**Unit 2: Using Objects**

**Topic 1 Lab 2: Intro to Objects Part 2**

|  |  |
| --- | --- |
| **Name:** |  |

|  |  |
| --- | --- |
| **More with Rectangles** |  |
| **1.**  Continue using your IntelliJ project from the Do Now (LastNameU2T1Lab2)  **2.** Create a new class in the **src** folder named **RectangleRunner** and give it a main method (no code to paste yet).  **3.** Create a second class in the **src** folder named **Rectangle** and copy/paste [**this code**](https://docs.google.com/document/d/1RU5ULGKzOzO5vwht7KQJ2Fz2tR1GsrebK04tPr6H5zk/edit?usp=sharing) (different class than the last lab). | |
| **4.** Look in the Rectangle class:   1. Which methods have **void** return types? 2. Which methods have ***non*-void** return types? What is the type(s) of their return values? 3. Which methods have *no* parameters? 4. Which methods have a parameter? | a.  b.  c.  d. [check](#_5326lhlsmqc2) |
| **5.** In the RectangleRunner class, create a Rectangle object named rect1. Give it a length and width of 10 and 20. On your object, call the printArea method and the printBoxVolume method with a height of 30. Run the code to see that it works; you should see the following printed output:    **Copy and paste the line(s) of code that you wrote below:** | |
| [check](#_76846bpwwlbn) | |
| **6.**  Add *another* line of code to call the printBoxVolume a second time with a height of 40. Run your code to make sure you see two different volumes printed (6000 and 8000). | |
| **7.** Why do *both* calls to printBoxVolume use the same 10 and 20 for the length and width, but *different* values for height? | [check](#_jbv7x4t2683z) |
| **8.** In the RectangleRunner class, keep the line of code where you create rect1 and delete the rest. Now write some code in the main method that use the two methods that *return values* to print a statement like the following, using a height of 25:    **Use two variables to store the *returned* area and volume!**  **Copy and paste the line(s) of code that you wrote below:** | |
| [check](#_ltixwne18i3n) | |
| **9.** Now modify your code to *remove* the variables that store the returned values and instead call both methods "**in line**" as part of the print statement.  **Copy and paste the line(s) of code that you wrote below:** | |
| [check](#_yxnppwgdigev) | |
| **10.** Here is a line of code that ***doesn't*** compile (disregard that is spans two lines in this document):  System.*out*.println("This rectangle has an area of " + rect1.printArea(); + " and a volume of " + rect1.printBoxVolume(25)); | |
| **11.** Explain why this line of code won't compile (if you need to copy/paste into IntelliJ to inspect the compiler error, feel free to do so!):  **Not sure?** Note the methods being called on the rect1 object… do they *return* values that can be printed, or not? | [check](#_r4h6i0q1e1rd) |
| **12.** In the Rectangle class, here is logic that calculates and returns area:    The other three methods in the Rectangle class *also* perform calculations that involve length \* width, which is redundant. Help **reduce redundancy** in the code by replacing length \* width in the other three Rectangle methods with calls to its own calculateArea method (review slide **19** for an example).  **Test** to make sure you didn't break anything by making a call to each revised method from your main method in RectangleRunner.  **Copy/paste the methods that you modified in Rectangle (you should have modified 3 methods):** | |
| [check](#_r0xsvksuf1bz) | |
| **13.** When calling calculateArea from the RectangleRunner class, we need to call it on an object using **dot notation**, like rect1.calculateArea(); why don't we use dot notation when calling the calculateArea method from other methods *inside* Rectangle? | [check](#_cz5czl1aoa8o) |

**Lab continues on the next page**

|  |
| --- |
| **14.** Delete the code in your RectangleRunner class, then **copy/paste** the following starter code:  import java.util.Scanner;  public class RectangleRunner {  public static void main(String[] args) {  Scanner myScanner = new Scanner(System.in);  System.out.print("Enter rectangle 1 length: ");  int rect1Length = myScanner.nextInt();  System.out.print("Enter rectangle 1 width: ");  int rect1Width = myScanner.nextInt();  System.out.print("Enter rectangle 1 height: ");  double rect1Height = myScanner.nextDouble();  System.out.print("Enter rectangle 2 length: ");  int rect2Length = myScanner.nextInt();  System.out.print("Enter rectangle 2 width: ");  int rect2Width = myScanner.nextInt();  System.out.print("Enter rectangle 2 height: ");  double rect2Height = myScanner.nextDouble();  **// write the rest of your program below**  }  }  Now, write a program in your main method to create two different Rectangle objects and call appropriate methods in order to produce the following printed output based on the test input:    **Copy/paste the code you wrote:** |
| [sample solution in case you need it](#_pfhvfg6weu56) |

