Advanced Programming (Java)

1995 James Gosling — Sun Microsystems

High level language.

3 categories of Java.

J2SE Standard Edition
Application based.

J2EE Enterprise Edition
socket programming
Servlets
Server-side programming
Networking with Java.

J2ME Mobile Edition

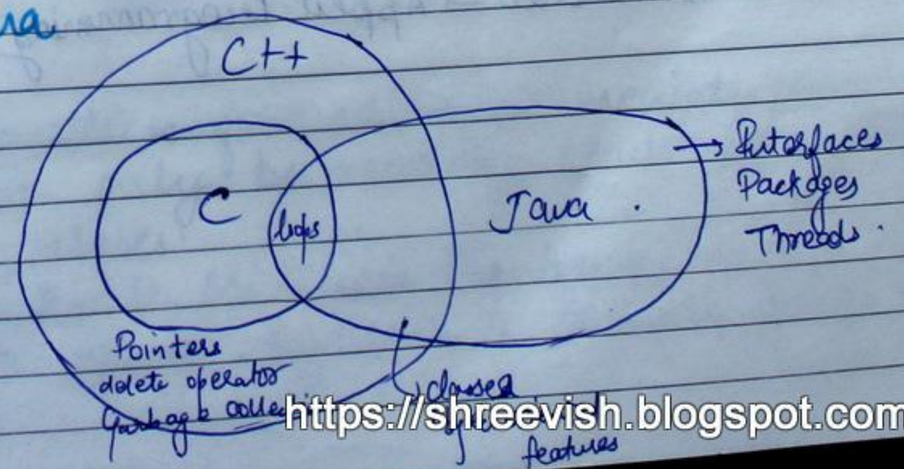
latest → Java 2
Java 1.6.

Java vs C++

C++ → semi object oriented.

Pure object oriented.
fully based on classes.

C, C++, Java



Java Development Kit (JDK).
It comes with a collection of tools that are used for developing & running Java programs.

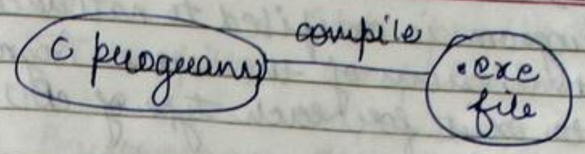
windows → Java compiler
↳ Java Interpreter
web based → Applet viewer.

2 Types of Applications can be developed by it.
- Web based
- system control Based.

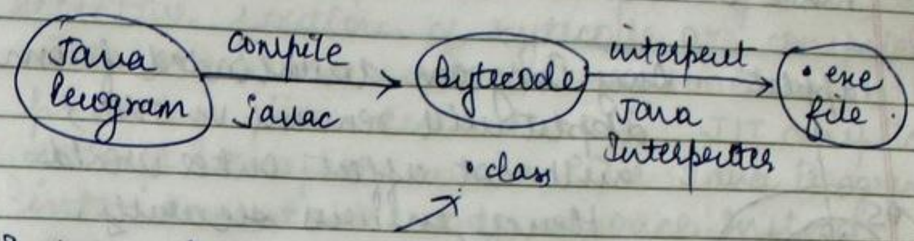
Java compiler:- It translates Java source code to byte code that the interpreter can understand.

Java Interpreter:- It runs applications by reading & interpreting byte code files.

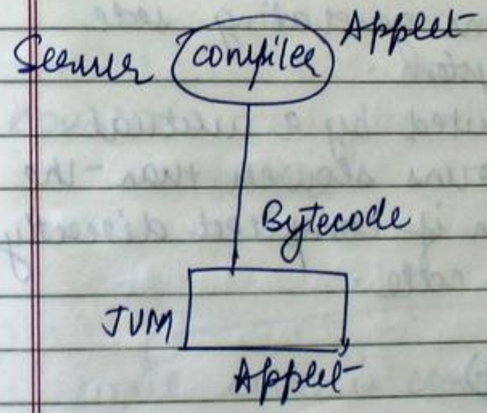
Applet viewer:- It helps to run Java Applets without actually using a Java compatible browser.
client side → Applet programming.



