In C

```
main()
{
  printf("Abhilipsa");
}
```

In C++

```
#include<iostream.h>
main()
{
  cout<<"Abhilipsa";
}
```

In Java

```
class Test
{
  Public static void main (String arg[])
  {
    System.out.println("Abhilipsa");
  }
}
```

println is a method of print stream class.

Ex:

```
class student
{
  int age;
  float mark;
  void initialize()
  {
    age = 5;
    mark = 20.57;
  }
  void show
  {
    System.out.println("age = "+age);
    System.out.println("Mark = "+mark);
  }
}
```



Scanned by CamScanner

```
class Test
{
    public static void main (String args[])
    {
        Student St = new Student();
        St.initialize();
        St.show();
    }
}
```

St.
age = 5
mark = 20.53
initialize();
show();

21/7/19 Q/ Why main method is public in Java?

→ main is public to access anything outside the class.

→ During execution of any java program, JVM calls the main method to start the execution. As JVM is present outside the class, main method needs to be public to provide access permission to JVM.

* → JVM comes into work during execution time.

→ If the main method is not public, the source code will be converted to byte code during compilation time but it will



Scanned by CamScanner

Q// Why main method is static?

As JVM is calling the main method before creating the object of the corresponding class to which main method belongs to which static keyword is used. Object once the control will enter the main method but the JVM's work will start before that so main method in Java is declared as static and JVM uses the class name to call the main method directly.

Q// Why the return type of main method is void in Java

→ To start the execution JVM calls the main method but in return it doesn't expect any value from the function defination so, the return type is specified as void.

Object Oriented Programming

4 different features makes are language as Object Oriented programming language

(1) Abstraction

(2) Encapsulation

(3) polymorphism

(4) Inheritance.



Scanned by CamScanner

① Abstraction
- It is a process of data hiding where the essential details are hidden to the outside world and the users are given their requirement.

② Encapsulation
- The grouping of datamembers and methods within a single block is known as encapsulation.
- This also helps in data hiding process

1/08/19.

③ Polymorphism
It means many forms ie. one object can behave differently at different instant of time known as polymorphism

It can be of 2 types :-
i) Compile time polymorphism
ii) Run time polymorphism

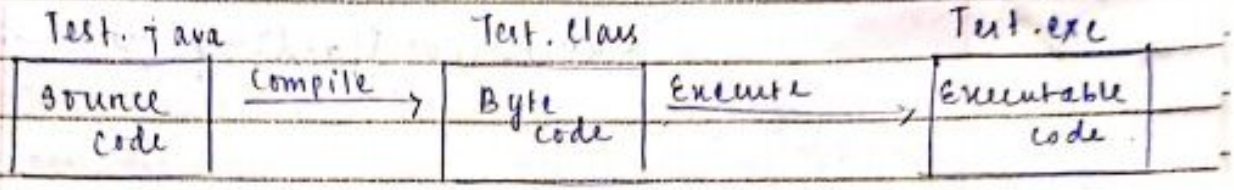
④ Inheritance
- Aquiring the properties from one existing class to new class is known as inheritance
- The existing class is known as base class or parent class
- The new class is known as derived or child class



Scanned by CamScanner

## Features of Java

(1) Object Oriented
(2) Platform independent (WORA)




- After sucessful compilation, the source code will be converted to its corresponding Byte code or known as .class file.
- Same byte code can be executed in any machine, irrespective of any operating system or environment.
- JVM is the destination machine is responsible to read the byte code and execute it.
- Java is popularly known as platform independent language as it follows the principle of WORA (Write One Run Anywhere).

(3) Portable
- As java is platform independent, it can shifted to any machine after compilation.

(4) Simple
- Java is very easy to simple learn and its syntax is simple and easy to understand.
- The syntax is simple because its syntax is based on C++.
- It has removed many confusing features like pointer, pre-processor directive, operator Overloading, templete as they are rarely used in programming language.



Scanned by CamScanner

- To deallocate memory, automatic garbage collection makes Java more simple.

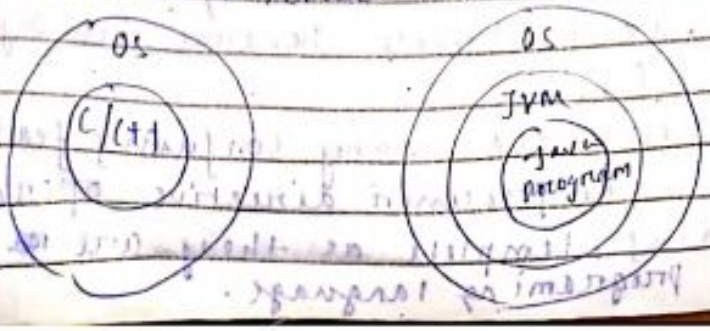
⑤ Robust — Memory Management
— Exception Handling Mechanism.

Java is considered as strong or Robust due to 2 different Mechanisms -
i) Memory Managment
ii) Exception Handling

Through Memory Management Mechanism the language will allocate or deallocate memory automatically as per the requirement of the use.

Similarly, different exception Handling Exception are followed to check diff. unhandled errors to provide normal termination to the program.

⑥ Secure
- Java is best known for its security
- It is secured beecz no explisit pointer is used and Java programs run inside a virtual machine.





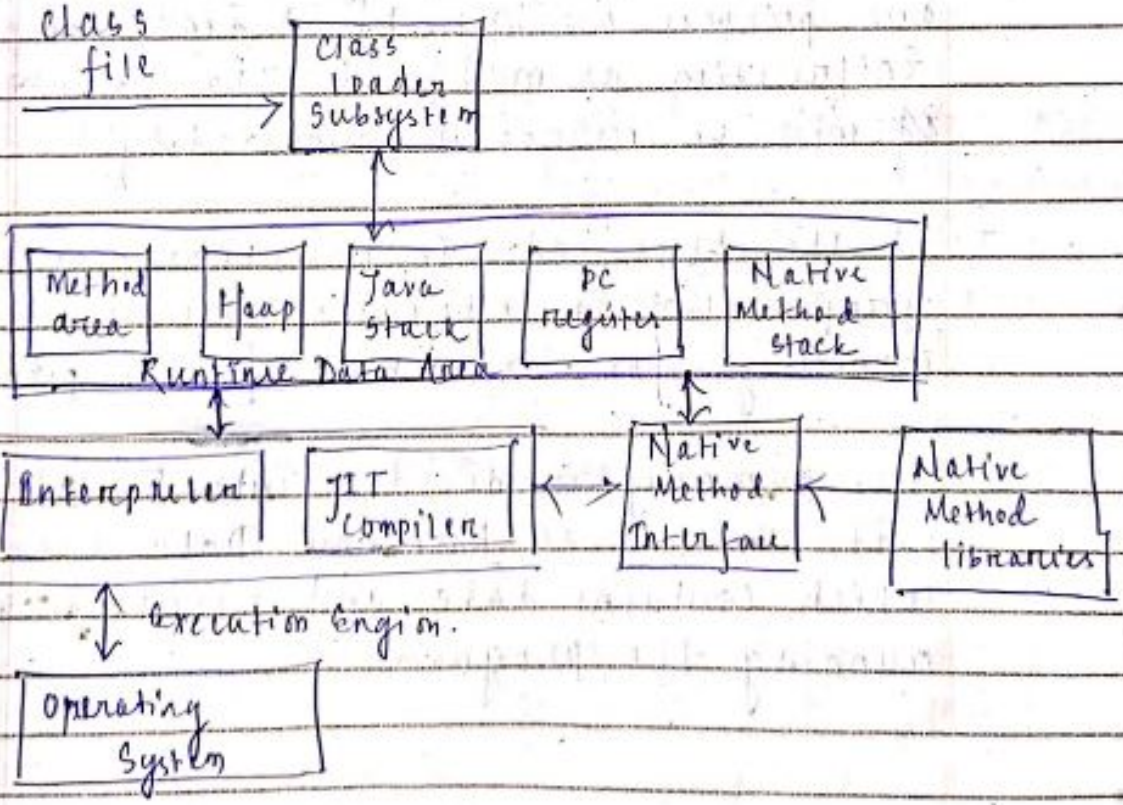

Scanned by CamScanner

7) Multithreaded

- 1 thread is like a separate program executing concurrently.
- Multiple tasks will run parallelly but doesn't occupy diff. resources on every thread.
- It shares a common memory area along with all memory resources allocated to the single thread. Important for web appl^.

