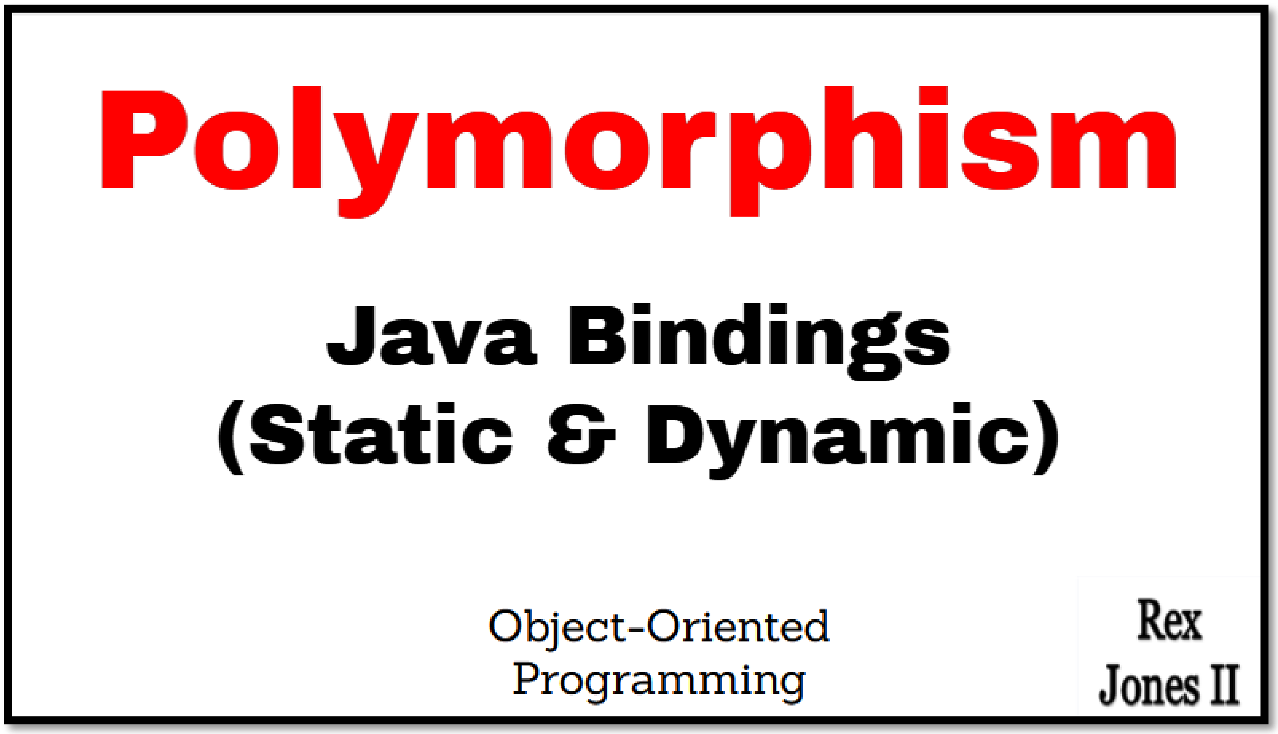
Rex Jones II

(Transcript) Polymorphism Java Bindings (Static & Dynamic)


Polymorphism

Java Bindings (Static & Dynamic)