Platform Dependent



Intermediate stage can be run on Multiple systems.
- Platform independent
- Security More.

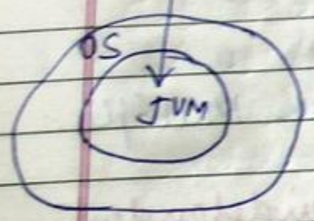


Bytecode is formed → intermediate code.
can only be read by Machine so security increases.
Bytecode Minimum requirement.
JVM convert Bytecode to Machine specific code.

Bytecode, JVM, JIT

If a Java Program is compiled to native code then different versions of the same Program would have to exist for each type of CPU connected to Internet.
A Java Programs executed by JVM also helps to make it secure.

applet — Any Program downloaded from applet will remain in JVM, will not affect outer world. Hence ensures security.



Because JVM is in control. It can contain the program & prevent it from generating side effects outside of the system.
When a program is executed by a virtual Machine, it generally runs slower than the same Program could run if compiled directly to executable Machine code.

source → Machine. (fast)

source → Bytecode → M/C code. (slow because)

JVM

↓

JIT compiler

It is possible to use on the fly compilation of byte code into M/C code in order to boost the Performance. This is done through the use of "Just In Time" compiler for bytecode. When JIT compiler is the part of JVM, selective portions of bytecode are compiled into executable code in realtime on a piece by piece demand basis. JIT compiler doesnot compile an entire Java Program into executable code at once. Instead JIT compiler compiles the code as it is needed during execution. The remaining code is interpreted.

slowness is due to use of interpreter.

sourcode → Bytecode → M/c.
compiler — Interpreter & JIT

some code is compiled & some code is interpreted.

Eg:- 180 lines → 90 compiles
90 interpretes.

JDK.

Tools
Javac
Java
Appletviewer

JSL
(Java standard library)
Predefined classes & Methods.

JSL stands for Java standard library.
It is part of JDK. It contains all the predefined classes & Methods grouped into packages.

Default Package → Java.lang
↳ Basic I/P, O/P & printing.

Package
↑
class
↑
Method

1) Language support Packages.
It is a collection of classes & Methods required for implementating basic features of Java (I/P, O/P).

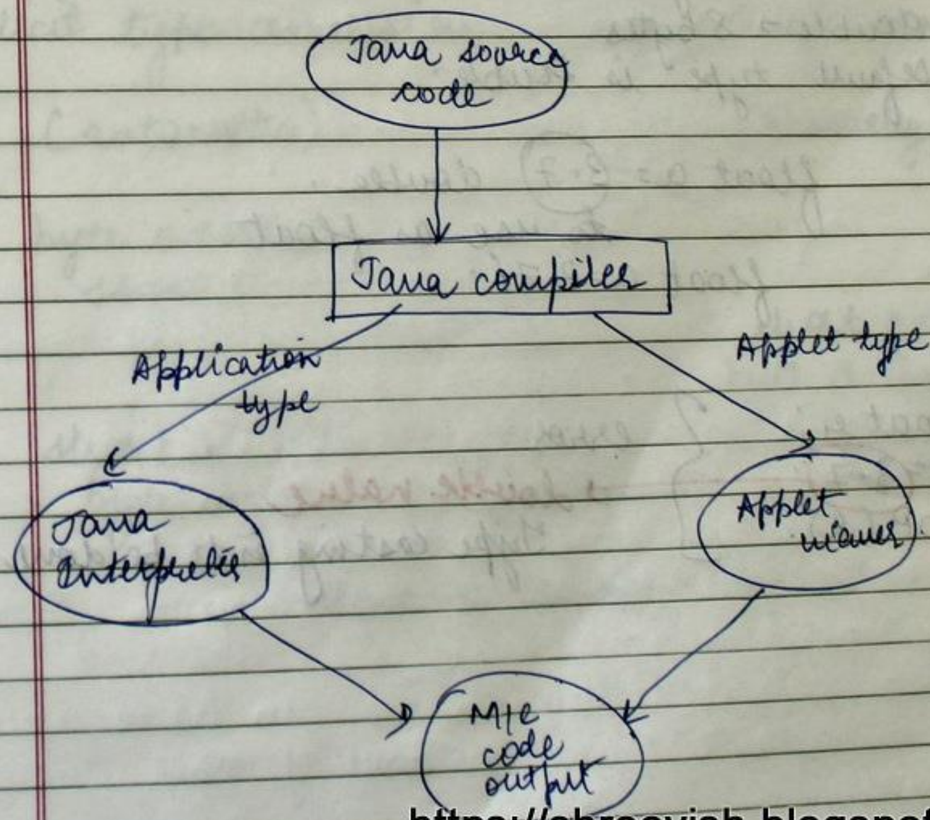
2) Networking Packages.
socket programming
It is used for networking purpose.

