DHAKA INTERNATIONAL UNIVERSITY

ESTD - 7th APril, 1995

Knowledge is Power

DIU

Estd. 7th April, 1995

ASSIGNMENT ON

Explain polymorphism and Inheritance with java code

Course No: CSE207
Course Name: Object Oriented Programming

SUBMITTED TO

Jahanur BIshwas
Lecturer, Department of CSE, DIU

SUBMITTED BY

Md. Sharif Uddin
Roll : 18, Batch : E83

SUBMITTED ON APRIL 12, 2022

Assignment No: 01

Assignment Name: Explain polymorphism with java code.

Polymorphism: Polymorphism in Java is the task that performs a single action in different ways.

Why we use Polymorphism in Java: Polymorphism in Java makes it possible to write a method that can correctly process lots of different types of functionalities that have the same name. We can also gain consistency in our code by using polymorphism.

Advantages of Polymorphism in Java

1. It provides reusability to the code. The classes that are written, tested and implemented can be reused multiple times. Furthermore, it saves a lot of time for the coder. Also, one can change the code without affecting the original code.
2. A single variable can be used to store multiple data values. The value of a variable you inherit from the superclass into the subclass can be changed without changing that variable's value in the superclass; or any other subclasses.
3. With lesser lines of code, it becomes easier for the programmer to debug the code.

Problems with Polymorphism in Java: With lots of advantages, there are also a few disadvantages of polymorphism.

- Polymorphism is quite challenging while implementation.
- It tends to reduce the readability of the code.
- It raises some serious performance issues in real-time as well.

Characteristics of Polymorphism: Polymorphism has many other characteristics other than Method Overloading and Method Overriding. They include:

- Coercion
- Internal Operator Overloading
- Polymorphic Variables or Parameters

Types of Polymorphism

You can perform Polymorphism in Java via two different methods:

1. Method Overloading
2. Method Overriding

Method overloading is the process that can create multiple methods of the same name in the same class, and all the methods work in different ways. Method overloading occurs when there is more than one method of the same name in the class.

Method overriding is the process when the subclass or a child class has the same method as declared in the parent class.

Also, Polymorphism in Java can be classified into two types

1. Static/Compile-Time Polymorphism
2. Dynamic/Runtime Polymorphism

Compile Time Polymorphism In Java is also known as **Static Polymorphism.** Furthermore, the call to the method is resolved at compile-time. Compile-Time polymorphism is achieved through **Method Overloading**. This type of polymorphism can also be achieved through **Operator Overloading**. However, Java does not support **Operator Overloading**.

Runtime polymorphism in Java is also popularly known as **Dynamic Binding or Dynamic Method Dispatch.** In this process, the call to an overridden method is resolved dynamically at runtime rather than at compile-time. You can achieve Runtime polymorphism via **Method Overriding.**

Polymorphism in Java Example: A superclass named "Shapes" has a method called "area()". Subclasses of "Shapes" can be "Triangle", "circle", "Rectangle", etc. Each subclass has its way of calculating area. Using Inheritance and Polymorphism means, the subclasses can use the "area()" method to find the area's formula for that shape.

```
class Shapes {
  public void area() {
    System.out.println("The formula for area of ");
  }
}
class Triangle extends Shapes {
  public void area() {
    System.out.println("Triangle is ½ * base * height ");
  }
}
class Circle extends Shapes {
  public void area() {
    System.out.println("Circle is 3.14 * radius * radius ");
  }
}
class Main {
  public static void main(String[] args) {
    Shapes myShape = new Shapes();
    Shapes myTriangle = new Triangle();
    Shapes myCircle = new Circle();
    myShape.area();
    myTriangle.area();
    myShape.area();
    myCircle.area();
  }
}
```

Output:

```
The formula for the area of Triangle is ½ * base * height

The formula for the area of the Circle is 3.14 * radius *
radius
```

END THE ASSIGNMENT 01

Assignment No: 02

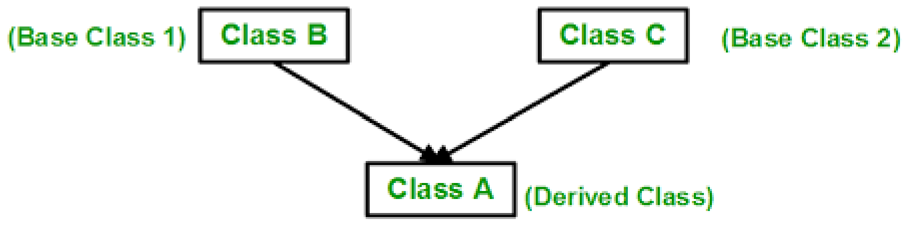
Assignment Name: Explain Inheritance with java code.

Inheritance: Inheritance is a method in which one object acquires/inherits another object's properties, and inheritance also supports hierarchical classification. The idea behind this is that we can create new classes built on existing classes, i.e., when you inherit from an existing class, we can reuse methods and fields of the parent class. Inheritance represents the parent-child relationship.

For example, a whale is a part of the classification of marine animals, which is part of class mammal, which is under that class of animal. We use hierarchical classification, i.e., top-down classification. If we want to describe a more specific class of animals such as mammals, they would have more specific attributes such as teeth; cold-blooded, warm-blooded, etc. This comes under the subclass of animals where animals come under superclass. The subclass is a class which inherits properties of the superclass. This is also called a derived class. A superclass is a base class or parental class from which a subclass inherits properties.

We use inheritance mainly for method overriding and R:

To inherit a class, we use the extend keyword.



Advantages of Inheritance in Java

1. Inheritance promotes reusability. ...
2. Reusability enhanced reliability. ...
3. As the existing code is reused, it leads to less development and maintenance costs.
4. Inheritance makes the sub classes follow a standard interface.
5. Inheritance helps to reduce code redundancy and supports code extensibility

There are five types of inheritance

1. Single
2. Multilevel
3. Multiple
4. Hybrid
5. Hierarchical.

Single level: In this one class i.e., derived class inherits properties from its parental class. This enables code reusability and also adds new features to the code.

Multilevel: This one class is derived from another class which is also derived from another class i.e., this class has more than one parental class, hence it is called multilevel inheritance.

Hierarchical level: In this one parental class has two or more derived classes or we can say that two or more child classes have one parental class.

Hybrid inheritance: This is the combination of multiple and multilevel inheritance and in java multiple inheritance is not supported as it leads to ambiguity and this type of inheritance can only be achieved through interfaces.

Consider that class a is the parental or base class of class b and class c and in turn class b and class c are parental or base class of class d. Class b and class c are derived classes from class a and class d is derived class from class b and class c.

Following program creates a super class called add and a subclass called sub, uses extend keyword to create a subclass add.

As displayed in the above figure, Programmer is the subclass and Employee is the superclass. The relationship between the two classes is **Programmer IS-A Employee**. It means that Programmer is a type of Employee.

Inheritance java example code: As displayed in the above figure, Programmer is the subclass and Employee is the superclass. The relationship between the two classes is Programmer IS-A Employee. It means that Programmer is a type of Employee.

```
class Animal {

  String name;
  public void eat() {
        System.out.println("I can eat");
  }
}

class Dog extends Animal {

  public void display() {
        System.out.println("My name is " + name);
  }
}

class Main {
  public static void main(String[] args) {

        Dog labrador = new Dog();

        labrador.name = "Rohu";
        labrador.display();
        labrador.eat();
  }
}
```

Output

```
My name is Rohu

I can eat
```

END THE ASSIGNMENT 02