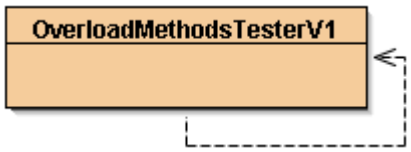
08.06 Lesson Instructions (Part 1)



Open the **`OverloadMethodsTesterV1.java`** file in the Mod08 Lessons folder and examine the source code, **which is contained in one class and one file**.

Three overloaded methods for calculating the area of a rectangle are provided. Notice that the signature of each method is different. There is something else interesting in the print statements you may not have seen before; the methods are called from within the argument of the **`println()`** method (i.e., within the parentheses) and concatenated to a **`String`** literal as indicated below..

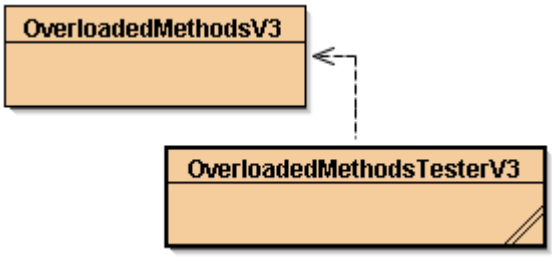
```
System.out.println("One int passed: " + overload.calcRectArea(9));
```

This shortcut is fairly common in Java, but such statements are very prone to syntax errors, so pay close attention when you write statements in this manner.

In the third **`calcRectArea()`** method, why is the return value cast to an **`int`**? Remove the casting and see what happens. What could you change in the method signature, to avoid the problem?

Examine the source code for the **`OverloadMethodsTesterV2.java`** file in the Mod08 Lessons folder and compare it to the first version. In this version, **all of the source code is in one file, but there are two classes**. Separating the implementation of a class from the **`main()`** method is good programming practice because it improves portability.

Once again in version 2, three overloaded methods for calculating the area of a rectangle are provided, but their signatures differ.



Open the **`OverloadMethodsTesterV3`** and the **`OverloadMethodsV3`** java files in the Mod08 Lessons folder and compare the source code to the first and second versions. In this example**, there are two separate classes in two separate files**.

Although the structural organization has changed, there are still three overloaded methods for calculating the area of a rectangle, each with a different signature.

Overloading is a simple strategy for increasing flexibility while maintaining order in object-oriented programs. Make sure you are comfortable with all three techniques illustrated.

For practice, add one pair of overloaded methods from the following list to each of the demo programs.

1. Write two overloaded methods to calculate the perimeter of a rectangle.
2. Write two overloaded methods to calculate the area of a circle.
3. Write two overloaded methods to calculate the circumference of a circle.
4. Write two overloaded methods to calculate the area of a triangle.

Use the existing print statements as models and call the methods in a similar fashion. Try additional parameter lists with different combinations of **`ints`** and **`doubles`**. It will probably be helpful to make a list of the possible combinations for one, two, and three parameters before you start.

What are the possible numeric combinations for a two parameter list?

What are the possible numeric combinations for a three parameter list?

What are your numeric alternatives for a one parameter list?

Overloading is an important OOP concept that applies to both methods and constructors. Be sure you try the suggested modifications in order to master this important technique.