Object-Oriented Programming

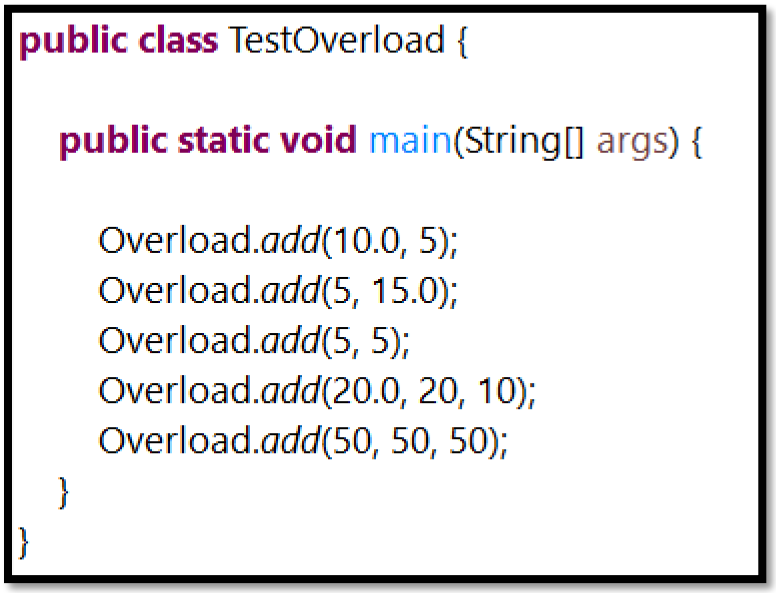
Rex Jones II

The Java Bindings are static and dynamic. Static happens at compile time while Dynamic happens at runtime. Method Overloading is an example of Static Binding and Method Overriding is an example of Dynamic Binding.

Let's start with the Static Binding by testing all 5 of these static overloaded methods. Static means they are shared between the whole class. As a result, static methods are bound to their implementations at compile time. The methods can be called directly by its class name. Therefore, it's not important to create an object to access the 5 static methods.

1. We can write the class name Overload. and we see all 5 methods. Select double, int then write 10.0 and 5.
2. Next is Overload. int, double and write 5, 15.0.
3. Third method is Overload. int, int. Let's write 5, 5.
4. Number 4 has 3 parameters Overload. double, int, int then write 20.0, 20, 10.

5. The last method had an int return type. Overload. Do you see how 4 methods have void as their return type and one method has int? Write 50, 50, 50. Let's Run. We see a print statement for each overloaded method.



```
public class TestOverload {

  public static void main(String[] args) {

    Overload.add(10.0, 5);
    Overload.add(5, 15.0);
    Overload.add(5, 5);
    Overload.add(20.0, 20, 10);
    Overload.add(50, 50, 50);
  }
}
```

Java restricts static methods from being overridden. Dynamic Binding allows for overridden methods which are multiple implementations that can be called depending on the instance. For example, let's think about the calculateGPA methods from the Student and Graduate classes. That we did in the previous session for Method Overriding. We can use a separate implementation for Student and for Graduate. However, I am going to implement one Graduate and another one for UnderGraduate.

