MODULE - 1

Object Oriented Concepts

Object :- It is a real world entity which has properties and it will perform tasks.

Definition of object is **class**. Also it is an object constructors or a blue print for creating objects.

Abstraction which shows only essential parts and hiding the implementation details.

Encapsulation

It is the mechanism of wrapping the data and code acting on the data together as a single unit.

Polymorphism

It is a concept by which we can perform a single action in different ways

types :- eg. dog's smell

- compile-time polymorphism
- runtime polymorphism

It can perform by the methods such as overloading and overriding.

So Object Oriented concepts are:-

→ Object
→ Class
→ abstract
→ Encapsulation
→ Polymorphism.

Implementation of Java

Program —compiler→ Javabyte code —interpretor using JVM→ Machine Language

Javabyte code is a form of program which can be executed by any machine. Also it is the instruction set for the Java virtual machine. As soon as java program is compiled Java bytecode will be generated.

It is not completely compiled but rather just an intermediate code sitting in the middle because it still has to be interpreted and executed by the JVM installed on the specific platform such as Windows, Mac or Linux.

Java Virtual Machine (JVM)

JVM is an abstract computing machine that enables a computer to run a Java program. It is us help to interpret

Java bytecode to Java machine language, and then executed.

Java

Java is a programming language and a platform. It is a high-level, robust portable secured and object-oriented programming language.

Applications of Java

1. Mobile
2. Embedded System
3. Enterprise Applications such as banking application.
4. Web applications such as irctc.co.in, javatpoint.com etc.
5. Desktop Applications such as acrobat reader, media player, antivirus etc.
6. Smart Card
7. Robotics
8. Games etc..

Features of Java

- Simple
- Object-Oriented.
- Platform independent
- secured
- Robust
- Dynamic
- Portable
- Distributed
- Multi threaded
- High Performance
- Architecture Neutral
- Interpreted.

same

⇒ public class prime {

public static void main (String[] args)

{ int num = 29;

boolean flag = false;

for (int i = 2; i <= num/2; ++i)

{ if (num % i == 0)

{ flag = true;

}

};

if (!flag)

system.out.println (num +

" is a prime");

else.

system.out.println (num +

" is not a prime");

}

}

2 ⇒ Factorial.

public class fact {

3 ⇒ Can we overload or override static method in java?

Overloading – diff parameters.

4 ⇒ Why the main method is static in java

1 ⇒ Write a program to implement prime using diff class.

5 ⇒ What are the uses of super keyword?

6 ⇒ Write note on abstract and interface classes in java.

7 ⇒ What are the different parameter passing techniques in java.

Object oriented Programming (OOP)

OOP is the methodology which helps to organize the complex programs through the use of processes like inheritance, encapsulation, polymorphism etc.

Features of Java.

Simple

Java is defined to be easy for professional programmers to learn and use effectively. Some of more confusing concepts from

C++ are either left out of Java or implemented in a cleaner, and more approachable manner. There are a small number of clearly defined ways to accomplish a given task.

Object Oriented

Java was not designed to be a source code, compatible with any other languages. One outcome of this was clean, usable, pragmatic approach to objects The object model in java is simple and easy to extend, while simple types, such as integers are kept as high performance non-objects.

Portable (platform-independant)

Many types of computers and operating systems are used throughout the world - and many are connected to internet. For programs to be dynamically downloaded to all the various types of platforms connected to internet, Java is help for that. So they are portable.

Secured

Major safety problems are viral infection or malicious intent. Java achieves this protection by confining a Java program to the Java execution environment and not allowing it to access to other parts of the computer.

Robust

Java forces users to find mistakes early in program development. Knowing that what you have written will behave in a predictable way under diverse conditions is a key feature of Java. Robust means strong. It helps in memory management allocation and deallocation dynamically.

Dynamic

Java program carry with them substantial amounts of run-time type information that is used verify and resolve accesses to objects at run time. This makes it possible to dynamically link code in a safe and expedient manner. It is crucial to the robustness of the applet

environment in which small fragments of bytecode may be updated dynamically on a running system.

Distributed. Two control statements.

1. if statement.

format is

if(condition) statement ;

If condition is true, then the statement is executed. If condition is false, then the statement is bypassed.

For example

if(num < 100) println("num is less than 100");

Here, if num contain value less than 100, then conditional expression is true. So println() will execute, Otherwise it is bypass

2. The for loop

format is :

for(initialization ; condition; iteration) statement;

eg:-

```
Class Test{
  Public static void main(String args[]){
    int x;
```

```
for(x=0; x<2; x++)
  System.out.println("This is x: " + x);
  }
}
```

Output:

This is x : 0.

