# Parallel and MultiThreaded Programming

# CSYE 7215

# Homework 1

# Due: January 20, 2020

Put all your java, compiled class files and documentation into a zip file named Homework1.zip and submit it via the dropbox on the blackboard before the END of due date. Put your name on all .java files. There will be a short quiz on this assignment.

1. Describe the followings:

Process?

Thread? Provide five examples, Advantages?

Task?

Yield?

States of Java Threads with Wait/Notify

2. Write a Java program to define a Student class with instance variables name, id, midterm, project, and final. Name is a string whereas others are all integers. Add a static variable nextId, which is an integer and statically initialized to 1. Have some overLoaded constructors. In each constructor, id should be assigned to the next available id given by nextId. The default constructor should set the name of the student object to “StudentX” where X is the next id. Add a calculateGrade() method which returns a string for the letter grade of the student, like “A”, “B”, “C”, “D” or “F”, based on the overall score. Overall score should be calculated as (30% midterm + 30% project + 40% final).

Write TestDriver to test your program. It should create 25 student objects with default constructor and invoke the setter methods for midterm, project and final with random numbers ranging from 50 to 100 inclusive. Then it should print the student information via the toString() method. Student information provided by toString() should include name, midterm, project, and final and the letter grade given by the calculateGrade() method.

3. Describe Java monitor under the following circumstances, show the sequence of Thread

executions:

A thread *t* is blocked on a *wait* invocation in a monitor.

Another thread *q* is blocked waiting to enter the monitor by calling a *synchronized*method.

A third thread *r* is in the monitor and calls *notify*.

The thread *q* enters the monitor before *t* regains the monitor lock.

4. Write Java code to create five threads, (4 ReaderThreads, and one WriterThread) where threads share access to method m1() and an ArrayList<Integer>. Design your program for threads to shared object ArrayList without creating any collisions and deadlock.

Note: Consider program to manage blocked and wait and notify states of thread.

5. Create file data.txt and add this line: a,b,c,d,b,c,a,b,b,c,d,e,d,d,e,e,a,b Write a program that reads text line from file using BufferedReader and FileReader:

a) print it on the console,

b) add it to a list,

c) read it from the list and change it to upper case,

d) count the number of strings ,

e) add it to hashMap with key/value (string, frequency)

Note: first use HashSet to get frequencies,

f) sort the map using TreeMap,

g) iterate through the map and print and Catch

for FileNotFoundException (file not found), and IOException (invalid file).

6. Create interface I that extends interface I1 with m1() and m2() methods, and interface I2 with m3() and m4() methods. Create class A that implements interface I. The implementation of each method must print the name of method. Create a Test class that instantiates an object of class A but uses interface I as its type. The Test class must execute all the methods of class A. Compile and run the code.

7. Compile and Run this code. Explain what and how it does it Step-by-Step?

public class Input {

int index;

int[] input = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};

public Input(){

index = 0;

}

public void print(int index){

System.out.println(input[index]);

}

synchronized public int getIndex(){

if(index == 15)

return -1;

return index++;

}

}

public class MyThread implements Runnable{

Input ip;

Object lock;

public MyThread(Input ip, Object lock){

this.ip = ip;

this.lock = lock;

}

@Override

public void run() {

int index = -1;

while((index=ip.getIndex())!=-1){

synchronized(lock) {

System.out.println(

Thread.currentThread().getName());

ip.print(index);

}

}

}

}

public class Caller {

public static void main(String[] args)

throws InterruptedException {

Input ip = new Input();

Object lock = new Object();

Thread t1 = new Thread(new MyThread(ip, lock), "Thread1");

Thread t2 = new Thread(new MyThread(ip, lock), "Thread2");

t1.start();

t2.start();

t1.join();

t2.join();

}

}

8. Write a TestClass to test the following code. Compile and Run the code.

**public** **class** Data {

**private** String packet;

    // True if receiver should wait

    // False if sender should wait

**private** **boolean** transfer = **true**;

**public** **synchronized** **void** send(String packet) {

**while** (!transfer) {

**try** {

                wait();

            } **catch** (InterruptedException e)  {

                Thread.currentThread().interrupt();

                Log.error(**"Thread interrupted"**, e);

            }

        }

        transfer = **false**;

**this**.packet = packet;

        notifyAll();

    }

**public** **synchronized** String receive() {

**while** (transfer) {

**try** {

                wait();

            } **catch** (InterruptedException e)  {

                Thread.currentThread().interrupt();

                Log.error(**"Thread interrupted"**, e);

            }

        }

        transfer = **true**;

        notifyAll();

**return** packet;

    } }

9. What is the result of this Lambda function?

Runnable task = () -> {

String threadName = Thread.currentThread().getName();

System.out.println("Hello " + threadName);

};

task.run();

Thread thread = new Thread(task);

thread.start();

System.out.println(“Done!");

Hello main

Hello Thread-0

Done!