5/08/19

## JVM (Java Virtual Machine)






Scanned by CamScanner

# Components of JVM Architecture

- JVM is the heart of entire Java program execution process.
- It is responsible for accepting the .class (.class) file or byte code and convert the instructions into machine language instruction.

## Class Loader Subsystem

- It loads the dof class file into memory. Then it verifies all byte code instructions are proper or not. If it finds any instruction as modified, the execution will be rejected immediately.
- If the byte code instructions are proper then it allocates necessary memory for execution of the program.
- This memory is divided into 5 different parts known as Runtime Data areas, which contains data and results while running the programs.



Scanned by CamScanner

## Method Area

- This memory block stores the class code, code of the variable and code of the method of the java program.
- It also allocates memory for static variables.

## Heap Area

Heap Area - In this area objects are created. This area is used for local variables and Arrays.

## Java stack

- While running a method, it need some more memory to store the data and memory. This is allotted from java stack area. Also JVM uses java stack for execution of different threads

## PC Register (Program Counter)

This Program counter Register is used to keep track of program execution. It keeps the address of next instruction to be executed.

## Native Method stack

Native Methods include C/C++ functions which are executed on native method stacks i.e the programmers can also write java program using C/C++ Syntax which is known as Native method Prog.



Scanned by CamScanner

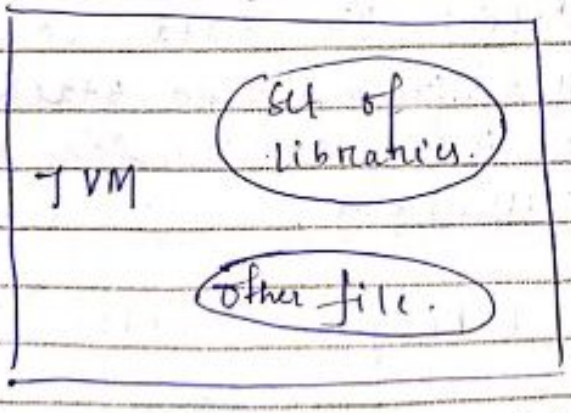
- To execute native methods, native method libraries are acquired.
- These loader files are located and connected to JVM by a program known as native Method stack.

## Interpreter

## JIT (Just In time)

7/8/19

## JRE (Java Runtime Environment).




- It is used to provide runtime environment to execute Java programming.
- It is the implementation of JVM including set of libraries and other files that JVM uses at run time



Scanned by CamScanner

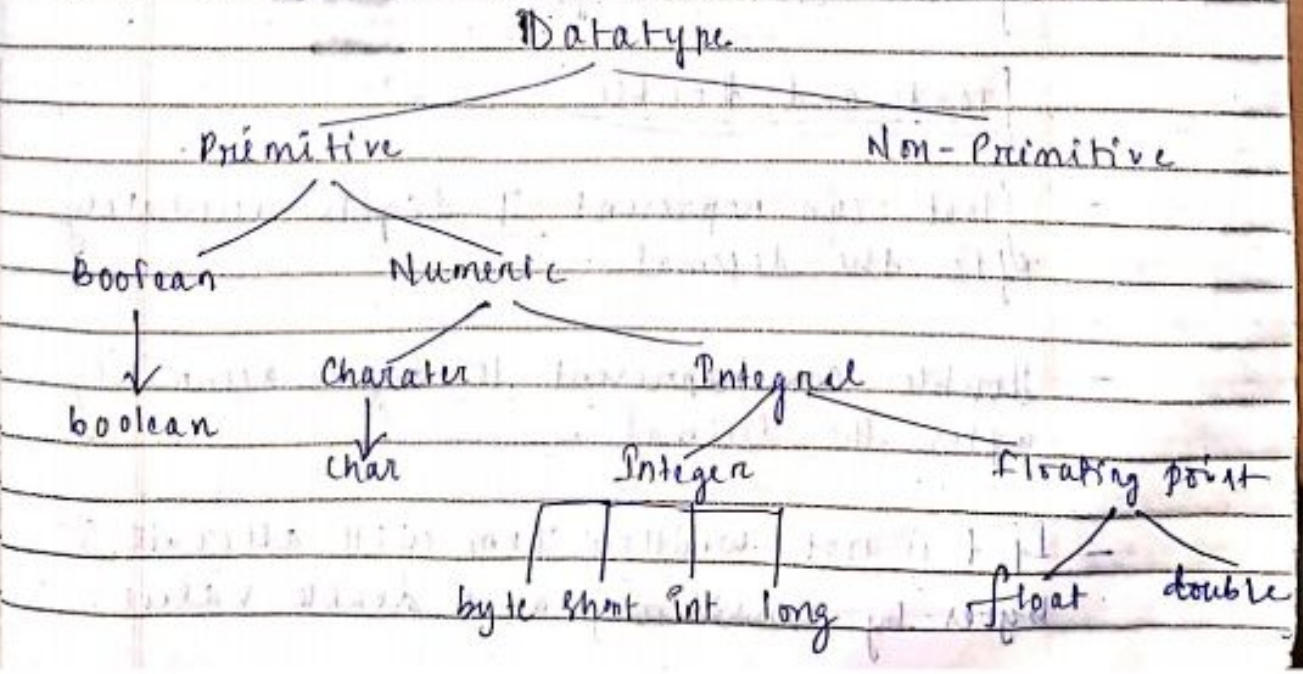
JDK (Java Development Kit)

| | |
|---|---|
| JRE | Development tools like javac, java etc. |

The Java development kit contains JRE along with development tools

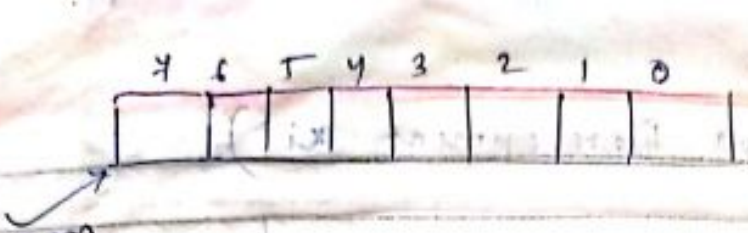
| Datatype | Default Value | Size | |
|---|---|---|---|
| boolean | false | 1 bit | |
| char | '\u0000' | 2 byte | |
| byte | 0 | 1 byte | |
| short | 0 | 2 byte | 16 bit |
| int | 0 | 4 byte | |
| long | 0L | 8 byte | |
| float | 0.0f | 4 byte | |
| double | 0.0 | 8 byte | |

Range of byte
$-2^7$ to $2^7 - 1$
$= -128$ to $127$






Scanned by CamScanner




$2^{-15}$ to $2^{15}-1$

-32768 to -32767

0 to 65535

65536

Q) why the size of char in Java is 2 bytes?

⇒ Java supports universal code or unicode character i.e the character datatype can accomodate all different types where 65536 diff types of characters/symbols are present. So, to support all varieties of symbols in Java language in the Java soft people have increased the size of character datatype to 2 bytes.

## float and double

- float can represent 7 digits accurately after the decimal.
- Double can represent 15 digits accurately after the decimal.
- If f is not written JVM will allocate 8 bytes by considering as a double value.



Scanned by CamScanner

long

If l is not written the one byte of memory will be allocated to no. 1 and 1 byte of ~~memory~~ is sufficient to store 25.

```
Class Test
{
    PSVM
    {
        float no = 35.5f;
        int no1 = (int)no;
        SOPln (no);   //35.5f
        SOPln (no1);  //35
    }
}
```

Q) Why there is no size of operator in Java?

Variable

There are three diff. kinds of variables available in Java

1) Local
2) Instance
3) static



Scanned by CamScanner

- A variable is a container which holds the value while java program is executed.
- Every value is assigned with a datatype.
- Also variable is the name of a memory location.

1) local variable

- In Java, any variable which is declared inside the body of the method is called local Variable i.e this kind of variable can only be used within the method that and other methods of that class are not aware about the existence of local variable.

```
Class Test
{
    void fun()
    {
        int no. = 20; // local variable
        System.out.println("Hello" + no);
    }
    Public static void main(String args[])
    {
        Test ob1 = new Test(); // obj of class
        ob.fun();
        SOP(no.); // undefined Error Test
    }
}
```

A local variable can't be defined using static keyword.



