CS 420
HOMEWORK ASSIGNMENT H3

DUE DATE: Wednesday, October 10

1. Multi-Threaded programming – Race Hazard – 1.

Write a Java application that creates two threads that share access to a single instance of the `Resource` class shown below. Each thread should repeatedly get the value of the `value` attribute and then change it. One thread should increment it by one and the other should decrement it by one. In order to simulate the way in which race hazards can arise, one thread should sleep a random time between getting and setting the value, while the other thread should sleep a random time after setting the value. Be sure to get the current value just before setting a new value. The `Resource` object will report a mismatch resulting from a race condition occurring. The user can terminate the program by typing ctl-c at the command line. Your code should use random sleep times between 10 and 200 milliseconds.

Create an MS-DOS batch file named `race1.bat` that will execute the program. Store all files necessary for the execution of your application in a folder named `Prob1` that you submit electronically for the assignment, as instructed below. Be sure the output of the program clearly indicates what the program is doing.

```
// A shared resource. Its value attribute should only
// be adjusted by +1 or -1 via setValue().
// Used to illustrate race hazards.

class Resource {
    private volatile int value;

    public int getValue(){
        return value;
    }

    // Value should only be changed by 1
    public void setValue(int v){
        if(Math.abs(value-v) != 1){
            System.out.println("Mismatch: "+value+" "+v);
        }
        value = v;
    }
}
```

2. Multi-Threaded programming – Race Hazard – 2.

The random sleeping in the first exercise above is only necessary to show a race hazard if the thread scheduling is not done using a round-robin algorithm. Remove the calls to `sleep()` and explain your observation of how the program operates with respect to the scheduling done by the JVM being used.

Create an MS-DOS batch file named `race2.bat` that will execute the program. Store all files necessary for the execution of your application in a folder named `Prob2` that you submit electronically for the assignment, as instructed below. Be sure the output of the program clearly indicates what the program is doing.

3. Multi-Threaded programming – Race Hazard – 3.

Assume that the `getValue()` and `setValue()` methods are declared as synchronized methods. Explain what this would do to prevent the race hazard. Demonstrate the operation of the modified `Resource` class with the application in exercise 1.

Create an MS-DOS batch file named `race3.bat` that will execute the program. Store all files necessary for the execution of your application in a folder named `Prob3` that you submit electronically for the assignment, as instructed below. Be sure the output of the program clearly indicates what the program is doing.

4. Multi-Threaded programming – Race Hazard – 4.

Modify the `Resource` class used above to implement a single synchronized method used by threads to alter the `value` attribute. Explain what this would do to prevent the race hazard. Demonstrate the operation of the modified `Resource` class with the application in exercise 1.

Create an MS-DOS batch file named `race4.bat` that will execute the program. Store all files necessary for the execution of your application in a folder named `Prob4` that you submit electronically for the assignment, as instructed below. Be sure the output of the program clearly indicates what the program is doing.

5. Process Scheduling – 1

Suppose that the following processes arrive for execution at the times shown in the following table. Each process' next CPU burst time is shown. Answer the following questions using non-preemptive scheduling and base all decisions only the information available at the time the decision must be made.

Process ID	Time of Arrival	Next CPU Burst Time
P_A	0.0	7
P_B	1.2	3
P_C	3.0	2

1. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

2. What is the average turnaround time for these processes with the SJF scheduling algorithm?

3. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P_A at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 2 units of time, and then SJF scheduling is used. Remember that processes P_A and P_B are waiting during this idle time, so their waiting time may increase.

6. Process Scheduling – 2

The following processes are currently in the ready state, with the length of the CPU-burst time given

in milliseconds. The processes are assumed to have arrived all at the same time in the order P_A, P_B, P_C, P_D, P_E,.

Process ID	Next CPU Burst Time	Process Priority (1=highest)
P_A	12	4
P_B	2	2
P_C	3	4
P_D	2	6
P_E	7	3

1. Draw Gantt charts that illustrate the execution of these processes using SJF, FCFS, non-preemptive priority, and RR (quantum = 1) scheduling.

2. Compute the turnaround time of each process for each of the scheduling algorithms above.

3. Compute the waiting time of each process for each of the scheduling algorithms above.

4. Which of the four schedules above results in the smallest average waiting time for this set of processes?

7. Process Scheduling – 3

One reason for using a quantum to interrupt a running process after a "reasonable" period is to allow the operating system to regain the processor and dispatch the next process. Suppose that a system does not have an interrupting clock and the only way a process can lose the processor is to relinquish it voluntarily. Suppose also that no dispatching mechanism is provided in the operating system.

(1) Describe how a group of user processes could cooperate among themselves to implement a user-controlled dispatching mechanism.
(2) What potential dangers are inherent in this scheme?
(3) What are the advantages to the users over a system-controlled dispatching mechanism?

Either type your solutions or print legibly. Solutions that cannot be easily deciphered are incorrect!

General Instructions:

- First four problems receive 70% of the total score for the assignment..
- Homework submissions must be prepared using computer document preparation applications such a word processor or similar editor. Handwritten solutions are not acceptable – neatness, readability and grammar count!
- Homework submissions will be clearly marked with the student's name, date and assignment identification at the top of the first page.
- All homework is to be completed by each student individually and represent that student's original, unassisted work. Any material copied in any way from other sources must be clearly identified and attributed.

- The non-programming problem solutions are printed on paper and submitted at the start of class on the due date.

Programming Instructions:

- Put a block of comments at the beginning of every physical file containing program source code that includes your name, the course name and number, information identifying what functions the program is designed to perform, and instructions how to execute the program. (required)

- For the programming problems, place the Java source files, class files, batch files and all other files necessary to execute each program you write into a separate Windows folder that is named `Prob1`, `Prob2`, etc.

- Place a batch file in each problem folder that will execute the program in that folder.

    - For example, The `prob1.bat` file is a text file with the following format (where `TheProgam` is the name of the class file containing the `main()` method):

      ```
      java TheProgam
      pause
      ```

    - Double-clicking the `prob1.bat` file should cause your program to execute.
    - Don't forget to put the `pause` command in the last line

- Place the programming problem folders into a zip file. Create the zip file so that the folder structure (path) is also recorded by selecting the "save full path info" option. Use your email ID as the zip file name. Example of file structure for submission:

    - Zip file named `brixiusn.zip` contains folders named `Prob1`, `Prob2 and Prob3`. Each folder has source code, class files and any required data files or batch files. The zip file records the path information for each file.

- Submit the zip file in the course Canvas site using the assignment submission capability in the same location you accessed this assignment information. You can also add comments when you submit a file for the assignment if you wish.

- Each assignment must be submitted on Canvas by the start of class on the day the assignment is due.

- Submit only one zip file with your entire assignment.

- If you have already submitted a homework assignment and then decide you must resubmit the assignment before it is due, you can submit another zip file to replace previous submissions. You can also use comments with the submission to further explain your submission to the grader. You may resubmit as many times as you find necessary before the assignment due date.

- DO NOT FORGET TO ALSO SUBMIT THE PRINTED NON-PROGRAMMING PROBLEM SOLUTIONS AND SOURCE CODE AT THE START OF CLASS ON THE DUE DATE.