EXERCISE 2, MORE ON STATICS

Exercise 2, More on statics

Objective

In this lab you will implement the factory pattern. You will get a 'Registration Plate' from a 'Registration Plate Factory' class.

Step by step

In this section of the lab you will beef up a `Vehicle` class to add both a static member and an instance member and use static members of a `Factory` class.

1. Create a new Java project called Labs
2. Create a Package called lab02.
3. Create a class in this package called Program with a *main()* method.
4. Create a class called Vehicle with integer fields called speed and lane.
   Also record the distance travelled using an int variable called distanceTravelled.
5. Create a constructor to set the speed and lane fields.
6. Add two methods called **`accelerate`** and **`brake`** methods.
   void accelerate(int amount)
   The accelerate method will increase the speed but never more than 200! It also adds to the distanceTravelled.
   void brake(int amount)
   // to set speed=0
7. Add another method as String getDetails() to get the vehicle's details.(speed, lane, distanceTravelled and plate *(see below)*)
8. Every vehicle has a registration plate which is a complex objects (has info about City, country and the year of registration) which should be defined as a class.
   a. Create a class separate called ***RegistrationPlate***.
   b. Give this class a field called ***regPlate***. Provide a getter method to read the value of this field.
   c. Create a constructor to set the *regPlate field*.
   d. Create a getter method for *regPlate*.
9. Add a new field to the Vehicle class called *registrationPlate* of type *RegistrationPlate*.
   Tip: ***RegistrationPlate*** *registrationPlate;*

   You'll set this field using a factory pattern.
   Please do not instantiate it here.

10. Let's create a factory class which creates instances of *RegistrationPlate*.
11. Create a separate class called ***RegistrationPlateFactory***
12. Add the following array of reg numbers to *RegistrationPlateFactory*.

```
private static String[] regPlates =
 { "MRB1G", "RU16", "TOYS4US", "HNZ57", "PUT3", "JB007" };
```

    Note, The array is static because it will be accessed by an static method.

13. Crate a static method called getNextRegistrationPlate()
    This method should return an instance of *RegistrationPlate*.
    To create an instance, you'll need the next registration plate from the regPlates array. How would you get the next regPlate index?
14. Back in the Vehicle class's constructor, assign a registration plate to the Vehicle using the *RegistrationPlateFactory* class.
    Tip:
    registrationPlate = RegistrationPlateFactory. getNextRegistrationPlate();
15. In the main() method, Create a `ArrayList` of Vehicles, populated with three new Vehicles.
16. Print details of the vehicles created in the above step.
    Please make sure the plates are correctly assigned.

Enhancing the Vehicle class

17. How would you count the number of vehicles created?
    I know you created 3! But what if different parts of your program created Vehicles?

    Create a static method called getCount() to return the count of vehicles.

    Tip: Vehicle's constructor is invoked whenever a vehicle is created. Please see the slides for more help.

Writing code to use the Vehicles

18. Return to the main method.

    a) The initial speed of the Vehicles is zero.

    b) Make sure they are placed in lanes 1, 2, 3

19. Write a while loop to race the cars (accelerate them)
    until the distance travelled by one of them is more than 1000.

    You'll accelerate each vehicle by a random number between 1-10 by using the following code:

    Random rand = new Random();

```
int n = rand.nextInt(10)+1;
```

20. Display details of each vehicle as they accelerate on each iteration of the while loop.
21. As soon as a vehicle has travelled 1000m or more, announce it as winner and break out of the loop.
22. You can get creative and assign drivers to each vehicle.

** End **