Homework 4

1. (20 pts) For the following program, explain the interesting elements related to threads. Focus on explaining the output of the program.

```
 1 public class TaskThreadDemo {
 2   public static void main (String args []) {
 3     String [] sa = {"a", "X", "+", "."};
 4     for (String s: sa) {
 5       Runnable ps = new PrintChar (s, 200);
 6       Thread ts = new Thread (ps, s);
 7       ts.start ();
 8     } // end for each character
 9   } // end main
10 } // end class TaskThreadDemo
11
12 class PrintChar implements Runnable {
13   String ch;
14   int times;
15
16   public PrintChar (String c, int n) {
17     ch = c;
18     times = n;
19   } // end constructor
20
21   public void run () {
22     for (int i = 0; i < times; i++) {
23       System.out.print (ch);
24     } // end for loop
25   } // end method run
26 } // end class PrintChar
```

I would say an interesting element is that the Thread class implements the Runnable class. This means a thread could be passed to a thread as an initialization parameter, though this is not the case for this program. When this program runs, four PrintChar instances, which implement Runnable, are created and passed as initialization parameters to a new Thread, which is then started.

Each of these threads will output the character passed to the PrintChar class 200 times. Since these are separate threads running concurrently, the order in which the characters appear is a bit jumbled. Sometimes there will be two or more of the same character in a row, while other times there may be many single characters. It all depends on how the processor passes control to different threads.

2. (20 pts) What is changed if the method called on line 7, start(), is replaced with run()? Explain (of course). Focus on explaining the output of the program.

If the method call is changed to run(), the program will still run, and each character will be displayed to the console 200 times. The difference is that the program will wait for the first Thread to finish before even creating the next Thread. This is because calling the run() method is simply calling the method as if it were any other normal method. The start() method creates a new Thread and calls the run() method within that Thread.

3. (20 pts) What is changed if the method Thread.yield() is added between lines 23 and 24? Explain. Focus on explaining the output of the program.

Functionally, I did not notice any difference after adding Thread.yield() inbetween lines 23 and 24. According to the Java 8 API (2020), the yield() method simply notifies the scheduler that the calling Thread is willing to let other processes go first. Since all of the generated Threads in this program, aside from the main thread, are all instances of the same class, they are all calling Thread.yield(). As such, even if the scheduler honored the notice, it would honor it for all at least once.

4. (20 pts) Modify the above program so that the Thread.sleep method is called after each character has been printed causing it to sleep for 500 milliseconds. Describe how that modification has altered the output and explain why the change had the effect that you described.

The modification altered the output to print each character one time followed by a short pause, then repeated. This change happened, because each thread is waiting 500 milliseconds before requesting use of the processor again. The processor was able to meet the demands of each thread before they requested the processor again.

5. (20 pts) Modify the above program so that the Thread.sleep method is called after each thread is created in the main method causing it to sleep for 500 milliseconds. Describe how that modification has altered the output and explain why the change had the effect that you described.

Modifying the program so the sleep was called after each thread was created (i.e. between lines 7 and 8), allowed enough time for each of the threads to print the character 200 times before the next thread started. While not exactly the same, it created a similar effect as calling run(). This is because the print (ch) method does not require many CPU cycles, so forcing the main thread to sleep for an entire half-second gives each thread more than enough time to complete.

Grading Rubric:

Attribute	Meets	Does not meet
Problem 1	**10 points** Lists the events associated with each provided component.	**0 points** Does not list the events associated with each provided component.
Problem 2	**10 points** Lists the methods JTable implements. Lists the methods which are required by the interfaces implemented by the JTable class beyond those interfaces implemented by the various parent classes of JTable.	**0 points** Does not list the methods JTable implements. Does not list the methods which are required by the interfaces implemented by the JTable class beyond those interfaces implemented by the various parent classes of JTable.
Problem 3	**10 points** Addresses the differences among the various layout managers. Focuses on their behavior as their container is resized.	**0 points** Does not address the differences among the various layout managers. Does not focus on their behavior as their container is resized.
Problem 4	**20 points** Explains what is wrong with everybody doing the actions provided. Explains how the actions be fixed to avoid deadlocks. Explains if the solution provided is starvation free.	**0 points** Does not explain what is wrong with everybody doing the actions provided. Does not explain how the actions be fixed to avoid deadlocks. Does not explain if the solution provided is starvation free.
Problem 5	**20 points** Explains what methods a class implementing the java.util.concurrent.locks.Lock interface must implement. Describes some of the expected characteristics of each of the methods of this interface.	**0 points** Does not explain what methods a class implementing the java.util.concurrent.locks.Lock interface must implement. Does not describe some of the expected characteristics of each of the methods of this interface.

References

Oracle. (2020, July 09). *Thread (Java Platform SE 8)*. Java™ Platform, Standard Edition 8 API Specification. https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html