Assignment Three

REPORT

Thomas Courtney 3175353 | SENG2200 | 4.6.18

Time Investment

Time was divided into DESIGNING, CODING, TESTING, REVIEWING and CORRECTING. Where ideally the completion of the project would occur in phases of that order. Things did occur in this order however a very large portion of time was dedicated to CORRECTION. Had the other phases been executed more effectively this may not have been the case.

Date	Time	Hours	Description
[Wk 11] Monday 21.5.18	1600 - 1800	2	Spec Breakdown [DESIGN]
[Wk 11] Thursday 24.5.18	1850 - 1910	0.33	Research into discrete time simulations [DESIGN]
Wk [11] Saturday 26.5.18	1300 - 1800	2	Design + Building Classes [DESIGN]
Wk [11] Sunday 27.5.18	1200 - 1530	3.5	Designing Factory Class + Stats + Item [DESIGN]
Wk [11] Sunday 27.5.18	1700 - 1800	1	Designing Stages [DESIGN]
Wk [12] Monday 28.5.18	1700 - 2100	4	Implementing Priority Queues [CODING]
Wk[12] Wednesday 30.5.18	2300 - 0100	2	Joining the structures together [CODING]
Wk[12] Thursday 31.5.18	1700 - 2130	4.5	Testing simple structure (2 stages 1 queue). [TESTING]
Wk[12] Friday 1.6.18	1400 - 1700	3	Implementing interactions between more complex structures. [CODING]
Wk[12] Saturday 2.6.18	1300 - 1600	3	Repairing logic of updating the factory. [REVIEWING]
Wk[12] Saturday 2.6.18	1600 - 2300	5	Improving readability of code. It was so bad. [REVIEWING]
Wk[12] Sunday 3.6.18	1200 - 2400	12	Debugging blocking logic. [CORRECTING]
Wk[13] Monday 4.6.18	2400 - 0300	3	Debugging blocking logic. [CORRECTING]
	TOTAL	**45.33**	

Figure 1: Time dedication for assignment three for seng2200.

Had more time been invested in the design phase the overall time investment for the project may have been reduced (See Table 2).

Phase	Percentage of Time (%)
DESIGNING	19
CORRECTING	20
CODING	10
TESTING	18
REVIEWING	33

Figure 2: Distribution of time as percentages for assignment three seng2200.

UML DIAGRAM OF DESIGN

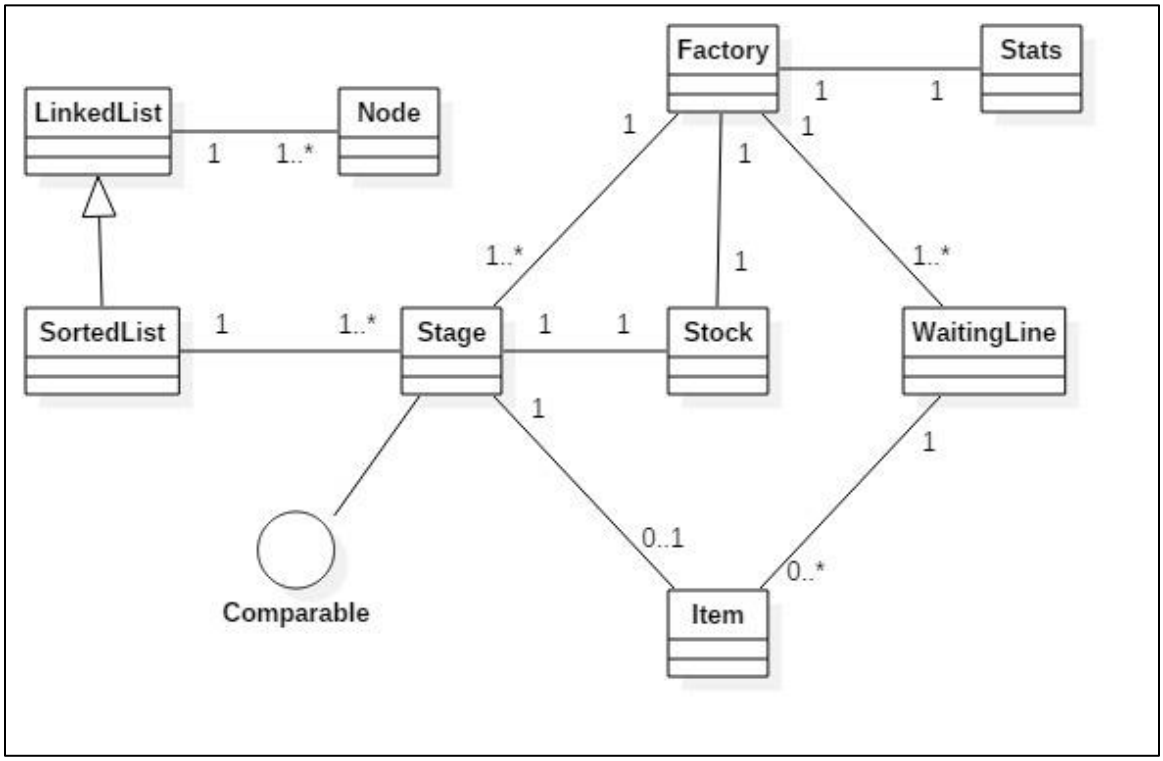

Figure 3: Class interactions for assignment three, seng2200

INHERITANCE AND POLYMORPHISM

There is little use of inheritance and polymorphism. One reason for this is that the program design consists of many small classes that perform specific tasks. The design could have been altered somewhat to include more polymorphism. For example, the Stock class could have been implemented as a special case of a Waiting Line. Otherwise it was not obvious given the design structure to use more inheritance or polymorphism.

The Sorted List data structure extends the Linked List class so that it may prioritize its contents based on their compareTo() method.

PROGRAM EXTENSIBILITY

The number of stages and queues and their structure will not disrupt the function of the simulation. This includes parallel queues and multiple start and end stages. The addition of general method of configuring the production line would be helpful but otherwise the way the program sorts and updates the factory would not be affected.

The only caveat being that the production lines structure is hardcoded into the factory class. This includes the number of queues and stages and their relationship to one another. Furthermore, the statistics class is specific to each structure. A user interface could be created to make it easier to configure the structure or subclasses created that inherit from factory but whose production lines have specific structures. Had the program being designed to simulate lots of different structures the preferred method would be a user interface that specifies the structure prior to simulation.

If the program had to take two different types of items and perform an operation on them then the design would have been altered to make certain things generic. Presently the production line caters only for Items. Restructuring the production would not be so difficult but catering for multiple item types and performing operations on them would be somewhat more involved due to the original design.