Exercise: Bob's Grocery Mart

Bob's Grocery Mart has 73 locations distributed across 12 States. Bob would like to increase the profitability of his stores by improving cashier scheduling. If too few cashiers are scheduled, customers become frustrated and may not return to the store. With too many cashiers, Bob pays more wages than necessary. You have been tasked with writing a checkout simulation program that will be used to analyze the impact of increasing or decreasing the number of cashiers.

Specification

Note: You will only be designing—not implementing—this specification today.

Your program should simulate the activity of any number of staffed check-out aisles in a Bob's Grocery Mart. The simulation will proceed in one-second time steps. It takes between one and five seconds for a cashier to process a single object. During each one-second interval there is a 0.5% chance that a new customer will finish shopping and enter a checkout aisle.

Customers purchase between 1–100 items. If there are any empty aisles, customers choose one arbitrarily. If all aisles are occupied, customers choose the aisle with the shortest line.

The program should simulate a single eight-hour shift. At the end of the simulation, the program should display:

- The total number of customers served.
- The average wait time across all customers.
- The maximum wait time for any individual customer.

Design Steps (30 min) **Start time:**

a) Highlight all of the nouns from the program specification above. Circle entries that are good candidates for classes, and cross out entries that are duplicates or do not need to be represented in the program. Any remaining highlighted words should represent entities that need to be represented in the program but should not be classes.

b) On the next page, draw a largish UML box for each of the classes that you selected in the previous step. In the upper part of the box, list an appropriate set of attributes along with their types. Be sure to think about *persistent values* that constitute the state of an object vs. *computed values* that make more sense as local variables.

c) In the lower part of the box, list the necessary constructors for your classes, along with the required arguments. Does it make sense to have a default constructor? Are there classes that may need more than one constructor?

d) For each class, define appropriate methods including getters and setters, `equals` and `toString`, and others. Think about the parameters and return types for each method. Should all methods be public, or are there some that are needed only within the class?