Step 5- Working with the Application

Now that we've built our Bus Depot to house our various buses, we can create some methods that might be part of an application class. We'll call our application class BusDepotApp.The following are methods that need to be implemented:

1. The method countMaxAbove will be given an ArrayList of BusDepot objects and a number of passengers. It should return the number of depots with maxPassengers that are above this number.

   - set up a forEach for the BusDepot list
   - for each BusDepot in the list:
     - if the BusDepot's maxPassengers is greater than the input val, increment a counter
   - return the counter result

2. The method findBus will be given an ArrayList of Buses and a search Bus object (Just an empty Bus object with the Licence Number filled in). This method should return a reference to the Bus with the corresponding licence number or null if no such bus exists.

   - set up a forEach for the list of Buses
   - check if each Bus in the list is equal to the Bus parameter object
   - as soon as a match is found, return a reference to the matching Bus(not the parameter)

3. The method countStowable is given an ArrayList of Buses. It should return a count of buses that are Stowable classified.

   - set up a forEach for the list of Buses
   - for each Bus, check if its an instance of Stowable
   - if so, check its canStow method and increment your counter
   - return the counter of Stowable buses

4. The gatherOrphans method will be given an ArrayList of buses and an ArrayList of depots. This method shall return a list of all buses that cannot park at any of the depots (as detailed in Step 4).

   - create an empty list to store your result, called OrphanList
   - you might approach the processing in two ways:
     for each Depot, check the buses against it
     for each Bus, check the depots against it
     - because we need to know if a Bus can park at any depot, we only add a Bus to our result list if the Bus has been checked agains ALL depots. This corresponds to the second option above so that's how we'll need to proceed
   - set up a forEach for the list of Buses.

- for each Bus, set up a forEach on the list of Depots.
  check each Depot by calling the Depot's canEnter method with the current bus as parameter
  if a bus canEnter on any Depot, skip to the next bus
  once this forEach has completed for a current Bus, that means the Bus cant enter any Depot and should be added to the OrphanList
- return the resulting OrphanList once both loops are complete

5. Create a setFeatures method to create features arrays for Greyhound instantiation.

   - set up your method to return a features array/ArrayList
   - create a blank features array/ArrayList.
   - you need to decide on a simple way to parameterize how to set which of the three possible features is to be added. One way is to take three booleans as parameters, corresponding to the three features, say (boolean movies,boolean gaming,boolean music) and do a simple if/else structure to check them and insert into the Array/ArrayList for the ones that are true.
   - once you've made any necessary insertions, return the resulting feature array

6. Since our buses will have passengers, we could add an arraylist to the buses to store a passenger list. What would you need to create and implement to do this. What are some of the potential implications of this?
   - Theres lots that could be done with this. The list could be a list of Strings representing the passengers names or we could model Passenger objects and implement a list of those objects.
   - method considerations: if we create a Passenger object we might need to have things like purchaseTicket, checkTicket, ... which then begs the question: do we need to model Tickets?
   - the above also has implications for the application. Do we need to have something to add Passengers to Buses? For ticket processing?