FULL STACK DEVELOPMENT – WORKSHEET 4

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

Answer:

OOP's: Stands for Object oriented programming language.

- **In object oriented programming we consider everything as an object.**
- **Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance some concepts:**
- **There are six pillars of OOP's or features of OOP's.**
  It is a programming concept that works on the principles of abstraction, encapsulation, inheritance, and polymorphism. It allows users to create objects they want and create methods to handle those objects. The basic concept of OOPs is to create objects, re-use them throughout the program, and manipulate these objects to get results.

  There are six player of OOP's concept.

  a) **Object: any real world entity which we can touch. or instance of class . as its real entity its takes memory space**

  b) **Class : class is blueprint to create instance of an object. or collection of objects , it is logical entity , we cannot touch it , no require memory space**

  c) **Inheritance :**
  it is procedure or concepts in which all the parents class properties and behavior is being acquired by child class.
  There are five types of Inheritance
  1) **Single level Inheritance:**
  2) **Multilevel Inheritance:**
  3) **Hierarchical Inheritance**
  4) **Multiple Inheritance**
  5) **Hybrid Inheritance**

  d) **Polymorphism: We can perform single task in many ways.**
  There are two types of polymorphism
  1) **Compile time Polymorphism:**
  There is one method in compile time polymorphism
  Which is method overloading

  Method overloading: Methods having same name with different parameters
  2) **Run time Polymorphism:**
  There is one method in run time polymorphism
  Which is method overriding.

Method overriding : Method having same name with same arguments .

e) Abstraction: Hiding implementation details and showing only functionability.
f) Encapsulation: Binding data and code in single unit.

Example: see java files.

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

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

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. 1,2,3 and 4

Q3. When does method overloading is determined?

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

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

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

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

Q6.Objects are the variables of the type ____?

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

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

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

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

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();
    }
}
```

Explanation with Output:

1) The Derived class extends the Base class and overrides the show() method to print "Derived::show() called".

2) When the show() method is called using the reference b, the overridden method in the Derived class is invoked due to dynamic method dispatch.

3) Hence, the output is "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();
    }
}
```

Explanation with Output:

1) In the main method, a reference of type Base is created and assigned an object of type Derived.

2) When the show() method is called using the reference b, the method from the Base class is invoked because it cannot be overridden.

3) Hence, the output is "Derived::show() called".

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();
    }
}
```

Explanation with Output:

1) When the show() method is called using the reference b, the static method from the Base class is invoked because static methods are resolved at compile-time based on the reference type.
2) Hence, the output is "Base::show() called".

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();
    }
}
```

Explanation with Output:

1)The overridden method prints "Test class" and then calls super.getDetails() to invoke the getDetails() method of the superclass (Derived class).
2)The getDetails() method of the Derived class is invoked and it prints "Derived class".
3) Hence, the output is "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");
   }
}
```

Explanation with Output:

The given code will produce a compilation error because of a method signature mismatch
Test class, the method In the getDetails(String temp) is declared with a return type of int, while in the Derived class, the corresponding method getDetails(String temp) has a return type of void. This creates a conflict as the Test class is trying to override the method from the Derived class.
To resolve this issue, the return type of the getDetails() method in the Test class should match the return type of the overridden method in the Derived class. Alternatively, if you want to have a different method signature in the Test class, you can give it a different name to avoid method overriding.

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);
        }
}
```

Explanation with Output:

The output of the given Java program will be:
to 100, x = 103 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);
  }
}
```

Explanation with Output:

1) The given code contains compilation error as second method m1 declaration is missing the body.
2) The method m1(int, float) is ambiguous for the type San.

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);
   }
}
```

Explanation with Output:

The Given java code will result in a compilation error because you cannot assign "null" value to a primitive data type like 'int'.