This is x : 1

Here we use '++' operator for increment i.e, increases its operand by one.

Also other operator '--', decrement operator. i.e, decreases its operant by one.

Blocks of code

Blocks can be used for if and for statements.

```
eg:- IF (x<y){
       x=y;
       y=0
     }
```

Here, if x less than y, the both statement inside block will be executed.

i.e, whenever you need to logically link two or more statements, you do so by creating a block.

Identifiers

Identifiers are used for class names, method names, and variable names.

Some identifiers are:

a4, $h, hai, Ha, ~~b~~, h_t, AvT,

Invalid identifiers are:

h-t, 24, Not/OK.

Seperators

() - Parentheses, { } - Braces, [ ] - Brackets

; - semicolon, , - Comma, . - Period.

Semicolon is used to terminate the statements.

Integers

long	8 bit
int	4 bit
short	2 bit
byte	1 bit.

Floating Point Types

double	8 bit
float	4 bit.

Characters :- char 1 bit.

Dynamic Initialization

Initialization expression may use any element valid at the time of the initialization, including calls to method,

other variables, or literals.

eg:-

```
class DynInit {
    public static void main (String args[]) {
    double a = 3.0, b = 4.0;
    double c = Math.sqrt(a * a + b * b);
    System.out.println("Hypotenuse is " + c);
    }
}
```

Output

Hypotenuse is 5.0.

Here, a and b are initialized by constants. However c is initialized dynamically.

Type conversion & casting

If the 2 types are compatible, its conversion can takes place automatically. Otherwise it is possible, but we should use 'cast', which will perform explict conversion between incompatible types.

Introducing Classes

The class is at the core of Java. It is the logical construct upon which the entire Java language is built because it defines the shape and nature of an object.

→ Class forms the basis for object oriented programming in Java

⇒ Any concepts you wish to implement in a Java program must be encapsulated in a class.

→ Class defines a new data type.

→ Once defined, this data type is used to create objects of that type.

→ Thus, a class is a template for an object.

⇒ An object is an instance for a class.

⇒ A class is declared by use of 'class' keyword.

General form:

```
class classname {
    type instance-variable 1;
    type instance-variable 2;
    //...
    type instance-variable N;
```

```
type methodname1(parameter-list){
    // body of method.
}
type methodename2(parameter-list){
    // body of method
}
//...
type methodnameN(parameter-list){
    // body of method.
}
```

⇒ Variable defined in a class is called instance variable. The code is contained within method.

Example for class.

```
class Box {
    double width;
    double height;
    double depth;
}
```

Object of the above class

```
Box mybox = new Box(); // mybox is
```

the object of class Box. Here name of the object is mybox.

To access the variables of the class we use dot(.) operator;

eg:- mybox.width = 100;

Example:-

```
class Box{
    double width;
    double height;
    double depth;
}
class BoxDemo{
    public static void main(String args[]){
        Box mybox = new Box();
        double Vol;
        mybox.width = 10;
        mybox.height = 20;
        mybox.depth = 15;
        Vol = mybox.width * mybox.height * mybox.depth;
        System.out.println("Volume is " + Vol);
    }
}
```

Output

Volume is 3000.0

Like, more than one object can be created for a particular class.

Declaring objects

```
Box mybox = new Box();
```

It can be also written as:

```
Box mybox; // declare reference to object
mybox = new Box(); // allocate a Box object.
```

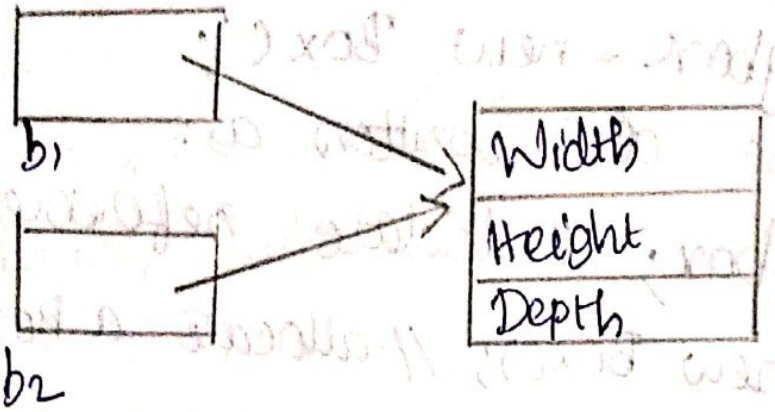
statement	Effect
`Box mybox;`	null mybox.
`mybox = new Box();`	mybox → Width, Height, Depth Box object

Assigning Object Reference Variables

```
Box b1 = new Box();
Box b2 = b1;
```

Here b1 and b2 will both refers to same object. The assignment of b1 to b2 did not allocate any memory or copy any part of original object. Thus, any changes

made to the object through b_2 will affect the object to which b_1 is referring, since they are the same object.




```
Box b1 = new Box();
Box b2 = b1;
// ....
b1 = null;
```

Here, b_1 has been set to null, but b_2 still points to the original object.

