FULL STACK DEVELOPMENT – WORKSHEET 4

Q1.Write in brief about OOPS Concept in java with Examples. (In your own words)

1. **Objects -** Objects are always called instances of a class which are created from a class in java or any other language. These objects always correspond to things found in the real world, i.e., real entities. So, they are also called run-time entities of the world.

   Syntax: Class_name object_name = new Class_name();

   Example – Student obj = new Student();

2. **Classes -** Classes are like object constructors for creating objects. The collection of objects is said to be a class. Class is also called a template of an object. Classes have members which can be fields, methods and constructors.
   A class declaration consists of:
   - **Modifiers:** These can be public or default access.
   - **Class name:** Initial letter.
   - **Superclass:** A class can only extend (subclass) one parent.
   - **Interfaces:** A class can implement more than one interface.
   - **Body:** Body surrounded by braces, { }.

   Syntax: modifier class class_name {
   }

   Example – public class Student {
   }

3. **Abstraction -** Abstraction is a process which displays only the information needed and hides the unnecessary information. Abstraction is achieved by two ways either through interfaces or through Abstract class

   Abstract class -
   An abstract class is a type of class that declares one or more abstract methods. An abstract method is a method that has a method definition but not implementation.

   Example –
```
abstract class Vehicle{
    abstract  void print();
    }
  class Honda extends Vehicle{
    void print(){
            System.out.println("printing Honda");
            }
            }
    class Abstraction{
    public void main(String[ ] args){
            Honda obj = new Honda();
            obj.print();
    }
    }
```

Interface- It is blueprint of a class but with keyword interface, used to achieve 100% abstraction and also multiple inheritance can be achieved with interfaces

Syntax: interface interface_name{
```
                              }
```

Example:
```
interface A{
               }
        Interface B{
               }
    class C implements A, B{
                          }
```

4. **Inheritance -** Inheritance is a method in which one object acquires/inherits another object's properties, and inheritance also supports hierarchical classification. Inheritance represents the parent-child relationship.
   There are five types of inheritance single, multilevel, multiple, hybrid and hierarchical.

   Single level inheritance – here only one base and one derived class
   Example –
   ```
   class A{
          }
          class B extends  A {
                             }
   ```
   Multilevel inheritance – here one base is used for derived class and later this derived class acts as base class for next derived class

   Example –
   ```
   class A{
          }
          class B extends  A {
                             }
          class C extends B{
                             }
          class D extends  C {
                             }
   ```

   Multiple inheritance – This is not possible using classes in Java, however it is possible using interfaces in Java

   Hierarchical inheritance – here only one base for two or more derived classes

   Example –
   ```
   class A{
          }
          class B extends  A {
                             }
          class C extends  A {
                             }
   ```

5. **Polymorphism -** Polymorphism refers to many forms, or it is a process that performs a single action in different ways. Polymorphism is of two different types, i.e., compile-time polymorphism and runtime polymorphism. By using method overloading and method overriding, we can perform polymorphism.

   Method Overloading- Compile-Time polymorphism in java is also known as **Static Polymorphism**. To resolved at compile-time which is achieved through the **Method Overloading.** Here we have same name methods but doing different tasks

```
Example – sum(1,2){
                  }
          sum(1,5,6){
                    }
          sum(11,45,85,96,78){
                             }
```

Method Overriding- Runtime polymorphism in java is also known as **Dynamic Binding** which is used to call an overridden method that is resolved dynamically at runtime rather than at compile time. Here if same method name is there in base class as well as derived class then the derived class method will override base class method.

```
Example – class District{
             void population(){
                        System.out.println("District Population");
                  }
                  }
            class State extends District{
              void population(){
                        System.out.println("State Population");
                  }
                  }
            class MethodOverriding{
              public void main(String[ ] args){
                  State obj = new State();
                  obj.population();
            }
            }
```

6. **Encapsulation -** It is the process that binds together the data and code into a single unit and keeps both from being safe from outside interference and misuse. Encapsulation is achieved by declaring the variables as private and providing public setter and getter methods to modify and view the variable values. In encapsulation, the fields of a class are made read-only or write-only.

```
Example – class Student{
            private String name;
            }
          public String getName(){
            return name;
            }
          public void setName(String name){
            this.name=name;
            }
```

Q2.Write simple programs(wherever applicable) for every example given in Answer 2.