3) AWT Packages :- Abstract window toolkit :-
It contains classes to implement graphical user Interface (GUI). It contains set of classes used to create Java Applets.

4. Applet Packages.

Java is used to develop two types of program.
1) Standalone Applications.
2) web Applet

- standalone applications are the programs written in Java carry out certain task on a local computer.
- web Applets :- Applets are the small Java programs developed for internal Applications.



Data types in Java

int a
Type of data
Amount of data

whole no's:

int → 4 bytes
long → 8 bytes
byte → 1 byte
short → 2 byte
default type in Java is short

Decimal No.

float → 4 bytes.
double → 8 bytes
default type is double.

float a = 3.7 double.
to use as float
float a = 3.7f.

float a;
a = 3.7;
SOP(a);

} error → double value
Type casting is to be done.

Character data type:

C++	Java
1 byte	2 bytes
256 ASCII codes	65536 unicodes.

String object
Str → seperate class.
Char a[10]

Type casting → Implicit type conversion
Explicit type conversion.

Implicit type conversion (Automatic)

byte c = 120;
short d = c;
small → big

short c = 234; → not possible
byte d = c;

Explicit type conversion (Manually)

short c = 234;
byte d = (byte) c;
big → small

Float to double → implicit.

Write a Java program to add 2 numbers :-

```
import java.lang.*;
class Add
{
  public static void main(String s[])
  {  int a,b,c;
     a=10;
     b=20;
     c=a+b;
     System.out.println(c);
  }
}
```

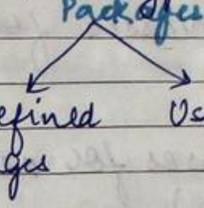
Cammelline conversion → addSubtract();

used in predefined Methods of Java.

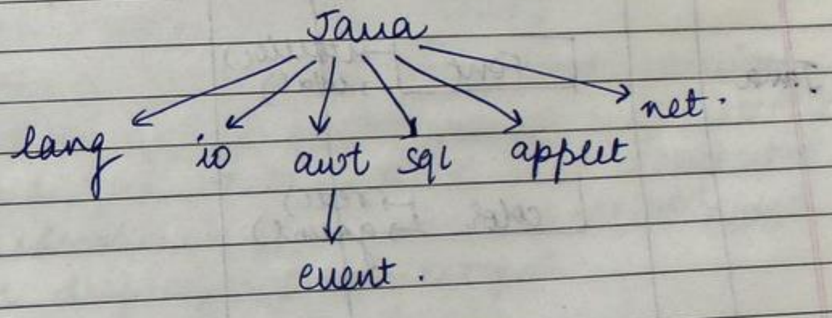
system Package
import java.lang.*;
↓ ↑ ↓
Package all classes of Packages

Import statement in Java is similar to # include in C++. This statement instructs the compiler to load the mentioned classes in the program from the mentioned package. using import statements, we can have access to classes that are the part of other Packages.

Java provides large number of classes grouped into different Packages according to their functionalities.
There are 2 types of Packages in Java.



System Defined.



java.lang Java.lang is the default package and not required to be imported. It is automatically imported & contains classes for datatypes, strings, input output function, math functions, threads etc.

