

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](#) (different class than the last lab).

4. Look in the `Rectangle` class:
 - a. Which methods have **void** return types?
 - b. Which methods have **non-void** return types? What is the type(s) of their return values?
 - c. Which methods have *no* parameters?
 - d. Which methods have a parameter?

- a.
- b.
- c.
- d.

[check](#)

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:

```
My area is 200
The volume is: 6000.0
```

Copy and paste the line(s) of code that you wrote below:

[check](#)

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](#)

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:

```
This rectangle has an area of 200 and a volume of 5000.0
```

Use two variables to store the *returned* area and volume!

Copy and paste the line(s) of code that you wrote below:

[check](#)

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](#)

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](#)

12. In the `Rectangle` class, here is logic that calculates and returns area:

```
public int calculateArea() {  
    return length * width;  
}
```

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](#)

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](#)

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:

```
Enter rectangle 1 length: 5  
Enter rectangle 1 width: 10  
Enter rectangle 1 height: 12.5  
Enter rectangle 2 length: 6  
Enter rectangle 2 width: 8  
Enter rectangle 2 height: 15.75  
Rectangle 1's Area: 50, Box Volume: 625.0  
Rectangle 2's Area: 48, Box Volume: 756.0
```

Copy/paste the code you wrote:

[*sample solution in case you need it*](#)

CHATBOT!

15. Create a new **ChatBot** class, then copy/paste [this code](#) 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 <i>don't</i> return a value?	
How many methods that <i>do</i> return a value?	
How many methods have at least one parameter?	
How many methods have <i>no</i> parameters?	

[check answers](#)

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:

- A. 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).
- B. 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](#)

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:

Turn in

4. Look in the Rectangle class:

- a. Which methods have **void** return types?
- b. Which methods have **non-void** return types?
What is the type(s) of their return values?
- c. Which methods have *no* parameters?
- d. Which methods have a parameter?

- a. **printArea** and **printBoxVolume**
- b. **calculateArea** and **calculateBoxVolume**
- c. **calculateArea** and **printArea**
- d. **calculateBoxVolume** and **printBoxVolume**

```
// method that calculates and returns area
public int calculateArea() {
    return length * width;
}

// method that calculates and prints area
public void printArea() {
    int area = length * width;
    System.out.println("My area is " + area);
}

// method that calculates and returns volume
// of a box with length, width, and height
public double calculateBoxVolume(double height) {
    return length * width * height;
}

// method that calculates and prints volume
// of a box with length, width, and height
public void printBoxVolume(double height) {
    double volume = length * width * height;
    System.out.println("The volume is: " + volume);
}
```

Answer 5 ([back](#))

```
public class RectangleRunner {  
    public static void main(String[] args) {  
        Rectangle rect1 = new Rectangle(10, 20);  
        rect1.printArea();  
        rect1.printBoxVolume(30);  
    }  
}
```


Answer 7 ([back](#))

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:

```
public class RectangleRunner {  
    public static void main(String[] args) {  
        Rectangle rect1 = new Rectangle(10, 20);  
        rect1.printArea();  
        rect1.printBoxVolume(30);  
        rect1.printBoxVolume(40);  
    }  
}
```

Answer 8 ([back](#))

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

```
public class RectangleRunner {  
    public static void main(String[] args) {  
        Rectangle rect1 = new Rectangle(10, 20);  
        int area = rect1.calculateArea(); // storing return value in a variable named area  
        double volume = rect1.calculateBoxVolume(25); // storing return value in a variable named volume  
        System.out.println("This rectangle has an area of " + area + " and a volume of " + volume);  
    }  
}
```

Answer 9 ([back](#))

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

```
public class RectangleRunner {  
    public static void main(String[] args) {  
        Rectangle rect1 = new Rectangle(10, 20);  
        System.out.println("This rectangle has an area of " + rect1.calculateArea() + " and a volume of " + rect1.calculateBoxVolume(25));  
    }  
}
```

Answer 11 ([back](#))

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...

