Classes & Objects

In C#, *classes* and *objects* are pivotal concepts in building console, web, and mobile applications. That means what you learn now are the exact same tools that professional developers use to build your favorite applications!

What is a class?

- A class is a template for an object. It defines what the object is made up of (member variables) and what actions the object can perform (member methods)

What is an object?

- An object is an instance of a class. Each instance of a class (object) contains the same member variables and methods. This means each instance of a class (object) has the same make up. What differs between the objects is what values are set equal to the member variables.
- To put this in perspective of what you are more familiar with, every time you run an application on your computer it is running an *instance* of that application. For example, you can have multiple Microsoft Word documents open at once. Each of these Microsoft Word documents may contain different text and fonts, but all of them have the same functionality.

The learning objective of this lab assignment is to get you thinking about classes and objects as real-world entities that you can use. We eventually will get to a point of building impressive web applications. For now, just concentrate on how a class operates and how objects work together.

The assignment: build a console-based version of an eCommerce site (not an actual website!). Think about Amazon and your interactions with the site as you are building your classes and objects. Putting yourself in the shoes of a consumer will allow you to build your application better. Professional developers call this the "user experience" (UX)

The steps:

1. Create a console application. Call it eCommercePlatform
2. