Lab 7

1. **Accept** the **Lab 7** assignment using GitHub Classroom link with your Github account.

2. Clone your GitHub Classroom repository using IntelliJ IDEA.

3. Complete **Lab 7 exercises 1-3** as described in the comments and the requirements in this file.

4. Test your solutions in the **main functions**.

5. Once your code runs correctly in **main**, use the provided **JUnit tests** to verify your work.

   - If the tests fail, use the **debugger** to fix your code.
   - **Do not modify the JUnit test files. Any changes will be detected.**

6. Use **Git** to submit your code to your GitHub Classroom repository.

   - Check the **GitHub Actions results** to confirm that your code passes all tests.

7. After completing all the steps above, be prepared to **demo** your code to me.

In this lab, you will use Java Threads to solve problems.

Exercise 1: Multi-threaded File Reader

Objective: Finish the given **MultiThreadedFileReader** program that reads data from multiple text files concurrently, count lines and characters, output the data, and join all three threads.

Instructions:

- The run() method should read the content of a file and print the number of lines and characters.
- Three instances of FileReaderThread are given, each reading from a different text file.
- Ensure the main thread waits for all FileReaderThread instances to complete using join().
- Implement proper error handling (e.g., file not found, I/O exceptions).
- Place some random Thread.sleep() into your code, to simulate a delay in reading and processing the file.
- Implement the run method in FileReaderThread.
- Finish implementing the main method.

Exercise 2: Producer-Consumer Problem

Objective: Finish implementing the classic producer-consumer problem using threads.

Instructions:

- Analyze the Buffer class that acts as a shared resource with synchronized put() and get() methods.
- Finish implementing the two given thread classes: Producer and Consumer. Both run() methods of Producer and Consumer need to be implemented. They should generate random integers and place them in the buffer, while the Consumer should retrieve and print them.
- Check how wait() and notifyAll() is handled in the Buffer class/thread.

Exercise 3: Thread Synchronization with Bank Account

Objective: Finish implementing a simulation of multiple threads accessing and modifying a bank account.

Instructions:

- A BankAccount class is given, with methods deposit(int amount) and withdraw(int amount).
- Finish implementing the run() method from the AccountHolder class.
- The run method creates random transactions with random amounts.
- Print the balance before and after each transaction and log any failed withdrawal attempts.