In this lab, you will enhance the Body Mass Index (BMI) calculator you created in a previous video that demonstrated utility classes. You will gain practical experience by creating and using multiple classes with different methods and invoking them in various contexts. Building on the fundamental features of the original BMI calculator, which included the ability to calculate BMI in both the imperial and metric systems, you will continue to focus on these core capabilities.

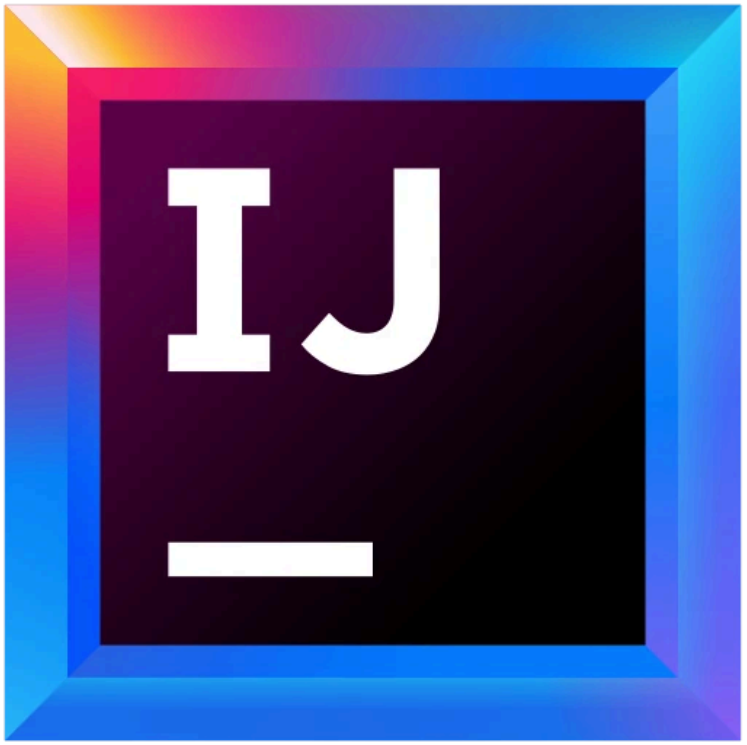
First, you'll obtain user input for their preferred BMI system, either imperial or metric. Next, you'll gather the user's weight and height. Using the chosen system, you will then calculate the BMI value accordingly. Following this, you will determine the BMI category based on the calculated value. Finally, you will display both the calculated BMI value and the corresponding category to the user. This exercise will reinforce your understanding of class interaction and method invocation while maintaining the essential functionality of the BMI calculator.

Goal

Develop a user-friendly application that calculates and categorizes BMI based on user input for weight, height, and preferred measurement system.

🖥️ Note: When encountering this icon, it’s time to get into your IDE and start coding!

In your lab environment, open IntelliJ by double-clicking on the icon.



Before you begin

You have received partial source code in *BMICalculator.java*, *UserInput.java*, and *Main.java*.

The **`UserInput.java`** class has the following methods already implemented for you:

- **`obtainBMISystem`**: Contains a **`Scanner`** that captures user input for the preferred BMI system, validates the user input, and returns it.
- **`obtainWeight`**: Uses a **`Scanner`** to capture user input for weight in pounds or kilograms based on the entered BMI system. It also features a **`switch`** statement that determines whether the weight should be obtained in pounds or kilograms based on the BMI system. Additionally, it includes a while loop that controls obtaining user input for weight until a valid value is entered.

The **`Main.java`** class has the following already implemented for you:

- Code that declares and initializes some required variables (primitive and reference).
- A partially written **`switch`** statement that obtains user input.
- Code to display outputs to the user.

Complete the following tasks to implement each method and attain your objectives. Good luck!

Open *BMICalculator.java* and declare the following four attributes/properties in the **`BMICalculator`** class to represent: weight in pounds, height in inches, weight in kilos, and height in meters. There should be a **`no-argument`** constructor that initializes the four properties to some initial values.

🖥️ It's time to get coding!

- 🖥️ TODO 1: Insert the relevant code into the **`BMICalculator`** class.

Complete the **`calculateBmiImperial`** method. The method takes two arguments of double type, weight, and height, calculates the BMI using the imperial system, and returns the result as a double.

The equation:

$$(703 * \text{weightInPounds})/(\text{heightInInches}*\text{heightInInches})$$

$$(703 * \textit{weightInPounds})/(\textit{heightInInches}*\textit{heightInInches})$$

left parenthesis, 703, space, times, space, w, e, i, g, h, t, I, n, P, o, u, n, d, s, right parenthesis, slash, left parenthesis, h, e, i, g, h, t, I, n, I, n, c, h, e, s, times, h, e, i, g, h, t, I, n, I, n, c, h, e, s, right parenthesis

Tip: Revisit the method you encountered in the *Grouping methods* video demonstrating utility classes enabling this functionality.

🖥️ It's time to get coding!

- 🖥️ TODO 2: Insert the relevant code into **`calculateBmiImperial()`** of the **`BMICalculator`** class.

Pass the **`calculateBmiMetric`** method, which takes in weight and height to calculate and return the BMI value in the metric system. Before calculating within the method, set the attributes with the values passed in as method arguments.

