Polymorphism in Java

Polymorphism in Java is a concept by which we can perform a single action in different ways. Polymorphism is derived from 2 Greek words: poly and morphs. The word "**poly**" means many and "**morphs**" means forms. So polymorphism means many forms.

There are two types of polymorphism in Java: **compile-time polymorphism** and **runtime polymorphism**. We can perform polymorphism in java by method overloading and method overriding.

Runtime Polymorphism in Java

Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

Since method invocation is determined by the JVM not compiler, it is known as runtime polymorphism.

Example 1

```
class Bike
{
  void run()
   {
     System.out.println("running");
    }
}
class Splendor extends Bike
{
  void run()
{
System.out.println("running safely with 60km");
}
  public static void main(String args[])
```

```
{
   Bike b = new Splendor();
   b.run();
  }
}
```

Java Runtime Polymorphism Example: Bank

Consider a scenario where Bank is a class that provides a method to get the rate of interest. However, the rate of interest may differ according to banks. For example, SBI, ICICI, and AXIS banks are providing 8.4%, 7.3%, and 9.7% rate of interest.

Example 2

```
class Bank
{
float getRateOfInterest()
{
return 0;
}
}
class SBI extends Bank
{
float getRateOfInterest()
{
return 8.4f;
}
}
class ICICI extends Bank
{
```

```
float getRateOfInterest()
{
return 7.3f;
}
}
class AXIS extends Bank
{
float getRateOfInterest()
{
return 9.7f;
}
}
class TestPolymorphism
{
public static void main(String args[])
{
Bank b;
b=new SBI();
System.out.println("SBI Rate of Interest: "+b.getRateOfInterest());
b=new ICICI();
System.out.println("ICICI Rate of Interest: "+b.getRateOfInterest());
b=new AXIS();
System.out.println("AXIS Rate of Interest: "+b.getRateOfInterest());
}
}
```

Java Runtime Polymorphism Example: Shape

```
class Shape
{
void draw()
{
System.out.println("drawing...");
}
}
class Rectangle extends Shape
{
void draw()
{
System.out.println("drawing rectangle...");
}
}
class Circle extends Shape
{
void draw()
{
System.out.println("drawing circle...");
}
}
class Triangle extends Shape
{
void draw()
{
System.out.println("drawing triangle...");
```

```
}
}
class TestPolymorphism2
{
public static void main(String args[])
{
Shape s;
s=new Rectangle();
s.draw();
s=new Circle();
s.draw();
s=new Triangle();
s.draw();
}
}
```

Java Runtime Polymorphism Example: Animal

```
class Animal
{
void eat()
{
System.out.println("eating...");
}
}
class Dog extends Animal
{
void eat()
```

```
{
System.out.println("eating bread...");
}
}
class Cat extends Animal{
void eat(){System.out.println("eating rat...");
}
}
class Lion extends Animal
{
void eat()
{
System.out.println("eating meat...");
}
}
class TestPolymorphism3
{
public static void main(String[] args)
{
Animal a;
a=new Dog();
a.eat();
a=new Cat();
a.eat();
a=new Lion();
a.eat();
```

```
}
}
```

Java Runtime Polymorphism with Multilevel Inheritance

Let's see the simple example of Runtime Polymorphism with multilevel inheritance.

```
class Animal
{
void eat()
{
System.out.println("eating");
}
}
class Dog extends Animal
{
void eat()
{
System.out.println("eating fruits");
}
}
class BabyDog extends Dog
{
void eat()
{
System.out.println("drinking milk");
}
public static void main(String args[])
```

```
{
Animal a1,a2,a3;
a1=new Animal();
a2=new Dog();
a3=new BabyDog();
a1.eat();
a2.eat();
a3.eat();
}
}
```