Scanned by CamScanner

a) Instance Variable

```
class Test
{
    int no = 20;  // instance variable
    void fun()
    {
        System.out.println("Hello "+no);
    }
    Public static void main (String args[])
    {
        Test ob = new Test();
        ob.fun();
        System.out.println(no); // error
    }
}
```

sop(ob.no); // executable

→ A variable declared inside the class but outside the body of the method is known as instance variable.

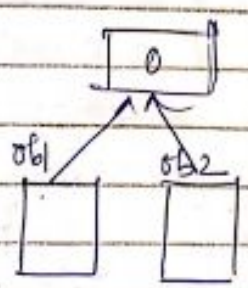
→ This variable is known as instance variable because its value is instance / object specific and is not shared among dif. instances.

```
class Test
{
    int no;
    void show()
    {
        System.out.println("Hello" + no);
    }
}
```



Scanned by CamScanner

```
void input()
{
    Scanner sc = new Scanner(System.in);
    sop("enter value");
    no = sc.nextInt();
}
Public static void main(String args[])
{
    Test ob1 = new Test();
    Test ob2 = new Test();
    ob1.input();
    ob2.input();
    ob1.show()
    ob2.show();
}
}
```




3) Static Variable

```
Class Test
{
    static int no;
    void show()
    {
        System.out.println("Hello"+no);
    }
    PSVM
    {
        Test ob1 = new Test();
        Test ob2 = new Test();
        ob1.show();
        ob2.show();
    }
}
```

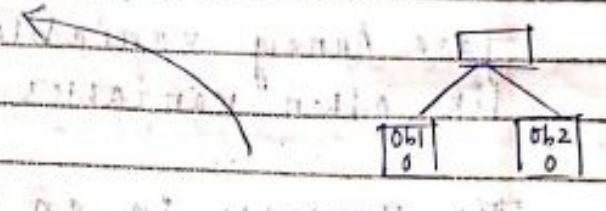
o/p Hello 0
Hello 0



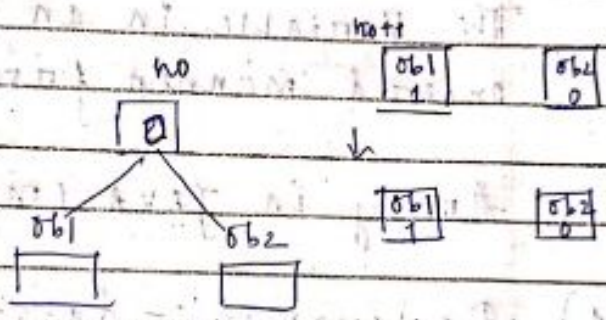
Scanned by CamScanner

```
    }
    SOP ("Hello" + no);
    no++;
    }
```

O/P Hello 0
Hello 0




```
static int no;
void show()
{
}
```




O/P Hello 0
Hello 1

→ A variable which is declared as static is called as static variable.

→ A single copy of the static variable is created and will be shared in all instances of class

→ Memory allocation of static happens only once when the class is loaded in memory.

Q) What is the diff. btn instance variable & static variable?



Scanned by CamScanner

# Array

Array is a collection of similar types of objects in java which gets memory location in contiguous memory allocation.

Java Array variables can also be declared like other variables.

The variable in an array ~~presentation~~ is in ordered manner from index 0

Array in JAVA consists of 3 phases -

1) Declaration :- datatype ~~arrangement~~ arrayname [];

2) Memory (arrays) Allocation / Instantiation :-
arrayname = new datatype [size];

3) Initialization : arrayname [index] = value.

Array name keeps the base address / starting block's address

```
class Test
{
    public static void main (String args[])
    {
        int index;
        arr = new int [30];
        int size;
        Scanner sc = new Scanner (System.in);
        System.out.println ("Enter size of
                                array");
```



Scanned by CamScanner

```
        size = sc.nextInt(); // 5
        for (index = 0; index < size; index++)
        {
            System.out.println("Enter Value");
            arr[index] = sc.nextInt();
        }
        System.out.println("Values are:");
        for (index = 0; index < size; index++)
        {
            system.out.println(arr[index]);
        }
    }
```

Q. Write a JAVA programe to search an element is present in an array /not. If present how many times and in which index its found

```
class Search
{
    public
    {
        int index, size, ele, freq = 0;
        int arr[];
        arr = new int[30];
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size
                            of the array");
        size = sc.nextInt();
        System.out.println("Enter the array
                            elements:");
        for (index = 0; index < size; index++)
        {
            System.out.println("Enter ar
                            elements");
```



Scanned by CamScanner

```
        arr[index] = sc.nextInt();
    }
    System.out.println("Enter element
                to be searched:");
    ele = sc.nextInt();
    for
    {
        if (arr[index] == ele)
        {
            freq++;
            SOP("present at "+index+"index");
        }
        if (freq == 0)
        {
            SOP("Not present");
        }
    }
}
```

## Multi-Dimensional Array.

Initialization

```
int arr[][] = new int [10][20];
int rows, cols;
for (i=0; i<row; i++)
{
    for (j=0; j<col; j++)
    {
        SOP("enter values");
        arr[i][j] = sc.nextInt();
    }
}
```



Scanned by CamScanner

17/8/19.

## Class and object

### Object

A object is an instance of a class i.e it is an entity that has state and behaviour and also identity. It can be physical and logical

1) State - It represents the data or value of an object.
2) behaviour - It represents the behaviour or functionality present inside the class
3) Identity - Every object is implemented through unique ID
   - The unique ID is not visible to the external uses. but internally it is implemented through JVM.

Q) Define an employee class with diff. data members and display the data of two employees

```
class Employee
{
    int empid;
    String name;
    float salary;
    {

    void input()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter employ...
                                    id
```



Scanned by CamScanner

```
empid = sc.nextInt();
System.out.println("Enter name");
name = sc.next();
System.out.println("Enter salary");
salary = sc.nextFloat();
}
void show()
{
System.out.println("Employee id is "+empid);
System.out.println("Employee name is "+name);
System.out.println("Employee salary is "+salary);
}
class Test
{
public static void static(String arg[])
{
Employee ob1 = new Employee();
ob1.input();
ob1.show();
Employee ob2 = new Employee;
ob2.input();
// ob1.hashCode(); } it will print the unique id of the object.
// ob2.hashcode();
}
}
```



Scanned by CamScanner

2+13.

Define a complex class. Declare two data members real and img. Define one method to take input and another method to find addition of two complex nos. Define another method to show the result.

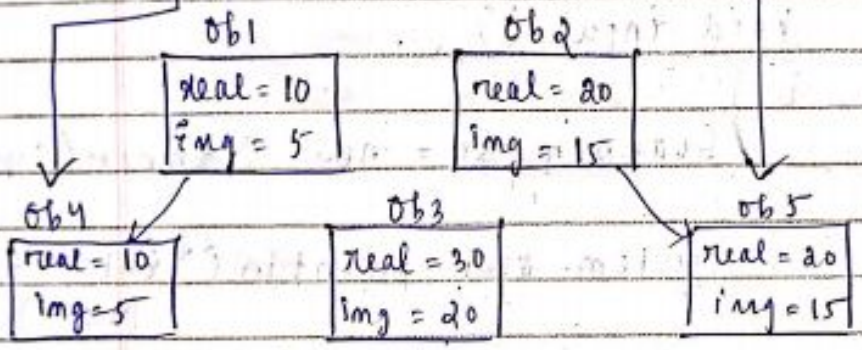
```
class Complex
{
    int img, real;
    Scanner sc=
    void input()
    {
        Scanner sc = new Scanner(System.in)
        System.out.println("Enter Real Part");
        real = sc.nextInt();
        system.out.println("Enter the imaginary part");
        img = sc.nextInt();
    }
    void cal(complex ob4, complex ob5)
    {
        real = ob4.real + ob5.real;
        img = ob4.img + ob5.being;
    }
    class Test
    {
        PSVM(String args[])
        {
```



Scanned by CamScanner

```
void show()
{
    SOPln("The sum of "+real+" + "+img+"i");
}
```

```
Complex ob1 = new complex();
Complex ob2 = new Complex();
ob1.Input();
ob2.input();
complex ob3 = new complex();
ob3.add(ob1, ob2);
ob3.show();
```




(30 + 20i)