The equation:

$$\text{weightInKilos}/(\text{heightInMeters}*\text{heightInMeters})$$

$$\textit{weightInKilos}/(\textit{heightInMeters}*\textit{heightInMeters})$$

w, e, i, g, h, t, I, n, K, i, l, o, s, slash, left parenthesis, h, e, i, g, h, t, I, n, M, e, t, e, r, s, times, h, e, i, g, h, t, I, n, M, e, t, e, r, s, right parenthesis

Tip: As before, revisit the method you encountered in the *Grouping methods* video demonstrating utility classes enabling this functionality.

🖥️ It's time to get coding!

- 🖥️ TODO 3: Insert the relevant code into the **`calculateBmiMetric()`** method of the **`BMICalculator`** class.

Pass the **`getBMICategory`** method, which takes in the BMI value and returns the BMI category based on the following parameters:

- **`bmi < 18.5 - Underweight`**
- **`bmi < 25 - Normal weight`**
- **`bmi < 30 - Overweight`**
- **`Else - Obese`**

Tip: A **`switch`** statement would not work here as it cannot be used with double values.

🖥️ It's time to get coding!

- 🖥️ TODO 4: Insert the relevant code into the **`getBMICategory()`** method of the **`BMICalculator`** class.

Pass the **`obtainHeight`** method, which obtains user input for height in inches or meters based on the entered BMI system. The **`obtainWeight`** method is already implemented for you. You can follow the same structure when implementing the body of the **`obtainHeight`** method.

🖥️ It's time to get coding!

- 🖥️ TODO 5: Insert this into the **`obtainHeight()`** method of the **`UserInput`** class.

- Initialize the reference variable **`UserInput userInput`** to create an object of the **`UserInput`** class.
- Initialize the reference variable **`BMICalculator bmiCalculator`** to create an object of the **`BMICalculator`** class.
- Complete the body of the **`switch`** statement:
    - **`case "IMPERIAL"`**: Call the relevant methods of the **`UserInput`** class to obtain **`weight`** and **`height`** and call the **`calculateBmiImperial()`** method of the **`BMICalculator`** class.
    - **`case "METRIC"`**: Call the relevant methods of the **`UserInput`** class to obtain **`weight`** and **`height`** and call the **`calculateBmiMetric()`** method of the **`BMICalculator`** class.
    - **`case ""`**: Display a message asking the user to enter a valid BMI system.

- Call the **`getBMICategory()`** method of the **`BMICalculator`** class and assign the result to the **`bmiCategory`** variable.

🖥️ It's time to get coding!

- 🖥️ TODO 6: Insert the relevant code into the **`main()`** method of the **`Main`** class.

Now that you have implemented all the necessary methods and logic in your BMI calculator, it's time to put your application to the test.

🖥️ It's time to get coding!

- 🖥️ TODO 7: Run your code using the IDE. Carefully input test values for both Imperial and Metric systems to ensure that the program correctly calculates BMI and identifies the appropriate category.

Testing your program

Compare the output of your program to the expected results. Here are some test cases you can use:

1. Imperial System Test Case:
   - Weight: 150 pounds
   - Height: 65 inches
   - Expected BMI: Approximately 24.96
   - Expected Category: Normal weight

```
Welcome to the BMI calculator app!
Please select the preferred BMI calculation system (Imperial/Metric):Imperial
Enter weight in Pounds:150
Enter height in inches:65
Your BMI is: 24.958579881656807
Your BMI category is: Normal weight

Process finished with exit code 0
```

2. Metric System Test Case:
   - Weight: 90 kilograms
   - Height: 1.75 meters
   - Expected BMI: Approximately 29.39
   - Expected Category: Overweight

```
Welcome to the BMI calculator app!
Please select the preferred BMI calculation system (Imperial/Metric):Metric
Enter weight in Kilos:90
Enter height in meters:1.75
Your BMI is: 29.387755102040817
Your BMI category is: Overweight

Process finished with exit code 0
```

If the results are not as expected, review your code to identify any issues, make necessary corrections, and retest until the functionality is as expected.

After you have tested and confirmed your project functionality, if you are happy with your results you can now submit your assignment for grading and feedback!

To submit your assignment, look for and click the blue "Submit Assignment" button in the top right corner of your lab environment. This sends your code off to the grader for evaluation.

When you submit your assignment, the grader is designed to check how well your code performs its intended functions based on the instructions for this assignment. Think of it as a thorough code review!

You can submit as many times as you'd like! Use the grader as a partner to help improve your code with every submission.

Here's the crucial part: the grader needs to compile and run your application in order to work. If it can't get past this stage, your grade will unfortunately be a 0. If you submit your work and the grade is a 0, there is a good chance that your hard work couldn't compile - not necessarily that your logic or functionality is wrong. So double-check that everything compiles and runs correctly in the IDE before submitting.

By completing this lab, you have demonstrated your ability to successfully write code to implement multiple classes and methods and invoke those methods in different classes. Classes act as templates that you can use to create objects. Grouping the data in attributes/properties and the functionality/behavior in methods within classes helps you implement code that adheres to object-oriented programming (OOP). Well done!