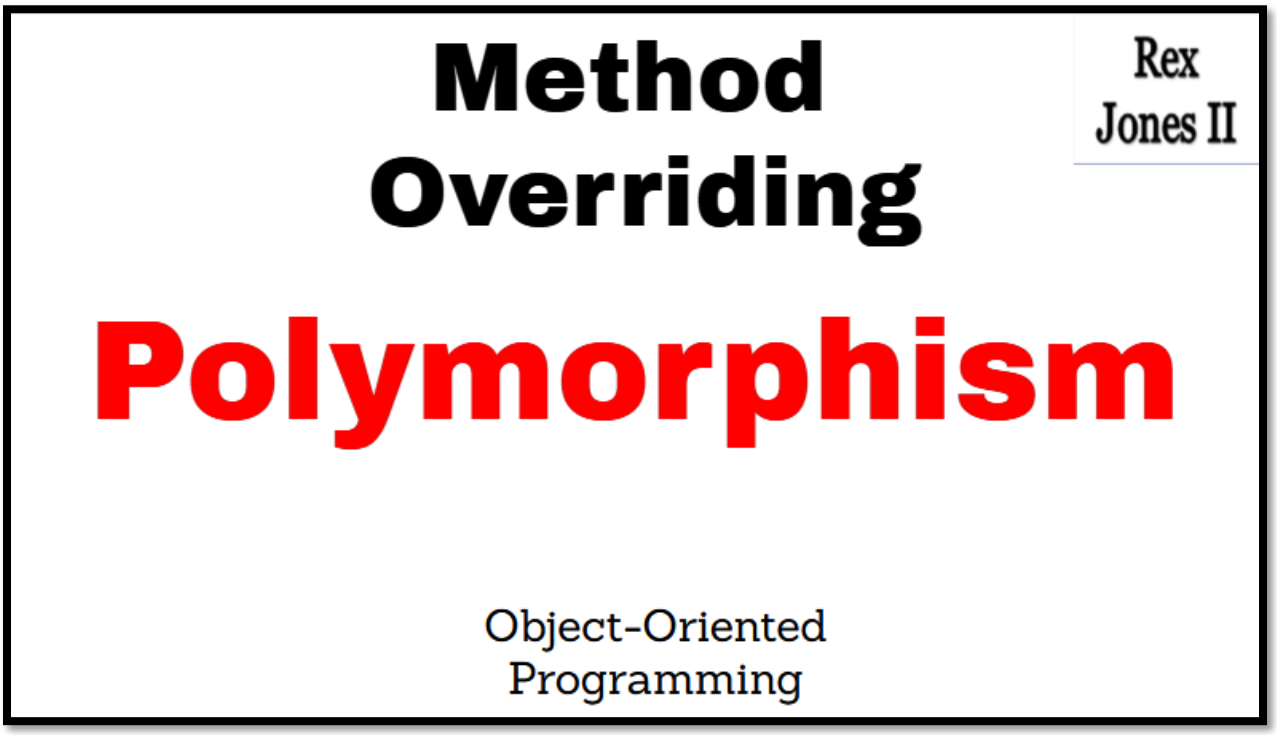
Rex
Jones II

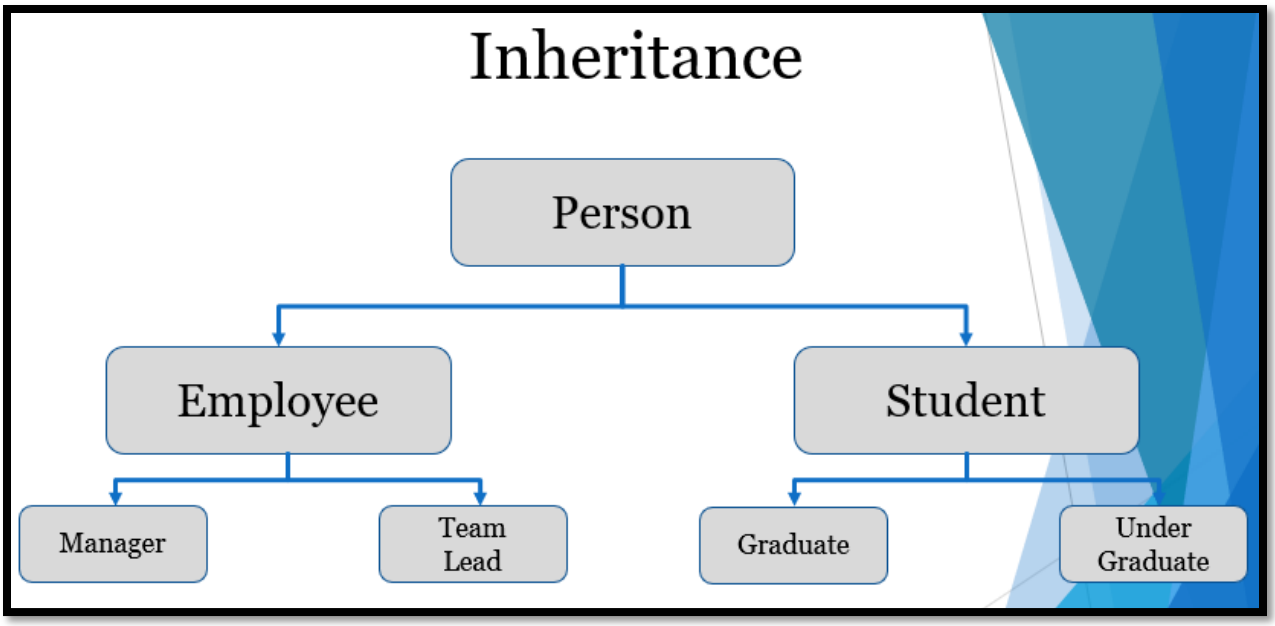
(Transcript) Polymorphism
Method Overriding



Method Overriding allows a superclass to specify a method that will be common to all subclasses. Combining inheritance with method overriding, satisfies the one interface – multiple methods aspect of Polymorphism. A superclass defines the general method that can be used by all of its subclasses. Each subclass is flexible to define its own specialized method while enforcing one consistent interface. Method Overriding only happens when a method in a subclass has the same return type and signature as a method in the superclass.

In our diagram, Student is a child of the Person class. It's also the parent of Graduate and Undergraduate. In real life, Graduate is a student who has a bachelor's degree and pursuing a master's or doctorate degree. An UnderGraduate is a student pursuing a bachelor's or associate degree. Both children classes will automatically inherit the same Student method if the method is public.

Rex Jones II



In this session, we already have our Person class and Student class from the previous Inheritance session. The Student class extends Person. Let me minimize these getters and setters. For starters, I will create a method to calculate the grades for each student: public int calculateGPA ()

We have to create an array to store a list for the grades by writing private int[] grades. This method returns 0. Now, let's calculate the GPA. First, we initialize int sum = 0 then a
for loop (int grade : grades) {
  sum is assigned to sum + the next grade
}
return sum / grades.length. length provides the number of elements in the array. Add getters and setters. Source, Generate Getters and Setters, select grades then OK.

Rex Jones II

```java
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;
    }

    public int[] getGrades() {
        return grades;
    }

    public void setGrades(int[] grades) {
        this.grades = grades;
    }
```

Let's imagine Graduate students only get credit for grades 75 and above. Start with a private int field called minimumGrade = 75. In addition to minimumGrade, we need a specialized method only for the Graduate class. This method will override the method in the Student class. With Inheritance, Graduate extends Student. CTRL + Space. Do you see how the calculateGPA method and these other methods that override a method in Student? We also see some override methods for the Person class.

Select calculateGPA and erase the comment. It's not required but its recommended we use this @Override annotation to prevent an error such as providing a different method name. Now, let's calculate the GPA based on grades 75 and higher.

int sum = 0
for (int grade : call the getGrades()) { method from the Student class
  if (grade >= minimumGrade) {
   sum += grade This is the shortcut syntax for sum = sum + grade

```
 }
}
```

return sum / getGrades().length

```java
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;
   }
}
```

This calculateGPA method is how we implement a specialized method for a child class to override its parent class. Next, we are going to look at the difference between both Java Bindings.