Operator '+' cannot be applied to 'java.lang.String', 'void'

...means you *can't* use "+" to concatenate a String with a void value

Answer ([back](#))

You should have made three replacements, outlined below:

```
// method that calculates and prints area
public void printArea() {
    int area = calculateArea();
    System.out.println("My area is " + area);
}

// method that calculates and returns volume
// of a box with length, width, and height
public double calculateBoxVolume(double height) {
    return calculateArea() * height;
}

// method that calculates and prints volume
// of a box with length, width, and height
public void printBoxVolume(double height) {
    double volume = calculateArea() * height;
    System.out.println("The volume is: " + volume);
}
```

Answer ([back](#))

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](#))

```
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();

        // creating two Rectangle objects
        Rectangle rect1 = new Rectangle(rect1Length, rect1Width);
        Rectangle rect2 = new Rectangle(rect2Length, rect2Width);

        // obtaining the area and volume
        int area1 = rect1.calculateArea();
        int area2 = rect2.calculateArea();
        double volume1 = rect1.calculateBoxVolume(rect1Height);
        double volume2 = rect2.calculateBoxVolume(rect2Height);

        // printing the required information
        System.out.println("Rectangle 1's Area: " + area1 + ", Box Volume: " + volume1);
        System.out.println("Rectangle 2's Area: " + area2 + ", Box Volume: " + volume2);
    }
}
```

Answers ([back](#))

How many instance variables?	2
How many constructors?	1
How many methods <i>don't</i> return a value?	3
How many methods that <i>do</i> return a value?	3
How many methods have at least one parameter?	4
How many methods have <i>no</i> parameters?	2

DETAILS

How many instance variables?	<p>2: name and number</p> <pre>// instance variables private String name; private int number;</pre>
How many constructors?	<p>1:</p> <pre>// constructor public ChatBot(String chatBotName, int faveNum) { name = chatBotName; number = faveNum; }</pre>
How many methods <i>don't</i> return a value?	<p>3: the methods with <code>void</code> (and no return statement)</p> <pre>public void greeting(String yourName) public void weather() public void favoriteNumber(int yourNumber)</pre>
How many methods that <i>do</i> return a value?	<p>3: all the methods with a return type <i>other than void</i> (and have a <code>return</code> statement):</p> <pre>public double convertFeetToMeters(int numFeet) public int addNumbers(int num1, int num2, int num3) public String goodbye()</pre>
How many methods have at	<p>4:</p>

least one parameter?	<pre>public void greeting(String yourName)</pre> <pre>public double convertFeetToMeters(int numFeet)</pre> <pre>public void favoriteNumber(int yourNumber)</pre> <pre>public int addNumbers(int num1, int num2, int num3)</pre>
How many methods have <i>no</i> parameters?	2: <pre>public void weather()</pre> <pre>public String goodbye()</pre>

Sample code ([back](#))

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

```
public class ChatBotRunner {  
    public static void main(String[] args) {  
        ChatBot debbie = new ChatBot("Debbie", 10);  
  
        // calling the void methods  
        debbie.greeting("Mr. Miller");  
        debbie.favoriteNumber(15);  
        debbie.weather();  
  
        // calling the non-void methods  
        double meters = debbie.convertFeetToMeters(20);  
        System.out.println("There are " + meters + " in 20 feet");  
  
        int sum = debbie.addNumbers(13, 16, 24);  
        System.out.println("The sum is " + sum);  
  
        String message = debbie.goodbye();  
        System.out.println(message);  
    }  
}
```

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

Hello, Mr. Miller my name is Debbie
and I am a chat bot! How are you today?

My favorite number is 10
That is 5 away from your number!

I actually don't know much about the weather! Ha ha!
But I know it's warm and dry inside a computer! Ha ha!

There are 6.096 in 20 feet

The sum is 53

It was nice talking with you! Have a great day! Sincerely, Debbie