**Course Project**

**DeVry University**

**College of Engineering and Information Sciences**

**Course Number: CEIS295**

# Module 3: Real-World Application Demo

# Objectives

* To learn about Call Routing Systems, also called “Automatic Call Distributor” (ACD)
* To use a Queue data structure in a real-life application scenario
* To learn about simulations

We cannot compare the Queue data structure speeds since it is based on a LinkedList and, therefore, would have the same speeds as a LinkedList. Instead, let’s look at a real-life scenario for the Queue data structure.

In a Call Center, the call is received by the automated system, which gets the person’s name and phone number. The call is then placed in a “call waiting” queue. We have all experienced the “call waiting” queue – “Thank you for holding. You are seventh in line…”

How does the “call waiting” queue actually work? We are going to create an application to simulate an Automatic Call Distributor (ACD). ACD’s are used in call centers to route incoming calls to available agents. The application should:

* Handle calls (each call is an object of the type Call)
* Call objects should have:
  + Client’s id
  + Caller’s name
  + Caller’s phone number
  + Date of the call
  + Time of the call
* Every second, the application should randomly complete one of three tasks
  + Receive a customer’s call and assign the call to the “call waiting” queue
  + Remove a waiting call as if the call is being sent to a service agent
  + Do nothing (no calls and no routing during this second of time)

# Steps

1. Create a new folder in your CEIS295 folder called “Week 3 Project”. Download the CallsData.csv file and place the file in your “Week 3 Project” folder. We will read the data in this file so we do not have to input the call information into our simulation.
2. Copy the Node.py file, LinkedList.py, and Queue.py files to this folder. We will use the Queue code to enqueue and dequeue the calls in our application. The Queue is the perfect data structure for this scenario since we can add new calls to the back of the line and process the calls from the front of the line.
3. In this same folder, create a Call class based on the following UML diagram. The \_\_str\_\_ method should return the Call information in this format:  
    client\_id, \_\_first\_name, and \_\_last\_name in this format: 100231, Black, Jack  
   Also, make sure your name and the current date are listed at the top of the code.  
     
   
4. In this same folder, create a file called AutomaticCallDistributor.py. Type your name and the current date at the top of the code. Then, import the Queue class, the Call class, the date module from the datetime library, the time module, and the random module.
5. In the same AutomaticCallDistributor.py file, display your name and the current date in the output to show that you are the author of this code.
6. In the same AutomaticCallDistributor.py file, create a list and read the records from the CallsData.csv file into Call objects and place the Call objects into the list.
7. In the same AutomaticCallDistributor code file, create an Queue object and call it calls\_waiting. Then, create a call\_number variable and set it to zero
8. Ask the user for the number of seconds to run the simulation. Create a for loop that will run one time for each second that was inputted. In the for loop, take these steps:
   1. Pause the application for two seconds using time.sleep(2)
   2. Generate a random number from 1 to 3
   3. If the random number is 1, add a call to the call\_waiting queue and then increment the call\_number variable. Then, show how many calls are currently in the call\_waiting queue.
   4. Otherwise, if the randomly generated number is 2, remove a call from the call\_waiting queue and tell the user that the call is being routed to a service representative. Display the call’s information. Then, show how many calls are currently in the call\_waiting queue.
   5. Otherwise, do nothing to the queue. Tell the user that nothing happened during this second of time. Then, show how many calls are currently in the call\_waiting queue.
9. Run the application several times so you can understand how these ADT’s actually work. Feel free to change the amount of time to pause – less time to pause to run faster and more time to pause to be able to read the events that are happening. The argument for the time.sleep() command is the number of second to pause.

# Deliverables Part 3

* Complete the Module 3 Course Project Presentation deliverable
  + Call.py code
  + AutomaticCallDistributor.py code
  + Screenshot showing the code running with your name and date in the output