Boolean literals represent only two values – true or false.

Example: CharStringBool.java
Literals.java

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
import static java.lang.System.*;
class Literal
{	public static void main(String args[])
	{	//integer literals
		int i1=10;
		int i2=0b1010;
		int i3=012;
		int i4=0xa;
		out.println("Integer literals");
		out.println("i1:\t"+i1);
		out.println("i2:\t"+i2);
		out.println("i3:\t"+i3);
		out.println("i4:\t"+i4);
		//floating point literals
		float f1=25555.50f;
		//exponential notation
		float f2=25.55550e3f;
		double d1=25555.50d;
		double d2=25555.50D;
		double d3=25555.50;
		double d4=2.555550e4;
		out.println("Float literals");
		out.println("f1:\t"+f1);
		out.println("f2:\t"+f2);
		out.println("d1:\t"+d1);
		out.println("d2:\t"+d2);
		out.println("d3:\t"+d3);
		out.println("d4:\t"+d4);
		char ch1='a';
		char ch2=97;
		char ch3=0b1100001;
		char ch4=0x61;
		char ch5=0141;
		//max value we can give '\uffff'=65535
		char ch6='\u0061';
		out.println("Character literals");
		out.println("ch1:\t"+ch1);
		out.println("ch2:\t"+ch2);
		out.println("ch3:\t"+ch3);
		out.println("ch4:\t"+ch4);
		out.println("ch5:\t"+ch5);
		out.println("ch6:\t"+ch6);

		boolean b1=false;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        boolean b2=true;
        out.println("Boolean literals");
        out.println("b1:\t"+b1);
        out.println("b2:\t"+b2);
        String str="Pranutna";
        out.println("String literals");
        out.println("str:\t"+str);
    }
}
```

What is separator?

A separator is a symbol which is used to inoform the java compiler of how things are grouped in the code. For example elements of an array are separated by commas. The most commonly used separator in java is the semicolon which is used to terminate the statements.

Symbol	Name	Purpose
;	Semicolon	Terminates statements.
,	Comma	Separates consecutive identifiers in a variable declaration. Also used to chain statements together inside a **for** statement.
{ }	Braces	Used to contain the values of automatically initialized arrays.Also used to define a block of code, for classes, methods, and local scopes.
( )	Parentheses	Used to contain lists of parameters in method definition and invocation.Also used for defining precedence in expressions, containing expressions in control statements. Also used for surrounding cast types.
[ ]	Brackets	Used to declare array types.Also used when dereferencing array values.
.	Period	Used to separate package names from subpackages and classes Also used to separate a variable or method from a reference variable.

What is Comments?

Comments are un-executable statements in a program; we have 3 types of comments

- Single line(//------)
- Multiline (/*----------*/)
- Documentation (/**----------*/)

Comments.java

```
package p1;
//This is program on comments
/*
Author:         Madhu.K
Org:    Kveninar Software Solutions Pvt.Ltd
*/
import static java.lang.System.*;
/**It is a class which contains two instance fields and one instance method*/
class One
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
{       /**It is an instance variables where we can store int value, it is created by the JVM at the of every object creation*/
        int a,b;
        /**It is an instance method, it displays the a,b values(object state)*/
        void display()
        {       out.println("Object state....");
                out.println("a:\t"+a);
                out.println("b:\t"+b);
        }
}
class Comments
{       public static void main(String args[])
        {       //o1 object state is 100,200
                One o1=new One();
                o1.a=100;
                o1.b=200;
                o1.display();
        }
}
```

Operators

Operator is a symbol, which is used to perform an operation

1. Arithmetic (+, - , *, /, %)
2. Relational (<, >, <=, >=, ==, !=, instanceof)
3. Logical (&&, ||, !)
4. Bitwise (&, |, ^, ~, <<, >>, >>>(zero fill right shift))
5. Assignment(=)
6. Conditional (?:)
7. Short cut (++, --, +=, -=, *=, /=, %=)
8. Other (new, . (memory access), () typecast operator)

Arithmetic.java

```
import static java.lang.System.*;
class Arithmetic
{       //instance and static fields contains default values...
        int a,b;
        static int s;
        // i am not using these variable now....
        public static void main(String args[])
        {
                if(args.length==2)
                {
                //We can't use a local var without assigning a value
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
		//local variables doesn't contain default values...
			float a,b;
			a=Float.parseFloat(args[0]);
			b=Float.parseFloat(args[1]);
			out.println("a:\t"+a);
			out.println("b:\t"+b);

			out.println("Sum:\t"+(a+b));
			out.println("Sub:\t"+(a-b));
			out.println("Mul:\t"+(a*b));
			out.println("Div:\t"+(a/b));
			out.println("Remainder:\t"+(a%b));
		}
	}
}
```

Relational.java

```
import static java.lang.System.*;
class Relational
{
	public static void main(String args[])
	{	if(args.length==2)
		{	int a,b;
			a=Integer.parseInt(args[0]);
			b=Integer.parseInt(args[1]);
			out.println("a:\t"+a);
			out.println("b:\t"+b);
			out.println("a>b:\t"+(a>b));
			out.println("a<b:\t"+(a<b));
			out.println("a>=b:\t"+(a>=b));
			out.println("a<=b:\t"+(a<=b));
			out.println("a!=b:\t"+(a!=b));
			out.println("a==b:\t"+(a==b));
		}
	}
}
```

What is a type Comparison Operator?

It is a new operator introduced in java, and it is used to find out the type of an object.

Instanceof	It compares an object to a specified type

InstanceOfDemo.java

```
import static java.lang.System.*;
class Ratnam
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
{}
class Srinivas extends Ratnam
{
}
class Harsha extends Srinivas
{}
class InstanceOfDemo
{       public static void main(String args[])
        {
                Harsha h=new Harsha();
                if(h instanceof Harsha)
                {out.println("Instance of Harsh... avunu....nizame..");
                }

                if(h instanceof Srinivas)
                {out.println("Instance of Srinivas... avunu....nizame..");
                }
                if(h instanceof Ratnam)
                {out.println("Instance of Ratnam... avunu....nizame..");
                }

                Ratnam r=new Ratnam();
                if(r instanceof Harsha)
                out.println("Thatha garu..... Harasha....ne.....");
                else
                out.println("Thatha garu..... Harasha....kaaane...kaadu..");
                boolean bl=r instanceof Harsha;
                out.println(bl);
        }
}
```

What are conditional operators (logical operators)?

The conditional operators are used to construct more complex decision making expressions. The && and || operators perform ***Conditional-AND*** and ***Conditional-OR*** operations on two boolean expressions. These operators provide short-circuiting behaviour that means the second operand is evaluated only if it is necessary.

Operator	Use	Return true if
&&	Op1 && op2	op1 and op2 are both true
||	Op1 || op2	Either op1 or op2 is true

When we will get a NullPointerException?

If we call the instance method, by using a reference variable which contains null then we will get a NullpointerException

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

When we will get Exception?

We will get exceptions at the time of runtime error occurs.

BelowExample Throws NullPointer Exception

```
import static java.lang.System.*;
class One
{       //instance means object
        //instance method(Object method)(non-static method)
        void display()
        {out.println("instance method ....");
        }
}
//we can call an instance method, in other classes by using object only..
// if you try to call the instance method by using a ref variable, which contians null,
then we will get an exception called NullPointerException.
class LogicalConditional
{       public static void main(String args[])
        {       One o1=new One();
                o1.display();
                o1=null;
                o1.display();
        }
}
E:\LocalNonLocal>java LogicalConditional
instance method ....
Exception in thread "main" java.lang.NullPointerException
      at LogicalConditional.main(LogicalConditional.java:14)
```

NullExDemo2.java

```
import static java.lang.System.*;
class StringDemo
{       public static void main(String args[])
        {       //String str1="Madhu";
                String str1=null;
                String str2="Madhu";
                if(str1.equals(str2))
                out.println("Both contents are equal");
                else
                out.println("Contents are different");
        }
}
```

LogicalAndDemo.java

```
import static java.lang.System.*;
class LogicalDemo
{       public static void main(String args[])
        {       int age=35;
                String desig1=new String("Programmer");
                String desig2=new String("Programmer");
                //equals() method is an instance method existed in the String class
                if(age>=35 && desig1.equals(desig2))
                out.println("Senior Programmer");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                else
                out.println("Junior Programmer");
        }
}
```

Example:

```
import static java.lang.System.*;
class Condition
{       public static void main(String args[])
        {       int age=20;
                String desig=null;
                if(age>30 && desig.equals("Programmer"))
                out.println("Senior Programmer");
                else
                out.println("Junior Programmer");
                if(age<=20 || desig.equals("Programmer"))
                {out.println("Junior Programmer...");
                }
        }
}
```

By studying the above explanation you may got a doubt, that what is the meaning of short-circuiting behaviour, I will explain you clearly now, Ok! See the above example in that I have declared a variable "desig" of type **String** and assigned with "null" value. Please remember that if u calls an instance method by using the reference variable "desig" you will get a **NullPointerException** because it contains null. But in our program we have invoked equals () method twice by using that "desig", but we won't get any exception why? To know that, observe the below table...

Operator	Working
&&	If the first expression if false then this operator returns false, without checking second expression. This behavior is called as short circuting behavior.
||	If the first expression is true, then this operator returns true without checking second expression. This behavior is called as short circuting behavior.

Note: In the above example if age variable contains 35

Example:

```
import static java.lang.System.*;
class Condition2
{       public static void main(String args[])
        {       int age=35;
                String desig=null;
                if(age>30 && desig.equals("Programmer"))
                out.println("Senior Programmer");
                else
                out.println("Junior Programmer");
                if(age<=20 || desig.equals("Programmer"))
                {out.println("Junior Programmer...");
                }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        }
}
```

When we run the above program we will get a **NullPointerException, because the *conditional AND*** operator checks the first expression, if it is true(it is true in the above example), it will also checks the second expression, while execution of second expression we will get a null pointer Exception.

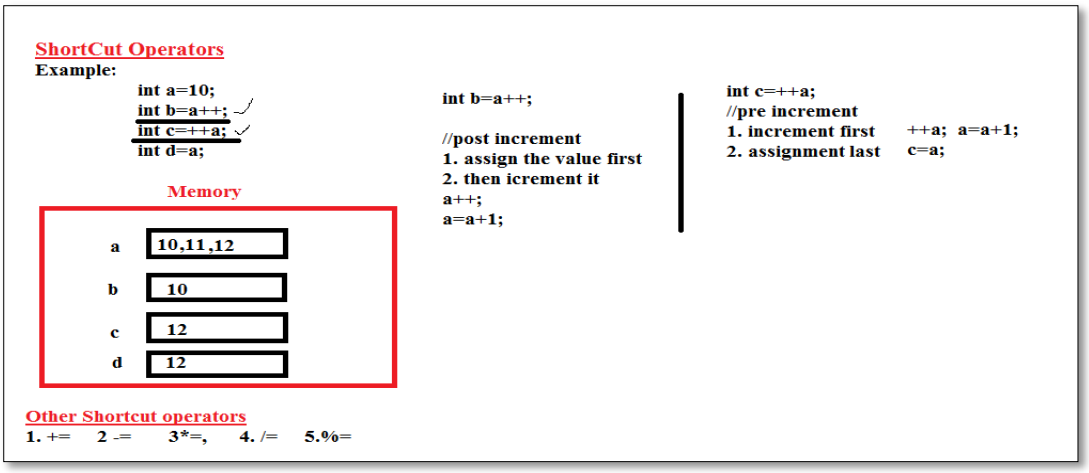
Logical.java

```
import static java.lang.System.*;
class Logical
{       public static void main(String args[])
        {       int m1=90,m2=99,m3=100;
                if(m1>=50 && m2>=50 && m3>=50)
                {
                        float avg=(m1+m2+m3)/3;
                        if(avg>=90)
                        out.println("A+ grade");
                        else if(avg>=80)
                                out.println("A grade");
                        else if(avg>=70)
                                out.println("B grade");
                        else if(avg>=60)
                                out.println("C grade");
                        else if(avg>=50)
                                out.println("D grade");
                }else
                out.println("Fail...");
        }
}
```

Logical not (!) operator

```
import static java.lang.System.*;
class LogicalNot
{       public static void main(String args[])
        {       String desig="user";
        //write a code to print "invalid user" output, if desig not contains a string called  "admin".
        //write the code without using else part.
                if(!desig.equals("admin"))
                {out.println("Invalid User.....");
                }
                //we can achieve it by using if-else but that code is complex than the above code.
        }
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.



Examples

ShortCutOperators.java

```
import static java.lang.System.*;
class ShortCutOperators
{	public static void main(String args[])
	{	 int a=10;
		int b=a++;
		int c=++a;
		int d=a;
		out.println("Values...");
		out.println("a:\t"+a);
		out.println("b:\t"+b);
		out.println("c:\t"+c);
		out.println("d:\t"+d);
	}
}
```

OtherShortCutOperators.java

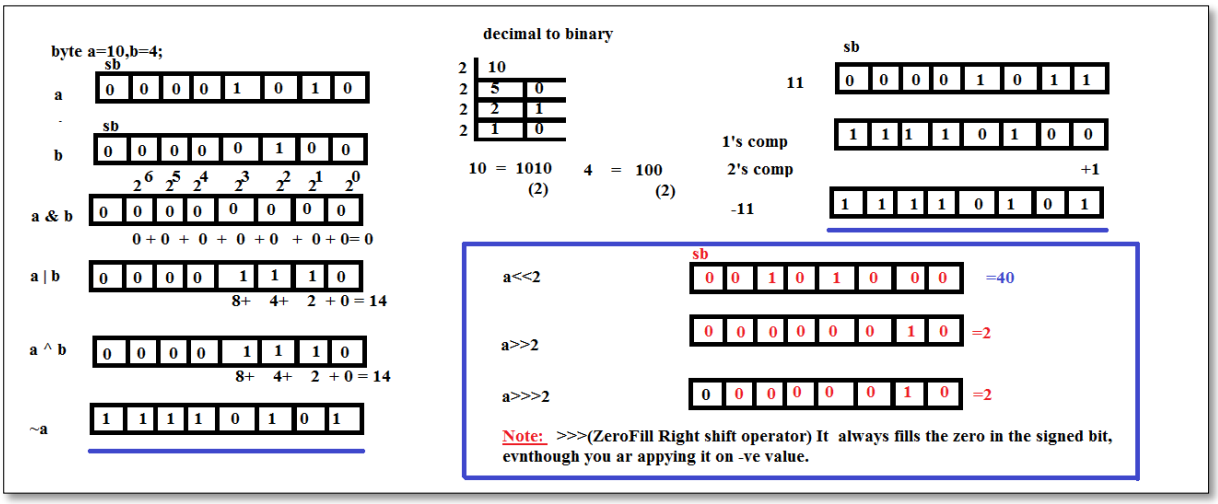
```
import static java.lang.System.*;
class OtherShortCutOperators
{	public static void main(String args[])
	{	int a=10;
		a+=2;	//a=a+2;
		out.println("a:\t"+a);
		a-=2;	//a=a-2;
		out.println("a:\t"+a);
		a*=2;	//a=a*2;
		out.println("a:\t"+a);
		a/=2;	//a=a/2;
		out.println("a:\t"+a);
		a%=2;	//a=a%2;
		out.println("a:\t"+a);
	}
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Bitwise Operators:

Bitwise operators are used to perform operations on bits. There are 7 bitwise operators available in java.

- Bitwise and(&)
- Bitwise or(|)
- Bitwise compliment(negation) (~)
- Bitwise Exclusive or (^)
- Bitwise Left shift operator (<<)
- Bitwise Right shift operator (>>)
- Bitwise Zerofill Right shift operator(>>>)



Note: Bitwise operator performs operation on int values only.

BitwiseDemo.java

```
import static java.lang.System.*;
class BitwiseDemo
{
	public static void main(String args[])
	{	byte a=10,b=4;
		out.println("a:\t"+a);
		//out.println("a: 10");
		out.println("b:\t"+b);
		out.println("a&b:\t"+(a&b));
		out.println("a|b:\t"+(a|b));
		out.println("a^b:\t"+(a^b));
		out.println("~a:\t"+(~a));
		out.println("a<<2:\t"+(a<<2));
		out.println("a>>2:\t"+(a>>2));
		out.println("a>>>2:\t"+(a>>>2));
		out.println("str:\t"+Integer.toBinaryString(a>>>2));
		out.println("str:\t"+Integer.toBinaryString(a<<2));
		out.println("str:\t"+Integer.toBinaryString(~a));
		int i=-11;
		out.println("str:\t"+Integer.toBinaryString(i>>>2));
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("a>>2:\t"+Integer.toBinaryString(i>>2));
        }
}
```

BitwiseDemo.java

```
import static java.lang.System.*;
class BitwiseDemo
{       public static void main(String args[])
        {       byte a=10,b=4;
                out.println("a:\t"+a);
                out.println("b:\t"+b);
                out.println("(a&b):\t"+(a&b));
                out.println("(a|b):\t"+(a|b));
                out.println("(a^b):\t"+(a^b));
                out.println("(~a):\t"+(~a));
                out.println("(a<<2):\t"+(a<<2));
                out.println("(a>>2):\t"+(a>>2));
                out.println("(a>>>2):\t"+(a>>>2));
        }
}
```

Run

```
E:\ThunderBolts>java BitwiseDemo
a:      10
b:      4
(a&b):  0
(a|b):  14
(a^b):  14
(~a):   -11
(a<<2): 40
(a>>2): 2
(a>>>2):        2
```

Conditional Operator (?:)

```
import static java.lang.System.*;
class ConditionalOperator
{       public static void main(String args[])
        {
                int a=100,b=20;
                int big=(a>b)?a:b;
                out.println("Big:\t"+big);
                int x=300,y=220,z=120;
                big=(x>y && x>z)?x:((y>z)?y:z);
                out.println("Big:\t"+big);
        }
}
```

Operators are divided into 3 categories based on the operands on which the operator performs an operation.

1. **Unary:** it performs operation on only one operand

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

2. **Binary:** it performs operation on two operands
3. **Ternary:** it performs operation on three operands

TypesOfOperators.java

```
import static java.lang.System.*;
class TypesOfOperators
{	public static void main(String args[])
	{	int a=2,b=10;
		a++;// ++ is unary
		a--;// -- is  unary
		int c1=a+b;//+ is binary
		int c2=a*b;//* is binary
		int big=(a>b)?a:b;//?: is ternary operator
		out.println("Values.....");
		out.println("a:\t"+a);
		out.println("b:\t"+b);
		out.println("c1:\t"+c1);
		out.println("c2:\t"+c2);
		out.println("Big:\t"+big);
	}
}
```

Operators are divided into 3 categories based the operands on which it perform an operation

1. Unary operators: **perform operations on single operand**
   Example: **i++, i--**
2. Binary operators: perform operations on two operands
   Example: **a+b**
3. Ternary Operators: perform operations on three operands
   Example: (a>b)?a:b

UBT.java

```
import static java.lang.System.*;
class UBT
{	public static void main(String args[])
	{	int a=2;
		out.println("a:\t"+a);
		a++;	//a=a+1;
		out.println("a:\t"+a);
		//Note:  in the initialization statement if you find increment or decrement operator first check wheter it is post increment or pre increment
		//post increment: if it is post increment postpone the  increment and do the initialization first
		//pre increment: if it is pre increment do the  increment first and then perform the initialization
		//a++ post increment
		int b=a++;//a=a+1;
		out.println("b:\t"+b);
```

```
                out.println("a:\t"+a);
                int c=++a;//a=a+1;
                out.println("c:\t"+c);
                out.println("a:\t"+a);

                //Binary operator
                int x=100+200;

                int i=100,j=20;
                int big=(i>j)?i:j;
                out.println("Big:\t"+big);
        }
}
```

Types of Conversions

Type conversions in java are divided into **13 categories** those are listed and briefly discussed below.

- Identity conversions
- Widening primitive conversions
- Narrowing primitive conversions
- Widening and Narrowing Primitive conversion
- Widening reference conversions
- Narrowing reference conversions
- Boxing conversions
- Un-boxing conversions
- Unchecked conversions
- Capture conversions
- String conversions
- Value set conversions
- Forbidden Conversion

Identity Conversion:

An identity conversion converts from any type to the same type.

Identify.java

```
import static java.lang.System.*;
class Identity
{       public static void main(String args[])
        {       int a=20;
                //identity conversion
                int b=a;
                out.println("a:\t"+a);
                out.println("b:\t"+b);
        }
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Widening Primitive Conversion

A widening primitive conversion does not lose information. We have 19 specific conversions on primitive types those are called as *widening primitive conversions*.

- byte to short, int, long, float, or double
- short to int, long, float, or double
- char to int, long, float, or double
- int to long, float, or double
- long to float or double
- float to double

Note: while executing a widening primitive conversion runtime error will not occur.

Narrowing Primitive Conversion

A narrowing primitive conversion may lose information. We have 22 specific conversions on primitive types those are called as *narrowing primitive conversions*.

- *short to byte or char*
- *char to byte or short*
- *int to byte, short, or char*
- *long to byte, short, char, or int*
- *float to byte, short, char, int, or long*
- *double to byte, short, char, int, long, or float*

Note: while compilation compiler displays an error, if we don't use the typecast operator. While executing a narrowing primitive conversion runtime error will not occur.

Note: there may be a chance of loss of data.

Widening and Narrowing Primitive conversion

Some times at the time of assigning one primitive type to another type, JVM automatically performs two conversions

- byte to char.

In the above conversion, first the byte is converted to an int it is called as widening and then the int is converted a char it is called as narrowing.

Widening reference conversion

Process of assigning reference of child type to a base type is called widening reference conversion.

Example:

```
import static java.lang.System.*;
class Base{
}
class Child extends Base{
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
}
class WRT{
        public static void main(String args[]){
                Child c=new Child();
                //Widening reference conversion
                Base b=c;
                out.println(b);
        }
}
```

Note: while executing a widening reference conversion runtime error will not occur.

Narrowing reference conversion

Process of assigning reference of base type to a child type is called narrowing reference conversion.

Example:

```
import static java.lang.System.*;
class Base{
        public void display()
        {out.println("Base");
        }
}
class Child extends Base{
        public void display()
        {out.println("Child display method");
        }
}
class Narrowing
{       public static void main(String args[])
        {       //Narrowing Reference conversion
                //But at runtime JVM throws an exception called ClastCastException
                //Child c2=(Child)new Base();
                Base b=new Child();
                //It will not throw an exception
                Child c=(Child)b;
                c.display();
        }
}
```

Note: while executing a narrowing reference conversion there is a chance to occur runtime exception.

Boxing Conversion

Process of converting primitive type to corresponding reference type is called boxing conversion. Specifically, the following nine conversions are called the *boxing conversions*:

- boolean to Boolean
- byte to Byte

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- short to Short
- char to Character
- int to Integer
- long to Long
- float to Float
- double to Double
- null to null

Boxing.java

```
import static java.lang.System.*;
class Boxing
{	public static void main(String args[])
	{	//Boxing(old)
		int a=100;
		Integer io1=new Integer(a);
		out.println("Old:\t"+io1);
		//Auto Boxing(new)
		Integer io2=a;
		out.println("New:\t"+io2);
	}
}
```

Un-boxing Conversion

Un-boxing conversion converts the reference type to primitive type. Specifically, the following eight conversions are called the *un-boxing conversions*:

- Boolean to boolean
- Byte to byte
- Short to short
- Character to char
- Integer to int
- Long to long
- Float to float
- Double to double

UnBoxing.java

```
import static java.lang.System.*;
class UnBoxing
{	public static void main(String args[])
	{	Integer io1=new Integer(100);
		//UnBoxing(old)
		int v=io1.intValue();
		out.println("Old:\t"+v);
		//Auto UnBoxing(new)
		int r=io1;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("New:\t"+r);
        }
}
```

Unchecked Conversion

Process of converting raw class or interface type to parameterized type is called unchecked conversion.

Unchecked.java

```
import static java.lang.System.*;
class One<T>
{T a;
}
class Unchecked
{       public static void main(String args[])
        {       //unchecked conversion
                One<Integer> o=new One();
                o.a=100;
                out.println(o.a);
        }
}
```

Capture Conversion

Capture conversion is the type conversion which is occurred when a reference of a generic type is created by passing the type parameters.

Example:

```
import static java.lang.System.*;
class One<T,E extends T>
{       T a;
        E b;
        void setOne(T a,E b)
        {       this.a=a;
                this.b=b;
        }
        public void display()
        {       out.println("Generic Types");
                out.println("a:\t"+a);
                out.println("b:\t"+b);
        }
}
class CaptureCon
{       public static void main(String args[])
        {One<Object,String> o=new One<Object,String>();//Where capture conversion takes place
        o.setOne("Madhu","Babu");
        o.display();
        }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

}

In the above example an instance of the One class is created by passing the arguments Object and String. Here After capture conversion, a new type is created where E is replaced by Object and F is replaced by String.

String Conversion

Process of converting any type to String is called *string conversion*.

Example:

```
import static java.lang.System.*;
class StringConversion
{	public static void main(String args[])
	{	int i=100;
		String si=""+i;
		String si2=String.valueOf(true);
		String si3=String.valueOf('c');
		char ch[]={'p','r','i','y','a'};
		String str=new String(ch);
		out.println("si:\t"+si);
		out.println("si2:\t"+si2);
		out.println("si3:\t"+si3);
		out.println("str:\t"+str);
	}
}
```

Forbidden Conversion

Any conversion that is not explicitly allowed is forbidden

Value Set Conversion:

Value set conversion is the process of mapping a floating-point value from one value set to another without changing its type. We will discuss about it later.

Accepting Input From the keyboard

ReadData1.java

```
import static java.lang.System.*;
class ReadData1
{	public static void main(String args[])throws Exception
	{	out.println("Enter any value");
//int read(): method reads a single char(0 to 255 range)  and returns the unicode
value of that character.
		int r=in.read();
		out.println("r:\t"+r);
	}
}
```

ReadDemo2.java

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
import static java.lang.System.*;
class ReadData2
{	public static void main(String args[])throws Exception
	{	out.println("Enter anything...");
		int r=0;
		String str="";
		while(  (r=in.read())!=13 )
		{	char ch=(char)r;
			str=str+ch;
			//1.   str=""+'a'==>"a";
			//2.   str="a"+'b'==>"ab";
			//3.   str="ab"+'c'==>"abc";
			//out.println(ch);
		}
		out.println("Str:\t"+str);
	}
}
```

DataInputStream:

- **It is a byte stream and also called as filter stream, which is used to read the data from any inputstream.**

ReadData3.java

```
import static java.lang.System.*;
import java.io.*;
class ReadData3
{
	public static void main(String args[])throws Exception
	{	DataInputStream dis=new DataInputStream(in);
		out.println("Enter anything...");
		String line=dis.readLine();
		out.println("Line:\t"+line);
	}
}
```

readLine() of DataInputStream class: It is used to read the total line at a time.

prototype

@Deprecated
public final String readLine()throws IOException

Draback of this method: by using it we can only read the characters range of 0 to 255. We can't read Unicode character set using this method.

What is the solution?

We have to use the readLine() method of BufferedReader class. it can read Unicode character set.

Reading data using BufferedReader

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

BufferedReader: It is a character stream and it is only used to read the data from other character stream

InputStreamReader: it is a mediater stream which is used to, convert bytestream into character stream

Example(ReadData4.java)

```
import static java.lang.System.*;
import  java.io.*;
class ReadData4
{	public static void main(String args[])throws Exception
	{
		InputStreamReader isr=new InputStreamReader(in);
		BufferedReader br=new BufferedReader(isr);
		out.println("Enter anything");
		String str=br.readLine();
		out.println("str:\t"+str);
	}
}
```

ReadData5.java

```
import static java.lang.System.*;
import static java.lang.Integer.*;
import  java.io.*;
class ReadData5
{
	public static void main(String args[])throws Exception
	{	InputStreamReader isr=new InputStreamReader(in);
		BufferedReader br=new BufferedReader(isr);
		out.println("Enter value for a");
		int a=parseInt(br.readLine());
		out.println("Enter value for b");
		int b=parseInt(br.readLine());
		int c=a+b;
		out.println("c:\t"+c);
	}
}
```

ReadData6.java

```
import static java.lang.System.*;
import static java.lang.Float.*;
import  java.io.*;
class ReadData6
{	public static void main(String args[])throws Exception
	{	InputStreamReader isr=new InputStreamReader(in);
		BufferedReader br=new BufferedReader(isr);
		out.println("Enter value for a");
		float a=parseFloat(br.readLine());
		out.println("Enter value for b");
		float b=parseFloat(br.readLine());
		float c=a/b;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("c:\t"+c);
        }
}
```

ReadData7.java

```
import static java.lang.System.*;
import static java.lang.Integer.*;
import static java.lang.Float.*;
import java.io.*;
class Emp
{       int eno;
        String ename;
        float sal;
        void set(int no,String name,float sa)
        {       out.println("Object initialization");
                eno=no;
                ename=name;
                sal=sa;
        }
        void display()
        {       out.println("Object state...");
                out.println("Eno:\t"+eno);
                out.println("Ename:\t"+ename);
                out.println("Sal:\t"+sal);
        }
}
class ReadData7
{       public static void main(String args[])throws Exception
        {       Emp e1=new Emp();
                BufferedReader br=new BufferedReader(new InputStreamReader(in));
                out.println("Enter eno:");
                int no=parseInt(br.readLine());
                out.println("Enter ename:");
                String ename=br.readLine();
                out.println("Enter Sal:");
                float sa=parseFloat(br.readLine());
                e1.set(no,ename,sa);
                e1.display();
        }
}
```

Reading Data with java.util.Scanner Class:

We can use **Scanner** class to read input from the keyboard or a text file. When the *Scanner* class of ***java.util*** package receives input, it breaks the input into several pieces, called tokens, these tokens can be retrieved from the Scanner object using the following methods.

- next(): to read a string

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- nextByte(): to read byte value
- nextInt(): to read an integer value
- nextLong(): to read long value
- nextFloat(): to read float value
- nextDouble(): to read double value
- nextLine(): to read total line at a time

to read input from keyboard, we have to create object for Scanner class

Scanner sc=new Scanner(System.in);

Now, if the user has given an integer value from the keyboard, it is stored into the Scanner object as a token. To retrieve that token, we can call method: sc.nextInt(). The following program will make this concept clear.

ScannerDemo1.java

```
import static java.lang.System.*;
import java.util.*;
import java.io.*;
class ScannerDemo
{	public static void main(String args[])
	{	Scanner s=new Scanner(in);
		//FileInputStream fis=new FileInputStream("Variables.java");
		//Scanner ss=new Scanner(fis);
		out.println("Enter a,b values");
		int a=s.nextInt();
		int b=s.nextInt();
		int c=a+b;
		out.println("c:\t"+c);
	}
}
```

ScannerDemo2.java

```
import java.util.*;
import static java.lang.System.*;
class ScannerDemo2
{	public static void main(String args[])
	{	Scanner sc=new Scanner(in);
		out.println("Enter eno,ename,sal");
		//101	madhu	20000
		int eno=sc.nextInt();
		String ename=sc.next();
		float sal=sc.nextFloat();
		out.println("\nGiven Employee Details\n");
		out.println("Eno:\t"+eno);
		out.println("Ename:\t"+ename);
		out.println("Sal:\t"+sal);
	}
}
```

Note:

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

nextInt(),nextByte(),nextShort(),nextLong(),nextFloat(), nextDouble(),nextBoolean(),next() methods ignore new line characters and spaces. But nextLine() methods reads those characters without ignore.

ScannerDemo3.java

```
import java.util.*;
import static java.lang.System.*;
class ScannerDemo3
{
	public static void main(String args[])
	{	Scanner sc=new Scanner(in);
		out.println("Enter ename,eno,sal");
		String ename=sc.nextLine();
		int eno=sc.nextInt();
		float sal=sc.nextFloat();
		out.println("\nGiven Employee Details\n");
		out.println("Eno:\t"+eno);
		out.println("Ename:\t"+ename);
		out.println("Sal:\t"+sal);
	}
}
```

ScannerDemo4.java

```
import java.util.*;
import static java.lang.System.*;
class ScannerDemo4
{	public static void main(String args[])
	{	Scanner sc=new Scanner(in);
		out.println("Enter eno,ename,sal");
		int eno=sc.nextInt();
		sc.nextLine();
		String ename=sc.nextLine();
		float sal=sc.nextFloat();
		out.println("\nGiven Employee Details\n");
		out.println("Eno:\t"+eno);
		out.println("Ename:\t"+ename);
		out.println("Sal:\t"+sal);
	}
}
```

Note:

1. If u want to read total line with combination of int, float..etc.. values, by using Scanner class object, you have to read the line first, after that you have to read int, float etc..... values, if you want to read the total line in middle of int, float, double etc... You have to call the nextLine() method two time.

1. nextLine() it removes unwanted chars
2. nextLine() it reads the exact line

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

CONTROL STATEMENTS IN JAVA

The most commonly used statements in any programming language are as follows.

- **Sequential Statements:** These are the statements which are executed by JVM one by one in a sequential manner. So they are called sequential statements.
  Example of sequential statements:

```
int x=10;
int y=20;
int z=x+y;
out.println(z);
```

- **Control Statements:** These are the statements that are executed randomly and repeatedly. Control statements alter the flow of execution and provide better control to the programmer on the flow of execution.

The following control statements are available in Java

- **If statement**
- **if …else statement**
- **while loop**
- **do…while loop**
- **for…loop**
- **for-each loop**
- **switch statements**
- **break statement**
- **continue statement**
- **return statement**

IfDemo.java

```
import static java.lang.System.*;
import java.util.*;
class IfConditionDemo
{public static void main(String args[])
        {       Scanner s=new Scanner(in);
                String performance="UnSatisfactory";
                float sales,target,bonus=0.0f;
                out.println("Enter Target");
                target=s.nextFloat();
                out.println("Enter Sales");
                sales=s.nextFloat();
                if(sales>=target)
                {       performance="Satisfactory";
                        bonus=(sales-target)*10/100.00f;
                }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("Sales:\t"+sales);
                out.println("Target:\t"+target);
                out.println("Performance:\t"+performance);
                out.println("Bonus:\t"+bonus);
        }
}
```

IfElseDemo.java

```
import static java.lang.System.*;
import java.util.*;
class IfElseDemo
{
        public static void main(String args[])
        {
                Scanner s=new Scanner(in);
                String performance;
                float sales,target,bonus;
                out.println("Enter Target");
                target=s.nextFloat();
                out.println("Enter Sales");
                sales=s.nextFloat();

                if(sales>=target)
                {       performance="Satisfactory";
                        bonus=(sales-target)*10/100.00f;
                }else
                {       performance="UnSatisfactory";
                        bonus=0.0f;
                }
                out.println("Sales:\t"+sales);
                out.println("Target:\t"+target);
                out.println("Performance:\t"+performance);
                out.println("Bonus:\t"+bonus);
        }
}
```

```
import static java.lang.System.*;
import java.util.*;
class IfElseDemo
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                String performance;
                float sales,target,bonus;
                out.println("Enter Target");
                target=s.nextFloat();
                out.println("Enter Sales");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                sales=s.nextFloat();
                if(sales>=target)
                {       performance="Satisfactory";
                        bonus=(sales-target)*10/100.00f;
                }else
                {       performance="UnSatisfactory";
                        bonus=0.0f;
                }
                out.println("Sales:\t"+sales);
                out.println("Target:\t"+target);
                out.println("Performance:\t"+performance);
                out.println("Bonus:\t"+bonus);
        }
}
```

ElseIfLadder.java

```
import static java.lang.System.*;
import java.util.*;
class ElseIfLadder
{
        public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("1. red");
                out.println("2. yellow");
                out.println("3. green");
                out.println("Enter your option(1 to 3)");
                int opt=s.nextInt();
                if(opt==1)
                        out.println("stop");
                else if(opt==2)
                        out.println("wait");
                else if(opt==3)
                        out.println("go..");
                else
                        out.println("Wrong input....");
        }
}
```

Switch statement

When there are several options and we have to choose only one option from the available ones, we can use switch statement. Depending on the selected option, a particular task can be performed. To switch statement we can pass int, char, String or Enum values.

Syntax:

```
switch(var)
{       case val1:      stmts1;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        case val2:      stmts2;
        case val3:      stmts3;

         -      -       -       -
         -      -       -       -
        case valn:      stmtsn;
        [default :      default_stmt]
}
```

Here, depending on the value of the **var**, a particular task (statements) will be executed. If the variable value is equal to *val1*, *stmts1* will be executed, if the variable value is *val2* then the *stmts2* will be executed, and so on. If the variable value does not equal to val1, val2… then none of the statements will be executed. In that case, default statements are executed.

SwitchDemo.java

```
import static java.lang.System.*;
import java.util.*;
class SwitchDemo
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("1. red");
                out.println("2. yellow");
                out.println("3. green");
                out.println("Enter your option(1 to 3)");
                int opt=s.nextInt();
                switch(opt)
                {       case 1:out.println("stop");
                                break;
                        case 2: out.println("wait");
                                break;
                        case 3: out.println("go..");
                                break;
                        default:        out.println("Wrong input....");
                }
        }
}
```

SwitchDemo1.java

```
import java.util.*;
import static java.lang.System.*;
class SwitchDemo1
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("Enter a val between(1 to 7)");
                int r=s.nextInt();
                switch(r)
                {       case 1:  out.println("Monday");
                                break;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                        case 2:  out.println("Tuesday");
                                 break;
                        case 3:  out.println("Wednesday");
                                 break;
                        case 4:  out.println("Thursday");
                                 break;
                        case 5:  out.println("Friday");
                                 break;
                        case 6:  out.println("Saturday");
                                 break;
                        case 7:  out.println("Sunday");
                                 break;
                        default:out.println("Give the Number Between 1 to 7");
                }
        }
}
```

SwitchDemo2.java

```
import java.util.*;
import static java.lang.System.*;
class SwitchDemo2
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("Enter today in short cut(in string)");
                String day=s.next();
                switch(day)
                {       case "mon":     out.println("Monday is Drowsy day");
                                        break;
                        case "sat":
                        case "sun":     out.println("Weekend days are jolydays");
                                        break;
                        default:out.println("Wrong option");
                }
        }
}
```

SwitchDemo3.java

```
import static java.lang.System.*;
class SwitchDemo3
{       public static void main(String args[])throws Exception
        {       out.println("enter a alphabet");
                char ch=(char)in.read();
                switch(ch)
                {       case 'a':
                        case 'e':
                        case 'i':
                        case 'o':
                        case 'u':
                                out.println("Vowel");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                                        break;
                                        default:out.println("Consonants");
                        }
                }
}
```

SwitchDemo.java

```
import java.util.*;
import static java.lang.System.*;
enum Color{RED,GREEN,BLUE,YELLOW}
class SwitchDemo
{       public static void main(String args[])
        {       //Color c=Color.YELLOW;
                Scanner s=new Scanner(in);
                out.println("Enter color");
                String cstr=s.next();
                Color c=Color.valueOf(cstr);
                switch(c)
                {       case RED:       out.println("neetho... musthu danger raaa bhai...");
                                        break;
                        case GREEN:     out.println("SwachaBharath");
                                        break;
                        case BLUE:      out.println("Blue Blue GodBlessYou");
                                        break;
                        case YELLOW:    out.println("Yellow... Yellow... dirty fellow");
                                        break;
                        default:        out.println("Wrong option");
                }
        }
}
```

SwitchDemo.java

```
import static java.lang.System.*;
import java.util.*;
enum Colors{RED,GREEN,BLUE,YELLOW,ORANGE,WHITE,PINK}
class SwitchDemo
{
        public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("ooo kalar yenter cheyaraaa babu(capital letters lo)");
//valueOf() method of Colors enum converts string to enum
                Colors c=Colors.valueOf(s.next());
                //out.println("c:\t"+c);
                switch(c)
                {
                        case RED: out.println("Neetho mahaaa danger raaa babu...");
                                break;
                        case GREEN:
                                out.println("Yetraa.. chetlu .... chemalu..... ani
brathikestaavaaa... yenti....");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                        break;
                case BLUE:
                        out.println("Yetraaa.... avataaar cinemaaalo..... avatar laaagaa... nee avataram... nuvvu....");
                        break;
                default:
                        out.println("Neee optionalaki ooo dandam raaa babu...");
                        break;
            }
        }
}
```

Unconditional statements (break, continue and return)

Break Statement

The break statement can be used in 3 ways:

- break is used inside a loop to come out of it.
- break is used inside the switch block to come out of the switch block.
- break can be used in nested blocks to go to the end of a block. Nested blocks represent a block written within another block.

BreakDemo1.java

```
import static java.lang.System.*;
class BreakDemo1
{       public static void main(String args[])
        {       out.println("Start of main");
                boolean bl=true;
                L1:{
                        L2:{
                                L3:{
                                out.println("block3");
                                if(bl)
                                break L1;
                                }//end of L3 block
                                out.println("block2");
                        }//end of L2 block
                        out.println("block1");
                }//end of L1 block
                out.println("End of main");
        }
}
```

BreakDemo2.java

```
import static java.lang.System.*;
class BreakDemo2
{       public static void main(String args[])
        {       int a[]={10,20,30,40,50,60,70,80,90,100};
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                java.util.Scanner s=new java.util.Scanner(in);
                out.println("Enter element to find");
                int n=s.nextInt();
                for(int i=0;i<a.length;i++)
                {       if(a[i]==n)
                        {
                         out.println("Found at "+i+" th index ");
                         break;
                         }
                }
        }
}
```

Why goto statements are not available in Java?

Goto statements lead to confusion for a programmer. Especially, in a large program, if several goto statements are used, it becomes very difficult for the programmer to understand the flow of the program.

Continue Statement

Continue is used inside a loop to repeat the next iteration of the loop. When continue is executed, subsequent statements in the loop are not executed and control of execution goes back to the next repetition of the loop.

Syntax: Continue;

ContinueDemo.java

```
import static java.lang.System.*;
class ContinueDemo
{       public static void main(String args[])
        {       int arr[]={1,2,3,4,5,6,7,8,9,10};
                //i=0,1,2,3
                //arr.length=10
                for(int i=0;i<arr.length;i++)
                {       if(arr[i]%2==0)
                        continue;
                        //break;
                        out.print(i+" th index:\t");
                        out.println(arr[i]);
                }
        }
}
```

Output:

```
E:\LocalNonLocal>javac ContinueDemo.java
E:\LocalNonLocal>java ContinueDemo
0 th index:     1
2 th index:     3
4 th index:     5
6 th index:     7
8 th index:     9
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

The **continue** statement can be used along with a label, like break statement as:

```
Continue label;
```

ContinueLabel.java

```
import static java.lang.System.*;
class ContinueLabel
{       public static void main(String args[])
        {       int i=1,j;
                lp1: while(i<=3)
                {       out.print(i+":    ");
                        lp2:for(j=1;j<=5;j++)
                        {       out.println("\t"+j);
                                if(j==3)
                                {       i++;
                                        continue lp1;
                                }
                        }
                        i++;
                }
        }
}
```

Output:

```
E:\>javac   ContinueLabel.java
E:\>java   ContinueLabel
1:      1
        2
        3
2:      1
        2
        3
3:      1
        2
        3
```

Return statement

Return statement is used in a method to come out of it to the calling method. For example, we are calling a method by the name add () from the main () method. If return is used inside add (), then the flow of execution comes out of it and goes back to main ().

ReturnDemo.java

```
import java.util.*;
import static java.lang.System.*;
class ReturnDemo
{       static int add(int a,int b)
        {return a+b;
        }
        public static void main(String args[])
        {       Scanner s=new Scanner(System.in);
                out.println("Enter a&b values");
                int sum=add(s.nextInt(),s.nextInt());
                out.println("Sum Is\t"+sum);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        }
}
```

ReturnDemo2.java

```
import static java.lang.System.*;
class Calc
{       int add(int a,int b)//non-static method(instance method)
        {       int c=a+b;
                return c;
        }
}
class ReturnDemo2
{       public static void main(String args[])
        {       Calc c=new Calc();
                //calling method of add is main method
                int sum=c.add(100,200);
                out.println("Sum:\t"+sum);
        }
}
```

LOOPS

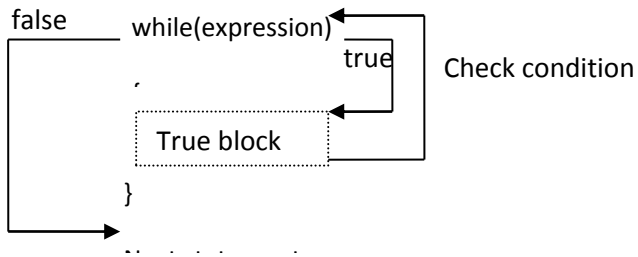
While loop

The while loop is used when the number of iterations to be performed are not known in advanced, While loop executes a statement or statements (which may be block of statements) while the condition is true. Once the condition is false, then the loop will be terminated.

Syntax:

```
while(condition){
      statements;
  }
```

The control flow of the while loop



Control flow of while loop

WhileDemo.java

```
import static java.lang.System.*;
import java.util.*;
class WhileDemo
{                       public static void main(String args[])
                        {  Scanner s=new Scanner(in);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                        out.println("Enter an int value");
                        int n=s.nextInt();
                        out.println("n:\t"+n);
                        int i=1;
                        while(i<=n)
                        {out.println("Welcome...");
                        i++;
                        }
                        out.println("Aaaparaaa....");
                    }
}
```

WhileDemo.java

```
import static java.lang.System.*;
class WhileDemo
{       public static void main(String args[])
        {               java.util.Scanner s=new java.util.Scanner(in);
                        out.println("Enter a no...");
                        int n=s.nextInt();
                        int i=1;
                        while(i<=n)
                        {out.print(i+"\t");
                        i++;
                        }
        }
}
```

do..while loop

If we want to execute the whole body of a while loop at least once, even though test expression returns false, we have to use do..while loop.

Syntax:

```
do{  statements;
 }
while(condition);
```

DoWhile.java

```
import static java.lang.System.*;
class DoWhile
{       public static void main(String args[])
        {       java.util.Scanner s=new java.util.Scanner(in);
                out.println("Enter a no...");
                int n=s.nextInt();
                int i=1;
                do
                {out.print(i+"\t");
                i++;
                }while(i<=n);
        }
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

For loop:

A for loop is used to execute, set of statements, for a specific no.of times. Syntax is given below.

Syntax:

```
for(initialization;condition;increment/decrement){  statements;
 }
```

Example: ForDemo1.java

```
import static java.lang.System.*;
class ForDemo
{           public static void main(String args[])
            {
            java.util.Scanner s=new java.util.Scanner(in);
            out.println("Enter a number");
            int n=s.nextInt();
            for(int i=1;i<=n;i++)
            {out.println("Cofeel thagara.....");
            }
            }
}
```

Example: ForDemo2.java

```
import static java.lang.System.*;
import java.util.*;
class ForDemo2
{           public static void main(String args[])
            {       int arr[]={10,20,30,40,50,60,70,80};
                    out.println("Elements in an array");
                    for(int i=0;i<arr.length;i++)
                    {out.println(arr[i]+"\t");
                    }
            }
}
```

Assignment.java

```
import static java.lang.System.*;
import java.util.*;
class Assignment
{                       public static void main(String args[])
                      {  Scanner s=new Scanner(in);
                         out.println("How many int elements you want to enter");
                         int n=s.nextInt();
                         String evenNos="";
                         String oddNos="";
                         int inputValue;
                         for(int i=1;i<=n;i++)
                         {          out.println("Enter element:\t"+i);
                                    inputValue=s.nextInt();
                                    if(inputValue%2==0)
                                    {
                                    evenNos=evenNos+inputValue+"\t";
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                                    //out.println(inputValue+" is even");
                                    }
                                    else
                                    {
                                    //out.println(inputValue+" is odd");
                                    oddNos=oddNos+inputValue+"\t";
                                    }
                            }
                            out.println("Even Nos:\t"+evenNos);
                            out.println("Odd Nos:\t"+oddNos);
                        }
}
```

For-each loop

It was introduced in **Java SE 5.0**. It is very power full looping construct that allows you to get the elements of an array (or) collection directly without having to-do with index values and conditional checking.

What is collection?

A collection is a group of elements like integer, float, char, or objects. Examples for collections are I) Arrays and II) classes of java.util package (like Stack, ArrayList, LinkedList, Vector, etc...)

The *enhanced* for loop

Syntax:

```
for (variable: collection)
 { Statements;
 }
```

Here the given variable is attached to the collection. This variable represents each element of the collection one by one. If, the collection contains 10 elements then this loop will be executed 10 times and the variable will represent these elements one by one.

Example:
```
for(int ele:a)
Out.println(ele);
 // You should read this loop as “for each element in a”.
```

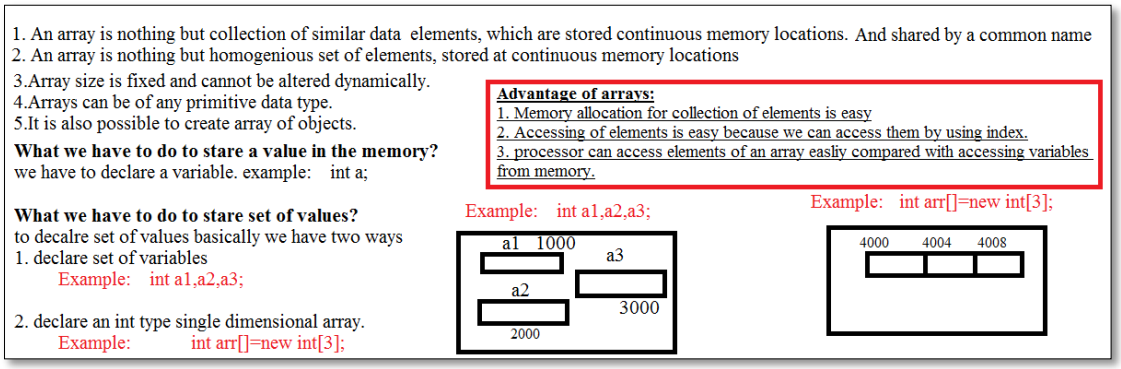
ForEachDemo.java

```
import static java.lang.System.*;
import java.util.*;
class ForEachDemo
{       public static void main(String args[])
        {       int arr[]={10,20,30,40,50};
                String names[]={"priya","madhu","vinnu","prannu"};
                out.println("values....");
                for(int v:arr)
                {out.println(v);
                }
               out.println("Names....");
                for(String name:names)
                {out.println(name);
                }
                //Vector is a collection
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                Vector<String> v=new Vector<String>();
                v.add("giri");
                v.add("sobha");
                v.add("shekar");
                v.add("madhu");
                v.add("narayanarao");
                v.add("krishnaveni");
                out.println("Names....");
                for(String name:v)
                {out.println(name);
                }
        }
}
```

Arrays

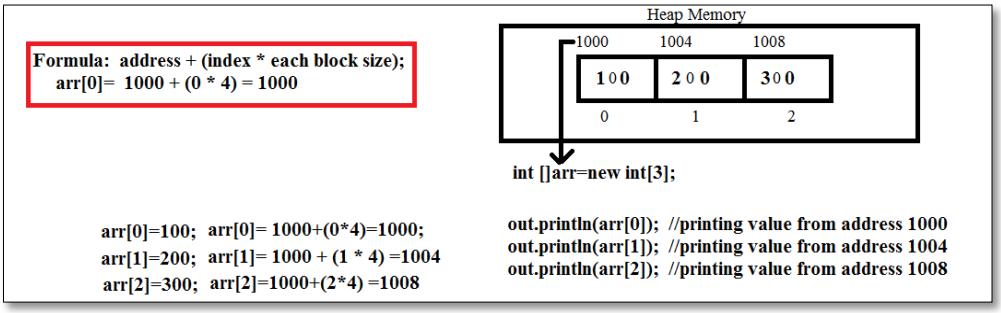


Types of arrays

- **Single dimensional Arrays**
- **Multi dimensional Arrays**

Single Dimensional Arrays (1 D array)

- It is an array with single subscript
- A list of items can be given one variable name using only one subscript such a variable is called a single-subscripted variable or a one-dimensional array.



|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

How to create an array

```
import static java.lang.System.*;
class ArrayDemo1
{	public static void main(String args[])
	{
		//at the time of declaring reference variable we can place the subscript either left side or right side of the variable.
		//but, at the time of creating array by using new operator the subscript must be placed after the data type.
		int []arr=new int[3];
		int []arr2=null;
		out.println("arr:\t"+arr);
		out.println("arr2:\t"+arr2);
		arr[0]=100;
		arr[1]=200;
		arr[2]=300;
		out.println("Elements in an array");
		for(int v:arr)
		{out.println(v);
		}
	}
}
```

SDA2.java

```
import static java.lang.System.*;
class SDA2
{	public static void main(String args[])
	{	int arr[]=new int[3];
		arr[0]=100;
		arr[1]=200;
		arr[2]=300;
		//i=0,1,2,3
		for(int i=0;i<arr.length;i++)
		{out.println(arr[i]);
		}
	}
}
```

ArrayDemo2.java

```
import static java.lang.System.*;
import static java.lang.System.*;
import java.util.*;
class ArrayDemo2
{		public static void main(String args[])
		{ int arr[]=new int[3];
		  Scanner s=new Scanner(in);
		  out.println("Enter "+arr.length+" elements");
		  //i=0,1,2,3
		  //arr.length=3
		  for(int i=0;i<arr.length;i++)
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                    {arr[i]=s.nextInt();
                    }
                    out.println("Elements in an array");
                    for(int v:arr)
                    {out.print(v+"\t");
                    }
                }
}
```

ArrayDemo2.java

```
import static java.lang.System.*;
import java.util.*;
class RW
{               static void read(int []arr)
                { out.println("arr.length:\t"+arr.length);
                  Scanner s=new Scanner(in);
                  out.println("Enter  "+arr.length+"Elements");
                  for(int i=0;i<arr.length;i++)
                  {         arr[i]=s.nextInt();
                  }
                }
                static void write(int[]  arr)
                { out.println("Elements in an array...");
                  for(int i=0;i<arr.length;i++)
                  {         out.print(arr[i]+"\t");
                  }
                }
}
class ArrayDemo2
{               public static void main(String args[])
                { int []arr=new int[3];
                  RW.read(arr);
                  RW.write(arr);
                }
}
```

ArrayDemo3.java

```
import static java.lang.System.*;
import java.util.*;
class RW
{               static void read(int []arr)
                { out.println("arr.length:\t"+arr.length);
                  Scanner s=new Scanner(in);
                  out.println("Enter  "+arr.length+"Elements");
                  for(int i=0;i<arr.length;i++)
                  {         arr[i]=s.nextInt();
                  }
                }
                static void write(int[]  arr)
                { out.println("Elements in an array...");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

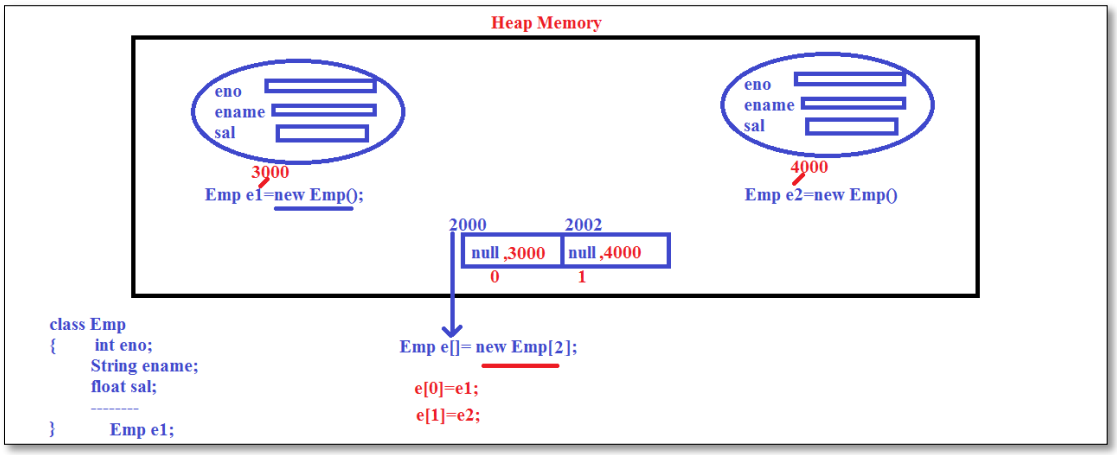
```
                    for(int i=0;i<arr.length;i++)
                    {       out.print(arr[i]+"\t");
                    }
                }
}
class ArrayDemo3
{               public static void main(String args[])
                {
                  //array initialization at the time of declaration
                  //int []arr=new int[]{10,20,30,40,50,60,70,80,90,100};
                  int arr[]={10,20,30,40,50,60,70,80,90,100};
                  RW.write(arr);
                }
}
```

ReadDemo5.java

```
import static java.lang.System.*;
import java.util.*;
class RW
{       static void read(int []arr)
        {       out.println("arr.length:\t"+arr.length);
                Scanner s=new Scanner(in);
                out.println("Enter  "+arr.length+"Elements");
                for(int i=0;i<arr.length;i++)
                {       arr[i]=s.nextInt();
                }
        }
        static void write(int[]  arr)
        {       out.println("Elements in an array...");
                for(int i=0;i<arr.length;i++)
                {       out.print(arr[i]+"\t");
                }
        }
}
class ArrayDemo4
{       public static void main(String args[])
        {       int arr[]=null;
                Scanner s=new Scanner(in);
                out.println("Enter the size of array..");
                int n=s.nextInt();
                arr=new int[n];
                RW.read(arr);
                RW.write(arr);
        }
}
```

Creation of array of objects

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.



ArrayOfObjects.java

```
import static java.lang.System.*;
class Emp
{	int eno;
	String ename;
	float sal;
	void setEno(int eno)
	{this.eno=eno;
	}
	//this means present object
	void setEname(String ename)
	{this.ename=ename;
	}
	//this means present object
	void setSal(float sal)
	{this.sal=sal;
	}
	void display()
	{	out.println("Emp details");
		out.println("Eno:\t"+eno);
		out.println("Ename:\t"+ename);
		out.println("Sal:\t"+sal);
	}
}
class ArrayOfObjects
{	public static void main(String args[])
	{	Emp e1=new Emp();
		e1.setEno(101);
		e1.setEname("Madhu");
		e1.setSal(30000.00f);

		Emp e2=new Emp();
		e2.setEno(102);
		e2.setEname("Shekar");
		e2.setSal(40000.00f);
		Emp e[]=new Emp[2];
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                e[0]=e1;
                e[1]=e2;
                out.println("Elements in an array ....");
                for(Emp emp:e)
                {emp.display();
                }
        }
}
```

Multidimensional Arrays:

- It is an array, which is created by using two or more subscripts.

Two dimensional arrays

- Collection of single dimensional arrays
- If we create an array with two subscripts then it is called as a two dimensional array.

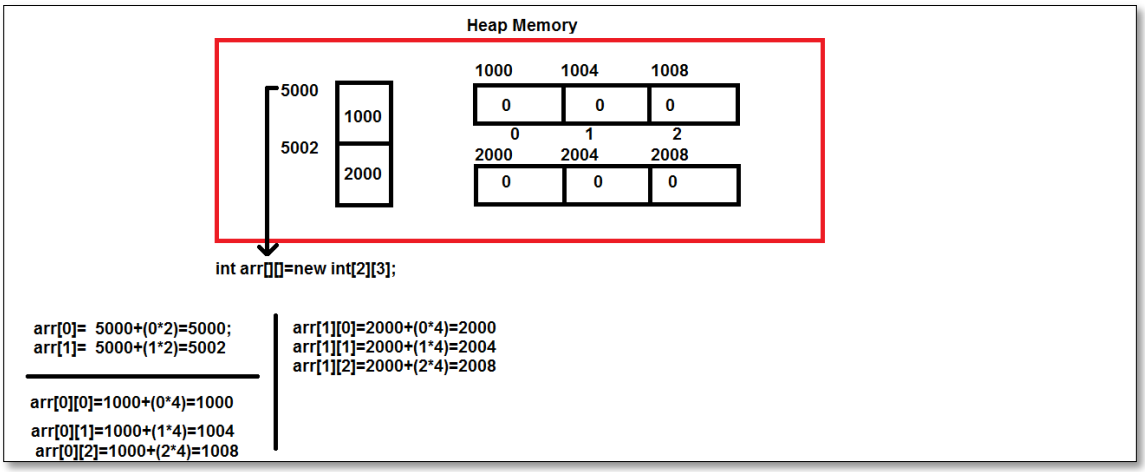
TwoDimArray1.java

```
import static java.lang.System.*;
class TwoDimArray1
{       public static void main(String args[])
        {       //a two dimensional array is nothing but collection of single dimensional arrays
                int arr[][]=new int[2][3];
                out.println(arr.length);
                //arr[0]=  5000+(0*2)=5000
                out.println(arr[0]);
                //arr[1]=  5000+(1*2)=5002
                out.println(arr[1]);
                out.println(arr[0].length);
                out.println(arr[1].length);
                //arr[0][0]=1000+(0*4)=1000;
                out.println(arr[0][0]);
                //arr[0][1]=1000+(1*4)=1004;
                out.println(arr[0][1]);
                //arr[0][2]=1000+(2*4)=1008;
                out.println(arr[0][2]);

                //arr[1][0]=2000+(0*4)=2000;
                out.println(arr[1][0]);

                //arr[1][1]=2000+(1*4)=2004;
                out.println(arr[1][1]);

                //arr[1][2]=2000+(2*4)=2008;
                out.println(arr[1][2]);
        }
}
```

TwoDimArray2.java

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
import static java.lang.System.*;
class TwoDimArray2
{	public static void main(String args[])
	{	int [][]arr=new int[2][3];
		out.println("arr:\t"+arr);
		out.println("arr[0]:\t"+arr[0]);
		out.println("arr[1]:\t"+arr[1]);
		out.println("arr.length:\t"+arr.length);
		out.println("arr[0].length:\t"+arr[0].length);
		out.println("arr[1].length:\t"+arr[1].length);
		out.println("Elements in an array...");
		//i=0,1,2
		//arr.length=2
		for(int i=0;i<arr.length;i++)
		{	//j=0,1,2,3
			//arr[1].length=3
			for(int j=0;j<arr[i].length;j++)
			{out.print(arr[i][j]+"\t");
			//j++;
			}
			out.println();
			//i++;
		}

	}
}
```



TwoDimArray3.java

```
import static java.lang.System.*;
import java.util.*;
class TwoDimArray3
{
	public static void main(String args[])
	{	int arr[][]=new int[2][3];
		Scanner s=new Scanner(in);
		//arr.length=2
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                //i=0,1,2
                out.println("Enter elements into an array.....");
                for(int i=0;i<arr.length;i++)
                {       //j=0,1,2,3
                        //arr[1].length=3
                        for(int j=0;j<arr[i].length;j++)
                        {arr[i][j]=s.nextInt();
                        }
                        out.println();
                }
                out.println("Elements of an array....");
                //i=0,1,2
                //arr.length=2
                for(int i=0;i<arr.length;i++)
                {
                        //j=0,1,2,3
                        //arr[1].length=3
                        for(int j=0;j<arr[i].length;j++)
                        {
                                out.print(arr[i][j]+"\t");
                        }
                        out.println();
                }
        }
}
```

TwoDimArray2.java

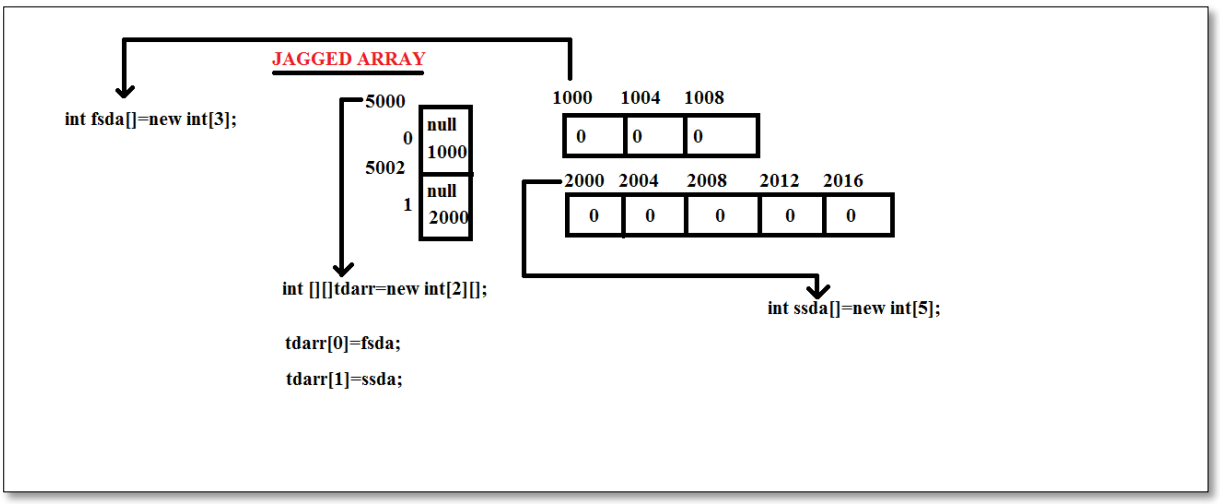
```
import static java.lang.System.*;
import java.util.*;
class RW
{       static void write(int [][]arr)
        {       out.println("Elements in an array...");
                for(int i=0;i<arr.length;i++)
                {       for(int j=0;j<arr[i].length;j++)
                        {out.print(arr[i][j]+"\t");
                        }
                        out.println();
                }
        }
        static void read(int [][]arr)
        {       Scanner s=new Scanner(in);
                out.println("Enter Elements....into 2d array...");
                for(int i=0;i<arr.length;i++)
                {
                        out.println("Enter Elements into row:  "+(i+1));
                        for(int j=0;j<arr[i].length;j++)
                        {arr[i][j]=s.nextInt();
                        }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                }
        }
}
class TwoDimArray2
{
        public static void main(String args[])
        {       int [][]arr=new int[2][3];
                RW.read(arr);
                RW.write(arr);
        }
}
```

JaggedArrayDemo.java

Jagged array is array of arrays such that member arrays can be of different sizes, i.e., we can create a 2-D arrays but with variable number of columns in each row. These type of arrays are also known as ragged arrays.



JaggedArrayDemo.java

```
import static java.lang.System.*;
import java.util.*;
class RW
{       static void write(int [][]arr)
        {       out.println("Elements in an array...");
                for(int i=0;i<arr.length;i++)
                {       for(int j=0;j<arr[i].length;j++)
                        {out.print(arr[i][j]+"\t");
                        }
                        out.println();
                }
        }
        static void read(int [][]arr)
        {       Scanner s=new Scanner(in);
                out.println("Enter Elements....into 2d array...");
                for(int i=0;i<arr.length;i++)
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
		{
			out.println("Enter Elements into row:  "+(i+1));
			for(int j=0;j<arr[i].length;j++)
			{arr[i][j]=s.nextInt();
			}

		}
	}
}
class JaggedArray1
{	public static void main(String args[])
	{	int arr[][]=new int[2][];
		out.println("arr:\t"+arr);
		out.println("arr[0]:\t"+arr[0]);
		out.println("arr[1]:\t"+arr[1]);
		int fsda[]=new int[2];
		int ssda[]=new int[3];
		arr[0]=fsda;
		arr[1]=ssda;
		RW.read(arr);
		RW.write(arr);
	}
}
```

JaggedArray2.java

```
import static java.lang.System.*;
import java.util.*;
class RW
{	static void write(int [][]arr)
	{	out.println("Elements in an array...");
		for(int i=0;i<arr.length;i++)
		{	for(int j=0;j<arr[i].length;j++)
			{out.print(arr[i][j]+"\t");
			}
			out.println();
		}
	}
	static void read(int [][]arr)
	{	Scanner s=new Scanner(in);
		out.println("Enter Elements....into 2d array...");
		for(int i=0;i<arr.length;i++)
		{
			out.println("Enter Elements into row:  "+(i+1));
			for(int j=0;j<arr[i].length;j++)
			{arr[i][j]=s.nextInt();
			}

		}
	}
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
class JaggedArray2
{	public static void main(String args[])
	{	int arr[][]=new int[][]{
					{10},
					{10,20},
					{10,20,30}
					};
		RW.write(arr);
	}
}
```

AnonymousArray.java

If we create an array without giving name and size then it is called as Anonymous Array.

Example

```
import static java.lang.System.*;
import java.util.*;
class RW
{	static void write(int []arr)
	{	out.println("Elements in an array...");
		for(int i=0;i<arr.length;i++)
		{	out.print(arr[i]+"\t");
		}
	}
}
class AnonymousDemo
{	public static void main(String args[])
	{	RW.write(new int[]{10,20,30,40,50});
	}
}
```

Foreach: It is an extended for loop. It is used to get the elements from an array or collection.

```
import static java.lang.System.*;
class AnonymousDemo
{	public static void main(String args[])
	{	int arr[]=new int[]{10,20,30,40,50};
		out.println("Elements in an array...");
		for(int v:arr)
		{	out.print(v+"\t");
		}
	}
}
```

ArraySort.java

```
import static java.lang.System.*;
import java.util.*;
class SortArray
{	public static void main(String args[])
	{	String names[]={"Madhu","Priya","Vinnu","Prannu","Swetha"};
		int marks[]={90,60,60,70,80};
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                Arrays.sort(names);
                out.println("Names");
                for(String name:names)
                {out.println(name);
                }
                Arrays.sort(marks);
                out.println("Marks");
                for(int mark:marks)
                {out.println(mark);
                }
        }
}
```

Object Oriented Programming Concepts

It is an approach which provides the way of modularising the programs, by creating portioned memory areas for both data and functions.

Class:

- It is a blue print for objects, which contains data and methods
- It is a model for objects, which contains data and methods
- It is a template or prototype for object, which contains data and methods
- It is a user defined data type or an abstract data type, which contains data and methods
- Collection of objects

Object:

- **Instance of a class**
- **It is also called as runtime entity because objects are created at runtime.**

In how many ways we can create an object

There are four ways to create an object

By using

- New operator and constructor calling
- newInstance() method of a class called java.lang.Class
- clone() method of a class called java.lang.Object class
- factory method.

ObjectDemo1.java

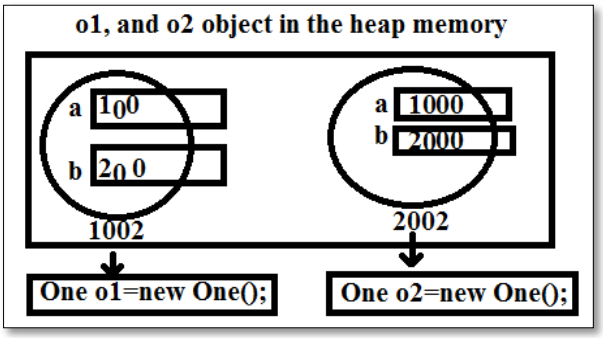
```
import static java.lang.System.*;
class One
{       int a,b;
}
class ObjectDemo1
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
{       public static void main(String args[])
        {       One o1=new One();
                out.println("Object initialization");
                o1.a=100;
                o1.b=200;

                out.println("Object state..");
                out.println("a:\t"+o1.a);
                out.println("b:\t"+o1.b);

                One o2=new One();
                out.println("o2 Object initialization");
                o2.a=1000;
                o2.b=2000;

                out.println("o2 Object state..");
                out.println("a:\t"+o2.a);
                out.println("b:\t"+o2.b);
        }
}
```

Output:

```
E:\LocalNonLocal>javac ObjectDemo1.java
E:\LocalNonLocal>java ObjectDemo1
Object initialization
Object state..
a:      100
b:      200
o2 Object initialization
o2 Object state..
a:      1000
b:      2000
```



ObjectDemo2.java

```
import static java.lang.System.*;
class One
{       int a,b;
        void setState(int i,int j)
        {a=i;
        b=j;
        }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        void display()
        {       out.println("Object state");
                out.println("a:\t"+a);
                out.println("b:\t"+b);
        }
}
class ObjectDemo2
{       public static void main(String args[])
        {       One o1=new One();
                o1.setState(10,20);
                o1.display();

                One o2=new One();
                o2.setState(100,200);
                o2.display();

                One o3=new One();
                o3.setState(1000,2000);
                o3.display();
        }
}
```

Encapsulation:

- Process of binding data with methods
- Process of binding data and methods into a block (or) wrapping up of data and methods into a block.

Encapsulation.java

```
import static java.lang.System.*;
class One
{       private int a;
        void setA(int x)
        {a=x;
        }
        int getA()
        {return a;
        }
}
class Encapsulation
{       public static void main(String args[])
        {       One o1=new One();
                o1.setA(10);
                int r=o1.getA();
                out.println("Returned Value:\t"+r);
        }
}
```

Constructors:

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- Constructor is a special method which has same name as class name.
- Constructor doesn't contain a return type
- It is invoked at the time object creation

Why we use constructors?

- Constructors are used to initialize objects

In the constructors we write the statements like,

- Data base connection establishment
- Setting background and foreground colours for a frame
- Establishing network connection

Etc...

There are two types of constructors we have

- Non parameterized constructors(default constructors)
- Parameterized constructors

DefaultConstructor.java

```
import static java.lang.System.*;
class One
{       private int a,b;
        //non-paramterized constructor
        One()
        {out.println("Default Constructor...");
        a=100;
        b=200;
        }
        void display()
        {out.println("Object state...");
        out.println("a:\t"+a);
        out.println("b:\t"+b);
        }
}
class DefaultConstructor
{       public static void main(String args[])
        {       One o1=new One();
                o1.display();
        }
}
```

Note: If you want to initialize every object with same values, write only default constructor in the class.

Note: if we don't write any constructor in a class, then java compiler creates a default constructor for that class.

DefaultConstructor.java

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
import static java.lang.System.*;
class One
{	private int a,b;
	One()
	{	out.println("Default constructor...");
		a=100;
		b=200;
	}
	void display()
	{	out.println("Object state...");
		out.println("a:\t"+a);
		out.println("b:\t"+b);
	}
}
class ConstructorDemo
{	public static void main(String args[])
	{	One o1=new One();
		One o2=new One();
		One o3=new One();
		o1.display();
		o2.display();
		o3.display();
	}
}
```

Note: If you want to initialize every object with different values, write only parameterized constructor in the class.

What is constructor overloading?

If we write more than one constructor with different signature in a class, then it is called as constructor overloading.

Q) ***What if somebody wants to initilaize object by calling default constructor and somebody wants to initialize the object by calling parameterized constructor?***

Write both default and parameterized constructors in the class

ConstructorOverload.java

```
import static java.lang.System.*;
class One
{	int a,b;
	//Default constructor(default constructor)
	One()
	{out.println("Default constructor..");
	a=100;
	b=200;
	}
	//parameterized constructor
	One(int x,int y)
	{out.println("Parameterized constructor..");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        a=x;
        b=y;
        }
        void display()
        {       out.println("Object state...");
                out.println("a:\t"+a);
                out.println("b:\t"+b);
        }
}
class ConstructorOverload
{       public static void main(String args[])
        {       //o1 ref var contains 1002(address of the object)
                One o1=new One();
                o1.display();
                One o2=new One(1000,2000);
                o2.display();
        }
}
```

Keyword "this"

How do you assign local variable value to instance variable if two variables are same?

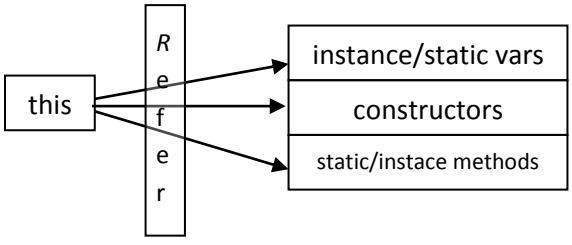
Example:

```
void set(int a)
{       a=a;
}
```

In the preceding code, the local variables is 'a' and the instance variable is also 'a'. if we use 'a' in set () method by default JVM accesses only local variable. Then how we can refer instance variable within set () method. For this purpose we use 'this' keyword.

The keyword 'this':

- The keyword 'this' refers the present class object.
- This is a pointer which points to the present object
- Generally we write instance variables and static variables, methods and constructors in a class, all these members are referenced by 'this' keyword.
- When an object is created to a class, a default reference is also created internally to that object, it refers every property of an object as shown in the below figure.



Where we use "this" keyword in most cases?

- If you want to assign the local variable values to the instance fields even though both local and instance variable names are same then we use this keyword.
- This keyword is also used to call the same class constructor, with in another constructor.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Now the set method can be written as

```
voi d set(int a)
{this.a=a;
}
```

ThisDemo.java

```
import static java.lang.System.*;
class One
{	int a;
	static int b=100;
	void set(int a,int b)
	{	this.a=a;
		this.b=b;
		//One.b=b;
		this.get();
	}
	static void get()
	{	out.println("One class get() method");
		out.println("static b:\t"+b);
	}
}
class ThisDemo
{	public static void main(String[] args)
	{	One o=new One();
		o.set(10,200);
	}
}
```

ThisDemo2.java

```
import static java.lang.System.*;
class One
{	int a,b;
	One(int a,int b)
	{out.println("Param constructor");
	this.a=a;
	this.b=b;
	}
	void display()
	{	out.println("Object state...");
		out.println("a:\t"+a);
		out.println("b:\t"+b);
	}
}
class ThisDemo1
{	  public static void main(String args[])
	{	One o2=new One(100,200);
		o2.display();
	}
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Note:

Using this keyword we can call the same class constructors. But constructor calling statement in a constructor must be the first statement.

ThisDemo2.java (calling same class constructor in another constructor)

```
import static java.lang.System.*;
class One
{       int a,b;
        One(int a)
        {out.println("1st param con...");
        this.a=a;
        }
        One(int a,int b)
        {this(a);
        out.println("Param constructor");
        this.b=b;
        }
        void display()
        {       out.println("Object state...");
                out.println("a:\t"+a);
                out.println("b:\t"+b);
        }
}
class ThisDemo2
{
        public static void main(String args[])
        {       One o2=new One(100,200);
                o2.display();
        }
}
E:\ThunderBolts>javac ThisDemo2.java
E:\ThunderBolts>java ThisDemo2
1st param con...
Param constructor
Object state...
a:      100
b:      200
```

ThisDemo3.java

```
import static java.lang.System.*;
class One
{       int a,b,c;
        One(int a)
        {out.println("initializing a ");
        this.a=a;
        }
        One(long b)
        {out.println("initializing b ");
        this.b=(int)b;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
	}
	One(float c)
	{out.println("initializing c ");
	this.c=(int)c;
	}
	void display()
	{	out.println("Object state...");
		out.println("a:\t"+a);
		out.println("b:\t"+b);
		out.println("c:\t"+c);
	}
}
class ThisDemo3
{	public static void main(String args[])
	{	One o1=new One(100);
		o1.display();
		One o2=new One(100l);
		o2.display();
		One o3=new One(100.00f);
		o3.display();
	}
}
```

MethodOverloading: We can write more than one method with same name with different signature in a class. It is called as method overloading.

What is method signature?

Method name and type, count and order of arguments are called as method signature

MethodOverload.java

```
import static java.lang.System.*;
class Calc
{
	int add(int a,int b)
	{return a+b;
	}
	int add(int a,int b,int c)
	{return a+b+c;
	}
	float add(int a,float b)
	{return a+b;
	}
	float add(float a,int b)
	{return a+b;
	}
	/*
	add(int,int);
	add(int,int,int);
	add(int,float);
	add(float,int);
	*/
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
}
class MethodOverload
{	public static void main(String args[])
	{	Calc obj=new Calc();
		float s1=obj.add(10,20.50f);
		int s2=obj.add(10,20,30);
		float s3=obj.add(10.50f,20);
		int s4=obj.add(10,20);
		out.println("s1:\t"+s1);
		out.println("s2:\t"+s2);
		out.println("s3:\t"+s3);
		out.println("s4:\t"+s4);
	}
}
```

Constructor Overloading:

We can write more than one constructor with different signature in a class. it is called as constructor overloading.

ConstructorOverload.java

```
import static java.lang.System.*;
class One
{	int a,b,c;
	One(int a)
	{this.a=a;
	}
	One(int a,int b)
	{One(a);
	this.b=b;
	}
	One(int a,int b,int c)
	{this(a,b);
	this.c=c;
	}
	void One(int a)
	{out.println("In method one");
	out.println("a:\t"+a);
	}
	void display()
	{	out.println("Object state...");
		out.println("a:\t"+a);
		out.println("b:\t"+b);
		out.println("c:\t"+c);
	}
}
class ConstructorOverload
{
	public static void main(String args[])
	{
		One o1=new One(10,20,30);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                o1.display();
        }
}
```

Note: we can write a method in the class with same name as class name.

Variables

Variables are divided into 3 categories based on the memory allocation area.

- Instance variables
- Static variables
- Local variables

Instance Variables:

- It is a variable which is declared with in a class outside the method and not declared as static.
- Instance variables are created during every object creation.
- We can access instance variable in other classes by using object only.

Static Variables:

- It is a variable which is declared with in a class outside the method and declared by using a non-access modifier called static.
- These variables are created only once during the program execution, at the time of class loads into the method area.
- We can access static variables in other classes by using class name or a reference variable even though it contains null.
- Static variable are created in method are.

Local Variables:

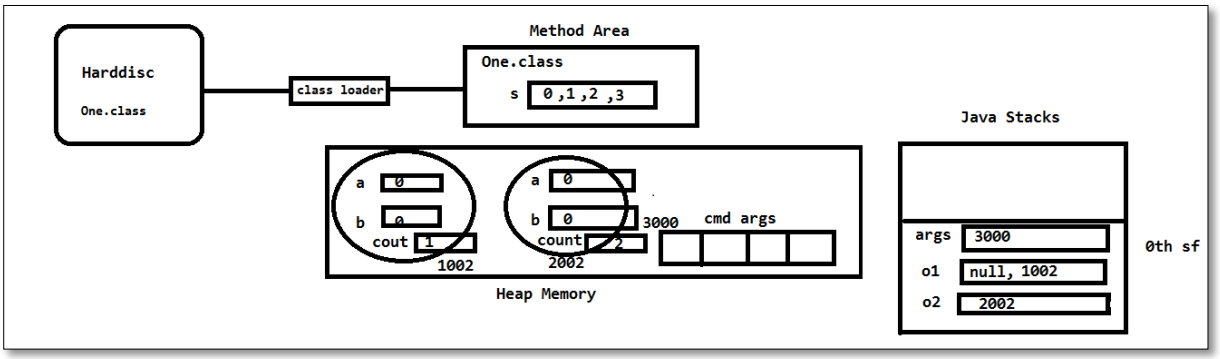
- If we declare a variable inside a block which is existed in a class then it is called as local variable.
- These variables are created during that particular block execution.

Fields.java

```
import static java.lang.System.*;
class One
{       static int s;
        int count;
        int a,b;
        static{
        out.println("Class is loaded..");
        s=1;
        }
        //instance block
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {count=s++;
        }
        One()
        {out.println("Default..constructor...");
        }
        One(int a,int b)
        {       out.println("Param...constructor...");
                this.a=a;
                this.b=b;
                count=s++;
        }
        void display()
        {       out.println("Object "+count+" state");
                out.println("a:\t"+a);
                out.println("b:\t"+b);
        }
}
class StaticFields
{       static public void main(String args[])
        {       out.println(One.s);
                One o1=null;
                out.println(o1.s);
                o1=new One();
                One o2=new One();
                One o3=new One(10,20);
                o1.display();
                o2.display();
                o3.display();
        }
}
```

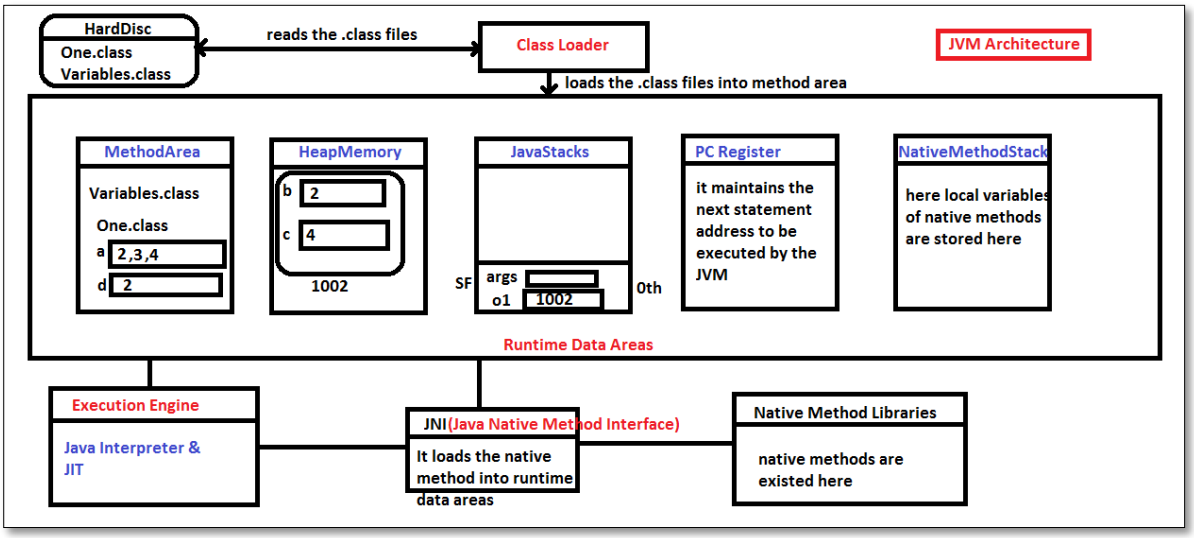


Variables.java

```
import static java.lang.System.*;
class One
{       static int a=2;
        int b=a++;
        int c=++a;
        static int d=a;
        int add(int x,int y)
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {int z=x+y;
        return z;
        }
        void display()
        {       out.println("values.....");
                out.println("a:\t"+a);
                out.println("b:\t"+b);
                out.println("c:\t"+c);
                out.println("d:\t"+d);
        }
}
class Variables
{       public static void main(String args[])
        {       One o1=new One();
                o1.display();
                One o2=new One();
                o2.display();
        }
}
```



Static and instance methods:

Instance method:

- It is a method which is not declared as static
- We can access instance methods in other classes by using Object
- It can access all the members of its class directly

Static methods:

- It is a methods which is declared by using a keyword called static
- We can access these methods in other classes by using class name or reference variable eventhough it contains null.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- Static methods can't access instance methods or fields, of the same class directly. But we can access them by using object.

Example:

```
import static java.lang.System.*;
class One
{       int a=100;              //instance variable
        static int s=1000;      //static variable
        void display()
        {       out.println("Instance method...");
                out.println("s:\t"+s);
                out.println("a:\t"+a);
                get();
        }
        static void get()
        {       out.println("static method.....");
                out.println("s:\t"+s);
                //Error: we can't use instance var in static block directly
                //out.println("a:\t"+a);
                // Error: we can't use instance method in static block directly
                //display();
        }
}
class StaticMethod
{       public static void main(String args[])
        {       One.get();
                One o1=new One();
                o1.display();
        }
}
```

Example:

```
class One
{
        int a=100;              //instance variable
        static int s=1000;      //static variable
        void display()
        {out.println("Instance method...");
        }
        static void get()
        {out.println("static method.....");
        One o=new One();
out.println("a:\t"+o.a);//Accessing  instance variable in static method by using object
        o.display();////Accessing instance method in static method by using object
        }
}
```

SIMethods.java

```
import static java.lang.System.*;
class One
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
{       int a;
        void display()
        {out.println("Instance method...");
        }
        static void get()
        {out.println("static method.....");
        }
}
class SIMethods
{       public static void main(String args[])
        {       One o1=null;
                One.get();
                o1.get();
                o1=new One();
                o1.display();
        }
}
```

Note: if you try to call the instance method by using reference variables which contains null. Then we will get a NullPointerException.

SIMethods2.java

```
import static java.lang.System.*;
class One
{       int a=100;              //instance variable
        static int s=1000;      //static variable
        void display()
        {out.println("Instance method...");
        out.println("a:\t"+a);
        out.println("s:\t"+s);
        }
        static void get()
        {       out.println("static method.....");
                out.println("s:\t"+s);
                One o=new One();
                out.println("a:\t"+o.a);
                o.display();
        }
}
class SIMethods2
{       public static void main(String args[])
        {       One.get();
                One o1=new One();
                o1.display();
        }
}
```

Static Block & Instance Block

Static Block:

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- It is a block which is declared by using a keyword called static and It contains no name.
- It is a block which is invoked at the time of class loads into the method area.
- If you want to execute set of statements at the time of class loading then we have to write those statements in the static block.

StaticBlock.java

```
import static java.lang.System.*;
class One
{	static int a;
	static{
		out.println("static block");
		a=10000;
	}
}
class StaticBlock
{	public static void main(String args[])
	{out.println(One.a);
	}
}
```

Instance Block:

- It is a block which contains no name and not declared as static.
- It is a block which is invoked during every object creation.
- If you want to execute set of statements during every object creation, we have to write those statements in an instance block.

InstanceBlock.java

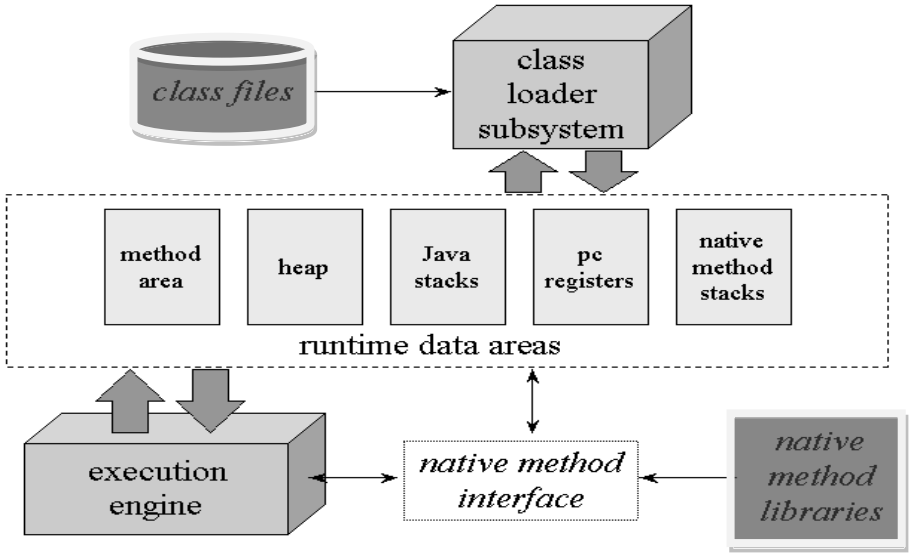
```
import static java.lang.System.*;
class One
{	int a;
	//instance block
	{	out.println("Instance block");
	}
	One()
	{out.println("Non-parameterized constructorn...");
	}
	One(int a)
	{out.println("Parameterized constructorn...");
	this.a=a;
	}
}
class InstanceBlock
{	public static void main(String args[])
	{	One o1=new One();
		One o2=new One(10000);
	}
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

JAVA VIRTUAL MACHINE

The Java Virtual Machine

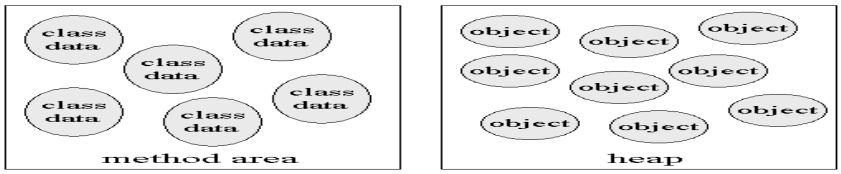
- Java virtual machine is a simulated computer in the computer which is used to run java programs.
- JVM(Java Virtual Machine) is the heart of java program execution process.
- JVM is responsible for taking '.class' files and converts the each byte code instruction in to the machine code instruction that is executed by the microprocessor. The below figure shows the architecture of the Java Virtual Machine



- Take the previous example **"Addition.java"** when we compile this program **,** the compiler generates two *".class" files* those are **Add.class** and **Addition.class.** Remember that java compiler is not part of **JVM.** Compiler is always existed outside the **JVM.**
- The Java virtual machine organizes the memory it needs to execute a program into several *runtime data areas*.
- when we run the program by using this command "java Addition" one new JVM instance will be created with method area, and heap memory.
- Each instance of the Java virtual machine has one *method area* and one *heap*.
- These areas are shared by all threads running inside the virtual machine.(ex: main thread (non daemon thread),garbage collector thread(daemon-thread) and etc…).
- After the new instance of JVM is created, the class loader (of Java virtual machine) loads the initial ".class" file (i.e… Addition.class),and then it parses the data contained in the class file in to a type, after it stores it in method area.
- Each Java virtual machine has an *execution engine*: a mechanism responsible for executing the instructions contained in the methods of loaded classes.
- As each new thread comes into existence, it gets its own *pc register* (program counter) and *Java stack*.
- If the thread is executing a Java method (not a native method), the value of the pc register indicates the next instruction to execute.
- A thread's Java stack stores the state of Java (not native) method invocations for the thread.
- The state of a Java method invocation includes its local variables, the parameters with which it was invoked, its return value (if any), and intermediate calculations.
- The state of native method invocations is stored in *native method stacks.*

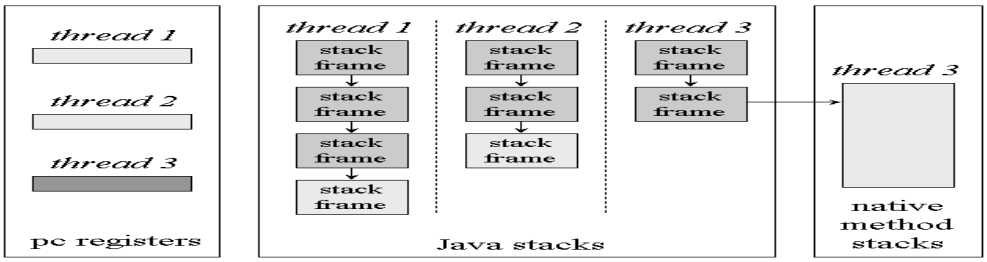
|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- The Java stack is composed of *stack frames* (or *frames*).
- A stack frame contains the state of one Java method invocation



Runtime data areas shared among all threads.

See The below Figure for a graphical depiction of the memory areas the Java virtual machine creates for each thread. These areas are private to the owning thread. No thread can access the pc register or Java stack of another thread.



Runtime data areas exclusive to each thread

- The above Figure shows a snapshot of a virtual machine instance in which three threads are executing.
- At the instant of the snapshot, threads one and two are executing Java methods. Thread three is executing a native method.
- The above Figure as in all graphical depictions of the Java stack in this book, the stacks are shown growing downwards.
- The "top" of each stack is shown at the bottom of the figure.
- Stack frames for currently executing methods are shown in a lighter shade.
- For threads that are currently executing a Java method, the pc register indicates the next instruction to execute.
- In The above Figure such pc registers (the ones for threads one and two) are shown in a lighter shaded
- Because thread three is currently executing a native method, the contents of its pc register--the one shown in dark gray--is undefined.

ClassLoader:

- when JVM wants to load a class file it uses class loader to locates the appropriate class files.
- Class loader reads the class file(using a binary stream) after that the class file is checked by the byte code verifier and then it will send to the **JVM**.
- **JVM** extracts binary data and stores information in the method area.

The Method Area:

- Method area contains class data.
- Static variable (class variables) are also stored here.
- The size of the method area need not be fixed. As the Java application runs, the virtual machine can expand and contract the method area to fit the application's needs.

Java Stack:

- A new Java stack is created for each thread, java stack is the combination of stack frames, each stack frame is used to store the state of a method (state of method means parameters,variables declared in that method and return values).
- When a thread invokes a method, the Java virtual machine pushes a new frame onto that thread's Java stack.
- When the method completes, the virtual machine pops and discards the frame for that method

PcRegister:

- Like java stack it will be created as new instance of thread comes into existence. the pc register indicates the next instruction to execute.

Native Method Stack:

- Which is used to store the state of a native method?

How both Java interpreter and JIT(Just Intime Compiler) works?

In JVM we got interpreter and JIT, both working at the same time on byte code to translate into machine code. Now, the main question is why both are needed and how both work simultaneously? To understand this, let us take some sample code. Assume these are byte code instructions.

```
out.println(a);

out.println(b);

for(int i=1;i<=10;i++)

out.println(a);
```

When, the interpreter starts execution from 1st instruction, it converts **out.println(a);** into machine code and gives it to the microprocessor. For this, say the interpreter has taken 2 nanoseconds time. The processor takes it, executes it, and the value of a is displayed. Now, the interpreter comes back into memory and reads the 2nd instruction **out.println(b);** to convert this into machine code, it takes another 2 nanoseconds. Then the instruction is given to the processor and it executes it.

Next, the interpreter reads the 3rd instruction, which is a looping statement **out.println(a)**; this should be done 10 times and this is known to the interpreter. If the interpreter executes this instruction it takes 10x2=20 nanoseconds because it converts the **out.println(a);** instruction 10 times. Because it will takes so much time JVM will not allots this code to the interpreter. It allots this code to the **JIT** compiler.

Let us see how the JIT compiler will execute the looping instruction. JIT compiler reads the **out.println(a);** instruction and converts that into machine code. For this, say, it is taking 2

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

nanoseconds. Then the JIT compiler allots a block of memory and pushes this machine code instruction into that memory. For this, say it is taking another 2 nanoseconds. That means **JIT** compiler has taken a total of 4 nanoseconds. Now, the processor will fetch this instruction from memory and executes it 10 times. Recognize that the first two instructions will not be allotted to JIT compiler by the JVM. The reason is clear. Here it takes 4 nanoseconds to convert each instruction whereas the interpreter actually took only 2 nanoseconds.

After loading the **.class** code into memory, JVM first of all identifies which code is to be left to interpreter and which one to JIT compiler so that the performance is better. ***The blocks of code allocated for JIT compiler are also called hotspots***. Thus, both the interpreter and JIT compiler will work simultaneously to translate the byte code into machine code.

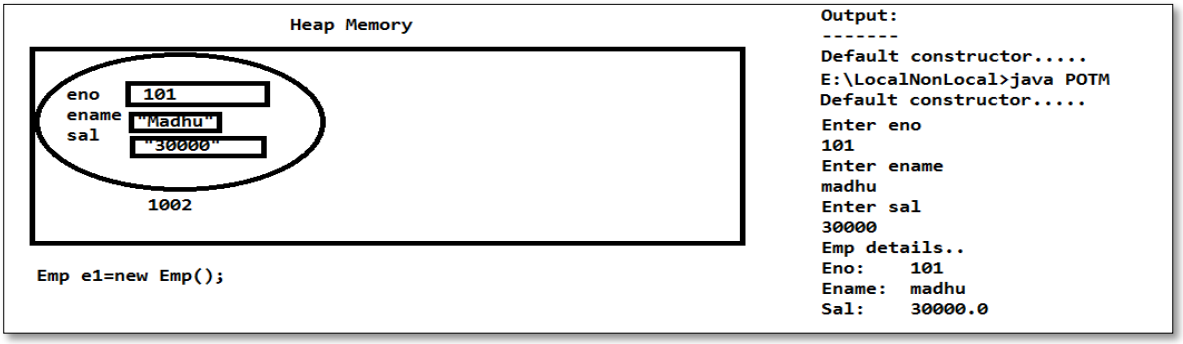
Passing Objects to methods

POTM.java

```
import static java.lang.System.*;
import  java.util.*;
class Emp
{       int eno;
        String ename;
        float sal;
        Emp()
        {out.println("Default constructor.....");
        }
        void setEno(int eno)
        {this.eno=eno;
        }
        void setEname(String ename)
        {this.ename=ename;
        }
        void setSal(float sal)
        {this.sal=sal;
        }
        void display()
        {       out.println("Emp details..");
                out.println("Eno:\t"+eno);
                out.println("Ename:\t"+ename);
                out.println("Sal:\t"+sal);
        }
}
class RW
{       static void read(Emp e)
        {       Scanner s=new Scanner(in);
                out.println("Enter eno");
                int eno=s.nextInt();
                e.setEno(eno);
                out.println("Enter ename");
                String ename=s.next();
                e.setEname(ename);
                out.println("Enter sal");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                float sal=s.nextFloat();
                e.setSal(sal);
        }
}
class POTM
{       public static void main(String args[])
        {       Emp e1=new Emp();
                RW.read(e1);
                e1.display();
        }
}
```

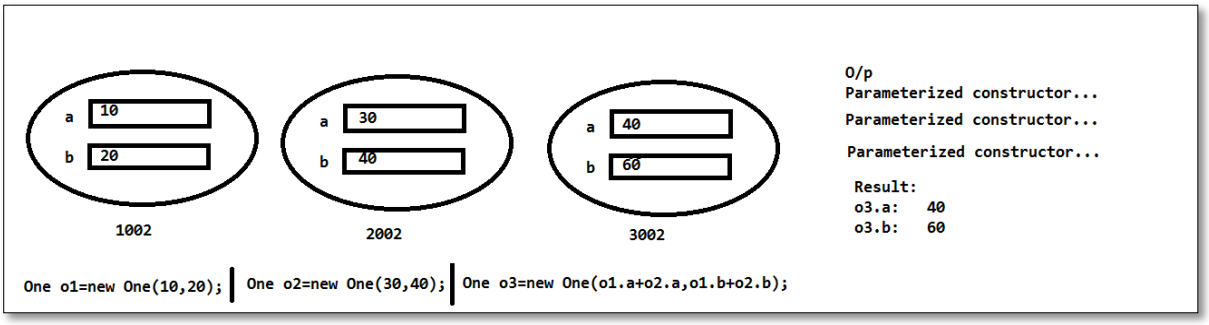


POTM2.java

```
import static java.lang.System.*;
class One
{       int a,b;
        One()
        {out.println("Default constructor...");
        }
        One(int a,int b)
        {out.println("Parameterized constructor...");
        this.a=a;
        this.b=b;
        }
}
class Demo
{       static One add(One o1,One o2)
        {       One o3=new One();
                o3.a=o1.a+o2.a;
                o3.b=o1.b+o2.b;
                return o3;
        }
}
class POTM2
{       public static void main(String args[])
        {       One o1=new One(10,20);
                One o2=new One(30,40);
                One o3=Demo.add(o1,o2);
                out.println("Result:\t");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("o3.a:\t"+o3.a);
                out.println("o3.b:\t"+o3.b);
        }
}
```



Inheritance

- Process of including members of one class into another class
- Advantage of inheritance is reusability.

Basically we have two forms of inheritance

- **Simple(single) level**
- **Multiple**

Other forms (extended forms)

- **Multilevel**
- **Hierarchical**
- **Hybrid**

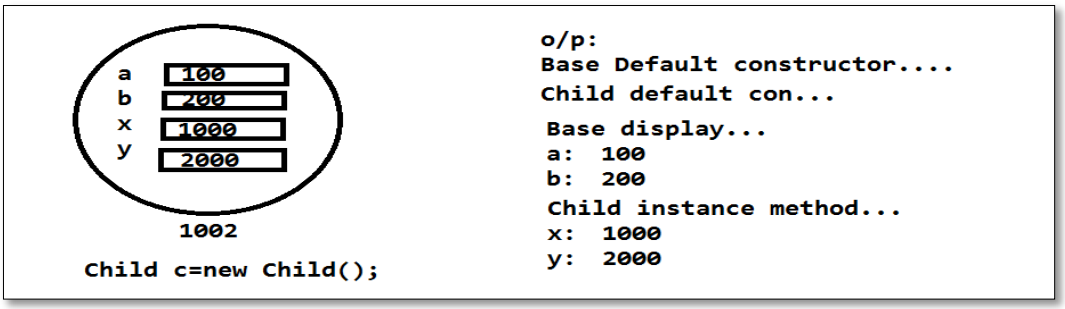
Single-level inheritance: If a child class inherits only one parent class then it is called as single level inheritance.

SingleLevel.java

```
import static java.lang.System.*;
//Base class is also called as parent or super class
class Base
{       int a,b;
        Base()
        {
        out.println("Base Default constructor....");
        a=100;
        b=200;
        }
        void display()
        {out.println("Base display...");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        out.println("a:\t"+a);
        out.println("b:\t"+b);
        }
}
class Child extends Base
{
        int x,y;
        Child()
        {       super(); //parent class default constructor will be invoked here
                out.println("Child default con...");
                x=1000;
                y=2000;
        }
        void get()
        {       out.println("Child instance method...");
                out.println("x:\t"+x);
                out.println("y:\t"+y);
        }
}
class SingleLevel
{       public static void main(String args[])
        {       Child c=new Child();
                c.display();
                c.get();
        }
}
```



SingleLevelInher2.java

```
import static java.lang.System.*;
class Bird
{       String name,color,food,typeOfBird;
        Bird()
        {out.println("Bird constructor...");
        }
        void eat()
        {out.println("Eat    "+food);
        }
        void display()
        {out.println(typeOfBird+" details");
        out.println("Name:\t"+name);
        out.println("Color:\t"+color);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
	}
}
class Parrot extends Bird
{	Parrot(String name,String color,String food,String typeOfBird)
	{out.println("Parrot constructor");
	this.name=name;
	this.color=color;
	this.food=food;
	this.typeOfBird=typeOfBird;
	}
	void fly()
	{out.println("It can fly");
	}
	void talk()
	{out.println("It can talk");
	}
	void sing()
	{out.println("It can sing");
	}
}
class SingleLevel
{	public static void main(String args[])
	{	Parrot p=new Parrot("Rama","Green","Guava","Parrot");
		p.display();
		p.eat();
		p.fly();
		p.talk();
		p.sing();
	}
}
```

SingleLevelInher3.java

```
import static java.lang.System.*;
class Base
{	int a,b;
	Base(int a,int b)
	{	out.println("Base Param constructor");
		this.a=a;
		this.b=b;
	}
	void display()
	{	out.println("a:\t"+a);
		out.println("b:\t"+b);
	}
}
class Child extends Base
{	int x,y;
	Child(int x,int y)
	{	super(10,20);
		out.println("Child param con..");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                this.x=x;
                this.y=y;
        }
        void get()
        {       out.println("x:\t"+x);
                out.println("y:\t"+y);
        }
}
class SingleLevelInher3
{       public static void main(String args[])
        {       Child c=new Child(100,200);
                c.display();
                c.get();
        }
}
```

SingleLevelInher4.java

```
import static java.lang.System.*;
class Base
{       int a,b;
        Base(int a,int b)
        {       out.println("Base Param constructor");
                this.a=a;
                this.b=b;
        }
        void display()
        {       out.println("a:\t"+a);
                out.println("b:\t"+b);
        }
}
class Child extends Base
{       int x,y;
        Child(int a,int b,int x,int y)
        {       super(a,b);
                out.println("Child param con..");
                this.x=x;
                this.y=y;
        }
        void get()
        {       out.println("x:\t"+x);
                out.println("y:\t"+y);
        }
}
class SingleLevelInher4
{       public static void main(String args[])
        {       Child c=new Child(100,200,300,400);
                c.display();
                c.get();
        }
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Super keyword:

- It is a pointer, which points to the super class object.
- By using super keyword we can access all the members of a base class including constructors in the child class

When we use super keyword?

- At the time of super class constructor calling in the child class constructor
- If we want to call the base class method in the child class after overriding it.

What is method overriding?

We can write a method in child class which is already existed in base class, it is called as method overriding

Rules to follow at the time of overriding a method

- Access modifier must be same or higher scope
- Return type, method name must be same
- Type, order and count of arguments must be same
- If base method throws an exception, the child method has to throw the same exception or child type of that exception or nothing
- If base method throws no exception then the child method can't throw any exception.

Multilevel: if a child class is derived from another child class then it is called as multilevel inheritance.

MultiLevelInher.java

```
import static java.lang.System.*;
class Base
{	int a;
	Base(int a)
	{out.println("Base Param constructor...");
	this.a=a;
	}
	void display()
	{out.println("a:\t"+a);
	}
}
class Child extends Base
{	int b;
	Child(int a,int b)
	{	super(a);
		this.b=b;
	}
	void display()
	{	super.display();
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("b:\t"+b);
        }
}
class SubChild extends Child
{       int c;
        SubChild(int a,int b,int c)
        {super(a,b);
        this.c=c;
        }
        void display()
        {super.display();
        out.println("c:\t"+c);
        }
}
class MultiLevelInher
{       public static void main(String args[])
        {
                SubChild sc=new SubChild(100,200,300);
                sc.display();
        }
}
```

Hierarchical Inheritance:

If a single base class is inherited by more than one child class, then it is called as hierarchical inheritance.

HierarchicalInheritance.java

```
import static java.lang.System.*;
class Bird
{       String name,color,food,typeOfBird;
        Bird(String name,String color,String food,String typeOfBird)
        {out.println("Bird constructor...");
        this.name=name;
        this.color=color;
        this.food=food;
        this.typeOfBird=typeOfBird;
        }
        void eat()
        {out.println("Eat    "+food);
        }
        void display()
        {out.println(typeOfBird+" details");
        out.println("Name:\t"+name);
        out.println("Color:\t"+color);
        }
}
class Parrot extends Bird
{       Parrot(String name,String color,String food,String typeOfBird)
        {super(name,color,food,typeOfBird);
        out.println("Parrot constructor");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        }
        void fly()
        {out.println("It can fly");
        }
        void talk()
        {out.println("It can talk");
        }
        void sing()
        {out.println("It can sing");
        }
}
class Peacock extends Bird
{       Peacock(String name,String color,String food,String typeOfBird)
        {super(name,color,food,typeOfBird);
        out.println("Peacock constructor");
        }
        void fly()
        {out.println("It can fly");
        }
        void dance()
        {out.println("It can dance");
        }
}
class HierarchicalInher
{       public static void main(String args[])
        {
        Parrot parrot=new Parrot("Rama","Green","Guava","Parrot");
        Peacock peacock=new Peacock("krish","Peacock Color","seeds","Peacock");
        parrot.display();
        parrot.fly();
        parrot.talk();
        parrot.sing();
        peacock.display();
        peacock.fly();
        peacock.dance();
        }
}
```

Final keyword: it is used to declare final fields, final methods and final classes.

- Final fields are not changeable during the program execution

Ex: final double pi=3.14; //now pi is a constant so we can't change it.

- We can't override final methods
- Final classes are not inheritable

Example: wew can't run the program, but when we compile the program we will get three errors.

```
final class Base
{       final int a=100;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        final void display()
        {a=1000;
        }
}
class Child extends Base
{       void display()
        {}
}
```

Relations

1. One of the main advantage of Object-Oriented programming is code reusability
2. We can reuse the code in two ways
    1. Inheritance (IS-A relationship)
    2. Composition (Strong HAS-A relationship).
    3. Aggregation (Weak Has-A Relationship)

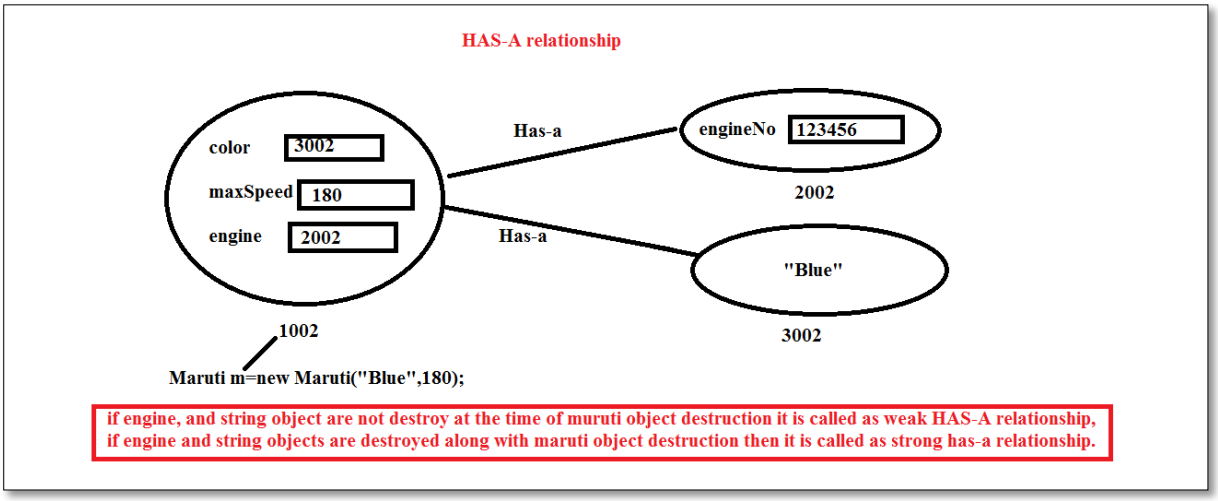
Inheritance/Generalization/Is-A-Relation: It is the process of extracting shared characteristics from two or more classes, and combining them into a generalized super class.

IsARelation.java

```
import static java.lang.System.*;
class Car
{       private String color;
        private int maxSpeed;
        Car(String color,int maxSpeed)
        {this.color=color;
        this.maxSpeed=maxSpeed;
        }
        public void carInfo()
        {out.println("Car Color= "+color + " Max Speed= " + maxSpeed);
        }
}
class Maruti extends Car
{       private Engine engine=new Engine(123456);
        Maruti(String color,int maxSpeed)
        {super(color,maxSpeed);
        }
        public void move()
        {engine.start();
        out.println("riahn....riahn....riahn....riahn....riahn....riahn....riahn");
        }
}
class Engine
{       int engineNo;
        Engine(int engineNo)
        {this.engineNo=engineNo;
        }
        public void start()
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {out.println("Engine No:"+engineNo+" started....");
        }
        public void stop()
        {out.println("Engine Stopped");
        }
}
class IsARelation
{       public static void main(String args[])
        {       Maruti m=new Maruti("Blue",180);
                m.carInfo();
                m.move();
        }
}
```



StrongHasARelation.java

```
import static java.lang.System.*;
final class Car
{       private final Engine engine;
        Car()
        {out.println("Cart creating");
        engine = new Engine("123456","Quadrajet");
        }
        void move()
        {engine.start();
        }
        public void finalize()
        {out.println("Car..Nepothunna..... pothunnaaa...poyaaaa....");
        }
}
class Engine
{       String no;
        String type;
        Engine(String no,String type)
        {out.println("Engine creating");
        this.no=no;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        this.type=type;
        }
        void start()
        {out.println("Car Started with an Engine no is:   "+no+" and type is "+type);
        }
        public void finalize()
        {out.println("Engine Nepothunna.....pothunnaaa...poyaaaa.....");
        }
}
class CompositionDemo
{       public static void main(String args[])
        {
                for(int i=1;i<=1;i++)
                {
                        Car car=new Car();
                        car.move();
                }
                //Explicit invocation of garbage collector
                System.gc();
        }
}
E:\LocalNonLocal>java WeakHashARelation
Start of main
Engine creating
Car Started with an Engine no is:   123456 and type is Quadrajet
End of main
Engine Nepothunna.....pothunnaaa...poyaaaa.....
Car...Nepothunna.....pothunnaaa...poyaaaa.....
```

WeakHasARelation.java

```
import static java.lang.System.*;
final class Car
{       private final Engine engine;
        Car(Engine engine)
        {out.println("Cart creating");
        this.engine=engine;
        }
        void move()
        {engine.start();
        }
        public void finalize()
        {out.println("Nepothunna.....pothunnaaa...poyaaaa.....");
        }
}
class Engine
{       String no;
        String type;
        Engine(String no,String type)
        {out.println("Engine creating");
        this.no=no;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        this.type=type;
        }
        void start()
        {out.println("Car Started with an Engine no is:   "+no+" and type is "+type);
        }
        public void finalize()
        {out.println("Engine Nepothunna.....pothunnaaa...poyaaaa.....");
        }
}
class WeakHasARelation
{       public static void main(String args[])
        {       Engine engine = new Engine("123456","Quadrajet");
                for(int i=1;i<=1;i++)
                {
                        Car car=new Car(engine);
                        car.move();
                }
                //Explicit invocation of garbage collector
                System.gc();
        }
}
Output:
E:\LocalNonLocal>java WeakHashARelation
Start of main
Engine creating
Cart creating
Car Started with an Engine no is:   123456 and type is Quadrajet
End of main
Car...Nepothunna.....pothunnaaa...poyaaaa.....
```

Abstract Class

- It is a class which contains, zero or more abstract methods and some implemented (concrete) methods.
- We can't create object for abstract classes
- We can inherit an abstract class.
- We have to declare an abstract class by using a keyword (non-access modifier) called abstract.
- We can declare a reference variable of abstract class type.

What is an abstract method?

It is a method which contains no implementation. It contains only method declaration (prototype).

Note: if a class contains one or more abstract methods, then we have to declare that class as an abstract class.

Can we declare a class as abstract without containing abstract methods?

Yes we can.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

AbstractDemo.java

```
aimport static java.lang.System.*;
//abstract class
abstract class Bank
{	float minBal=500.00f;
	void openAccount()
	{out.println("Account is created..");
	}
	abstract void withdraw();
	abstract void deposit();
}
//concreate class
class CityBank extends Bank
{
	void withdraw()
	{out.println("withdra...");
	}
	void deposit()
	{out.println("deposit...");
	}
}
class AbstractDemo
{	public static void main(String args[])
	{	CityBank cb=new CityBank();
		cb.openAccount();
		cb.deposit();
		cb.withdraw();

		//upcasting
		//Bank b=cb;
		Bank b=new CityBank();
		b.openAccount();
		b.deposit();
		b.withdraw();
	}
}
```

AbstractDemo2.java

```
import static java.lang.System.*;
import java.util.*;
abstract class Search
{	int arr[]=new int[10];
	void read()
	{	Scanner s=new Scanner(in);
		out.println("Enter "+arr.length+" Elements");
		for(int i=0;i<arr.length;i++)
		{arr[i]=s.nextInt();
		}
	}
	abstract boolean find(int n);
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
class Find  extends Search
{	public boolean find(int n)
	{	boolean flag=false;
		for(int i=0;i<arr.length;i++)
		{	if(arr[i]==n)
			{
			flag=true;
			break;
			}
		}
		return flag;
	}
}
class Find2  extends Search
{	public boolean find(int n)
	{	boolean flag=false;
		for(int v:arr)
		{	if(v==n)
			{
			flag=true;
			break;
			}
		}
		return flag;
	}
}
class Usage
{	public static void main(String args[])
	{	Find f=new Find();
		f.read();
		out.println("Eneter element to found");
		int n=new Scanner(in).nextInt();
		if(f.find(n))
		out.println("Element found...");
		else
		out.println("Element Not found...");
	}
}
```

AbstractDemo1.java

```
import static java.lang.System.*;
abstract class Shape
{	abstract double area();
}
class Triangle extends Shape
{	double base,height;
	Triangle(double base,double height)
	{this.base=base;
	this.height=height;
	}
	public double area()
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {       double area=(0.5)*(base*height);
                return area;
        }
}
class AbstractDemo1
{       public static void main(String args[])
        {       Shape s;
                Triangle triangle=new Triangle(10,20);
                //upcasting
                s=triangle;
                double a=triangle.area();
                out.println("Area:\t"+triangle.base);
                out.println("Area:\t"+triangle.height);
                out.println("Area:\t"+triangle.area());
        }
}
```

AbstractDemo3.java

```
import static java.lang.System.*;
abstract class Shape
{       abstract double area();
}
class Circle extends Shape
{       final double pi=3.14;
        double radius;
        Circle(double radius)
        {this.radius=radius;
        }
        public double area()
        {double carea=pi*(radius*radius);
        return carea;
        }
}
class Rectangle extends Shape
{       double length,breadth;
        Rectangle(double length,double breadth)
        {       this.length=length;
                this.breadth=breadth;
        }
        public double area()
        {       return length*breadth;
        }
}
class AbstractDemo3
{       public static void main(String args[])
        {       Circle c=new Circle(2.00);
                double carea=c.area();
                out.println("Area of the circle:\t"+carea);

                Rectangle r=new Rectangle(4.00,2.00);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                double rarea=r.area();
                out.println("Area of the rectangle:\t"+rarea);a
        }
}
```

Why we write an abstract class?

- If a class has different implementations in different situations, then we have to write an abstract class.
- Best example for an abstract class is **GenericServlet** class.

AbstractDemo2.java

```
import static java.lang.System.*;
abstract class Kite
{       void design()
        {out.println("Design is implemented...in square shape");
        }
        abstract void stickTail();
        abstract void threadTied();
}
class MyKite extends Kite
{       void stickTail()
        {out.println("Tail is attested...");
        }
        void threadTied()
        {out.println("Thread is tied ...");
        }
        void kiteFlying()
        {out.println("Kite is flying...");
        }
}
class AbstractDemo2
{       public static void main(String args[])
        {       MyKite m=new MyKite();
                m.kiteFlying();
        }
}
```

AbstractDemo3.java

```
import static java.lang.System.*;
abstract class Biryani
{        String rice;
        String masala;
        String spices;
        String other;
        Biryani(String rice,String masala,String spices,String other)
        {       this.rice=rice;
                this.masala=masala;
                this.spices=spices;
                this.other=other;
        }
        void doGingerGarlicPaste()
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {out.println("GingerGarlic Paste done...");
        }
        abstract void cookBiryani();
}
class DumBiryani extends Biryani
{       DumBiryani(String rice,String masala,String spices,String other)
        {super(rice,masala,spices,other);
        }
        void cookBiryani()
        {       out.println(other+" Dum Biryani is cooked with:");
                out.println(rice+" Rice, mixed with  "+spices+" and "+masala);
        }
}
class AbstractDemo3
{       public static void main(String args[])
        {
        DumBiryani db=new DumBiryani("Basmathi","MTS masala"," spices ","Chicken");
        db.cookBiryani();
        }
}
```

Why we write an abstract method?

If a method has different implementations in different situations then we write an abstract method.

What is a concreate class?

It is a class which contains all implemented methods, and we can create object for it.

What is a concreate method?

It is a method which contains method body.

Interface

- Interface is a pure abstract class, which contains only abstract methods and final fields.
- We can't create objects for interfaces, but we can declare a reference variable of interface type.
- All the fields of an interface are by default declared as public, static and final.
- All the methods of an interface are, by default declared as public and abstract.
- We define an interface by using a keyword called "interface"
- One interface can extend another interface
- One class can inherit (implement) any no. of interfaces at a time.

Note: if we declare a variable in an interface, you have to initialize at the time of declaration only. But if you declare an instance variable of a class as final either we can assign it at the time of declaration or by using constructor if it is not initialized at the time of declaration.

InterfaceDemo1.java

```
import static java.lang.System.*;
interface BankInf
{       double minBal=500.00;
        void openAccount();
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        void deposit();
        void withdraw();
}
class CityBank implements BankInf
{       public void openAccount()
        {out.println("Account is created..");
        }
        public void deposit()
        {out.println("Deposited");
        }
        public void withdraw()
        {out.println("Withdraw");
        }
}
class InterfaceDemo1
{       public static void main(String args[])
        {       BankInf bank=new CityBank();
                out.println("minBal:\t"+BankInf.minBal);
                bank.openAccount();
                bank.deposit();
                bank.withdraw();
        }
}
```

InterfaceDemo.java

```
import static java.lang.System.*;
interface Shape
{       double pi=3.14;
        void draw();
        double getArea();
}
class Rectangle implements  Shape
{       double length,breadth;
        Rectangle(double length,double breadth)
        {this.length=length;
        this.breadth=breadth;
        }
        @Override
        public void draw()
        {out.println("Rectangle is drawn.. with length:\t"+length+" and breadth: "+breadth);
        }
        @Override
        public double getArea()
        {return length*breadth;
        }
}
class Circle implements  Shape
{       double radius;
        Circle(double  radius)
        {this.radius=radius;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        }
        public void draw()
        {out.println("Circle is drawn.. with radius:\t"+radius);
        }
        public double getArea()
        {return pi*(radius*radius);
        }
}
class InterfaceDemo2
{       static public void main(String yendukooooo[])
        {       Rectangle r=new Rectangle(10,6);
                r.draw();
                out.println("Area of rectangle:\t"+r.getArea());
                Circle c=new Circle(6);
                c.draw();
                out.println("Area of Circle:\t"+c.getArea());
        }
}
```

InterfaceDemo2.java(Same example with up casting)

```
import static java.lang.System.*;
interface Shape
{       double pi=3.14;
        void draw();
        double getArea();
}
class Rectangle implements  Shape
{       double length,breadth;
        Rectangle(double length,double breadth)
        {this.length=length;
        this.breadth=breadth;
        }
        public void draw()
        {out.println("Rectangle is drawn.. with length:\t"+length+" and breadth: "+breadth);
        }
        public double getArea()
        {return length*breadth;
        }
}
class Circle implements  Shape
{       double radius;
        Circle(double  radius)
        {this.radius=radius;
        }
        public void draw()
        {out.println("Circle is drawn.. with radius:\t"+radius);
        }
        public double getArea()
        {return pi*(radius*radius);
        }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
}
class InterfaceDemo
{       static public void main(String yendukooooo[])
        {       Shape s=new Rectangle(10,5);
                s.draw();
                out.println("Area:\t"+s.getArea());

                s=new Circle(2);
                s.draw();
                out.println("Area:\t"+(s.getArea()));
        }
}
```

Why we write an interface?

- We write an interface to implement data abstraction(security)
- I implement an interface and bind in server, but to inform the client we can provide an interface to him, by using that interface he can call the method existed (bind) in the server.
- To provide security
- The team leader allots interfaces to the team members to implement it.
- To implement multiple inheritance
- We write an interface to bind abstract methods and final fields

Final keyword:

Final is a keyword, which is used declare final fields, final methods and final classes.

Final fields: if we declare a field as final, then we can't modify it throughout the program execution. So final fields is nothing but constants

Final methods: if we declare a method as final, then we can't override it in the child classes.

Final classes: final classes are not inheritable

FinalDemo.java (not to run)

```
final class Base
{       final float pi=3.14f;
        final void display()
        {pi=3.24f;
        }
}
class Child extends Base
{       void display()
        {
        }
}
```

Compile:

```
E:\ThunderBolts>javac FinalDemo.java
FinalDemo.java:8: error: cannot inherit from final Base
class Child extends Base
                    ^
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
FinalDemo.java:5: error: cannot assign a value to final variable pi
        {pi=3.24f;
         ^
FinalDemo.java:10: error: display() in Child cannot override display() in Base
        void display()
             ^
  overridden method is final
3 errors
```

What is multiple inheritances?

If a single child inherits more than one base class then it is called as multiple inheritance. But java doesn't support multiple inheritances directly; we can achieve it by using interfaces. That is in java a single child class can inherit (implements) any no. of interfaces at a time.

MultipleInheritance.java

```
import static java.lang.System.*;
interface Inf1
{int add(int a,int b);
}
interface Inf2 extends Inf1
{int sub(int a,int b);
}
interface Inf3
{int mul(int a,int b);
}
class Base
{       int div(int a,int b)
        {return a/b;
        }
}
class Child extends Base implements Inf2,Inf3
{       public int add(int a,int b)
        {return a+b;
        }
        public int sub(int a,int b)
        {return a-b;
        }
        public int mul(int a,int b)
        {return a*b;
        }
}
class MultipleInher
{       public static void main(String args[])
        {       Child c=new Child();
                out.println(c.add(20,10));
                out.println(c.sub(20,10));
                out.println(c.mul(20,10));
                out.println(c.div(20,10));
                //upcasting(widening reference conversion)
                Inf1 i1=c;
                Inf2 i2=c;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                Inf3 i3=c;
                Base b=c;
                //Down casting(narrowing reference conversion)
                Child c2=(Child)i1;
                c2=(Child)i2;
                c2=(Child)i3;
                c2=(Child)b;
        }
}
```

Static & Dynamic Binding

Static Binding

- Static binding is also called as compile time binding or compile time polymorphism
- At the time of compilation compiler decides which members of a class has to execute at runtime.
- All the members of a class except instance methods support compilation time binding.
- Static binding depends on the reference variable type.

Dynamic binding

- Dynamic binding is also called as, runtime binding or runtime polymorphism
- At the time of program execution JVM decides, which members of a class has to execute.
- Only instance methods support dynamic binding.
- Dynamic binding depends on the object type.

```
import static java.lang.System.*;
class Base
{       int a=100;
        static int s=200;
        void display()
        {out.println("Base instance method");
        }
        static void get()
        {out.println("Base static method");
        }
}
class Child extends Base
{       int a=1000;
        static int s=2000;
        void display()
        {out.println("Child instance method");
        }
        static void get()
        {out.println("Child static method");
        }
}
class StatDynaBinding
{       public static void main(String args[])
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {       Base b=new Child();
                out.println(b.a);
                out.println(b.s);
                b.get();
                b.display();
                Child c=(Child)b;
                out.println(c.a);
                out.println(c.s);
                c.get();
                c.display();
        }
}
```

Object class and its methods

Object class: It is a predefined class existed in the java.lang package. It is the base class for every class in java, which means every class in java (it may be user defined/predefined) will inherit the members of Object class.

It has the following methods

clone()	Creates and returns a copy of the present object
equals()	Compares two objects
finalize()	This method is called by the garbage collector, before object get destroyed.
getClass()	It returns the object of a class called Class. Which contains the information about a particular class on which it is invoked.
hashCode()	Returns the hash code value of the object
toString()	Returns a string representation of the object.
wait()	It makes the current thread to wait until another thread invokes the notify() method or the notifyAll().
wait(long timeout)	It makesthe current thread to wait() until another thread invokes the notify() method or the notifyAll() or a specified amount of time.
wait(long timeout,int nanos)	It makes the current thread to wait() until another thread invokes the notify() method or notifyAll() or some other thread interrupts the current thread or a specified amount of real time.
notify()	Used to convert a thread from suspended state to runnable state
notifyAll()	Used to convert all the thread from suspended state to runnable state

ObjectClassDemo1.java

```
import static java.lang.System.*;
class One
{}
class ObjectClassDemo1
{       public static void main(String args[])
        {       One o1=new One();
                if(o1 instanceof Object)
                out.println("Yes o1 is instanceof Object");
                else
```

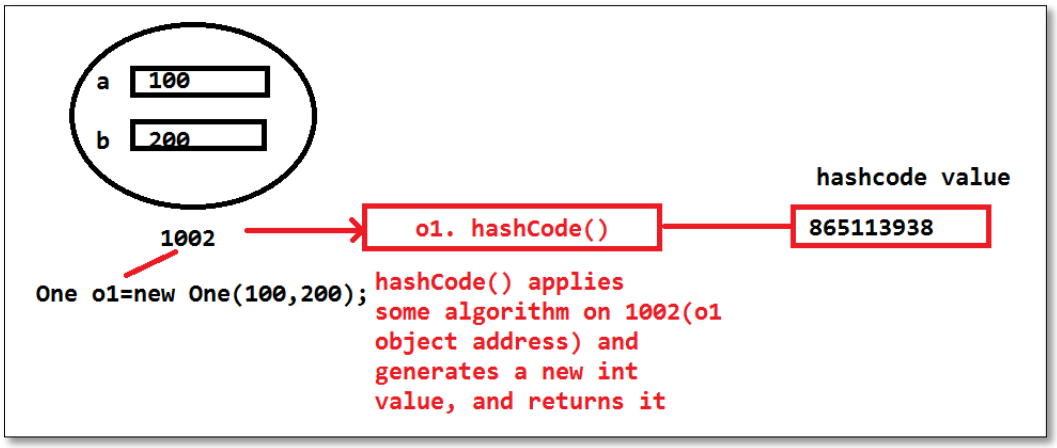
```
                out.println("No o1 is not instance of Object");
                if(o1 instanceof One)
                out.println("Yes o1 is instanceof One");
                else
                out.println("No o1 is not instance of One");
        }
}
```

ObjectClassDemo2.java

```
import static java.lang.System.*;
class One
{       int a,b;
        One(int a,int b)
        {       this.a=a;
                this.b=b;
        }
        void display()
        {       out.println("Object state");
                out.println("a:\t"+a);
                out.println("b:\t"+b);
        }
}
class ObjectClassDemo2
{       public static void main(String args[])
        {       Object []arr=new Object[4];
                arr[0]=new One(10,20);
                arr[1]=new One(100,200);
                arr[2]=new String("Poojitha");
                arr[3]=new Integer(100);
                out.println("Object states.....");
                for(Object o:arr)
                {       if(o instanceof One)
                        {       One obj=(One)o;
                                obj.display();
                        }else
                        out.println(o);
                }
        }
}
```

hashCode(): hashCode() method of Object class returns the hash code value(int value) which is generated by applying some algorithm on the address of the present object.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.



Why we use hashCode value?

We use hashCode value to compare two objects

HashCodeDemo.java

```
import static java.lang.System.*;
class One
{	int a,b;
	One(int a,int b)
	{	this.a=a;
		this.b=b;
	}
	void display()
	{	out.println("Object state");
		out.println("a:\t"+a);
		out.println("b:\t"+b);
	}
}
class HashCode
{	public static void main(String args[])
	{	One o1=new One(100,200);
		o1.display();
		int hcv=o1.hashCode();
		out.println("hashCodeValue:\t"+hcv);
	}
}
```

Why we override the hashCode() method?

- We override the hashCode() To generate hashCode value based on the object state.
- If hashCode() method generates the hashCode value based on the object state, by using that value we compare objects based on the state not based on the address.

HashCode.java

```
import static java.lang.System.*;
class One
{	int a,b;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
	One(int a,int b)
	{	this.a=a;
		this.b=b;
	}
	void display()
	{	out.println("Object state");
		out.println("a:\t"+a);
		out.println("b:\t"+b);
	}
	@Override
	public int hashCode()
	{	int hash = 5;
		hash = 97 * hash + this.a;//585
		hash = 97 * hash + this.b;//56945
		return hash;
	}
}
class HashCode
{	public static void main(String args[])
	{
		One o1=new One(100,200);
		One o2=new One(100,200);
		if(o1.hashCode()==o2.hashCode())
		out.println("Same...Same...Both are same");
		else
		out.println("Not Same....");
	}
}
```

public boolean equals(Object o)

It is existed in the java.lang.Object class, which is used to compare two object based on their addresses.

ObjectDemo.java

```
import static java.lang.System.*;
class One
{	int a,b;
	One(int a,int b)
	{this.a=a;
	this.b=b;
	}
	void display()
	{out.println("Object state..");
	out.println("a:\t"+a);
	out.println("b:\t"+b);
	}
}
class ObjectDemo
{	 public static void main(String args[])
	{	 One o1=new One(10,20);
		One o2=new One(100,200);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                if(o1.equals(o2))
                out.println("addresses are same");
                else
                out.println("addresses are differnt");
        }
}
```

Example on overriding equals method

EqualsDemo.java

```
import static java.lang.System.*;
class One
{       int a,b;
        One(int a,int b)
        {this.a=a;
        this.b=b;
        }
        void display()
        {out.println("Object state..");
        out.println("a:\t"+a);
        out.println("b:\t"+b);
        }
        //Here we have overriden the equals method of Object to compare the objects based on the content.
        public boolean equals(Object o)
        {       One o2=(One)o;
                if(this.a==o2.a && this.b==o2.b)
                return true;
                else
                return false;
        }
}
class EqualsDemo
{
        public static void main(String args[])
        {
                One o1=new One(10,20);
                One o2=new One(10,20);
                if(o1.equals(o2))
                out.println("objects are same");
                else
                out.println("Objects state is differnt");
        }
}
```

toString(): toString() method of Object class returns the hashCode value in the form of hexadecimal.

Why we override toString() method?

To see the object state

ToStringDemo.java

```
import static java.lang.System.*;
class One
{	int a,b;
	One(int a,int b)
	{this.a=a;
	this.b=b;
	}
	@Override
	public String toString()
	{return "One@[a="+a+",b="+b+"]";
	}
}
class ToStringDemo
{	public static void main(String args[])
	{	One o1=new One(100,200);
		out.println(o1);
		/*int hcv=o1.hashCode();
		String hcvInHexa=Integer.toHexString(hcv);
		out.println(hcvInHexa);
		*/
		out.println(o1.toString());
	}
}
```

ObjectMethods.java

```
import static java.lang.System.*;
/*
public class java.lang.Object
{	public native int hashCode()
	{	//it returns hasCode value
		//it returns the int value which is generated by applying some algorithm on the present
object address
	}
	public boolean equals(Object o)
	{//it compares objects based on the address
	}
	public String toString()
	{return "ClassName@"+Integer.toHexString(hashCode());
	//it returns the hashcode value in hexa decimal format as a string. preceded by class name and
@ symbol
	}
}
*/
class One
{	int a,b;
	One(int a,int b)
	{this.a=a;
	this.b=b;
	}
	@Override
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
	public int hashCode()
	{	int hash=5;
		hash=197 * hash + this.a;//585
		hash=197 * hash + this.b;//56945
		return hash;
	}
	@Override
	public boolean equals(Object o)
	{	if(o==null)
		return false;
		else if(this==o)
		return true;
		One o2=(One)o;
		if(this.a==o2.a && this.b==o2.b)
		return true;
		else
		return false;
	}
	@Override
	public String toString()
	{return getClass().getName()+"@[a="+a+",b="+b+"]";
	}
}
class EqualsDemo
{	public static void main(String args[])
	{	One o1=new One(100,200);
		One o2=new One(100,200);
		if(o1.equals(o1))
		out.println("Same");
		else
		out.println("Different");
		out.println(o1.toString());
		out.println(o1);

		String str=new String("Madhu");
		out.println(str);

		Integer io=new Integer(200);
		out.println(io);

	}
}
//Base b=new Child();
//Child c=(Child)b;
//Object o=o2;
//One o2=(One)o;
```

getClass():

- It is a predefined method existed in the Object class.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- It returns the java.lang.Class class object.

GetClassDemo.java

```
import static java.lang.System.*;
class One
{	int a,b;
	One()
	{out.println("Default constructor....");
	a=10;
	b=20;
	}
	One(int a,int b)
	{this.a=a;
	this.b=b;
	}
	@Override
	public String toString()
	{return "One@[a="+a+",b="+b+"]";
	}
}
class GetClassDemo
{	public static void main(String args[])throws Exception
	{	One o1=new One(100,200);
		out.println(o1);
		//c is the representative for One class
		Class c=o1.getClass();
		out.println(c.getName());
		String str="Madhu";
		//sc is the representative for String class
		Class sc=str.getClass();
		out.println(sc.getName());
		out.println(str.getClass().getName());
		Class oc=One.class;
		Class ic=Integer.class;
		out.println(oc.getName());
		out.println(ic.getName());
		Class oc1=Class.forName("One");
		Class ic1=Class.forName("java.lang.Integer");
		out.println("oc1:\t"+oc1.getName());
		out.println("oc1:\t"+ic1.getName());
	}
}
```

When we declare method return type as Object?

If we write a method which can return any object, which is decided dynamically, then we have to declare its return type as Object.

ObjectReturnType.java

```
import static java.lang.System.*;
class Demo
{	static Object get1(int opt)
	{	if(opt==1)
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                return "Hai";
                else if(opt==2)
                return new One(10,20);
                else if(opt==3)
                return new Integer(100);
                else
                return null;
        }
        static Object get2(int opt)
        {       Object o=null;
                if(opt==1)
                {o="Hai";
                }
                else if(opt==2)
                {o=new One(10,20);
                }
                else if(opt==3)
                {o=new Integer(100);
                }
                return o;
        }
}
class UpDownCast
{       public static void main(String args[])
        {       /*Object o=Demo.get1(1);
                String str=(String)o;
                */
                String str=(String)Demo.get1(1);
                out.println("str:\t"+str);
                One o=(One)Demo.get2(2);
                out.println("One:\t"+o);
        }
}
```

Output:

```
E:\ThunderBolts>java UpDownCast
Str:    Hai
One:    One@[a=10,b=20]
```

Clone() method: it is used to create copy for the already existed object

```
import static java.lang.System.*;
class One implements Cloneable
{       int a,b;
        One()
        {out.println("Default constructor....");
        a=10;
        b=20;
        }
        One(int a,int b)
        {this.a=a;
        this.b=b;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
	}
	@Override
	public String toString()
	{return "One@[a="+a+",b="+b+"]";
	}
	public Object clone()throws CloneNotSupportedException
	{	Object o=super.clone();
		return o;
	}
}
class CloneDemo
{	public static void main(String args[])throws Exception
	{	One o1=new One(100,200);
		One o2=(One)o1.clone();
		out.println(o1);
		out.println(o2);
	}
}
```

Finalize() : it is invoked by the garbage collector at the time of object is destroyed from heap.

FinalizeDemo.java

```
import static java.lang.System.*;
class One
{	int a,b;
	One(int a,int b)
	{this.a=a;
	this.b=b;
	}
	@Override
	public String toString()
	{return getClass().getName()+"@[a="+a+",b="+b+"]";
	}
	@Override
	public void finalize()
	{out.println("Finalized method is invoked.....");
	}
}
class FinalizeDemo
{	public static void main(String args[])
	{	for(int i=1;i<=1;i++)
		{	One o=new One(100,200);
			out.println(o);
		}
		System.gc();
		out.println("end of main");
	}
}
```

Packages

What is Package?

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

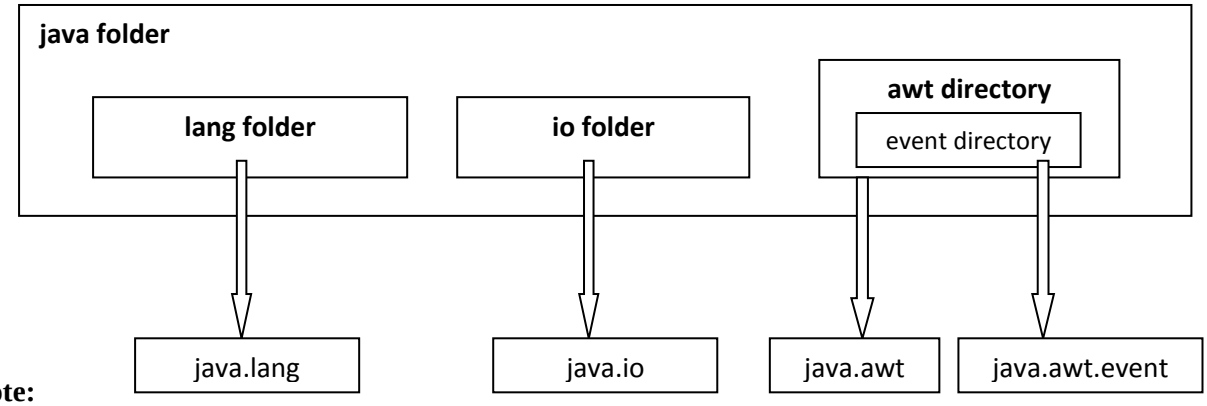
Definitions1: Packages are containers for classes or we can say package is nothing but collection of classes, interfaces and enums. The package is a both name and visibility control mechanism.

Definitions2: a package represents a directory that contains related group of classes, interfaces and enums. For example, when we write statements like:

```
import java. lang.*;
```

Here we are importing all the classes of **java.lang** package. Here java is a folder name and lang is the subfolder name of java

View of a package



Note:

If you want to use the classes of awt folder in your program you have to import java.awt package. If you import java.awt package you can't use the classes of event folder to use the classes of event folder you must import the java.awt.event package.

Advantages of packages:

- Package is a both name and visibility control mechanism.
- We can define classes inside a package that are not accessible by code outside that package.
- Packages provide reusability.
- Packages are useful to arrange related classes and interfaces into a group.

We can create a package in two ways

- **First way of creation of package**

Step-1

Create a folder with the package name. For Example if you want to create a package with the name "p1", first create a folder with name "p1".

```
E:\NUBatch>md p1
E:\NUBatch>cd p1
```

Write java program with the package statement, and save the file in p1 folder. See the below example.

Note: package statement must be the first statement in a java program.

Example: E:\NUBatch\p1>notepad First.java

```
package p1;
public class First
{       public int add(int a,int b)
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {return a+b;
        }
        public int sub(int a,int b)
        {return a-b;
        }
        public int mul(int a,int b)
        {return a*b;
        }
}
```

6. compile the java program

```
E:\NUBatch\p1>javac First.java
```

When we compile the program **First.class** will be created in the **"p1"** folder, now we can say we have created a package **"p1"** and it contains a class **"First"**.

Usage of the package

If we set the classpath for "p1"package then we can use the **"p1"** package in the programs written anywhere within the computer.

If we don't set the class path then we can use the "p1"package only in the java programs written in the root/parent folder of "p1".

Usage example:

- Write a java program in "NUBatch" (root) folder

```
E:\NUBatch>notepad Usage.java
import p1.First;
import static java.lang.System.*;
class Usage
{       public static void main(String args[])
        {       First f=new First();
                int sum=f.add(10,20);
                int sub=f.sub(110,20);
                int mul=f.mul(10,2);
                out.println("Sum:\t"+sum);
                out.println("Sub:\t"+sub);
                out.println("Mul:\t"+mul);
        }
}
Output:
E:\NUBatch>javac Usage.java
E:\NUBatch>java Usage
Sum:    30
Sub:    90
Mul:    20
```

What is CLASSPATH?

Class path is an environment variable that tells the java compiler where to look for class files to import.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Note: Class path is generally set to a directory (folder) or a JAR (Java Archive) file, which contains .class files or packages.

How to see the already existed classpath?

```
C:\> echo %classpath%
```

We can set the class path in two ways

- **Temporary**
- **Perminant**

Tempory classpath:

```
d:\>set classpath=%classpath%;E:\NUBatch;
```

In the above command, we are setting the classpath for "**E:\NUBatch;"** folder, and also to the old classpath.

Now write a program in any folder of the computer

D:\>notepad Use.java

Use.java

```
import p1.First;
import static java.lang.System.*;
class Use
{       public static void main(String args[])
        {       First f=new First();
                out.println(f.add(10,20));
                out.println(f.mul(10,20));
                out.println(f.sub(10,2));
        }
}
```

Compile & run

```
D:\>javac Use.java
D:\>java Use
30
200
8
```

Permanent classpath:

MyComputer→ properties → AdvancedSystemSettings→ Environment Variables→New→

Variable Name: classpath

Variable Value E:\NUBatch;

→Ok → Ok→Ok

Then classpath for **"E:\NUBatch"** folder will be set.

After setting the classpath we can check like below

```
F:\>javap p1.First
Compiled from "First.java"
public class p1.First {
  public p1.First();
  public int add(int, int);
  public int sub(int, int);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
  public int mul(int, int);
}
```

When you type the above command on the command prompt, we will get information about the mentioned class (First), if you get this information, which means class path is set perfectly. Otherwise will get the below error.

Error: class not found: p1.First

Second way of creation of package (By the compiler):

- Write a java program with the package statement in the parent folder directly.

```
E:\NUBatch>notepad Second.java
package p2;
import static java.lang.System.*;
public class Second
{       public void display()
        {out.println("p2.Second.display");
        }
}
```

- Compile the java program (Second.java) slight differently than before see below

Example: E:\NUBatch>javac -d . Second.java

1. Here **–d** is an option given to the compiler, which tells the java compiler to creates a folder with the package name and to place the .class files into that folder.

2. As a second option, we can give the path where you want to create the package. Here **"."** represents the present path (folder), when you give the **"."** The compiler create a package in present folder called **"NUBatch".**

Note: In the place of "." We can give any path where you want to create the package.

- We can use this package in the programs written anywhere within the system(if we set the classpath) otherwise we can use in the "NUBatch" folder only.

Example: e:\NUBatch \>notepad Use.java

```
import static java.lang.System.*;
import p2.Second;
class Use
{       public static void main(String args[])
        {Second s=new Second();
        s.display();
        }
}
```

Output:

```
p2.Second.display
```

Creation of SubPackages

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Create a sub package (i.e. a package inside a package) in the below example "org" is the sub package.

```
package p1.org;
import static java.lang.System.*;
public class HelloWorld
{       public void hello()
        {out.println("Hello...hai....vonakkam.... addab...");
        }
}
```

Use the p1.org subpackage in User.java program

Use.java

```
import static java.lang.System.*;
import p1.org.HelloWorld;
class Use
{       public static void main(String args[])
        {       HelloWorld hw=new HelloWorld();
                hw.hello();
        }
}
```

The Jar Files

A JAR (Java Archive) file is a file that contains compressed version of .class files, audio files, image files or directories. We can imagine a .jar file as a zip file (.zip); the difference is that a .jar file can be used as it is whereas we have to extract the .zip file to use.

Let us see how to create a .jar file

- **To create a jar file, Java Soft people have provided jar command, which can be used in the following way.**

Syntax: jar -cvf jarfilename inputfiles

In the above syntax **"c"** means create new archive files and **"v"** means display verbose output on standard output device and **"f"** specify archive file name

Ex: jar –cvf all.jar pack1*.* pack2*.*

- **To view the contents of jar file**

Ex: **jar tf all.jar**

Here in the above example **"t"** means table view of contents and **"f"** means specify file name

- **To extract the files from a .jar file, we can use:**

Ex: jar -xf all.jar

In the above example **"xf"** represents extract files from the jar file

Now we know how to create a jar file, let us see how it can be used. For example you think you have created all.jar file and stored in d:\temp folder. If you want to make all the classes and package available in all.jar file available anywhere in the computer system we have to set the CLASSPATH for all.jar file.

Steps to set the classpath permanently for a jar file

- My Computer→properties→advanced→click on environment variables button
- Go to user variables and click on New button
- Give variable name: CLASSPATH
- Give variable value: D:\temp\all.jar;

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- Then click on OK→OK→OK

WRAPPER CLASSES

Wrapper classes are used to convert objects to primitive types and primitive types to objects. We see many applications on Internet which receive data from the user and send it to the server. For example, in a business application, we type our details like empName, empNo, address, etc. and send them to the server. The server expects this data in the form of objects and hence we have to send objects. In our data 'name' is a String type object but empNo is just int type value, which is not an object. This primitive datatype should also be converted into an object and then sent to the server. To do this conversion, we need wrapper classes.

Why do we need Wrapper classes?

- In Java, collection classes are defined in **java.util. package** which handle only objects, not the primitive data types. Hence, if we want to use collection classes on primitives, we should convert them into objects. Here, also we need the help of wrapper classes.
- Wrapper classes are used to convert one type to another type very easily like
  1. Primitive type to objects
  2. Objects to primitive types

This is very useful in web applications and applications on network since there we send or receive data only as objects. The below table contains the list of **wrapper classes** that are defined in **java.lang** package. They are useful to convert the primitive data types into object form.

Primitive data type	Wrapper classes
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Number Class:

Number is an abstract class and it is the Base class for the following 6 wrapper classes

1. Byte
2. Short
3. Integer
4. Long
5. Float
6. Double

Number class methods

Method	explanation
byte byteValue()	It converts an object into byte value. The object may be of Byte, Short, Integer, Long, Float, or Double class.
short shortValue()	It converts an object into short value, and then returns it to us.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

int intValue()	It converts an object into int value, and then returns it to us.
long longValue()	It converts an object into long value, and then returns it to us.
float floatValue()	It converts an object into float value, and then returns it to us.
double doubleValue()	It converts an object into double value, and then returns it to us.

Character Class

The Character is a class, which contains a char type field. It has only one constructor which takes char value as an argument.

CharacterDemo.java

```
import static java.lang.System.*;
import java.io.*;
class CharacterDemo
{	public  static void main(String args[])throws Exception
	{	char ch;
		BufferedReader br=new BufferedReader(new InputStreamReader(in));
		out.println("Enter a char");
		ch=(char)br.read();
		if(Character.isDigit(ch))
		out.println("It is Digit");
		else if(Character.isUpperCase(ch))
		out.println("It is Uppercase");
		else if(Character.isLowerCase(ch))
		out.println("It is Lowercase");
		else if(Character.isSpaceChar(ch))
		out.println("It is Space");
		else if(Character.isWhitespace(ch))
		out.println("It is White Space");
		Character ch1=new Character('a');
		Character ch2=new Character('A');
		int r=ch1.compareTo(ch2);
		if(r==0)
		out.println(ch1+"="+ch2);
		else if(r>0)
		out.println(ch1+">"+ch2);
		if(r<0)
		out.println(ch1+"<"+ch2);
		out.println(ch1.toString());
		Character ch3=Character.valueOf('m');
		out.println(ch3);
		if(Character.isLetterOrDigit(ch3))
		out.println("It is letter or digit");
	}
}
```

ByteDemo.java

```
import static java.lang.System.*;
import java.util.*;
class ByteDemo
{	public static void main(String args[])
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {       Scanner s=new Scanner(in);
                out.println("Enter two integers");
                byte v1=s.nextByte();
                String v2=s.next();
                Byte b1=new Byte(v1);
                Byte b2=new Byte(v2);
                int r=b1.compareTo(b2);
                if(r==0)
                out.println("Both objects are same");
                if(r>0)
                out.println("b1 is greater than b2");
                if(r<0)
                out.println("b1 is less than b2");
        }
}
```

ShortDemo.java

```
import static java.lang.System.*;
import java.util.*;
class ShortDemo
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("Enter two values");
                String v1=s.next();
                String v2=s.next();
                Short s1=new Short(v1);
                Short s2=new Short(v2);
                int r=s1.compareTo(s2);
                if(r==0)
                out.println("Both objects are same");
                if(r>0)
                out.println("s1 is greater than s2");
                if(r<0)
                out.println("s1 is less than s2");
        }
}
```

IntegerDemo.java

```
import static java.lang.System.*;
import java.util.*;
class IntegerDemo
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("Enter an int value");
                String v=s.next();
                int iv=Integer.parseInt(v);
                out.println("Decimal value:\t"+iv);
                String bv=Integer.toBinaryString(iv);
                out.println("Binary value:\t"+bv);
                String str1=Integer.toHexString(iv);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("To Hexa String:\t"+str1);
                String str2=Integer.toOctalString(iv);
                out.println("To Octal String:\t"+str2);
        }
}
```

LongDemo.java

```
import static java.lang.System.*;
import java.util.*;
class LongDemo
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("Enter an int value");
                long lv=s.nextLong();
                Long l1=new Long(lv);
                Long l2=new Long("30000");
                if(l1.compareTo(l2)==0)
                out.println("Both are same");
                long ll1=Long.parseLong("123400000");
                out.println(ll1);
                long ll2=Long.valueOf("2345665");
                out.println(ll2);
        }
}
```

FloatDemo.java

```
import static java.lang.System.*;import java.util.*;
class FloatDemo
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("Enter an int value");
                String f=s.next();
                Float fo1=new Float(f);
                Float fo2=Float.valueOf(f);
                out.println(fo1);
                out.println(fo2);
                float fv=Float.parseFloat("2345.56f");
                out.println(fv);
        }
}
```

DoubleDemo.java

```
import static java.lang.System.*;
import java.util.*;
class DoubleDemo
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("Enter an int value");
                String str=s.next();
                Double do1=new Double(str);
                Double do2=Double.valueOf(str);
```

```
                out.println(do1);
                out.println(do2);
                double dv=Double.parseDouble("2345.56f");
                out.println(dv);
        }
}
```

BooleanDemo.java

```
import static java.lang.System.*;
import java.util.*;
class BooleanDemo
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                out.println("Enter an int value");
                String str=s.next();
                Boolean b1=new Boolean(str);
                Boolean b2=Boolean.valueOf(str);
                Boolean b=new Boolean(true);
                out.println(b1);
                out.println(b2);
                out.println(b);
                boolean b3=Boolean.parseBoolean("true");
                out.println(b3);
        }
}
```

WrapperClassesDemo.java

```
import static java.lang.System.*;
class WrapperClassesDemo
{       public static void main(String args[])
        {       Integer oi=new Integer(100);
                Long ol=new Long(200);
                Float of=new Float(300.34f);
                Double od=new Double(234.456);
                float f=oi.floatValue();
                double d=ol.doubleValue();
                int i=of.intValue();
                short s=od.shortValue();
                out.println("f:\t"+f);
                out.println("d:\t"+d);
                out.println("i:\t"+i);
                out.println("s:\t"+s);
        }
}
```

Exception Handling

Exception handling is a technology which is used to handle runtime errors. If we handle runtime errors, the program will not be terminated in middle of execution.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

What is an exception?

It is an object which is thrown by the JVM at the time of runtime error occurs. It contains run time error information like.

1. Cause of the exception(why it was occurred)
2. Type of the exception(what type of exception it is)
3. In which statement is it occurred(where it was occurred)

We need to know about the following 5 keywords, to use exception handling in our programs

- try:
- catch
- finally
- throw
- throws

Try block:

- It is used to place the statements, which may throw an exception.
- After try block we have to write either catch or finally block. If we implement try with resources concept, then we can write a try block without using catch or finally block.
- We can try any no. of catch blocks for a try block.

ExDemo1.java

```
import static java.lang.System.*;
import java.util.*;
class ExDemo1
{	public static void main(String args[])
	{	out.println("Start of main");
		Scanner s=new Scanner(in);
		int a,b,c=0;
		try{
		out.println("Enter a,b values...");
		a=s.nextInt();
		b=s.nextInt();
		c=a/b;
		}catch(ArithmeticException ae)
		{out.println(ae);
		}
		out.println("c:\t"+c);
		out.println("End of main");
	}
}
```

ExDemo2.java

```
import static java.lang.System.*;
import java.util.*;
class ExDemo2
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
{       public static void main(String args[])
        {       out.println("Start of main");
                Scanner s=new Scanner(in);
                int a,b,c=0;
                try{
                out.println("Enter a,b values...");
                a=s.nextInt();
                b=s.nextInt();
                c=a/b;
                }catch(ArithmeticException ae)
                {out.println(ae);
                }
                catch(InputMismatchException ime)
                {//out.println(ime);
                ime.printStackTrace();
                }
                out.println("c:\t"+c);
                out.println("End of main");
        }
}
```

Nested try: we write a try block with in a try block, it is called as nested try.

ExDemo3.java

```
import static java.lang.System.*;
import java.util.*;
class ExDemo3
{       public static void main(String args[])
        {       out.println("Start of main");
                Scanner s=new Scanner(in);
                int a,b,c=0;
                try{
                        try{
                        out.println("Enter a value");
                        a=s.nextInt();
                        }catch(InputMismatchException ime)
                        {       out.println("Error while reading a value...");
                                out.println("Enter a value again");
                                s.next();
                                a=s.nextInt();
                        }
                        out.println("Enter b value");
                        b=s.nextInt();
                        c=a/b;
                }catch(ArithmeticException ae)
                {out.println(ae);
                }
                catch(InputMismatchException ime)
                {ime.printStackTrace();
                }
                out.println("c:\t"+c);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("End of main");
        }
}
```

Finally block:

It is a block which is executed by the JVM every time compulsorily during program execution. In this block we write the statements like.

1. Stream closings
2. Db connection closings etc...

FinallyBlock.java

```
import static java.lang.System.*;
import java.io.*;
import java.util.*;
class FinallyDemo
{       public static void main(String args[])
        {       FileInputStream fis=null;
                Scanner s=new Scanner(in);
                try{    out.println("Enter file name to read");
                        String fileName=s.next();
                        fis=new FileInputStream(fileName);
                        int r=0;
                        while(  (r=fis.read())!=-1)
                        {out.print((char)r);
                        }
                        fis.close();
                }catch(FileNotFoundException fe)
                {//fe.printStackTrace();
                //out.println(fe);
                out.println(fe.getMessage());
                }
                catch(IOException ie)
                {out.println(ie);
                }finally{
                        out.println("Finally block is executed....");
                        try{
                                if(fis!=null)
                                {fis.close();
                                }
                        }catch(IOException ie)
                        {out.println("Error while closing connection..");
                        }
                }
        }
}
```

throw: it is used to throw an exception by the programmer.

ThrowDemo.java

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
import static java.lang.System.*;
import  java.util.*;
class ThrowDemo
{	public static void main(String args[])
	{	int a,b,c=0;
		Scanner s=new Scanner(in);
		try{
		out.println("Enter a,b values..");
		a=s.nextInt();
		b=s.nextInt();
		if(b==0)
		throw new ArithmeticException("B value zero.... vundakoodath annaa.a....");
		c=a/b;
		}catch(ArithmeticException ae)
		{out.println(ae);
		}
		out.println("c:\t"+c);
	}
}
```

Throws: it is used to throw an unreported exception out of a method.

ThrowsDemo.java

```
import static java.lang.System.*;
import  java.io.*;
class One
{	static String readFile(String fileName)throws FileNotFoundException,IOException
	{	String data="";
		FileInputStream fis=new FileInputStream(fileName);
		int r;
		while((r=fis.read())!=-1)
		{data=data+(char)r;
		}
		fis.close();
		return data;
	}
}
class ThrowsDemo
{	public static void main(String args[])
	{	try{
		String fileData=One.readFile("ExDemo1.java");
		out.println("Data of a file\n-------------------------");
		out.println(fileData);
		}catch(FileNotFoundException e)
		{e.printStackTrace();
		}
		catch(IOException ioe)
		{ioe.printStackTrace();
		}
	}
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

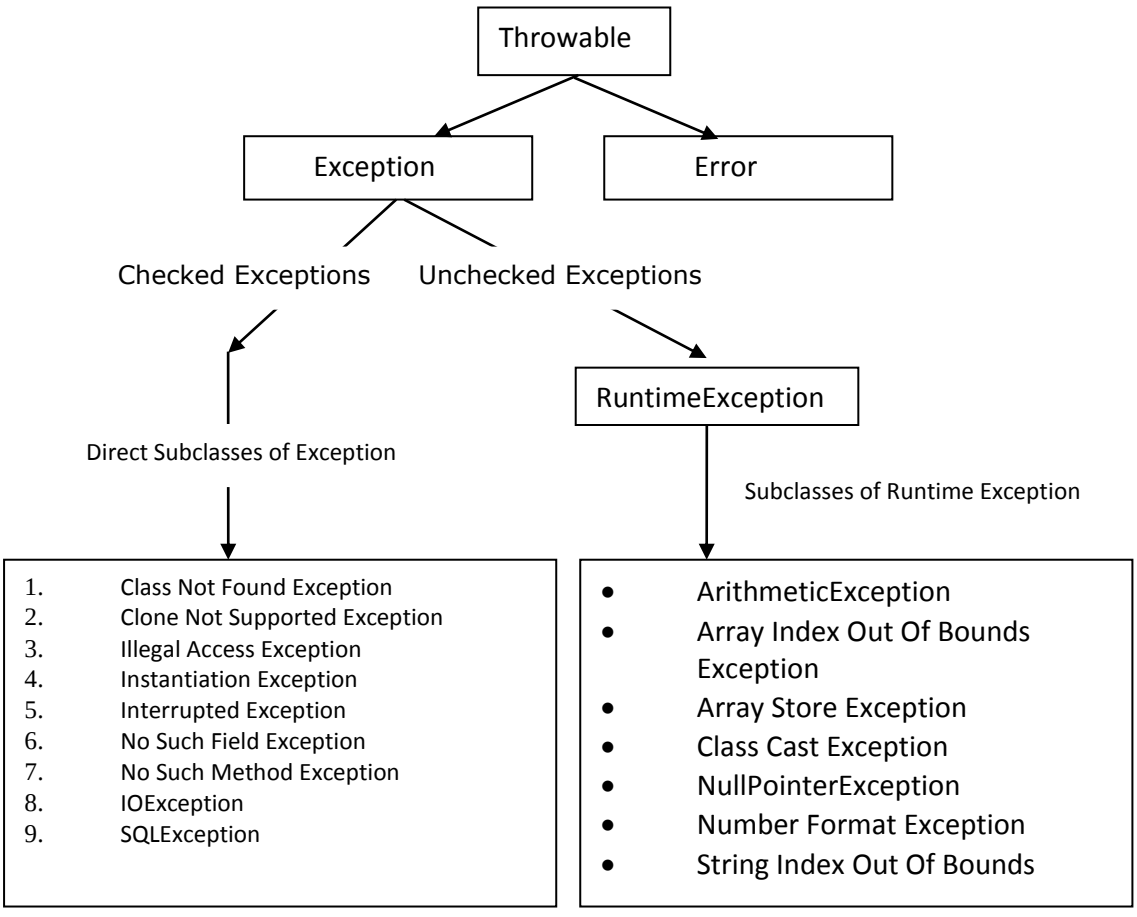
Types of Exceptions

There are two types of exceptions we have

- Reported exceptions(unchecked exceptions)
- Unreported exceptions(checked exceptions)

Unreported Exceptions

- All the unreported exceptions are direct sub classes of Exception class
- Unreported exceptions are recognized by the compiler at the time of compilation



Handling all exceptions by using single catch block

ExDemo3.java

```
import static java.lang.System.*;
import java.util.*;
class ExDemo3
{       public static void main(String args[])
        {       out.println("Start of main");
                Scanner s=new Scanner(in);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                int a,b,c=0;
                try{
                out.println("Enter a,b values...");
                a=s.nextInt();
                b=s.nextInt();
                c=a/b;
                }catch(Exception ae)
                {out.println(ae);
                }
                out.println("c:\t"+c);
                out.println("End of main");
        }
}
```

What is the difference between Exception and Error?

We can handle the Exceptions, but we can't handle the Errors, and both are child classes of Throwable class.

Userdefined Exceptions

- We can create our own exceptions either by extending Exception or RuntimeException class

UDException1.java

```
import static java.lang.System.*;
import java.util.*;
//Unreported exception
class URZeroException extends Exception
{       URZeroException(String cause)
        {super(cause);
        }
}
//reportedException
class RZeroException extends RuntimeException
{       RZeroException(String cause)
        {super(cause);
        }
}
class UDException1
{       public static void main(String args[])
        {       Scanner s=new Scanner(in);
                int a,b,c=0;
                try
                {       out.println("Enter a,b values...");
                        a=s.nextInt();
                        b=s.nextInt();
                        if(b==0)
                        {//throw new URZeroException("Denominator must not be zero");
                        throw new RZeroException("Denominator must not be zero");
                        }
                        c=a/b;
                }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                //catch(URZeroException e)
                catch(RZeroException e)
                {e.printStackTrace();
                }
                out.println("c:\t"+c);
        }
}
```

Multi-Catch Exceptions:

In Java SE 7 a single catch block can handle more than one type of exception. This concept is called as Multi-catch Exceptions.

Note: If a catch block handles more than one exception type, then the catch parameter is implicitly final. In this example, the catch parameter "**ex**" is final and therefore you cannot assign any values to it within the catch block.

MultiCatch.java

```
import java.io.*;
import static java.lang.System.*;
class MultiCatch
{               public static void main(String args[])
                {DataInputStream dis=new DataInputStream(System.in);
                 out.println("Start of main method....");
                 int c=0,a=0,b=0;
                 try
                 {out.println("enter a value");
                 a=Integer.parseInt(dis.readLine());
                 out.println("enter a value");
                 b=Integer.parseInt(dis.readLine());
                 c=a/b;
                 dis.close();
                 }catch(ArithmeticException | NumberFormatException | IOException ae)
                 {      out.println(ae);
                        out.println(ae.getMessage());
                 }
                 out.println("c=\t"+c);
                 out.println("End of main method....");
                }
}
```

Try-with-Resources: It is a new feature introduced in 1.7. It makes all the resources to be closed automatically, without closing explicitly.

Example: TryWithResource.java

```
import java.io.*;
import static java.lang.System.*;
class TryWithResource
{               public static void main(String args[])
                {       out.println("Start of main method....");
                        int c=0,a=0,b=0;
                        try(BufferedReader br=new BufferedReader(new InputStreamReader(in)))
```

```
                    {       out.println("enter a value");
                            a=Integer.parseInt(br.readLine());
                            out.println("enter a value");
                            b=Integer.parseInt(br.readLine());
                            c=a/b;
                    }catch(ArithmeticException | NumberFormatException | IOException ae)
                    {       out.println(ae);
                            out.println(ae.getMessage());
                    }
                    out.println("c=\t"+c);
                    out.println("End of main method....");
            }
}
```

Re-throwing an exception

When an exception occurs in a try block, it is caught by a catch block. This means that the thrown exception is available to that catch block only. If u you want to make that exception available to other class, we have to re throw it. The following code shows how to re-throw the same exception out from the catch block.

```
try{throw exception;
}catch(Exception e)
{throw exception;//re throwing an exception
}
```

Suppose there are two class One and Two. If an exception occurs in class One, we want to display some message to the user and then we want to re-throw it. This re-thrown exception can be caught in class B where it can be handled. Hence re-throwing exceptions is useful especially when the programmer wants to propagate the exception details to another class. In this case (below example), the exception details are sent from class One to class Two where some appropriate action may be performed.

RethrowDemo.java

```
import static java.lang.System.*;
class One
{       void display()
        {
                try{
                        //name variable contains a string with 17 chars
                        String name="Mathu Tech Skills";
                        /*exception is thrown in the below statement
                        because there is no char existed in the 17th index*/
                        for(int i=0;i<=name.length();i++)
                        {       char ch=name.charAt(i);
                                out.print(ch);
                        }
                }catch(StringIndexOutOfBoundsException so)
                {       out.println("\nplease see the index is it within the range");
                        throw so;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                }
        }
}class Two
{       public static void main(String[] args)
        {       One o=new One();
                try{o.display();
                }catch(StringIndexOutOfBoundsException so)
                {       out.println("rethrown exception is caught here");
                        out.println(so);
                }
        }
}
```

Output:

```
D:\mts>javac RethrowDemo.java
D:\mts>java Two
Mathu Tech Skills
please see the index is it within the range
rethrown exception is caught here
java.lang.StringIndexOutOfBoundsException: String index out of range: 17
```

Explanation for the above program:

In the above program, StringIndexOutOfBoundsException is thrown in method display () of class One which is caught by catch block in that method. Then the catch block is re-throwing it into main () method of class Two.

Rethrow.java

```
import java.util.*;
import static java.lang.System.*;
class One
{       Scanner s=new Scanner(System.in);
        int [] read()
        {       int a[]=new int[3];
                try{
                out.println("Enter 3 elements");
                for(int i=0;i<3;i++)
                a[i]=s.nextInt();
                }catch(InputMismatchException ie)
                {       out.println("Error in read:\n"+ie);
                        throw ie;
                }
                return a;
        }
}class ReThrow
{       public static void main(String args[])
        {       One o=new One();
                try{
                int a[]=o.read();
                }catch(InputMismatchException ie)
                {       out.println("Error in main:\nGiven Input is not correct");
                }
        }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
}
```

Output:

```
D:\mts>java ReThrow
Enter 3 elements
23s
Error in read:
java.util.InputMismatchException
Error in main:
Given Input is not correct
```

(Userdefined Exceptions)

Creating Your Own Exception Subclasses

Although Java's built-in exceptions handle most common errors, you will probably want to create your own exception types to handle situations specific to your applications. This is quite easy to do: just define a subclass of **Exception.** Here is an example that uses nested **UserDefinedException.**

ExDemo8.java

```
import static java.lang.System.*;
class MyException extends Exception
{       MyException(String cause)
        {super(cause);
        }
}class ExDemo8
{       public static void main(String args[])
        {       java.util.Scanner s=new java.util.Scanner(System.in);
                out.println("enter uname:");
                String uname=s.next();
                out.println("enter password:");
                String pwd=s.next();
                try
                {if(uname.equals("madhu") && pwd.equals("vij"))
                out.println("Welcome To Madhu Tech Skills   ....\n      VIJAYAWADA");
                else
                throw new MyException("Invalid User name and password");
                }
                catch(MyException me)
                {       me.printStackTrace();
                }
        }
}
```

What is the difference between throws and throw?

Throws clause is used when the programmer does not want to handle the exception and throw it, out of a method. Throw clause is used, when the programmer wants to throw an exception explicitly and wants to handle it using catch block.

Some important methods of the Throwable class are:

METHOD	DESCRIPTION
toString()	It returns Type of Exception and the cause as string;

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

getMessage()	It returns only the cause as string
printStackTrace()	It returns Type of excepion, cause of exception and Where it was occured.
getCause()	It returns the exception object that causes the current exception to occur.

Note:

The methods in Throwable class are inherited by every exception, so we can use these methods by using any exception object.

What is Chained Exception or Nesting exception?

Whenever in a program the first exception causes another exception to occur, It is termed as **Chained Exception**. Exception chaining is also known as "nesting exception". It is a technique for handling the exceptions, which are occured one after another.

ChainedEx.java

```
import static java.lang.System.*;
class DivideEx extends Exception
{	DivideEx(String cause,Throwable t)
	{super(cause,t);
	}
}
class One
{	 static int div()throws DivideEx
	{	java.util.Scanner s=new java.util.Scanner(in);
		int a=0,b=0,c=0;
		try{
		out.println("Enter a,b values..");
		a=s.nextInt();
		b=s.nextInt();
		c=a/b;
		}catch(java.util.InputMismatchException | ArithmeticException e)
		{throw new DivideEx("Input may be worng or denom may be zero",e);
		}
		return c;
	}
}
class ChainedExDemo
{	public static void main(String args[])
	{	 try{
		out.println(One.div());
		}catch(DivideEx e)
		{//out.println(e);
		//Throwable t=e.getCause();
		//out.println(t);
		e.printStackTrace();
		}
	}
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

MULTI THREADING

Introduction:

Unlike other languages Java provides built-in support for *multithreaded programming.* Multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a *thread*, and each thread defines a separate path of execution. Thus, multithreading is a specialized form of multitasking.

Thread: a Thread is a sub program in a multithreaded program (or) part of a multithreaded program and each part is used to execute a task.

Now-a-days almost every knows about multitasking because every modern operating system supports multitasking.

Types of Multitasking: There are two t types of multitaskings available those are:

- **Process-based**
- **Thread-based**

Process-Based:

A process is nothing but a program in execution. Thus, *process-based* multitasking is the feature that allows your computer to run two or more programs concurrently. For example, process-based multitasking enables you to install oracle software, at the same time that you are listening to songs on Windows Media Player.

Thread-Based:

In a *thread-based* multitasking environment, a single program can perform two or more tasks simultaneously. For instance, a text editor can format text at the same time that it is printing.

Note: Processes are heavyweight tasks that require their own separate address spaces. Context switching from one process to another is also costly. Threads, on the other hand, are lightweight. They share the same address space and cooperatively share the same heavyweight process, Interthread communication is inexpensive, and context switching from one thread to the next is low cost.

What is lightweight process?

A thread is considered to be a light weight process because it runs within the context of a program (within the process area) and utilizes the resources allocated to that program(process). And it utilizes minimum resources of the system.

What is heavyweight process?

Whenever a new process starts execution system allocates separate memory and resources for each process separately.

Important Points to Remember:

In heavy weight process, the control changes in between threads belonging to different programs (i.e. two different processes). But, in lightweight process, the control changes in between the threads belonging to the same (one) process.

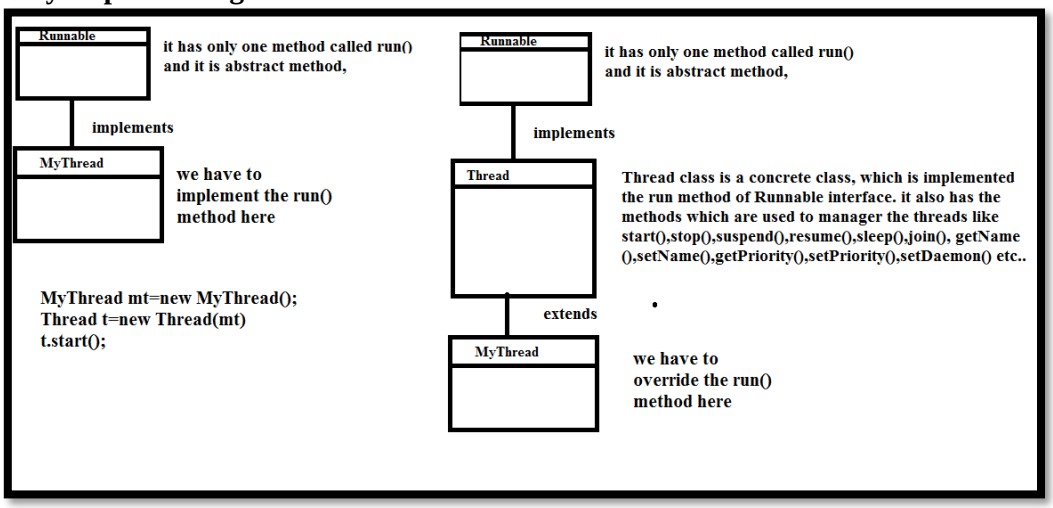
|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Use: Multithreading enables you to write very efficient programs that make maximum usage of the CPU time, because idle time can be kept to a minimum.

Creation of Threads

We can create threads in two ways

- **By extending a Thread class**
- **By Implementing a Runnable interface**



ThreadDemo1.java(First way of creation of thread)

```
import static java.lang.System.*;
class MyThread extends Thread
{       public void run()
        {       for(int i=1;i<=10;i++)
                {out.println(i);
                }
        }
}
class ThreadDemo1
{       public static void main(String args[])
        {        MyThread t1=new MyThread();//thread is in new born state
                t1.start();          //Runnable state
        }
}
```

ThreadDemo2.java

```
import static java.lang.System.*;
class MyThread implements Runnable
{       public void run()
        {       for(int i=1;i<=10;i++)
                {out.println(i);
                }
        }
}
class ThreadDemo2
```

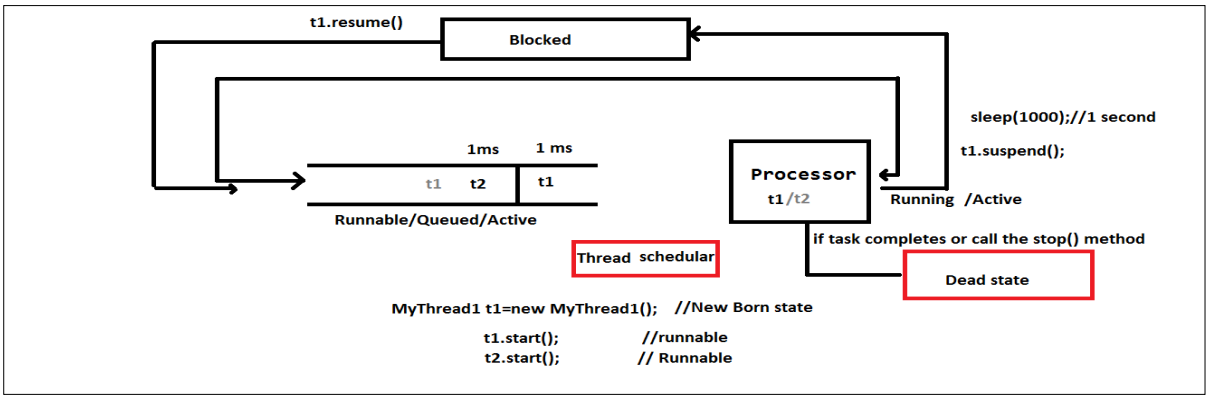

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
{       public static void main(String args[])
        {       MyThread runnable=new MyThread();//it is a Runnable object
                 //the run method of Thread class calls the run method of runnable object
                Thread t=new Thread(runnable);
                t.start();
        }
}
```

ThreadDemo3.java

```
import static java.lang.System.*;
class MyThread1 extends Thread
{       public void run()
        {       for(int i=1;i<=10;i++)
                {out.println(getName()+":\t"+i);
                }
        }
}
class MyThread2 extends Thread
{       public void run()
        {       for(int i=100;i<=110;i++)
                {out.println(getName()+":\t"+i);
                }
        }
}
class ThreadDemo3
{       public static void main(String args[])
        {        MyThread1 t1=new MyThread1();//Thread-0
                MyThread2 t2=new MyThread2();//Thread-1
                t1.start();
                t2.start();
        }
}
```

Life cycle of a thread



Thread Scheduler

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Allocation of microprocessor time for al the threads is controlled by ThreadShedular, managed by the operating system. For allocation of microprocessor time, thread scheduler takes into consideration many factors like a) the waiting period of thread b) the priority of thread etc. If a thread is in inactive state (as in sleep () method), thread scheduler keeps it away from the pool o active threads. When the inactive state is over, the thread joins back the pool of threads waiting for microprocessor time. It is the job of thread scheduler to manage the pool of threads – when and which thread must be given microprocessor time. Algorithm of thread scheduler is operating system dependent. Different operating systems adopt different algorithms.

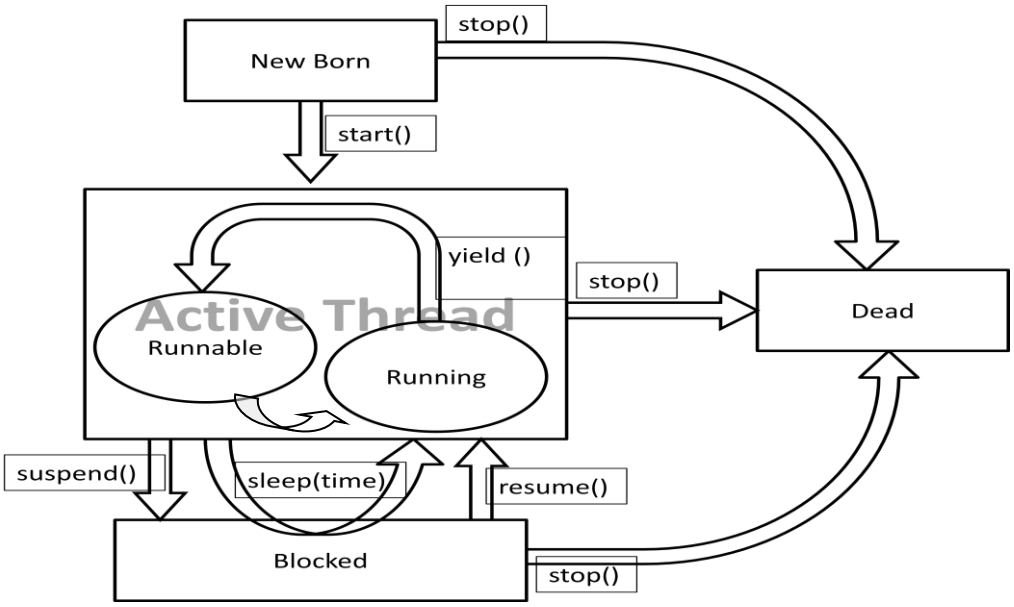
Thread scheduler allocates the processor time first to the higher priority threads. To allocate microprocessor time in between the threads of the same priority, thread scheduler follows round-robin fashion.

The Thread class defines several methods that help manage threads. The ones that will be used in this chapter are shown here:

Method	**Meaning**
getName	Obtain a thread's name.
setName	It sets a name for a thread
getPriority	Obtain a thread's priority.
isAlive	Determine if a thread is still running.
Join	This method waits until the thread on which it is called terminates.(join method does not allow the parent thread to die until all its child threads die)
Run	Entry point for the thread.
Sleep	Suspends a thread for a period of time.
Suspend	It suspends a thread (sends it into a blocked state)
Resume	It gets a thread into Runnable state from blocked state
wait and notify or notifyAll	wait and notify methods are defined in Object class and inherited by Thread class . wait method is used to suspend notify method is used to resume a thread and notifyAll method is used to resume more than one thread at a time but these methods only used with synchronization.
Start	Starts a thread by calling its run method.
Yield	It gets a thread into Runnable state from running state
getPriority	It gets the priority of a thread
setPriority	It sets the priority for a thread
isDeamon	Returns true if the thread is daemon thread else false

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

getThreadGroup	Returns the name of the thread group for which the thread belongs.



Thread Life Cycle

What is Thread LifeCycle

Different states in which a thread exist between its birth and garbage collection is called the life cycle of thread.

States of a thread: every thread exists in some of the following five states, between it's creation and destruction those are.

- **New Born**
- **Runnable**
- **Running**
- **Blocked**
- **Dead**

Born state:

When a thread is created, we can say that it is in the new born state. A thread in born state is inactive and thereby not eligible for microprocessor time; but still occupies memory in the RAM.

```
MyThread mt=new MyThread();
```

In the above statement, "**mt**" thread is created but it is not active. To make it active, we have to call **start ()** method.

Runnable state: It is also called as ready to run sate or active state. When a thread is created, it is inactive and to make it active, we call start () method on it as follows:

```
mt.start ();
```

When you call the **start ()** method the thread will goes to the **runnable** state. Means not in a running state but it is in a ready to run state (ready to share the microprocessor time).

Running State: The thread, which is in runnable state automatically goes to the Running state when processor invokes the run() method of it.

Blocked State: A running thread can be made temporarily inactive for a short period by getting it into blocked state. The thread in blocked state is inactive and not eligible for processor time.
Dead State: When the thread's run method execution is over, it automatically goes to the dead state. If you want, you can send a thread from any state to dead state by calling stop () method.

Thread Priorities:

Thread priorities are used by the thread scheduler to decide when each thread should be allowed to run. In theory, higher-priority threads get more CPU time than lower priority threads. In practice, the amount of CPU time that a thread gets often depends on several factors besides its priority. (For example, how an operating system implements multitasking can affect the relative availability of CPU time.) A higher-priority thread can also preempt a lower-priority one. For instance, when a lower-priority thread is running and a higher-priority thread resumes (from sleeping or waiting on I/O, for example), it will preempt the lower-priority thread.

In theory, threads of equal priority should get equal access to the CPU. But you need to be careful. Remember, Java is designed to work in a wide range of environments. Some of those environments implement multitasking fundamentally differently than others. For safety, threads that share the same priority should yield control once in a while. This ensures that all threads have a chance to run under a non preemptive operating system. In practice, even in non preemptive environments, most threads still get a chance to run, because most threads inevitably encounter some blocking situation, such as waiting for I/O. When this happens, the blocked thread is suspended and other threads can run. But, if you want smooth multithreaded execution, you are better off not relying on this. Also, some types of tasks are CPU-intensive. Such threads dominate the CPU. For these types of threads, you want to yield control occasionally, so that other threads can run. To set a thread's priority, use the **setPriority ( )** method, which is a member of Thread. This is its general form: final void setPriority (int *level*) Here, *level* specifies the new priority setting for the calling thread. The value of *level* must be within the range **MIN_PRIORITY** and **MAX_PRIORITY**. Currently, these values are 1 and 10, respectively. To return a thread to default priority, specify **NORM_PRIORITY**, which is currently 5. These priorities are defined as **final** variables within **Thread**.

You can obtain the current priority setting by calling the **getPriority ( )** method of **Thread**, shown here:

```
final int getPriority( )
```

Implementations of Java may have radically different behavior when it comes to scheduling. The **Windows XP/98/NT/2000** version works, more or less, as you would expect. However, other versions may work quite differently

Example on thread priorities

ThradDemo2.java

```
import static java.lang.System.*;
class MyThread1 extends Thread
{       public void run()
        {       for(int i=1;i<=10;i++)
                {out.println(getName()+":\t"+i);
                }
        }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
}
class MyThread2 extends Thread
{	public void run()
	{	for(int i=100;i<=110;i++)
		{out.println(getName()+":\t"+i);
		}
	}
}
class ThreadDemo4
{	public static void main(String args[])
	{
		MyThread1 t1=new MyThread1();
		t1.setName("One");
		MyThread2 t2=new MyThread2();
		t2.setName("Two");
		//default priority of a thred is 5
		out.println("t1 priority:\t"+t1.getPriority());
		out.println("t2 priority:\t"+t2.getPriority());
		//we set the max priority
		t1.setPriority(Thread.MAX_PRIORITY);
		//t2.setPriority(Thread.NORM_PRIORITY);
		//t1.setPriority(10);
		//we set the min priority
		//t2.setPriority(1);
		t2.setPriority(Thread.MIN_PRIORITY);
		t2.start();
		t1.start();
	}
}
```

Sleep and interrupt usage

```
import static java.lang.System.*;
class MyThread1 extends Thread
{
	MyThread1(String name)
	{super(name);
	}
	public void run()
	{	for(int i=1;i<=10;i++)
		{	if(i==3)
			{	try{
				//1000 milliseconds =1 sec
				out.println("Mummy nenu indrabothunn... mudukaale jepthunn.... 1
second varku nidra lepoddu...");
				sleep(1000);
				out.println("Thanks mummy.....");
				}catch(InterruptedException ie)
				{out.println("Yevadra..... nanu interrupt
chesindhi.....dishkyam.....diskyam....");
				}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                        }
                        out.println(getName()+":\t"+i);
                }
        }
}
class MyThread2 extends Thread
{
        MyThread1 t1;
        MyThread2(String name,MyThread1 t1)
        {super(name);
        this.t1=t1;
        }
        public void run()
        {       for(int i=100;i<=110;i++)
                {out.println(getName()+":\t"+i);
                }
                out.println("Yentha.... sepu nidrapothaave..... ley... legu... mee friend vachondi.....?");
                t1.interrupt();
        }
}
class ThreadDemo5
{       public static void main(String args[])
        {
                MyThread1 t1=new MyThread1("One");
                MyThread2 t2=new MyThread2("Two",t1);
                t1.start();
                t2.start();
        }
}
```

SuspendResume example

ThreadDemo6.java

```
import static java.lang.System.*;
class MyThread1 extends Thread
{
        MyThread1(String name)
        {super(name);
        }
        public void run()
        {       for(int i=1;i<=10;i++)
                {       if(i==3)
                        {
out.println("Mummy nenu indrabothunn..... Yedaina... urgent ithe.... nidra lepu...mummy");
                                suspend();
                                out.println("Yenti mummy.....work yemaina unda...");
                        }
                        out.println(getName()+":\t"+i);
                }
        }
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
class MyThread2 extends Thread
{
	MyThread1 t1;
	MyThread2(String name,MyThread1 t1)
	{super(name);
	this.t1=t1;
	}
	public void run()
	{	for(int i=100;i<=110;i++)
		{out.println(getName()+":\t"+i);
		}
		out.println("Nidra leraaa nanna..... mee friend vachondi....");
		t1.resume();
	}
}
class ThreadDemo6
{	public static void main(String args[])
	{	MyThread1 t1=new MyThread1("One");
		MyThread2 t2=new MyThread2("Two",t1);
		t1.start();
		t2.start();
	}
}
```

Join (): it makes the parent thread to wait, until joined thread completes its task.

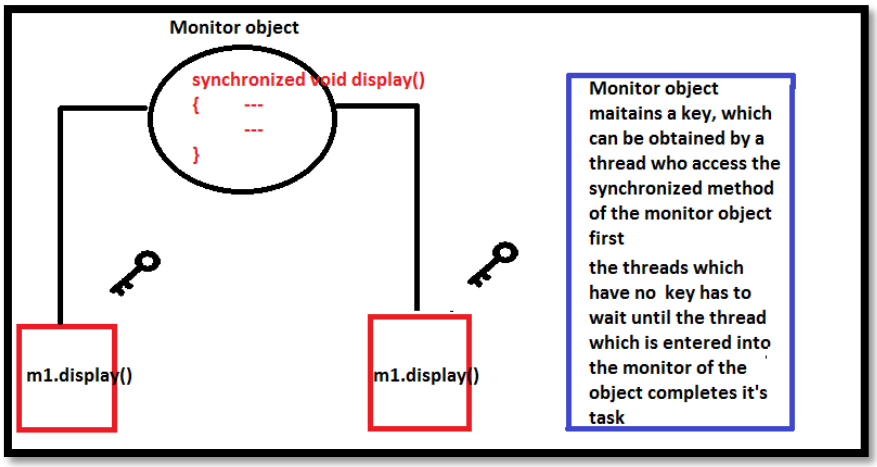
```
import static java.lang.System.*;
class MyThread1 extends Thread
{	MyThread1(String name)
	{super(name);
	}
	public void run()
	{	for(int i=1;i<=10;i++)
		{out.println(getName()+":\t"+i);
		}
	}
}
class MyThread2 extends Thread
{	MyThread2(String name)
	{super(name);
	}
	public void run()
	{	for(int i=100;i<=110;i++)
		{out.println(getName()+":\t"+i);
		}
	}
}
class ThreadDemo7
{	public static void main(String args[])throws Exception
	{	out.println("Start of main");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                MyThread1 t1=new MyThread1("One");
                MyThread2 t2=new MyThread2("Two");
                t1.start();
                t1.join();
                t2.start();
                out.println("End  of main");
        }
}
```

What is thread synchronization?

Generally, one thread executes one resource of data at a time. There may be situation when multiple threads access the same data at a time. For example, the joint account holders of an account may access their bank account at a time from different teller counters in bank. This at times lead to data inconsistency, it is necessary to carefully coordinate the actions between threads especially when multiple threads access the same data. To overcome this situation, the threads are to be coordinated or synchronized. **Synchronization is the way to see only one thread access one resource at a time.**



We can achieve synchronization In two ways

- **By declaring methods as synchronized**
- **By using synchronized block**

SynchDemo1.java (By declaring methods as synchronized)

```
import static java.lang.System.*;
class Monitor
{       synchronized public void display()
        {       Thread t=Thread.currentThread();
                out.print(t.getName()+"  [");
                for(int i=1;i<=10;i++)
                out.print(i+" ");
                out.println("]");
        }
}
class MyThread1 extends Thread
{       Monitor m;
        MyThread1(Monitor m)
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {this.m=m;
        }
        public void run()
        {m.display();
        }
}
class MyThread2 extends Thread
{       Monitor m;
        MyThread2(Monitor m)
        {this.m=m;
        }
        public void run()
        {m.display();
        }
}
class SynchDemo1
{       public static void main(String arg[])
        {       Monitor m=new Monitor();
                MyThread1 t1=new MyThread1(m);
                MyThread2 t2=new MyThread2(m);
                t1.setName("First");
                t2.setName("Second");
                t1.start();t2.start();
        }
}
```

SynchDemo2.java (by using synchronized block)

```
import static java.lang.System.*;
class Monitor
{       public void display()
        {       Thread t=Thread.currentThread();
                out.print(t.getName()+"  [");
                for(int i=1;i<=10;i++)
                out.print(i+" ");
                out.println("]");
        }
}
class MyThread1 extends Thread
{       Monitor m;
        MyThread1(Monitor m)
        {this.m=m;
        }
        public void run()
        {       synchronized(m)
                {m.display();
                }
        }
}
class MyThread2 extends Thread
{       Monitor m;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        MyThread2(Monitor m)
        {this.m=m;
        }
        public void run()
        {       synchronized(m)
                {m.display();
                }
        }
}
class SynchDemo2
{       public static void main(String arg[])
        {       Monitor m=new Monitor();
                MyThread1 t1=new MyThread1(m);
                MyThread2 t2=new MyThread2(m);
                t1.setName("First");
                t2.setName("Second");
                t1.start();
                t2.start();
        }
}
```

Interrupted Exception

We can interrupt a thread which is in blocked state, by calling interrupt () method on that thread. When you interrupt a thread, JVM throws an Interrupted exception, and also makes that thread to come in to runnable state (active state).

InterruptDemo.java

```
import static java.lang.System.*;
class Thread1 extends Thread
{       public void run()
        {       for(int i=0;i<15;i++)
                {
                if(i==0)
                try{sleep(10000);
                }catch(Exception e)
                {out.println(e);}
                out.println(i);
                }
        }
}
class Thread2 extends Thread
{       Thread1 t1;
        Thread2(Thread1 t1)
        {this.t1=t1;
        }
        public void run()
        {       for(int i=0;i<20;i++)
                {
                out.println("Hai");
                if(i==5)
                t1.interrupt();
```

```
        }
        }
}
class InterruptDemo
{       public static void main(String args[])
        {       Thread1 t1=new Thread1();
                Thread2 t2=new Thread2(t1);
                t1.start();t2.start();
        }
}
```

Inter-thread communication (Co-operation)

It is all about making synchronized threads communicate with each other cooperation is a mechanism in which a thread is paused running in its critical section and another thread is allowed to run (or lock) in the same critical section to be executed. It is implemented by following methods, which are existed in the **java.lang.Object** class.

- wait()
- notify()
- notifyAll()

wait(): It causes current thread to release the lock and wait until either another thread invokes the notify() method or notifyAll() method for this object, or a specified amount of time has elapsed.
notify(): It makes the thread to move from blocked state to runnable state.
notifyAll(): It makes all the threads to move which are in blocked state to runable state

Example: InterThreadCommuni.java

```
import static java.lang.System.*;
class Customer
{       int amount;
        int flag;
        public synchronized  int withdraw(int amount)
        {       out.println(Thread.currentThread().getName()+"  Is going to withdraw");
                if(flag==0)
                {       try{    out.println("waiting");
                                wait();
                        }catch(Exception e)
                        {
                        }
                }
                this.amount-=amount;
                out.println("withdraw compleated and Bal:\t"+this.amount);
                return amount;
        }
        public synchronized  void deposit(int amount)
        {       out.println(Thread.currentThread().getName()+"  Is going to deposit");
                this.amount+=amount;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                flag=1;
                notifyAll();
                out.println("deposit compleated  & Tot Amt    "+this.amount);
        }
}
class One extends Thread
{       Customer c;
        One(Customer c)
        {this.c=c;
        }
        public void run()
        {c.withdraw(1000);
        }
}
class Two extends Thread
{       Customer c;
        Two(Customer c)
        {this.c=c;
        }
        public void run()
        {c.deposit(10000);
        }
}
class InterThreadCommuni
{       public static void main(String args[])
        {       final Customer c=new Customer();
                One t1=new One(c);
                Two t2=new Two(c);
                t1.start();
                t2.start();
        }
}
```

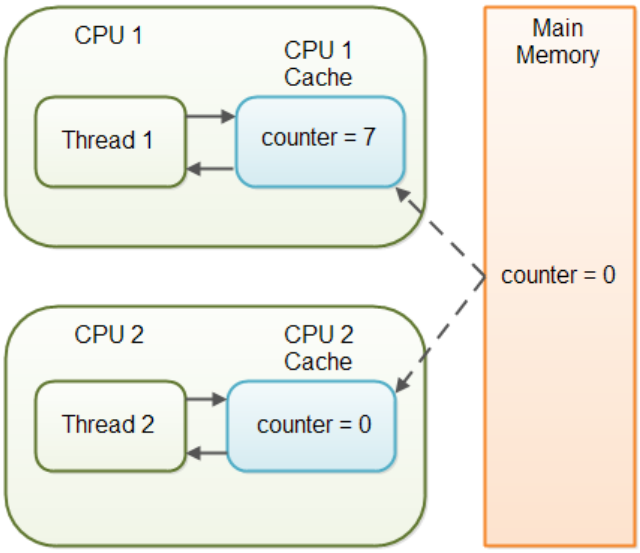
What is Volatile?

Volatile is a modifier and it is useful in multi threaded program. In multi thread program each program maintains cache of local variables for fast computing purpose. Updating the value will be done in cache not in main memory.

Declaring a variable volatile means

* There will be no cache maintained by each thread that means all the changes made in main memory.
* Access to this variable acts as synchronized block.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.



If you observe the above example each thread is maintaining the separate catch for counter variable. If Thread-1 modifies the counter variable it will be reflected in catch memory not in main memory, in such a situation if Thread-2 calls access the counter variable from meory it will get only value zero (0) not the modified one by the Thread-1. To overcome this problem we have to declare that counter variable as volatile, for volatile variable there is not cache will be maintained by JVM. So threds directly access from main memory, modify the counter variable directly in the main memory.

VolatileDemo.java

```
import static java.lang.System.*;
class SharedObject
{public volatile int counter = 0;
}
class One extends Thread
{       SharedObject so;
        One(SharedObject so)
        {this.so=so;
        }
        public void run()
        {       for(int i=1;i<=10;i++)
                {out.println(getName()+":\t"+(++so.counter));
                //so.counter++;
                }
        }
}
class Two extends Thread
{       SharedObject so;
        Two(SharedObject so)
        {this.so=so;
        }
        public void run()
        {       for(int i=1;i<=10;i++)
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                {out.println(getName()+":\t"+so.counter);
                }
        }
}
class Volatile
{       public static void main(String args[]) throws InterruptedException
        {
                SharedObject so=new SharedObject();
                One t1=new One(so);
                Two t2=new Two(so);
                t1.start();
                t2.start();
        }
}
```

What is Daemon Threads?

A daemon thread is a thread that runs for the benefit of other threads. Daemon threads are known as service threads. Daemon threads run in the background and come into execution when the microprocessor is idle. The garbage collector is a good example for a daemon thread. We can set a thread to be a daemon one, by calling the method **setDaemon (true)**. To make daemon thread as user thread you have to call the **setDaemon (false).**

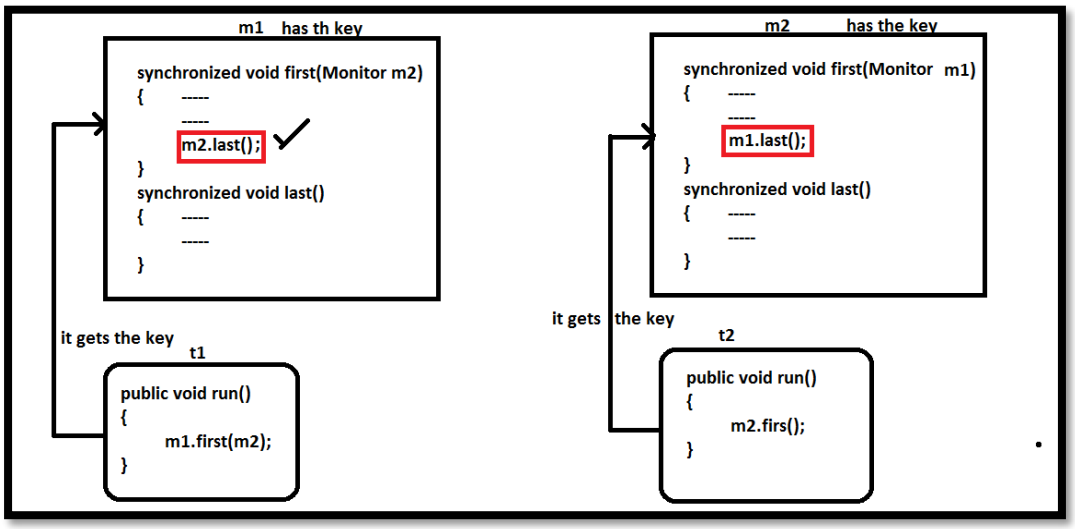
DaemonDemo.java

```
import static java.lang.System.*;
class MyThread1 extends Thread
{       public void run()
        {       for(int i=1;i<=10;i++)
                out.println(getName()+":\t"+i);
        }
}
class MyThread2 extends Thread
{       public void run()
        {       for(int i=1;i<=10000;i++)
                out.println(getName()+":\t"+i);
        }
}
class DaemonDemo
{       public static void main(String arg[])
        {       MyThread1 t1=new MyThread1();
                MyThread2 t2=new MyThread2();
                t1.setName("UserThread");
                t2.setName("DaemonThread");
                t2.setDaemon(true);
                t1.start();
                t2.start();
        }
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Dead Lock

In a multithreaded program, if we do not synchronize multiple threads properly, a deadlock situation may arise and if such a situation comes, the program simply hangs. Deadlock comes between synchronized threads only. It occurs when two or more threads wait indefinitely for each other to obtain locks (that is, two threads depends on each other). Say for example, there are two threads; one thread's output is another's input. Now it is a deadlock. Deadlock will not allow both the threads to go further. That is, for execution of one thread depends on the other.



DeadLockDemo.java

```
import static java.lang.System.*;
class Monitor1
{       synchronized void first(Monitor2 m2)
        {       Thread t=Thread.currentThread();
                out.println(t.getName()+" Entered Into the monitor1");
                try{
                t.sleep(2000);
                }catch(Exception e){}
                m2.last();
        }
        synchronized void last()
        {out.println("last method of Monitor 1");
        }
}
class Monitor2
{       synchronized void first(Monitor1 m1)
        {       Thread t=Thread.currentThread();
                out.println(t.getName()+" Entered Into the monitor 2");
                try{
                t.sleep(2000);
                }catch(Exception e){}
                m1.last();
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        }
        synchronized void last()
        {out.println("last method of Monitor 2");
        }
}
class Thread1 extends Thread
{       Monitor1 m1;Monitor2 m2;
        Thread1(Monitor1 m1,Monitor2 m2)
        {this.m1=m1;
        this.m2=m2;
        }
        public void run()
        {m1.first(m2);
        }
}
class Thread2 extends Thread
{       Monitor1 m1;Monitor2 m2;
        Thread2(Monitor1 m1,Monitor2 m2)
        {this.m1=m1;
        this.m2=m2;
        }
        public void run()
        {m2.first(m1);
        }
}
class DeadLockDemo
{       public static void main(String args[])
        {       Monitor1 m1=new Monitor1();
                Monitor2 m2=new Monitor2();
                Thread1 t1=new Thread1(m1,m2);
                Thread2 t2=new Thread2(m1,m2);
                t1.start();t2.start();
        }
}
```

Thread Groups

Every thread belongs to some thread group. That is, no thread exists without a thread group (like no file can exist without a directory). When a thread is created without assigning a group name, by default it is put into the main thread group. A thread group can have both threads and other sub thread groups as members (like a directory can have both directories and files). Main is a thread in the thread group called main.The following is the signature of the class: **public class ThreadGroup extends Object**

The importance of a thread group is that all the threads in a thread group can be stopped together by calling tg.stop () or can be suspended by calling tg.suspend () or can be resumed by

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

calling tg.resume () where tg is a thread group name. **But we can't start by calling tg.start ().** Each thread in a group must be started individually.

ThreadGroupDemo.java

```
import static java.lang.System.*;
import java.io.*;
class MyThread1 extends Thread
{	MyThread1(ThreadGroup tg,String name)
	{super(tg,name);
	}
	public void run()
	{	for(int i=0;i<10;i++)
		{	try
			{if(i==0)
			sleep(2000);
			}catch(Exception e){}
		}
	}
}
class MyThread2 extends Thread
{	ThreadGroup tg1;
	MyThread2(ThreadGroup tg,String name)
	{	super(tg,name);
		tg1=tg;
	}
	MyThread2(String name)
	{	super(name);
	}
	public void run()
	{	for(int i=0;i<10;i++)
		{
		try
		{if(i==0)
		sleep(2000);
		}catch(Exception e){}
		}
	}
}
class ThreadGroupDemo
{	public static void main(String a[])throws Exception
	{		Thread main=Thread.currentThread();
			ThreadGroup mtg=main.getThreadGroup();
			ThreadGroup mtgp=mtg.getParent();
			ThreadGroup tg1=new ThreadGroup("TG1");
			out.println("tg1 parent:\t"+(tg1.getParent()).getName());
			MyThread1 t1=new MyThread1(tg1,"First");
			MyThread2 t2=new MyThread2(tg1,"Second");
			MyThread2 t3=new MyThread2(tg1,"Third");
			t1.start();t2.start();t3.start();
			tg1.list();
			tg1.suspend();
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                        out.println("D U Want to resume (yes/no)");
                        if(((new DataInputStream(System.in)).readLine()).equals("yes"))
                        tg1.resume();
                        out.println("End of main method...");
        }
}
```

GENERIC TYPES

A generic class represents a class that is type-safe. This means a generic class can act upon any data type. Similarly, a generic interface is also type-safe and hence it can use any data type. Generic classes and Generic interfaces are also called as **"parameterized types",** because they use a parameter that determines which data type they should contain.

GenType1.java

```
import static java.lang.System.*;
class One<T>
{       T a;
        One(T a)
        {this.a=a;
        }
        void display()
        {out.println("a:\t"+a);
        }
}
class GenType1
{       public static void main(String args[])
        {       Integer io=new Integer(100);
                One<Integer> o1=new One<Integer>(io);
                Float f=new Float(123.34f);
                One<Float> o2=new One<Float>(f);
                o1.display();
                o2.display();
        }
}
```

GenType2.java

```
import static java.lang.System.*;
class One<T1,T2>
{       T1 a;
        T2 b;
        One(T1 a,T2 b)
        {this.a=a;
        this.b=b;
        }
        void display()
        {out.println("Values....");
        out.println("a:\t"+a);
        out.println("b:\t"+b);
        }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
}
class GenType2
{	public static void main(String args[])
	{	Integer io=new Integer(100);
		One<Integer,String> o1=new One<Integer,String>(io,"Madhu");
		Float f=new Float(123.34f);
		One<Float,Double> o2=new One<Float,Double>(f,245.50);
		o1.display();
		o2.display();
	}
}
```

GenType3.java

```
import static java.lang.System.*;
class One<T extends Number>
{	T a;
	One(T a)
	{this.a=a;
	}
	@Override
	public String toString()
	{	return "One@[a="+a+"]";
	}
}
class GenTypes3
{	public static void main(String args[])
	{	One<Integer> o1=new One<Integer>(100);
		One<Long> o2=new One<Long>(100l);
		One<Float> o3=new One<Float>(100.00f);
		out.println(o1);
		out.println(o2);
		out.println(o3);
	}
}
```

GenType3.java

```
import static java.lang.System.*;
class One<T1 extends Number,T2 extends Number>
{	T1 a;
	T2 b;
	One(T1 a,T2 b)
	{this.a=a;
	this.b=b;
	}
	public void add()
	{out.println("add:\t"+(a.doubleValue()+b.doubleValue()));
	}
}
class GenType3
{	public static void main(String args[])
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
	{	One<Integer,Float> o1=new One<Integer,Float>(100,100.50f);
		One<Integer,Long> o2=new One<Integer,Long>(100,10l);
		o1.add();
		o2.add();
	}
}
```

GenType4.java

```
import static java.lang.System.*;
class One<T1 extends Number,T2 extends Number>
{	T1 a;
	T2 b;
	One(T1 a,T2 b)
	{	this.a=a;
		this.b=b;
	}
	public double add()
	{return a.doubleValue()+b.doubleValue();
	}
	public double sub()
	{return a.doubleValue()-b.doubleValue();
	}
	public double mul()
	{return a.doubleValue()*b.doubleValue();
	}
}
class GenType4
{	public static void main(String args[])
	{	One<Integer,Integer> o1=new One<Integer,Integer>(10,20);
		One<Integer,Integer> o2=new One<Integer,Integer>(100,20);
		out.println("-----o1 object-----");
		out.println(o1.add());
		out.println(o1.sub());
		out.println(o1.mul());
		out.println("----o2 object----");
		out.println(o2.add());
		out.println(o2.sub());
		out.println(o2.mul());
	}
}
```

Generic Methods

Just like generic classes, we can write generic methods and these methods can act upon different types of arguments.

```
import static java.lang.System.*;
class RW
{	static public <E> void write(E []arr)
	{	out.println("Elements in an array");
		for(E v:arr)
		{out.print(v+"\t");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                }
        }
}
class GenMethods
{       public static void main(String args[])
        {       RW.write(new Integer[]{10,20,30,40,50});
                out.println();
                RW.write(new String[]{"Nandini","Tirumala","Joyes(NL)","Sruthi","Roja"});
        }
}
```

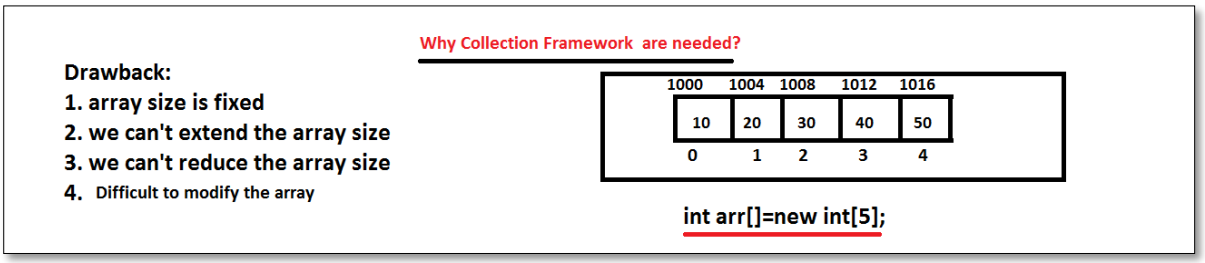
Collection Framework

What is framework?

A software framework is a re-usable design for a software system. It may include support programs, code libraries like set of classes etc…

Why we use collection framework?

By using it we can manage the collection of objects. That is we can add, remove, sort and we can do many more things efficiently.



Advantages of Collection Framework

1. We can create dynamic arrays, which can be easily resizable, manageable.
2. We can sort collectin of elements very easily
3. We can add, remove, replace an element easily
4. Element finding becomes easy
5. We can maintain collection of object with duplicates, without duplicates, with null values, without null values, with order without order, with insertion order, with sorting order.

What is Collection?

A Collection represents a group of objects (elements). This framework is provided in the java.util package. Collections can be used in various scenarios like storing phone numbers, employee names etc. They are basically used to group multiple elements into a single unit. Some Collections allow duplicate elements while others do not. Some collections are ordered and others are not.

9 key interfaces of Collection

1. Collection

2. List
3. Set
4. Sorted Set
5. Navigable Set
6. Queue
7. Map
8. Sorted Map
9. Navigable Map

What is a collection class?

It is a class which implements collection interface

Collection interface:

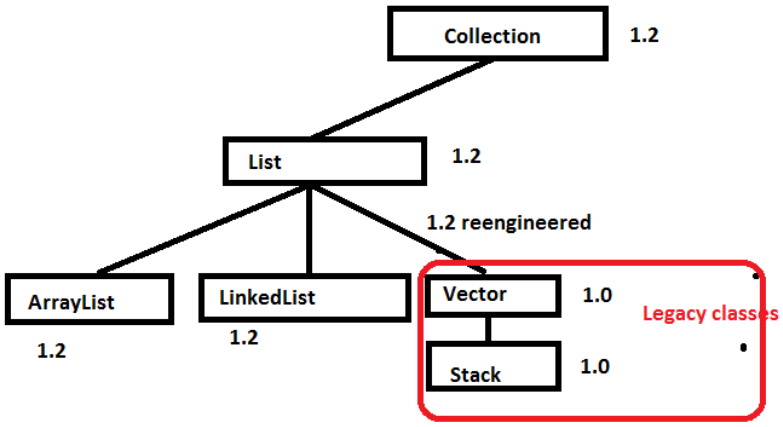
1. If you want to represent group of objects as a single entity then we have to use collection interface.
2. It contains the methods which are applicable for any collection object. Some of those methods are used to
    a. Add element
    b. Remove element
    c. Replace element
    d. Is element is existed or not
    e. Etc...
3. In general collection interface is consider as root interface of collection framework.
4. There is no concrete class which implements collection interface directly

What is the Difference between Collection and Collections?

Collection	Collections
It is an interface, which can be used to represent group of objects	It is a utility class, existed in java.util package it contains several utility methods which are used to sort, search objects etc... of a collection.

List (I): It is the child interface of Collection, if we want to represent a group of individual objects as a single entity, where duplicates are allowed and insert order must be preserved then we should go for list.

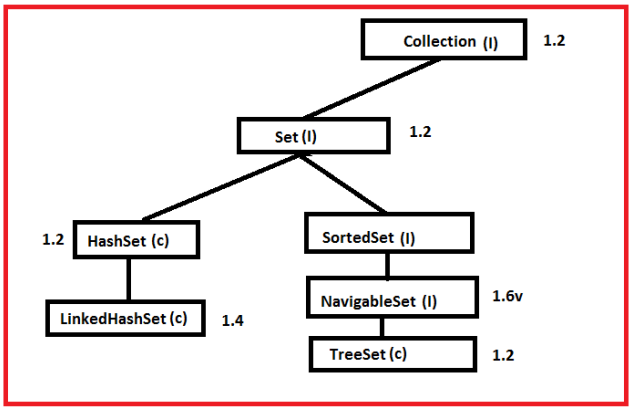
1. Duplicate are allowed.
2. It supports Insertion order.
3. It accepts null values.



1. 2 version vector and stack classes are reengineered or modified to implements List interface

Set (I): It is the child interface of Collection, if we want to represent a group of individual objects as a single entity, where duplicates are not allowed and insertion order not required. Then we should go for set interface.

1. where duplicates are not allowed and insertion order not required
2. it accepts null values.



Java.util.NavigableSet

```
E:\MeeruSuperehe>
javap java.util.NavigableSet
Compiled from "NavigableSet.java"
public interface java.util.NavigableSet<E> extends java.util.SortedSet<E> {
  public abstract E lower(E);
  public abstract E floor(E);
  public abstract E ceiling(E);
  public abstract E higher(E);
  public abstract E pollFirst();
  public abstract E pollLast();
  public abstract java.util.Iterator<E> iterator();
  public abstract java.util.NavigableSet<E> descendingSet();
  public abstract java.util.Iterator<E> descendingIterator();
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
  public abstract java.util.NavigableSet<E> subSet(E, boolean, E, boolean);
  public abstract java.util.NavigableSet<E> headSet(E, boolean);
  public abstract java.util.NavigableSet<E> tailSet(E, boolean);
  public abstract java.util.SortedSet<E> subSet(E, E);
  public abstract java.util.SortedSet<E> headSet(E);
  public abstract java.util.SortedSet<E> tailSet(E);
}
```

What is the difference between HashSet and LinkedHashSet?

A **HashSet** is unordered and unsorted Set. **LinkedHashSet** is the ordered version of **HashSet**.The only **difference between HashSet and LinkedHashSet** is that **LinkedHashSet** maintains the insertion order. When we iterate through a **HashSet**, the order is unpredictable while it is predictable in case of **LinkedHashSet**.

Note: **TreeSet** is sorted, it supports ascending order.

Important points to remember

1. There's no need to call equals if hashCode differs
2. There's no need to call hashCode if (obj1 == obj2)
3. There's no need for hashCode and/or equals just to iterate - you're not comparing objects
4. When needed to distinguish in between object.

HashMap actually calculates the hash to find the correct bucket and uses it before comparing object identity with == or calling equals(), but otherwise this is correct. equals() isn't called in the second example because == already detected the duplicate.

There's no need for hashCode and/or equals just to iterate - you're not comparing objects --- Then how would it get the current bucket if it don't call hashCode in iteration?

SortedSet (I): it is the child interface of Set, if we want to represent a group of individual objects as a single entity, where duplicates are not allowed and all objects should be inserted, according to some sorting order then we should go for sorted set.

Navigable Set (I): It is the child interface of SortedSet it contains several methods for Navigation purposes.

TreeSet: It is implementing NavigableSet, which provides methods for navigation purpose. Because it is implementing SortedSet it maintains elements in ascending order. To the TreeSet collection we have to pass objects which implements java.lang.Comparable<T> interface

TreeSetDemo.java

```
import static java.lang.System.*;
import java.util.*;
class One implements Comparable<One>
{        int a,b;
         One(int a,int b)
         {this.a=a;
         this.b=b;
         }
         public String toString()
         {return getClass().getName()+"@[a="+a+",b="+b+"]";
         }
         @Override
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
	public int hashCode() {
		int hash = 5;
		hash = 23 * hash + this.a;
		hash = 23 * hash + this.b;
		return hash;
	}
	public int compareTo(One o)
	{//return this.hashCode()-o.hashCode();
			Int ar=this.a-o.a;
		   Int br=this.b-o.b;
			If(ar==0 && b==0)
			Return 0;
			If(ar!=0)
			Return ar;
			If(br!=0)
			Return br;
	}
}
class TreeSetDemo
{	public static void main(String args[])
	{
		/*TreeSet<String> ts=new TreeSet<String>();
		ts.add("bcd");
		ts.add("abc");
		ts.add("cde");
		out.println(ts);
		*/
		TreeSet<One> ts=new TreeSet<One>();
		ts.add(new One(10,20));
		ts.add(new One(120,10));
		ts.add(new One(100,200));
//ts.add(null);//we will get runtime exception
		out.println(ts);
		/*One o1=new One(10,20);
		One o2=new One(20,10);
		out.println(o1.compareTo(o2));
		*/
	}
}
```

LinkedHashSetDemo.java (Check the same program after removing equals () method of One class)

```
import static java.lang.System.*;
import java.util.*;
class One// implements Comparable<One>
{	 int a,b;
	One(int a,int b)
	{this.a=a;
	this.b=b;
	}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
	public String toString()
	{return getClass().getName()+"@[a="+a+",b="+b+"]";
	}
	@Override
	public int hashCode() {
		int hash = 5;
		hash = 23 * hash + this.a;
		hash = 23 * hash + this.b;
		return hash;
	}
@Override
   public boolean equals(Object obj) {
      if (this == obj) {
         return true;
      }
      if (obj == null) {
         return false;
      }
      if (getClass() != obj.getClass()) {
         return false;
      }
      final One other = (One) obj;
      if (this.a != other.a) {
         return false;
      }
      if (this.b != other.b) {
         return false;
      }
      return true;
   }
	/*public int compareTo(One o)
	{return this.hashCode()-o.hashCode();
	}*/
}
class HashSetDemo
{	public static void main(String args[])
	{
		/*TreeSet<String> ts=new TreeSet<String>();
		ts.add("bcd");
		ts.add("abc");
		ts.add("cde");
		out.println(ts);
		*/

		LinkedHashSet<One> ts=new LinkedHashSet<One>();
		ts.add(new One(122,20));
		ts.add(new One(120,10));
		ts.add(new One(120,10));
		ts.add(new One(12220,200));
		out.println(ts);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

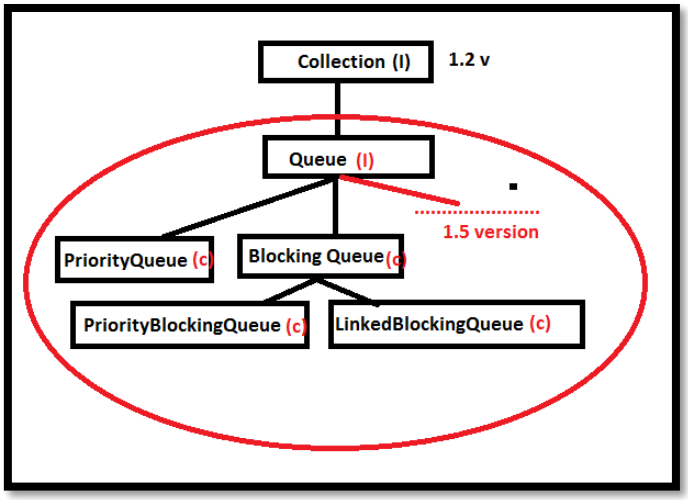
```
                /*One o1=new One(10,20);
                One o2=new One(20,10);
                out.println(o1.compareTo(o2));
                */
        }
}
```

Differences between List and Set

List	Set
1. Duplicates are allowed	1. Duplicates are not allowed
2. Insertion order preserved	2. Insertion order not preserved

Queue (I): Queue is child interface of Collection; if we want to represent a group of individual objects prior to processing () then we should go for queue. Usually queue follows first in first out order but Based on our requirement we can implement our own priority order also.

Ex: before sending a mail we have to store all mail id's in some data structure, in which order we added mail id's in the same order only mail should be delivered for this requirement queue is the best choice.

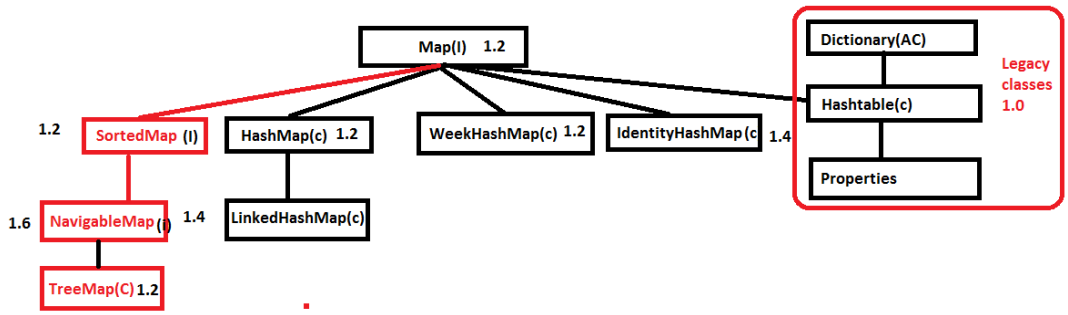


All the above interfaces (Collection, List, Set, SortedSet, NavigableSet and Queue) are used for representing a group of individual objects. If you want to represent a group of objects a key value pairs then we should go for Map.

Map (I):

1. It is not child interface of Collection interface but it is the part of Collection Framework
2. If you want to represent a group of objects as key value pairs then we should go for Map
3. Duplicate keys are not allowed but the values can be duplicated
4. Both key and values are objects only

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.



SortedMap: it is the child interface of Map. If we want to represent a group of key, value pairs according to some sorting order of keys, then we should go for sorted map.

1. In SortedMap the sorting should be based on key but not based on value.

Navigable Map: It is the child interface of SortedMap. It defines several methods for Navigation Purposes.
TreeMap: it is the implemented class for Navigable Map interface
Sorting: If you want default natural setting comparable and if you want customized sorting then we should go for comparator

1. **Comparable**
2. **Comparator**

Cursors

1. **Enumeration**
2. **Iterator**
3. **ListIterator**

Utility Classes

1. **Collections**
2. **Arrays**

The following are legacy characters present in collection framework

1. Enumeration(I)
2. Dictionary(AC)
3. Vector(c)
4. Stack(c)
5. Hashtable(c)
6. Properties(c)

Examples

Commonly used methods in Collection interface

- public boolean add(Object element): is used to add an element to a collection
- public boolean addAll(Collection c): used to add specified collection into present collection
- public boolean remove(Object element): it removes the specified element from the collection
- public boolean removeAll(Collection c): used to delete the collection of elements from a collection
- public boolean retainAll(Collection c):It removes the all elements except specified collection of elements
- public int size(); it returns the size of the collection
- public void clear(): removes all elements from collection
- public boolean contains(Object elemenet):used to search an element
- public boolean containsAll(Collection c):used to search specified collection
- public Iterator iterator():returns an iterator object

ArrayList:

1. Array List is a dynamic array that stores objects. (size increases when you add and shrinks when you delete).
2. It extends **AbstractList** class and implements **List, RandomAccess, Cloneable, Serializable** interfaces
3. It can contain duplicate elements
4. It Maintains insertion order
5. It is not synchronized
6. It accepts Random access because array works at the index basis.
7. It uses an array interanally to manage collection of objects.

Drawback:

1. Manipulation is slow because a lot of shifting needs to be occured.

ArrayListDemo1.java

```
import static java.lang.System.*;
import java.util.*;
public class ArrayListDemo1
{	public static void main(String args[])
	{	ArrayList<String> al=new ArrayList<String>();
		al.add("AshokChawan");
		al.add("ChanduBorde");
		al.add("KiranMorey");
		out.println("Elements in an array");
		Iterator<String> io=al.iterator();
		while(io.hasNext())
```

```
                {       String ele=io.next();
                        out.println(ele);
                }
        }
}
```

ArrayListDemo2.java

```
import static java.lang.System.*;
import java.util.*;
class Emp
{       int eno;
        String ename;
        Emp(int eno,String ename)
        {       this.eno=eno;
                this.ename=ename;
        }
        public void display()
        {       out.println("Employee Details");
                out.println("Eno:\t"+eno);
                out.println("Ename:\t"+ename);
        }
}
public class ArrayListDemo2
{       public static void main(String args[])
        {       Emp e1=new Emp(101,"Raju");
                ArrayList<Object> al=new ArrayList<Object>();
                al.add("AshokChawan");
                al.add(e1);
                al.add(100);
                out.println("Elements in an array");
                for(Object ele:al)
                {       if(ele instanceof Emp)
                        {((Emp)ele).display();
                        }else
                        out.println(ele);
                }
        }
}
```

ArrayListDemo3.java

```
import static java.lang.System.*;
import java.util.*;
public class ArrayListDemo3
{       public static void main(String args[])
        {       Emp e1=new Emp(101,"Raju");
                ArrayList<Object> al=new ArrayList<Object>();
```

```
                al.add("AshokChawan");
                al.add(e1);
                al.add(100);
                Iterator io=al.iterator();
                out.println("Elements emp is removed");
                while(io.hasNext())
                {       Object ele=io.next();
                        if(ele instanceof Emp)
                        io.remove();
                }
                out.println("Elements Are.....");
                for(Object ele:al)
                out.println(ele);
        }
}
```

ArrayListDemo4.java

```
import static java.lang.System.*;
import java.util.*;
public class ArrayListDemo4
{       public static void main(String args[])
        {       ArrayList<String> al=new ArrayList<String>();
                al.add("SaiSri");
                al.add("AreSiva");
                ArrayList<String> al2=new ArrayList<String>();
                al2.add("Matladamak");
                al2.add("Ninnukuda....are...KK");
                al.addAll(al2);
                out.println("Elements in an Array...");
                for(String ele:al)
                out.println(ele);
        }
}
```

ArrayListDemo5.java

```
import static java.lang.System.*;
import java.util.*;
public class ArrayListDemo5
{       public static void main(String args[])
        {       ArrayList<String> al=new ArrayList<String>();
                al.add("SaiSri");
                al.add("NaaperuSiva");
                al.add("AreAshok");
                ArrayList<String> al2=new ArrayList<String>();
                al2.add("SaiSri");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                al2.add("AreAshok");
                al.removeAll(al2);
                out.println("Elements in an Array...");
                for(String ele:al)
                out.println(ele);
        }
}
```

ArrayListDemo6.java

```
import static java.lang.System.*;
import java.util.*;
public class ArrayListDemo6
{       public static void main(String args[])
        {       ArrayList<String> al=new ArrayList<String>();
                al.add("SaiSri");
                al.add("NaaperuSiva");
                al.add("AreAshok");
                ArrayList<String> al2=new ArrayList<String>();
                al2.add("SaiSri");
                al2.add("AreAshok");
                al.retainAll(al2);
                out.println("Elements in an Array...");
                for(String ele:al)
                out.println(ele);
        }
}
```

ArrayListDemo7.java

```
import static java.lang.System.*;
import java.util.*;
public class ArrayListDemo
{       public static void main(String arg[])
        {       ArrayList<String> al=new ArrayList<String>();
                al.add("GiriBabu");
                al.add("Sobha");
                al.add("Madhu");
                //set method is used to replace an element of  specified index
                /*al.set(2,"Shekar");
                al.set(0,"Narayan");
                al.set(1,"KrishnaVeni");*/
                //add method adds an element in specified index,  and moves the existed element to
                //the next index
                al.add(2,"Shekar");
                al.add(0,"Narayan");
                al.add(1,"KrishnaVeni");
                out.println("Elements in an ArrayList...");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
            for(String name:al)
            out.println(name);
        }
}
```

Vector Class:

- Vector came along with first version of java but ArrayList was introduced in java version 1.2
- All the methods of Vector is synchronized but the ArrayList methods are not synchronized.
- Vector and ArrayList both uses Array internally as data structure.
- They are dynamically resizable
- By default Vector doubles the size of its array when its size is increased.
- An application can increase the capacity of a vector before inserting a large number of components; this reduces the amount of incremental reallocation.

VectorDemo.java

```
import java.util.*;
import static java.lang.System.*;
class Emp
{       int eno;
        String name;
        Emp(int eno,String name)
        {this.eno=eno;this.name=name;
        }
        void getDetails()
        {       out.println("Eno:\t"+eno);
                out.println("Ename:\t"+name);
        }
}
public class VectorDemo
{       public static void main(String args[])
        {       Vector<Object> v=new Vector<Object>();
                Emp e=new Emp(101,"Bhuvan");
                byte b=20;
                Byte b1=new Byte(b);
                v.add(10);
                v.add(10.50f);
                v.add("Shekar");
                v.add(b1);
                v.add(b1);
                v.add(e);
                v.add(new Emp(102,"Prathyusha"));
                v.add(0,"Zero ");
                out.println("Size of the Vector:\t"+v.size());
                Object o=v.get(1);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println(o);
                out.println("default capacity of v :\t"+v.capacity());
                v.add(3," Manisha");
                v.remove(e);
                out.println("deleted object is:\t"+e);
                out.println("\nEnumeration object....\n");
                Enumeration totele=v.elements();
                while(totele.hasMoreElements())
                {       o=totele.nextElement();
                        if(o instanceof Emp)
                        ((Emp)o).getDetails();
                        else
                        out.println(o);
                }
                Iterator it=v.iterator();
                out.println("\nIterator\n");
                while(it.hasNext())
                {out.println(it.next());
                }
        }
}
```

Stack Class: A stack represents a group of elements stored in LIFO(Last In First Out). This means that the element which is stored as a last element into the stack will be the first element to be removed from the stack. Inserting elements (objects) into the stack is called 'push operation' and removing elements from stack is called 'pop operation'. Searching for an element in the stack is called 'peep operation'. Insertion and deletion of elements take place only from one side of the stack, called 'top' of the stack.

Stack Class Methods

- boolean empty(): This method tests whether the stack is empty or not.
- Object Peek(): this method returns the top-most object from the stack without removing it.
- Object pop(): this method pops the top-most element from the stack ant returns it.
- Object push(Object): this method pushes an element(object) on to the top of the stack and returns that element.
- int **search(object)**: this method returns the position of an element from the top of the stack. If the element(object) is not found in the stack then it returns –ve value

Now I am creating a stack that can store Integer type objects

Stack<Integer> st=new Stack<Integer>();

if u want to store integer elements since int is a primitive data type, we should convert the int values into Integer objects and then store into the stack for this purpose, we can write a statement as:

st.push(10);

here we pass int element to push() method. But the method automatically converts it into **Integer** Object and then stores it into the stack st. This is called **'auto boxing'**

StackDemo.java

```
import java.io.*;
import java.util.*;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
import static java.lang.System.*;
class StackDemo
{	public static void main(String args[])throws Exception
	{	Stack<Object>  st1=new Stack<Object>();
		int ch=0;
		int position,element;
		BufferedReader br=new BufferedReader(new InputStreamReader(in));
		while(ch<5)
		{	out.println("Stack Operation");
			out.println("1 push");
			out.println("2 pop");
			out.println("3 search");
			out.println("4 View");
			out.println("5 exit");
			out.println("Choice ..!");
			ch=Integer.parseInt(br.readLine());
			switch(ch)
			{	case 1:  out.println("Enter element");
					element=Integer.parseInt(br.readLine());
					st1.push(element);
					break;
				case 2:
					Integer ele=(Integer)st1.pop();
					out.println("popped element is:\t"+ele);
					break;
				case 3:
					out.println("which element do u wonna search?");
					Integer el=new Integer(br.readLine());
					position=st1.search(el);
					if(position!=-1)
					out.println("Element found at :\t"+position);
					else
					out.println("Element Not found");
					break;
				case 4:
					out.println(st1);
					break;
				default:
					return;
			}
		}
	}
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Linked List:

- LinkedList class uses doubly linked list to store elements and it extends AbstractSequentialList class and implements List,Deque,Cloneable interfaces.
- It can contains duplicate elements
- Maintain Insertion Order
- It is not synchronized
- No random access
- Manipulation Fast because no shifting needs to be occurred
- It can be used as list,stack, or queue

Linked list includes many convenient methods to manipulate the elements stored. Apart from the methods of super class List... it adds its own methods also of which few are illustrated here

1. Void addFirst(Object obj)
2. Void addLast(Object obj);
3. Object getFirst();
4. Object getLast();
5. Object removeFirst();
6. Object removeLast();

LinkedListDemo.java

```
import java.util.*;
import static java.lang.System.*;
class Emp
{       int eno;
        String name;
        Emp(int eno,String name)
        {this.eno=eno;this.name=name;
        }
        public String toString()
        {return getClass().getName()+"@[Eno="+eno+",Ename="+name+"]";
        }
}
public class LinkedListDemo
{
        public static void main(String args[])
        {       LinkedList<Object> ll=new LinkedList<Object>();
                byte b=40;
                Emp e=new Emp(101,"Priya");
                ll.add(e);
                ll.add(null);
                ll.add(new Character('c'));
                ll.add(new Byte(b));
                ll.add(new Long(2345678));
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
            ll.add(new Integer(345));
//          out.println("First:\n"+ll.getFirst());
//          out.println("\nLast:\t"+ll.getLast());

            //out.println("\n Two Items are Entered");
            ll.addFirst(new Float(234.56f));
            ll.addLast(new Double(123.45));
            //out.println("\nFirst:\t"+ll.getFirst());
            //out.println("\nLast:\t"+ll.getLast());
            //out.println("Is Emp object contains:\t"+ll.contains(e));
            ListIterator itr=ll.listIterator(3);//4 is a starting index number
            while(itr.hasNext())
            {out.println(itr.next());
            }
        }
}
```

List Interface

List interface is the sub interface of Collection. It contains methods to insert and delete elements in index basis it is a factory of ListIterator interface.

Commnly used method of List Interface

- Public void add(int index, Object element);
- Public Boolean addAll(int index,Collection c);
- Public Object get(int indexPosition);
- Public Object set(int index,Object element);
- Public Object remove(int index);
- Public ListIterator listIterator();
- Public ListIterator listIterator(int i);

ListIterator:

It is used to traverse the elements in backward and forward direction. ListIterator is a sub class of Iterator interface

Commnly used method of ListIterator Interface

- public abstract boolean hasNext();
- public abstract java.lang.Object next();
- public abstract boolean hasPrevious();
- public abstract java.lang.Object previous();
- public abstract int nextIndex();
- public abstract int previousIndex();
- public abstract void remove();
- public abstract void set(java.lang.Object);
- public abstract void add(java.lang.Object);

Example: ListIteratorDemo.java

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
import static java.lang.System.*;
import java.util.*;
public class ListIteratorDemo
{       public static void main(String args[])
        {       ArrayList<String> al=new ArrayList<String>();
                al.add("Priya");
                al.add("Madhu");
                al.add("Giri");
                al.add("Shekar");
                al.add("Sobha");
                ListIterator li=al.listIterator();
                out.println("Traversing Elements in Forward Direction");
                while(li.hasNext())
                {out.println(li.next());
                }
                out.println("Traversing Elements in Backward Direction");
                while(li.hasPrevious())
                {out.println(li.previous());
                }
        }
}
```

Difference Between List and Set

List contains duplicate elements but those elements are in insertion order, whereas set contains unique elements only and doesn't guarantee that the order will remain constant.

HashSet class

- Uses the hashtable to store the elements. It extends the AbstractSet class and implements Set interface.
- It contains unique element only
- It permits null values
- It is Unsynchronized
- There is no guarantee that the order will remain constant.

Example: HashSetDemo.java

```
import java.util.*;
import static java.lang.System.*;
class HashSetDemo
{       public static void main(String ar[])
        {       HashSet<String> hs=new HashSet<String>();
                hs.add("one");
                hs.add("Two");
                hs.add(null);
                hs.add("one");
```

```
                hs.add("one");
                hs.add("one");
                hs.add("one");
                out.println(hs.size());
                Iterator it=hs.iterator();
                while(it.hasNext())
                {out.println(it.next());
                }
        }
}
```

How we can make HashSet as synchronized?

In Collections class we have a method called synchronizedSet, that takes the hashset object and makes it synchronized and then return it to us.

Example To Make HashSet To Synchronized

```
import java.util.HashSet;
import java.util.Collections;
import java.util.Set;
public class HashSetToSynchronized
{       public static void main(String[] args)
        {       HashSet hashSet = new HashSet();
                Set set = Collections.synchronizedSet(hashSet);
        }
}
```

LinkedHashSet class

- It doesn't allow duplicate elements
- It extends HashSet class and implements Set interface
- Maintains insertion order
- It is not synchronized

Example: LinkedHashSetDemo.java

```
import java.util.*;
import static java.lang.System.*;
class LinkedHashSetDemo
{       public static void main(String ar[])
        {       LinkedHashSet<String> hs=new LinkedHashSet<String>();
                hs.add("one");
                hs.add("Two");
                hs.add(null);
                hs.add("one");
                hs.add("one");
                hs.add("one");
                hs.add("one");
                out.println(hs.size());
                Iterator it=hs.iterator();
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                while(it.hasNext())
                {out.println(it.next());
                }
        }
}
```

Tree Set class

The TreeSet class implements NavigableSet interface that extends the SortedSet interface

- It will not allow duplicat values
- Maintains ascending order
- It doesnot permits null values
- It is Unsynchronized
- The elements which are added to the TreeSet class must implement Comparable interface

TreeSetDemo.java

```
import java.util.*;
import static java.lang.System.*;
class TreeSetDemo
{       public static void main(String ar[])
        {TreeSet<Object> ts=new TreeSet<Object>();
        ts.add("Jyothi");
        ts.add("Gayathri");
        ts.add("VN Reddy");
        ts.add("Kumar");
        ts.add("Raju");
        Iterator it=ts.iterator();
        while(it.hasNext())
        {out.println(it.next());
        }
        //Starting from "Jyothi" to before "Raju"
        SortedSet ss=ts.subSet("Jyothi","Raju");
        out.println(ss);
        ss=ts.tailSet("Jyothi");
        out.println(ss);
        ss=ts.headSet("Jyothi");
        out.println(ss);
        }
}
```

TreeSetDemo2.java

```
import java.util.*;
import static java.lang.System.*;
class Emp implements Comparable<Emp>
{       int eno;
        String ename;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        Emp(int eno,String ename)
        {this.eno=eno;
        this.ename=ename;
        }
        public String toString()
        {return getClass().getName()+"@[Eno="+eno+",Ename="+ename+"]";
        }
        public  int compareTo(Emp e2)
        {       //return this.eno-e.eno;
                return this.ename.compareTo(e2.ename);
        }
}
class TreeSetDemo2
{       public static void main(String ar[])
        {       TreeSet<Object> ts=new TreeSet<Object>();
                ts.add(new Emp(111,"Madhu"));
                ts.add(new Emp(110,"Shekar"));
                ts.add(new Emp(102,"Giri"));
                Iterator it=ts.iterator();
                while(it.hasNext())
                {out.println(it.next());
                }
        }
}
```

Queue Interface

The Queue interface basically orders the elements in FIFO (First in First Out) manner, which means first element is removed first and last element is removed at last.

Mehtods of Queue Interface

- Public boolean add(Object o): adds an element to the queue, it returns true if it is success, if no space is available(if the queue is created with specified size) it throws **IllegalStateException**
- Public boolean offer(Object o): adds an element to the queue, it returns true if it is success(we use this method if the queue is created without specific size)
- Public Object remove(): It retrieves and removes head of the queue element. If no element is found it throws **NoSuchElementException.**
- Public Object poll():It retrieves and removes head of the queue element if no element is found it return **null**.
- Public Object element(): It retrieves head element of the queue, if queue is empty it throws **NoSuchElementException**
- Public Object peek():It retrieves head element of the queue if queue is empty it returns **null**

PriorityQueue class

- It provides the facility of using queue. but doesnot orders the elements in fifo manner
- There is no guarantee that the order will remain constant.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- It doesn't permit null values.
- The elements in the queue must be comparable type
- String and wrapper classes are comparable by default. To add user-defined objects in priorityqueue, you need to implement comparable interface.

PriorityQueueDemo1.java

```
import static java.lang.System.*;
import java.util.*;
class PriorityQueueDemo4
{	public static void main(String args[])
	{
		PriorityQueue<String> q=new PriorityQueue<String>();
		//to add element
		q.add("abc");//IllegalStateException
		q.offer("abc");//it returns false
		//to remove head element
		out.println("Removed element:\t"+q.remove());
		out.println("Removed Element:\t"+q.poll());
		out.println("Removed Element:\t"+q.poll());//it returns null
		out.println("Removed Element:\t"+q.remove());//NoSuchElementException

		//to see the head element
		out.println(q.peek());//it returns null
		out.println(q.element());//it returns NoSuchElementException
	}
}
```

Note: we see the natural ordering while performing operations on PriorityQuer not when we display elements by using Iterator object.

Java Deque Interface

Java Deque Interface is a linear collection that supports element insertion and removal at both ends. Deque is an acronym for **"double ended queue".**

Methods of Java Deque Interface

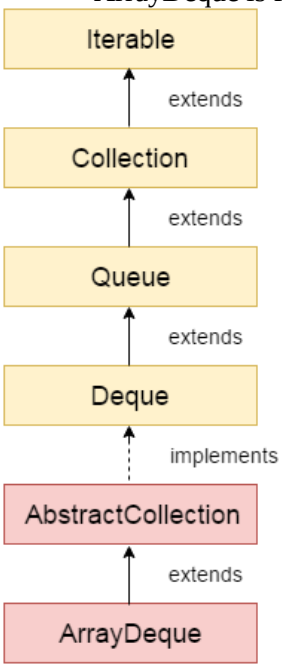
Method	Description
boolean add(object)	It is used to insert the specified element into this deque and return true upon success.
boolean offer(object)	It is used to insert the specified element into this deque.
Object remove()	It is used to retrieves and removes the head of this deque.
Object poll()	It is used to retrieves and removes the head of this deque, or returns null if this deque is empty.
Object element()	It is used to retrieves, but does not remove, the head of this deque.
Object peek()	It is used to retrieves, but does not remove, the head of this deque, or returns null if this deque is empty.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

ArrayDeque class

The ArrayDeque class provides the facility of using deque and resizable-array. It inherits AbstractCollection class and implements the Deque interface.
The important points about ArrayDeque class are:

- Unlike Queue, we can add or remove elements from both sides.
- Null elements are not allowed in the ArrayDeque.
- ArrayDeque is not thread safe, in the absence of external synchronization.
- ArrayDeque has no capacity restrictions.
- ArrayDeque is faster than LinkedList and Stack.



Example

```
import java.util.*;
public class DequeExample {
public static void main(String[] args) {
    Deque<String> deque=new ArrayDeque<String>();
    deque.offer("arvind");
    deque.offer("vimal");
    deque.add("mukul");
    deque.offerFirst("jai");
    System.out.println("After offerFirst Traversal...");
    for(String s:deque){
        System.out.println(s);
    }
    //deque.poll();
    //deque.pollFirst();//it is same as poll()
    deque.pollLast();
    System.out.println("After pollLast() Traversal...");
    for(String s:deque){
        System.out.println(s);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
    }
  }
  }
```

Map

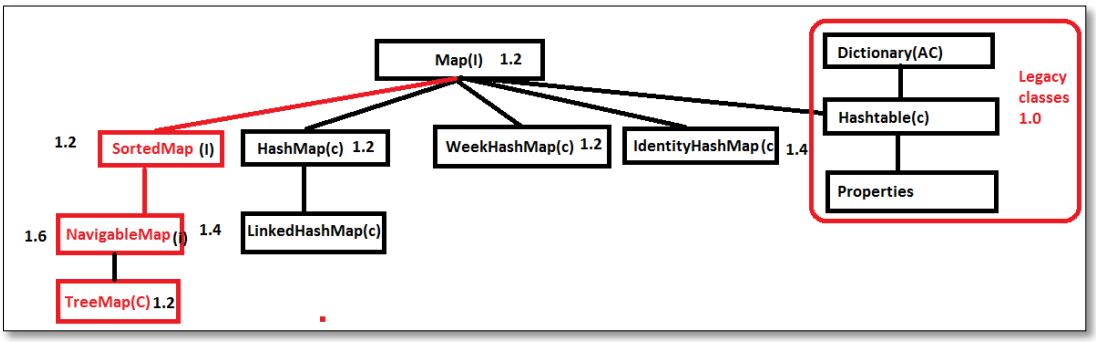
A Map is an object that maps keys to values. It allows you to store pairs of elements, termed "keys" and "values" where each key maps to one value. Thus the keys in a map must be unique.The below methods return objects which allow you to traverse the elements of the Map, and also delete elements from the Map.

Method Name	Description
entrySet()	It returns a set which contains collection Map.Entry class objects and each Entry class object contains key value pairs, we can access them with getKey() and getValue() methods of Entry class and also we have a setValue() method which is used to set the value to a particular key.
keySet()	It returns Set object which contains set of keys of the map
values()	It returns a Collection object which contains set of values of the map
put(Object key,Object value)	It is used to insert an entry in this map

What is an Entry class?

It is the sub class of Map interface, it has the following method.

MethodName	Description
getKey()	Returns the key
getValue()	Returns the value
setValue()	Used to set new value for the already existed one.



HashMap

1. To store key value pairs
2. It is not synchronized
3. It accepts only one null key
4. It also any number of null values
5. There is no guarantee about the order

HashMapDemo1.java

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
import static java.lang.System.*;
import java.util.*;
class Emp
{	int eno;
	String ename;
	Emp(int eno,String ename)
	{this.eno=eno;
	this.ename=ename;
	}
	public int getEno()
	{return eno;
	}
	public String getEname()
	{return ename;
	}
	@Override
	public String toString()
	{return getClass().getName()+"@[Eno="+eno+",Ename="+ename+"]";
	}
}
class HashMapDemo
{	public static void main(String args[])
	{	Emp e1=new Emp(101,"Jyostna");
		Emp e2=new Emp(102,"Jyothi");
		Emp e3=new Emp(103,"Poojitha");
		Emp e4=new Emp(104,"RamaKanth");
		Emp e5=new Emp(105,"Baby");
		Emp e11=new Emp(101,"Jyostna2");
		HashMap<Integer,Emp> hm=new HashMap<Integer,Emp>();
		hm.put(e1.getEno(),e1);
		hm.put(e2.getEno(),e2);
		hm.put(e3.getEno(),e3);
		hm.put(e4.getEno(),e4);
		hm.put(e5.getEno(),e5);
		hm.put(e11.getEno(),e11);
		Emp e=hm.get(101);
		out.println(e);
		out.println(hm.size());
	}
}
```

HashMapDemo2.java

```
import static java.lang.System.*;
import java.util.*;
class Emp
{	int eno;
	String ename;
	Emp(int eno,String ename)
	{	this.eno=eno;
		this.ename=ename;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
	}
	public int getEno()
	{return eno;
	}
	public String getEname()
	{return ename;
	}
	@Override
	public String toString()
	{return getClass().getName()+"@[Eno="+eno+",Ename="+ename+"]";
	}
}
class HashMapDemo2
{
	public static void main(String args[])
	{	Emp e1=new Emp(101,"Jyostna");
		Emp e2=new Emp(102,"Jyothi");
		Emp e3=new Emp(103,"Poojitha");
		Emp e4=new Emp(104,"RamaKanth");
		Emp e5=new Emp(105,"Baby");
		Emp e11=new Emp(101,"Jyostna2");
		HashMap<Integer,Emp> hm=new HashMap<Integer,Emp>();
		hm.put(e1.getEno(),e1);
		hm.put(e2.getEno(),e2);
		hm.put(e3.getEno(),e3);
		hm.put(e4.getEno(),e4);
		hm.put(e5.getEno(),e5);
		hm.put(e11.getEno(),e11);
		Emp e=hm.get(101);
		out.println(e);
		out.println(hm.size());
		//get all values(emp objects) as collection
		Collection<Emp> c=hm.values();
		out.println("Elements in a collection...");
		for(Emp ele:c)
		{out.println(ele);
		}
		//get all the keys(int values) as Set
		Set<Integer> keys=hm.keySet();
		out.println("Keys in a collection...");
		for(Integer k:keys)
		{out.println(k);
		}
		Set<Map.Entry<Integer,Emp>> kvSet=hm.entrySet();
		for(Map.Entry<Integer,Emp> entry:kvSet)
		{	out.println(entry.getKey()+"\t"+entry.getValue());
			if(entry.getKey()==105)
			entry.setValue(new Emp(105,"Baby Kiranmayee"));
		}
		out.println(hm);
	}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
}
```

What is the difference between *HashMap* and *HashTable*?

(The **HashMap** class is roughly equivalent to **Hashtable**, except that it is unsynchronized and permit nulls as keys.) This class makes no guarantees as to the order of the map; in particular, it does not guarantee that the order will remain constant over a time.

What is HashTable?

- **It is the child class of Dictionary class.**
- **Introduced in 1.0 version (Legacy class).**
- **It is synchronized.**
- **It stores the data as key and value pairs.**
- **It doesn't accept null values as keys or values**

HashTableApp.java

```
import static java.lang.System.*;
import java.util.*;
class Emp
{	int eno;
	String ename;
	Emp(int eno,String ename)
	{	this.eno=eno;
		this.ename=ename;
	}
	public int getEno()
	{return eno;
	}
	public String getEname()
	{return ename;
	}
	@Override
	public String toString()
	{return getClass().getName()+"@[Eno="+eno+",Ename="+ename+"]";
	}
}
class HashTableDemo
{	public static void main(String args[])
	{	Emp e1=new Emp(101,"Jyostna");
		Emp e2=new Emp(102,"Jyothi");
		Emp e3=new Emp(103,"Poojitha");
		Emp e4=new Emp(104,"RamaKanth");
		Emp e5=new Emp(105,"Baby");
		Hashtable<Integer,Emp> hm=new Hashtable<Integer,Emp>();
		hm.put(e1.getEno(),e1);
		hm.put(e2.getEno(),e2);
		hm.put(e3.getEno(),e3);
		hm.put(e4.getEno(),e4);
		hm.put(e5.getEno(),e5);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println(hm.get(101));
                out.println(hm.size());
                //get all values(emp objects) as collection
                Collection<Emp> c=hm.values();
                out.println("Elements in a collection...");
                for(Emp ele:c)
                {out.println(ele);
                }
                //get all the keys(int values) as Set
                Set<Integer> keys=hm.keySet();
                out.println("Keys in a collection...");
                for(Integer k:keys)
                {out.println(k);
                }
                Set<Map.Entry<Integer,Emp>> kvSet=hm.entrySet();
                for(Map.Entry<Integer,Emp> entry:kvSet)
                {       out.println(entry.getKey()+"\t"+entry.getValue());
                        if(entry.getKey()==105)
                        entry.setValue(new Emp(105,"Baby Kiranmayee"));
                }
                //out.println(hm);
        }
}
```

There are several differences between HashMap and Hashtable in Java:

HashMap	Hashtable
Not Synchronized	Synchronized
It is better for non-threaded applications as unsynchronized objects typically perform better than synchronized onces	It is not better for non-threaded applications
It accepts one null key and any number of null values.	It doesn't accepts null keys or values

LinkedHashMap:

- It is the implementation of Hashtable and LinkedList in a Map.
- It maintains insertion order(based on the keys)
- It maintains a doubly-linked list for iteration
- Insertion order is not affected if a key is *re-inserted* into the map
- It is not synchronized.
- It doesn't accept null keys.
- It accepts null values.

LinkedHashMap.java

```
import static java.lang.System.*;
import java.util.*;
class Emp
{       int eno;
```

```
        String ename;
        Emp(int eno,String ename)
        {       this.eno=eno;
                this.ename=ename;
        }
        public int getEno()
        {return eno;
        }
        public String getEname()
        {return ename;
        }
        @Override
        public String toString()
        {return getClass().getName()+"@[Eno="+eno+",Ename="+ename+"]";
        }
}
class LinkedHashMapDemo
{       public static void main(String args[])
        {       Emp e1=new Emp(101,"Jyostna");
                Emp e2=new Emp(102,"Jyothi");
                Emp e3=new Emp(103,"Poojitha");
                Emp e4=new Emp(104,"RamaKanth");
                Emp e5=new Emp(105,"Baby");
                LinkedHashMap<Integer,Emp> hm=new LinkedHashMap<Integer,Emp>();
                hm.put(e5.getEno(),e5);
                hm.put(e1.getEno(),e1);
                hm.put(e4.getEno(),e4);
                hm.put(e2.getEno(),e2);
                hm.put(e3.getEno(),e3);
                hm.put(101,null);
                hm.put(102,null);
                Set<Map.Entry<Integer,Emp>> kvSet=hm.entrySet();
                for(Map.Entry<Integer,Emp> entry:kvSet)
                {       out.println(entry.getKey()+"\t"+entry.getValue());
                        if(entry.getKey()==105)
                        entry.setValue(new Emp(105,"Baby Kiranmayee"));
                }
        }
}
```

TreeMapDemo.java

- A TreeMap contains values based on the key. It implements the NavigableMap interface and extends AbstractMap class.
- It contains only unique elements.
- It cannot have null key but can have multiple null values.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- It is same as HashMap instead maintains ascending order.
- It is unsynchronized.

TreeMapDemo.java

```
import static java.lang.System.*;
import java.util.*;
class Emp
{	int eno;
	String ename;
	Emp(int eno,String ename)
	{	this.eno=eno;
		this.ename=ename;
	}
	public int getEno()
	{return eno;
	}
	public String getEname()
	{return ename;
	}
	@Override
	public String toString()
	{return getClass().getName()+"@[Eno="+eno+",Ename="+ename+"]";
	}
}
class TreeMapDemo
{	public static void main(String args[])
	{	Emp e1=new Emp(101,"Jyostna");
		Emp e2=new Emp(102,"Jyothi");
		Emp e3=new Emp(103,"Poojitha");
		Emp e4=new Emp(104,"RamaKanth");
		Emp e5=new Emp(105,"Baby");
		TreeMap<Integer,Emp> hm=new TreeMap<Integer,Emp>();
		hm.put(e5.getEno(),e5);
		hm.put(e1.getEno(),e1);
		hm.put(e4.getEno(),e4);
		hm.put(e2.getEno(),e2);
		hm.put(e3.getEno(),e3);
		hm.put(101,null);
		hm.put(102,null);
		Set<Map.Entry<Integer,Emp>> kvSet=hm.entrySet();
		for(Map.Entry<Integer,Emp> entry:kvSet)
		{	out.println(entry.getKey()+"\t"+entry.getValue());
			if(entry.getKey()==105)
			entry.setValue(new Emp(105,"Baby Kiranmayee"));
		}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        }
}
```

What are Java Comparators and Comparables?

These are used to compare objects in Java. Using these concepts; we can sort Java objects in an order.

Comparable

A comparable object is used to compare itself with another object. The class itself must implement the **java.lang.Comparable** interface in order to compare its instances.

Comparator

A comparator object is used to compare two different objects. This class will not compare its instances, but some other class's instances. This comparator class must implement the **java.util.Comparator** interface.

How to use these?

There are two interfaces in Java to support these concepts, and each of these has one method to be implemented by user. Those are.

Methods of java.lang.Comparable:

Method Name	Description
int compareTo(Object o1)	This method compares this object with o1 object and then returns an int value this value has the following meanings. • positive – this object is greater than o1 • zero – this object equals to o1 • negative – this object is less than o1

Methods of java.util.Comparator

Method Name	Description
int compare(Object o1, Object o2)	This method compares o1 and o2 objects and then returns an int value this value has the following meanings. • positive – o1 is greater than o2 • zero – o1 equals to o2 • negative – o1 is less than o2

Sorting a List:

We can sort a list or List of objects in two ways

Ordering	Description
Natural ordering:	java.util.Collections.sort(List) and java.util.Arrays.sort(Object[]) methods can be used to sort using natural ordering of objects.
Ordering by using Comparator object	java.util.Collections.sort(List, Comparator) and java.util.Arrays.sort(Object[], Comparator) methods can be used if a Comparator is available for comparison.

First we'll write a simple Java bean which implements Comparable interface **to represent the Employee.**

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

What will happen if you don't implement Comparable interface?

If you don't implement the Comparable interface, the sort() method of Collections interface will not compare/sort the list of employee objects.

NaturalOrdering.java

```
import static java.lang.System.*;
import java.util.*;
class Emp implements Comparable<Emp>
{	int eno;
	String ename;
	Emp(int eno,String ename)
	{	this.eno=eno;
		this.ename=ename;
	}
	public int getEno()
	{return eno;
	}
	public String getEname()
	{return ename;
	}
	@Override
	public String toString()
	{return getClass().getName()+"@[Eno="+eno+",Ename="+ename+"]";
	}
	@Override
	public int compareTo(Emp e2)
	{	return this.eno-e2.eno;
	}
}
class NaturalOrdering
{
	public static void main(String args[])
	{	Emp e1=new Emp(101,"Jyostna");
		Emp e2=new Emp(102,"Jyothi");
		Emp e3=new Emp(103,"Poojitha");
		Emp e4=new Emp(104,"RamaKanth");
		Emp e5=new Emp(105,"Baby");
		ArrayList<Emp> al=new ArrayList<Emp>();
		al.add(e5);
		al.add(e1);
		al.add(e4);
		al.add(e2);
		al.add(e3);
		Collections.sort(al);
		out.println(al);
	}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
}
```

ComparatorDemo.java

```
import static java.lang.System.*;
import java.util.*;
class Emp
{	int eno;
	String ename;
	Emp(int eno,String ename)
	{	this.eno=eno;
		this.ename=ename;
	}
	public int getEno()
	{return eno;
	}
	public String getEname()
	{return ename;
	}
	@Override
	public String toString()
	{return getClass().getName()+"@[Eno="+eno+",Ename="+ename+"]";
	}
}
class ByEno implements Comparator<Emp>
{	public int compare(Emp e1,Emp e2)
	{return e1.eno-e2.eno;
	}
}
class ByEname implements Comparator<Emp>
{
	public int compare(Emp e1,Emp e2)
	{return e1.ename.compareTo(e2.ename);
	}
}
class ComparatorDemo
{
	public static void main(String args[])
	{	Emp e1=new Emp(101,"Jyostna");
		Emp e2=new Emp(102,"Jyothi");
		Emp e3=new Emp(103,"Poojitha");
		Emp e4=new Emp(104,"RamaKanth");
		Emp e5=new Emp(105,"Baby");
		ArrayList<Emp> al=new ArrayList<Emp>();
		al.add(e5);
		al.add(e1);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                al.add(e4);
                al.add(e2);
                al.add(e3);
                out.println("Enter your option");
                Scanner s=new Scanner(in);
                out.println("1. Sort By Eno");
                out.println("2. Sort By Ename");
                int opt=s.nextInt();
                switch(opt)
                {
                        case 1:
                                Collections.sort(al,new ByEno());
                                break;
                        case 2:
                                Collections.sort(al,new ByEname());
                                break;
                }
                out.println(al);
        }
}
```

Properties class:

- The Properties class represents a persistent set of properties.
- The Properties can be saved to a stream or loaded from a stream.
- Each key and its corresponding value in the property list is a string.

Note: Because Properties inherits from Hashtable, the put and putAll methods can be applied to a Properties object. Their use is strongly discouraged as they allow the caller to insert entries whose keys or values are not Strings.

PropertiesDemo1.java

```
import java.text.*;
import static java.lang.System.*;
import java.util.*;
public class PropertiesDemo
{       public static void main(String args[])
        {       Properties p=new Properties();
                p.put("user","scott");
                p.put("password","tiger");
                p.list(out);
        }
}
```

Db.properties

```
user=system
password=manager
url=jdbc:oracle:thin:@localhost:1521:ORCL
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
driver=oracle.jdbc.driver.OracleDriver
```

PropertiesDemo2.java

```
import java.text.*;
import static java.lang.System.*;
import java.util.*;
import java.io.*;
public class PropertiesDemo2
{	public static void main(String args[])throws Exception
	{	FileInputStream fis=new FileInputStream("db.properties");
		Properties p=new Properties();
		p.load(fis);
		p.list(out);
	}
}
```

PropertiesDemo3.java

```
import java.util.*;
import static java.lang.System.*;
class PropertiesDemo3
{	 public static void main(String args[])
	{	Properties p=System.getProperties();
		p.list(out);
	}
}
```

Util Package Important Classes

What is StringTokenizer?

The StringTokenizer class is used to break a given strings into tokens (substrings) (words, numbers, operators, or whatever). The tokens are separated by one or more delimiters (characters). **It has been replaced by regular expression tools.** A more powerful solution is to use *regular expressions*, and the easiest way to do that is use the java.util.Scaner class, the String split (..) method, or the Pattern and Matcher classes.

A StringTokenizer constructor takes a string to break into tokens and returns a **StringTokenizer** object for that string. Each time its nextToken() method is called, it returns the next token in that string. If you don't specify the *delimiters* (separator characters), blanks are the default delimiters. Some constructors are given below.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

StringTokenizer *st* = new StringTokenizer(*s*);
Creates a StringTokenizer for the String *s* that uses whitespace (blanks, tabs, newlines, returns, form feeds) as delimiters.
StringTokenizer *st* = new StringTokenizer(*s*, *d*);
Creates a StringTokenizer for the String *s* using delimiters from the String *d*.
StringTokenizer *st* = new StringTokenizer(*s*, *d*, *f*);
Creates a StringTokenizer for the String *s* using delimiters from the String *d*. If the boolean *f* is true, each delimiter character will also be returned as a token.

Methods of StringTokenizer

Assume that st is a StringTokenizer.

- *st*.hasMoreTokens() -- Returns true if there are more tokens.
- *st*.nextToken() -- Returns the next token as a String.
- *st*.countTokens() -- Returns the int number of tokens. This can be used to allocate an array before starting, altho it can be inefficient for long strings because it has to scan the string once just to get this number. Using a Vector and converting it to an array at the end may be a better choice.

StringTokenizerDemo.java

```
import java.util.*;
import static java.lang.System.*;
class StringTokenizerDemo
{       public static void main(String args[])
        {Scanner s=new Scanner(System.in);
        out.println("Enter any String");
        String str=s.nextLine();
        StringTokenizer st=new StringTokenizer(str,",. ");
        out.println(st.countTokens());
        while(st.hasMoreTokens())
        {out.println(st.nextToken());
        }
        }
}
```

DateDemo1.java

```
import java.util.*;
import static java.lang.System.*;
class DateDemo1
{       public static void main(String args[])
        {Date d=new Date();
        out.println("General Format of Date:"+d);
        String
days[]={"","Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"};
        String
mon[]={"Jan","Feb","Mar","Apr","May","Jun","July","Aug","Sep","Oct","Nov","Dec"};
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        GregorianCalendar gc=new GregorianCalendar();
        out.println("ToDay is:\t"+days[gc.get(Calendar.DAY_OF_WEEK)]);
        out.println(mon[gc.get(Calendar.MONTH)]+"-"+""+gc.get(Calendar.DAY_OF_MONTH)+"-
"+gc.get(Calendar.YEAR));
        }
}
```

See the below table which describes the Calendar class and values returned by the get() method of Gregorian Calendar class

Field(of Calendar class)	Possible values returned by get() of GregorianCalendar
AM_PM	0 or 1
DAY_OF_WEEK	Sunday, Monday etc…to sat 1 to 7
DAY_OF_YEAR	1 to 366
MONTH	0 to 11
DAY_OF_MONTH	1 TO 31
WEEK_OF_MONTH	1 TO 6
WEEK_OF_YEAR	1 to 54
HOUR_OF_DAY	0 to 23
HOUR	1 to 12
MINUTE	0 to 59
SECOND	0 to 59
MILLISECOND	0 to 999
YEAR	CURRENT YEAR

SimpleDateFormatDemo.java

```
import java.text.*;
import static java.lang.System.*;
import java.util.*;
public class SDFDemo
{
        public static void main(String args[])
        {       Date d=new Date();
                SimpleDateFormat sdf=new SimpleDateFormat("dd/MMMMM/yyyy");
                out.println(sdf.format(d));
                sdf=new SimpleDateFormat("hh:mm:ss");
                out.println(sdf.format(d));
        }
}
```

SEE THE BELOW TABLE WHICH DESCRIBES DEFFERENT FORMATS

Letter	Date/Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800

Lang Package IMP Programs

Runtime class:

1. It contains methods which are used to run another processes in our program
2. Every Java application has a single instance of class Runtime
3. To create Runtime class object we have to call the getRuntime() method of Runtime class itself.

RuntimeDemo1.java

```
import static java.lang.System.*;
import java.util.*;
import java.io.*;
class RuntimeDemo
{	public static void main(String arg[])throws Exception
	{	Runtime r=Runtime.getRuntime();
		Process p=r.exec("C:\\Windows\\notepad.exe");
		//it makes the mainthread to wait until the process p ends
		p.waitFor();
		out.println("End of main");
	}
}
```

RuntimeDemo2.java

```
import static java.lang.System.*;
import java.util.*;
import java.io.*;
public class RuntimeDemo2
{	static boolean flag;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
	public static void write(InputStream is)throws Exception
	{	int r=0;
		while((r=is.read())!=-1)
		{	out.print((char)r);
			flag=true;
		}
	}
	public static void main(String arg[])throws Exception
	{	out.println("Enter FileName to Compile and Run");
		Scanner s=new Scanner(in);
		String fname=s.next();
		Runtime r=Runtime.getRuntime();
		Process p=r.exec("javac "+fname);
		InputStream errSt=p.getErrorStream();
		write(errSt);
		if(flag==true)
		out.println("Error Unnai");
		else
		{out.println("Error Levu....");
		int i=fname.indexOf(".java");
		String fnwe=fname.substring(0,i);
		Process p2=r.exec("java "+fnwe);
		write(p2.getErrorStream());
		write(p2.getInputStream());
		}
	}
}
```

RuntimeDemo3.java

```
import static java.lang.System.*;
import java.util.*;
import java.io.*;
public class RuntimeDemo3
{	public static void main(String arg[])throws Exception
	{	Scanner s=new Scanner(in);
		out.println("Enter .html file(with extension) to run in browser");
		String fname=s.next();
		Runtime r=Runtime.getRuntime();
		Process p=r.exec("cmd  /c "+fname);
	}
}
```

RuntimeDemo4.java

```
import static java.lang.System.*;
import java.util.*;
import java.io.*;
public class RuntimeDemo4
{	public static void main(String arg[])throws Exception
	{	Runtime r=Runtime.getRuntime();
		out.println("Total Memory\nIn Bytes:\t"+r.totalMemory());
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("In Kilo Bytes:\t"+r.totalMemory()/1024);
                out.println("In Mega Bytes:\t"+(r.totalMemory()/1024)/1024.00f);
                out.println("\nFree Memory");
                out.println("In Mega Bytes:\t"+(r.freeMemory()/1024)/1024.00f);
        }
}
```

What are varargs?

It was introduced in java **1.5(Tiger)** version. It is used when you don't know how many arguments you will pass to a method. Vararg must be the last argument in the method. To use varargs you follow the type of the last parameter by an ellipse(three dots,…), then a space and the parameter name .

Example

```
import static java.lang.System.*;
class College
{
        static void display(int i,String ... arguments)
        {       if(i==1)
                out.println(i+"  Name\n----------------");
                else if(i>1)
                out.println(i+"  Names\n----------------");
                for (String argument : arguments)
                {out.println(argument);
                }
        }
}
class VarArgs
{       public static void main(String args[])
        {       College.display(1,"Abdur");
                College.display(4,"Rehman","Akbar","Mazar","Hafeez");
                College.display(0);
        }
}
```

What is factory method?

Factory method is a method that creates and returns the same class objects. In most cases, we declare the factory methods as static method but we can write instance factory methods.

FactoryDemo.java

```
import static java.lang.System.*;
class College
{       int cid;
        String cname;
        private College()
        {       cid=1;
                cname="MTS";
        }
        private College(int cid,String cname)
        {       this.cid=cid;
                this.cname=cname;
        }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        @Override
        public String toString()
        {return getClass().getName()+"@[Cid="+cid+",Cname="+cname+"]";
        }
        static College getInstance()
        {return new College();
        }
        static College getInstance(int cid,String cname)
        {return new College(cid,cname);
        }
}
class FactoryDemo
{       public static void main(String args[])
        {
                College c1=College.getInstance();
                College c2=College.getInstance(2,"Ramakanth");
                College c3=College.getInstance(2,"Umakanth");
                out.println(c1);
                out.println(c2);
                out.println(c3);
        }
}
```

Nested Classes

Nested classes:

In Java we can write a class with in another class. Such classes are called as Nested classes or Inner classes. See the below syntax:

```
Class OuterClass{
        …….
        Class InnerClass{
                ……
        }
}
```

Q. Can we declare an outer class as private?

No we can't declare an Outer class either as private or protected. If you declare an outer class as private or protected compiler shows error.

Important Points:

- A Nested class is a member of it's enclosing(Outer) class
- We can declare an inner class as public or private or protected or default
- But we can't declare an Outer (non-inner)class as private or protected
- We can declare an outer class only either as public or default
- We can declare an inner class as static
- But we can't declare an outer class as static
- An inner class object contains an additional invisible field 'this$0' that refers to its outer class object.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- When same names are used for instance members of inner and outer classes. then we can refer the outer class instance members in the inner class, as:
  1. <OuterClassname>.this.member;

(Nested classes) are of four categories

- **Non Static Inner classes**
- **Static inner-classes**
- **Local Inner classes**
- **Anonymous inner classes**

Non Static Inner class: An inner class which is declared with a class and it is not declared as static is said to be non static inner class. A Non Static Inner class can access all the members of the outer class directly. But an outer class can't access inner class members directly, it can access only by using the inner class object.

Important points

3. We can't write static methods in non-static inner class.
4. We can declare only constant fields as static i.e we can't declare static variables in non-static inner class.
5. We can create Nonstatic inner class object outside the outer class if it is not declared as private.
6. We have to create and use the outer class object, to create nonstatic inner class object outside the outer class.

What is <OuterClassname>.this?

It is an Outer class pointer(object) which is used to access the outer class instance members in the Inner class even though both Outer and Inner instance member names are same.

InnerDemo1.java

```
import static java.lang.System.*;
class Outer
{       int a,b;
        Outer(int a,int b)
        {this.a=a;
        this.b=b;
        }
        void display()
        {       out.println("Outer display");
                out.println(a+"+"+b+"="+i.add());
                out.println(a+"-"+b+"="+i.sub());
        }
        Inner i=new Inner();
        private class Inner
        {       int c;
                int add()
                {return c=a+b;
                }
                int sub()
                {return c=a-b;
                }
        };
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
};
class InnerDemo1
{	public static void main(String[] args)
	{	Outer o=new Outer(10,20);
		o.display();
	}
}
```

Here in the above example "Inner" class is declared as private so which is accessible only within the outer class. In this way we can hide a class, if you declare same class as default we can use this class anywhere in the same package, if you declare it as protected we can access this inner class in the same package and also outside the package child class, if you declare it as public it can be accessible anywhere.

Can we create an inner class object outside the outer class?

Yes we can create an inner class object (it may be a static inner class or non-static inner class) outside the outer class, If it is not declared as private. If the inner class is private we can't create object outside the outer class.

How do you create an object for non-static inner class object outside the outer class?

We can create an object for non-static inner class object outside the outer class only by using outer class object, so we have to create outer class object first, then we can create inner class object.