21/8/19
Wed

**How to initialize the variable of a class ?**

```
class student
{
    String name;
    float mark;  int roll;
    void show()
    {
        System.out.println("Name: "+name);
        System.out.println("Salary: "+mark);
        System.out.println("Roll: "+roll);
    }
```



Scanned by CamScanner

```
void initialize (String name1, int roll1,
                 float mark1)
{
    name = name1;
    roll = roll1;
    mark = mark1;
}
}

class Test
{
    public String void main (String
                             args)
    {
        Student st = new student();
        st.initialize ("Shree", 31, 25.6f);
        st.show();
    }
}
```

# To make input from the keyboard.

```
class Test
{
    PSVM
    {
        Student st = new student();
        String name2;
        int roll2;
        float mark2;
        Scanner sc = new Scanner (System.in
        System.out.println("name:      );
        System.out.println("roll:      );
        System.out.println("mark:      );
```



Scanned by CamScanner

```
sop ("Name : ");
name = sc.nextline();
sop ("Roll : ");
roll = sc.nextint();
sop ("mark : ");
mark = sc.nextfloat();
sc.initialize(name2, roll2, mark2);
St.show();
    }
}
```

Q/ WAP to define a class time having datamembers hr, min, sec. Use initialize method to initialize the data member and show method to display the method. Use add method to find the addition of two time. Display the result in proper format HH:MM:SS

```
Class Time
{
    int hr, min, sec;
    void show()
    {
        System.out.println("Hour: "+hr);
        System.out.println("Minute: "+min);
        System.out.println("Second: "+sec);
    }
}
```



Scanned by CamScanner

```
void initialize (int hour, int minute,
                 int sec)
{
    hr = hour;
    min = minute;
    sec = second;
}
void addadd (Time ob1, Time ob2, Time &
             ob3)
{
    hr = ob1.hr + ob2
    min = ob1.min +
    sec = ob1.sec +
```



Scanned by CamScanner

obj1, obj2
num, largest;

Q) Define a class number with data member num of integer type. Input method will accept the value and display will print the result. Define a large method which will find largest among two nos.

```
Class Number
{
    int num, largest num1;
    void input()
    {
        Scanner sc = new scanner
                     (System.
        System.out.println("enter the
                            values");
        num = sc.nextInt();
        num1 = sc.nextInt();
    }
    void
    void large (Number obj1, Number obj
    {
        if (obj1.num > obj2.num)
        {
            num = obj1.num;
        }
        else (obj1.num < obj2.num)
        {
            num = obj2.num;
        }
    }
```



Scanned by CamScanner

```
else
{
    num = -1;
}
}
void show()
{
    if (num == -1)
    {
        cout("Not greater");
    }
    else
    {
        cout("Greater is " + num);
    }
}
```



Scanned by CamScanner

22/8/17

adv

## Method overloading

- It is a mechanism of keeping multiple methods with the same name but all methods are different wrt number of arguments or type of arguments.

```
Class overloading Example
{
    int res;
    void add (int no1, int no2)
    {
        res = no1 + no2;
        SOP ln ( "The result is = " + res);
    }
    void add (int no1, int no2, int no3)
    {
        res = no1 + no2 + no3;
        SOP ln ("Sum = " + res);
    }
    void add (float no1, float no2)
    {
        float sum;
        Sum = no1 + no2;
        SOP ln ("Sum = " + Sum);
    }
}
Class Test
{
    PSVM (String args[])
    {
    OverloadingExample ob = new
                            OverloadingExample();
    ob1.add (5, 10);
    ob1. add (10, 20, 30);
    ob1. add (2.5f, 3.5f);
    }
}
```



Scanned by CamScanner

The linking will be established during the compilation time because due to the method signature it can be differentiated to which method body will be executed during the method call.

## Constructor

```
class ExampleConstructor
{
    int no1, no2;
    ExampleConstructor()
    {
        no1 = 10;
        no2 = 20;
    }
    void show()
    {
        System.out.println("no1 = "+no1);
        System.out.println("no2 = "+no2);
    }
}
Class Test
{
    public static void main (String args[])
    {
        ExampleConstructor ob = new ExampleConstructor();
        ob.show();
    }
}
```

Knowing the calling of constructor automatically



Scanned by CamScanner

- Constructor in Java is a special type of method that is used to initialize the object.

- It is being called Automatically during the time of object creation.

- As it constructs the value i.e. provides data for an object, so it is called

- Constructor name must be same as the class name.

- Constructor doesn't have any return type neither void

Imp | What is the diff. betn Constructor and method in Java?

## Types of Constructors

There are 2 diff types of Constructors present in Java -

i) Default Constructor / Zero-Argument Constructor
ii) Parametrized Constructor / Constructor with Arguments.



Scanned by CamScanner

## Default Constructor.

A constructor which has no arguments / parameters is called as default constructor

- If there is no constructor in a class, JVM will automatically create a default constructor
- Default constructor is used to assign same value to all objects.

Ex-

```
Class ExampleConstructor
{
    int code;
    ExampleConstructor()
    {
        code = 20;
    }
    void show()
    {
        System.out.println("code = "+code);
    }
}
class Test
{
    Public static void main(String args[])
    {
        ExampleConstructor obj = new
                                ExampleConstructor();
        obj.show();
        obj2.show();
        obj3.show();
    }
}
```



Scanned by CamScanner

ExampleConstructor ob2 = new ExampleConstructor();
ExampleConstructor ob3 = new ExampleConstructor();
ob1.show();
ob2.show();
ob3.show();
}
}

code=0 ob1
code=20 ob2
code=20 ob3

## parameterized Constructor

```
class Example
{
    int code;
    Example()
    {
        code=20;
    }
    void show()
    {
        SOPln(" code = " + code);
    }
}
class Test
{
    PSVM(String args)
    {
        Example ob1 = new Example();
        Example ob2 = new Example(30);
        Example ob3 = new Example(40);
        ob1.show();
        ob2.show();
        ob3.show();
    }
}
```

code=20 ob1
code=30 ob2
code=40 ob3

Example(int value)
{
    code = value;
}



Scanned by CamScanner

| Constructor | Method |
| --- | --- |
| - Constructor is used to initialize an object. | - Method is used to exhibit functionality of an object. |
| - Doesn't return any value | - May or may not return value. |
| - Constructor should be of the same name as that of class. | - Method name should not be of the same name as that of class. |

Q1)

```
class Time
{
    int hr, min, sec;
    Time(Construction (int hr, int min, int sec)
    {
        hr = hr1;
        min = min1;
        sec = sec1;
    }
    void add (Time t4, Time t5)
    {
        hr = t4.hr + t5.hr;
        min = t4.min + t5.min;
        sec = t4.sec + t5.sec;
        if (sec >= 60)
        {
            min = min + (sec/60);
            sec = sec % 60;
        }
    }
```



Scanned by CamScanner

```
    void show()
    {
        sop ln( hr + ":" + min + ":" + sec);
    }
}
class Test
{
    PSVM
    {
        Time Constructor t1 = new TimeConstruct
                              (2, 60, 78);
        TimeConstructor t2 = new TimeConstruct
                             (1, 12, 3);
        Time t3 = new Time();
        t3.add(t1, t2);
        t3.show()
    }
}
```

Q)

```
class Number
{
    int num;
    NumberConstructor (int value)
    {
        num = value;
    }
    void largest (Number ob4, Number ob5)
    {
        if (ob4.num > ob5.num)
            num = ob4.num;
        else if (ob4.num < ob5.num)
            num = ob5.num;
```



Scanned by CamScanner

```
        else
            num = -1;
    }
    void display()
    {
        if (num == -1)
        {
            SOPln("Most are equal");
        }
        else
        {
            SOPln("largest : " + num);
        }
    }
}
class Test
{
    psvm (String args[])
    {
        Number construction = new Numberconstruction
                              (24);
        Number construction = new Numberconstruction
                              (45);
        Numbr ob3 = new Number();
        ob3.largest(ob1, ob2);
        ob3.display();
    }
}
```



Scanned by CamScanner

Q)

```
Class Student
{
    String name;
    int roll;
    float mark;
    StudentConst (String name1, int roll1,
                  float mark1)
    {
        name = name1;
        roll = roll1;
        mark = mark1;
    }
    void show()
    {
        System.out.println("Name : " + name + "roll" +
                roll + "Mark" + mark);
    }
}
Class Test
{
    Public static void main (String args[])
    {
        Student Const st = new. StudentCost
                ("Abhi", 52, 25.6);
        st. show();
    }
}
```



Scanned by CamScanner

Mon
26/8/19

| Constructor | Method |
| --- | --- |
| - Constructor is used to initialize the state of an object. i.e it initializes the variables of an object | - Method is used to describe the behaviour of an object i.e it is used for specifying the operations of the class |
| - It must not have any return type | - It must have return type |
| - Constructor is called automatically when the object is created. | - Method is explicitly called by the user. |
| - The java compiler provides a default constructor if the user dont have any constructor method. | - Method is not provided automatically by the compiler in any case |
| - Constructor name must be same as the class name. | |
| - Syntax | - syntax |
| - Ex | - Ex - |

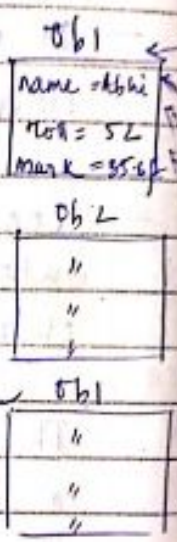


Scanned by CamScanner

Q) Diff b/w default constructor & parameterized constructor.