Introducing Methods

general form:

```
type name(parameter list) {
    // body of method
}
```

If method does not return any value, then type is void, otherwise it is related to the return type.

For return value, we use the keyword 'return'.

eg:- return value; // 'value' is the value returned.

Example for method

```
do void volume(){
    System.out.print("Volume is ");
    System.out.println(width*height*depth);
}
```

Call the method:

```
mybox.volume();
```

Returning a value in methods

eg:-

```
double volume(){
    return width*height*depth;
}
```

Calling the method:

```
double vol;
vol = mybox.volume()
System.out.println("Volume is " +vol);
```

Instructions:-

- The type of data returned by a method must be compatible with the return type specified ~~is~~ by the method

- The variable recieving the value returned by a method must also be compatible with the return type specified for the method.

Constructors

A constructor initializes an object immediately upon creation. It has same name as the class in which it resides and is syntactically similar to a method.

eg:-

```
Box() {
    System.out.println("Constructing Box");
    width = 10;
    height = 10;
    depth = 10;
}
```

Parameterized Constructors

The easy solution is to add parameters to the constructor. As you can probably guess, this makes them much more useful.

eg:-

```
Box ( double w, double h, double d) {
    width = w;
    height = h;
    depth = d;
}
```

'this' keyword

'this' can be used inside any method to refer to the current object. i.e, 'this' is always a reference to the object on which the method was invoked.

eg :-

```
Box ( double w, double h, double d) {
    this.width = w;
    this.height = h;
    this.depth = d;
}
```

Finalize ( ) Method

By using finalization, we can define specific actions that will occur when an object is just about to be reclaimed by garbage collection.

keyword used is :

```
finalize ( )
```

eg:-

```
protected void finalize()
{
  // finalization code here
}
```

keyword 'protected' is a specifier that prevents access to finalize() by code defined outside it class.

Closer Looks at Methods and Classes

Overloading Methods

The methods are said to be overloaded, and the process is referred to as method overloading. It is one of the way that Java implements polymorphism.

When an overload method is called, Java looks for a match between the arguments used to call the method and the method's parameters.

Argument Passing

There are 2 ways

1. Call by value
2. Call by reference.

Call by value :- This methods copies the value of an argument into the formal parameter of the subroutine.

Call-by-reference :- Here, a reference to an argument is passed to the parameter.

Recursion

Recursion is the attribute that allows a method to call itself. A method that calls itself is said to be recursive.

example :-

```
class Factorial {
    int fact(int n) {
      int result;
  if (n==1) return 1;
   result = fact(n-1) * n;
   return result;
  }
}class Recursion{
    Public static void main (String args[]){
        Factorial f = new Factorial ();
        System.out.println ("Factorial of
                              3 is " + f.fact(3));
  }
}
```

Access Control

Java supplies a rich set of access specifiers. They are :-
Public, Private and Protected.

Public :- Member of class is modified by public specifier, then that member can be access by any other code.

Private :- When it is private, it can only accessed by other members of its class.

Protected :- It can accessed by its inherited class.

Static keyword

When a member is declared static, its can be accessed before any objects of its class are created, and without reference to any object. We can declare both methods and variables to be static. Methods declared static have several restrictions, they are :-

- They can only call other static methods

- They must only access static data
- They cannot refer to this or super in any way.

Keyword final

A variable can be declared as a 'final'. Doing so prevents its contents from being modified. We must initialize a final variable when it is declared.

example:- final int FILE_NEW = 1;

Nested and Inner Classes

Class within another class is called nested class. They are of 2 types, static and non-static.

> A static nested class is one which has the static modifier applied. Because, it is static and it access the members of its enclosing class through a object.

Inner class :- It is a non-static nested class. It has access to all the variable and methods of its outer class and may refer to them directly in same way

that other non-static member of the outer class do.

