VIPS
Technical Campus
योग: कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

SCHOOL OF ENGINEERING AND TECHNOLOGY

Course : OBJECT ORIENTED PROGRAMMING

Paper Code: AIML-202,IIOT-202

Faculty : Dr. Shivanka

Assistant Professor

VIPS

What is Polymorphism?

- The derivation of the word Polymorphism is from two different Greek words- poly and morphs. “Poly” means numerous, and “Morphs” means forms. So, polymorphism means innumerable forms.Polymorphism, therefore, is one of the most significant features of Object-Oriented Programming.

Real-Life Examples of Polymorphism

- An individual can have different relationships with different people. A male can be a father, a customer, an employee, and a friend, all at the same time, i.e. he performs other behaviors in different situations.

Real-Life Examples of Polymorphism

- The human body has different organs. Every organ has a different function to perform; the heart is responsible for blood flow, the lungs for breathing, the brain for cognitive activity, and the kidneys for excretion. So we have a standard method function that performs differently depending upon the organ of the body.

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.

Compile-Time Polymorphism in Java

- Compile Time Polymorphism exist at the tIme of compilation is called compile time polymorphism in Java is also known as Static Polymorphism or early binding. .
- 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.

Compile-Time Polymorphism in Java

- Method Overloading:- whenever a class contain more than one method or multiple methods with the same name, but types of parameters and the return type of the methods are different are called Method Overloading.
- Syntax: return_type method_name(para1);
- return_type method_name(para1,para2);

Compile-Time Polymorphism in Java


```
package compiletimepolymorphism;
public class methodoverloading{
void add(){
int a=10,b=20,c;
c=a+b;
System.out.println("Addition Value of int a=10 & int b =20: ");
System.out.println(c);
}
void add(int x,int y){
int c;
c=x+y;
System.out.println("Addition Value of parameter int x =34 & int y=10 :
");
System.out.println(c);
}
void add(int x,double y){
double c;
c=x+y;
System.out.println("Addition Value of int x=16 & double y =10.5:");
System.out.println(c);
}

public static void main(String[] args){
methodoverloading r=new methodoverloading();
r.add();//call add() method with no parameters
r.add(34,10); //call add() method with two  int parameters
 r.add(16,10.5);//call add() method with int and double parameters

}
}
```

Run Time Polymorphism in Java Example


```
public class polymorphismshapes {
  public void area() {
    System.out.println("The formula for area of ");
  }}
class Triangle extends polymorphismshapes{
  public void area(){
    System.out.println("Triangle is ½ * base * height ");
  }}
class Circle extends polymorphismshapes {
  public void area() {
    System.out.println("Circle is 3.14 * radius * radius ");
  }}
```

```
 class Main {
  public static void main(String[] args) {
 polymorphismshapes myShape = new
polymorphismshapes();
// Create a polymorphismshapes class object
    polymorphismshapes myTriangle = new Triangle();
// Create a Triangle object
    polymorphismshapes myCircle = new Circle();
// Create a Circle object
    myShape.area();
    myTriangle.area();
    myShape.area();
    myCircle.area();
  }}
```

Polymorphism in Java Example

- Output:
- The formula for the area of the Triangle is ½ * base * height
- The formula for the area of the Circle is 3.14 * radius * radius

- In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

- Let's first understand the upcasting before Runtime Polymorphism.

- Upcasting
- If the reference variable of Parent class refers to the object of Child class, it is known as upcasting. For example

Run-Time Polymorphism


```
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");
    }
}
```

```
class Square extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a square");
    }
}
class Main {
    public static void main(String[] args) {
        Shape s1 = new Circle();
        Shape s2 = new Square();
        s1.draw(); // Output: "Drawing a circle"
        s2.draw(); // Output: "Drawing a square"
    }}
```

Output

- The program will output: “Drawing a circle” and “Drawing a square

Thank You