Unit 2: Using Objects

Topic 1: Intro to Objects

Name:	

Rectangles!	
0. Create a new IntelliJ project named LASTNAMEU2T1Lab1 (e.g. "KaufmanU2T1Lab1") and login to GitHub on IntelliJ. **1.** Create a new class in the **src** folder named **`RectangleRunner`** and copy/paste **this code**. **2.** Create a second class in the **src** folder named **`Rectangle`** and copy/paste **this code**.	
3. The `RectangleRunner` class has a `main` method but the `Rectangle` class does not; why is that? Note how there are no ▶▶ on the left side of the `Rectangle` class!	Because RectangleRunner is intended to be executed (run), but the Rectangle is not (is serves as a blueprint for Rectangle objects to be used by other classes)
4. Find the "constructor" method in the `Rectangle` class at **line 10**. **A.** What do you notice about how it's named? **B.** What do you think it does?	A. The constructor has the same name as the class itself: `public Rectangle(int len, int wid)` B. It is a special method that gets called (using the new keyword) to create new Rectangle objects

5. Jump back to the `RectangleRunner` class.
 	
A. What keyword (in orange) is used to create two different `Rectangle` **objects** on lines 5 and 9? Where have we used this keyword before?
 	
B. This code: `new Rectangle(5, 6)` calls the *constructor* method of the `Rectangle` class, passing 5 and 6 as "actual parameters" into the method. Look at the constructor again in the `Rectangle` class; what does the 5 represent? What does the 6 represent? How can you tell which is which?
	A. **new**
	
	B. In this line, the 5 and 6 ("actual parameters")…
	`new Rectangle(5, 6);`
	
	…get passed into the constructor:
	`public Rectangle(int len, int wid)`
	
	in the order listed, and so the 5 is the length and 6 is the width.

6. Execute (run) the `RectangleRunner` class and view its output!	

7. Lines 6 and 10 of `RectangleRunner` are examples of "*calling a method*" on a particular `Rectangle` **object**. Look at the output, then go to the `Rectangle` class and find the `printArea()`	by multiplying the two instance variables, length and width:
	`// method that calculates and prints area`
	`public void printArea() {`

method. How does the method calculate the area?

```
    int area = length * width;
    System.out.println("My area is " +
area);
}
```

8. In the `RectangleRunner` class, create another `Rectangle` object named `rect3`. Give it a length and width of your choosing. Then call the `printArea()` method on your new object. Run the code to test that it works!

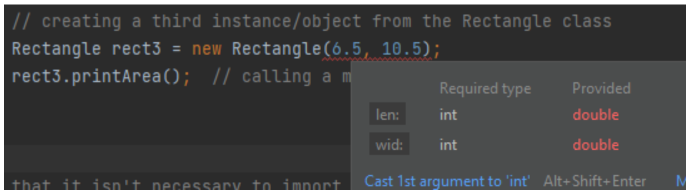
Copy and paste the line(s) of code that you wrote below:

```
// creating a third instance/object from the Rectangle class
Rectangle rect3 = new Rectangle(4, 15);
rect3.printArea();  // calling a method on the object
```

9. Try creating another `Rectangle` object, `rect4`, with a width of 6.5 and length of 10.5. What happens and why do you think that is? Hover your mouse over the red squiggly in IntelliJ and see if you can figure out the error!

Is it a syntax/compiler error, or a runtime error?

Once you have the problem figured out, go ahead and delete `rect4`.



```
// creating a third instance/object from the Rectangle class
Rectangle rect3 = new Rectangle(6.5, 10.5);
rect3.printArea();  // calling a m
that it isn't necessary to import
```

	Required type	Provided
len:	int	double
wid:	int	double

Cast 1st argument to 'int' Alt+Shift+Enter M

It gives a syntax/compiler error (since red squiggy!) because the Rectangle constructor is looking for two int parameters, and you are trying to pass in doubles

10. Go into the `Rectangle` class and try writing a new method, `printPerimeter()`, which calculates and prints the perimeter of the `Rectangle`, similar to area. Start the method with "`public void`" like `printArea()` -- we will talk about what these words mean soon!