```
7.	Outer o=new Outer();
8.	Outer. Inner oi=o.new Inner();
```

InnerDemo2.java

```
import static java.lang.System.*;
class Outer
{	int c=10;
	void cube ()
	{out.println(c+"  cube   "+(c*c*c));
	}
	class Inner
	{	int c;
		void add(int a,int b)
		{	c=a+b;
			out.println(a+"+"+b+"="+(a+b));
		}
		void sub(int a,int b)
		{	c=a-b;
			out.println(a+"-"+b+"="+c);
		}
		void cube()
		{Outer.this.cube();//we are calling the outer cube in inner method
		}
	};
};
class InnerDemo2
{	public static void main(String[] args)
	{	Outer o=new Outer();
		Outer.Inner oi=o.new Inner();
		oi.add(10,20);
		oi.sub(40,10);
		oi.cube();
	}
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

InnerDemo3.java

```
import static java.lang.System.*;
class Outer
{       int a=100;
        static int s=200;
        void display()
        {out.println("Outer display..");
        }
        static void disp()
        {out.println("static method..");
        }
        class Inner
        {       final static int sv=100;
                int a=10000;
                void get()
                {       out.println("Inner a:\t"+a);
                        out.println("Outer a:\t"+Outer.this.a);
                        out.println("Outer s:\t"+s);
                        display();
                        disp();
                }
                void display()
                {out.println("Outer display..");
                Outer.this.display();
                }
        }
}
class InnerClassDemo1
{       public static void main(String args[])
        {       Outer o=new Outer();
                Outer.Inner io=o.new Inner();
                io.get();
        }
}
```

Static Inner classes:

- It is an inner class which is declared as static
- Static inner class can access only static members of the outer class directly
- We can't use <outerclassname>.this pointer int static inner class
- In the static inner class we can define any type of members(fields and methods)
- We can create static inner class object outside the outer class without using outer class object.

Note: we can't use this or super or <outerclassname>.this with in a static method or static block or static inner class.

How do you create an object for non-static inner class outside the outer class?

We can create an object for static inner class outside the outer class directly without using outer class object. Observer the below example.

Outer. Inner oi=new Outer. Inner();

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

StaticInnerDemo.java

```
import static java.lang.System.*;
class Outer
{	static int a=100;
	int b=200;
	void display()
	{	out.println("Outer display");
		out.println("a:\t"+a);
		out.println("b:\t"+b);
	}
	static class Inner
	{	int c=300;
		void display()
		{out.println("static outer inner a:\t"+a);
		//out.println("non-static outer b:\t"+b); inner class can't access b because it is not a static
		out.println("static outer inner c:\t"+c);
		}
	};
};
class StaticInnerDemo
{	public static void main(String[] args)
	{	Outer.Inner oi=new Outer.Inner();
		oi.display();
	}
}
```

Is all the members of a static inner class are static?

All the static Inner class members are not-static members by default; so you have to declare explicitly as static if you want any static members within static inner class.

Local Inner Classes

- It is an inner class written with in a block, method, or a constructor
- It is visible only with in that block
- If it contains in the instance method or instance block It can access any members of the class directly
- If it contains in the static method or static block It can access only static members of the class directly

LocalDemo.java

```
import static java.lang.System.*;
class One
{
	int a=100;
	static int b=200;
	final int c=200;
	void display()
	{	out.println("instance method");
		class Inner
		{	void get()
			{out.println("a:\t"+a);
			 out.println("b:\t"+b);
			 out.println("c:\t"+c);
			}
		}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                new Inner().get();
        }
        static void sdisplay()
        {       out.println("static method");
                class Inner
                {       void get()
                        {out.println("b:\t"+b);
                        }
                }
                new Inner().get();
        }
}
class LocalDemo
{       public static void main(String args[])
        {       One o=new One();
                o.display();
                One.sdisplay();
        }
}
```

Anonymous Inner Class

- It is a class which has no name.
- We can create only one object for an anonymous Inner class.
- Anonymous Inner class implementation and object creation must be in a single statement.
- We can create an anonymous inner class, by using already existed classes or interfaces these may be predefined or used defined.
- We can use anonymous Inner classes in event handling of GUI programming.
- Anonymous Inner classe behaves like a child classe for an already existed class.

When we write anonymous class?

If you want an object for an already existed class with different implementation instantly **Or** if u want an object for an interface instantly we will go for anonymous class.

AnonDemo1.java

```
import static java.lang.System.*;
class Don
{       String hero="Amitab";
        void display(){
                out.println("Don Movie");
                out.println("Hero:\t"+hero);
                out.println("Traditional Fighting Sequences");
        }
}
class Cinema
{
        Don d=new Don(){
                void display()
                {       hero="Sharukh";
                        out.println("Don(2008) Movie");
                        out.println("Hero:\t"+hero);
                        out.println("Hollywood Fighting Sequences");
                }
```

```
                };
}
class AnonDemo
{       public static void main(String args[])
        {       Cinema c=new Cinema();
                c.d.display();
        }
}
```

AnonDemo2.java

```
import static java.lang.System.*;
class Don
{       String hero="Amitab";
        void display()
        {       out.println("Don Movie");
                out.println("Hero:\t"+hero);
                out.println("Traditional Fighting Sequences");
        }
}
class Cinema
{       void billa()
        {       new Don(){
                void display()
                {       super.display();
                        hero="Sharukh";
                        out.println("Don(2008) Movie");
                        out.println("Hero:\t"+hero);
                        out.println("Hollywood Fighting Sequences");
                }
                }.display();
        }
}
class AnonDemo2
{       public static void main(String args[])
        {Cinema c=new Cinema();
        c.billa();
        }
}
```

Note: we can't use this keyword in a static method; in the same manner we can't use <OuterClassName>.this in static inner class.

Java Reflections

Java Reflection makes it possible to inspect classes, interfaces, fields and methods at runtime, without knowing the names of the classes, methods etc. at compile time. It is also possible to instantiate new objects, invoke methods and get/set field values using reflection.

Example-1(Reflect1.java)

```
import java.lang.reflect.*;
import static java.lang.System.*;
import java.util.*;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
public class Reflect1
{	public static void main(String str[])
	{	Class c1=Integer.class;
		String cname = c1.getName();
		out.println("Class:\t"+cname);
		Class sc=c1.getSuperclass();
		out.println("SuperClass:\t"+sc.getName());
		Class c2 =List.class;
		Class[] li = c2.getInterfaces();
		int length= li.length;
		for (int i =0; i < length; i++)
		{out.println(li[i]);
		}
	}
}
```

Example-2: FindMethFieldCons.java

```
import java.lang.reflect.*;
import static java.lang.System.*;
public class FindMethFieldCons
{	public static void main(String[] args)
	{	Class c =Integer.class;
		Field fields[] = c.getFields();
		out.println("Fields of:\t"+c.getName()+" Class");
		for(Field f:fields)
		{out.println(f.getName());
		}
		out.println("Methods of:\t"+c.getName()+" Class");
		Method[] lm=c.getMethods();
		for(Method m:lm)
		{out.println(m.getName());
		}
		Constructor cons[] = c.getConstructors();
		out.println("Constructors of:\t"+c.getName()+" Class");
		for(Constructor cn:cons)
		{out.println(cn);
		}
	}
}
```

Reflect3.java

```
import static java.lang.System.*;
import java.lang.reflect.*;
class One
{	public int a,b;
	public One()
	{out.println("Default con...");
	}
	public One(int a,int b)
	{this.a=a;
	this.b=b;
	}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        public void display()
        {out.println("Instance method of One class");
        }
        public void get()
        {out.println("static method of One class");
        }
        public String sayHello(java.lang.String name)
        {return "Hello:\t"+name;
        }
}
class Reflect3
{       public static void main(String args[])throws Exception
        {       Class c1=One.class;
                Method m=c1.getMethod("display");
                out.println(m);
                m.invoke(c1.newInstance());

                Method shm=c1.getMethod("sayHello",String.class);
                out.println(shm);
                String str=(String)shm.invoke(c1.newInstance(),"Sridevi");
                out.println("sayHello:\t"+str);
        }
}
```

Example-4

```
import java.lang.reflect.*;
import static java.lang.System.*;
public class InvokeMethod
{       public static void main(String[] args) throws Exception
        {       Class c=String.class;
                Method cm= c.getMethod("concat",c);
                String result = (String)cm.invoke("Madhu","Babu");
                out.println("Result:\t"+result);
        }
}
```

Assertions

What is Assertion?

An *assertion* is a statement that enables you to test your assumptions about your program. Each assertion contains a boolean expression if that expression returns false the system will throw an error.

The assertion confirms your assumptions about the behavior of your program, increasing your confidence that the program is free of errors.

Experience has shown that writing assertions while programming is one of the quickest and most effective ways to detect and correct bugs.

The assertion statement has two forms

First and simplest form is

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Example: assert expression1;

In the above example **expression1** is a Boolean expression. When the system runs the assertion. It evaluates expression1 and if it is false throws an **AssertionError** with no detail message.

The second form is

Example: assert expression1: expression2;

In the above example **expression1** is a Boolean expression and **expression2** is an expression that has a value.

Use this version of the assert statement to provide a detail message for the AssertionError. The system passes the value of $Expression_2$ to the appropriate AssertionError constructor, which uses the string representation of the value as the error's detail message.

We can enable or disable assertions whenever we want. We will discuss about it later.

The assertions are of two types

Preconditions: Preconditions are the assertions which invokes when a method is invoked

Postconditions: Postconditions are the assertions which invokes after a method finishes.

Example-1

```
import java.util.*;
import static java.lang.System.*;
public class AssertEx1
{       public static void main( String args[])
        {       Scanner scanner = new Scanner(in );
                out.print( "Enter a number between 0 and 20: " );
                int value = scanner.nextInt();
                assert( value >= 0 && value <= 20 );
                //assert( value >= 0 && value <= 20 ) :"Invalid number: " + value;
                out.printf( "You have entered %d\n", value );
        }
}
```

To run the above example:

Compile: javac AssertEx1.java

Run: java –ea AssertEx1

Note:

To enable assertions at runtime, -ea command line option is used. To disable assertions at runtime –da command line option is used.

I/O STREAMS

Introduction:

So far we have used variables and arrays for storing data inside the programs. This approach causes the following problems.

- The data is lost either when a variable goes out of scope or when the program is terminated. That is storage is temporary.
- It is difficult to handle large volumes of data using variables and arrays

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

We can overcome these problems by storing data on secondary storage devices such as floppy disks or hard disks. The data is stored in these devices using the concept of files. Data is stored in files is often called persistent data.

A file is a collection of related records placed in a particular area on the disk. A record is composed of several fields and a field is a group of characters. Characters in java are Unicode characters composed of two bytes, each byte containing eight binary digits, 1 or 0.

Concepts of Streams:

In file processing, input refers to the flow of data into a program and output means the flow of data out of a program.

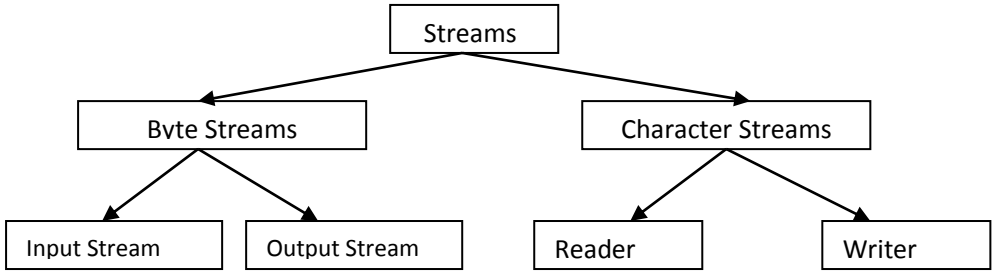
A stream represents a uniform, easy-to-use, object-oriented interface between the program and the input/output devices.

A stream in java is a path along which data flows (like a river or a pipe along which water flows). It has a source (of data) and a destination (for that data). Both the source and the destination may bphysical devices or programs or other streams in the same program.

The concept of sending data from one stream to another (like one pipe feeding into another pipe) has made streams in java a powerful tool for file processing. We can build a complex file processing sequence using a series of simple stream operations. This feature can be used to filter data along the pipeline of streams so that we obtain data in a desired format. For example, we can use one stream to get raw data in binary format and then use another stream in series to convert it to integers.

Types of Streams:

The **java.io** package contains large number of stream classes that provide capabilities for processing all types of data. These classes may be categorized into two groups based on the data type on which they operate.



- **Bytestream:** classes provide support for handling I/O operations on bytes.
- **Characterstream:** classes provide support for managing I/O operations on characters.

What is stream?

A stream is a pipe through which we can read or write the data from one place to another place.

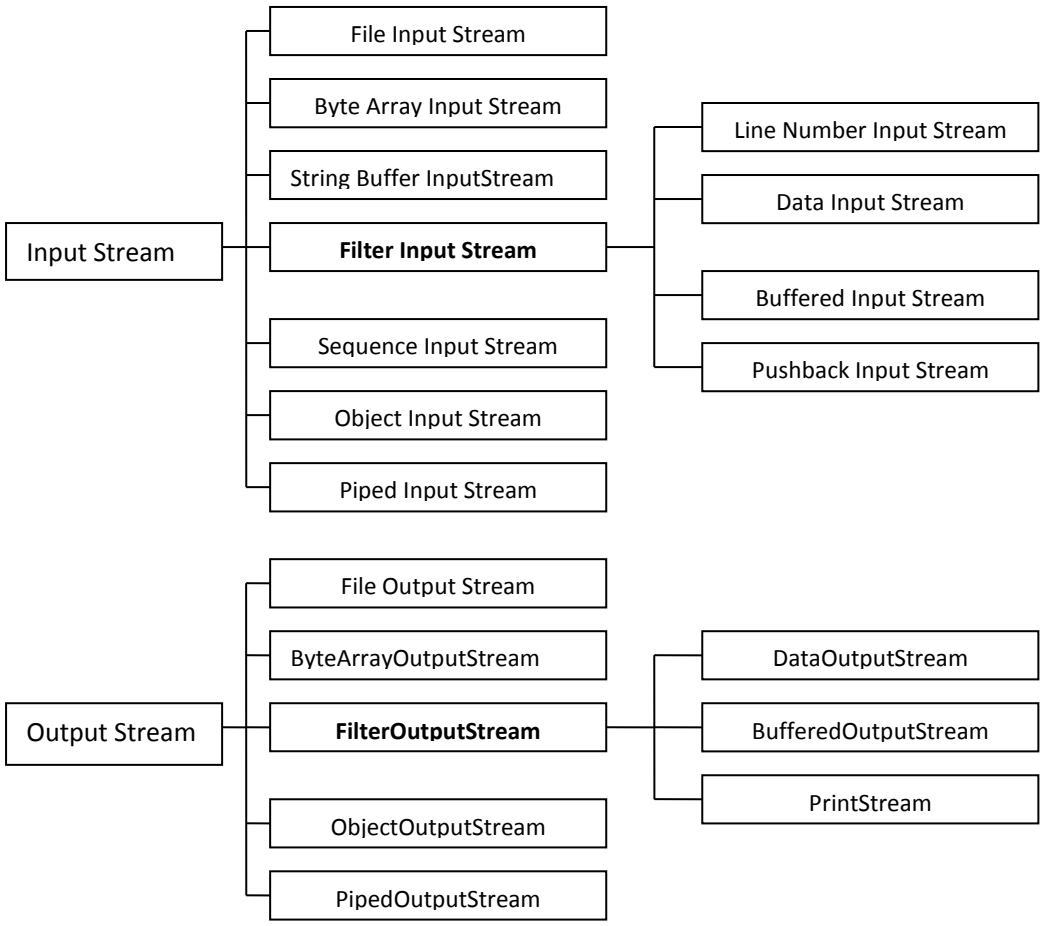
Byte Stream Classes:

Byte stream classes have been designed to provide functional features for creating and manipulating streams and files for reading and writing bytes. Since the streams are unidirectional,

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

they can transmit bytes in only one direction and therefore java provides two kinds of byte stream classes:

- **InputStream** classes: all input stream classes are the child classes of **java.io.InputStream** class.
- **OutputStream** classes: all output stream classes are the child classes of **java.io.OutputStream** class.

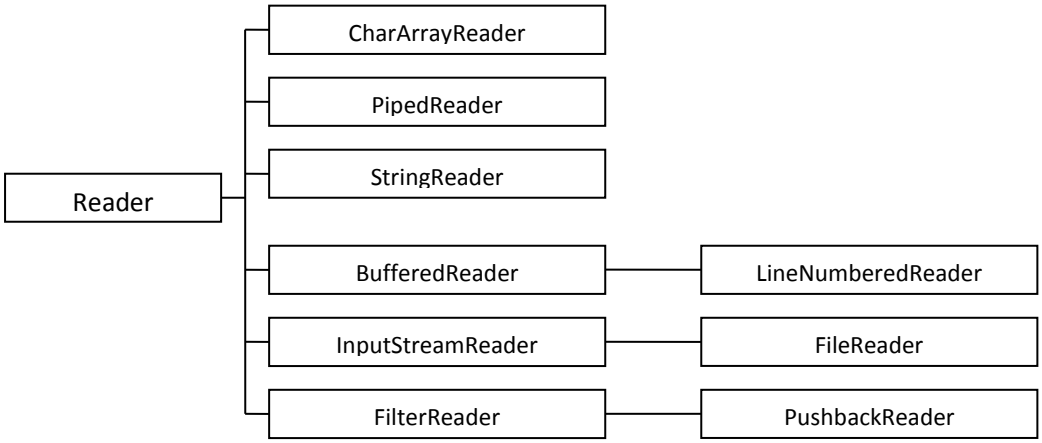


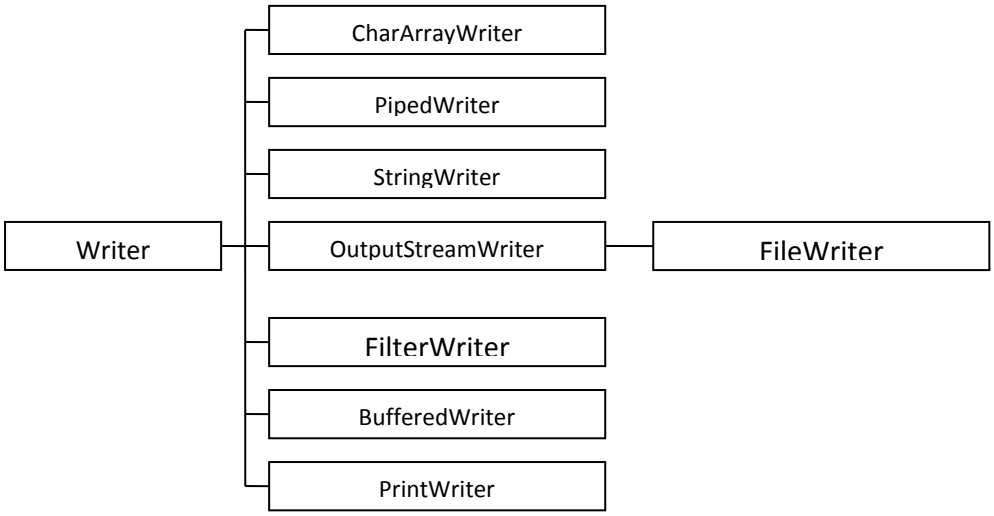
Character Stream Classes:

Character stream classes were not a part of the language when it was released in 1995. They were added later when the version 1.1 was announced, character streams can be used to read and write 16-bit Unicode characters. Like byte streams, there are two kinds of character stream classes, namely.

- **Reader stream** classes: all the reader stream classes are the child classes of the **java.io.Reader** class.
- **Writer stream** classes.: All the Writer streams classes are the child classes of the **java.io.Writer** class.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.





What is the use of streams?

Streams concept is used to send the data from one place to other. This very useful in networking programs where we send and receive the data from one place to other, and we already knows that, these streams are used take the input and print the data to output.

What is the difference between byte streams and character streams?

Byte streams are used to read or write the data in the form of individual bytes. Character streams are used to get or send data in the form of characters each of 2 bytes. Character streams are used to read/write Unicode character set. Where as byte streams are especially used to read/write images.

What is the use of File Class?

File class is used to create new file,create new directory, to check whether file or directory is existed or not, used to check whether File is read only or not, Hidden or not, is it directory or not, whether it is file or not, using it we can get the last accessed time. Used to remove file or directory and used to see the list of files, and directoris of a particular path and so many.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

FileDemo.java

```
import java.io.*;
import static java.lang.System.*;
import java.util.*;
public class FileDemo
{	public static void main(String args[])throws Exception
	{
		File f=new File("Two.txt");
		if(f.createNewFile())
		out.println("New file is created");
		else
		out.println("Already have a file with this name");

		File d=new File("madhu");
		if(d.mkdir())
		out.println("New Dir is created");
		else
		out.println("Already have a Dir with this name");

		File ff=new File("E:\\LocalNonLocal\\Yenti\\Yenduku\\Yela");
		if(ff.mkdirs())
		out.println("Yes");
		else
		out.println("No...no..no...noo...");

		out.println(f.getName()+"  Is File:\t"+f.isFile());
		out.println(f.getName()+"  Is Directory:\t"+f.isDirectory());
		out.println(d.getName()+"  Is Directory:\t"+d.isDirectory());
		out.println(d.getName()+"  Is File:\t"+d.isFile());
		out.println(f.getName()+"  Is Hidden:\t"+f.isHidden());
		out.println(d.getName()+"  Is Hidden:\t"+d.isHidden());
		out.println(f.getName()+"  Is Read:\t"+f.canRead());
		out.println(f.getName()+"  Is Write:\t"+f.canWrite());
		f.setReadOnly();
		out.println(f.getName()+"  Is Write:\t"+f.canWrite());

		out.println(new Date(f.lastModified()));
		//out.println(f.lastModified());
		out.println("Size of the "+f.getName()+"    "+f.length()+"  bytes");
		f.delete();
		if(f.exists())
		out.println(f.getName()+" file Undi babu Undi ");
		else
		out.println(f.getName()+" File ledu bangaram....");
		//File li=new File("E:\\LocalNonLocal");
		//File li=new File(".\\");
		File li=new File("./");
		String arg[]=li.list();
		out.println("\nfiles and directories in the  "+li.getPath()+" Path----------\n");
		out.println("\nfiles and directories in the  "+li.getAbsolutePath()+" Path----------
\n");
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                for(String name:arg)
                out.println(name);
        }
}
```

AllDirectories.java

```
import java.io.*;
import java.util.*;
import static java.lang.System.*;
class GetFiles
{       public void getAll()
        {       ArrayList<String> al=new ArrayList<String>();
                File[] files =File.listRoots();
                for(File f :files)
                {       out.println(f.getPath()+"-------directory----------------------");
                        File ff=new File(f.getPath());
                        File files2[]=ff.listFiles();
                        if(files2!=null)
                        for(File f1:files2)
                        {       out.println(f1);
                                out.println(f1.getName());
                        }
                }
                }
}
class AllDirectories
{       public static void main(String args[])
        {       GetFiles gf=new GetFiles();
                gf.getAll();
        }
}
```

What is Buffer?

buffer is a storage place where data can be kept before it is needed by a program that reads or writes that data. By using a buffer, you can get data without always going back to the original source to the data

Byte Streams:

FileInputStream and FileOutputStream: These streams are used to read the data from file and write the data to the file. But, they are very poor in performance because they read and write the data byte by byte. For this reason, programmers prefer these streams when the file size is of less size. The methods they include are read (), write (), available (), and close () etc…

FISFOS.java

```
import java.io.*;
import static java.lang.System.*;
class FISFOS
{       public static void main(String args[])throws Exception
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {FileInputStream fis=new FileInputStream("FileInputStreamDemo.java");
        FileOutputStream fos=new FileOutputStream("FISD.java");
        int r=0;
        while((r=fis.read())!=-1)
        {       out.print((char)r);
                fos.write(r);
        }
        fis.close(); fos.close();
        }
}
```

FOS.java

```
import java.io.*;
import static java.lang.System.*;
class FOS
{       public static void main(String args[])throws Exception
        {       DataInputStream dis=new DataInputStream(in);
                out.println("Do you want to creat a new file?? Give File Name");
                String fileName=dis.readLine();
                File f=new File(fileName);
                f.createNewFile();
                FileOutputStream fos=new FileOutputStream(f,true);
                String data;
                out.println("Write the data into the file...give 'end' to close ");
                out.println("After it open the file through word pad to get correct result");
                while(!(data=dis.readLine()).equals("end"))
                {       fos.write((data+"\n").getBytes());
                }
                fos.close();
        }
}
```

StringBufferInputStream: It is used to read the data from a string or used to convert string into a stream. But it is deprecated class and replaced by ***StringReader*** class because it doesn't properly convert characters in to bytes.

SBIS.java

```
import java.io.*;
import static java.lang.System.*;
public class SBIS
{       public static void main(String args[])throws Exception
        {StringBufferInputStream sb=new StringBufferInputStream("Take Risks in Life,If u win you
                        can lead,If u loss you can guide");
                DataInputStream dis=new DataInputStream(sb);
                /*byte[] b=new byte[65];
                //sb.read(b,0,3);
                sb.read(b);
                out.println(new String(b));
                */
                out.println(dis.readLine());
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                dis.close();
                sb.close();
        }
}
```

ByteArrayInputStream: The ByteArrayInputStream is used to read the data from a byte array. There are following forms of constructors to create ByteArrayInputStream objects

1. Takes a byte array as the parameter:

ByteArrayInputStream bArray = new ByteArrayInputStream(byte [] a);

2. Another form takes an array of bytes, and two ints, where off is the first byte to be read and len is the number of bytes to be read.

ByteArrayInputStream bArray = new ByteArrayInputStream(byte []a,int off,int len)

Note: if you want to convert a byte array into a stream use ByteArrayInputStream.

BAIS.java

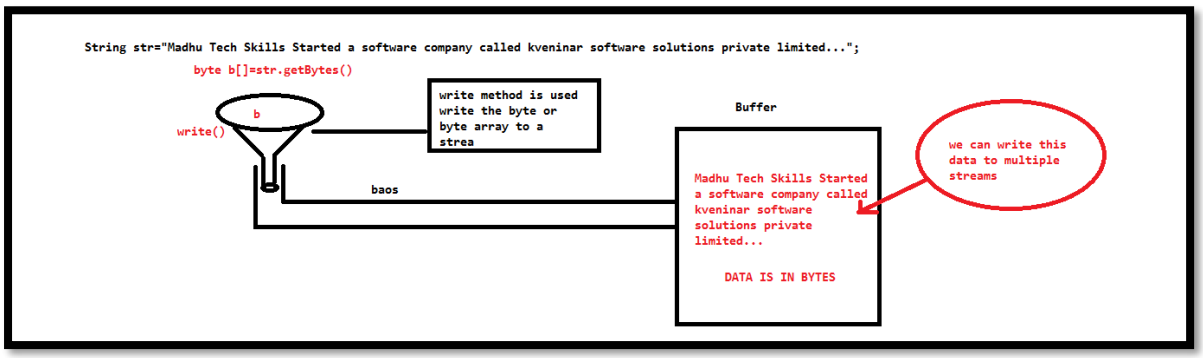
```
import static java.lang.System.*;
import java.io.*;
class BAIS
{       public static void main(String args[])throws Exception
        {       String str="Before you say: I can't ...say I will try..then give your best..";
                ByteArrayInputStream bis=new ByteArrayInputStream(str.getBytes());
                //constructor takes bytearray,starting index, no of bytes to read
                //ByteArrayInputStream bis=new ByteArrayInputStream(str.getBytes(),0,6);
                bis.skip(16);
                int r=0;
                while( (r=bis.read())!=-1)
                {out.print((char)r);
                }
                //it moves the cursor to the first position
                bis.reset();

                //Anothoer Usage
                byte b2[]=new byte[6];
                bis.read(b2,0,6);
                out.println("\n"+new String(b2));

                //it moves the cursor to the first position
                bis.reset();
                //Returns an int that gives the number of bytes to be read.
                out.println(bis.available());
        }
}
```

Note: Closing a ByteArrayInputStream has no effect. The methods in this class can be called after the stream has been closed without generating an IOException.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

ByteArrayOutputStream:

Java ByteArrayOutputStream is used to write the data to multiple streams. The ByteArrayOutputStream holds a copy of data and forwards it to multiple streams. The buffer of ByteArrayOutputStream automatically grows according to data.



BAOS.java

```
import static java.lang.System.*;
import java.io.*;
class BAOS
{	public static void main(String args[])throws Exception
	{	FileOutputStream fos1=new FileOutputStream("Navvu.java");
		FileOutputStream fos2=new FileOutputStream("Kovvu.java");
		//this stream connected to  buffer ( with 512 bytes)
		//ByteArrayOutputStream bos=new ByteArrayOutputStream();
		//this stream connected to  buffer ( with 1024 bytes)
		ByteArrayOutputStream bos=new ByteArrayOutputStream(1024);
		String str="Before you say: I can't ...say I will try..then give your best..";
		bos.write(str.getBytes());
		bos.writeTo(fos1);
		bos.writeTo(fos2);
		fos1.close();
		fos2.close();
	}
}
```

Note: Closing a ByteArrayOutputStream has no effect. The methods in this class can be called after the stream has been closed without generating an IOException.

SequenceInputStream: This stream is used to merge two files into one file and read the data from more than one file at a time. I.e. it is used to copy a number of source files into one destination file.

SIS1.java

```
import java.io.*;
import java.util.*;
import static java.lang.System.*;
public class SIS1
{	public static void main(String args[])throws Exception
	{	FileInputStream fis1=new FileInputStream("FOS.java");
		FileInputStream fis2=new FileInputStream("BAOS.java");
		SequenceInputStream sis=new SequenceInputStream(fis1,fis2);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                DataInputStream dis=new DataInputStream(sis);
                String str;
                while((str=dis.readLine())!=null)
                out.println(str);
                fis1.close(); fis2.close(); sis.close();dis.close();
        }
}
```

SIS2.java

```
import java.io.*;
import java.util.Enumeration;
import java.util.*;
import static java.lang.System.*;
public class SIS2
{       public static void main(String args[])throws Exception
        {       FileInputStream fis1=new FileInputStream("FOS.java");
                FileInputStream fis2=new FileInputStream("BAOS.java");
                FileInputStream fis3=new FileInputStream("SBIS.java");
                Vector  coll=new Vector();
                coll.add(fis1);
                coll.add(fis2);
                coll.add(fis3);
                coll.add(new FileInputStream("SIS.java"));
                Enumeration e=coll.elements();
                SequenceInputStream sis=new SequenceInputStream(e);
                DataInputStream dis=new DataInputStream(sis);
                String str;
                while((str=dis.readLine())!=null)
                out.println(str);
                fis1.close();fis2.close();fis3.close();sis.close();dis.close();
        }
}
```

PipedInputStream and PipedOutputStream: These streams are used to transfer the data between two running threads. PipedInputStream can read the data only from PipedOutputStream

PipedInputStreamDemo.java

```
import static java.lang.System.*;
import java.io.*;
class First extends Thread
{
        PipedOutputStream pos;
        First(PipedOutputStream pos)
        {this.pos=pos;
        }
        public void run()
        {       String str="Meenakshi, Lakshmi Durga";
                byte b[]=str.getBytes();
                for(int i=0;i<b.length;i++)
                {
                        try{pos.write(b[i]);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                        }catch(Exception e){out.println(e);}
                }
        }
}
class Second extends Thread
{       PipedInputStream pis;
        Second(PipedInputStream pis)
        {this.pis=pis;
        }
        public void run()
        {
                for(;;)
                {       try{
                                int r=pis.read();
                                out.print((char)r);
                        }catch(Exception e)
                        {//out.println(e);
                        break;
                        }
                }
        }
}
class PISPOS
{
        public static void main(String args[])throws Exception
        {
        PipedOutputStream pos=new PipedOutputStream();
        PipedInputStream pis=new PipedInputStream(pos);
        First t1=new First(pos);
        Second t2=new Second(pis);
        t1.start();
        t2.start();
        }
}
```

LineNumberInputStream: It is used to add line numbers that does not exist in the source file. An important method is **getLineNumber().**

LineNumberDemo.java

```
import java.io.*;
import static java.lang.System.*;
public class LNIS
{       public static void main(String args[])throws Exception
        {       FileInputStream fis=new FileInputStream("PISPOS.java");
                LineNumberInputStream ln=new LineNumberInputStream(fis);
                DataInputStream dis=new DataInputStream(ln);
                String str=null;
                while((str=dis.readLine())!=null)
                {out.println(ln.getLineNumber()+" "+str);
                }
                fis.close();
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                ln.close();
                dis.close();
        }
}
```

DataInputStream and DataOutputStream: These streams increase the prerformance to a great extent. They include special methods like readInt(). readDouble(). readLine(), writeInt(), writeDouble() and writeBytes() etc., which can read or write an integer, a double or a line at a time (not byte by byte).

DIODIS.java

```
import static java.lang.System.*;
import java.io.*;
class DISDOS
{       public static void main(String arg[])throws Exception
        {       FileOutputStream fos=new FileOutputStream("mydata.txt");
                DataOutputStream dos=new DataOutputStream(fos);
                FileInputStream fis=new FileInputStream("mydata.txt");
                DataInputStream dis=new DataInputStream(fis);
                byte b=20;
                short s=50;
                dos.writeInt(10);
                dos.writeBoolean(true);
                dos.writeByte(b);
                dos.writeShort(s);
                dos.writeFloat(10.00f);
                dos.writeDouble(44.44);
                dos.writeUTF("Vinutna,Pranutna");
                dos.close();
                out.println(dis.readInt());
                out.println(dis.readBoolean());
                out.println(dis.readByte());
                out.println(dis.readShort());
                out.println(dis.readFloat());
                out.println(dis.readDouble());
                out.println(dis.readUTF());
                dis.close();
        }
}
```

PrintStream: The PrintStream class provides methods to write the data to another stream. The PrintStream class automatically flushes the data so there is no need to call flush() method. The important methods are println(), print(), printf() and format().

PrintStreamDemo.java

```
import static java.lang.System.*;
import java.io.*;
class PSDemo
{       public static void main(String args[])throws Exception
        {       File f=new File("Ayyare.java");
                f.createNewFile();
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                PrintStream p=new PrintStream(f);
                p.println("Madhu Tech");
                p.println("Vijayawada-10");
                //used to write the data to other stream
                FileOutputStream fos=new FileOutputStream("Poyyare.java");
                PrintStream p2=new PrintStream(fos);
                p2.println("Tokkale");
                p2.println("Apararai...");
                int a=10;
                int b=20;
                int c=a+b;
                p2.format("%d+%d=%d",a,b,c);
                p2.printf("%d+%d=%d",a,b,c);
                p2.print(a+"+"+b+"="+c);
                p2.close();
        }
}
```

SetOut.java

```
import static java.lang.System.*;
import java.io.*;
class SetInSetOut
{
        public static void  main(String args[])throws Exception
        {
                System.setOut(new PrintStream("output.txt"));
                System.out.println("Welcome to java");
        }
}
```

FilterInputStream and FilterOutputStream: For get more functionality from streams, we can chain (link) streams. The functionality may include the concepts like to add the line numbers in the destination file that do not exists in the source file or to increase the performance. Filter streams are abstract classes. The subclasses of these streams are called as high-level streams and the remaining streams in the hierarchy are called low-level streams.

BufferedInputStream and BufferedOutputStream: These are high-level streams being the subclasses of Filter streams and they link (attach) a buffer memory to an input/output streams These are used to increase the performance.

BufferedInputStreamDemo.java

```
import java.io.*;
import static java.lang.System.*;
class BISBOS
{       public static void main(String args[])throws Exception
        {       FileInputStream fis=new FileInputStream("FileDemo.java");
                BufferedInputStream bis=new BufferedInputStream(fis);
                FileOutputStream fos=new FileOutputStream("FD.java");
```

```
                BufferedOutputStream bos=new BufferedOutputStream(fos);
                int r;
                while((r=bis.read())!=-1)
                {       out.print((char)r);
                        bos.write(r);
                }
                fis.close();bis.close();
                bos.flush();
                //bos.close();
                fos.close();
        }
}
```

What is the default buffer size used by any buffered class?

The default buffer size is 512 bytes

What is the use of BufferedOutputStream is it providing any efficiency?

Normally, when you write data into a file using **FileOutputStream** as **fos.write(ch),** the Operating System is invoked to write the character into the file. So, if we write 10 characters, the underlying Operating system would be called 10 times and it will write the characters into the file. This will take lot of time and it is burden to the processor

On the other hand, if Buffered classes are used, they provide a buffer (temporary block of memory), which is first filled with characters and then all the characters from the buffer are at once written into the file by the operating system. This takes less time and hence efficiency is more.

PushbackInputStream: Pushback is used on an input stream to allow a byte to be read and then returned (that is, "pushed back") to the stream.

PushBackDemo.java

```
import static java.lang.System.*;
import java.io.*;
class PBIS
{       public static void main(String args[])throws Exception
        {       String str="Bhasha,Raja,Gowti,Simha,Phani";
                ByteArrayInputStream bis=new ByteArrayInputStream(str.getBytes());
                PushbackInputStream pbis=new PushbackInputStream(bis);
                int r=pbis.read();
                out.println((char)r);
                pbis.unread(r);
                r=pbis.read();
                out.println((char)r);
        }
}
```

OBJECT PERSISTENCE AND SERIALIZATION

Serialization (or) Object serialization:

Serialization is the process, used for the persistence of an object by writing the object's state to a stream of data. Object serialization takes an object's state, and converts it to a byte stream. Object

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

serialization provides a program the ability to read or write a whole object at a time from and to a raw byte stream. A serialized format contains the object's state information.

What is data/object persistence?

Object persistence is the ability of an object to *record* its state in the file this is called data persistence. So it can be reproduced in the future, perhaps in another environment.

What is Deserialization

Deserialization is the process of converting or reconstructing the state of the serialized object to the original state at a later time

In general there are three approaches to serialization in java

1. Implement Serializable and use default protocol.
2. Implement Serializable and get a chance to modify the default protocol.
3. Implement Externalizable and write your own protocol to implement serailization.

Note: Here protocol means the way object is serialized and de-serialized

1. Implement Serializable and use default protocol:

Not all classes are capable of being serialized. Only classes that implement the Serializable or Externalizable interface can be serialized. Both of these interfaces are in the **java.io** package.

The *Serializable* interface

The Serializable interface does not have any methods, it is a marker interface. When a class implements **Serializable** interface, it is declaring that it participates in the serializable protocol. And the objects of it's class support serialization, When an object is serializable we can write the objects state to a file through a stream.

ObjectInputStream and ObjectOutputStream: These streams are used to read objects and write objects to a file (for a permanent storage). When we write an object to file, along with the object the encapsulated variables also go and stored in the file. With this, a lot of code comes down. The methods they include are readObject () and writeObject () etc.

