Abi Abi OOPs
MCA A

Date __/__/____ Page No.: 1

Assignment-1

1. Define JVM. What do you mean by Byte Codes?

Java Virtual Machine (JVM) is a virtual machine that provides runtime environment to execute java byte code.
The JVM does not support/understand Java byte, that's why we compile our .java files to obtain .class files that contain the bytecodes understandable by the JVM.
JVM control execution of every Java program.

Bytecode is computer object code that is processed by a program, usually referred to as a virtual machine, rather than by the real computer machine, the hardware processor.

2. What do you mean by classes and objects?

Class is a blueprint for objects.
A class is a user-defined type that describes what a certain type of object will look like.
A class description consists of a declaration and a definition.

Object

An object is a single instance of a class.
We can create many objects from the same cl

```
class Demo
{
    Demo()
    {
        printf("Hi");
    }
}
public class Demo2
{
    public static void main(String args[])
    {
        Demo d = new Demo();
    }
}
```

O/P

Hi

3. What is Constructor?

A constructor is a special member function of a class whose purpose is usually to initialize the members of an object.

A construct is easy to recognize because:

1. It has the same name as the class
2. It has no return type.

Constructor will be called automatically when object of the class is created.

```
class Circle
{
    Circle()
    {
        printf("In Circle Constructor");
    }
}
public class Demo
{
    public static void main(String args[])
    {
        Circle c = new Circle();
    }
}
```

O/P

In Circle Constructor

4. What is the use of this keyword.

The this keyword refers to the current object in a method or constructor. The most common use of the this keyword is to eliminate the confusion between the class attributes and parameters with the

Date __/__/____

same name.

```
public class Demo
{
    int x;
    Demo (int x)
    {
        this.x = x;
    }
    public static void main(String args[])
    {
        Demo d = new Demo(10);
        System.out.println("Value of x = "+d.x);
    }
}
```

```
class OverloadingEx
{
    public void display()
    {
        S.o.p("Method without argument");
    }
    public void display(int x)
    {
        S.o.p("Method with one parameter");
    }
    public void display(int x, int y)
    {
        System.out.printfln("Method with 2
                parameters");
    }
}
public class Demo
{
    public static void main(String args[])
    {
        OverloadingEx ovc = new OverloadingEx(10);
        OverloadingEx ovc2 = new OverloadingEx();
    }
}
```

5. What is method overloading? How method overloading is same as constructor overloading?

Method overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor loading that allows a class to have more than one constructor having different argument lists.

6. What is inheritance and types of inheritance ?
Why java does not support multiple inheritance using classes?

Inheritance is a mechanism in which one object acquires all the properties and behaviors of a parent object.
It is an important part of OOP's.

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes.

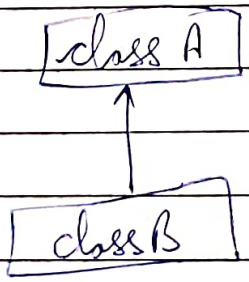
Inheritance represents the IS-A relationship which is also known as parent-child relationship.

Subclass is a class which inherits the other class.

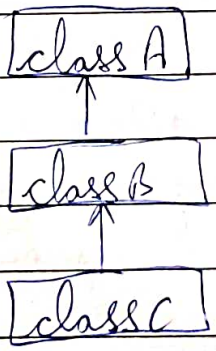
Super class is the class from where a subclass inherits the features.

Types of Inheritance

(i) Single

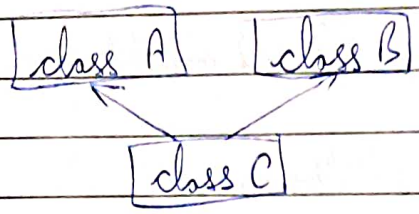


(ii) ~~Multiple~~ Multilevel

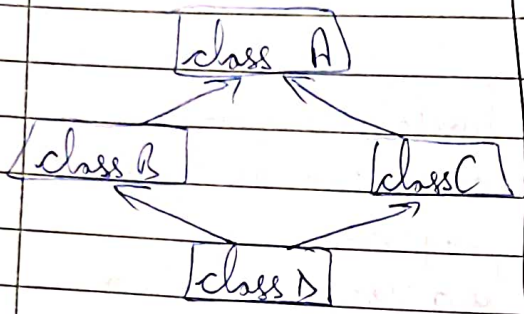


Date ___/___/___

(iii) Multiple Inheritance



(iv) Hybrid



Multiple Inheritance is not supported in Java

Let's we have three classes A, B & C
The class C inherits A and B classes.
If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

