**Program Analysis:**

**Pseudo Code:**

class Main {

final int MAX = 20;

Lock lock = new Lock();

Thread CountUpThread {

run() {

lock.lock();

for (int i = 1; i <= MAX; i++) {

print("Counting up: " + i);

}

lock.unlock();

}

}

Thread CountDownThread{

run() {

lock.lock();

for (int i = MAX; i >= 0; i--) {

System.out.println("Counting down: " + i);

}

lock.unlock();

}

}

main() {

CountUpThread t1 = new CountUpThread();

CountDownThread t2 = new CountDownThread();

t1.start();

t2.start();

}

}

**Src Code:**

/\*\*

\* Alec Mcdaugale

\* CSC450-mod8-concurrency

\* Brief:

\* This application should create two threads that will act as counters. One thread should count up to 20. Once thread

\* one reaches 20, then a second thread should be used to count down to 0.

\*/

import java.util.concurrent.locks.ReentrantLock;

/\*\*

\* Main Class.

\*/

public class Main {

/\*\*

\* Max value to count up to.

\*/

private static int MAX = 20;

/\*\*

\* Lock prevent threads from counting concurrently.

\*/

private static final ReentrantLock lock = new ReentrantLock();

/\*\*

\* A class to count up on a thread.

\*/

private static class CountUpThread extends Thread {

public void run() {

// Hold lock to block another thread from executing.

lock.lock();

try {

for (int i = 1; i <= MAX; i++) {

System.out.println("Counting up: " + i);

}

} finally {

// Release thread to allow the next thread to execute.

lock.unlock();

}

}

}

/\*\*

\* A class to count down on a thread.

\*/

private static class CountDownThread extends Thread {

public void run() {

lock.lock();

try {

for (int i = MAX; i >= 0; i--) {

System.out.println("Counting down: " + i);

}

} finally {

lock.unlock();

}

}

}

public static void main(String[] args) {

CountUpThread t1 = new CountUpThread();

CountDownThread t2 = new CountDownThread();

// These threads will execute concurrently, this will test if locks work.

t1.start();

t2.start();

}

}

**Screen Shot:**



**GitHub:**