Operating System - Assignment 2

Task description:

Mutex locks and semaphores, as discussed in class, are software solutions to solve the race condition and to ensure an efficient synchronization between cooperating threads. You will use the Java IDE you find appropriate (e.g., NetBeans, Eclipse) to solve this assignment.

Step1: Create a new Java project with a name of your choice.

Step2: Create a package with a name of *Threads_Synchornization_CProblems*

Step3: Add the three .java files you find in the assignment folder to the package you created in *step2* (e.g., copy and paste the files, drag and drop the files).

The files are for three different synchronization problems, with full skeleton. The comments in the java files explain the required functionalities and guide you to ***where exactly you should place your implementation***, read them carefully before start coding. **DO NOT CHANGE** any already existed code (e.g., function or variable name).

- In the ***first question***, the *deposit and withdraw functions* share a bank account to add certain amount or subtract certain amount from the balance, respectively. Use *semaphore lock(s)* to implement the synchronization.

- In the ***second question***, the *producer and consumer functions* share an array of integer values to add items or remove items from the buffer, respectively. Use mutex lock(s) to implement the synchronization.

- In the ***third question***, the *Ascending and Descending functions* share an array of integer values to sort them in ascending or descending order, respectively. Use either mutex lock(s) or semaphore lock(s) to implement the synchronization. Implement the sorting method you find appropriate (e.g., selection sort, bubble sort, insertion sort), but don't use collection methods/APIs (e.g., Arrays.sort).

Note: the threads running the above questions are implemented to run in an infinite loop, so at some point you may need to force stop the program.

Submission:

1. This is an **individual assignment** -- Cheating/plagiarism will be checked and will receive zero.

2. Submit only **ONE ZIP** named with ***yourLastname_youFirstname.zip***, to the folder titled Assignment 2 under the D2L Assignments tab (*other formats will not be accepted*).

3. The .ZIP file contains:
   - The ***three java files*** you worked on.
   - One ***Runtime.pdf***: - For each of the three java files, copy the **full** java code and add a clear screenshot of the **output after running the code**.

4. The assignment is due 10:00pm - **Wednesday March 4th**. You can submit your assignment, within 24 hours after this due date, to be graded out of 50% of the assignment's grade. After this grace period your late submission will not be accepted.

5. **Also Print the developed Runtime.pdf and hand it to me during the lecture of Thursday March 5th.**