## Java Copy Constructor

```
class Student
{
    String name;
    int roll;
    float mark;
    Student ()
    {
        name = "Abhi"
        roll = 52;
        mark = 82.5f;
    }
    Student ( obj Student obj1)
    {
        name = obj1.name;
        roll = obj1.roll;
        mark = obj1.mark;
    }
    void show ()
    {
        SOP ("Name :" + name);
        SOP ("Roll : " + roll);
        SOP ("mark : " + mark);
    }
}
```



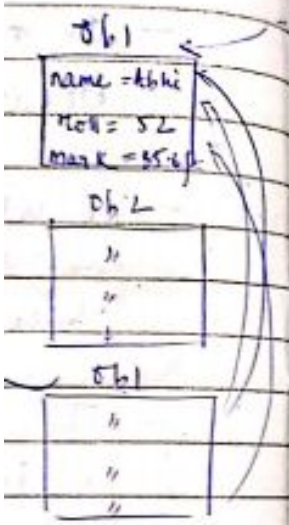



Scanned by CamScanner

11/8/26

& parameterized

```
Class Test
{
    PSVM(String args[])
    {
        Student ob1 = new Student();
        Student ob2 = new Student(ob1);
        ob1.show();
        ob2.show();
    }
}
```




While creating the object we are assigning the contents of an existing object through constructor is known as copy constructor.

## Constructor Overloading

```
Class Student
{
    String name;
    int roll;
    float mark;
    Student()
    {
        name = "abhi";
        roll = "52";
        mark = 82.5f;
    }
    Student(Student ob1)
    {
        name = ob1.name;
        roll = ob1.roll;
        mark = ob1.mark;
    }
```



Scanned by CamScanner

```
Student (String name1, int roll1,
                        float mark1)
{
    name = name1;
    roll = roll1;
    mark = mark1;
}
Student (String name1)
{
    name = name1;
}
void show()
{
    Sop("name:" +name+ "roll: "+roll+
        "mark: " + mark)"
}
}
class Test
{
    PSVM
    {
        Student ob1 = new student ();
        Student ob2 = new student (ob1);
        Student ob3 = new student ("Abhi", 52,
        Student ob4 = new student ("lipsa");
        ob1.show();
        ob2.show();
        ob3.show();
        ob4.show();
    }
```

Multiple constructor with same name but different wrt no. of arguments or type of arguments is called constructor overloading



Scanned by CamScanner

- The compiler differentiates these constructors by taking into account the no. of parameters in the list and the types.

## Java static keyword

The static keyword in Java is used for memory management. This can be applied at 4 diff. situations -

i) Variables (also known as class variable)
ii) Method (known as class method)
iii) block
iv) Nested class

The static keyword belongs to class not to the object.

## Static Method

- A static method belongs to the class rather than object of a class.
- A static method can access only static data members and can call only static methods.
- A static method can be called without the use of object or before creating the object of class.



Scanned by CamScanner

```
class Student
{
    int roll;
    String name;
    static String college = "FBS";
    static void
    {
        college = "silicon";
    }
    void show()
    {
        SOPln("Name :" + name);
        SOPln("Roll :" + roll);
        SOPln("College:" + college);
    }
    Student(int roll1, String name1)
    {
        roll = roll1;
        name = name1;
    }
}
Class Test
{
    PSVM
    {
        Student ob1 = new Student(
        Student ob2 = new Student(
        ob1.show();
        ob2.show();
        Student.change();
        ob3.show();
        ob4.show();
    }
}
```



Scanned by CamScanner

```
Student ob3 = new Student (41, "Isha")
Student ob4 = new Student (42, "Ishu")
ob3.show();
ob4.show();
    }
}
```

Static Block

```
Class Test
{
    Test()
    {
        System.out.println ("Hello");
    }
    static
    {
        System.out.println ("static Block");
    }
}
Class Demo
{
    PSVM
    {
        Test ob = new Test();
    }
}
```

O/P : Static Block
Hello



Scanned by CamScanner

Date :

# Package

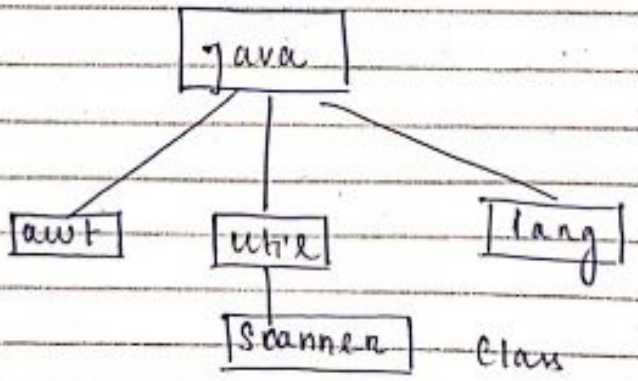
It is a group of similar type of classes, interfaces and some packages

- In Java, package is categorized in two diff. forms -

i) Built - in Package / Pre defined package
ii) User - defined package

- It is used to categorize the classes and interfaces so that they can be easily maintained.
- Java package provides access permission
- It removes naming collision.

## Built-in package




- The built in package consist of pre-defined classes with their meaning, this can't be changed but can be used inside the program as per the requirement of user.



Scanned by CamScanner

Usen-package

```
package mypack; // create a folder with
                   name mypack

class Test
{
    Public static void main (String args[])
    {
        System.out.println ("Welcome to package");
    }
}
```

compile: java <space> -d <space>.<space>
javafilename

javac -d . Test.java

execute: java mypack.Test

The -d is a switch that tells the compiler where to put a class file i.e it represents the current destination.

'.' → represent a current directory.

Q) How to access package from another package



Scanned by CamScanner

There are 3 diff package to access the package from outside the package —

i) import package name.*;
ii) import packgemame.classname;
iii) Fully qualified name;

4.09.19

There are 4 different types of Access modifiers in java with which specifies method, constructor or class

i) Private Access Modifier.

- It is accessible only within class

ii) Default Access Modifier

- If you donnot use any modifier it is treated as default

- Default modifier is accessible only within package

iii) Protected access modifier.

- It is accessible within package and outside the package but through inheritance only

- It can be applied on the data member, method and constructor but can't be applied to class.



Scanned by CamScanner

iv) Public Access Modifier

- It is accessible everywhere i.e inside the package as well as outside the package.
- It is the widest scope among all the modifier

5/04/19

## Inheritance

Inheritance is a mechanism in which one object aquires the properties and behaviour of another existing object.

The idea behind inheritance in Java is that the user can create new classes that are build upon existing claises i.e when the user inherits from an existing class from the methods & data members can be automatically transfered to the new class and also the user can add new methods and data members in the new class.

## Advantage of inheritance

- Code reusability
- Method Overriding (runtime polymorphism can be achieved)



Scanned by CamScanner

## Types of Inheritance

In Java there are 3 diff. types of inheritance mechanism are used -

i) Single inheritance.
ii) Multilevel inheritance.
iii) Hierarchical

### (1) Single Inheritance

```
class Base
{
    statement;
}
class Derived extends Bank
{
    statement
}
```

(extends → keyword)

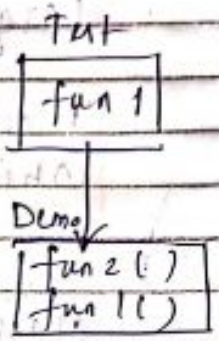

→ Type of inheritance where a single type class will aquire the properties of only one parent class. This is known as single inheritance.



