Combined Programming Assignment 1 & 2: Disaster Evacuation Coordination

For this assignment, you will be building a class, creating objects, doing comparisons and doing some calculations. You need to remember what you have learned in class, lab, books and your assignments. Be sure to refer to them when you need to.

For this assignment, you will use the following:

- Classes and Objects
- Arrays
- Inheritance
- Interfaces
- Exceptions
- Sorting Algorithms

There are 2 parts to this assignment. In the first part, you are going to be given a problem and you will then need to create a class structure, flow chart(s) and write algorithms in pseudocode to solve it. In the second part, you'll be translating your project plan into a java program.

So let's get started!

Part 1: Helping Evacuate the Cute Fuzzy Animals!

You are always striving to help those in need, especially if they are cute fuzzy animals. You've noticed that when a disaster strikes, trying to coordinate an evacuation is pretty chaotic. So you've decided to come to the rescue! You are going to create a program that helps organize and schedule evacuation caravans from a disaster area for pets. To start off, your program will focus on caravans that are carrying cats, dogs and horses (your favorite animals!)

The general flow is as follows:

- The volunteer coordinator (user) signs in (just ask their name) and is greeted by the program.
- A vehicle pulls up so that it can be assigned to 1 of 3 caravans available each day (The vehicle type and animals should be randomly generated).
- The coordinator views the available caravans and assigns the vehicle to one of the caravans.
- The coordinator can then either
    - See the next vehicle
    - Create a new caravan (only up to 3 per day—must include exception handling for this)
    - Send 1 of the caravans on 1 of 3 evacuation routes

    - If this option is selected, then a line should be printed with the sorted caravan, the route taken (with its expected % of failures) and whether the caravan succeeds or fails to reach its destination
  - Quit the program

Constraints:

- Each caravan can have ***up to*** 10 vehicles
- Can have ***up to*** 3 caravans a day
- Right before a caravan is sent on its route, it is sorted from smallest to largest vehicle
- Once a route is used by a caravan, it cannot be used again until the next day
- Each route will have a % chance that the caravan will not make it to its destination and will have to turn back.
  - One of your routes should have a 20% chance of failure.
  - You (programmer) get to choose the % failure rate for the other two caravans
- If the coordinator attempts to add a vehicle to a caravan that already has 10 vehicles, the program must throw a **custom exception** indicating the problem and rectifying the situation (have the coordinator choose another caravan)

Needed Information:

- 4 different size vehicles & number of animals that they carry
  - Compact car (2 animals)
  - Midsize car (5 animals)
  - SUV (10 animals)
  - Truck (3 animals)
- 2 different types of vehicles
  - Can carry only cats and dogs
  - Can pull a large animal trailer (SUV and Truck only)
  - For the SUV and Truck, you must include the following information
    - Whether it has a trailer
    - Trailer length
    - Trailer capacity (2 horses or 4 horses)
- For each animal, we need to know the following
  - Name
  - Weight in pounds
  - ID number
  - Owner name
- For cats, we need to know
  - Whether it has a litter box
  - Last time the litter box was cleaned (Morning, Afternoon, evening, night, previous day
- For dogs, we need to know

  - Whether it has a leash
  - Last potty break (morning, afternoon, evening, night)
- For horses, we need to know
  - Whether it has hay

When the user quits the game, any existing caravans that have not yet left for that day must leave. You must then print out a final summary of their session, including:

- Volunteer Coordinator's name
- Number of days in operation
- Total number of caravans
- Number and percent of successful trips
- Number and percent of failed trips (where the caravan had to turn back)
- Total number of animals, broken down by animal type (cat, dog, horse)
- Number of animals saved, broken down by animal type
- Number of each size vehicle
- Number of trailers, broken down by trailer size (2 or 4 horse trailer)
- User classification:
  - Beginner (<6 successful trips)
  - Amateur (7-12 successful trips)
  - Intermediate (13-18 successful trips)
  - Advanced (19-24 successful trips)
  - Expert (25+ successful trips)

For Part 1, create the class structures and algorithms for your program, and then do several iterations of tests (i.e., analyze it and step through to make sure that it is logically correct). Also write the pseudocode for your tester class (where your main will go). Put these in a Word or Open Office document. You'll turn that document in with the program that you create in Part 2.

Important! As you are working on this, be sure to break this down into smaller pieces. Take it step-by-step, and don't try to finish this in one sitting. It will make it MUST easier.

Part 2: Creating your Butterfly Colony program

Once you are done writing and testing your class structure and algorithm, you are ready to start coding!

1. Once again, you first you need to create a project. Here's a nice tutorial on how to do that in Netbeans. If you are using Dr. Java or Eclipse, just do a quick search on youtube.com and you'll find lots of candidates.

   http://www.youtube.com/watch?v=ezUHG1cuxkM

Be sure to give your project a *nice, meaningful name* (and make sure it adheres to Java's naming conventions).

2. Once you have your shell ready, there are a few things to know before you start translating your algorithm into code
   - At the top of your class file, be sure to include the following:

```
//*****************************************************************************
// Name:  [Your Name]
// FIU email: [Your FIU email]
// PantherID:  [Your PantherID]
// CLASS: COP 3337 – [Semester Year]
// ASSIGNMENT # [#]
// DATE: [Date]
//
// I hereby swear and affirm that this work is solely my own, and not the work
// or the derivative of the work of someone else.
//*****************************************************************************
```

   - Remember your random number generator? You'll need to use some for this assignment:

     i. Include the following code at the top of your class file (so that you can use this class:

     ```
     import java.util.Random;
     ```

     To find out more about this, go to http://java.sun.com/javase/7/docs/api/index.html (like you did in Lab Assignment 2)

     ii. You'll need to use some variables. Here's how you get a random number:

     ```
     Random r = new Random();
     int x = 1 + r.nextInt(10);
     ```

     Upper limit of the random number generated (exclusive)

     Note that the number in the parens (e.g., 10 above) is the upper limit of the random number, exclusively. So, the random number that you get here will be an integer between 1 and 10, once you add the 1. Need a larger range? Just change the 10 to the top of your range.

     Here's another example, in this case if you are printing a random number to the console:

```
System.out.print( 1 + r.nextInt(5) + " " );
```

3. Now start translating your algorithm into java code.
    - Remember to code and then compile frequently. It will make it easier to find any bugs.
    - Also remember that you will need to create at least one **separate class** (where your main method will reside).

4. Once you get your program running correctly, there is one more thing to do. *Any input requested from the user and/or output received from the user should be in a window* (see E.1.14 and E.1.15 from lab 1). At this point, you probably have your output going to the console. For your final submission, your output can go to the console or to a window (JOptionPane). Don't forget any additional libraries that you need to import to do this.

That's it! *Now you can help get all those cute fuzzy animals to safety*! Of course, you'll also need to turn it in to Canvas.

Submission Requirements
You must upload a zip file to Canvas that includes your complete source project in Netbeans, ready to load, and also contains the output in separate data files, and your Word/Open Office document with your algorithm.

VERY IMPORTANT: If you do not provide output in separate, easy to find data files, I will assume that your program does not work on those test cases, and grade accordingly. *Do not embed the output in your source code.*