JavaMultiThreadYieldWithPriority

```
/*This program is a multithreaded JAVA Program Example.  The program creates three threads each printing
characters.  Two threads print characters and a third prints integers.  The threads are given priorities
at the beginning of the run.  The number print thread has lowest priority. These threads do yield after
certain number of prints*/
import java.io.*;
import java.io.IOException;
import javax.swing.*;
import java.util.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.lang.IllegalStateException;
import java.util.NoSuchElementException;
import java.lang.*;// this allows the threads to be created as objects from the Thread class
//import java.lang.Thread;//this allows the threads to be run


public class JavaMultiThreadYieldwithPriority
 {
   public static void main(String[] args)throws Exception
    {
    // Create a PrintWriter for the tasks.
      PrintWriter outf1;
      outf1=new PrintWriter(new File("JavaMultiThreadOutPriority1.txt"));
     // First we must create tasks.  We will pass these tasks to the threads for running
        Runnable printA=new PrintChar('a',100,outf1);
        Runnable printB=new PrintChar('b',100,outf1);
        Runnable print100=new PrintNum(100, outf1);
        Runnable printbreak=new Pbreak();
      // Now Create the Threads
         Thread thread1=new Thread(printA);
         Thread thread2=new Thread(printB);
         Thread thread3=new Thread(print100);
   //Now give them priorities
         thread1.setPriority(Thread.MAX_PRIORITY);
```

```java
            thread2.setPriority(Thread.NORM_PRIORITY);
            thread3.setPriority(Thread.MIN_PRIORITY);


        // Thread thread4=new Thread(printbreak);
        // Now start the threads
          thread1.start();
          thread2.start();
          thread3.start();
        //  thread4.start();

        outf1.flush();
            }
        }      // Now let us create a class for each Task Object
        // first we create a class for the PrintChar Task Object
        class PrintChar implements Runnable{
         private char charToPrint;// this is the character to print
         private int times; // this is the number of times to repeat the print
         private PrintWriter outf;
         // now for the constructor
         public PrintChar(char c, int t, PrintWriter outf1)
          { charToPrint=c;
            times=t;
            outf=outf1;
           }
          /* every task object must have a run() method.  This overrides the system run() method
           and tells the system what task to perform */
           public void run()
            {
              for(int i=1; i<=times; i++)
               { System.out.print(charToPrint);
               outf.print(charToPrint);
                outf.flush();
               if (i%10==0)
                  {System.out.println();
                  outf.println();
                   outf.flush();
                 //Thread.yield(); Releases Processor
                  }
                }
            }// end of run()
        }// end of class PrintChar
    // Now let us create a class for the integer print
      class PrintNum implements Runnable{
        private int lastNum; // this is the last integer to print
        private PrintWriter outf; // this is the output text file from this thread
        // now for the constructor
        public PrintNum(int n, PrintWriter out1)
            {lastNum=n;
             outf=out1;
            }//end of constructor
           // now the task run method
           public void run()
            {/*NOTE that System.out.print is a system function that is synced with the
               processing thread so no flush is necessary to assure the output buffer is
               empty.  BUT
               Printing to a text file from a PrintWriter is not synced.  So IT IS NECESSARY TO
               ALWAYS FLUSH THE PrintWriter BUFFER after each print to assure it gets on the
               text file before the thead looses control.*/

              for(int i=1;i<=lastNum; i++)
               {System.out.print("  "+i);
                outf.print("  "+i);
                outf.flush();
               if (i%5==0)
                 {System.out.println();
                  outf.println();
                  outf.flush();
                // Thread.yield();
                 }
                }
             }// end of run()
            }//end of class PrintNum



  b lab 2 3 4 5
```