Scanned by CamScanner

Example

```
class Test
{
    int no;
    void fun1()
    {
        SOP ("Base class");
    }
}
class Demo extends Test
{
    void fun2()
    {
        SOP ("child class");
    }
}
class Example
{
    psvm (St...
    {
        Demo ob = new Demo();
        ob.fun1();
        ob.fun2();
    }
}
```






Scanned by CamScanner

WAP to define a base class having data member no1, define a child class with data member no2. and methods input, large and display

Find the largest among two nos. using inheritance.

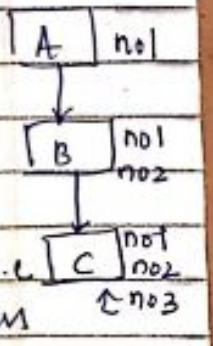
```
class Base
{
    int no1;
};
class Derived extends Base
{
    int no2,res;
    void input()
    {
        SOP("
    }
    void large()
    {
        if (no1 > no2)
        {   res = no1;
            SOP("no1 is largest")
        }
        else
        {   res = no2;
            SOP("no2 is largest")
        }
};
```



Scanned by CamScanner

```
void display()
{
    SOP("Between "+no1+" and "+no2+"
        "+res is largest");
}
}
PSV
class Example
{
    PSVM
    {
        Derived ob = new Derived();
        ob.input();
        ob.large();
        ob.display();
    }
}
```

## Multilevel Inheritance




The type of inheritance where one can inherit from a derived class thereby making there derived class the base class for the new class i.e one child class of one parent class can behave like parent for another child class.



Scanned by CamScanner

```
class Base1
{
    void fun1()
    {
        SOP ("Base 1");
    }
}
class Base2 extends Base1
{
    void fun2()
    {
        SOP ("Base 2");
    }
}
class Derived extends Base2
{
    void fun3()
    {
        SOP ("Base");
    }
}
class example
{
    psvm
    {
```



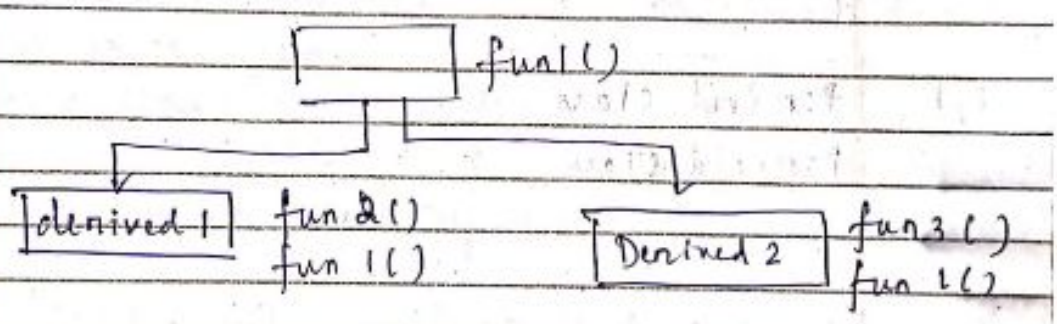
Scanned by CamScanner

There 3 classes Base1 — no1, input1
Base2 — no2, input2
Base3 — no3, input3

To take input for corresponding of remember. using scanner class
Before smallest method is derived to find smallest among 3 nos. using conditional operator. Define a display method to print the smallest value

(3) Hierarchical

The type of inheritance in which a single base class can contain multiple child class is known as Hierarchical inheritance.




9/9/19 Problem of inheritance

## Method Overriding

```
Class Base
{
    void show()
    {
        SOP ("Base class");
    }
}
```



Scanned by CamScanner

```
class Derived extends Base
{
    void show()
    {
        SOP("a Derived class");
    }
}
class Test
{
    PSVM(String args[])
    {
        Derived ob = new Derived;
        ob.show();
        ob.show();
    }
}
```

O/P Derived class
Derived class

→ If Sub-class or child class have the same method as declared in the parent class, is known as method overriding. i.e if a child class provide the specific implementation of the method that has been declared by one of a parent class, is known as method overriding



Scanned by CamScanner

## Rules

- The method must have same name as the in parent class.
- The method must have the same parameter as in the parent class.
- There must be an IS-A relationship i.e inheritance between the class.

⇒ Due to inheritance when the object of the child class is created, all the methods of base class will come to child class. So, if any method having same name as in the derived class is present in the base class will also get memory inside the child class object. So, while calling the method any no. of times using child class object, the derived class method gets executed and the base class method gets overridden by the derived class method

Imp * Q/ find the diff. bet^n method overloading & method overridding.

## Inheritance and Constructor

```
Class Base
{
    int no1;
    Base()
    {                    ①
        SOP ("Default of Base");
    }
}
```



Scanned by CamScanner

```
Class Derived extends Base
{
    Derived ()
    {                    ②
        SOP ("Default of derived");
    }
}
Class Test
{
    PSVM
    {
        Derived ob = new derived ();
    }
}
```

Inheritance and parameterized constructo

```
Class Base
{
    int no1;
    Base ()
    {
        SOP ("Default of Base");
    }
    Base (int value)
    {
        no1 = value1;
    }
}
```



Scanned by CamScanner

```
class Derived extends Base
{
    int no2;
    Derived()
    {
        SOP("Default of derived");
    }
    Derived(int value)
    {
        no2 = value;
    }
    void show()
    {
        SOP("no1+" "+no2);
    }
}
Class Test
{
    PSVM(String Arg[])
    {
        Derived ob = new Derived(20);
        ob.show();
    }
}
```

o/p 0 + 20

when object of derived class is created with argument then derived class parameterized constructor will be called i.e default base class parameterized constructor will not be called automatically like default constructor of base class.



Scanned by CamScanner

# Final

It is a keyword in Java and is applic
for variable method and class

## final variable

```
class Test
{
    PSVM (String args[])
    {
        final int no=5;
        no++;
        SOP (no);
        no = 10;
        SOP (no);
    }
}
```

⇒ Any variable which is declared with final variable, its value remains constant through out the program.

⇒ Modification to the content of variable throws errror to the user i.e the content of the final variable can't be modified.



Scanned by CamScanner

## Final Method

```
class Base
{
    final void show()
    {
        SOP("Base");
    }
}
class Derived extends Base
{
    void show()   // Error
    {
        SOP("Derived");
    }
}
```

⇒ If any base class method is declared with final keyword, no method with the same name can be present in the child class i.e method overriding is not allowed using the concept of final method

## Final Method Class

```
final class Base
{
    void show()
    {
        SOP("Base");
    }
}
```



Scanned by CamScanner

```
class Derived extends Base
{
    void show()
    {
        sop ("Derived");
    }
}
```

→ Any class which is declared with final keyword cannot be inheritaded further i.e final class doesnot have any child class.

## Super

→ Super keyword is a reference variable which is used to refer immediate parent class object i.e whenever the user creats the instance of child class, an instance of parent class is created automatically which is referred by ~~sp~~ super.

→ Super can be used for refering immediate parent class instance variable.

→ Super can be used to call the parent class method

→ Super can be used to call the immediate parent class constructor.



Scanned by CamScanner

```
Class Base
{
    int no = 5;
}
Class Derived extends Base
{
    int no = 10;
    void show()
    {
        SOP(no + " " + super.no);
    }
}
Class Test
{
    PSVM
    {
        Derived ob = new Derived();
        ob.show();
    }
}
```

(10) (5)

O/P - 10 5

- We can use super keyword to access the datamember of the parent class.
Most importantly it is used when parent and child class has same variable name.



