**Portfolio Project: Part 2**

**Pseudocode, Source Code, and Execution Screenshot**

Jeremy Matthews

Colorado State University Global

CSC450 – Programming III

Dr. Reginald Haseltine

Due Date: 08/04/2024

**Portfolio Project: Part 2 - Pseudocode, Source Code, and Execution Screenshot**

**Pseudocode**

Program CounterApp

Declare lock as new ReentrantLock

Declare condition as newCondition from lock

Declare isReady as boolean and set to false

Function main

Create executor as newFixedThreadPool with 2 threads

Execute CountUp using executor

Execute CountDown using executor

Shutdown executor

While executor is not terminated

Do nothing (busy wait)

Print "All tasks finished"

Class CountUp implements Runnable

Function run

Lock the lock

Try

For i from 0 to 20

Print "Count Up: " + i

Sleep for 100 milliseconds

Set isReady to true using CounterApp.setReady

Signal condition

Catch InterruptedException

Print error message

Interrupt the current thread

Finally

Unlock the lock

Class CountDown implements Runnable

Function run

Lock the lock

Try

While isReady is false using CounterApp.isReady

Wait for condition

For i from 20 to 0

Print "Count Down: " + i

Sleep for 100 milliseconds

Catch InterruptedException

Print error message

Interrupt the current thread

Finally

Unlock the lock

Function isReady as synchronized boolean

Return isReady

Function setReady as synchronized void

Input: ready as boolean

Set isReady to ready

**Source Code**

package java\_concurrency\_counter;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

public class CounterApp {

private static final Lock lock = new ReentrantLock(); // Create a lock for synchronization

private static final Condition condition = lock.newCondition(); // Create a condition for synchronization

private static boolean isReady = false; // Flag to indicate that the CountUp thread has finished

public static void main(String[] args) {

ExecutorService executor = Executors.newFixedThreadPool(2); // Create a thread pool with 2 threads

executor.execute(new CountUp()); // Execute the CountUp task

executor.execute(new CountDown()); // Execute the CountDown task

executor.shutdown(); // Shutdown the executor

while (!executor.isTerminated()) { // Wait for the executor to finish

}

System.out.println("All tasks finished"); // Print a message indicating that all tasks have finished

}

// Define the CountUp task

static class CountUp implements Runnable {

@Override

public void run() {

lock.lock(); // Acquire the lock

try {

for (int i = 0; i <= 20; i++) { // Count from 0 to 20

System.out.println("Count Up: " + i); // Print the count

Thread.sleep(100); // Wait for 100ms between each count

}

CounterApp.setReady(true); // Set the ready flag to true

condition.signal(); // Signal the CountDown thread

} catch (InterruptedException e) {

System.err.println("Count Up interrupted: " + e.getMessage()); // Print an error message

Thread.currentThread().interrupt(); // Interrupt the thread

} finally {

lock.unlock(); // Release the lock

}

}

}

// Define the CountDown task

static class CountDown implements Runnable {

@Override

public void run() {

lock.lock(); // Acquire the lock

try {

while (!CounterApp.isReady()) { // Wait until the CountUp thread has finished

condition.await(); // Wait for the signal

}

for (int i = 20; i >= 0; i--) { // Count from 20 to 0

System.out.println("Count Down: " + i); // Print the count

Thread.sleep(100); // Wait for 100ms between each count

}

} catch (InterruptedException e) {

System.err.println("Count Down interrupted: " + e.getMessage()); // Print an error message

