FULL STACK DEVELOPMENT – WORKSHEET 4

Ans1- *OOPs* or *Object Oriented Programming system* as the name says refers to programming that uses Objects as primary method to implement what is to happen in code and to solve Problems.

Concepts of OOPs:

(a) Object- Object is an entity that has state and behaviour. It can be physical or logical.

Eg- Chair, Car, Table, Banking system, etc.

(b) Class- Class is a group of objects which have common properties. It is a blueprint from which objects are created. It is a logical entity.

(c) Inheritance- When a class(subclass) acquires all the properties and behaviour of the parent class(superclass) it is called Inheritance. It provides code reusability.

(d) Polymorphism- When one task is performed in different ways it is called Polymorphism.

Types of Polymorphism:

(i)Method Overloading- It allows different methods have same name, but With different input parameters or different data type of input parameters.

(ii)Method Overriding- When a method has same name, same parameters, Same return type as its parent class , it is considered as method Overriding. The method in subclass will override the method in parent Class.

(e) Abstraction- The process of identifying only the required characteristics of an object ignoring the irrelevant details is called Abstraction.

(f) Encapsulation- Binding code and data together into a single unit is called Encapsulation. It is a protective shield that prevents the data from being accessed by the code outside this shield.

Ans2- Examples and Program for above:

- **Class and Objects**

```
class Employee
{
int id;
```

```
String name;
    public static void main(String args[])
  {
    Employee e1=new Employee(); //creating an object of class Employee
     System.out.println(e1.id);
    System.out.println(e1.name);
  }
  }
```

- **Inheritance**

```
class Bike{
String name(){return "n";}
int startingPrice(){return 0;}
            }
class RoyalEnfield extends Bike{    //keyword "extends" is used for inheritance
 String name(){return "Standard";}
int startingPrice(){return 150000;}
   }
 class Hero extends Bike{
String name(){return "Splender";}
int startingPrice(){return 55000;}
}
class Bajaj extends Bike{
String name(){return "Discover";}
int startingPrice(){return 60000;}
}
class Main{
public static void main(String args[]){
Bike b;
b=new RoyalEnfield();
System.out.println("Price of Royal Enfield "+b.name()+" is "+b.startingPrice()+".");
b=new Hero();
```

```
System.out.println("Price of Hero "+b.name()+" is "+b.startingPrice()+".");
b=new Bajaj();
System.out.println("Price of Bajaj "+b.name()+" is "+b.startingPrice()+".");
}
}
```

- ***Polymorphism***

*Method Overloading

```
public class Sum {
public int sum(int x, int y) { return (x + y); }
// This sum takes three int parameters
public int sum(int x, int y, int z)
{return (x + y + z);}

// This sum takes two double parameters
public double sum(double x, double y)
{return (x + y);}
public static void main(String args[])
{
Sum s = new Sum();
System.out.println(s.sum(10, 20));
System.out.println(s.sum(10, 20, 30));
System.out.println(s.sum(10.5, 20.5));
}
}
```

- ***Abstraction***

```
        abstract class Animal {
         public abstract void animalSound();
  public void sleep() {
    System.out.println("brrrr brrrr");
  }
}
class Dog extends Animal {
  public void animalSound() {
    System.out.println("The dog says: Bow Bow");
  }
}

class Main {
   public static void main(String[] args) {
    Dog d = new Dog(); // Create a Pig object
    d.animalSound();
    d.sleep();
  }
}
```

- ***Encapsulation***

```
        class Student {
  // private variables are declared in Encapsulation use public to access them
  private String name;
  private int rollno;
  private int marks;
```

```
    public int getRollNo() { return rollno; }

    public String getName() { return name; }

    public int getMarks() { return marks; }

    public void setRollNo(int newRollno) { rollno=newRollno; }

    public void setName(String newName)
    {name = newName;}

    public void setMarks(int newMarks) { marks = newMarks; }
}

public class Main{
    public static void main(String[] args)
    {
        Student s1 = new Student();
```

```
        // setting values of the variables
        s1.setName("Sagar");
        s1.setRollNo(22);
        s1.setMarks(85);


        // Printing values of the variables
        System.out.println("Student's name: " + s1.getName());
        System.out.println("Student's roll no: " + s1.getRollNo());
        System.out.println("Student's marks: " + s1.getMarks());
    }
}
```

MCQs

Ans1- A. Making at least one member function as pure virtual function. Use of "Abstract" keyword is necessary while doing the same.

Ans2- A. 1, 3 and 4

Ans3- B. At compile time

Ans4- A. 0

Ans5- A. Dot Operator

Ans6- C. Class

Ans7- A. Private data

Ans8- B. 0

Ans9- A. Only 1, 2 and 3

Ans10- Output- Derived::show() called

Reason- public method show is called with the instance or object b.

Ans11- It will cause compile time error because final is used as the modifier for the method show so, it cannot be called.

Ans12- Output- Base::show() called

Reason- When a function is static runtime Polymorphism doesn't happen.

Ans13- Compile time error due to access modifier in child class.

Ans14- Compile time error due to access modifier in child class as well as different return type of getdetails in child class.

Ans15- Output- Adding to 100, x = 104

Adding to 0, y = 3 3 3

Reason- because of static keyword before variables x and y.

Ans16- Compile time error.

Reason- public void m1(float f,int i); when declaring a method " ; " or semicolon is not used after declaring parameters instead it requires {body}.

Ans17- Compile time error.

Reason- cannot convert null to int.

Ans18- Output- 0 0

Reason- because int x and y has not been assigned any value so the computer will print the default value that is 0 for data type int.

Ans19- Output- Constructor called 10

Constructor called 5

Ans20- Output- 7

Reason- because 7 is at 1^{st} row index in 2^{nd} column.

Ans21- Output- 2

Reason- r is reference of type A, the program assigns a reference of object obj2 to r and uses that reference.

Ans22- Output- 2

Ans23- Output- 1 2

Ans 24- Compilation error because semicolon is missing after

“System.out.println(obj.i + " " + obj.j)” statement.

Ans25- Output- obj1.a = 4 obj1.b = 3

obj2.a = 4 obj1.b = 3