1. **Abstract class**

```java
package com.java.Abstraction;
import java.util.Scanner;
abstract class Lunch{
     int price = 2999;
     abstract void cuisine();
     void price() {
         System.out.println("the price is " + price);
     }
}
class Menu extends Lunch{
    void cuisine() {
        Scanner sc = new Scanner(System.in);
        System.out.println("type 1 for Indian and 2 for Chinese");
        int choice = sc.nextInt();
        if (choice == 1) {
            System.out.println("you will get indian thali");
        }
        else if (choice == 2) {
            System.out.println("you will get noodles");
        }
        else {
            System.out.println("invalid choice");
        }
        System.out.println(super.price);
    }
}
public class Abstraction {
public static void main(String[] args) {
    Menu obj = new Menu();
    obj.cuisine();
    obj.price();

}
}
```

2. Interface

```
package com.java.Interface;
interface Calculator{
//  in interface only abstract, static , default and private methods can be defined
    abstract public void message();
//  even if abstract keyword is not used , we can declare above method
    static int sum(int num1, int num2) {
        return num1+num2;
    }
//  multiple static, abstract, default and private methods can be used in a interface
    static int sub(int num1, int num2) {
        return num1-num2;
    }
    default void mul(int num1, int num2){
    System.out.println("multiplication is" + num1 * num2);
    System.out.println("division is " + div(num1,num2));
    }
//  private methods need to called in the same class and can be called in default method,
cannot be called in static method
    private int div(int num1, int num2) {
        return num1/num2;
    }
}
interface Successful{
    public void note();
}
class Result implements Calculator, Successful{
    public void message() {
        System.out.println("the result as per calculation is");
    }
    public void note() {
        System.out.println("the calculation is done successfully");
    }
}
public class MultipleInheritence {
public static void main(String[] args) {
    Calculator obj = new Result();
    Successful obj1 = new Result();
    obj.message();
    System.out.println("sum is "+ Calculator.sum(56,44));
    System.out.println("substraction  is "+ Calculator.sub(8,2));
    obj.mul(25, 5);
    obj1.note();
}
}
```

3.Inheritance

```
package com.java.Inheritance;

class House {
    int room;
    void Total() {
        int bedrooms = 5;
        int toilets = 4;
        int hall = 2;
        int kitchen = 1;
        System.out.println("total bedrooms =  "+ bedrooms +"\n toilets="+toilets+"\n
hall="+hall+"\n kitchen="+kitchen);
    }
}
//  example of single level inheritance
 class kitchen extends House{
    void food() {
        System.out.println("meals are being made");
    }
}
//example of hierarchical inheritance
 class bedroom extends House{
    void infra() {
        System.out.println("it has one bed, one dressing and one bathroom");
    }
}
//example of multilevel inheritance
class dressing extends bedroom{
    void beauty(){
        System.out.println("it has all make up products");
    }
}
class display{
    public static void main(String[]args) {
        kitchen obj = new kitchen();
        bedroom obj1 = new bedroom();
        dressing obj2 = new dressing();
        System.out.println("through single level inheritance we can know things in
kitchen and house");
        obj.Total();
        obj.food();
        System.out.println("through heirarchical inheritance we can know about house,
bedroom");
        obj1.Total();
        obj1.infra();
        System.out.println("through multilevel inheritance we can know about house,
bedroom and dressing");
        obj2.infra();
        obj2.beauty();
        obj2.Total();
    }
}
```

4. Polymorphism

a) Method Overloading

```
package com.java.Polymorphism;
class Standard5{
    void student() {
        System.out.println("details of student is");
    }
    int student(int id) {
        return id;
    }
    String student(String name) {
        return name;
    }
    float student(float marks) {
        return marks;
    }
}
public class MethodOverloading {
    public static void main(String[]args) {
        Standard5 obj = new Standard5();
        obj.student();
        System.out.println("id is "+ obj.student(123456));
        System.out.println("name is " + obj.student("Pallavi"));
        System.out.println("id is " + obj.student(90.2f));
    }

}
```

b) Method Overriding

```
package com.java.Polymorphism;
 class School{
     int students(int num){
         System.out.println("total number of students in base class are");
         return num;
     }
 }
 class Standard7 extends School{
     int students(int num){
         System.out.println("total number of students in derived class are");
         return num;
     }
 }

public class MethodOverriding {
public static void main(String[]args) {
    Standard7 obj = new Standard7();
    System.out.println(obj.students(56));
}
}
```

5. Encapsulation

```
package com.java.Encapsulation;
class Hospital{
    private String patientName;
    private int patientid;
    private int roomnum;
    public String getpatientName() {
    return patientName;
    }
    public String setpatientName(String patientName) {
     return this.patientName = patientName;
    }

    public int getPatientid() {
        return patientid;
    }
    public void setPatientid(int patientid) {
        this.patientid = patientid;
    }
    public int getRoomnum() {
        return roomnum;
    }
    public void setRoomnum(int roomnum) {
        this.roomnum = roomnum;
    }

public class EncapsulationinJava {
public static void main(String[] args) {
    Hospital obj = new Hospital();
    obj.setpatientName("Anandbhai");
    obj.setPatientid(12345);
    obj.setRoomnum(420);
    System.out.println("Patient name is "+obj.getpatientName()+"patient id is "+
obj.getPatientid()+"is living in room number"+obj.getRoomnum());

}
}
}
```

Multiple Choice Questions

Q1. Which of the following is used to make an Abstract class?

