CPSC 121
Lab 04: Conditionals

For this exercise, create one folder for VS Code titled Lab_04. Inside this folder, you should have four classes, each representing the exercises below. For example, you will have a class called ***DollarEvaluator*** *which will host the solution to 1.1. The remaining classes should be titled,* ***FitnessTracker****,* ***HeartRateMonitor****, and finally,* ***RestaurantEvaluator****.*

1.1: DollarEvaluator

Write an application that accepts from the user the number of dollars he/she has and prints how wealthy they are based upon the number entered. Prompt for and read an integer value (use the `Scanner` class) for the number of dollars. Write an if statement that prints "Sorry kid. You're broke!" if the dollars variable has a value of 0. Observe the result.

Refactor (modify) your application to use an if-else that prints "Sorry kid. You're broke!" if the dollars variable has a value of 0, but prints "You've got some spending money!" otherwise. Confirm the behavior is working.

Finally, update your application to use an if-else-if that prints "Sorry kid. You're broke!" if the dollars variable has a value of 0, but prints "You've got some spending money!" if the dollars variable is less than 100, and prints "Looks to me like you're rich!" otherwise. Confirm the behavior is working.

1.2: FitnessTracker

Write an application that represents a fitness tracker and gives encouragement as it tracks your number of steps. Prompt the user for their current number of steps taken today and read (use the `Scanner` class) in the number into an integer variable. Create an integer constant that represents the daily step goal and set it equal to 10,000. Write an if-else statement that will print "You're almost halfway there!" if the number of steps is less than half of the step goal and will print "You're over halfway there!" if the steps are greater than or equal to half of the step goal. Confirm the behavior.

Refactor (modify) your application to use an if-else-if that prints "Way to get a good start today!" if the number of steps is less than a tenth of the step goal, prints "You're almost halfway there!" if the number of steps is less than half of the step goal and will print "You're over halfway there!" if the steps are greater than or equal to half of the step goal. Only one statement should be printed to the user. Confirm the behavior.

1.3: HeartRateMonitor

Write an application that represents a heart rate monitor that helps keep the user's heart rate within a desire range during a workout. Prompt and read (use the `Scanner` class) in the current heart rate into an integer variable. Create two integer constants which represent the target lower bound (assign the value 120) and the target upper bound (assign the value 150). Write an

if-else-if statement that will print “You’re right on track!” if the user is between the upper and lower bounds (inclusive), print “You’re doing great, but try to push it a bit!” if the user is below the lower bound and print “You’re on fire! Slow it down just a bit.” if the user is above the upper bound. Confirm the behavior.

1.4: RestaurantEvaluator

Write an application that will help determine whether a restaurant satisfies all of your friends’ needs. Two of you have very strong opinions and have laid out the following requirements:

- You want to eat somewhere that has either fish or pizza
- Your friend wants to eat somewhere with vegan options

Prompt and read (use the `Scanner` class) from the user three boolean variables that represent whether the restaurant has fish, has pizza, and has vegan options. Write an if-else statement that will print “Let’s go!” if the restaurant matches the group’s dietary requirements and otherwise will print “Sorry, we’ll have to think of somewhere else.” Confirm the behavior (see truth table below).

Finally, refactor (modify) your code so that you replace the if-else with a proper use of the ternary (conditional) operator. Confirm the behavior remains unchanged.

Has Fish?	Has Pizza?	Has Vegan?	We Can Go?
true	false	false	false
true	true	false	false
true	false	true	true
true	true	true	true
false	false	false	false
false	true	false	false
false	false	true	false
false	true	true	true

1.5: README

Update the plain text file called README and write your name and class section on the top. Follow the directions as described below.

Documentation: README

Update the plain text file called README and write your name and class section on the top.

1. Fill out the **Overview** section
   Concisely explain what the program does. If this exceeds a couple of sentences, you are going too far. I do not want you to just cut and paste, but paraphrase what is stated in the project specification.
2. Fill out the **Compiling and Using** section
   This section should tell the user how to compile and run your code. It is also appropriate to instruct the user how to use your code. Does your program require user input? If so, what does your user need to know about it to use it as quickly as possible?
3. Fill out the **Discussion** section
   Think about and answer the following reflection questions as completely as you can. You may not have an "earth-shattering" reflection response to each one but you should be as thoughtful as you can.

   Reflections...
   - What problems did you have? What went well?
   - What process did you go through to create the program?
   - What did you have to research and learn that was new?
   - What kinds of errors did you get? How did you fix them?
   - What parts of the project did you find challenging?
   - Is there anything that finally "clicked" for you in the process of working on this code?
   - Is there anything that you would change about the lab?
   - Can you apply what you learned in this lab to future projects?
4. Fill out the **Testing** section
   You are expected to test your projects before submitting them for final grading. Pretend that your instructor is your manager or customer at work. How did you ensure that you are delivering a working solution to the requirements?

Rubric

- Do the programs work as described? (Needs my sign-off): **10 points**
- Did you follow the directions?: **5 points**
- Did you follow style and coding practices: **10 points**
- **Total: 25 points**

Submitting the Lab

Compress the submission files (see below) into a .zip file named: **Lab04*_LastName_FirstName*.zip** When you are done and ready to submit, submit the following files in the .zip file:

- `DollarEvaluator.java`
- `FitnessTracker.java`
- `HeartRateMonitor.java`
- `RestaurantEvaluator.java`