Process Creation Assignment

Project objective:

Compare the performance of process creation and destruction when implemented with and without linked lists.

Overview:

Method 1 of the process creation hierarchy uses linked lists to keep track of child processes as described in section 5.1 (within **Week 5 – Class 9 and 10**) **The process control block**, subsection **The PCB data structure**.

For the purposes of performance evaluation, the PCBs are simplified as follows:

a. Implementation of the table of all the PCBs is an array of size n.
b. Each process is referred to by a PCB index, 0 through n-1.
c. Each PCB is a structure consisting of only the two fields:
    1. Parent: a PCB index corresponding to the process's creator.
    2. Children: a pointer to a linked list, where each list element contains the PCB index of one child process.

The necessary functions are simplified as follows:

a. Create(p) represents the create function executed by process PCB[p]. The function creates a new child process PCB[q] of process PCB[p] by performing the following tasks:
    1. Allocate a free PCB[q].
    2. Record the parent's index, p, in PCB[q].
    3. Initialize the list of children of PCB[q] as empty.
    4. Create a new link containing the child's index q and appending the link to the linked list of PCB[p].
b. Destroy(p) represents the destroy function executed by process PCB[p]. The function recursively destroys all descendent processes (child, grandchild, etc.) of process PCB[p] by performing the following tasks for each element q on the linked list of children of PCB[p]:
    1. Destroy(q). /* recursively destroy all descendants. */
    2. Free PCB[q].
    3. Deallocate the element q from the linked list.

Method 2 of the same process creation hierarchy uses no linked lists. Instead, each PCB contains the 4 integer fields parent, first_child, younger_sibling, and older_sibling, as described in the subsection **Avoiding linked lists**.

Design:

1. Implement the two methods of the process creation hierarchy in Java.

2. Assume that PCB[0] is the only currently existing process and write a test procedure that performs a series of process creations and destructions. For example:

   ```
   Create(0)   /* creates 1st child of PCB[0] at PCB[1]*/
   Create(0)   /* creates 2nd child of PCB[0] at PCB[2]*/
   Create(2)   /* creates 1st child of PCB[2] at PCB[3] */
   Create(0)   /* creates 3rd child of PCB[0] at PCB[4] */
   Destroy(0)   /* destroys all descendents of PCB[0], which includes processes PCB[1] through PCB[4] */
   ```

3. The test procedure should do a significant number of process creations and deletions with the calls to create and destroy intermingled

4. Run the test procedure repeatedly in a long loop on each method, with the test procedure computing and then printing the running time for the method. This should show how much time method 2 saves, which avoids dynamic memory management.

5. One or more classes implement the test procedure and the two methods.

6. 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 essentially executes the test procedure on each method.

   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.

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

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

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 project.

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.

Save 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 **Assignments** tab in the red area on the left side of the webpage.
You will see the **Process Creation 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. Tuesday, October 27, 2020.