STATICS

Objective

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

Step-by-step instructions

In this section you will create a `Vehicle` class with a static member and instance members and use static members of a `Factory` class.

1. Create a class called **Program** with a *main()* method in a package called **exercise3**.
2. Create a class called **Vehicle** in the exercise3 package with integer fields called **speed** and **lane**.
   Also record the distance travelled using an int variable called **distanceTravelled**.
3. Create a constructor to set the initial speed and lane fields.
4. Add two methods called **`accelerate`** and **`brake`**.
   void accelerate(**int** amount)

   The accelerate method will increase the speed but never more than **200**!
   It also adds to the **distanceTravelled** (add 1m per unit of speed).

   void brake(**int** amount)
   // to reduce the speed by the amount
5. It is never a good idea to use magic numbers like 200 (for maximum speed). Replace 200 with a constant like `private static final int MIN_SPEED = 0;`
6. Add another method as **String** **getDetails()** to get the vehicle's details such as speed, lane, distanceTravelled and plate (see below).
7. Every vehicle has a registration **plate** which is a complex object (has information about the city, the country, and the year of registration) which should be defined as a class.
   a. Create a class called ***RegistrationPlate***.
   b. Give this class a String field called ***regPlate***. Provide a getter method to read the value of this field.
   c. Create a constructor to set the *regPlate field*.
8. Add a new field to the **Vehicle** class called ***registrationPlate*** of type *RegistrationPlate*.
   Tip: ***RegistrationPlate*** *registrationPlate;*

   You'll soon set this field using the **Factory pattern**.
   Please **do not** instantiate it yet!

Registration plate factory

9. Let's create a new **factory** class that creates instances of ***RegistrationPlate***.
10. Create a separate class called ***RegistrationPlateFactory***.

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

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

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

12. Create 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?

    Return **null** if you run out of regPlates.

13. Back in the Vehicle class's constructor, assign a registration plate to the vehicle using the *RegistrationPlateFactory* class.

    Tip:
    this.registrationPlate = ***RegistrationPlateFactory.*** **getNextRegistrationPlate**();

14. In the main() method, Create a `ArrayList` of Vehicles, populated with three new Vehicles.
15. Print details of the vehicles created in the above step.
    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 three! But what if different parts of your program create 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

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 1,000.

    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;
    ```

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 (create a Driver class).

Part 2 – using Statics with the graphical project you created earlier

In this part you'll make use of the **static** keyword.

Clearly setting the world dimension to 300px (height and width) is not a good idea.

All balls share the same world coordinates. This is a perfect place to use the **static** keyword.

Step by step

1. Create a **static int** field in Ball called **worldW** and another called **worldH** to represent the width and height of the world. You can add the X and Y of the World as well if you like but we assume the world starts at x=0 and y=0.
2. Create static method as
   public static void setWorld(int w, int h)
   to set the static *worldW* and *worldH* values.
3. Just before creating an instance of **Game** in the main() method, make a call to **Ball's setWorld()** like **Ball.setWorld(0,0,400,300)**
4. Change the move() method in the Ball class to use the Ball.worldH and Ball.worldW values instead of 300px. You'll also need to change the paint() method to use the new static values to draw the world's rectangle.
5. Run and test your code.