OISOOS.java

```
import java.io.*;
import static java.lang.System.*;
class Student implements Serializable
{       int sno;
        transient String sname;
        Student(int sno,String sname)
        {this.sno=sno;
        this.sname=sname;
        }
        public String toString()
        {return "Sno:=\t"+sno+"\nSname:\t"+sname;
        }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
}
class Emp implements Serializable
{       int eno;
        String ename;
        Emp(int eno,String ename)
        {this.eno=eno;
        this.ename=ename;
        }
        public String toString()
        {return "Eno:=\t"+eno+"\nEname:\t"+ename;
        }
}
class OISOOS
{       public static void main(String args[])throws Exception
        {       Student s=new Student(101,"Murali");
                Emp e=new Emp(102,"Kiran");
                FileOutputStream fos=new FileOutputStream("Objets.data");
                ObjectOutputStream oos=new ObjectOutputStream(fos);
                oos.writeObject(s);
                oos.writeObject(e);
                oos.close();
                fos.close();
                FileInputStream fis=new FileInputStream("Objets.data");
                ObjectInputStream ois=new ObjectInputStream(fis);
                for(int i=1;i<=2;i++)
                {out.println(ois.readObject());
                }
                ois.close();
                fis.close();
        }
}
```

What is the use of transient keyword?

If u declare a field of a class as transient that will not support serialization. So If you don't want to serialize some fields of a class declare those fields as transient.

2.Implement Serializable and get a chance to modify the default protocol

In the above class we just uses the standard mechanism and demonstrates serialization using Emp and Student classes.

Customize Java Serialization

We know Serializable is a java marker interface. When a class implements Serializable interface it gives information to the JVM that the instances of these classes can be serialized. Along with that, there is a special note to the JVM which tells that *look for following two methods in the*

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

class that implements Serializable. If found invoke them and continue with serialization process else directly follow the standard serialization protocol.

Those methods are given below

- private void writeObject(ObjectOutputStream out) throws IOException;
- private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException;

In the below example I have modified the Emp class by including these two methods . This is not overriding or overloading methods and this is a mechanism provided by serialization.

Note:

These two included methods are declared private but JVM can access the private methods of an object. There is no change to the class that does the serialization and de-serialization

Example: OISOOS2.java

```
import java.io.*;
import static java.lang.System.*;
class Emp implements Serializable
{       private int eno;
        private String ename;
        Emp(int eno,String ename)
        {this.eno=eno;
        this.ename=ename;
        }
        public void setEno(int eno)
        {this.eno=eno;
        }
        public int getEno()
        {return eno;
        }
        public void setEname(String ename)
        {this.ename=ename;
        }
        public String getEname()
        {return ename;
        }
        public String toString()
        {return "Eno:=\t"+eno+"\nEname:\t"+ename;
        }
        private void writeObject(ObjectOutputStream out) throws IOException
        {       setEno(100);
                setEname("Madhu");
                out.defaultWriteObject();
        }
        private void readObject(ObjectInputStream in)throws IOException, ClassNotFoundException
        {in.defaultReadObject();
        }
}
class OISOOS2
{       public static void main(String args[])throws Exception
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
{       Emp e=new Emp(102,"Kiran");
        FileOutputStream fos=new FileOutputStream("Objets.data");
        ObjectOutputStream oos=new ObjectOutputStream(fos);
        oos.writeObject(e);
        oos.close();
        fos.close();
        FileInputStream fis=new FileInputStream("Objets.data");
        ObjectInputStream ois=new ObjectInputStream(fis);
        out.println(ois.readObject());
        ois.close();
        fis.close();
}
}
```

Disadvantage:

If we want to implement our own serialization mechanism by implementing the **Serializable** interface(yes it is also possible by defining **writeObject** and **readObject** methods) we don't need to override or implement any method.

The JVM calls the serialization methods from our class **using reflection**. In early JVM implementations **reflection performance was** kind of **slow,** so to overcome this problem the **Externalizable** interface was introduced.

3.Implement Externalizable and write your own protocol to implement serailization.

Externalizable is an interface that enables you to define custom rules and your own mechanism for serialization. Implementing the **Externalizable** interface means that we must override some of its methods, namely the **writeExternal** and **readExternal** methods. These methods will be called when you serialize (or deserialize) a given instance.

The Externalizable interface is defined as:

public interface Externalizable extends Serializable

The method of Externalizable interface

public void writeexternal (ObjectOutput out) throws IOException;
public void readexternal (ObjectInput in) throws IOException, ClassNotFoundException;

writeExternal():

a writeExternal() method for storing its state during serialization.

readExternal():

a readExternal() method for restoring its state during deserialization.

Example: OISOOS.java

```
import java.io.*;
import static java.lang.System.*;
class Emp implements Externalizable
{       private int eno;
        private String ename;
        Emp(int eno,String ename)
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {this.eno=eno;
        this.ename=ename;
        }
        public Emp()
        {out.println("Called...");
        }
        public void setEno(int eno)
        {this.eno=eno;
        }
        public int getEno()
        {return eno;
        }
        public void setEname(String ename)
        {this.ename=ename;
        }
        public String getEname()
        {return ename;
        }
        public String toString()
        {return "Eno:=\t"+eno+"\nEname:\t"+ename;
        }
        @Override
        public void writeExternal(ObjectOutput out) throws IOException
        {       out.writeInt(eno);
                out.writeObject(ename);
        }
        @Override
        public void readExternal(ObjectInput in) throws IOException,ClassNotFoundException
        {       eno = 100;//in.readInt();
                ename ="Madhu";// (String) in.readObject();
        }
}
class OISOOS3
{       public static void main(String args[])throws Exception
        {       Emp e=new Emp(102,"Kiran");
                FileOutputStream fos=new FileOutputStream("Objets.data");
                ObjectOutputStream oos=new ObjectOutputStream(fos);
                oos.writeObject(e);
                oos.close();
                fos.close();
                FileInputStream fis=new FileInputStream("Objets.data");
                ObjectInputStream ois=new ObjectInputStream(fis);
                out.println(ois.readObject());
                ois.close();
```

```
                fis.close();
        }
}
```

CHARACTER STREAMS

FileReader and FileWriter: File Reader is used to read the data from a file character by character and File Writer is used to write character by character to a file.

FRFW.java

```
import java.io.*;
import static java.lang.System.*;
class FRFW
{       public static void main(String args[])throws Exception
        {       FileReader fr=new FileReader("FileDemo.java");
                FileWriter fw=new FileWriter("MyFileDemo.java");
                int b;
                while((b=fr.read())!=-1)
                {       out.print((char)b);
                        fw.write(b);
                }
                fr.close(); fw.close();
        }
}
```

BufferedReader and Buffered Writer: They buffer the data while reading and writing, thereby reducing the number of transfers required on the original data source. Buffered streams are typically more efficient than similar non-buffered streams.

BRBW.java

```
import java.io.*;
import static java.lang.System.*;
class BRBW
{       public static void main(String args[])throws Exception
        {       FileReader fr=new FileReader("FRFW.java");
                BufferedReader br=new BufferedReader(fr);
                FileWriter fw=new FileWriter("OurFileDemo.java");
                BufferedWriter bw=new BufferedWriter(fw);
                int b=0;
                while((b=br.read())!=-1)
                {       out.print((char)b);
                        bw.write(b);
                }
                br.close();fr.close();bw.close();fw.close();
        }
}
```

BRBW2.java

```
import java.io.*;
import static java.lang.System.*;
```

```
public class BRBW2
{	public static void main(String args[])throws Exception
	{	FileWriter fw=new FileWriter("haa.java");
		BufferedWriter bw=new BufferedWriter(fw);
		bw.write("java is an oop language\n",0,23);
		bw.write("java is an oop language\n",0,23);
		bw.flush();
		FileReader fr=new FileReader("haa.java");
		BufferedReader br=new BufferedReader(fr);
		String str=null;
		while((str=br.readLine())!=null)
		{out.println(str);
		}
	}
}
```

CharArrayReader and Char ArrayWriter: CharArrayReader is used to convert char array to stream.CharArrayWriter is used to write the data to more than one Writer stream.

CharArrayReaderWriterDemo.java

```
import static java.lang.System.*;
import java.io.*;
class CARCAW
{
	public static void main(String args[])throws Exception
	{	char ch[]={'J','a','v','a',' ','i','s'};
		CharArrayReader car=new CharArrayReader(ch);
		CharArrayWriter caw=new CharArrayWriter();
		int r=0;
		while((r=car.read())!=-1)
		{out.println((char)r);
		caw.write(r);
		}
		FileWriter fw=new FileWriter("Gowti.java");
		caw.writeTo(fw);
		caw.close();
		fw.close();
		car.close();
	}
}
```

FilterReader and Filter Writer: These are abstract classes, which add special effects like adding line numbers or increasing the performance etc. They define the interface for filter streams, which filter data as it's being read or written. Filter streams provide additional functionality to the streams. A filter is a type of stream that modifies the way of handling the existing stream.

InputStreamReader and OutputStreamWriter: These streams are used to convert by stream into character stream and vice versa. They form a bridge between byte streams and character streams. An

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

InputStreamReader reads bytes from an InputStream and converts them to characters. Similarly, an OutputStreamWriter converts characters to bytes and writes those bytes to an OutputStream.

StringReader and StringWriter: String Reader is used to read data from a string which is stored in buffer(ram). String Writer is used to write data to buffer(ram).

StringReaderWriter.java

```
import java.io.*;
import static java.lang.System.*;
public class SRSW
{	public static void main(String args[])throws Exception
	{	StringWriter sw=new StringWriter();
		FileReader fr=new FileReader("SRSW.java");
		int ch=0;
		while((ch=fr.read())!=-1)
		{sw.write(ch);
		}
		out.println(sw);
		StringReader sr=new StringReader(sw.toString());
		while((ch=sr.read())!=-1)
		{out.print((char)ch);
		}
	}
}
```

LineNumberReader: this Line Number Reader Stream is used to print line numbers while reading data from a file.

PushbackReader: The PushbackReader class is used for special purpose. Once the program has determined that the current aggregate of data is complete, the extra character is "pushed back" onto the stream. PushbackInputStream is used by programs like compilers that parse their inputs, that is break them into meaningful units (like keywords, identifiers etc).

PrintWriter: These class contains convenient printing methods like println () and print (). These are the easiest streams to write to monitor.

Random AccessFile: Random Access File is used to read the data from file in randomly not in sequential manner.

RandomAcceeDemo.java

```
import java.io.*;
import static java.lang.System.*;
public class RAC
{	public static void main(String[] args)throws Exception
	{	File f=new File("SBIS.java");
		RandomAccessFile file=new RandomAccessFile(f,"rw");
		//file.seek(0);
```

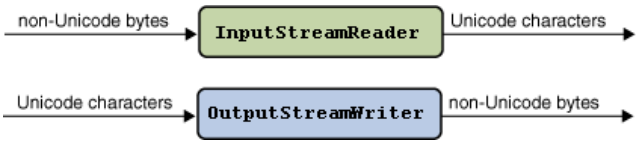

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
            out.println(file.readLine());
            out.println(file.getFilePointer());
            file.seek(54);
            out.println(file.readLine());
            file.writeBytes("/*Aie   Matladamak*/");
        }
}
```

Character and Byte Streams:

In Java we have streams those allows us to convert between Unicode character streams and byte streams of non-Unicode text. Those are

1. InputStreamReader
2. OutputStreamWriter



Using InputStreamReader we can convert byte streams to character streams. And using OutputStreamWriter we can convert character streams to byte streams.

ISROSW.java

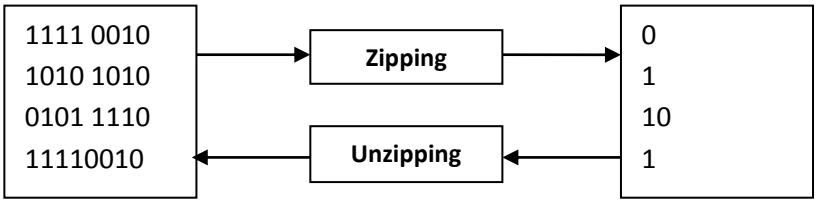
```
import static java.lang.System.*;
import java.io.*;
class ISROSW
{       public static void main(String args[])throws Exception
        {       FileOutputStream fos=new FileOutputStream("Abba.java");
                //Converting character stream to byte stream
                OutputStreamWriter osw=new OutputStreamWriter(fos);
                BufferedWriter bw=new BufferedWriter(osw);
                FileInputStream fis=new FileInputStream("FileDemo.java");
                //Converting byte stream to character stream
                InputStreamReader isr=new InputStreamReader(fis);
                BufferedReader br=new BufferedReader(isr);
                int r=0;
                while((r=br.read())!=-1)
                {out.print((char)r);
                bw.write(r);
                }
                br.close();isr.close();fis.close();bw.close();osw.close();fos.close();
        }
}
```

Zipping and Unzipping Files

Introduction:

We know that some softwares like "WinZip" and "WinRAR" provide zipping and unzipping of file data. When you zip/unzip the file contents, following two things will happen:

9. The file contents are compressed and hence the size of the file will be reduced.
10. The format of data will be changed into unreadable.



DeflaterOutputStream and InflaterInputStream:

DeflaterOutputStream is used to zip the file and **InflaterInputStream** is used to unzipping the file. These classes are available in **java.util.zip** package

One.txt

MadhuTechSkills is a very good institute,
Madhu Tech Skills is providing live projects,
It is providing free demo classes for 7 days
If you don't like the class fees will be refunded

1. Write a program to compress the "one.txt" into "two.zip" file

Zipping.java

```
import java.io.*;
import java.util.zip.*;
class Zipping
{	public static void main(String args[])throws Exception
	{	FileInputStream fis=new FileInputStream("one.txt");
		FileOutputStream fos=new FileOutputStream("two.zip");
		DeflaterOutputStream dos=new DeflaterOutputStream(fos);
		int r;
		while((r=fis.read())!=-1)
		{	dos.write(r);
		}
		fis.close(); dos.close(); fos.close();
	}
}
```

The resultant file is

Two.zip

xœ%‹A*f*0

ïHüa□Ñ÷lŠÛ„ ¯*f*"ß·´·□¦ŠîNQ,mÀóöŒnóT• 9mÝÁ·—Â;±Ÿð*f*éÙÓ¾„Ë_2Dežþ÷írõã‰-Ì VÖ"¤[< 2øØª,m

2. Write a program to de-compress the "two.zip" into "three.txt" file

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

UnZipping.java

```
import java.io.*;
import java.util.zip.*;
import static java.lang.System.*;
class UnZipping
{       public static void main(String args[])throws Exception
        {       FileInputStream fis=new FileInputStream("two.zip");
                InflaterInputStream iis=new InflaterInputStream(fis);
                FileOutputStream fos=new FileOutputStream("three.txt");
                int r;
                while((r=iis.read())!=-1)
                {       fos.write(r);
                        out.print((char)r+"");
                }
                iis.close(); fis.close();  fos.close();
        }
}
```

three.txt

MadhuTechSkills is a very good institute,
Madhu Tech Skills is providing live projects,
It is providing free demo classes for 7 days
If you don't like the class fees will be refunded

STRINGS AND STRING HANDLING

Usually a string is nothing but collection of characters. In C/C++ a string represents an array of characters, where the last character will be '\0' (called null character) represents the end of the string, but in Java String is an object of String class. In java we have character array also, but Strings are given different treatment because of their extensive use on Internet.

In java, Strings can be handled with four classes

- **java.lang.String class**
- **java.lang.StringBuffer(Synchronized) class**
- **java.lang.StringBuilder(not Synchronized) class**
- **java.util.StringTokenizer class.**

What is the difference between String and StringBuffer?

When we reassign a value to a variable, system changes the value and the location; but not the location itself. That is, in the same location, the new value of the variable is changed. But incase of strings, it is completely different. Incase of strings, if a string is reassigned a new value, a new location is created where the new value is stored and the old location (where old value is stored) is garbage collected. That is , a string value cannot be changed in the same location. This is called immutable and is an overhead to the processor and a negative point for performance. To over come this, Java designers introduced StringBuffer. Generally, incase of a String Buffer, if a value is changed, the location is not changed and in the old location only the value changed.

What are mutable and immutable objects?

Mutable objects are those objects whose contents can be modified. Immutable objects are objects whose contents can't be changed.

String class:

Eventhough , String is a class, it is used often in the form of a data type, as:

```
String str="Madhu Tech Skills";  //Here str is the variable of type String
```

There are three ways to create strings in Java:

11. we can create a string just by assigning a group of characters to a string type variable

```
String str="Madhu Tech Skills";
```

12. we can create using new operator

```
String str=new String("Madhu Tech Skills");
```

13. we can create a string class object by passing character array to constructor

```
char arr[]={'m','a','d','h','u'};
String str=new String(arr);  //now string object str contains "madhu"
String str=new String(arr,1,3);//now String object 'str' contains "adh"
//Here  jvm gets 3 character starting from 1st index
```

String Class Methods

Method	What it does
String concat(String s)	It concatenates or joins two strings
int length()	Returns the no of characters of a strings
char charAt(int i)	It gets the character from the specified location
int compareTo(String s)	Used to compare two strings, if two strings are equal it returns 'zero', if first string is greater than second string it returns a positive number, if first string is less than second string then it returns negative value.
int compareToIgnoreCase(String s)	This is same as above method but it compares by ignoring case here 'madhu' and 'MADHU' is same
boolean equals(String s)	This method returns true if two string are equal otherwise it returns false
boolean equalsIgnoreCase(String s)	It compares two string by ignoring the case if both are equal it returns true other wise it returns false
boolean startsWith(String s)	This method returns true if the string starts with string 's' otherwise it returns false
boolean endsWith(String s)	This method returns true if the string ends with string 's' otherwise it returns false
int indexOf(String s)	It returns the first occurence index position of the given string s
int lastIndexOf(String s)	It returns the last occurence index position of the given string s
String replace(char c1,char c2)	It replaces all occurences of c1 with c2 character
String substring(int i)	It is used to get the substring from a string starting from position i until the end of the string
String substring(int i1,int i2)	It is used to get the substring from a string starting from position i1 until the position (i2-1).

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

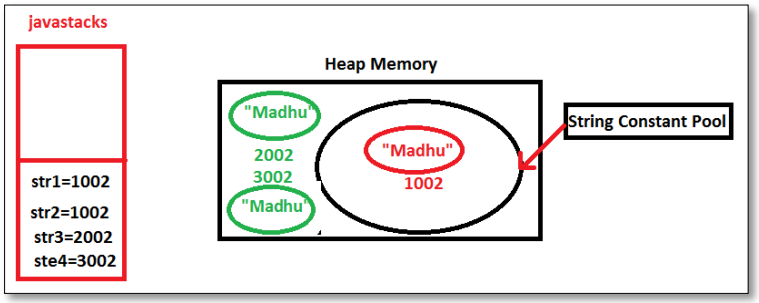
String toLowerCase()	It converts all the characters of a string into lower case
String toUpperCase()	It converts all the characters of a string into Upper case
String trim()	This method removes spaces from the beginning and ending of a string
void getChars(int i1,int i2,char arr[],int i3)	This method copies characters from a string into a character array. The characters starting from i1 to (i2-1) int the string are copied into the array 'arr' to a location starting from i3
String [] split(String delimeter)	This method is used to break a string into no of pieces based on the given delimeter.

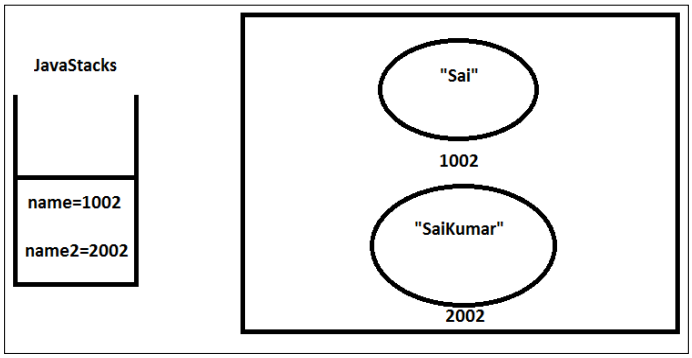
StringDemo1.java

```
import static java.lang.System.*;
class StringDemo
{
	public static void main(String args[])
	{	String str1="Madhu";
		String str2="Madhu";

		if(str1==str2)
		out.println("Addresses are same");
		else
		out.println("Addresses are different");

		String str11=new String("Madhu");
		String str22=new String("Madhu");
		if(str11==str22)
		out.println("Addresses are same");
		else
		out.println("Addresses are different");

		if(str1.equals(str2))
		out.println("Content is same");
		else
		out.println("Content is different");

		if(str11.equals(str22))
		out.println("Content is same");
		else
		out.println("Content is different");

		str1="Priya";
		str2="priya";


		if(str1.equals(str2))
		out.println("Content is same");
		else
		out.println("Content is different");

		if(str1.equalsIgnoreCase(str2))
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println("Content is same");
                else
                out.println("Content is different");

                str1="a";
                str2="A";
                int r=str1.compareTo(str2);

                if(r>0)
                out.println("str1 > str2");
                else if(r<0)
                out.println("str1 < str2");
                else
                out.println("str1 == str2");

                r=str1.compareToIgnoreCase(str2);
                if(r>0)
                out.println("str1 > str2");
                else if(r<0)
                out.println("str1 < str2");
                else
                out.println("str1 == str2");
        }
}
```





String object are immutable

StringDemo3.java

```
import static java.lang.System.*;
class StringDemo3
```

```
{
	public static void main(String args[])
	{
		String data="java is an object oriented programming language";
		out.println(data.length());
		out.println(data.charAt(20));
		out.println(data.indexOf("ie"));

		out.println(data.lastIndexOf("i"));
		data=data.concat("...!");
		out.println(data);
		byte b[]=data.getBytes();

		for(byte v:b)
		out.print((char)v);

		char dst[]=new char[47];
		data.getChars(0, 47,dst,0) ;

		out.println("\n");
		for(char ch:dst)
		out.print(ch);
		out.println();
		out.println(data.substring(11));
		out.println(data.substring(11,35));
		out.println(data.toUpperCase());
		out.println(data.toLowerCase());


		String str=String.valueOf(100);
		out.println("str:\t"+str);
		str=String.valueOf(100.45f);
		out.println("str:\t"+str);
		str=String.valueOf(true);
		out.println("str:\t"+str);
		str=String.valueOf(34.34);
		out.println("str:\t"+str);
		str=String.valueOf('h');
		out.println("str:\t"+str);
		str='h'+"";
		out.println("str:\t"+str);
		str=100+"";
		out.println("str:\t"+str);
		out.println(data.replace("java","JAVA"));
		String cts=String.copyValueOf(dst);
		out.println(cts);
		cts=String.copyValueOf(dst,0,10);
		out.println(cts);

		str=new String(dst);
		out.println(str);
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                String s1="                madhu babu              ";
                out.println(s1.length());
                s1=s1.trim();
                out.println(s1.length());
                out.println("s1="+s1);
                out.println("Elements in an array....");
                s1="Madhu Tech Skills";
                String tokens[]=s1.split(" ");
                for(String ss:tokens)
                out.println(ss);
                out.println(s1.startsWith("Madhu"));
                out.println(s1.endsWith("u"));
        }
}
```

StringBuffer Class

We know that Strings are immutable; to over come this we have another class in **Java** that is **StringBuffer class.** It represents strings in a way that their data can be modified. It means **StringBuffer** class objets are mutable.

There are two ways to create a StringBuffer object.

14. We can create a StringBuffer object by using new operator and pass the string to the object.

```
StringBuffer sb=new StringBuffer("MadhuTechSkills");
```

15. Another way of creating a StringBuffer object

```
StringBuffer sb=new StringBuffer();
```

Here, we are creating a StringBuffer object with the default capacity of 16 characters.

```
StringBuffer sb=new StringBuffer(50);
```

Here, we are creating a StringBuffer object with the capacity of 50 characters.

StringBuffer Class Methods

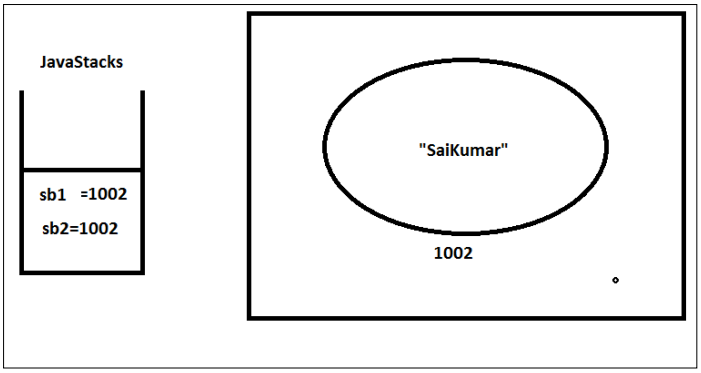
Method	What it does
StringBuffer append(x)	This method is used to add anything to the StringBuffer Object. Here **x** may be byte, short, int, long, float, double, char ,char array, boolean or another StringBuffer.
StringBuffer insert(int i,x)	Here x may be any type it will be inserted into the StringBuffer at the position represented by i.
StringBuffer delete(int I,int j)	It removes the characters from ith position till (j-1)th position in the StringBuffer.
StringBuffer reverse()	It reverse the string in the StringBuffer object
int length()	It returns the no of characters of a StringBuffer
int indexOf(String s)	It returns the first occurence index position of the given string s
Int lastIndexOf(String s)	It returns the last occurrence of substring 's' in the StringBuffer object.
StringBuffer replace(int i,int j,String str)	This method replaces the characters from i to j-1 with the string 'str'.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

StringBuffer substring(int i)	It is used to get the substring from a StringBuffer starting from position i until the end of the string
StringBuffer substring(int i,int i2)	It is used to get the substring from a StringBuffer starting from position i1 until the position (i2-1).

StringBufferDemo.java

```
import static java.lang.System.*;
public class SBDemo
{
	public static void main(String args[])
	{	StringBuffer data=new StringBuffer("java is an object oriented programming
language");
		out.println(data.length());
		out.println(data.charAt(20));
		out.println(data.indexOf("i"));
		out.println(data.lastIndexOf("i"));
		data=data.append("...!");
		out.println(data);
		out.println(data+"...!");
		char dst[]=new char[47];
		data.getChars(0, 47,dst,0) ;
		out.println();
		for(int i=0;i<dst.length;i++)
		out.print(dst[i]);
		out.println();
		out.println(data.substring(11));
		out.println(data.substring(11,35));
		out.println(data.insert(5," 1991 "));
		out.println(data.reverse());
		out.println(data.reverse());
		out.println(data.replace(0,4,"Cpp"));
		out.println(data.deleteCharAt(4));
		out.println(data.delete(0,9));

	}
}
```



StringBuffer Objects are mutable

StringBuilder Class

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

StringBuilder class is introducted in jdk1.5, this class is similar to StringBuffer except StringBuffer is synchronized and StringBuilder is not.

Following ways are used to create StringBuilder class object

- StringBuilder sb=new StringBuilder("MadhuTechSkills");
- StringBuilder sb=new StringBuilder();
- StringBuilder sb=new StringBuilder(50);

StringBuilder Class Methods

Method	What it does
StringBuilder append(x)	This method is used to add anything to the StringBuilder Object. Here **x** may be byte, short, int, long, float, double, char ,char array, boolean or another StringBuilder.
StringBuilder insert(int i,x)	Here x may be any type it will be inserted into the StringBuilder at the position represented by i.
StringBuilder delete(int I,int j)	It removes the characters from ith position till (j-1)th position in the StringBuilder.
StringBuilder reverse()	It reverse the string in the StringBuilder object
int length()	It returns the no of characters of a StringBuilder
int indexOf(String s)	It returns the first occurence index position of the given string s
Int lastIndexOf(String s)	It returns the last occurrence of substring 's' in the StringBuilder object.
StringBuilder replace(int i,int j,String str)	This method replaces the characters from i to j-1 with the string 'str'.
StringBuilder substring(int i)	It is used to get the substring from a StringBuilder starting from position i until the end of the string
StringBuilder substring(int i,int i2)	It is used to get the substring from a StringBuilder starting from position i1 until the position (i2-1).

What is strictfp?

It is a Java keyword used to restrict floating-point calculations to ensure portability. The modifier was introduced into the Java programming language with the Java virtual machine version 1.2.When this modifier is specified, the JVM sticks to the Java specifications and returns the consistent value, independent of the platform. That is, if you want the answers from your code (which uses floating point values) to be consistent in all platforms, then you need to specify the strictfp modifier. Java's default class Math uses this strictfp modifier as shown below

Ex: public final strictfp class Math{ }

StrictFPDemo.java //this program shows the real values

```
import static java.lang.System.*;
public class SFP
{       public static strictfp void main(String[] argv)
        {       out.println(sqr1(1234.45f));
                out.println(sqr2(1234.45f));
        }
        static float sqr1(float a)
        {return a * a;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        }
        static strictfp float sqr2(float a)
        {return a *a;
        }
}
```

Annotations

What is Annotation?

In General an annotation is a note, but in java it is a way to add meta-data to a Java Element (an element can be a class, method, or anything) and these meta-datas are processed by the tools (compilers, javadoc, etc.). Annotations are differentiated from other elements like class, interface etc. by preceding an '@' symbol before it. Annotations provide data about a program that is not part of the program itself. They have no direct effect on the operation of the code they annotate.

This feature was added to Java 5.0, annotations in java can be seen anywhere in a program. It can be seen in a class declaration, method declaration, field declaration etc… applying annotations to Java elements have proven themselves to be considerably useful in many instances consider the following class definition.

```
final class Emp
{       private String name;
        private String eno;
        // Getters and setters for Employee class.
}
```

The above class is declared as final so it can't be inherited by child class. So the introduction of the final keyword adds some additional information (or adds some constraints) over the class definition telling that no other class can extend the Emp class. Therefore the final keyword is providing some metadata information to the class definition. So, this is one variation of Annotation.

Defining an annotation is looks like the following

```
@Target(ElementType.METHOD)
public @interface TestAnnotation
{// Property Definition here.
}
```

Don't get confused with the interface keyword. It has nothing to do with annotations. '@' along with interface is the start of the annotation definition and **Test Annotation** in the above case is the name of the annotation.

To Which Element we can apply this annotation?

Whether annotation can be applied to class (a class-level annotation), or a method (method-level annotation) or a field (field-level annotation) is specified in the declaration of the annotation itself. This is referred to as **Annotating an Annotation** itself we will talk later about it.

Types of annotations

There are three types of annotations available in Java

1. **Marker Annotations**

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

2. **Single-value Annotations**
3. **Full Annotations**

Marker Annotation: Like the marker interface, a marker annotation does not contain any elements except the name itself. An annotation whose members all have default values can also be used as a marker since no information must be passes in.

Example

```
public @interface TestAnnotation
{
}
```

Usage:

```
@TestAnnotation
public void display()
{
}
```

SingleValueAnnotations: This type of annotations provides only single value. It means that these can be represented with the data and value pair or we can use the shortcut syntax (just by using the value only within the parenthesis).

Example

```
public @interface TestAnnotation
{String value();
}
```

Usage:

```
@TestAnnotation("Hi! How r u…")
public void display()
{
}
```

Example

```
import static java.lang.System.*;
import java.lang.reflect.*;
import java.lang.annotation.*;
@Retention(RetentionPolicy.RUNTIME)//meta annotation
@interface TestAnnotation//userdefined(custom) annotations
{String  value();
}
class One
{       //@TestAnnotation("Hi! How r u…")//short cut
        @TestAnnotation(value="Hi! How r u…")
        public void display()
        {       out.println("Welcome ....");
        }
}
class AnnDemo
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
{       public static void main(String args[])throws Exception
        {       Class c=One.class;
                Method ms[]=c.getMethods();
                for(Method m:ms)
                {       if(m.getName().equals("display"))
                        {       //getAnnotation() returns TestAnnotation object if method m is
                                annotated with that annotation
                                //otherwise this method returns null
                                TestAnnotation ta=m.getAnnotation(TestAnnotation.class);
                                out.println(ta);
                        }
                }
        }
}
```

Note: If a Target meta-annotation is not present on an annotation type declaration, the declared type can be used on any program element. If no Retention annotation is present on an annotation type declaration, the retention policy defaults to RetentionPolicy.CLASS.

Full Annotations:

These types of annotations can have multiple data members. Therefore use full value annotations to pass the values to all the data members... Here parentheses follow the annotation name, and all members are assigned with values.

Example

```
@interface TestAnnotation
{       String showSomething();
        int num() default 1;
        String name() default "Appu";
}
```

Usage:

```
@TestAnnotation(showSomething="Hi! How r u…",num=10,name="Madhu")
public void display()
{
}
```

Example

```
import static java.lang.System.*;
import java.lang.annotation.*;
import java.lang.reflect.*;
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@interface MyAnnotation
{int count()default 1;
}
class One
{       @MyAnnotation(count=5)
        public void display()
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
        {out.println("Hai");
        }
}
class AnnoDemo2
{       public static void main(String args[])throws Exception
        {       Class c=Class.forName("One");
                Method m=c.getMethod("display");
                MyAnnotation ma=(MyAnnotation)m.getAnnotation(MyAnnotation.class);
                for(int i=1;i<=ma.count();i++){
                m.invoke(c.newInstance());
                }
        }
}
```

Rules Defining The Annotation Type: Here are some rules that one should follow while defining and using annotations types.

- Start the annotation declaration starting with the symbol "at" @ following the interface and using annotations types.
- Method declaration should not throw any exception
- Method declaration should not contain any parameters
- Method using annotations should return a value, one of the types given below:
    - String
    - Primitive
    - Enum
    - Class
    - Array of the above types

Jdk 5(Tiger) contains two types of annotations

- **Simple Annotations**
- **Meta Annotations**

Simple Annotations: Jdk 5 includes three types of simple annotations

- **@Deprecated**
- **@Override**
- **@SuppressWarnings**

@Deprecated: The @Deprecated annotation indicates that the marked class or method or field is deprecated and should no longer be used. The compiler shows a warning whenever a program uses deprecated element.
@Override: The @Override annotation informs the compiler that the method is going to override a method declared in a base class. The compiler shows an error if a method marked as @Override is not

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

correctly override method of super class. (Compile the below example by changing the name of the toString() method then you can understand)

AnnDemo1.java

```
import static java.lang.System.*;
@Deprecated
class One
{	int a;
	One(int a)
	{this.a=a;
	}
	@Override
	public String toString()
	{return "a:\t"+a;
	}
}
class AnonDemo1
{	public static void main(String args[])
	{	One o=new One(100);
		out.println(o);
	}
}
```

@SuppressWarnings: Suppress Warnings is used to suppress compiler warnings. You can apply @SuppressWarnings to types, constructors, methods, fields, parameters, and local variables.

SupressWarningDemo.java

```
import java.io.*;
import static java.lang.System.*;
class One<T>
{	T a;
	One(T a)
	{this.a=a;
	}
	void display()
	{out.println(a);
	}
}
class SupressWarningDemo
{	@SuppressWarnings({"unchecked","deprecation"})
	public static void main(String args[])throws Exception
	{DataInputStream dis=new DataInputStream(System.in);
	out.println("Enter any value");
	one o1=new One(Integer.parseInt(dis.readLine()));
	o1.display();
	}
}
```

Meta-Annotations

There are 4 types of Meta-Annotations that would be covered in the forthcoming sections are

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

- **Target Annotations**
- **Retention Annotations**
- **Documented Annotations**
- **Inherited Annotations**

Target Annotation

For example, if you want to apply @TestAnnotation annotation to only methods, then it is said to be Meta-Annotation (meta-data about meta-data). For example the following is the declaration of the @TestAnnotation annotation along with some meta-data that states the elements that this annotation can be applied to.

```
@Target(ElementType.METHOD)
public @interface TestAnnotation
{// Property Definitions here.
}
```

From the above, we can see that the Annotation @TestAnnotation is annotated with @Target. This kind of annotation chaining is always possible. The target element tells that the @TestAnnotation annotation can applied only to methods and not to any other element types. The argument to @Target Annotation can be one from the possible set of values of any Java Element, which is defined in a well-defined Enum called ElementType. Here are the possible values taken by this Enum.

Value	Description
TYPE	Applied only to Type. A Type can be a Java class, Java interface or an Enum or even an Annotation.
FIELD	Applied only to Java Fields (Objects, Instance or Static, declared at class level)
METHOD	Applied only to methods
PARAMETER	Applied only to method parameters
CONSTRUCTOR	Applied only to constructor of a class
LOCAL_VARIABLE	Applied to only local variables
ANNOTATION_TYPE	Applied only to Annotation Types.
PACKAGE	Applicable only to a package.

Retention Annotation:

Another commonly used Meta-data for an Annotation is the Retention Policy. Assume that we have some annotations defined in the source code. We have a mechanism through which we can say that to what extent the annotations should be retained. The three possible ways of telling this are.

- Retain the annotation in the source code only
- Retain the annotation in the Class file also
- Retain the annotation Definition during the Run-time so that JVM can make use of it.

The Annotation that is used to achieve this is @Retention and it takes possible values of **SOURCE**, **CLASS**, and **RUNTIME** defined in **Retention Policy** Enumeration.

|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.CLASS)
 public @interface TestAnnotation
 {// Property Definitions here.
 }
```

Documented Annotations:

Documented annotation is existed in the package called "**java.lang.annotation**", and it is used to annotate an annotation. If you annotate an element by an annotation which is annotated by **@Documented** annotation, the annotation will be displayed in the document created by javadoc like tool.

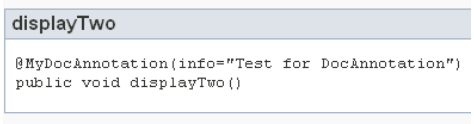
Example: Test.java

```
import java.lang.annotation.*;
import static java.lang.System.*;
@Documented
@interface MyDocAnnotation
{String info();
}
@interface MyNonDocAnnotation {
String nonInfo();
}
public class Test
{       public static void main(String arg[])
        {       Test t=new  Test();t.displayOne();
                t.displayTwo();
        }
        @MyNonDocAnnotation(nonInfo="Test for non doc annotation")
        public void displayOne()
        {       out.printf("Testing NonDocumented annotation\n");
        }
        @MyDocAnnotation (info="Test for DocAnnotation")
        public void displayTwo()
        {       out.printf("Testing 'Documented' annotation");
        }
}
```

Step-3: create docuentatin for: **Test.java**

Command to create documentation is:- javadoc Test.java

In the above example after creation of documentation for **Test.java** file, open the **index.html** file, then you will find your "displayTwo()" is annotated by "@MyDocAnnotation ".



|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet),
Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

Inherited Annotation: Java enables you to specify whether the annotations should be inherited in subclasses or not, while defining your annotations **java.lang.annotation.Inherited** annotation is used for this purpose. So, if you define your annotation as in the following snippet, you will find them in your subclasses, too.

To explain about an **@Inherited** annotation listen to my story, suppose we have an annotation(@MyAnnotation) which is annotated by **@Inherited** annotation. If you annotate @MyAnnotation annotation to any type(Base class), it will reflect to Base class and the child classes of Base class. If your (@MyAnnotation) annotation is not annotated by the @Inherited annotation it will not reflect to child classes of Base class. It will reflect to only to a class to which @MyAnnotation is annotated.

Example: MyAnnotation.java

```
package mts;
import java.lang.annotation.*;
@Inherited
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface MyAnnotation
{       boolean isInherited() default true;
        String show() default "Madhu Tech Skills";
}
```

Base.java

```
package mts;
@MyAnnotation
public class Base{
}
```

Note: @MyAnnotation Annotation is reflected to Child class because **@MyAnnotation** annotation is annotated by **@Inherited** annotation.

Child.java

```
package mts;
public class Child extends Base{
}
```

Usage.java

```
package mts;
import static java.lang.System.*;
import java.lang.*;
import java.lang.annotation.*;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
public class Usage
{       public static void main(String args[])
        {       Base b=new Base();
                Class<?> cls = b.getClass();
                MyAnnotation ann1= cls.getAnnotation(MyAnnotation.class);
                out.println(ann1.show());
                Child c=new Child();
                MyAnnotation ann2=c.getClass().getAnnotation(MyAnnotation.class);
                out.println(ann2.show());
        }
}
```

Custom Annotations

Annotations created by us are said to be custom annotations. Observe the below example which shows how to create a custom annotations and the usage of it.

Creation and Usage of Custom Annotations

Call.java

```
package mts;
import java.lang.annotation.*;
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Call
{int noOfTimes() default 1;
}
```

Use.java

```
import java.util.*;
import mts.Call;
import static java.lang.System.*;
import java.lang.reflect.*;
class Demo
{       @Call(noOfTimes=5)
        public void one()
        {out.println("one");
        }
        @Call
        public void second()
        {out.println("Second");
        }
}
public class Use
{       public static void main(String args[])throws Exception
        {       Class c=Demo.class
                Method cm[]=c.getMethods();
                Demo d=new Demo();
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
            for(Method  m:cm)
            {
                  //out.println(m.getName());
                  Call u=m.getAnnotation(Call.class);
                  if(u!=null)
                  for(int i=1;i<=u.noOfTimes();i++)
                  {//m.invoke(c.newInstance());
                     m.invoke(d);
                  }
            }
      }
}
```

Enum Data Types

It is a new data type which is introduced in java 1.5 verson. Enum type is a type which consists of fixed set of constant fields like direction, days and months etc... Since they are constants we will take these values as uppercase letters.

Note:

At the time of compilation if the compiler encounters an enum type, it generates a class that extends the **java.lang.Enum** class.so all the Enums implicitly extend **java.lang.Enum**.

In java we define an enum type by using the enum keyword. For example we can specify the days-of-the-week enum type as:

```
public enum Days{
    SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATAURDAY
}
```

Example: EnumDemoOne.java

```
import static java.lang.System.*;
enum Days{
    SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATAURDAY
}
public class EnumDemoOne
{       public static void main(String args[])
        {       Days day=Days.SUNDAY;
                out.println(day);
                out.println(Days.SUNDAY);
        }
}
```

Example: EnumDemo2.java

```
import static java.lang.System.*;
enum Days{
    SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY
}
class Test
```

```
{       Days day;
        Test(Days day)
        {this.day=day;
        }
        public void details()
        {       switch(day)
                {       case SUNDAY:out.println("Sunday is Holidy");
                                break;
                        case MONDAY:out.println("Monday is Mad day");
                                break;
                        case SATURDAY:out.println("Saturday is Joly day");
                                break;
                        default:
                                out.println("Middle Week days are hot hot");
                                break;
                }
        }
}
public class EnumDemo2
{       public static void main(String args[])
        {       Test t1=new Test(Days.MONDAY);
                t1.details();
                Test t2=new Test(Days.SATURDAY);
                t2.details();
                Test t3=new Test(Days.SUNDAY);
                t3.details();
        }
}
```

Note:

The compiler automatically adds some special methods at the time of **Enum** creation. For example, they have a static method called **values()** that returns an array containing all of the values of the enum in the order they are declared.

Example: EnumDemo3.java

```
import static java.lang.System.*;
enum Days{
    SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATAURDAY
}
public class EnumDemo3
{       public static void main(String args[])
        {       out.println("Elements in an Enumeration");
                for(Days day:Days.values())
                {out.println(day);
                }
        }
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
}
```

Note:

While using Enums we have to declare the constants first, prior to any fields or methods. Also, when there are fields and methods, the list of enum constants must end with a semicolon.

Note:

The constructor for an enum type must be **package-private** or **private access**. It automatically creates the constants that are defined at the beginning of the enum body. You cannot invoke an enum constructor yourself.

Example: EnumDemoFour.java

```
import static java.lang.System.*;
enum Days
{       SUNDAY(1),
        MONDAY(2),
        TUESDAY(3),
        WEDNESDAY(4),
        THURSDAY(5),
        FRIDAY(6),
        SATAURDAY(7);
        private int i;
        Days(int i)
        {this.i=i;
        }
        public int getI()
        {return i;
        }
}
public class EnumDemoFour
{       public static void main(String args[])
        {       Days day=Days.MONDAY;
                Out.println(day.getI());
                day=Days.SUNDAY;
                Out.println(day.getI());
        }
}
```

Underscore in numeric literals

```
int one_million = 1_000_000;
```

Example: Underscore.java

```
import static java.lang.System.*;
class Underscore
{       public static void main(String args[])
        {       int one_million=100_00_00;
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.

```
                out.println(one_million);
        }
}
```

Binary Literals: In Java we can assign, binary value directly to an integer variable. But before, that binary value we have to put "ob"(Zero and b), then only java compiler and JVM treats that number as binary and converts it into decimal after that assigns it to the variable.

```
int binary = 0b1001_1001;
```

Example: BinaryLit.java

```
import static java.lang.System.*;
public class BinaryLit
{               public static void main(String args[])
                {       int a = 0b0000_1010;
                        out.println(a);
                }
}
```


|Dno: 39-8-39, Behind Hotel Gatewy, Paidaiah Street (Goodluck Teas centerStreet), Labbipet, Vijayawada-10, Ph.No:99489 63306, 0866-6640526.