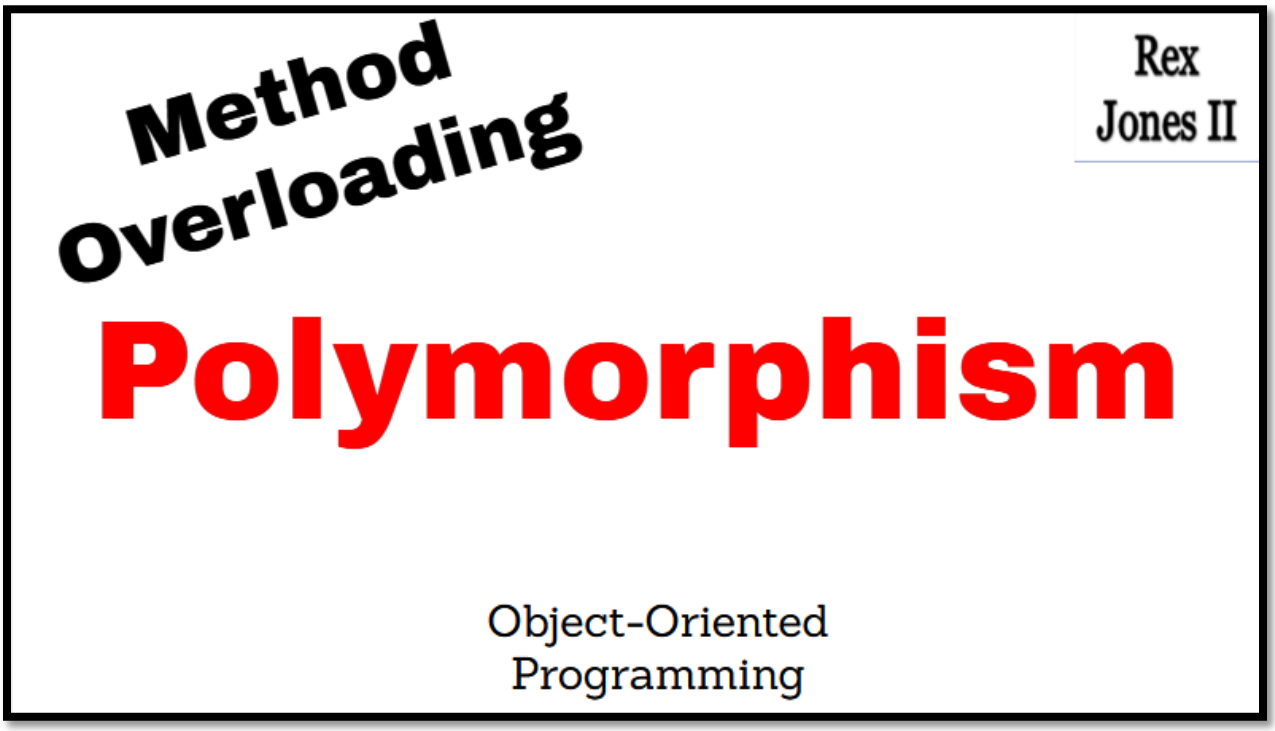
Rex
Jones II

(Transcript) Polymorphism
Method Overloading



Polymorphism

Introduction

Polymorphism is closely related to Inheritance. You know how Inheritance represents an “is a” relationship. Polymorphism also has that same kind of relationship so UnderGraduate “is a” Student and Student “is a” Person. Therefore, one instance such as UnderGraduate can pick up the behaviors and attributes of more than one class. That’s how Polymorphism maintains the capacity to take on many forms.

With this diagram, UnderGraduate has 3 forms. UnderGraduate has the form of a Person, UnderGraduate has the form of a Student, and UnderGraduate also has the form of its own type UnderGraduate.

There are 2 ways to implement Polymorphism: Method Overloading and Method Overriding. Method Overloading involves the same class and Method Overriding involves different classes.

In this Polymorphism session, we will investigate Method Overloading, Method Overriding, and both Java Bindings. Method Overloading refers to Static Binding while Method Overriding refers to Dynamic Binding. Let’s start with Method Overloading.

Rex Jones II

Method Overloading

In Java, Method Overloading is when more than 1 method has the same name but different parameters. Go to Eclipse. We see 2 methods named add. However, there's an error. The error states "Duplicate method add(int, int) in type Overload". That means both methods share the same name and share the same parameter declarations.

```
5   public static void add (int num1, int num2) {
6      System.out.println("Add 2 int Numbers = " + (num1 + num2));
7   }
8
9   Duplicate method add(int, int) in type Overload
10     System.out.println("Add int & int Numbers = " + (num1 + num2));
11  }
```

To overload a method, the parameter list must be unique. The type and/or number of parameters must not be the same. Let's start with the type. Change the 2nd type to double, include double in the print statement, save, and now the error goes away. Copy and Paste.

```
public static void add (int num1, int num2) {
   System.out.println("Add 2 int Numbers = " + (num1 + num2));
}

public static void add (int num1, double num2) {
   System.out.println("Add int & double Numbers = " + (num1 + num2));
}
```

We can also swap the types int and double to overload a method. Also swap in the print statements. All 3 methods have a unique order of types. First method int, int. Second method has int, double and Third method has double, int.

```
public static void add (double num1, int num2) {
   System.out.println("Add double & int Numbers = " +
                      (num1 + num2));
}
```

Rex Jones II

Now, let's look at the number of parameters. Copy and Paste then add int num3. We see every way to overload a method because the type and/or number of parameters are unique. Also update the print statement: 2 int Numbers and include num3 in the parenthesis.

```
public static void add (double num1, int num2, int num3) {
    System.out.println("Add double & 2 int Numbers = " +
                       (num1 + num2 + num3));
}
```

The return type has no impact on overloading a method because Java does not decide which method to execute based on the return type. If I copy and paste

then change the return type to int. Remove the print statement then assign (num1 + num2 + num3) to int sum =. Print statement sysout("Add 3 int Numbers =" + sum) and return statement includes sum. Hover both errors and they show "Duplicate method add(double, int, int) in type Overload". The return types are different but there's still an error. A difference in return types cannot be used as an overloaded method. Change the first type to int and now we have our overloaded method.

```
public static int add (int num1, int num2, int num3) {
    int sum = num1 + num2 + num3;
    System.out.println("Add 3 int Numbers = " + sum);
    return sum;
}
```

A constructor can be overloaded by using this same principle. Next, we will take a look at Method Overriding.