CS003B
Erick Bravo
Design Document P8 LiftShare

Step 1: Find out which methods you are asked to supply.
create an lift share app where it will simulate a 1000 runs of a car picking up a user and dropping them off

Step 2: Specify the public interface.
public class uberride
Public passenger
Public station getDestination()
Public station getstartstation()
Public int pay fare
Public void generateCars()
Public void dropPassemgnger()
Public void loadunloadpassengers()

Step 3: Document the public interface.
Public class uberride

Step 4: Determine instance variables.

```
    private Station destination;
    private Station startStation;
    private ArrayList<Station> stations;
    private int currentStationIndex;
    private int miles;
    private int revenue;

   private Station destination;
   private Station currentStation;
   private Station startStation;
   private int currentPassengers;
   private static final int capacity = 3;
   private ArrayList<Passenger> passengers;
   private int revenueCollected;

    private ArrayList<Uber> cars;
    private ArrayList<Passenger> passengers;
    private int totalRevenueGenerated;
    private int totalMiles;
    private final int stationNumber;
```

Step 5: Implement constructors and methods.

```
   // drops off the passenger once destination is arrived
```

```
public void dropPassenger()
{
   // if the cars empty then leave
   if (passengers.isEmpty())
   {
      return;
   }

   // dumps remaining passengers untill the cars empty
   for (int idx = 0; idx < passengers.size(); idx++)
   {
      Passenger currentPassenger = passengers.get(idx);

      if (currentPassenger.checkDestinationReached(currentStation))
      {
         revenueCollected += currentPassenger.payFare();
         passengers.remove(idx--);
         currentPassengers--;
      }
   }
}

@Override
public String toString()
{
   StringBuilder sb = new StringBuilder();
   sb.append("Start Station: "+startStation+"; Current Station: "
         +currentStation+"; Destination: "+destination+"; Current Number "
               + "of Passengers: "+currentPassengers+"; Revenue collected"
         +revenueCollected);
   return sb.toString();
}

 public void collectRevenue()
 {
    // runs for how many people were inside the car and adds to the totals
    for (int idx = 0; idx < cars.size(); idx++)
    {
       Uber current = cars.get(idx);

       if (current.isDestinationReached())
       {
          totalRevenueGenerated += current.getRevenue();
          totalMiles+= current.getMilesDriven();
          cars.remove(idx--);
       }
    }
 }

// loads users and then dumps them out once a place is reached
 public void loadUnloadPassengers()
 {
    // runs a loop to see how many passengers are iniside
    for (int idx = 0; idx < cars.size(); idx++)
```

```
    {
        cars.get(idx).dropPassenger();
    }

    // if empty returns the car
    if (passengers.isEmpty())
    {
        return;
    }

    // if there are users, checks to see if its full
    for (int idx = 0; idx < cars.size(); idx++)
    {
        Uber current = cars.get(idx);

        if (!current.spaceAvalible())
        {
            continue;
        }

        int requiredPassengers = 3 - current.getCurrentPassengers();

        // runs a loop to see the users going from and to destinations
        for (int jdx = 0; jdx < requiredPassengers; jdx++)
        {
            for (int kdx = 0; kdx < passengers.size(); kdx++)
            {
                if (passengers.get(kdx).getDestination().getStationNumber()
                        <= current.getDestination().getStationNumber())
                        {
                            current.pickPassenger(passengers.get(kdx));
                            passengers.remove(kdx--);
                            break;
                        }
            }

            // if there is space avalible or if there is all the space
            if (!current.spaceAvalible() || passengers.isEmpty())
            {
                break;
            }

            if (passengers.isEmpty())
            {
                break;
            }
        }

  }
  }
```

Step 6: Test your class.

```
/*
--- exec-maven-plugin:1.5.0:exec (default-cli) @ FakeUber ---
Average revenue per per mile in 1000 situations is:
0.0
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.819 s
Finished at: 2020-07-09T20:25:52-07:00
------------------------------------------------------------------------
*/
```