**Question 16.5** – Suppose you have a class with three methods (first, second and third). Suppose you created an instance of this class and passed it to three methods. Re-write first, second and third so that you will be guaranteed that they will get executed in the order first, second and third, regardless of what the threads do.

|  |
| --- |
| **public** **class** Question {  **public** Semaphore sem1, sem2, sem3;  **public** Question() {  **try** {  sem1 = **new** Semaphore(1);  sem2 = **new** Semaphore(1);  sem3 = **new** Semaphore(1);  sem1.acquire();  sem2.acquire();  sem3.acquire();  } **catch** (Exception e) {  }  }  **public** **void** first() {  **try** {  //perform steps here  sem1.release();  } **catch** (Exception e) {}  }    **public** **void** second() {  **try** {  sem1.acquire();  sem1.release();  //perform steps here  sem2.release();  } **catch** (Exception e) {}  }  **public** **void** third() {  **try** {  //perform steps here  sem2.acquire();  sem2.release();  } **catch** (Exception e) {  }  }  } |

* When an object calls “aquire” on a semaphore, it will either move on to the next line of code (if the semaphore is not already being aquired by some other object) or it will wait until it can aqure it. The aquired method of a semaphore probably checks one boolean variable (something that states whether it is free or currently in use) and just gets stuck in some forever for loop until it is free.
* The thread that initialized the contructor aquired all the semaphores. If the other threads call “aquire” before “release” they will get trapped waiting.
* First doesn’t call aquire at all. So it can be executed. It also calls release. This allows second to be able to be executed.
* Second calls release on semaphore 2. Now third can be executed.
* The reason we used semaphores instead of locks (like the re-entrant lock) is that semaphores can be released or aquired by any thread. For example, any thread can call first and release semaphore 1. With locks, only the object that called lock and owns the lock can unlock it.
* The difference is that locks dictate WHO can run the first and second operation. Semaphores are only concerned with the maintaining an order of execution. If there is a section that is protected by a lock, it means that you can’t enter that section unless the lock owner allows you to (i.e. unlocks it for you). If there is a section that is protected by a semaphore, there should be an event or condition that should release the semaphore.