Summery Documentation

The main threads in my program include the customer threads, announcer thread, information desk, and the agent threads. For the program, I have split up my program into five classes: informationDesk, announcer, agent, customer, and departmentOfMoterVehicles. The departmentOfMoterVehicles class hosts all the semaphores, queues, and the main method. When creating the threads, I create an object of each class and the constructor for the object starts the threads.

The semaphores hosted in the **departmentOfMoterVehicles class** are initialized as public static class attributes, so that all the objects created have access to them. The class also has a count variable that keeps track of the number of customers in the DMV with a number, and a capacity variable that hosts the max number of customers that can enter the DMV.

- In the departmentOfMoterVehicles class:
    1. Declare and initialize all the semaphores needed for the simulation
    2. Create the informationDesk object
    3. Create the announcer object
    4. Create two agent objects
    5. Create twenty customer objects
    6. While the number of customer threads joined is less than the capacity, the simulation will continue to run
    7. When the number of customer threads joined is equal to the capacity, the simulation will end

The **customer class** two main attributes, the customerID and the assignedNumber. The customerID is assigned to the object on creation.

- In the customer class:

- In the constructor:
    1. The customerID is set
    2. The customer thread is initialized and started
- When the thread starts to run:
    1. Wait for the customersWaitingForANumberMutex to be available
    2. Customer is added to the customersWaitingForANumber queue
    3. Release the customersWaitingForANumberMutex
    4. Wait for the information desk to be available
    5. Print out that the customer has entered the DMV
    6. Signal that the customers ready for a number
    7. Signal that the information desk is open for another customer
    8. Signal that customer ready for an agent

The **informationDesk class** one main attribute, the ID which is assigned to the object on creation.

- In the informationDesk class:
    - In the constructor:
        1. The ID is set
        2. The informationDesk thread is initialized, daemon is set to true, and is started
    - When the thread starts to run while the number of people that entered the DMV is less than the capacity:
        1. Wait until a customer is ready for a number
        2. Wait for the customersInWaitingAreaMutex to be available
        3. Wait for the customersWaitingForANumberMutex to be available
        4. The customer is assigned a number, and the assignedNumber attribute is set
        5. Printing out that the customer has received a number

6. Add the customer to the customersInWaitingArea queue
7. Remove the customer from the customersWaitingForANumber queue
8. Release the customersWaitingForANumberMutex
9. Release the customersInWaitingAreaMutex
10. Signal that the customer has been assigned a number

The **announcer class** one main attribute, the ID which is assigned to the object on creation.

- In the announcer class:
  - In the constructor:
    1. The ID is set
    2. The announcer thread is initialized, daemon is set to true, and is started
  - When the thread starts to run:
    1. Waits for the agentLine to be available
    2. Waits for a customer to be ready for an agent
    3. Waits for the customersInWaitingAreaMutex to be available
    4. Waits for the customersWaitingForAnAgentMutex to be available
    5. Prints out that the announcer has announced a number, and that the customer has been moved to the agent line
    6. Adds the customer to the customersWaitingForAnAgent queue
    7. Removes the customer from the customersInWaitingArea queue
    8. Releases the customersWaitingForAnAgentMutex
    9. Releases the customersInWaitingAreaMutex
    10. Waits for an agent to be available
    11. Signals that the agentLine has been decremented
    12. Signals that the customer is ready for service
    13. Signals the agent

The **agent class** two main attributes, the ID which is assigned to the object on creation, and a customer object which is used during the thread execution.

- In the agent class:
    - In the constructor:
        1. The ID is set
        2. The agent thread is initialized, daemon is set to true, and is started
    - When the thread starts to run:
        1. Waits for a customer to be ready for service
        2. Waits for the customersWaitingForAnAgentMutex to be available
        3. Set the customer object equal to the head of the customersWaitingForAnAgent queue
        4. Remove the customer from the customersWaitingForAnAgent queue
        5. Print that the agent is serving the customer
        6. Releases the customersWaitingForAnAgentMutex
        7. Prints out the services
        8. Join the customer thread
        9. Increment the customersJoined variable in the DMV class

One of the difficulties that I faced while programing the simulation is nullPointerExecptions. I was receiving a nullPointerExecption error from the information desk class once the last person has entered the DMV and was assigned a number. The way I overcame this issue is by restricting the thread to run while the number people in the DMV is less than or equal to the capacity. Once all the people have entered the DMV, the informationDesk thread will join and stop executing.

Another difficulty I faced was not setting the Daemon for the threads to true. This caused the threads to complete executing after serving one round of customers. The obvious solution to this was to setDaemon to true, and adding a while true loop to the thread execution logic. This way, the agent threads and the announcer threads stop executing when the all the customers have joined and the main method calls the exit function.