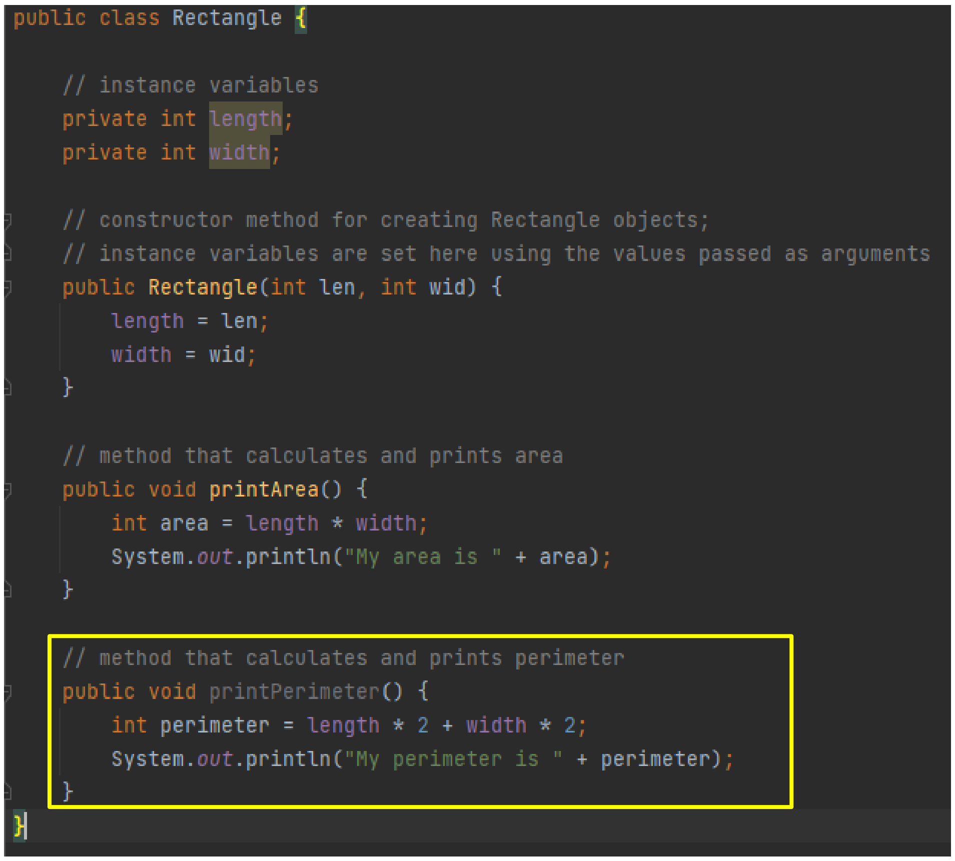
Then test your method by going to `RectangleRunner` and adding code to call the new method on each of the `Rectangle` objects (`rect1, rect2, rect3`). Confirm the output is what you expect!

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

ADDED TO THE RECTANGLE CLASS under the printArea method (or above it, it doesn't matter):

```
// method that calculates and prints perimeter
public void printPerimeter() {
    int perimeter = length * 2 + width * 2;
    System.out.println("My perimeter is " + perimeter);
}
```

Your Rectangle class should look like this:

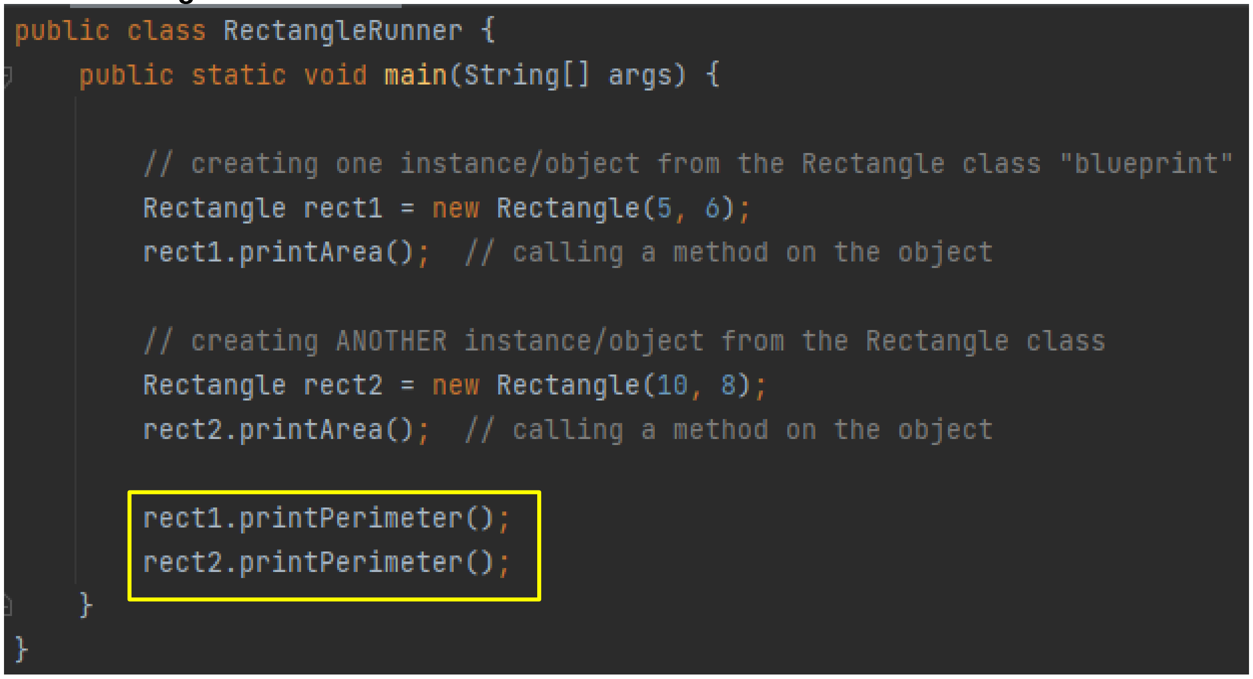


```java
public class Rectangle {

    // instance variables
    private int length;
    private int width;

    // constructor method for creating Rectangle objects;
    // instance variables are set here using the values passed as arguments
    public Rectangle(int len, int wid) {
        length = len;
        width = wid;
    }

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

    // method that calculates and prints perimeter
    public void printPerimeter() {
        int perimeter = length * 2 + width * 2;
        System.out.println("My perimeter is " + perimeter);
    }
}
```

ADDED TO THE RECTANGLERUNNER CLASS (this code "calls" the new method on each object):

```java
rect1.printPerimeter();
rect2.printPerimeter();
```

Your RectangleRunner class should look like this:



```java
public class RectangleRunner {
    public static void main(String[] args) {

        // creating one instance/object from the Rectangle class "blueprint"
        Rectangle rect1 = new Rectangle(5, 6);
        rect1.printArea();  // calling a method on the object

        // creating ANOTHER instance/object from the Rectangle class
        Rectangle rect2 = new Rectangle(10, 8);
        rect2.printArea();  // calling a method on the object

        rect1.printPerimeter();
        rect2.printPerimeter();
    }
}
```

11. If you had to guess, what do you think `public` means? Not sure? Try making the method private	public means you can call the method on an object; if you make it private, it won't work and IntelliJ says

instead… what happens? What about `void`?	"has private access" and code won't compile void means the method returns no value (we will learn about this soon!)

Cats!