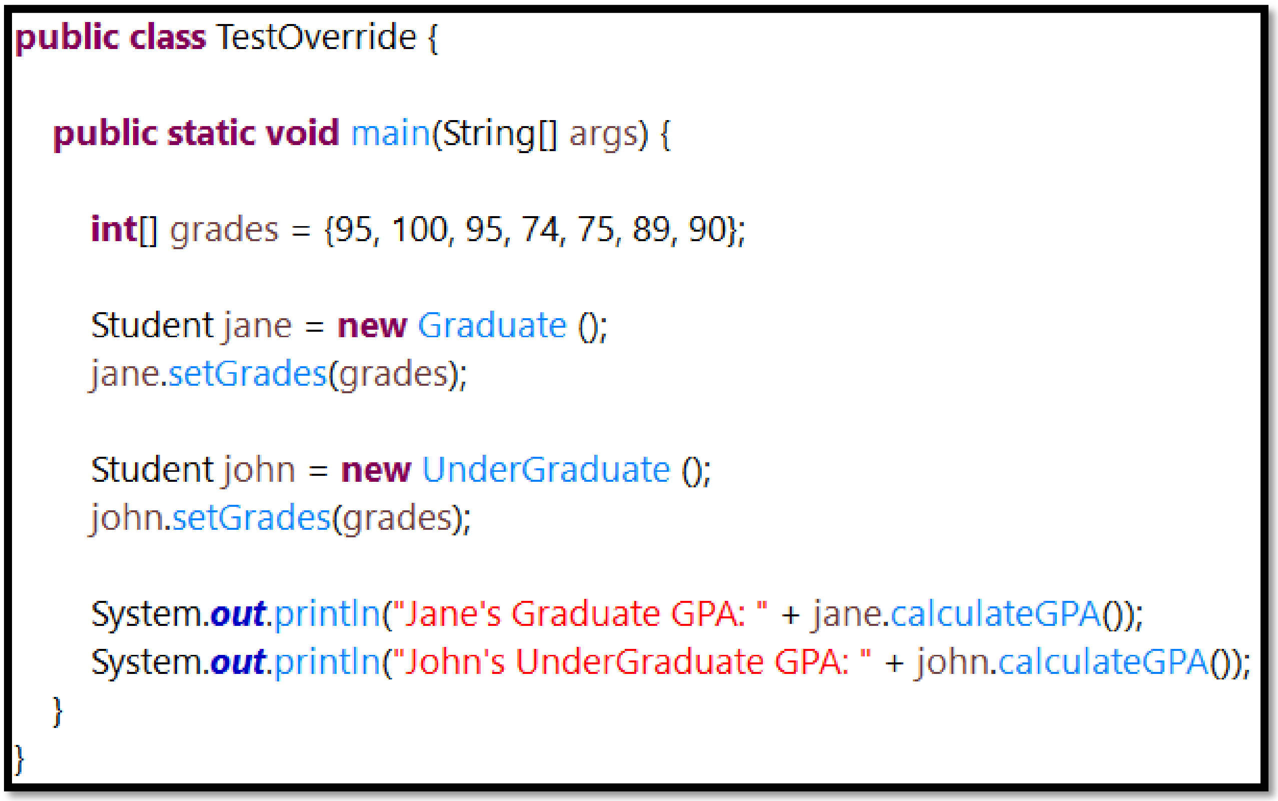
Recall, Graduate has a method to override the Student method. UnderGraduate does not have a method to override Student so it will use the method created in the Student class. Here's the scenario with 2 instances. Jane is a Graduate and John is an UnderGraduate. We are going to imagine both students have 7 grades and the same grades for each class. They have 7 classes and the same grade for each class. The difference is John gets credit for every class but Jane only gets credit for classes 75 and above.

We have a field for minimumGrade equal to 75 and a calculateGPA method. Let's create an array for grades: int[] grades = Let's assign 7 grades. {95, 100, 95, 74, 75, 89, 90}. Bingo, Now our objects.
Student jane = new Graduate()
jane.setGrades(grades)

The next student is John.

Student john = new UnderGraduate ()
john.setGrades(grades)

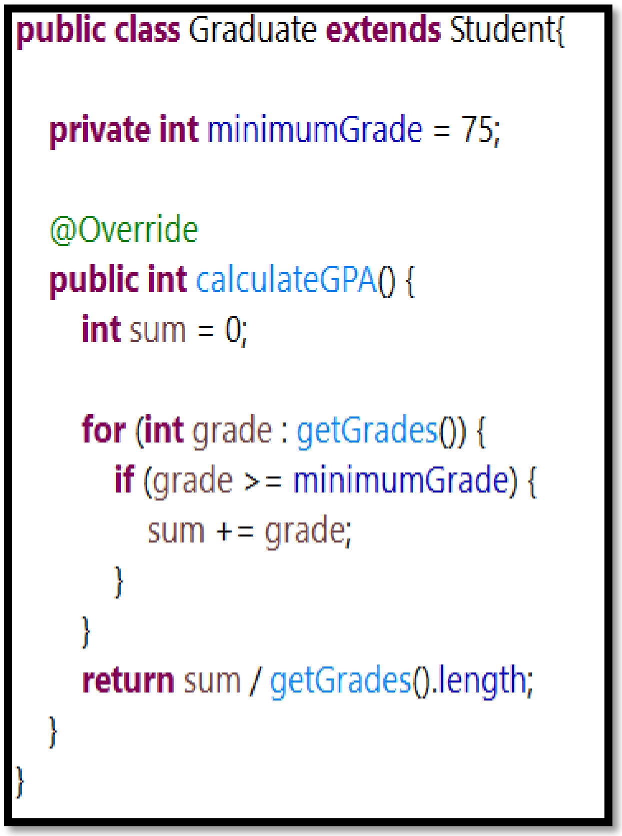
Both students have the same grades but Jane gets credit for 6 of the 7 grades. She does not get credit for the 74. Let's print the grade point average.
sysout("Janes' Graduate GPA: " + jane.calculateGPA())
sysout("John's UnderGraduate GPA: " + john.calculateGPA)