Inheritance

To inherit a class, we simply incorporate the definition of one class into another by using the 'extends' keyword.
example: super class A and a subclass B which extends A

i.e,

```
class A {
    // body
}
class B extents A {
    // body.
}
```

Java doesnot support the inheritance of the multiple supers class into a single subclass.

Super class Reference a Subclass Variable

A reference variable of a superclass can be assigned a reference to any subclass derived from that superclass.

It is important that the type of the reference variable - not the type of the object that it refers to - that determines what members can be accessed. i.e when a reference to a subclass object is assigned ~~is~~ to a superclass reference variable, we will have access only to those parts of the object defined by super class.

Super keyword

Whenever a subclass needs to refer to its immediate superclass, it can do so by use of the keyword 'super'.

It has 2 general forms.

1. Superclass' constructor is called
2nd ~~call~~ is used to access a member of the superclass that has been hidden by a member of a subclass.

Method Overriding

In a class hierarchy, when a method in a subclass has the same name and type signature as a method in its superclass.

⇒ Then the method in the subclass is said to be override the method in super class.

UML

Unified Machine Language.

⇒ UML is a standard language for specifying, visualizing, constructing and documenting the artifacts of software systems.

⇒ UML is different from the other common programming languages such as C++, Java, COBOL etc.

⇒ UML is a pictorial language used to make software blue prints.

⇒ UML is not a programming Language but tools can be used to generate code in various language using UML diagrams.

Uses of Class Diagram

⇒ Describing the static view of the system.

⇒ Showing the collaboration among the elemen of the static view.

⇒ Describing the functionalities performed by the system.

⇒ Construction of software applications using object oriented ~~software~~ language.

⇒ It is the base for component and deployment diagrams.

Types of UML diagrams

1. Class diagram
2. Use case diagram
3. Object diagram
4. Sequence diagram
5. collaboration diagram
6. Activity diagram
7. State diagram
8. Deployment diagram
9. Component diagram.

Write a Java program to implement calculator

```
public class A {
    public static void main(String args[]){
        System.out.println("1.add");
        cal c = new cal();
        int ch = c.add(10,20);
        System.out.println("sum is" + ch);
    }
}
```

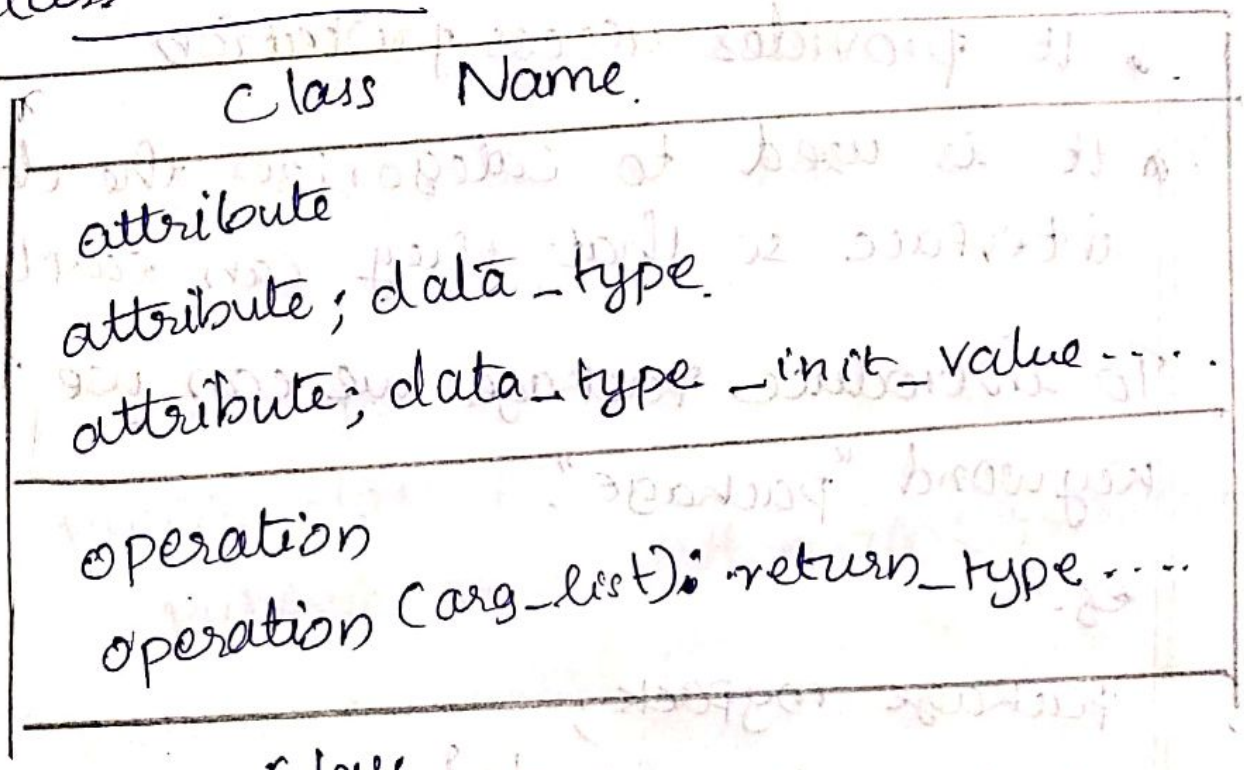
```
class Cal {
public int add(int a, int b) {
    int sum = a+b;
    return sum;
  }
}
```

