Q1.Write in brief about OOPS Concept in java with Examples.

ANS: Object-Oriented Programming (OOP) is a programming paradigm that revolves around the concept of "objects". In Java, OOP is a fundamental aspect of the language. It is based on four main principles:

Encapsulation: Encapsulation is the process of bundling data (attributes) and methods (functions) that operate on the data, restricting the access to some of the object's components. This is achieved through access modifiers like public, private, protected, etc. Example:

```
class Person {
   private String name;
   private int age;

   public void setName(String name) {
      this.name = name;
   }

   public String getName() {
      return this.name;
   }

   public void setAge(int age) {
      if(age > 0) {
         this.age = age;
      }
   }

   public int getAge() {
      return this.age;
   }
}
```

Inheritance: Inheritance allows a class (subclass) to inherit the attributes and methods of another class (superclass). This promotes code reuse and the creation of a hierarchy of classes. Example:

```
class Animal {
```

```
    public void sound() {
        System.out.println("Some generic sound");
    }
}

class Dog extends Animal {
    @Override
    public void sound() {
        System.out.println("Bark");
    }
}
```

Polymorphism: Polymorphism allows objects to be treated as instances of their superclass. This means that a method can behave differently based on the object that it is acting upon. Example:

```
class Shape {
    public void draw() {
        System.out.println("Drawing a shape");
    }
}

class Circle extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a circle");
    }
}
```

Abstraction: Abstraction involves hiding the implementation details of an object and exposing only the relevant operations or behavior. Abstract classes and interfaces are used to achieve abstraction. Example:

```
abstract class Shape {
    abstract void draw();
```

```
}

class Circle extends Shape {
   @Override
   void draw() {
      System.out.println("Drawing a circle");
   }
}
```

Encapsulation: Encapsulation is the concept of restricting access to some of the object's components. This is achieved through access modifiers like public, private, protected, etc. Example:

```
class Person {
   private String name;
   private int age;

   public void setName(String name) {
      this.name = name;
   }

   public String getName() {
      return this.name;
   }

   public void setAge(int age) {
      if(age > 0) {
         this.age = age;
      }
   }

   public int getAge() {
      return this.age;
   }
```

```
}
```

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

```
// Encapsulation: Using getters and setters to access private attributes

class Person {
    private String name;
    private int age;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    public void setAge(int age) {
        if (age > 0) {
            this.age = age;
        }
    }

    public int getAge() {
        return this.age;
    }
}

// Inheritance: Creating a subclass that inherits from a superclass
```

PFA WORKSHEET 4 SOLUTION

```
class Student extends Person {
    private int studentId;

    public void setStudentId(int id) {
        this.studentId = id;
    }

    public int getStudentId() {
        return this.studentId;
    }
}

// Polymorphism: Overriding a method in the subclass

class Shape {
    public void draw() {
        System.out.println("Drawing a shape");
    }
}

class Circle extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a circle");
    }
}

// Abstraction: Using an abstract class

abstract class Animal {
```

PFA WORKSHEET 4 SOLUTION

```
    abstract void sound();
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Bark");
    }
}

public class Main {
    public static void main(String[] args) {
        // Encapsulation example
        Person person = new Person();
        person.setName("John Doe");
        person.setAge(30);
        System.out.println("Name: " + person.getName());
        System.out.println("Age: " + person.getAge());

        // Inheritance example
        Student student = new Student();
        student.setName("Jane Doe");
        student.setAge(20);
        student.setStudentId(12345);
        System.out.println("Name: " + student.getName());
        System.out.println("Age: " + student.getAge());
        System.out.println("Student ID: " + student.getStudentId());

        // Polymorphism example
        Shape shape = new Circle();
        shape.draw();
```

```
        // Abstraction example
        Animal animal = new Dog();
        animal.sound();
    }
}
```

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 : B

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.

ANS: 1,3,4

When does method overloading is determined?
A. At run time
B. At compile time
C. At coding time
D. At execution time

ANS:B

What is the number of parameters that a default constructor requires?
A. 0
B. 1
C. 2
D. 3

ANS: A

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: A. Dot Operator

In Java, the dot operator (`.`) is used to access the members (variables and methods) of a class.

Objects are the variables of the type ____?
A. String
B. Boolean
C. Class
D. All data types can be included

ANS: C. Class

Objects are variables of a class type. They are instances of a class, created based on the blueprint provided by the class.

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. Private data

A non-member function (a function that is not a member of the class) cannot access the private data members of a class.

Q. 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. 0

In Java, if you don't explicitly initialize a member variable, it is assigned a default value. For integer types like `int`, the default value is 0. Therefore, in this program, `t.i` will have the value of 0, and the output will be 0.

Q. 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 : B. Only 1, 2 and 4

Explanation:

1) Every class is part of some package.

- This is true. If a class doesn't explicitly specify a package, it is part of the default package.

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

- This is true. In Java, a source file can contain at most one public class, and the file should be named after the public class. All classes in the file, whether they're public or not, belong to the same package.

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

- This is true. Classes without an explicit package declaration belong to the default package (also known as the unnamed package).

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

- This is not true. In Java, packages are not automatically created based on the folder structure. Package names are declared explicitly in the source code.

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: 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 : compilation error.

error: show() in Derived cannot override show() in Base

Explanation:

1. In the `Base` class, the `show()` method is declared as `final`. This means that it cannot be overridden in any subclass.

2. In the `Derived` class, when we attempt to override `show()`, we get a compilation error because it's not allowed to override a `final` method.

Therefore, the program will not compile successfully.

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 : Base::show() called

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 : The provided code will not compile due to a method signature mismatch between the `Derived` class and the `Test` class.

Here's the specific issue:

In the `Derived` class:

public void getDetails(String temp)

In the `Test` class:

public int getDetails(String temp)

The `getDetails` method in the `Test` class attempts to override the method in the `Derived` class, but the return type is different. This is not allowed in Java. The return type must be the same or a subclass of the return type of the overridden method.

To fix this issue, the method signatures in both classes should match.

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 : Adding to 100, x = 103

Adding to 0, y = 1 1 1

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 : The provided Java code contains a syntax error. Specifically, there's a semicolon (;) after the second method declaration.

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 : error: incompatible types: <null> cannot be converted to int

int temp = null;

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 : The given Java program will not compile successfully because the classes `Test` and `Main` are not in the same package, and `x` and `y` have package-level visibility (since no access modifier is specified).

Here's the error you'll encounter:

error: x has protected access in Test

System.out.println(t.x + " " + t.y);

error: y has protected access in Test

System.out.println(t.x + " " + t.y);

To fix this, you can either put both classes in the same package, or you can change the access modifier of `x` and `y` in the `Test` class to `public` or `private`.

If you change the `Test` class as follows:

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

Then the output will be: 0 0

This is because when you create an instance of `Test` with `new Test()`, `x` and `y` will be initialized to their default values, which is 0 for `int`.

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 : 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 : Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 1 out of bounds for length 1

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();
}
}
```

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 : obj1.a = 2 obj1.b = 3

obj2.a = 2 obj2.b = 3