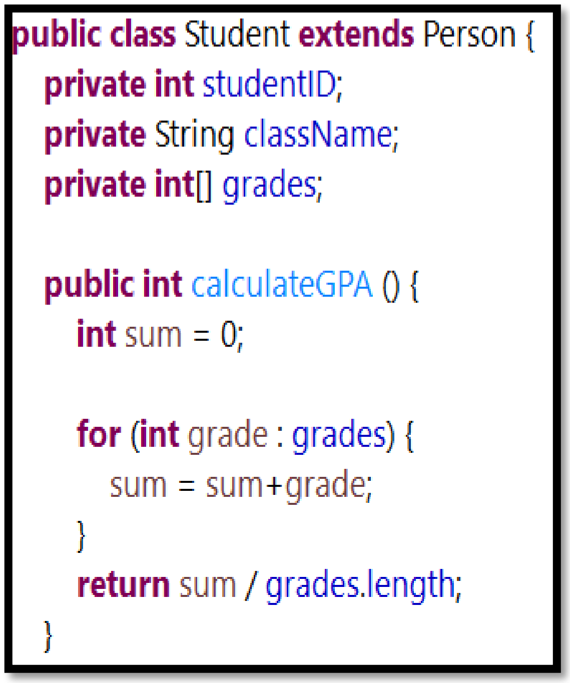
```java
public class TestOverride {

    public static void main(String[] args) {

        int[] grades = {95, 100, 95, 74, 75, 89, 90};

        Student jane = new Graduate ();
        jane.setGrades(grades);

        Student john = new UnderGraduate ();
        john.setGrades(grades);

        System.out.println("Jane's Graduate GPA: " + jane.calculateGPA());
        System.out.println("John's UnderGraduate GPA: " + john.calculateGPA());
    }
}
```

Let's break this down before I run. Here's what's going to happen when I execute. Dynamic Binding will look at the Graduate instance type for Jane then call the override calculateGPA method in the Graduate class. For John, the instance type is UnderGraduate but the UnderGraduate class does not have an override method to calculate the GPA. Therefore, it will use the calculateGPA method from the Student class.

Rex Jones II



```
public class Graduate extends Student{

  private int minimumGrade = 75;

  @Override
  public int calculateGPA() {
    int sum = 0;

    for (int grade : getGrades()) {
      if (grade >= minimumGrade) {
        sum += grade;
      }
    }
    return sum / getGrades().length;
  }
}
```



```
public class Student extends Person {
  private int studentID;
  private String className;
  private int[] grades;

  public int calculateGPA () {
    int sum = 0;

    for (int grade : grades) {
      sum = sum+grade;
    }
    return sum / grades.length;
  }
```

Let's Run. Jane's GPA is 77 while John's GPA is 88. The console shows a different GPA for Jane and John although both students have the same grades. That's what we call Dynamic Binding. We are finished with Polymorphism for Dynamic Binding, Static Binding, Method Overriding, and Method Overloading. In the next session, we will dive into Abstraction.