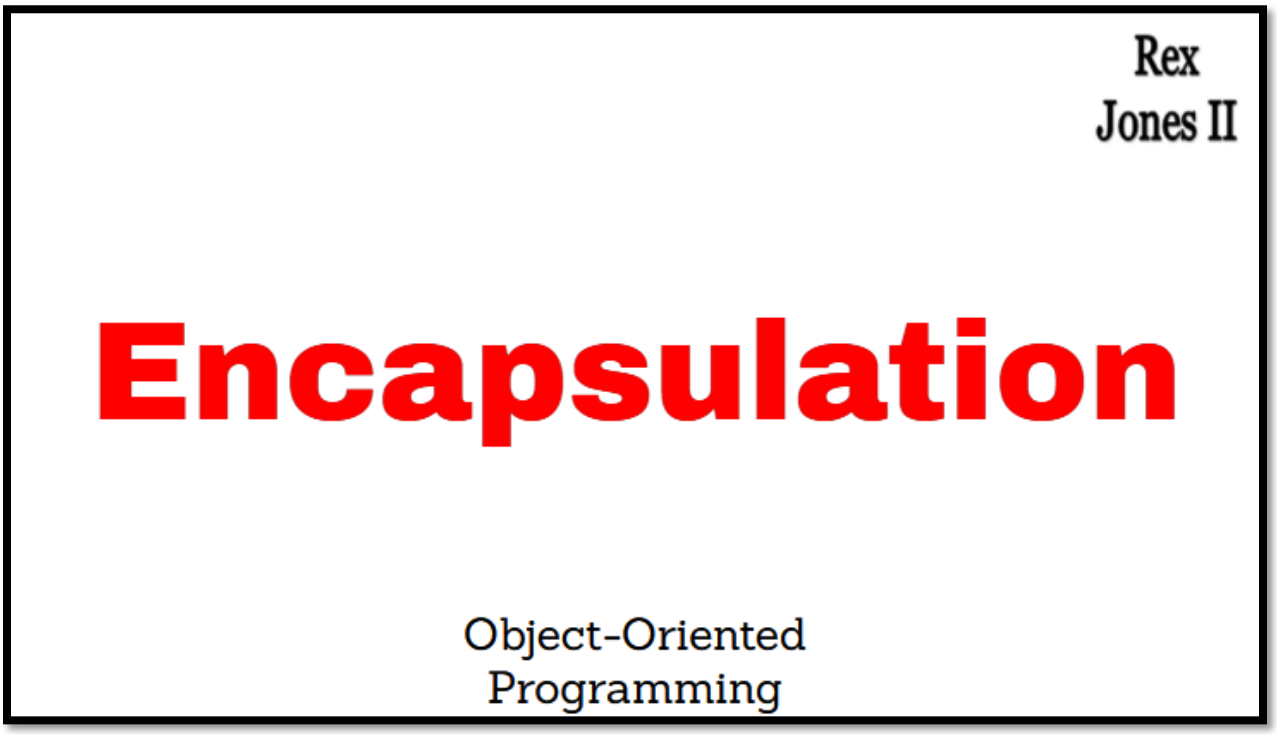
Rex
Jones II

(Transcript) Java Encapsulation (OOP)



Encapsulation is the process of protecting the internal representation of an object. It's known as data hiding because we make the data private so the data cannot be accessed from outside the class.

Hi my name is Rex and I like to share programming and automation knowledge through books, blogs and videos. If you are interested in the content, feel free to connect with me on LinkedIn, YouTube, Twitter, Facebook, and GitHub.

Let's go to Eclipse.

We have a class called Employee with a double data type for salary and bonus. The method calculateTotalPay performs an action on the data by adding salary and bonus then assigning the value to totalPay.

Rex Jones II

```
public class Employee {
   double salary;
   double bonus;

   public void calculateTotalPay () {
      double totalPay = salary + bonus;
      System.out.println("Total Pay = " + totalPay);
   }
```

Let's look at why we need to protect our data. In this class called TestEmployeeObject, Jane's salary should be set to $100,000 and bonus set to $10,000. However, check out what happens when we calculate total pay. Jane made over $1,000,000 with a total pay of $1,010,000. The salary is not correct at $1,000,000. This is a prime example on why we need to protect our data.

A person can make a mistake and enter the wrong data. So, we protect the data using a private access modifier with getter and setter methods. A getter method is used to get and view values while a setter method is used to set or modify a value. Let's say the company's policy is no employee gets paid less than $50,000 and no employee gets paid more than $250,000.

We protect the data by changing salary and bonus to private then Save. Do you see how instantly jane.salary and jane.bonus both have an error? They have an error because they are no longer visible. Next, we create a setter method for salary. Let's start with salary:

We write public void setSalary. The parameter will be (double salary) inside the parenthesis.
if salary is greater than or equal to $50,000 and salary is less than or equal to $250,000
this.salary equals salary
else this.salary = $0. Let's add a print statement: sysout ("Salary Is Incorrect")

```
public void setSalary (double salary) {
   if (salary >= 50000 && salary <= 250000) {
      this.salary = salary;
   }
   else {
      this.salary = 0;
      System.out.println("Salary Is Incorrect");
   }
}
```

The keyword this for this.salary refers to the field private double salary and not the parameter salary in parenthesis. Now, we have a boundary for setting our salary. For bonus, there is no limit so I'm going to create a setter method by writing
public void setBonus (double bonus) as the parameter
this.bonus = bonus

```
public void setBonus (double bonus) {
   this.bonus = bonus;
}
```

In the TestEmployeeObject class, we must call the methods by writing jane.set. The intellisense shows setBonus and setSalary: select setSalary then pass $1,000,000 again as an argument. Also write jane.setBonus then pass $10,000.

Let's Run. We protected salary so the console shows Salary Is Incorrect. Change $1,000,000 to $100,000 and run. Now, salary is correct. The Console shows Total Pay equals $110,000. How can we print the salary and/or bonus from the TestEmployeeObject class?

sysout ("Jane's Salary = " + jane.) We know the data is not available because it's private. That's where the concept of getter methods come into action. It will help us get the data.

We write public. Since the data type is double for salary then the return type is double.
getSalary as the method name no parameters and we are going to return salary }. From the beginning, if you did not want to write each getter and setter method then Eclipse can generate both methods for us. Right click > Select Source > then Generate Getters and Setters then check bonus and click OK. There it is: public double getBonus and return bonus. Now, let's finish our print statements.

jane.getSalary and sysout ("Jane's Bonus = " + jane.getBonus).

```java
public class TestEmployeeObject {

   public static void main(String[] args) {
      Employee jane = new Employee();

      jane.setSalary(100000);
      System.out.println("Jane's Salary = " + jane.getSalary());

      jane.setBonus(10000);
      System.out.println("Jane's Bonus = " + jane.getBonus());

      jane.calculateTotalPay();
   }
}
```

Let me go Save over here. Let's Run. Jane's Salary is $100,000, Bonus is $10,000, and Total Pay is $110,000. Bingo, next we are going to look at Inheritance.