Department: BCA
Faculty Name: Er. Ashish Kumar
Subject: Java
Topic: Polymorphism in Java

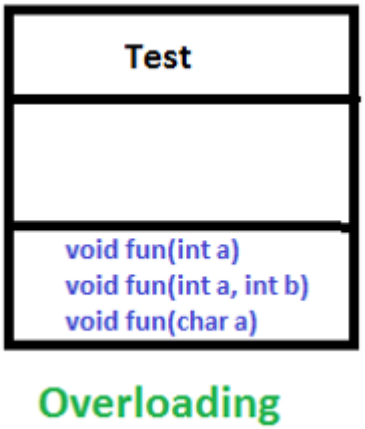
Polymorphism in Java

- Polymorphism is considered as one of the important features of Object Oriented Programming.
- Polymorphism allows us to perform a single action in different ways.
- In other words, polymorphism allows you to define one interface and have multiple implementations.
- The word "poly" means many and "morphs" means forms, So it means many forms.

In Java polymorphism is mainly divided into two types:

1. Compile time Polymorphism
2. Runtime Polymorphism

1. **Compile time Polymorphism:** It is also known as static polymorphism. This type of polymorphism is achieved by method overloading or operator overloading.



- **method overloading:** Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters or both. Overloading is related to compile-time (or static) polymorphism.

Example:

C:\Users\ashishjha\Desktop\java\Sum.java - Notepad++

```java
//example of method overloading
public class Sum
{


    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[])
    {
        Sum s = new Sum();
        System.out.println(s.sum(10, 20));
        System.out.println(s.sum(10, 20, 30));
        System.out.println(s.sum(10.5, 20.5));
    }
}
```

Command Prompt

```
a

C:\Users\ashishjha\Desktop\java>java Sum
30
60
31.0

C:\Users\ashishjha\Desktop\java>
```

- **Operator Overloading:** Java also provide option to overload operators. For example, we can make the operator ('+') for string class to concatenate two strings. We know that this is the addition operator whose task is to add two operands. So a single operator '+' when placed between integer operands, adds them and when placed between string operands, concatenates them. In java, Only "+" operator can be overloaded

Example

C:\Users\ashishjha\Desktop\java\Test.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

abc.html | sss.html | slides.html | Main.java | XYZ.java | Sum.java | Test.java

```java
//Example of operator overloading
class OperatorOverload {

    void operator(String str1, String str2)
    {
        String s = str1 + str2;
        System.out.println("Concatinated String - "
                                + s);
    }

    void operator(int a, int b)
    {
        int c = a + b;
        System.out.println("Sum = " + c);
    }
}

class Test {
    public static void main(String[] args)
    {
        OperatorOverload obj = new OperatorOverload();
        obj.operator(10, 20);
        obj.operator("Ashish", "Jha");
    }
}
```

Command Prompt

```
C:\Users\ashishjha\Desktop\java>javac Test.java

C:\Users\ashishjha\Desktop\java>java Test
Sum = 30
Concatinated String - AshishJha
```

Example:

```java
class Num {

    // Method with 2 parameter
    static int mul(int a, int b)
    {
        return a * b;
    }

    // Method with the same name but 3 parameter
    static int mul(int a, int b, int c)
    {
        return a * b * c;
    }
}

class Main1 {
    public static void main(String[] args)
    {

        Num obj=new Num();
        System.out.println(obj.mul(2,4));
        //System.out.println(Num.mul(2, 4));

        //System.out.println(Num.mul(2, 7, 3));
        System.out.println(obj.mul(2, 7, 3));
    }
}
```

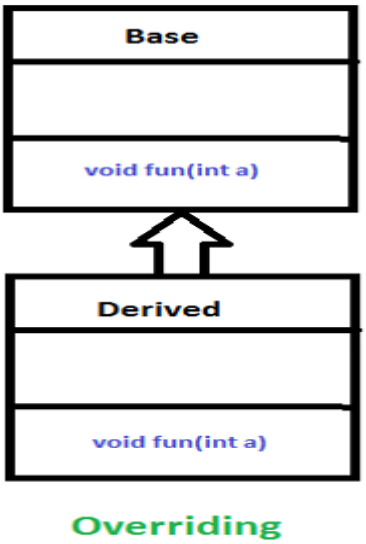
```
C:\Users\ashishjha\Desktop\java>javac Main1.java

C:\Users\ashishjha\Desktop\java>java Main1
8
42

C:\Users\ashishjha\Desktop\java>
```

2. **Runtime polymorphism:** It is also known as Dynamic Method Dispatch (dynamic polymorphism). It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding.

- **Method overriding**, on the other hand, occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be **overridden**.



Example:

```java
//example of method overriding.
class Bike
{
  void run()
  {
  System.out.println("running...");
  }
}
class Splendor extends Bike
{
  void run()
  {
  System.out.println("running safely with 60km");
  }
}
class Pulsar extends Bike
{
  void run()
  {
  System.out.println("running safely with 70km");
  }
}
class TestRun
{
  public static void main(String args[])
  {
    Bike obj1 = new Splendor();//upcasting
    Bike obj2 =new Pulsar();
    obj1.run();
    obj2.run();
  }
}
```

```
C:\Users\ashishjha\Desktop\java>javac TestRun.java

C:\Users\ashishjha\Desktop\java>java TestRun
running safely with 60km
running safely with 70km

C:\Users\ashishjha\Desktop\java>
```

Example:

C:\Users\ashishjha\Desktop\java\TestBank.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

abc.html | sss.html | slides.html | Main.java | XYZ.java | Sum.java | Test.java | Main1.java | TestRun.java | TestBank.java

```java
class RBI
{
    float getInterest()
    {
    return 0;
    }
}
class SBI extends RBI
{
    float getInterest()
    {
    return 3.0f;
    }
}
class CBI extends RBI
{
     float getInterest()
     {
     return 3.5f;
     }
}
class PNB extends RBI
{
     float getInterest()
     {
     return 4.0f;
     }
}
class TestBank
{
     public static void main(String args[])
      {
     RBI b;
     b=new SBI();
     System.out.println("SBI Rate of Interest: "+b.getInterest());
     b=new CBI();
     System.out.println("CBI Rate of Interest: "+b.getInterest());
     b=new PNB();
     System.out.println("PNB Rate of Interest: "+b.getInterest());
      }
}
```

Command Prompt

```
C:\Users\ashishjha\Desktop\java>javac TestBank.java

C:\Users\ashishjha\Desktop\java>java TestBank
SBI Rate of Interest: 3.0
CBI Rate of Interest: 3.5
PNB Rate of Interest: 4.0

C:\Users\ashishjha\Desktop\java>
```