MM♣Mbabb 6abaaaa 7 8a 9b 10aabbb
MM♣M
MM♣M
MM♣M 11aaaaaaaaaa
MM♣Mba 12b 13 14 15
MM♣M 16ba 17aaaab 18abbb 19abab 20a
MM♣Maabaaaaaaaa
MM♣M
MM♣M
MM♣Maa 21ba 22ba 23ba 24ba 25ba
MM♣Mba 26ba 27ba 28b
MM♣M 29ba 30
MM♣Ma
MM♣Mba 31ba 32ba 33b 34ab 35ab
MM♣Mab 36ab 37ab 38
MM♣Mb 39aaaa
MM♣Maaaa 40ba
MM♣Mba 41b
MM♣M 42ba 43ba 44ba 45ba
MM♣Mab 46 47 48 49abb
MM♣Mbbba 50bab
MM♣Mab 51ab 52
MM♣Mb 53ab 54ab 55
MM♣M 56a
MM♣M 57ab 58ab 59ab 60ab
MM♣Mab 61ab 62ab 63
MM♣Mb 64ab 65ab
MM♣Ma
MM♣Mab 66a 67ba 68ba 69ba 70ba
MM♣Mba 71b
MM♣M 72b 73b 74b 75
MM♣M
MM♣Mb 76b 77b 78b 79b 80b
MM♣Mb 81b 82b 83b 84
MM♣M 85b
MM♣Mb 86b 87b 88b 89b 90b
MM♣Mb 91b 92b 93
MM♣M 94 95

Fall 2017 CSC 406 Problem 1

Consider now a computer system with four Process threads. The system has only one processor (yes I know this is a very old system) so all four threads must share the same processor. The processor rules are as follows;

1. **The GUI.** The GUI runs in the fore ground (Highest Priority) and continually polls the user for input through the keyboard or the mouse. The system has one function to Poll. The system polls one time and then releases the processor to one of the other functions. The GUI has the highest priority of all functions.
2. **The Word Processor.** The Word Processor runs in the mid ground. It has a function that accepts input from the keyboard. The WP runs 10 cycles and then releases the processor to another process.
3. **The Data Storage Device.** The Data Storage Device runs in the background. It retrieves/stores data from the disk. The Data Storage Device runs 3 cycles and then releases the processor.
4. **The Printer Device.** The printer Device has the same background priority as the Data Storage Device. The printer prints 6 lines and then releases the processor to another thread.

Write a program that simulates the computing system. Simulate the devices as follows;

1. **The GUI.** Input to the GUI task is the number of seconds that the system must run. Remember that the GUI polls the two times per second. Simulate the GUI by having the thread print "GUI Poll" and then skip a line each time the GUI polls.
2. **The WP.** The WP processes numbers of characters. Have your WP task take as input the character and the number of times the character must be processed. Remember that the WSP processes 10 characters and then releases the processor. Simulate the WP by having the task take as input the character and the number of times it must be processed. Have your program print WPx each time the WP has processed 10 characters. Skip a line each time the WP has processed 10 characters.
3. **The Data Storage.** Input to the data storage device is the number of characters the device must retrieve. The data storage device simply retrieves 20 characters per each of its three cycles and then yields the processor. Input to the DS task will be the number of characters to retrieve. To simulate the DS task have your DS function print DSx where x is the number of the character retrieved.
4. **The Printer Device.** The printer prints 6 lines before yielding the processor. The printer is configured so that each line has 60 characters. As input to the printer device, accept the number of characters to be printed. As output, from your simulated function for each line write "Print Line X characters AA thru BB". Where X is the line number and AA is the number of the initial character on the line and BB is the number of the ending character on the line.

Set up your simulation to launch all 4 system threads with an appropriate priority. Create a text output file from the simulation.

You as Chief Systems Engineer have been asked if the thread priorities (all equal priorities) are "correct". The idea is that if the system is running properly it has the following characteristics;

1. The GUI runs the entire period of the run. All things must shut down before the GUI. 4
2. The WP must be fast and responsive when given a processing load. The faster it finishes, the happier the user is. 1
3. The Data Storage Device must finish before the GUI but normally slower than the WP. 2
4. The Printer can finish anytime. 3

Consider the following test for the system for a run of 5 minutes. With the following thread calls:

GUI(300 seconds) Runs 600 times

WP(A, 50x10x2)

DS(2500)

Printr(3600)

Run your simulation once with the processing priorities as given by the system design team. Then run again with a better priority schema that you purpose. Turn in both runs and justify (this means a written paragraph or two) why your schema is better than the design team's schema.

25 points Due 22 Sept. You must turn a written justification for your schema.

about 10 pages output

where 1st device finished