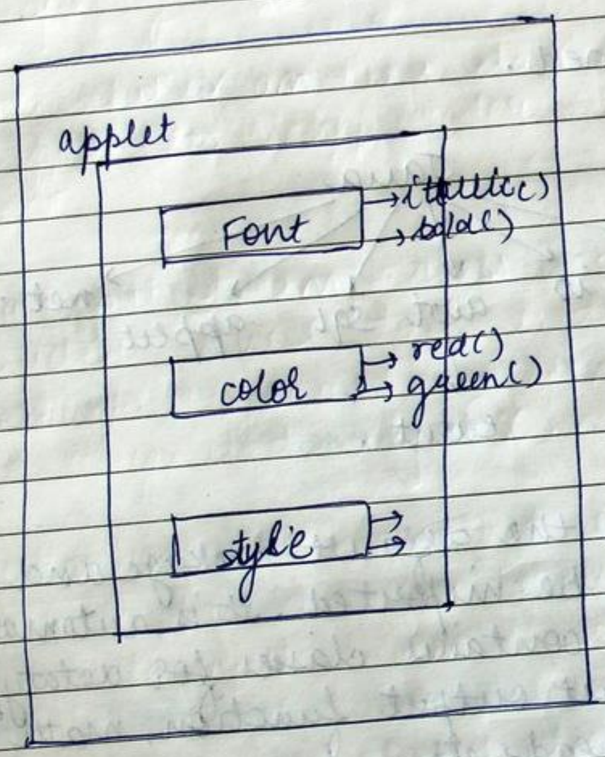
java.io Java.io. This package supports Input output classes.

java.awt This Package contains set of classes for implementing graphical user Interface (GUI) checkboxes etc.

java.applet :- This Package provides classes & Methods for creating and implementing applets.

java.sql :- It contains classes for database connectivity.

java.net :- It contains classes for networking (socket programming).



java.applet.Color; Color class can be used.
java.applet.*; Font, color, style can be used.

public static void main

Main The execution of any program starts from main method. It must be declared public because it is executed by the code outside the class. It is the Java interpreter that invokes the Main Method.

static It allows main Method to be executed independent of any object. This is necessary because main Method is executed by JVM before any objects are made.

System.out.println (SOP)
System is predefined class that provides access to the system stream
out is the output stream that is connected to console & println is the method used for displaying the output.

Write a program to add 2 numbers through keyboard

```
import java.io.DataInputStream;
class Add
{ public static void main
    PSVm (String s[]) throws Exception
    { int a,b,c;
DataInputStream x = new DataInputStream (System.in);
    SOP ("enter first no");
a = Integer.parseInt (x.readLine());
SOPt "enter second no");
b = Integer.parseInt (x.readLine());
```

```
        c=a+b;
        SOP ("result is" +c);
    }
}
```

[→ part of default package java.lang.

readline() → to read data to take I/P.
we need a reader to use it, so io package is required.
object of readline will be in java.io.

DataInputStream → is a class in java.io. which provides the reader.]

DataInputStream :- class is used to create a reader whose task is to read the data through keyboard by using readline Method. and assign it to object 'x'.

System.in is a class that contains all the read Method. This class can be used as parameter for data input stream class.

[DataInputStream(System.in)

System.in
↓
readline() → DataInput stream helps to bring it to the program

class [DataInputstream] ← can't be directly accessed so need a reader which is present in DIS class.
↓
[reader]
[System.in class]
↑
[readline()] Method.

[if import java.io.*; → compilation time increases, but the classes of io can be accessed.]

Indirectly we are accessing System class through object x of data input stream class. DataInputStream is a class that creates a reader which helps to access system.in through which all the read Methods can be accessed.

[Throws Exception :-
to handle default Exceptions.)
system generate these default exceptions. (Pre-defined Exceptions).
Main will handle them.]

This is always used when we have to enter data through keyboard.
Exam # in whatever method, readline() is declared, that function or method will handle Exception.

