Process Implementation Assignment

Assignment objective:

Implement the process creation and destruction cycle within an operating system via simulating processes using threads.

Overview and Design:

1. Create a class called Kolonel which will act as the part of the operating system known as the kernel which creates processes, destroys processes, and keeps track of the Process Control Blocks (PCBs) of all the currently existing processes.
   a. The kernel maintains a table of all the PCBs which is an array of size n.
   b. A unique PCB index, 0 through n-1, refers to each process that exists in the system.
   c. Each PCB is a structure consisting of only the two fields:
      1. Parent: a PCB index corresponding to the process' creator.
      2. Children: a pointer to a linked list, where each list element contains the PCB index of one child process.
   d. The Create method of the Kolonel class creates a new process. The parameter to the Create method is the index of process p requesting the kernel to create a new child process q. The create method returns a pointer to a new process object. The process creation task performs the following steps:
      1. Allocate a free PCB for process q.
      2. Record the parent's index, p, in the PCB for process q.
      3. Initialize the list of children in the PCB of process q as empty.
      4. Add the index of process q to the list of children for process p.
      5. Create a new process object for process q and start its execution.
      6. Return the pointer to the new process object.
   e. The Destroy method of the Kolonel class removes the PCB of a process from the table of process control blocks maintained by the kernel. The parameter to the destroy method is the index of the process p for removal. The process destruction task performs the following steps:
      1. Remove the index p from the list of children of process p's parent.
      2. Deallocate the PCB from element p in the table of process control blocks.
   f. The Create and Destroy methods are the only public methods of the Kolonel class.

2. Create a class called ProcessT which will simulate a process using a thread. A process performs the following steps:
   a. Print a message indicating the process' creation. Include its index in the message.
   b. Use the random number generator to produce a number (in the range of 0 to 5) of child process' to create.
   c. For each child process creation, the process sleeps for a certain number of seconds and then call the kernel to create the child process. The process uses the random number generator to produce the sleep length of time, a number in the range of 11 to 30.
   d. After creating all the child processes, join with each child process (waiting for the child process to terminate) and then call the kernel to destroy the child process. The method of the Thread class you'll need for this is join().
   e. Print a message indicating process' deletion. Include its index in the message.

3. A thread which executes a call to the Thread.sleep(m) method sleeps, does nothing, for m milliseconds. Since the value to put a thread to sleep is given in seconds then that value must be multiplied by 1,000 before passing it as the parameter to the Thread.sleep method.

4. Create a single random number generator object that every process object uses.

5. Create a driver class and make the name of the driver class **Assignment1** containing only one method: `public static void main(String args[]).`
   The main method performs the following steps:
   a. Create the random number generator object.
   b. Create the Kolonel object.
   c. Call the create method of the Kolonel object to create the very first process. Use the number -1 as the index of the parent of this process since it has no parent process.

   I compile and run your program via the command line using the Java JDK. Therefore, the command I type to execute your program is **java Assignment1**. Make sure your project compiles and runs via the command line using the Java JDK. I will not use any other Java development tool to compile and run your project code.

6. You must declare public each class you create which means you define each class in its own file.

7. You must declare private all the data members in every class you create.

8. You can use "**implements Runnable**" or "**extends Thread**" to implement the class(es) defining the threads in this assignment. You can't use extends in this assignment in defining any class except if you are using "**extends Thread**" to define a thread class.

9. **Tip:** Make your program as modular as possible, not placing all your code in one .java file. You can create as many classes as you need in addition to the classes described above. Methods being reasonably small follow the guidance that "A function does one thing and does it well." You will lose a lot of points for code readability if you do not make your program as modular as possible. But, do not go overboard on creating classes and methods. Your common sense guides your creation of classes and methods.

10. Do **NOT** use your own packages in your program. If you see the keyword **package** on the top line of any of your .java files then you created a package. Create every .java file in the **src** folder of your Eclipse project, if you're using Eclipse.

11. Do **NOT** use any graphical user interface code in your program!

12. Do **NOT** type any comments in your program. If you do a good job of programming by following the advice in number 9 above then it will be easy for me to determine the task of your code.

Grading Criteria:

The assignment is worth a total of 20 points, broken down as follows:

1. If your code does not implement the task described in this assignment, then the grade for the assignment is zero.
2. If your program does not compile successfully then the grade for the assignment is zero.
3. If your program produces runtime errors which prevents the grader from determining if your code works properly then the grade for the assignment is zero.

Important: Make sure your project compiles and runs via the command line using the Java JDK because I will not use any other Java development tool to compile and run your project code.

If the program compiles successfully and executes without significant runtime errors then the grade computes as follows:
Followed proper submission instructions, 4 points:

1. Was the file submitted a zip file.
2. The zip file has the correct filename.
3. The contents of the zip file are in the correct format.
4. The keyword **package** does not appear at the top of any of the .java files.

Code implementation and Program execution, 12 points:

- The driver file has the correct filename, **Assignment1.java** and contains only the method **main** performing the exact tasks as described in the assignment description.
- The code performs all the tasks as described in the assignment description.
- The code is free from logical errors.
- Program output, the program produces the proper results for the assignment.

Code readability, 4 points:

- Good variable, method, and class names.
- Variables, classes, and methods that have a single small purpose.
- Consistent indentation and formatting style.
- Reduction of the nesting level in code.

Late submission penalty: assignments submitted after the due date are subjected to a 2-point deduction for each day late.

Late submission policy: you **CAN** submit your assignment early, before the due date. You are given plenty of time to complete the assignment well before the due date. Therefore, I do **NOT** accept any reason for not counting late points if you decide to wait until the due date (and the last possible moment) to submit your assignment and something happens to cause you to submit your assignment late. I only use the date submitted, ignoring the time as well as Blackboard's late submission label.

Submission Instructions:

Go to the folder containing the .java files of your assignment and select all (and **ONLY**) the .java files which you created for the assignment in order to place them in a Zip file. The file can **NOT** be a **7z** or **rar** file! Then, follow the directions below for creating a Zip file depending on the operating system running on the computer containing your assignment's .java files.

Creating a Zip file in Microsoft Windows (any version):

1. Right-click any of the selected .java files to display a pop-up menu.
2. Click on **Send to**.
3. Click on **Compressed (zipped) Folder**.
4. Rename your Zip file as described below.
5. Follow the directions below to submit your assignment.

Creating a Zip file in Mac OS X:

1. Click **File** on the menu bar.
2. Click on **Compress ? Items** where ? is the number of .java files you selected.
3. Mac OS X creates the file **Archive.zip**.
4. Rename **Archive** as described below.
5. Follow the directions below to submit your assignment.

Rename the Zip file with the filename having the following format:

your last name,
followed by an underscore _,
followed by your first name,
followed by an underscore _,
followed by the word **Assignment1**.

For example, if your name is John Doe then the filename would be: **Doe_John_Assignment1**

Once you submit your assignment you will not be able to resubmit it!
Make absolutely sure the assignment you want to submit is the assignment you want graded.
There will be **NO** exceptions to this rule!

You will submit your Zip file via your CUNY Blackboard account.
The only accepted submission method!

Follow these instructions:

Log onto your CUNY Blackboard account.
Click on the CSCI 340 course link in the list of courses you are taking this semester.
Click on **Programming Assignments** tab in the red area on the left side of the webpage.
You will see the **Process Implementation Assignment**.
Click on the assignment.
Upload your Zip file and then click the submit button to submit your assignment.

Due Date: Submit this assignment on or before 11:59 p.m. Monday, April 19, 2021.