**Methodology and Design Guidelines**

**Introduction**

This document goes into detail on the design and methodologies of a multi-threaded simulation of a burrito making restaurant. Separate Threads will be represented by servers, while other threads will represent customers. The problem this challenges is that threads must not cause concurrent modifications exceptions and alter data at the same time. To solve this issue of concurrent threading and synchronization, we will incorporate the Java Semaphore class to represent locking of threads. This will allow the program to output information in the command line interface in a clean chronological format, while also creating a safer code behind the scenes free of threading complications.

Functions and limitations of this project will goes as follows. “X” amount of customers will be created while only “Y” amount of these customers will be allowed to enter the store at any given time. The remaining “Z” amount of customers must wait outside the store until the “Y” amount of customers is below its threshold, at which point the outside customer will be added to the inside line. Once inside the restaurant, the new customer is added to the ordering line. Every time a customer is added to the ordering line, it is re-arranged by the customer’s order size from smallest to largest. The First server in line will then grab the first customer in line, which is the customer with the smallest order. They will continue to proceed with the order and cook up to max 3 burritos per customer. If the customer’s burrito order size is decremented by 3 and becomes equal to 0 or less then that customer will be sent to the register line. If the customer’s order size does not become 0 or less, they will be added to the ordering line again, and put in order depending on smallest order size. While the current server is added to the end of the serving line, and the next server grabs the next smallest order guest to serve. The register line can only hold up to 3 customers and if a third customer is added to it, then the current server will cash out that customer and they will leave satisfied. Meanwhile, that current server will be added back to the end of the serving line and the next server in line grabs the next smallest order customer. Once all outside customers, ordering line customers, and register line customers have all been handled and left satisfied the remaining 3 servers will clock out and the restaurant will close.

**Methodology**

**The objective of the methodology is to discuss the approach that was used to implement a**

**solution to a given problem. In this deliverable it is necessary to use prose not just simple**

**narratives. Your work should establish any assumptions for the assignment and explain**

**your direction. In this section you should discuss the details of your study.**

**In essence, a methodology covers the steps taken to produce the resulting design**

**document. This portion of the assignment provides the background on the solution.**

**Design**

**The design portion of the assignment must provide a detailed discussion of the**

**components used in the system. Information contained in the design document is**

**explicitly detailed. Once again, this document must contain sufficient detail such that a**

**programmer can implement your design, but does not include the actual code that**

**resulted from your design. It is necessary to include a title page and a table of contents**

**with your design.**

**The following outline is potential subject matter that may be included in your design**

**document.**

**Scope/Overview**

**The scope/overview should have introduced the problem and the design. Information**

**contained in this section should give an overview of what is to follow, as well as stating**

**any limitations, etc.**

**Data Design**

**In this section,**

**the design should have discussed each of the data items that are used in**

**the system.**

**Architectural Design**

**This section will discuss, in detail, each of the system components.**

**Class and Object Design/Modules**

**This section provides the details of the**

**classes and objects used in the modules that were**

**designed for the system. It is not acceptable to extract portions of the actual code and**

**include them in this section.**

**Interface Design**

**Include any human**

**-**

**computer interaction, external interfaces, or**

**internal interfaces in this**

**section.**

**Test Provisions**

**This section would include your testing and verification designs.**

**Annotations**

**Pg 128**

The most commonconcurrency problems discussedin a systems class are raceconditions, deadlocks, andstarvation. Race conditionsoccur when two threads cansimultaneously modify a shareddata structure. Shared accesscan lead to problems where thevalue of a set of operationsdepends on the order ofexecution of the threads ratherthan the ordinary semantics ofthe operation

Marmorstein, R. (2015). Teaching semaphores using... semaphores. *Journal of Computing Sciences in Colleges*, *30*(3), 117–125. doi: ISSN: 1937-4771 EISSN: 1937-4763