Integer.parseInt()

by default, nfrateres entered from keyboard is taken as string.

a = 10
b = 20

c = 1020
contatonation.

a = Integer.parseInt(br.readLine())
int ↑ int ↑ string (10)
int

Integer ← Wrapper classes.
Float

Buffer → More efficient than DataInputStream.

Wrapper classes :-

are in java.lang, part of default Package.

Each of Java's primitive data types has a class dedicated to it. These are known as wrapper classes because they wrap the primitive data type into an object of that class. Wrapper classes are the part of java.lang (default) Package.

Primitive data type	Wrapper class
Byte	Byte
Short	Short
int	Integer
long	Long
float	Float
double	Double

wrapper classes are helpful in conversion from one type into another type.
They are used to convert →

1. Primitive type to object Type.
2. object type to primitive type.
3. Primitive type to String.
4. String to primitive.

String to Primitive Type
Using wrapper classes

Data entered through keyboard is in the form of string int, float, long.

parse method parsexxx( ) in wrapper class is used to convert the string to primitive data type.

Integer.parseInt()

Eg: parseInt() → it is static or dynamic?
static as no object is created.
x.parseInt() → x it is directly accessed.
method called without use of object is static)

* Parsexxx() is a static Method. It is not called through an instance of the class.

Primitive to string using wrapper classes.

toString (value that needs to be converted to string)

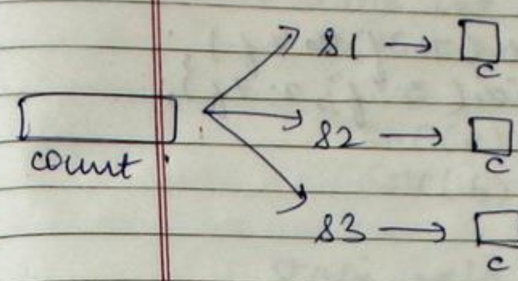
eg :- String s = Integer.toString(1234);
It converts int to string object and returns a reference to the object.

"Static Methods"

Sometimes we want define a class that will be used independently of any object of that class. Normally a class member must be accessed through an object of its class. However, it is possible to use the members of a class without creating an instance of that class. To create such a member 'static' keyword is used.

When a member (variable, functions) is declared as static, it can be accessed before any object of its class are created. Both the variables & methods can be declared as static. Since static members are associated with the class rather than individual objects, the static variables & methods are also referred to as class variables or class members.

C → non static
count → static



Instance variable → C
class variable → count
↳ existence of s1, s2 & s3 existence for whole class.

called through object so non-static

class object
System.out.println(); → non static
Math.sqrt(); → static
↳ without any object called

```
class xyz
{ static float mul(float x, float y)
    { return (x*y);
    }
  static float divide(float x, float y)
    { return (x/y);
    }
}
```

```
class demo
{ psvm ( )
  { float a,b;
    a = xyz.mul (7.3f, 4.7f);
    b = xyz.divide( 3.5f, 2.7f);
    sop(a);
    sop(b);
  }
}
```

if nonstatic then →

```
class demo
{ psvm ()
  { float a,b;
    xyz x = new xyz ();
    a = x.mull 7.3f, 4.7f);
    b = x.divide( 3.5f, 2.7f);
  }
}
```

* static can be accessed through object also.

Restrictions with static Methods :-

1. ~~Static~~ variables can be accessed directly without object inside a non-static method.
2. Non static variables cannot be accessed directly inside a static Method.
3. Static Methods can only call other static Methods directly. It is illegal to refer to any instance variable inside a static Method without object.

```
class static_method
{ int a=29;
  static int b=30;
  void display ()
  {
    sop(a);
    sop(b);
  }
  static void displaystatic
  { sop(a);
    sop(b);
  }
  P.S.V.M (-)
  { static_method cl = new static_method ();
```

Can be accessed directly ✓ display_static();

Object through can be done ✓ cl.display_static();

✓ static_method.display_static();

```
    display();
    static_method.display();
    cl.display();
  }
}
```

Rule 2.x. should be called through object

sop(a) ⇒ s.i.a.

