Advanced Programming 2025 – Year 2
Labwork 5: (5% - or 50 points out of 300 points for labwork this semester)

NOTE: ALL LABS TO BE COMPLETED IN PROJECTS USING ECLIPSE OR EQUIVALENT IDE (NO MORE TEXTPAD, EVER EVER EVER!!!)

TOPICS INCLUDE: Threads, synchronized and some regex (regular expressions)

IMPORTANT NOTES:

- **NO COPYING PERMITTED AND ZERO MARKS WILL APPLY TO COPIED WORK. FURTHER ACTION MAY BE TAKEN AGAINST STUDENTS THAT HAVE BEEN FOUND TO COPY WORK.**

- **ASSESSMENT WILL INVOLVE ONE-TO-ONE QUESTIONS ABOUT YOUR SUBMITTED WORK. A COMPLETED SELF-ASSESSMENT SHEET WILL BE USED TO GUIDE THE ASSESSMENT. USE COMMENTS IN YOUR CODE TO ENSURE YOU DON'T FORGET WHY YOU WROTE CODE YOU MAY LATER BE ASKED ABOUT.**

- **ALL WORK MUST BE SUBMITTED TO MOODLE BY DATES SPECIFIED (SUBMISSION DEADLINES WILL BE POSTED ON MOODLE).**

- **MANY OF THE TASKS ASSIGNED BELOW CAN BE COMPLEX AND\OR THE DESCRIPTIONS MAY REQUIRE FURTHER CLARIFICATIONS. PLEASE USE THE AVAILABLE LAB TIMES TO ASK FOR CLARIFICATIONS AND ADVICE\HINTS ON THE TASKS BELOW.**

- **YOU MUST USE AN IDE TO COMPLETE TASKS (e.g., Eclipse or IntelliJ or NetBeans or similar). Support and help provided for Eclipse primarily.**

- **CHATGPT and other similar AI tools that can code simple solutions are NOT PERMITTED. THEY DO NOT TEACH YOU HOW TO BECOME A GOOD PROGRAMMER EITHER!**

Part 1 – Use the 'extends Thread' approach for threading (10 points)

Create an Eclipse Project called **Lab5Part1**. Create a class called **ThreadUsingExtends** that will print the letters "A", "B", "C" and "D" (from a **static** array of Strings) to the default output device using a loop. Add a constructor to the **ThreadUsingExtends** class that will number and name each of the thread's created…use your surname as the thread base name, e.g., "Raeside 1" will be the name of my first thread and Raeside 2,3,4 etc., will be the name of the successive threads I would create. Print the name of the current thread with each letter in the loop, e.g., the output should output "Raeside 1 A", "Raeside 2 A" etc. The printing of the array must be done within the thread run method. Use the **extends Thread** approach to add threads to this program.
Create a second class called **TestThread**, which will test the creation and execution of the **ThreadUsingExtends** program. Create at least **FIVE INSTANCES** of the **ThreadUsingExtends** class and call start on all threads created. Note that the output should be unpredictable, as the threads cannot guarantee the order of execution. You may have to run this a few times to see the order of output is unpredictable.

Required activities and marking guideline:

- Implement thread at class level using the **extends** approach (2 points)
- Set the name of the threads and set the count of each thread +1 (3 points)
- Implement the run method to output thread name plus A,B,C,D (3 points)
- Write and run the TestThread application with min FIVE instances (2 points)

Part 2 – Use the 'implements Runnable' approach for threading (10 points)

Create an Eclipse Project called **Lab5Part2**. Create a class called **ThreadUsingRunnable** that will print the numbers 1, 2, 3 and 4 (from a **static** array of integers) to the default output device using a loop.
Set the name of the current thread to your surname, e.g., the output should be "Raeside 1", "Raeside 2" etc. The printing of the array must be done within the thread run method. Use the **implements Runnable** approach to add threads to this program.
Create a second class called **TestThread**, which will test the creation and execution of the **ThreadUsingRunnable** program. Create at least **FIVE INSTANCES** of the **ThreadUsingRunnable** class and call start on each (**Note:** because this uses the Runnable interface you MUST wrap the creation of the **ThreadUsingRunnable** instances in a Thread before calling *start()*). Again the output should be unpredictable, as with Part 1. Note: Setting the Thread name in this part is different to Part 1 above because this is the **Runnable** interface and contains no setName method – access the current thread statically to set it's name.

Required activities and marking guideline:

- Implement thread at class level using the **implements** approach (2 points)
- Set the name of the threads (static access to current thread) (3 points)
- Implement the run method to output thread name plus 1,2,3,4 (3 points)
- Write and run the TestThread application with min FIVE instances (2 points)

Part 3 – Synchronized (controlling thread access) and Regex
(15 points – 7 points for part (a) 8 points for part (b))

a) Create an Eclipse Project called **Lab5Part3a**. Create a new versions of **Part1** and class above so the threaded application ALWAYS prints A followed by B followed by C and D for **Part1** using a correctly placed **synchronized** block (Note: Because the operating system scheduling has ultimate control the behavior of the threads is not always 100% reliable on all platforms, if your tests fail still put the synchronized block in anyway!). Test the fixed version of Part 1 by running the test program several times.

Required activities and marking guideline:

- Identified code needing restriction; **fix** with **synchronized** block (2 points)
- Create the new version of **Part 1** (3 points)
- Test fixed classes so that the output is always A,B,C,D [in order] (2 points)

b) Create an Eclipse Project called **Lab5Part3b**. Create a random String of about 50 words long. Place randomly in this string several instances of your first name and surname (at least FIVE OF EACH). Write a regex pattern to find all instances of either your first name OR second name within the String. Show the location and string found in each match.

Required activities and marking guideline:

- Create the random String with your first and surname in it (x 5) (1 point)
- Create the Pattern to search for either first OR second name (3 points)
- Create the Match from the pattern (2 points)
- Show the location and pattern found for all matches in the String (2 points)

Part 4 – A GUI Example (with Graphic) that needs threads (15 points)

Create an Eclipse Project called **Lab5Part4**. Create a JFrame program called **StopTheLights** which draws a sequence of traffic lights to the screen in the usual pre-defined sequence for traffic lights continuously with reasonable time lapses, i.e., red, amber and green (use Graphics and drawOval and/or fillOval to draw the shapes\lights). Provide two buttons on the GUI to start the playing of the traffic lights sequence and one to stop the traffic light sequence. YOU MUST implement thread programming in the traffic lights painting sequence so that the GUI button(s) can be released to stop the lights.

Required activities and marking guideline:

- Code Graphics lights sequence in Frame (red/amber/green) (5 points)
- Implement relevant listeners etc. (2 points)
- Implement the threads in the draw sequence (5 points)
- Achieve and demonstrate the stopping of the light sequence (3 points)