German University in Cairo
Department of Computer Science
Dr. Haythem O. Ismail

Operating Systems, Spring Term 2012
Project 1: Sirens and Pirates

Due: April 19, at 23:59

1. Project Description

On some Pacific island, a group of sirens and pirates are planning to sail to a nearby island with abundant rainfall. The problem is that only one rowboat is available. The boat can ferry exactly four individuals; it will not leave the shore if less, or more, than four individuals are aboard. After boarding the boat, one individual will take charge of the oars in order to row the group to safety. To guarantee the safety and sanity of both sirens and pirates, it was agreed that no siren should be in the boat with three pirates, and no pirate should be in the boat with three sirens. Any other combination is possible.

In this project, you will write a Java program to simulate the activities of the sirens and pirates. Each siren and/or pirate will be represented by a Java thread. Mutual exclusion and synchronization will be enforced using semaphores and barriers. Since Java provides neither semaphores nor barriers (only monitors), you will need to implement semaphore and barrier ADTs. You will do this using *only* methods in the Java Thread class, in addition to wait, notify, and synchronized.

Thus, there are three tasks that you need to complete:

a) Implementing a semaphore ADT, using *only* the methods available in the Java Thread class, wait, notify (and/or notifyAll), and synchronized.

b) Implementing a barrier ADT, using *only* the methods available in the Java Thread class, wait, notify (and/or notifyAll), and synchronized.

c) Using your implementation of semaphores and barriers to establish the synchronization and mutual exclusion required to simulate siren and pirate activities. In particular, your implementation should include (at least) two methods:

   1. Board, which is executed by each siren and/or pirate as they board the boat. You should ensure that all four threads from each boatload call Board before any of the threads from the next boatload do.
   2. rowBoat, which is executed by exactly one of the threads in a boatload, indicating that this thread is in charge of the oars. Once this call is made, you may assume that the boat empties its load onto the rainy island and magically returns back for the next load.

2. Groups: You may work in groups of **at most three.**

3. Deliverables

a) Source Code

- You should implement a semaphore ADT as explained in task (a) above.
- You should implement a barrier ADT as explained in task (b) above.
- Your program should simulate the activities of sirens and pirates. In your simulation, all sirens and all pirates should be safely transferred to the rainy island.
- Your program should take as input the number of sirens and the number of pirates. (You may make any reasonable assumptions about these numbers.)
- A fancy user interface is not required, but your output should be presented in a user-friendly way that would show
  1. who boards the boat when, and
  2. who executes `rowBoat` when.

b) Project Report, including the following.

- A discussion and analysis of the synchronization problems in the sirens and pirates problems.
- A description of the main classes you implemented.
- A discussion of how you implemented semaphores.
- A discussion of how you implemented barriers.
- A discussion of how you used semaphores and barriers to solve the sirens and pirates problem.
- A description of how to run and exit the program.
- Proper citation of any sources you might have consulted in the course of completing the project.
- If you use code available in library or internet references, a full explanation of how the code works.
- If your program does not run, a discussion of what you think the problem is and any suggestions you might have for solving it.

4. Important Dates

Source code. On-line submission by April 19 at 23:59. Directions for on-line submission are available on the course web site.

Project Report. A hard-copy should be submitted. You have two options:

a) April 19, by 16:00, or

b) April 21, by 16:00, provided that an *identical* on-line version is submitted with the code (by April 19 at 23:59).

Brainstorming Session. In tutorials.