Rule 2 → x

x

Java ✓

MST

class
Object
Constructors.
Java vs c/c++
Features of Java
No Destructors are there in Java, Garbage collection is there.

Local variables :-

↳ local variable

```
void display (int z)
  {

  }
```

Variables defined inside the Methods. The variables will be declared & initialized within the Method.

Instance variables :-

These variables are defined within a class but outside any Method. These variables are initialized when the class is loaded.

```
class Halli
{ int a=29;

}
```

class variables :- are declared within a class outside any Method with static keyword. They can be called without any object

```
class static_method
{
static int b=30;
}
```

Overloading / Polymorphism

In C++.

1) Method overloading.
2) Operator overloading.

Java.

Method overloading.

Does not support operator overloading.

In C

Function abs().

If its values in (-ve) abs() used to make it the.

for float → fabs()

for long → labs().

C++ / Java.

abs() for int, float, long.

abs()

if parameter for integer then abs() is used.

one function is used for three.

this is called Method overloading (same name different parameters) in C++ / Java.

* Multiple Methods of same name with different arguments in a single class is called Method overloading.

It is used when objects are required to perform similar tasks but using different parameters.

when we called a method in an object java matches the method name first & then the number and the type of parameters to decide which of the definitions to execute.

This is also called polymorphism.

while overloaded methods may have different return type.
The return type long is insufficient to differentiate between 2 versions of ~~long~~ method. When Java encounters a call to an overloaded methods, it simply executes the version of the methods whose parameters used in the call.

```
class overloaddemo
{ void test
  {
  SOP("no parameters");
  }
  voidtest (int a)
  {
  SOP("a" + a);
  }
  void test (int a, int b)
  { SOP("a" + a);
    SOP("b" + b);
  }
  double test (double a)
  { SOP("double" + a);
    return (a * a);
  }
}.
```

```
class Demo
{ PSUM(-).
  {
  double rul;
  overloaddemo d:
  new overloaddemo();
  d.test;
  d.test (10);
  d.test (10,20);
  rul = d.test (123.45);
  SOP("rul" + rul);
  }
}.
```

A compiler error would occur when 2 methods with the same parameter list but different return types are created.

visibility controls in Java (access Modifiers)

4 types of visibility modifiers can be used in Java Program.

Package
↓
class
↓
subclass
↓
Method.

access location \ access modifier	Public	Private	default	Protected
same class	✓	✓	✓	✓
Subclass in same pkg.	✓	×	✓	✓
other class in same pkg.	✓	×	✓	✓
subclass in other pkg.	✓	×	×	✓
other class in other pkg.	✓	×	×	×

Public :- A method or variable declared as public is accessible to the class in which defined, all subclasses of that packages, all classes in same package, any other class outside that pkg.

Private :- only 2 other methods in the same class, not in subclass, not in other classes in same pkg, not in any class outside the pkg.

Protected :- Accessible to i) same class ii) any subclass in the same package iii) any class to the same package iv) from a sub class outside the same package v) not by any class outside the class.

default :- i) All the other classes in the same method,
ii) not accessible outside that package.
iii) All the subclasses in the same pkg.

command line

```
import java.io.DataInputStream;
class AddDemo {
    public static void main(String s[]) throws Exception
    {
    int a,b,c;
    DataInputStream x = new DataInputStream(System.in);
    a = Integer.parseInt(s[0]);
    b = Integer.parseInt(s[1]);
    c = a+b;
    SOP("result is "+c);
    }
}
```

javac AddDemo.java.
java AddDemo 7 5 ↵.
12.

C++
int a[10];

Arrays in Java.

How to declare an array in Java?.

Java type arrayname [] = new type [size]

eg:- int a[] = new int [10];
↓ memory allocate hogi is us.
a used as object.
↳ reference variable.

The variable 'a' holds a reference to memory allocated by new operator. This memory is large enough to hold 10 elements of type integer.

bound - checking.
- In C, C++, bound-checking is not possible.

```
int a[10]
for(i=0 to 9)
    {, printf(a[i]);
```

In C++, Garbage values
In Java, 0.
initialized as 0 in Java.