12. Create two new classes in the src folder: **`CatRunner`** and **`Cat`.**

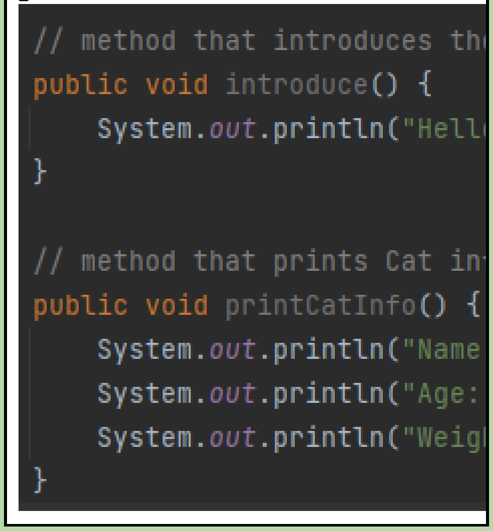
Give the `CatRunner` class a `main` method, and for the `Cat` class, copy/paste **this code**.

Notice that you now have two classes in your project, `RectangleRunner` and `CatRunner`, which both have main methods, so both of them are executable:

13.
A. In your `CatRunner`'s `main` method, write code to create a `Cat` object named `cat1`. You will need to look at the constructor in the `Cat` class to determine what values should get passed in as parameters! Choose the parameter values for your cat.

B. Next, look into the `Cat` class and find two different methods. What are the two methods named?

`introduce()` and `printInfo()`



```
// method that introduces th
public void introduce() {
    System.out.println("Hell
}

// method that prints Cat in
public void printCatInfo() {
    System.out.println("Name
    System.out.println("Age:
    System.out.println("Weigh
}
```

C. Pick one of the methods then write code to call that method on `cat1`. Run the code to see the output.

D. Write code to call the *other* method on `cat1`, then run the code to see the output.

E. Write code to create a second `Cat` object, `cat2` (again choosing your own parameter values), and call both methods on `cat2`.

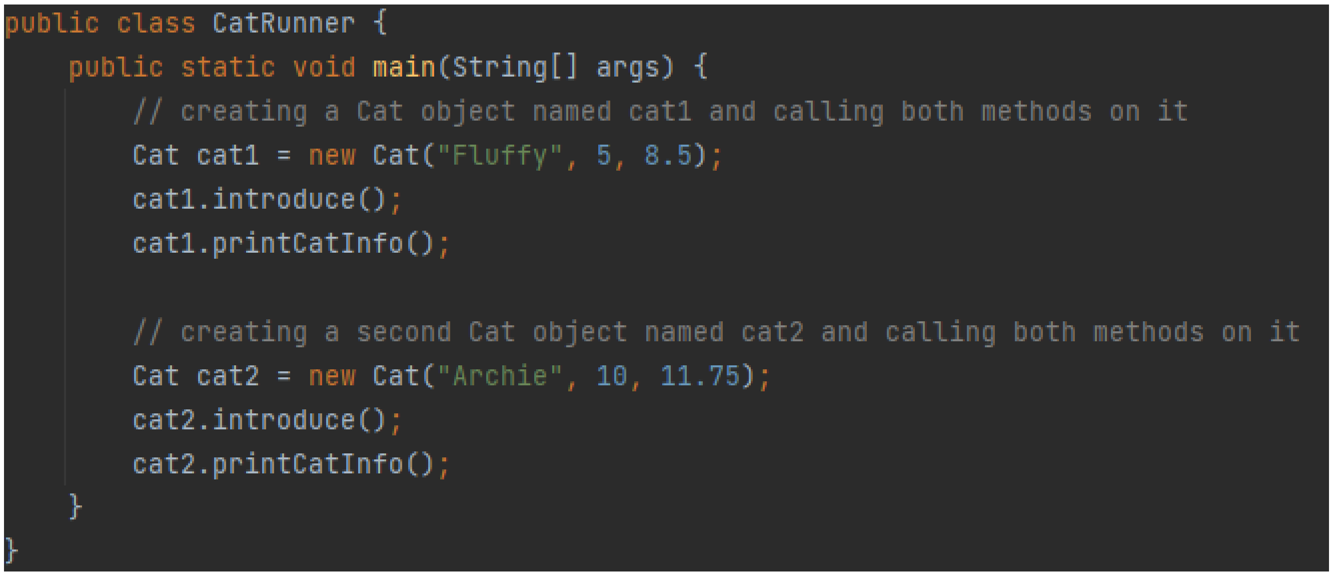
Copy and paste the line(s) of code that you wrote below:

ADDED TO THE CATRUNNER CLASS (inside the main method):

```
// creating a Cat object named cat1 and calling both methods on it
Cat cat1 = new Cat("Fluffy", 5, 8.5);
```

```
cat1.introduce();
cat1.printCatInfo();

// creating a second Cat object named cat2 and calling both methods on it
Cat cat2 = new Cat("Archie", 10, 11.75);
cat2.introduce();
cat2.printCatInfo();
```

Your CatRunner class should look like:



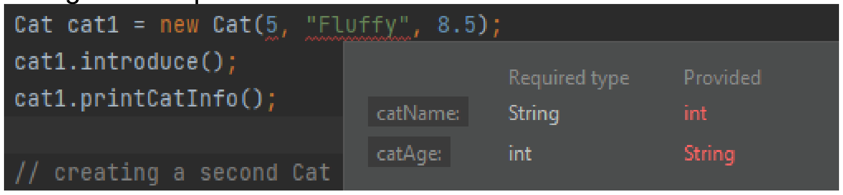
```
public class CatRunner {
    public static void main(String[] args) {
        // creating a Cat object named cat1 and calling both methods on it
        Cat cat1 = new Cat("Fluffy", 5, 8.5);
        cat1.introduce();
        cat1.printCatInfo();

        // creating a second Cat object named cat2 and calling both methods on it
        Cat cat2 = new Cat("Archie", 10, 11.75);
        cat2.introduce();
        cat2.printCatInfo();
    }
}
```

14. What happens if you switch up the order of the parameters when you create your `Cat` objects? In other words, determine if both of these are "valid" calls to the `Cat` constructor (i.e. will both compile):

```
new Cat(5, "Fluffy", 8.5)
new Cat("Fluffy", 5, 8.5)
```

What happens when you mix them up and why?

You get a compiler error:



```
Cat cat1 = new Cat(5, "Fluffy", 8.5);
cat1.introduce();
cat1.printCatInfo();

// creating a second Cat
```

	Required type	Provided
catName:	String	int
catAge:	int	String

This is because the parameter types no longer match up to the order of the parameters in the Cat constructor:

```
public Cat(String catName, int catAge, double catWeight)
```

The constructor expects a **String, int, double** -- *in that order!*

15. Challenge! In the `Cat` class, modify the `introduce()` method so that it prints: "Hello my name is ___ and I am a younger cat" if `age` is less than 7, and "Hello my name is ___ and I am an older cat" if `age` is 7 or older.

Run your `CatRunner` again to test! You should see the updated output for both cats. Make sure one of the cat's ages is less than 7 and one is greater than 7, just to test the method!

Copy and paste your updated method below:

```
// method that introduces the Cat
public void introduce() {
    if (age < 7) {
        System.out.println("Hello my name is " + name + " and I am a younger cat");
    } else {
        System.out.println("Hello my name is " + name + " and I am an older cat");
    }
}
```

OR

```
// method that introduces the Cat
public void introduce() {
    if (age < 7) {
        System.out.println("Hello my name is " + name + " and I am a younger cat");
    }
    if (age >= 7) {
        System.out.println("Hello my name is " + name + " and I am an older cat");
    }
}
```