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 Console application called Lab02
2. Open the Program class with the *Main()* method.
3. Create a class called Vehicle with integer fields called speed and lane.
   Also record the distance travelled using a readonly int property called distanceTravelled.
   Tip: create an auto_implemented property with private set like: public int DistanceTravelled {get; private set;}
4. Create a constructor to set the speed and lane fields.
5. 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
6. Add another method to get the Car's details
   (speed, lane, distanceTravelled, plate *(see below)*)
   string getDetails()
7. 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 separate class called *RegistrationPlate*.
   b. Give this class a private *string* fields called *regPlate*.
   c. Create a "property get" for *regPlate*. You can call this property RegPlate
8. Create a constructor to set the *regPlate* field.
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*.

    Create a separate class called ***RegistrationPlateFactory***

11. Add the following array of reg numbers to *RegistrationPlateFactory*.

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

12. Crate a static method called GetNextRegistrationPlate()
    This method will 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?

13. Back in the Vehicle class's constructor, assign a registration plate to the Vehicle using the *RegistrationPlateFactory* class.
    Tip:
    *registrationPlate = **RegistrationPlateFactory**.*GetNextRegistrationPlate();

14. In the Main() method, Create a `List`<Vehicles> of 3 Vehicles.

15. Print details of the three vehicles.
    Please make sure the plates are correctly assigned.

Enhancing the Vehicle class

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

    Tip: Vehicle's constructor is invoked whenever a vehicle is created. View the code on the slides for more help.

Writing code to race the Vehicles

17. 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

18. 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.Next(10)+1;
```

19. Display details of each vehicle as they accelerate on each iteration of the while loop.

20. As soon as a vehicle has travelled 1000m or more, announce it as winner and break out of the loop.
21. You can get creative and assign a driver to each vehicle.

** End **