Karran Gowda

CS 640 Operating Systems Professor Gregory Simco

The Design and Programming Assignment took me three weeks to complete. I would work on the assignment every weekend to finish it. By working in increments, I was able to go through almost every component of the program. I worked on semaphores to make sure that I was able to get a program with semaphores working. I eventually got one program working. I got it to work by creating two classes called semThread and shareThread. I got two threads to work with the run method in the semThread class. The threads used a character variable and the run method reads the character variable. I created the main class called Burrito that contains the main method. I used code online to help me get this first program working (GeeksforGeeks). It works by using an extension called Runnable that automatically creates a new object by executing the run method in the program. The run method checks the character and then prints that the specific thread is waiting for a permit. The lock is acquired so the thread gets the permit. The thread sleeps for 50 milliseconds and then releases the permit. During that time, the program allows a switch to happen for another thread to execute.

After that, I wrote the methods and classes I wanted to use. I worked on the class anatomy for each class to make sure that I had the variables and methods I needed for this project. I worked on the Customer class constructing the run, start, waiting room, wait and signal methods. The run method was the method mainly created for the synchronization with semaphores. It is similar to the run method for semThread. I wanted to use semaphores and could not get it to work leading to many hours wasted. I needed to make sure that I had semaphores in the constructor and the constructor already had two parameters (String name and int num). I would have to create another constructor to get it to work or start over from scratch. I commented out the semaphores and still had the methods in the Customer class serve the metaphorical waiting room for the overall project. The Server class needed three servers for the project: the private beef area, the cheese area and the tortilla area. The Shop class needed the shared cash register and three counter locations. I first used print statements (for the Server class and the Shop class) since I needed to point out the status of every customer buying burritos. Next, I had to pass a name and a value to identify every object. I needed to make sure that the program tracks the specific object. This time I was able to get the customer name and the number of burritos the customer gets.

The following diagram describes the routing of a semaphore used in Java.

Get initial value for counter

Thread gets to acquire the permit(sem.acquire)

If count = 0 No

Yes

Shared resource to the thread is granted access by the semaphore

Thread releases the permit(sem.release)

I could not switch from one customer to another using ArrayList. I had the following method:

Public void setBurrito(int[] list, String name, int value)

{

int num = Integer.parseInt(name);

list.set(num, value); <= gives error

}

I was not able to sort the customers on shortest amount first. Java’s Runnable would not make it possible. I next changed the code for the main method to sort an array of random numbers in ascending order. I created a new array with 15 elements, assigned random values for each of them, and used Arrays.sort(array) to sort the array from the smallest to the largest amount. I used that to have Customer 1 get the shortest amount of burritos first. I could not get the Burrito file to have <E> at the end. I got an error for that.

The following diagram shows the process for a Java program that uses semaphores (TechBlogStation).

|  |
| --- |
| Semaphore |
| Permits = 3 |
| Permits = 2 |
| Permits = 1 |
| Permits = 0 |
| blocked |
| Permits = 1 release() – Thread - 2 |
| Permits = 0 |

Thread 1 – acquired – permits = 3

Thread 2 – acquired – permits = 2

Thread 3 – acquired – permits = 1

Thread 4 – acquired – permits = 0

GeeksforGeeks. (2018, October 12). Semaphore in Java. <https://www.geeksforgeeks.org/semaphore-in-java/>

*Semaphore in Java*. (2019, November 3). Tech Blog Station. https://techblogstation.com/java/java-semaphore/