Thread.currentThread().interrupt(); // Interrupt the thread

} finally {

lock.unlock(); // Release the lock

}

}

}

public static synchronized boolean isReady() { // Define a synchronized getter method for the ready flag

return isReady; // Return the ready flag

}

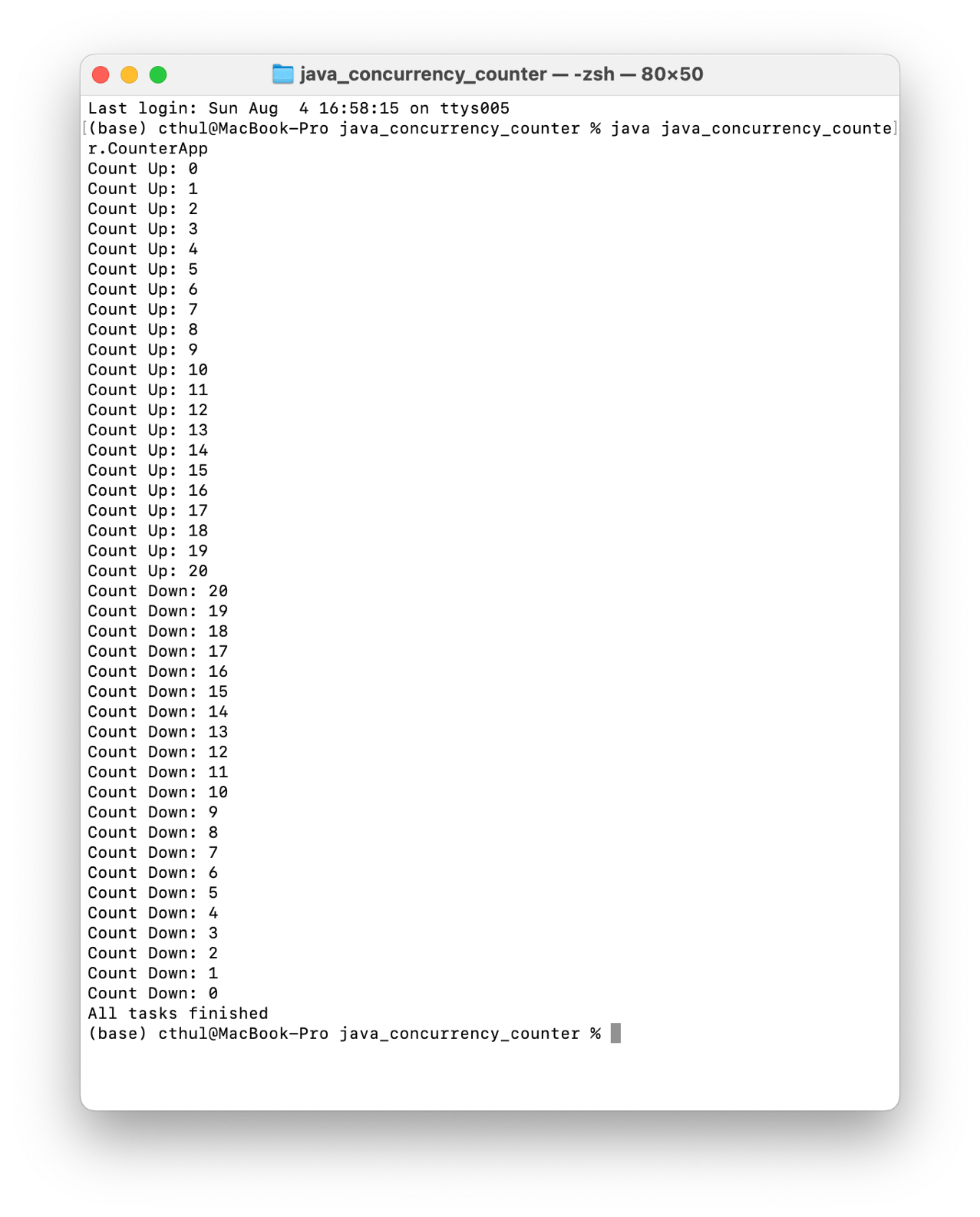
public static synchronized void setReady(boolean ready) { // Define a synchronized setter method for the ready flag

isReady = ready; // Set the ready flag

}

}

**Program Execution Screenshot**

****