7. What is method overriding?

If subclass has the same method as declared in the parent class it is known as method overriding.

More than one
Method with same name and same argument lists in the class.

Method overriding is used to provide the specific implementation of a method which is already provided by its super class

Method overriding is used for:
runtime polymorphism.

```
class Vehicle
{
    void run() { System.out.println("Vehicle is running"); }
}
class Bike extends Vehicle
{
    public static void main(String args[])
    {
        Vehicle Bike b = new Bike();
        b.run();
    }
}
```

8. What is polymorphism?

The word polymorphism means having many forms.
We define polymorphism as the ability of a message to be displayed in more than one form.

There are two types of Polymorphism.
- Compile time Polymorphism
- Run time Polymorphism.

9. What is compile time & run time polymorphism

Compile time Polymorphism also known as static polymorphism.
This type of polymorphism is achieved by function overloading.

```
class Helper
{
    static int Multiply(int a, int b)
    {
        return a*b;
    }
    static int Multiply(int a, int b, int c)
    {
        return a*b*c;
    }
}
```

```
class Demo
{
    public static void main(String args[])
    {
        S.o.p(Helper.Multiply(2,4));
        S.o.p(Helper.Multiply(2,7,3));
    }
}
```

Runtime Polymorphism also known as dynamic polymorphism
It is a process in which a method call to the overridden method is resolved at runtime.

This type of polymorphism is achieved by Method Overriding.

```
class Parent
{
    void display()
    {
        S.o.p("Parent Class");
    }
}
class subclass1 extends Parent
{
    void display()
    {
        S.o.p("SubClass 1");
    }
}
```

Date __/__/__

```
class subclass2 extends Parent
{
    void display()
    {
        S.o.p("subclass 2");
    }
}
class Demo
{
public static void main(String args[])
{
    Parent a = new subclass1();
    a.display();
    Parent a = new subclass2();
    a.display();
}
}
```

o/p

subclass1
subClass2

10. What do you understand with Encapsulation?

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates.

Encapsulation can be achieved by declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

It is more defined with the setter and getter method.

```
class EncapsulateEx
{
    private String name;
    private int age;
    public int getAge() { return age; }
    public String getName() { return name; }
    public void setAge(int newAge)
    {
        age = newAge;
    }
    public void setName(String newName)
    {
        name = newName;
    }
}
public class Demo
{
    p s v m(String args[])
    { EncapsulateEx obj = new EncapsulateEx();
      obj.setName("Arif");
      obj.setAge(22);
    S.o.p("Name: " + obj.getName());
    S.o.p("Age: " + obj.getAge());
}
```

11. What do you mean by Abstraction?

Abstraction is the process of hiding certain details and showing only essential information to the user.

Abstraction can be achieved with either abstract classes or interface

On We can achieve 100% abstraction using interfaces.

If abstract class can have all abstract method then also 100% abstraction can be achieved.

If there is no abstract method in abstract class then here in this case 0% abstraction is there.

12. What are differences b/w Abstract class & Interface?

- Abstract class can have abstract and non-abstract methods
- Interface can have only abstract methods.

→ Abstract class does not support multiple inheritance

→ Interface supports multiple inheritance.

☆ Abstract class can have final, non final, static and non-static variables.

☆ Interface has only static & final variables.

○ The abstract keyword is used to declare abstract class

◎ The interface keyword is used to declare interface.

□ An abstract class can be extended using extends keyword.

□ An interface can be implemented using keyword 'implements'.