A. Making at least one member function as pure virtual function
B. Making at least one member function as virtual function
C. Declaring as Abstract class using virtual keyword
D. Declaring as Abstract class using static keyword

Ans: A

Q2. Which of the following is true about interfaces in java.

1) An interface can contain the following type of members.
....public, static, final fields (i.e., constants)
....default and static methods with bodies
2) An instance of the interface can be created.
3) A class can implement multiple interfaces.
4) Many classes can implement the same interface.

A. 1, 3 and 4
B. 1, 2 and 4
C. 2, 3 and 4
D. 1,2,3and 4

Ans: A

Q3. When does method overloading is determined?

A. At run time
B. At compile time
C. At coding time
D. At execution time

Ans: B

Q4.What is the number of parameters that a default constructor requires?

A. 0
B. 1
C. 2
D. 3

Ans: A

Q5.To access data members of a class, which of the following is used?

A. Dot Operator
B. Arrow Operator
C. A and B both as required
D. Direct call

Ans: C

Q6.Objects are the variables of the type ____?

A. String
B. Boolean
C. Class
D. All data types can be included

Ans: C

Q7.A non-member function cannot access which data of the class?

A. Private data
B. Public data
C. Protected data
D. All of the above

Ans: A

Q8. Predict the output of following Java program

```
class Test {
  int i;
}
class Main {
  public static void main(String args[]) {
      Test t = new Test();
      System.out.println(t.i);
         }
        }
```

A. garbage value
B. 0
C. compiler error
D. runtime Error

Ans: B

Q9.Which of the following is/are true about packages in Java?

1) Every class is part of some package.

2) All classes in a file are part of the same package.

3) If no package is specified, the classes in the file go into a special unnamed package

4) If no package is specified, a new package is created with folder name of class and the class is put in this package.

A. Only 1, 2 and 3
B. Only 1, 2 and 4
C. Only 4
D. Only 1, 3 and 4

Ans: A

For Q10 to Q25 find output with explanation.

Q10.Predict the Output of following Java Program.

```
class Base {
    public void show() {
      System.out.println("Base::show() called");
    }
}
class Derived extends Base {
    public void show() {
      System.out.println("Derived::show() called");
    }
}
public class Main {
    public static void main(String[] args) {
        Base b = new Derived();;
        b.show();
    }
}
```

Ans: Output : Derived::show() called

Q11. What is the output of the below Java program?

```
class Base {
    final public void show() {
      System.out.println("Base::show() called");
    }
}
class Derived extends Base {
    public void show() {
      System.out.println("Derived::show() called");
    }
}
class Main {
    public static void main(String[] args) {
        Base b = new Derived();;
        b.show();
    }
}
```

Ans: Final Methods cannot be overridden – Compile time error

Q12.Find output of the program.

```
class Base {
    public static void show() {
       System.out.println("Base::show() called");
    }
}
class Derived extends Base {
    public static void show() {
       System.out.println("Derived::show() called");
    }
}
class Main {
    public static void main(String[] args) {
        Base b = new Derived();
        b.show();
    }
}
```

Ans: Output : Base::show() called
When function is static, runtime polymorphism that is method overriding cannot happen

Q13.What is the output of the following program?

```
class Derived
{
    public void getDetails()
    {
        System.out.printf("Derived class ");
    }
}
public class Test extends Derived
{
    public void getDetails()
    {
        System.out.printf("Test class ");
        super.getDetails();
    }
    public static void main(String[] args)
    {
        Derived obj = new Test();
        obj.getDetails();
    }
}
```

Ans: Output: Test class Derived class

Q14. What is the output of the following program?

```
class Derived
{
   public void getDetails(String temp)
   {
      System.out.println("Derived class " + temp);
   }
}
public class Test extends Derived
{
   public int getDetails(String temp)
   {
      System.out.println("Test class " + temp);
      return 0;
   }
   public static void main(String[] args)
   {
      Test obj = new Test();
      obj.getDetails("Name");
   }
}
```

