
LAB 7

Program to demonstrate the concept of polymorphism

Definitions

Polymorphism – Polymorphism is another important OOP feature. With the help of this feature, we can perform a single action in multiple ways. In java there are 2 kinds of polymorphism – compile-time and runtime.

Compile-time polymorphism – Compile-time polymorphism is achieved through method overloading. We can have multiple methods with the same name, but with different number/type of parameters and different return type. The method to be used is decided during compile time. It is also called static or early binding

Method overriding - In overriding, we have 2 methods with the same name and same prototype, but in different classes (parent and child class). The implementation of the method is different in both classes. An object of the child class will access the method in the child class and an object of the parent class will access the method in the parent class.

Runtime polymorphism – This is also called dynamic method dispatch. An object of child class is made with a reference to the superclass. When we call methods using this object, it calls the method of the child class. However, if there is an overloaded method where the child class has different number/type of parameters or different return type, then we can call the method of the parent class by passing appropriate parameters.

Static methods cannot be overridden but they can be overloaded.

Code

```
/* Program to demonstrate
   the concept of polymorphism */

import java.util.Random;
import java.io.*;

class CalculateSum {

    static int Calculate(int a, int b) {
        return a + b;
    }

    int Calculate(int a, int b, int c) {
        return a + b + c;
    }

    double Calculate(double a, double b) {
        return a + b;
    }
}
```

```

    }
}

class CalculateDifference extends CalculateSum{

    static int Calculate(int a, int b) {
        return a - b;
    }

    double Calculate(double a, double b) {
        return a - b;
    }
}

public class Lab7 {
    public static void main(String[] args) {

        /* We have two classes, calculate sum and
        calculate difference.
        They both have overridden and overloaded funtion Calculate.
        We call this function for both classes to observe
        polymorphism in runtime and compile time. */

        //We first create random integer and double values
        Random rand = new Random();
        int I1 = rand.nextInt(50);
        int I2 = rand.nextInt(50);
        int I3 = rand.nextInt(50);
        double D1 = rand.nextDouble(50);
        double D2 = rand.nextDouble(50);

        //Creating objects of both classes
        CalculateSum CS = new CalculateSum();
        CalculateDifference CD = new CalculateDifference();

        //We output into a file called Polymorphism.txt
        try (FileWriter f = new FileWriter("Polymorphism.txt");
            PrintWriter p = new PrintWriter(f);) {

            p.println("The random integers are: ");
            p.println("I1 = " + I1 + "; I2 = " + I2 + "; I3 = " + I3);

            p.println("The random doubles are: ");
            p.printf("D1 = %.2f; D2 = %.2f\n", D1, D2);
            p.println();

            p.println("Calling Calculate function of class CalculateSum:");
            p.println(I1 + " + " + I2 + " = " + CalculateSum.Calculate(I1, I2));
            p.println(I1 + " + " + I2 + " + " + I3 + " = " + CS.Calculate(I1, I2,
I3));

            p.printf("%.2f + %.2f = %.2f\n", D1, D2, CS.Calculate(D1, D2));
            p.println();

```

```

        p.println("Calling Calculate function of class CalculateDifference:");
        p.println(I1 + " - " + I2 + " = " + CalculateDifference.Calculate(I1,
I2));

        p.printf("%.2f - %.2f = %.2f\n", D1, D2, CD.Calculate(D1, D2));
        p.println();

        p.println("Using Dynamic Method Dispatch:");
        CalculateSum dynamic = new CalculateDifference();
        p.println(I2 + " + " + I3 + " = " + dynamic.Calculate(I2, I3));
        p.printf("%.2f - %.2f = %.2f\n", D1, D2, dynamic.Calculate(D1, D2));
        p.print(I1 + " + " + I2 + " + " + I3);
        p.print(" = " + dynamic.Calculate(I1, I2, I3));

        System.out.println("File Polymorphism.txt created and written.");

    } catch (IOException i) {
        i.printStackTrace();
    }
}

