Object Oriented Programming using Java

Prepared By:
Suyel, PhD
Assistant Professor
Dept. of CSE, NIT Patna

Outline

1. Polymorphism
2. Compile-Time Polymorphism
3. Runtime Polymorphism
4. Demonstration of Polymorphism

Polymorphism

- 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.
- Example:
  - A person at the same time can have different characteristics like a man at the same time a father, a brother, an employee, etc.
  - So, the same person posses different behaviour in different situations. This is called polymorphism.

Polymorphism (Cont…)

- It is considered as one of the important features of OOP.
- It allows us to perform a single action in different ways.
- In other words, polymorphism allows us to define one interface and have multiple implementations.
- There are two types of polymorphism in Java:
  1. Compile-time polymorphism
  2. Runtime polymorphism.
- We can perform polymorphism in Java by method overloading and method overriding.

Compile-Time Polymorphism

- Polymorphism that is resolved during compiler time is known as compile-time polymorphism.
- It is also known as static polymorphism. This type of polymorphism is achieved by method overloading.
- At compile time, Java knows which method to invoke by checking the method signatures.

Compile-Time Polymorphism (Cont…)

```
class Method5
{
   public int sum(int x, int y)
   {
      return (x + y);
   }

   public int sum(int x, int y, int z)
   {
      return (x + y + z);
   }

   public double sum(double x, double y)
   {
      return (x + y);
   }

   public static void main(String args[])
   {
      Method5 obj = new Method5();
      System.out.println(obj.sum(5, 10));
      System.out.println(obj.sum(5, 10, 20));
      System.out.println(obj.sum(20.5, 40.3));
   }
}
```

Compile-Time Polymorphism (Cont…)

- Output

  15

  35

  60.8

Runtime Polymorphism

- Runtime polymorphism (dynamic polymorphism or dynamic method dispatch) is a process in which a call to an overridden method is resolved at runtime.
- This type of polymorphism is achieved by method overriding.
- In this process, an overridden method is called through the reference variable of a superclass.
- If the reference variable of parent class refers to the object of child class, it is known as **upcasting**.
- For upcasting, we can use the reference variable of class type or an interface type.

Runtime Polymorphism (Cont…)

```
class Animal
{
   public void disp()
   {
      System.out.println("It is an animal");
   }
}

class Dog extends Animal
{
   public void disp()
   {
      System.out.println("It is actually a dog");
   }
}

class Over
{
   public static void main(String args[])
   {
      Animal obj = new Dog();
      obj.disp();
   }
}
```

Runtime Polymorphism (Cont…)

- Output

  It is actually a dog

Demonstration of Polymorphism

- Memory is allocated for reference variable at compile time and method overloading depends on the type of reference variable.
- Object is created at runtime and calling of overriding method depends on object.
- There are mainly four cases:
  1. Parent reference and parent object
  2. Child reference and child object
  3. Parent reference and child object
  4. Child reference and parent object

Demonstration of Polymorphism (Cont…)

- Parent reference and parent object

```
class Father
{
   public void color()
   {
      System.out.println("Medium fair from Father Class");
   }

   public void work()
   {
      System.out.println("Employee from Father class");
   }
}

class Child extends Father
{
   public void color()
   {
      System.out.println("Fair from Child class");
   }

   public void work()
   {
      System.out.println("Employee from Child class");
   }
}

class Over1
{
   public static void main(String args[])
   {
      Father obj = new Father();
      obj.color();
      obj.work();
   }
}
```

Demonstration of Polymorphism (Cont…)

- Parent reference and parent object (Cont…)
  - Output

    Medium fair from Father Class

    Employee from Father class

Demonstration of Polymorphism (Cont…)

- Child reference and child object

```
class GrandFather
{
   void color()
   {
      System.out.println("Medium fair");
   }
}

class Father extends GrandFather
{
   void color()
   {
      System.out.println("Medium fair");
   }
}

class Child extends Father
{
   void color()
   {
      System.out.println("fair");
   }

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

Demonstration of Polymorphism (Cont…)

- Child reference and child object (Cont…)
  - Output

    Fair

Demonstration of Polymorphism (Cont…)

- Parent reference and child object

```
class Father
{
   public void color()
   {
      System.out.println("Medium fair from Father Class");
   }

   public void work()
   {
      System.out.println("Employee from Father class");
   }
}

class Child extends Father
{
   public void color()
   {
      System.out.println("Fair from Child class");
   }

   public void work()
   {
      System.out.println("Employee from Child class");
   }
}

class Over3
{
   public static void main(String args[])
   {
      Father obj = new Child();
      obj.color();
      obj.work();
   }
}
```

Demonstration of Polymorphism (Cont…)

- Parent reference and child object (Cont…)
  - Output

    Fair from Child class

    Employee from Child class

Demonstration of Polymorphism (Cont…)

- Child reference and parent object

```
class Father
{
   public void color()
   {
      System.out.println("Medium fair from Father Class");
   }

   public void work()
   {
      System.out.println("Hello, how are you?");
   }
}

class Child extends Father
{
   public void color()
   {
      System.out.println("Fair from Child class");
   }

   public void work()
   {
      System.out.println("NIT Patna");
   }
}

class Over4
{
   public static void main(String args[])
   {
      Child obj1 = new Father();
      obj1.color();
      obj1.work();
   }
}
```

Demonstration of Polymorphism (Cont…)

- Child reference and parent object (Cont…)
  - Output

```
Over6.java:32: error: incompatible types: Father cannot be converted to Child
      Child obj1 = new Father();
                   ^
1 error
```

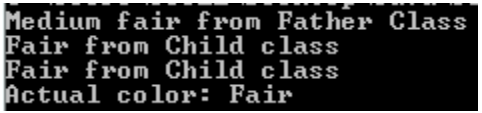
Demonstration of Polymorphism (Cont…)

- Combining all four concepts

```
class Father
{
    public void color()
    {
        System.out.println("Medium fair from Father Class");
    }
}

class Child extends Father
{
    public void color()
    {
        System.out.println("Fair from Child class");
    }

    public void color(String str)
    {
        System.out.println("Actual color: " + str);
    }
}

class Over5
{
    public static void main(String args[])
    {
        Father obj1 = new Father();
        obj1.color();
        Child obj2 = new Child();
        obj2.color();
        Father obj3 = new Child();
        obj3.color();
        Child obj4 = new Child();
        obj4.color("Fair");
        //Father obj5 = new Father();
        //obj5.color("Medium Fair");
        //Father obj6 = new Child();
        //obj6.color("Dark");

    }
}
```

Demonstration of Polymorphism (Cont…)

- Combining all four concepts (Cont…)
  - Output




thank
you . . .

Slides are prepared from various sources, such as Book, Internet Links and many more.