Scanned by CamScanner

Example 2

```
Class Base                    → Home
{
    void show()               → AC
    {
        SOP("Base");   ①
    }
}
Class derived extends Base
{                             ↙ you
    void show()
    {
        super.show();  // You are accessing AC
        SOP("Derived");  ②    home.
    }
}
Class Test
{
    PSVM
    {
        Derived ob = new Derived();   → friend.
        ob.show();  // Accessing you
    }
}
```

Example 3

```
Class Base
{
```

Ba



Scanned by CamScanner

Example -3

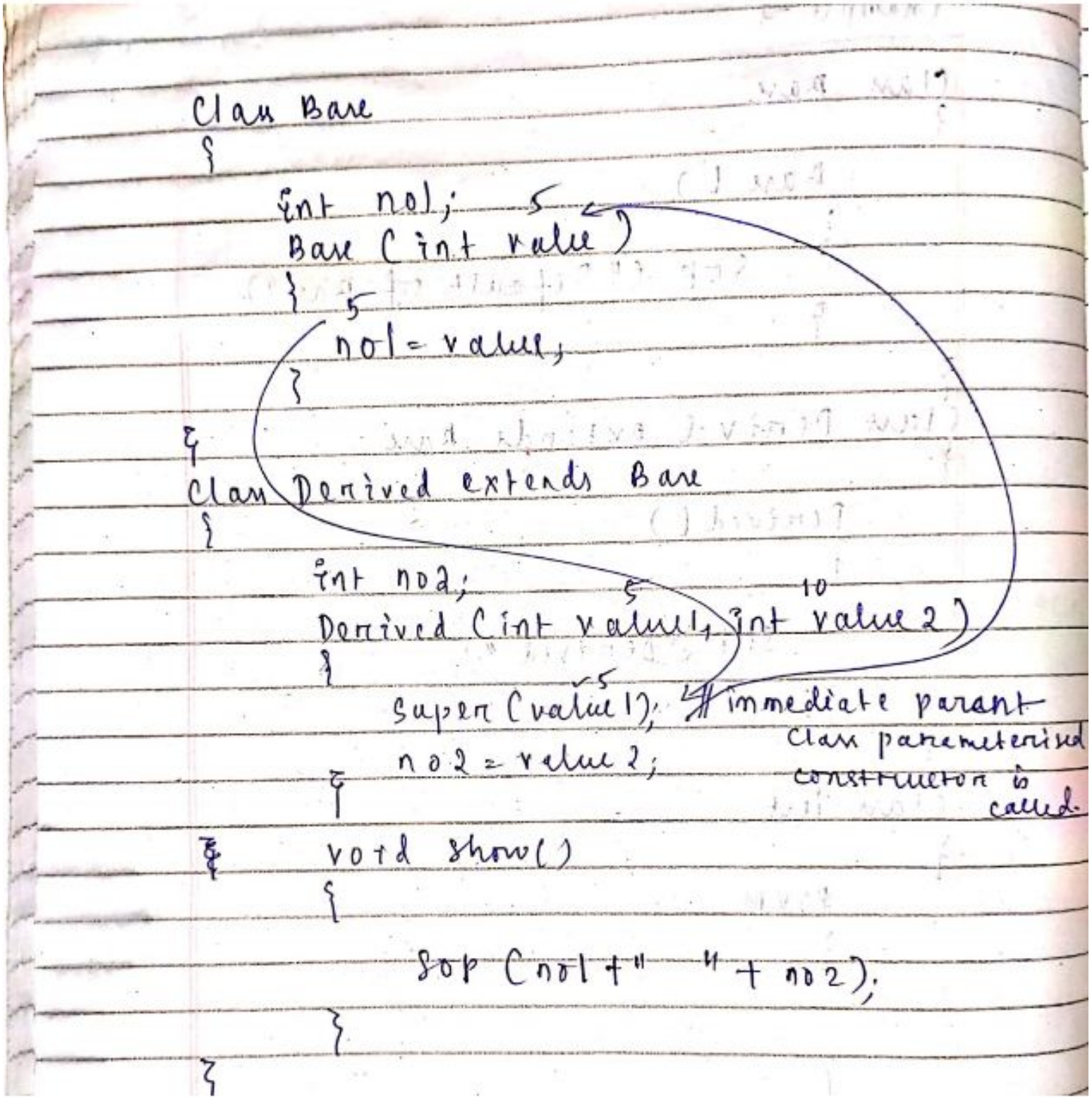
```
Class Base
{
    Base()
    {
        SOP("Default of Base");
    }
}
Class Derived extends Base
{
    Derived()
    {
        SOP("Derived");
    }
}
Class Test
{
    PSVM
    {

    }
}
```

How to initialise / call Base Class parameterised constructor?

P+o.



Scanned by CamScanner






Scanned by CamScanner

11/9/19

## Multiple Inheritance


→ In Object oriented programming if a single child & class inherits the properties of base class is known as multiple inheritance.

**Multiple inheritance is not available in java.**

## Interface

An interface in java is a blueprint of a class, which is required to achieve multiple inheritance in java.

- It is a mechanism to achieve abstraction
- There can be only abstract method and public static final variables in the body of the interface.

eg:-

```
interface Test
{
    int no;
    void show();
}
```

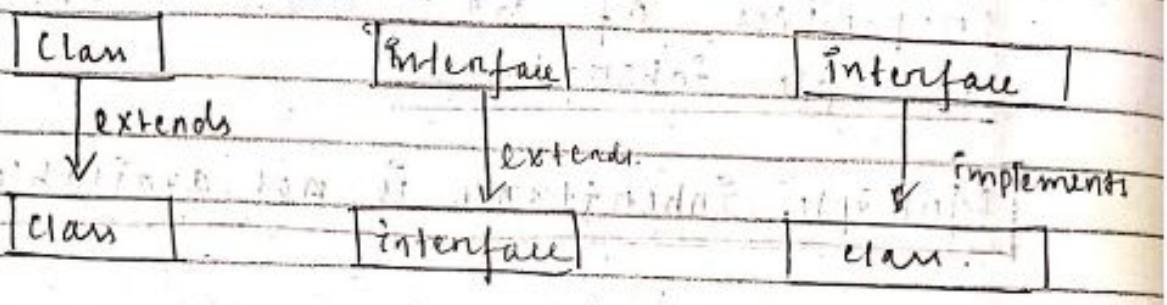
```
Interface Test
{
    public static final int no;
    public abstract void show();
}
```



Scanned by CamScanner

→ The java compiler adds public and abstract keyword before the interface method & also public, static, final keywords before the datamembers.

Relation between class and Interface:-




Q) Why interface cannot inherit the properties of a class?

→ As class can contain both abstract and non-abstract method and also data members with public, static and final and also without using this, so as per inheritance if interface will inherit the properties of a class then the rule of interface will change i.e interface has to allocate all kind of variables i.e all methods. So to keep the existence of interface, it cannot inherit another class.

```
interface Test
{
    public abstract void display();
           void show();
}
```



Scanned by CamScanner

```
class Demo implements Test
{
    public void show()
    {
        SOP ("Hello");
    }

    Public void display()
    {
        SOP ("Hi");
    }
}

Class Example
{
    PSVM
    {
        Demo Ob = new Demo();
        Ob. display();
        Ob. show();
    }
}
```



Scanned by CamScanner

1-9-19

Page No.
Date

## Polymorphism

→ polymorphism in java is a concept by which we can perform a single action by different ways.

→ There are two diff. types of polymorphism are present in java.

1) Compile time Polymorphism / static binding / Early binding

2) Run time polymorphism / Dynamic binding / late binding

- Compile time polymorphism can be achieved through method overloading.

- But Run time polymorphism can be achieved through method overriding or Dynamic Method Dispatch.

Example :-

```
Class Base
{
    void show()
    {
        SOP("Base");
    }
}
Class Derived extends Base
{
    void show()
    {
        SOP("Derived");
    }
```



Scanned by CamScanner

```
class Test
{
    PSVM (String args[])
    {
        Base ob;
        ob = new Base();
        ob.show();    // call to Base class
        ob = new Derived();
        ob.show();    // call to Derived class.
    }
}
```

Example

```
class Base
{
    void show()
    {
        SOP ("Base");
    }
}
class Derived extends Base
{
    void show()
    {
        SOP ("Derived");
    }
}
class Derived1 extends Derived
{
    void show()
    {
        SOP ("Derived1");
    }
}
```



Scanned by CamScanner

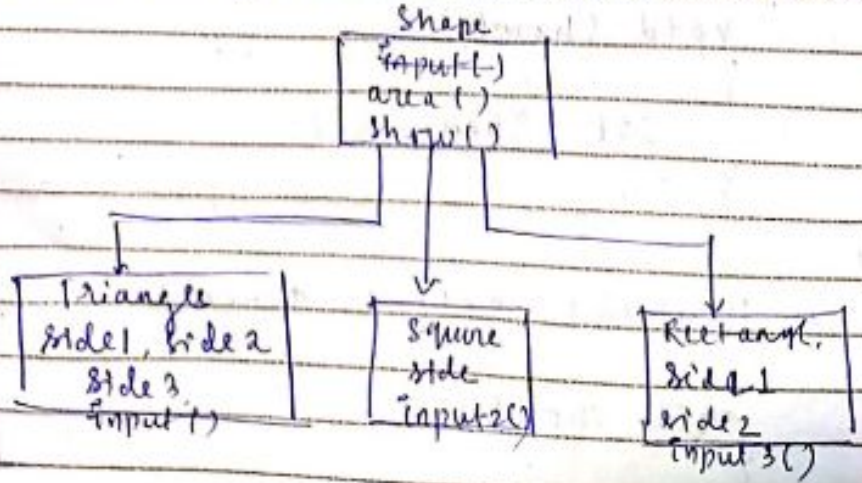
```
class Test.
{
    PSVM
    {
        Base ob;
        ob = new Derived();
        ob.show();
        ob = new Base();
        ob.show()
        ob = new Derived1();
        ob.show();
    {
}
```

## Abstract class

A class i.e declared with abstract keyword is known as abstract class in java.

