SA421 – Simulation Modeling
Assoc. Profs. D. Phillips and N. Uhan

Fall 2017

Assignment 2

Instructions. Work with a partner on this assignment.

Suppose the Nimitz Coffee Bar is more complex than the version we encountered in Lesson 3. If a customer orders drip coffee, they do not need a barista. Instead, the cashier gives the customer an appropriately sized cup, and the customer goes to a separate queue to wait for a self-serve dispenser to pour their own coffee. In this assignment, you will model this additional feature.

Make the following assumptions:

- 40% of customers order drip coffee;
- Nimitz Coffee Bar has one drip coffee dispenser which we assume does not run out of coffee;
- The time to pour coffee is best modeled by an exponential random variable with mean 40 seconds.

Start by downloading the base JaamSim model (i.e. the model we created in Lesson 3) from here:

```
https://github.com/sa421-usna/assignment-02/zipball/master
```

Implementing the drip coffee customers can be done in the following way in JaamSim:

1. Model the process of customers using the self-service dispenser. Use the Queue, Seize, EntityDelay, Release, Resource, and ExponentialDistribution objects to model this part of the cafe.
2. Divide the flow of customers from the cashier. This can be accomplished by using a **Branch** object after the Cashier. A Branch object directs an incoming entity to a destination that is chosen from a list of alternatives. Some key inputs for Branch objects:

Keyword	What is this?
NextComponentList	A list of possible objects to which the incoming entity can be passed
Choice	A number that determines the choice of the next component: 1 = first in NextComponentList, 2 = second in NextComponentList, etc.

   Set up the Branch object by putting the BaristaQ and Queue for the self-serve dispenser in the NextComponentList. For now, set the Choice field to 1 or 2. Run the simulation and see what happens.

3. Model the customer type. We will accomplish this using **attributes**. Attributes can be defined for each generated entity in a model – e.g. each arriving customer. (They can also be defined for objects.)

   In the Customer object, set up an Attribute called DrinkType with a default value of 1 by setting AttributeDefinitionList to

   ```
   DrinkType 1
   ```

   Immediately after a Customer enters, we will randomly assign it a type, 1 or 2. To do this, first place a DiscreteDistribution object called DrinkDist in your model, and set ValueList in this object to

```
1 2
```

and ProbabilityList to

```
.6 .4
```

This object will generate a value 1 with probability .6 and 2 with probability .4. To obtain a random sample from this distribution elsewhere in our model, we can write

```
[DrinkDist].Value
```

as we'll see below.

Next, place an **Assign** object between the Entrance and Cashier objects. The Assign object assigns one or more Attribute values to an incoming entity. Some key inputs for Assign objects:

Keyword	What is this?
NextComponent	This is where the incoming entity goes next
AttributeDefinitionList	A list of attribute assignments

An attribute assignment looks like this:

```
{ 'this.obj.DrinkType = [DrinkDist].Value' }
```

In the attribute assignment above, `this.obj` refers to the incoming entity. Here, the incoming entity's DrinkType attribute is set to a random sample of DrinkDist.

Finally, in the Branch object you placed right after the cashier, change the value of Choice so that 40% of the customers go to the self-serve dispenser.

4. The Graph object in the base model tracks average delay in the BaristaQ Queue. Study this Graph object, and add a Graph that tracks average delay in the Queue object you set up for the self-serve dispenser.