Lab18: Vehicle Class

Part 1: Vehicle.communicate()

Description: We will examine how vehicles generally communicate with each other, or at least how people attempt to communicate with each other in their vehicles. Every class should have 1 no argument constructor and 1 useful constructor.

Create a class called Vehicle. It should contain the number of seats, the current number of passengers, currentPosition as a Point, destination as a Point, top speed in mph (int), and a communicate() method. In any vehicle, the driver can attempt to communicate with other vehicles by shouting at them, "Hey you!" (we will print this message to the screen). Vehicles should contain a toString() method that returns "This is a vehicle."

Create a class called Point. It should contain x/y coordinates as doubles. It should also have the following methods:

```
  public String toString(); // Returns "(1,2)"
  public double distance(Point p); // Returns distance between this point and p.
```

Create a class called Airplane. It should inherit from Vehicle. Airplanes should additionally store a flight number, such as flight "A101". Instead of shouting "Hey you!" the airplane should attempt to communicate with other vehicles by radioing the tower the message, "Tower, this is flight A101. Requesting permission to land." Airplanes should contain a toString() method that returns "This is an airplane, flight number A101."

Create a class called Car. It should inherit from Vehicle. Cars should additionally store the number of doors and engine/motor type, "4 cylinder", "V6", "V8", and "Electric". Cars should communicate with others by printing "Honk honk!" Cars should contain a toString() method that returns "This is a V6 car with 4 doors." Proper grammar in toString().

Create a class called Train. It should inherit from Vehicle. Trains should additionally store the number of boxcars (not Cars). Trains should communicate with others by printing "I think I can, I think I can!" Trains should contain a toString() method that returns "This is a train, all aboard!"

Part 2: speedCompareTo()
Description: Each vehicle has an integer speed in mph. Create a Vehicle method speedCompareTo(Vehicle v) that returns an int representing the difference in speed. If this Vehicle is faster, the difference in speed should be positive, and if this Vehicle is slower, the difference should be negative. If the Vehicles have the same speed, the difference should be 0.

Part 3: Create a distanceTo() method in the Vehicle class that returns the distance to another Vehicle.