CS 316 Project 5: Call Center Simulation

When it is due:

Monday April 29th, at 11:59pm.

What to submit:

1. Please submit the completed **CallCenter.java** file.

Objectives:

1. Use the Java Executor framework to create multi-threaded applications.
2. Demonstrate understanding of the concepts of critical sections and synchronization.
3. Use synchronization primitives to properly synchronize multiple threads.

Instructions:

1 In this project you will implement a multi-threaded Java program that simulates a call center. The Java program has these four classes:

   a. the **Agent** class that simulates a customer service agent who answers customer calls.

      i. Each agent has a unique ID.

   b. the **Greeter** class that simulates the automated greeting service.

   c. the **Customer** class that simulates a customer call.

      i. Each customer has a unique ID.

   d. the **CallCenter** class that drives the overall simulation.

2 The simulation uses two separate queues: a wait queue and a dispatch queue.

3 Upon arrival a customer call places itself in the *wait queue.*

   a. Each customer call should be implemented as a separate thread

4 The Greeter thread removes a customer from the *wait queue*, greets the customer, and then places the customer into the *dispatch queue*; it repeats this process.

   a. The greeter must announce the customer's place in the *dispatch queue* when greeting the customer -- see the attached sample output for an example.

   b. The greeter thread should exit after it has served *NUMBER_OF_CUSTOMERS* customers.

5 An agent removes a customer from the *dispatch queue* and serves the customer; it repeats this process.

   a. Each agent should be implemented as a separate thread.

b. An agent must call the provided *serve* method to serve a customer.

c. An agent should exit after it has served *CUSTOMERS_PER_AGENT* customers.

Sample Output:

1. It is OK if the messages are printed in different order (There is some randomness in this simulation). Just make sure that:
    a. Each agent serves the exact number of customers as specified by *CUSTOMERS_PER_AGENT*.
    b. Each customer is served by a single agent.
    c. The program exits and does not hang. (Program hanging is usually a symptom of synchronization issues.)

```
Greeting customer 1: your place in queue is 1
Agent 1 is serving customer 1
Greeting customer 2: your place in queue is 1
Agent 2 is serving customer 2
Greeting customer 3: your place in queue is 1
Agent 3 is serving customer 3
Greeting customer 4: your place in queue is 1
Greeting customer 5: your place in queue is 2
Greeting customer 6: your place in queue is 3
Greeting customer 7: your place in queue is 4
Greeting customer 8: your place in queue is 5
Agent 3 is serving customer 4
Greeting customer 9: your place in queue is 5
Greeting customer 10: your place in queue is 6
Agent 2 is serving customer 5
Greeting customer 11: your place in queue is 6
Agent 1 is serving customer 6
Greeting customer 12: your place in queue is 6
Greeting customer 13: your place in queue is 7
Agent 1 is serving customer 7
Greeting customer 14: your place in queue is 7
Greeting customer 15: your place in queue is 8
Agent 3 is serving customer 8
Agent 2 is serving customer 9
Agent 1 is serving customer 10
Agent 2 is serving customer 11
Agent 1 is serving customer 12
Agent 3 is serving customer 13
Agent 2 is serving customer 14
Agent 3 is serving customer 15
```

Grading:

This project has 100 points. Each of the four Java classes (**CallCenter**, **Agent**, **Greeter**, and **Customer**) carries 25 points.