**Oops Assignment**

1. Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.

2. **Subclass :** A class that is derived from another class is called a subclass (also a derived class, extended class, or child class).
   **Superclass :** The class from which the subclass is derived is called a superclass (also a base class or a parent class).

3. **extends** is the keyword used to inherit the properties of a class. Below given is the syntax of extends keyword.

   ```
   class A {
   public A() {
   System.out.println("New A");

   }

   }

   class B extends A {

   public B() {

   super();

   System.out.println("New B");

   }

   }
   ```

   super keyword is used to call the constructor of the base (parent) class.

4. The word polymorphism means having many forms. In simple words,we can define polymorphism as the ability of a message to be displayed in more than one form.
   In Java polymorphism is mainly divided into two types:
   •Compile-time Polymorphism
   •Runtime Polymorphism

5. **Method Overloading:**

• Method overloading in Java means having two or more methods (or functions) in a class with the same name and different arguments (or parameters). It can be with a different number of arguments or different data types of arguments.
• Method overloading is a compile-time polymorphism.
• It helps to increase the readability of the program.
• Method overloading may or may not require inheritance.

**Method Overriding:**

• Method overriding occurs when a subclass provides a particular implementation of a method declared by one of its parent classes.
• Method overriding is a run-time polymorphism.
• Method overriding always needs inheritance.

6. **Abstraction :** Abstraction is a process of hiding the implementation details and showing only functionality to the user.

```
abstract class Bike{
abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

7. **Abstract Method:**

A method declared using the abstract keyword within an abstract class and does not have a definition (implementation) is called an abstract method.

```
abstract class
abstract class Multiply {
public abstract int MultiplyTwo (int n1, int n2);
public abstract int MultiplyThree (int n1, int n2, int n3);

  public void show() {
   System.out.println ("Method of abstract class Multiply");
  }
}
class AbstractMethodEx1 extends Multiply {

  public int MultiplyTwo (int num1, int num2) {
   return num1 * num2;
  }
  public int MultiplyThree (int num1, int num2, int num3) {
   return num1 * num2 * num3;
```

```
  }


  public static void main (String args[]) {
   Multiply obj = new AbstractMethodEx1();
   System.out.println ("Multiplication of 2 numbers: " + obj.MultiplyTwo (10, 50));
   System.out.println ("Multiplication of 3 numbers: " + obj.MultiplyThree (5, 8, 10));
   obj.show();
  }
}
```

**Final Method:**

using the final keyword in a method declaration to indicate that the method cannot be overridden by subclasses.

```
class FinalDemo {

   public final void display() {
     System.out.println("This is a final method.");
   }
}

class Main extends FinalDemo {
   public final void display() {
   System.out.println("The final method is overridden.");
 }

 public static void main(String[] args) {
   Main obj = new Main();
   obj.display();
 }
}
```

8. In Java, the final class cannot be inherited by another class.

```
final class FinalClass {
 public void display() {
   System.out.println("This is a final method.");
 }
}


class Main extends FinalClass {
 public  void display() {
   System.out.println("The final method is overridden.");
 }
```

```
  public static void main(String[] args) {
    Main obj = new Main();
    obj.display();
  }
}
```

9. **Abstraction:**
   Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essential units are not displayed to the user.
   **Encapsulation:**
   In encapsulation the variables or data of a class is hidden from any other class and can be accessed only through any member function of its own class in which they are declared. As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding. 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.

10. **Runtime Polymorphism:**
    Whenever an object is bound with the functionality at run time, this is known as runtime polymorphism. The runtime polymorphism can be achieved by method overriding. Java virtual machines determine the proper method to call at the runtime, not at the compile time. It is also called dynamic or late binding. Method overriding says the child class has the same method as declared in the parent class. It means if the child class provides the specific implementation of the method that has been provided by one of its parent classes then it is known as method overriding.

    **Compile time Polymorphism:**
    Whenever an object is bound with its functionality at the compile time, this is known as the compile-time polymorphism. At compile-time, java knows which method to call by checking the method signatures. So this is called compile-time polymorphism or static or early binding. Compile-time polymorphism is achieved through method overloading. Method Overloading says you can have more than one function with the same name in one class having a different prototype. Function overloading is one of the ways to achieve polymorphism but it depends on technology and which type of polymorphism we adopt. In java, we achieve function overloading at compile-Time.