Kumaravinash Konduru

Kxk151330

CS 4348.501

Through this project I feel that I was able to better understand how threads and semaphores work when used to make and create a Java program. And it is through making such a project that I now understand why they are used and the benefits of using them within a project or program.

Looking at the project it has many moving parts to it that once stacked onto each other make it quiet complex in design. The core part of this project was the 3 thread that were used, labeled as Guest, FrontDesk, and Bellhop. It was instances of each of these threads that were used within the project to make it work. The FrontDesk thread had 2 instances of it created, the Bellhop thread had 2 instances of itself created, and finally the Guest thread had 25 instances of itself created to run the program. In order to coordinate all of these parts of the programs together a lot of semaphore were used. Conversely it could also have been done using thread processes such as thread waiting, but due to the fact that using such methods for thread coordination were explicitly not allowed for use within the project. In addition to the use of semaphores within the project I also used a couple arrays, initialized as a semaphore array or as an in array. The semaphore array was used as a control mechanism for each individual Guest thread while the int arrays where mainly used as a way to store information about the front desk employee and bellhop used by each individual Guest instance. By far the use of these arrays allowed most the control mechanism necessary for getting information to and from each individual Guest instance was well as allowing for the control of these instances. On top of this another thing that was used by this program and this project was various queues which were very important because they were used in order to queue the number of Guests instances that were waiting to go to the Front Desk or the get assistance of a Bellhop. These queues were used within my project as another control mechanism that allowed for the control of various Guest instances. To make sure that each of the instances of all of the 3 thread ran properly I made sure that each thread class would also implement the runnable class which was used to create thread and then subsequently start them once needed. And once the process for each instance of the thread was over it could then be possible to join the instance of the thread to the man thread class and subsequently reduce its space in memory. Overall these were the methods that I was able to use in order to make the project work.

When it came to some of the difficulties with completing the project I feel that the main one that was of concern was just to think of how each thread and how each element of each thread could and would interact with each other. Because of the complexity of the interaction that were going to be needed I knew that I needed a more clear was of understand and a more basic level how this project was going to work and in that case doing the pseudocode I feel help a lot in trying to understand how each thread and each part of the project would fit in into the greater whole of the entire program. Most importantly writing the pseudocode beforehand allowed me to understand which semaphores would be needed and how many of those semaphores world be needed. Ultimately though by understanding how each semaphore would work in relation to another actually doing it as a program became a lot easier. Another difficulty that I experience while working on this project that I felt was a major inconvenience was just coordinating all of the instances of each thread to work in the way that I wanted them to work. In too many instances where there time that when I would compile me program it would provide an output that was wholly different than what was supposed to happen and this was made more difficult knowing that all of the project had to be done without the use of having the thread be able to sleep or do a busy wait. But eventually once I was able to get a handle of how I could control these instances without the need for thread sleeping or busy wait, through the use of numerously more semaphores, it became easier to construct and finish the rest of the project.