Write an example for function overriding & overloading

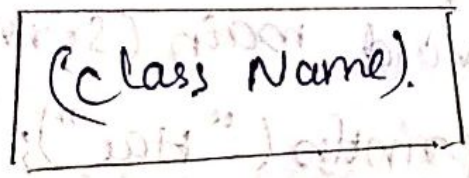
```
class A {
public static void main(String args[]){
    B b = new B();
    C c = new C();
    b.add(2,3);
    cb.add(2,3,4);
  }
}
class B {
  int add(int x, int y) {
    return (a+b);
  }
}
class C extends B {
```

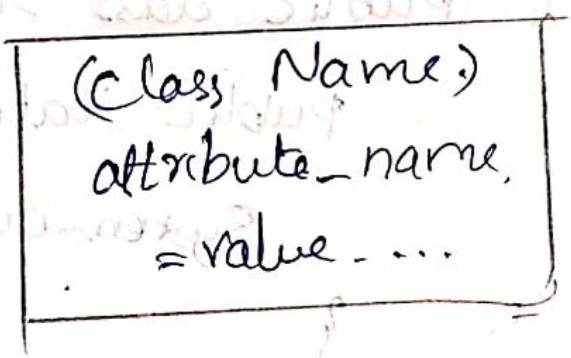
```
int add (int p, int q, int r){
        return (a+b+c);
    }
}
```

Class Diagram

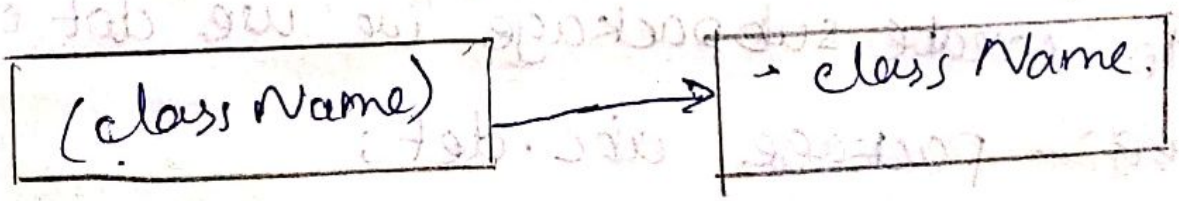


Class





Object Instances.



Packages

It is a group of similar classes, interfaces and sub-packages

Advantages

- It removes name collision.
- It provides access protection.
- It is used to categorizes the classes and interface so that they can easily maintain.

To introduce package we can use the keyword "package".

eg:-

```
package roypack;
 public class Simple {
     public static void main (String args[]) {
        System.out.println ("Hai");
     }
 }
```

To create subpackage we use dot operator

eg:- package abc.def;

abc - main package

def - sub package.

Access package from another package;

import package.*;
packname.
using this keyword we can access all classes inside this package.

Access particular class from another package

keyword : import packagename.classname;

Without using import, we can access

eg :-

```
Package pack;
public class A{
    public static void main(String args[]){
System.out.println("
    }
}

package mypack;
public class simple{
    public static
    {
     pack.A obj = new pack.A();
    }
}
```

Inorder to create two classes, we can create 2 packages with same name;

eg:-

```
package pack;
public class A { }
```

```
package pack;
public class B { }
```

Java Exception

It is one of powerful mechanism to handle the runtime errors of application so that normal flow can be maintained.

Exception is an object of class.

Exception can be handled, but Error cannot be handled eg:- division by zero.

eg:- Not much memory, virtual machine error, stack overflow error.

Java Exception keywords

- try:- exception code is placed inside try.
- catch :- It must preceeded by try block
- finally :- Used to execute the important code of the program
- throw :- used to throw an expression
- throws :- used to declare exception. It specifies there may over an exception in the method.