Refactoring with enums and objects

Continue on last week's Console app with the Ticket Office, by refactoring (that is changing the code in order to improve it) with an enum and an object.

First step

Change the string type for the choices of "Seated" or "Standing" for place preference into an enum. Therefore add a new class of enum type.

As a consequence, the PriceSetter method needs to change its parameter list from accepting a string *place* to accepting an enum *place* instead.

When this refactoring is done, commit the changes to git and push to GitHub.

Second step

Refactor the solution in a major way by adding a new class called *Ticket*.

The class *Ticket* should have a constructor with two parameters, an int *age* and an enum *place*.

Give *Ticket* three properties, with simple getters and setters:
- int *Age*
- enum *Place*
- int *Number*

Move the *PriceSetter* method into the *Ticket* class. Rename the method to *Price*. The *Price* method no longer takes any parameters. Instead it is getting the values from the properties. The *Price* method still returns an int with the price.

Also move the *TaxCalculator* method into the *Ticket* class. Rename the method to *Tax*. The *Tax* method no longer takes any parameters. Instead it is getting the price value by calling the *Price* method. The *Tax* method still returns a decimal with the tax value.

Leave the TicketNumberGenerator method outside of the Ticket class, but call the TicketNumberGenerator method from the constructor for setting the Number property.

Something like this code should be possible to write in the Program.cs (or the class used for the main application):

```
Ticket ticket = new Ticket(24, Place.Seated);
ticket.Age;
ticket.Place;
ticket.Price();
ticket.Tax();
```

When this refactoring is done, commit the changes to git and push to GitHub.

Finish the assignment by Friday