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);
   }
}
```

Explanation with Output:

Here we create object of Test class using new keyword. And initialize its member.

As instance variables are not assigned explicitly. default constructor gives default value to int data type.

Default value for int data type is 0.

So Output is 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);
        }
}
```

Explanation with Output:

1) In this code there are two classes Test1 & Test2
2) First we create constructor of class Test1 and passing integer parameter into it.
3) Then creates obect t1 of class Test1 in class Test2 and passes integer parameter value as a 10.
4) Then in main method we create object t2 of the class Test2 and passing parameter value as a 5 into it. This triggers execution 'Test2' constructor, which in turn create a new 'Test1' object with the value of 5.
5) So output is "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]);
  }
}
```

Explanation with Output:

The output is 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();
    }
}
```

Explanation with Output:

The output is 2.

1) **In the code, there are two classes ‘A’ & ‘B’ , where ‘B’ is a subclass of ‘A’. Both have a method named ‘display()’**
2) **In the ‘main’ method, an object of class ‘B’ is created and assigned to a reference variable ‘r’ of type ‘A’. Since ‘B’ is a subclass of ‘A’. it is allowed to assign a ‘B’ object to an ‘A’ reference.**
3) **When the ‘display()’ method is called using reference ‘r’, the method from the**

actual object type 'B' is invoked ,not the method from the reference type 'A'. this is known as dynamic method dispatch. Where there method to be called is determined runtime based on the actual object type.

4) Since the value of 'j' in the 'B' objects is '2'. The 'display()' method of class 'B' will be executed , and it will print '2' as the output.

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();
    }
}
```

Explanation with Output:

The output is 2.

1) In the code , there are two classes 'A' and 'B', where 'B' is a subclass of 'A'. both classes have a method named 'display()'.
2) In the "main" method , an object of class 'B' is created and assigned to the reference variable 'obj'. Since 'B' is a subclass of 'A', it inherites 'display()' method from class 'A'. However, class 'B' also overrides the 'display()' method with its own implementation.
3) When the 'display()' method is called using 'obj', the method from the actual object type 'B' is invoked due to dynamic method dispatch. This means that the 'display()' method in class 'B' will be executed , which prints the value of 'j' which is '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();
    }
}
```

Explanation with Output:

The output is '1 2'.

1) **When the 'display()' method is calling using 'obj', it accesses the 'i' and 'j' variables from class 'A' and print their values.**

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)
    }
}
```

Explanation with Output:

The output is '1 2'

1) **In the code, there are three classes: 'A' , 'B' and 'super_use'. Class 'A' has two public instance variables 'i', and 'j', and a constructor 'A()' that initializes the values of 'i' and 'j' to '1' and '2' respectively.**
2) **Class 'B' is a subclass of 'A' and it has an additional instance variable 'a'. it also has a constructor 'B()' that calls the default constructor of 'A' using the 'super()' keyword.**
3) **In the 'super_use' class, an object of class 'B' is created and assigned to the reference variable 'obj'. when 'obj.i' and 'obj.j' are accessed , they refer to the values of 'i' and 'j' inherited from class 'A'. Since the constructor of class 'A' is called during the construction of 'B' object, the values of 'i' and 'j' are set to '1' and '2' respectively.**

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);

   }
}
```

Explanation with Output:

The output is

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

1) **In the 'Test' class, there are two instance variables 'a' and 'b' and a 'func()' method. The 'func()' method takes an object of type 'Test' as an argument, perform some operations on the object's variables, and return a new object of type 'Test'.**
2) **In the 'main()' method , an object 'obj1' of type 'Test' is created . Then, the 'func()' method is called on 'obj1' and the returned object is assigned to 'obj2'.**
3) **Inside the 'func()' method, a new object 'obj3' is created and initially assigned the reference of the 'obj' argument. Then the 'a' variable of 'obj3' is updated using the expression 'obj.a++ + + ++obj.b'. Here, "obj.a++" increments the value of 'obj.a' by 1 and returns the original value, and '++obj.b' increments the value of 'obj.b' by 1 and returns the updated value. The sum of these values is assigned to 'obj3.a'.**
4) **After the 'func()' method call, the value of 'obj1.a' and 'obj1.b' remain unchanged because the modifications were made on a different object ('obj3'). Similarly, the values of 'obj2.a' and 'obj2.b' are also unchanged as 'obj2' is referencing the same object as 'obj1'**
   Therefore , the output shows
   'obj1.a = 4 obj1.b = 3
   obj2.a = 4 obj1.b = 3'