10 elements declared.
→ we enter 20 elements c++ possible.

int a [10]
for (i to 20)
{ a[20].

a[10] -
i=0 to 20.
↳ 10.
(11) stop.

a.s can be dynamically changed.

array size remains constant

Java.

1. size remains constant
2. Bound checking is possible
3. Java initialise elements to 0.

C++

1. size can be resized dynamically.
2. Bound checking is not there.
3. initialised to garbage values.

To calculate the length of an array, a special variable called length variable is used.
a. length

ArrayIndexOutOfBounds Exception.
5 element
as we enter 6th element, This Exception occurs.

Write a Program to combine 2 Arrays into 3rd Array where the first & 2nd array contains 10 elements each.

WAP to count no. of even & odd elements in an array.

```
class ArrayDemo
{ Public static void main (String s[]).
  { int odd, even;
    int a[] = new int [10];
    SOP("Enter 10 elements");
    for ( i=0; i<10; i++).
    { a[i] = Integer.parseInt (a[i].readline());
      if (a[i]/2 == 0).
        even = even + 1;
      else
        odd = odd + 1;
    }
    SOP ("even=" + even);
    SOP ("odd=" + odd);
}
```

```
import java.io.DataInputStream;
class ArrayDemo
{ public static void main(String s[]) throws Exception.
  { int a[] = new int [10]; int odd, even;
DataInputStream x = new DataInputStream (System.in);
  SOP ("Enter 10 elements");
for ( i=0; i <= 10; i++) .
{a[i] = Integer.parseInt (x.readline ());
  if (a[i]%2 == = 0) .
        even = even + 1;
    else
        odd = odd + 1;
  }
SOP ("odd" = " + odd);
SOP ("even = " + even);
}.
```

Inheritance in Java :-

It is nice if we could reuse something that already exists rather than creating the same again. Java supports this concept of reusability in the form of inheritance. Mechanism of deriving an old class from an old one is called inheritance. It is a mechanism of sharing the members among different classes. constructors cannot be inherited by subclass but the constructors of superclass can be invoked from the subclass. Different type of inheritance:

1) Single Inheritance :- A superclass extends a subclass
2) Multiple Inheritance :- Multi superclasses extend a subclass.
3) Multilevel :- class A derives class B, class B derives class C.

Multiple Inheritance is not directly possible in java. we have to use interfaces to implement Multiple Inheritance.

Syntax to Implement Inheritance in Java :- keyword extends is used to inherit one class into another class.

```
class subclass extends superclass
  {
      variables
      methods
  }
```

Single inheritance :-

```
class superclass
{ public void supermethod()
    {
      SOP("superclass");
    }
}
```

```
class subclass extends superclas
{   Public void submethod()
      {   SOP("subclass");
      }
}
class Inherit
{
    PSVM (—)
    {  subclass b1 = new subclass();
       b1.submethod();
       b1.supermethod();
    }
}
```

example to justify that

```
class Room
{  int l,b;
   void room_method(int x, int y)
      {  l=x;
         b=y;
      }
   int area()
      {  return (l*b);
      }
}
class Room1 extends Room.
{  int h;
   Room1(int z)
      {  h=z;
      }
```

If we use this method as a constructor.

```
Room(int x, int y)
{  l=x;
   b=y;
}
```

```
int volume()
   {  return (l*b*h);
   }
}
class Inheritance.
{  PSVM (—).
   {  Room1 r1 = new Room1(30);
      r1.room_method(10,20);
      int a1 = r1.area();
      int v1 = r1.volume();
      SOP(a1);
      SOP(v1);
   }
}
```

If we want to call constructor of base class from derived class. We use super keyboard.

```
super(x,y);
h=z;
```

Super keyword:-

- super keyword with constructor :- constructor in the derived class uses the super keyword to pass the values that are required by base constructors.
- super may only be used within a subclass constructor method.
- The call to super class constructor through super keyword must appear as the first statement within the subclass constructor.

- The parameters in the super ~~class~~ call must match the order & the type of instance variables declared in the super class