|  |
| --- |
| **CHATBOT!** |
| **15.** Create a new **ChatBot** class, then copy/paste [**this code**](https://docs.google.com/document/d/1RVxr2EG3DuGo_-baatxykCJPhDDqOTMOy6qurZotoOE/edit?usp=sharing) into it.  **16.** Study the code for the ChatBot class, and determine how many of each of the  following the class has:   |  |  | | --- | --- | | **How many instance variables?** |  | | **How many constructors?** |  | | **How many methods *don't* return a value?** |  | | **How many methods that *do* return a value?** |  | | **How many methods have at least one parameter?** |  | | **How many methods have *no* parameters?** |  |  [check answers](#_zhfkuehe0nf6) **17.** Create a new **ChatBotRunner** class (a **client** class that will *use* the ChatBot class), write the class and the main method, and in the main method, do the following:   1. Create a ChatBot object using the constructor; store the object in a variable named debbie (or whatever you want to name the variable), and pass in appropriate values that *you choose* as parameters to the constructor (you will need to look at the constructor to know how many values to pass in, and of what type). 2. Write a program of your choosing that uses *each* of the 6 ChatBot methods at least once.  * For the methods that have *non*-void return values, you can decide whether to store the return values in variables or call the methods "in line"; either way, you should include printed output that displays the returned values in some way.   **6. Copy/paste the code you wrote in your ChatBotRunner class below that calls each method at least once:** |
| [sample code](#_qf06m5cj08) |
| **Insert a screenshot of the printed output:** |
| **Freestyle!**  **7.** Add **two** new methods of your choosing to the ChatBot class:   * one method that is a **void** method (returns no value) * one that *returns* some value (i.e. a non-void method)   **8.** Add a comment above each new method in the ChatBot class that explains what it does.  **9.**  Write some code in your ChatBotRunner class to call each new method, and for the non-void method, do something with the returned value. |
| **Copy/paste the new methods you added to ChatBot:** |
| **Copy/paste the code you wrote in ChatBotRunner to call your two new methods:** |

###### 

**Done!**

Submit in Google Classroom:

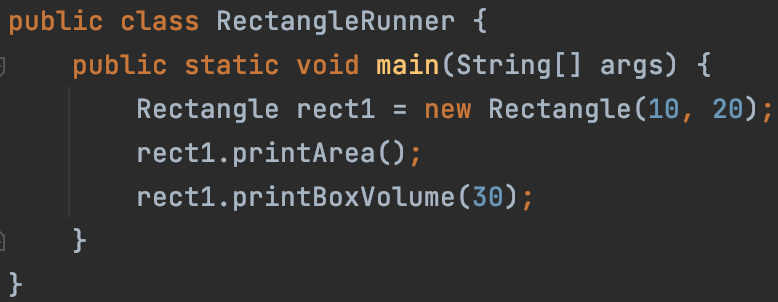


### Answer 4 ([back](#_8p99pn4wlai2))

|  |  |
| --- | --- |
| **4.** Look in the Rectangle class:   1. Which methods have **void** return types? 2. Which methods have ***non*-void** return types? What is the type(s) of their return values? 3. Which methods have *no* parameters? 4. Which methods have a parameter? | a. printArea and printBoxVolume  b. calculateArea and calculateBoxVolume  c. calculateArea and printArea  d. calculateBoxVolume and printBoxVolume |



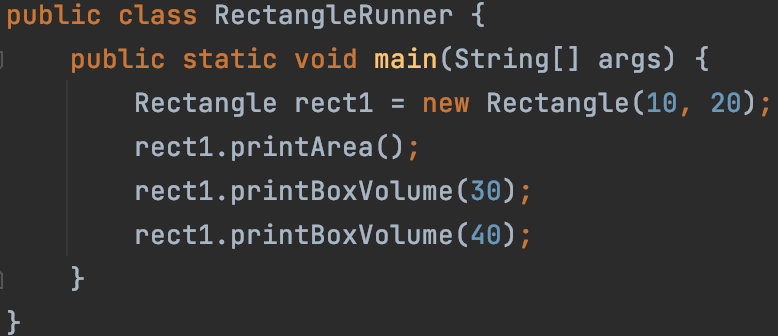
### Answer 5 ([back](#_9k5hgupa8uk2))