```

This line gives us a warning because we are trying to override a static method

Explanation

- We create 2 classes CalculateSum and CalculateDifference. CalculateDifference is a child of CalculateSum.
- Both classes have methods called Calculate.
- In CalculateSum, we have 1 static method called calculate which takes 2 integer parameters and returns an integer. For the explanation let us call it *Static1*.
- CalculateSum has 2 more methods which are not static.
- The first one takes 3 integers and returns an integer which is the sum of the three and the next takes 2 double values and returns their sum as a double.
- CalculateDifference also has 2 methods. The first one is static and takes 2 integer parameters, returning their difference as an integer. Let us call it *Static2*.
- The other method is not static; it takes 2 doubles and returns their difference as a double.
- We create an object for each class and call the Calculate function for them. We vary the number and type of parameters and get different results. This shows us method overloading.
- We next create an object of CalculateDifference by referencing CalculateSum. We call it dynamic.

```
CalculateSum dynamic = new CalculateDifference();
```

- When we try to call dynamic.Calculate(I2, I3), we get a warning. This is because we are trying to override the static function from parent class in the child class. Since this is not allowed, the output will be a sum of integers instead of a difference. i.e., Static1 is called instead of Static2.

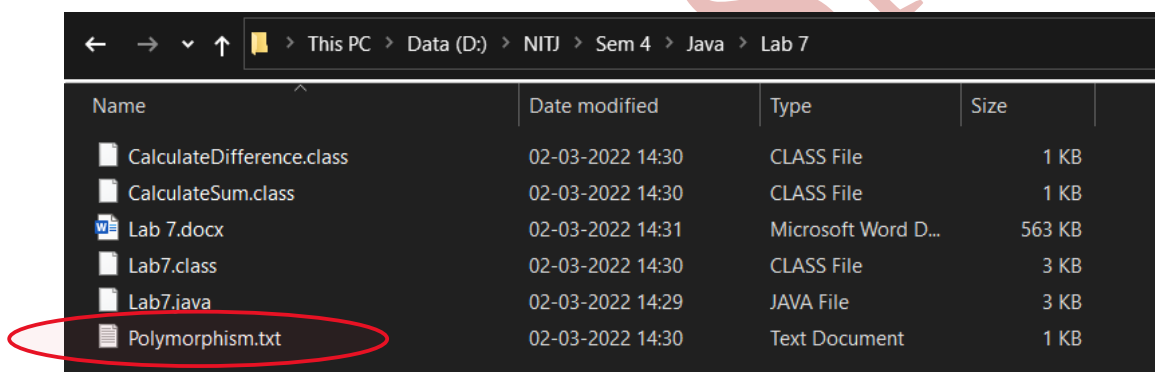
- However, when we call `dynamic.Calculate(D1, D2)`, it calls the function present in the child class and gives us the difference.
- We can also call the method from the parent class which takes 3 integer parameters.

```
p.print(I1 + " + " + I2 + " + " + I3);
p.print(" = " + dynamic.Calculate(I1, I2, I3));
```

- This gives us the sum. This is an example of dynamic method dispatch. It occurs during runtime.

Output

```
PS D:\NITJ\Sem 4\Java\Lab 7> javac Lab7.java
PS D:\NITJ\Sem 4\Java\Lab 7> java Lab7
File Polymorphism.txt created and written.
PS D:\NITJ\Sem 4\Java\Lab 7> █
```



Name	Date modified	Type	Size
CalculateDifference.class	02-03-2022 14:30	CLASS File	1 KB
CalculateSum.class	02-03-2022 14:30	CLASS File	1 KB
Lab 7.docx	02-03-2022 14:31	Microsoft Word D...	563 KB
Lab7.class	02-03-2022 14:30	CLASS File	3 KB
Lab7.java	02-03-2022 14:29	JAVA File	3 KB
Polymorphism.txt	02-03-2022 14:30	Text Document	1 KB

```
Polymorphism.txt - Notepad
File Edit Format View Help
The random integers are:
I1 = 28; I2 = 36; I3 = 44
The random doubles are:
D1 = 49.28; D2 = 46.66

Calling Calculate function of class CalculateSum:
28 + 36 = 64
28 + 36 + 44 = 108
49.28 + 46.66 = 95.94

Calling Calculate function of class CalculateDifference:
28 - 36 = -8
49.28 - 46.66 = 2.62

Using Dynamic Method Dispatch:
36 + 44 = 80
49.28 - 46.66 = 2.62
28 + 36 + 44 = 108
```