Ans: Compile time error since overriding method don’t have same return type

Q15.What will be the output of the following Java program?

```
   class test
{
        public static int y = 0;
}
class HasStatic
{
        private static int x = 100;

        public static void main(String[] args)
        {
                HasStatic hs1 = new HasStatic();
                hs1.x++;
                HasStatic hs2 = new HasStatic();
                hs2.x++;
                hs1 = new HasStatic();
                hs1.x++;
                HasStatic.x++;
                System.out.println("Adding to 100, x = " + x);
                test t1 = new test();
                t1.y++;
                test t2 = new test();
                t2.y++;
                t1 = new test();
                t1.y++;
                System.out.print("Adding to 0, ");
                System.out.println("y = " + t1.y + " " + t2.y + " " + test.y);
        }
}
```

Ans: Output:

```
Adding to 100, x = 104
Adding to 0, y = 3 3 3
```

Q16.Predict the output

```
class San
{
 public void m1 (int i,float f)
 {
  System.out.println(" int float method");
 }
 public void m1(float f,int i);
  {
  System.out.println("float int method");
  }
  public static void main(String[]args)
  {
    San s=new San();
        s.m1(20,20);
  }
}
```

Ans: Compile time error since while calling function, datatype is ambiguous

Q17.What is the output of the following program?

```
public class Test
{
    public static void main(String[] args)
    {
        int temp = null;
        Integer data = null;
        System.out.println(temp + " " + data);
    }
}
```

Ans: Compilation error
temp is a primitive data type.
Primitive data types cannot be assigned null values.

Q18.Find output

```
class Test {
    protected int x, y;
}

class Main {
    public static void main(String args[]) {
        Test t = new Test();
        System.out.println(t.x + " " + t.y);
    }
}
```

Ans: Output: 0 0

Q19.Find output

```
// filename: Test2.java
class Test1 {
        Test1(int x)
        {
                System.out.println("Constructor called " + x);
        }
}
class Test2 {
        Test1 t1 = new Test1(10);
        Test2(int i) { t1 = new Test1(i); }
        public static void main(String[] args)
        {
                Test2 t2 = new Test2(5);
        }
}
```

Ans: Output:

```
Constructor called 10
Constructor called 5
```

Q20.What will be the output of the following Java program?

```
 class Main
 {
  public static void main(String[] args)
   {
   int []x[] = {{1,2}, {3,4,5}, {6,7,8,9}};
   int [][]y = x;
   System.out.println(y[2][1]);
   }
 }
```

Ans: Output: 7

Q21.What will be the output of the following Java program?

```
 class A
 {
     int i;
     public void display()
     {
         System.out.println(i);
     }
 }
 class B extends A
{
     int j;
     public void display()
     {
         System.out.println(j);
     }
 }
 class Dynamic_dispatch
{
     public static void main(String args[])
     {
         B obj2 = new B();
         obj2.i = 1;
         obj2.j = 2;
         A r;
         r = obj2;
         r.display();
     }
}
```

Ans: Output: 2

Q22. What will be the output of the following Java code?

```
class A
{
    int i;
    void display()
    {
        System.out.println(i);
    }
}
class B extends A
{
    int j;
    void display()
    {
        System.out.println(j);
    }
}
class method_overriding
{
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
        obj.display();
    }
}
```

Ans: Output: 2

Q23.What will be the output of the following Java code?

```
class A
{
    public int i;
    protected int j;
}
class B extends A
{
    int j;
    void display()
    {
        super.j = 3;
        System.out.println(i + " " + j);
    }
}
class Output
{
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
        obj.display();
    }
}
```

Ans: Output: 1 2

Q24.What will be the output of the following Java program?

```
class A
{
    public int i;
    public int j;
    A()
    {
        i = 1;
        j = 2;
    }
}
class B extends A
{
    int a;
    B()
    {
        super();
    }
}
class super_use
{
    public static void main(String args[])
    {
        B obj = new B();
        System.out.println(obj.i + " " + obj.j)
    }
}
```

Ans: Output: 1 2

Q 25. Find the output of the following program.

```
class Test
{
    int a = 1;
    int b = 2;

    Test func(Test obj)
    {
        Test obj3 = new Test();
        obj3 = obj;
        obj3.a = obj.a++ + ++obj.b;
        obj.b = obj.b;
        return obj3;
    }

    public static void main(String[] args)
    {
        Test obj1 = new Test();
        Test obj2 = obj1.func(obj1);

        System.out.println("obj1.a = " + obj1.a + "  obj1.b = " + obj1.b);
        System.out.println("obj2.a = " + obj2.a + "  obj1.b = " + obj2.b);

    }
}
```

Ans: Output:

```
obj1.a = 4 obj1.b = 3
obj2.a = 4 obj1.b = 3
```