**Homework 4**

**Name: Sairam Soundararajan**

1. (20 pts) For the following program, explain the interesting elements related to threads. Focus on explaining the output of the program.
2. public class TaskThreadDemo {
3. 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
11. class PrintChar implements Runnable {
12. String ch;
13. int times;

15

1. public PrintChar (String c, int n) {
2. ch = c;
3. times = n;
4. } // end constructor 20
5. public void run () {
6. for (int i = 0; i < times; i++) {
7. System.out.print (ch);
8. } // end for loop
9. } // end method run 26 } // end class PrintChar

The string variable *sa* holds an array of 4 characters. For each value, the runnable object is created to print each character 200 times. The PrintChar class implements runnable since its instances are intended to be executed by the thread. A runnable object is assigned to a new PrintChar constructor, which takes in the instance of the string array and the numeric value of how many times each value in the array will be executed. A new thread object takes in a runnable object and the string instance in the thread constructor. When the program is run the output will result in a randomized order of each character. The order will repeat as many times as possible until there are 200 of each character displayed in the output. Running this program again will result in another random order.

For example, this is one of the outputs when using the start() method:

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXaaaaaaaaaaaaaaaa+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX...................................................................................................................................................................................+++++.....................

The start() method results in randomized output because it makes the code execute on a new thread each time. For each character, when start() is called, it begins printing. However, another thread starts printing the array before the current thread finishes printing all the characters.

1. (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.

When start is replaced with run(), each character of the string array will print exactly each character 200 times in the order of the array. This will be true no matter how many times the program is run.

The following example illustrates the console output when run() is used instead start():

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++........................................................................................................................................................................................................

When calling the run() method, the run() method is not creating a new thread. So each time the run() method is called, it will, instead, execute on the current thread. You can’t have multiple instances of the run method concurrently running on separate threads. This means that when you call the run() method, it completely finished before the next call.

1. (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.

The output does not appear to get changed even if Thread.yield() is added between lines 23 and 24. However, the yield method may help to reproduce bugs due to race conditions. It is also useful when designing concurrency control constructs.

1. (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.

When thread.sleep is inside the for loop, it would cause the program to run very slowly. When the first character prints for the first time and then sleeps in 500 ms, all the other threads will be able to start and print their first character and sleep for 500 seconds as well. Then, it will print a second time followed by each of the other threads printing again when 500ms are up for the first thread. This will continue until all 200 of each character is printed in order.

1. (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.

Since thread.sleep is being called in the main method, this will cause the main thread to sleep after creating the new thread. The new thread will have 500 ms to run completely and print all characters. Then once the 500 ms are up, it will start the next thread to print the next character 200 times and the main thread will sleep again, letting the character be printed 200 times. All 200 of each character will print in order.

**Grading Rubric:**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Meets** | **Does not meet** |
| Problem 1 | **20 points**  Explains the interesting elements related to threads. Focuses on  explaining the output of the program. | **0 points**  Does not explain the interesting elements related to threads. Does not focus on  explaining the output of the program. |
| Problem 2 | **20 points**  Explains what is changed if the method called on line 7, start(), is replaced with run().Focuses on  explaining the output of the program. | **0 points**  Does not explain what is changed if the method called on line 7, start(), is replaced with run(). Does not focus on explaining  the output of the program. |
| Problem 3 | **20 points**  Explains what is changed if the method Thread.yield() is added between lines 23 and 24. Focuses on explaining the output of the program. | **0 points**  Does not explain what is changed if the method Thread.yield() is added between lines 23 and 24. Does not focus on explaining the output of the program. |
| Problem 4 | **20 points**  Explains how the output is changed if the Thread.sleep method is called after each character has been  printed. | **0 points**  Does not explain how the output is changed if the Thread.sleep method is called after each character has been  printed. |
| Problem 5 | **20 points**  Explains how the output is changed if the Thread.sleep method is called after each thread is created in the  main method. | **0 points**  Does not explain how the output is changed if the Thread.sleep method is called after each thread is created in the  main method. |