### Answer 7 ([back](#_sa3na0aw7xug))

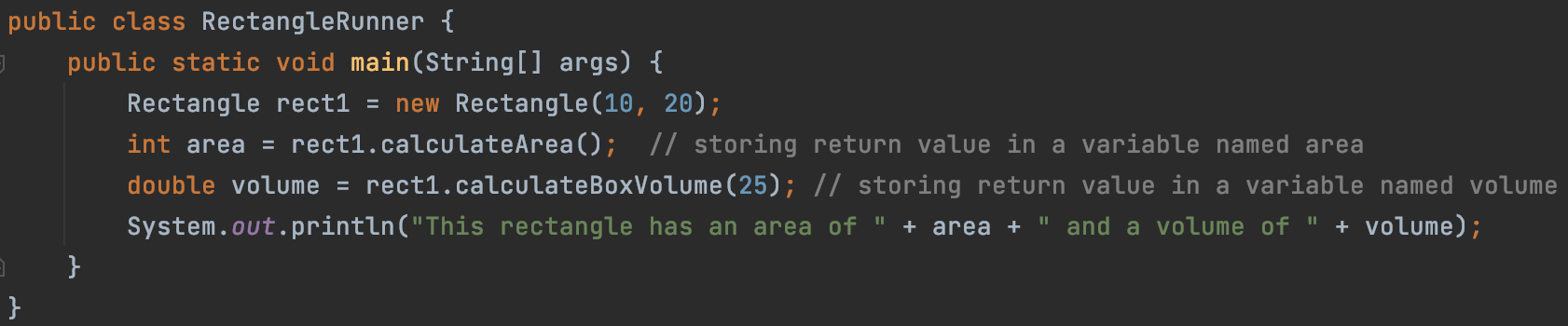
|  |  |
| --- | --- |
| **7.** Why do *both* calls to printBoxVolume use the same 10 and 20 for the length and width, but *different* values for height? | Because both are being called on the rect1 object, which is a Rectangle initialized with length 10 and width of 20, so both of those values are used in each printBoxVolume method call. The *height* is different though because that value is being passed by the client as a parameter. |

Your code should look like:



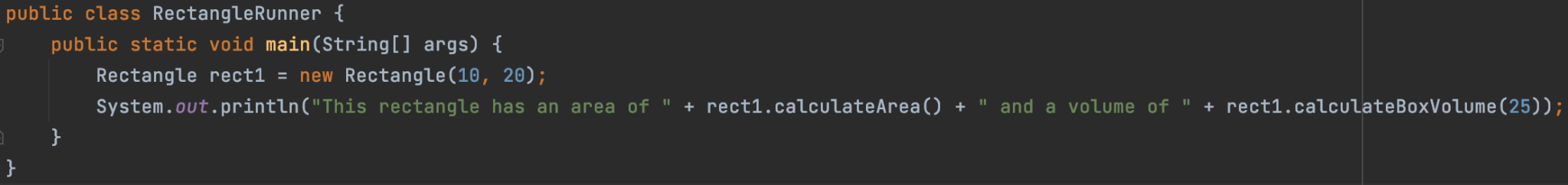
### Answer 8 ([back](#_nax0b99ucqdy))

Storing the **returned** values of calculateArea and calculateBoxVolume method in variables:



### Answer 9 ([back](#_ax1xw9fbb3d))

Calling the two methods that return values "**in line**" as part of a print statement:



### Answer 11 ([back](#_7kk1kt2g236))

You **can't** call *void* methods **in line**; printArea and printBoxVolume are both methods that return *no* values (they have *void* return types), and so there is no value that can be printed in a string!

This compiler error message in IntelliJ…



…means you *can't* use "+" to concentrate a String with a void value

### Answer ([back](#_3l6atzkrh3ys))

You should have made three replacements, outlined below:



### Answer ([back](#_jfsxpiesrawz))

|  |  |
| --- | --- |
| **13.** When calling calculateArea from the RectangleRunner class, we need to call it on an object using **dot notation**, like rect1.calculateArea(); why don't we use dot notation when calling the calculateArea method from other methods *inside* Rectangle? | Because *outside* the Rectangle class, like in the RectangleRunner client class, we need to create a Rectangle object *first* and use that object to call the method.  But *inside* the Rectangle class, one method can call another method without first creating an object -- this is because we are inside the *class definition* itself, and methods can call each other freely.  **We will talk more about this in Unit 5 when we write our own classes!** |

### Sample solution ([back](#_rpp46yr1g2yr))



### Answers ([back](#_bvz32olsua22))

|  |  |
| --- | --- |
| **How many instance variables?** | **2** |
| **How many constructors?** | **1** |
| **How many methods *don't* return a value?** | **3** |
| **How many methods that *do* return a value?** | **3** |
| **How many methods have at least one parameter?** | **4** |
| **How many methods have *no* parameters?** | **2** |

**DETAILS**

|  |  |
| --- | --- |
| **How many instance variables?** | **2: name** and **number** |
| **How many constructors?** | **1:** |
| **How many methods *don't* return a value?** | **3**: the methods with void (and no return statement) |
| **How many methods that *do* return a value?** | **3**: all the methods with a return type *other than void* (and have a return statement): |
| **How many methods have at least one parameter?** | **4**: |
| **How many methods have *no* parameters?** | **2:** |

### Sample code ([back](#_8nqtwgdpgll0))

Sample code showing the creation of a ChatBot object, then calling all of its 6 various methods.



**Printed output (colors match the code above that produced the output)**