- One abstract class can contain abstract and non-abstract methods within the class.

This is required to achieve abstraction.






Scanned by CamScanner

```
abstract class
{   abstract void area();
    abstract void show();
}
class Test
{
    psvm
    {
        Shape ob;
        ob = new Triangle;
        ob.&input();
        ob.area()
        ob.show();
        ob = new rectangle;
        ob.input();
        ob.area();
        ob.show();
        ob = new square();
        ob.input();
        ob.area();
        ob.show();
    }
{
```



Scanned by CamScanner

Example of abstract class with method and Constructor :-

```
abstract class Vehicle.
{
  int no;
  Vehicle (int no)
  {
    this.no = no;
    sop (" parameterised of Base");
  }
  Vehicle()
  {
    sop(" Default of Base)
  }
  abstract void run();
  Void change()
  {
    s.op.("changed");
  }
}
class Honda extends vehicle
{
  Honda()
  {
    sop(" Default of derived");
  }
  Honda (int no)
  {
    super (no);
    sop (" parameterised of derived");
  }
  void run().
  {
    sop (" Running");
  }
```

```
void change ():
{
  Super.change ();
  Sop(" changed inderived");
}
class Test
{
  PSVM (--)
  {
    ~~Vehicle ob = ob = new Vehicle~~
    Vehicle ob;
    Ob = new Honda ();
    Ob.run ();
    ob.change ();
  }
}
```

Vehicle ob = new Vehicle
error.

```
o/p : parameterised of base.
      parameterised of derived.
      Running.
      changed.
      changed in derived.
```

→ Abstract Method always stay in a abstract class
→ But an abstract class can both contain abstract or non-abstract method.
→ Interface contains only abstract methods

If you are implementing an interface & you don't require all methods of interface, then declare the class as abstract.

```
Interface first
{
    void a ();
    void b ();
    void c ();
    void d ();
}
```

abstract c
{
Publ

```
abstract class Second implements first
{
Public void c()
    {
      SOP("Method c");
    }
}

class Third extends Second
{
    void a()
    {
    }
    void b()
    {
    }
    Void d()
    {
    }
}
class Test
{
    PSVM()
    {
      Third ob = new Third();
       ob.a();
       ob.b();
       ob.c();
       Ob.d();
    }
}
```

18/08/19

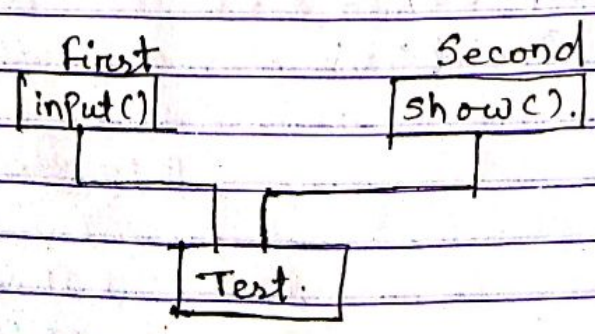
Difference between abstract class & Interface.

| Abstract class | Interface. |
|---|---|
| → It can have abstract & nonabstract method | → It can have only abstract method. |
| → final, nonfinal, static & nonstatic variable | → Only static final & variable. |
| → Can't inherit Multiple inheritance | → Can inherit multiple inheritance. |
| → Provide implementation of interface | → Can't provide impl of abstract class. |
| → Abstract keyword is required to declare the Abstract class. | → Interface keyword is used to declare. |
| → Syntax | |
| → Example. | |

How through interface Multiple Inheritance can be achieved ?

```
interface first
{
   void input();
}
```

```
interface Second
{
   void show input();
}
```



```
class Test implements first, second.
{
    int no;
Public void input()
    {
        no = i/p
    }
public void show()
    {
        sop(" Value=" + no);
    }
```

```
Class Demo
{
PSVM (- -)
{
    Test ob = new
                Test();
    ob.input();
    ob.show();
} }
```

## Exception :-

The exception Handling in Java is the one of the powerfull mechanism to find out the errors so that normal flow of application can be maintained.

→ Exception is an event that restricts the normal flow of program. In Java it is an object which is thrown at run time.

→ Through Exception Handling Mechanism our run time error known as exception can be handeled.

Types of exception :-

Two different types of exception present in Java.

(i) Checked Exception

(ii) Unchecked Exception.

(i) Checked Expression :— The type of exception which occurs during i/p ; o/p operation, insert / delete operations i.e. The exception occurs during compilation time during unavailability of files, class or databases curce called checked exception

Example: IOException
ClassNotFoundException
SQLException.

Unchecked Exception :-
The kind of exception occurs during run time or execution time are called Unchecked Exception.
Example: ArrayIndexOutOfBoundException.

19/08/19

Hierarchy OF Exception Classes

Throwable
↳ Parent of all Exception classes.

Exception
⇓
compile time exception
→ IOException
→ SQLException
→ ClassNotFoundException
→ RuntimeException.
↳ ArithmaticException
↳ NullPointerException.
↳ NumberFormatException
↳ IndexOutOfBoundsException.
↳ ArrayIndexOutOfBoundsException.
↳ StringIndexOutOfBoundsException.

Error
⇓
→ StackOverflowError.
→ VirtualMachineError.
→ OutOfMemoryError.

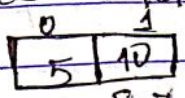
Example-1 : ArithmaticException.
int no=50/0; // This scenario occurs when any number is divided by zero.
Th.

Example-2 : NullPointerException.
String str = Null;
SOP(str.length());
// If you have null value in any variable & try to perform operation on it, you will generate this Exception.

Example-3 : NumberFormatException.

Javac add.java.

java add 5 10




```
class Add
{
 PSVM (String args[])
 {
    int no1, no2, sum;
    no1 = Integer.parseInt(args[0]); // 5
    no2 = Integer.parseInt(args[1]); // 10.
    sum = no1 + no2
    SOP("sum = " + sum);
 }
}
```

```
String str = "Silicon";
int no = Integer.parseInt(str); // NumberFormatException
```

The wrong formatting of any value may cause number formal exception i.e. the string variable with characters when converted into integer or digits, then this number format exception take this Exception takes place.

Example-4 : ArrayIndexOutOfException

```
int arr[] = new int[5];
arr[5] = 10; // Exception.
```

If the user inputs any value in wrong index or the user accesses any wrong index in array, the array-index-out-of-Exception occurs

# JAVA Exception Handling keywords:-

5 different key words:-

1- try
2- catch
3- finally
4- throw
5- throws.

## 1. try block :-

This is used to enclose the code that might throw an exception. It must be used within the method.

## 2. catch block :-

This block is used to handle the exception. It must be used after the try block only.

Q: WAP to enter two numbers & find the division

```
import java.util.*;
class Divison
{ psvm()
 {int a, int b; int div;
  Scanner sc = new Scanner();
  SOP("Enter the two values");
  a = sc.nextInt();
  b = sc.nextInt();
  div = a/b;
  SOP(div);
}}
```

Using Exception handling :-

```
class Divison
{
   psvm()
   {
      Scanner - - - -
      int a, b, div;
      a = i/p;
      b = i/p;
```

```
try
{
    res = no1 / no2 ;
    SOP (" dev ") ;
}
catch ( ArithmaticException ob )
{
    SOP (" Divison by zero is not allowed ");
    SOP (ob);
    SOP (ob.getMessages ());
}
}
```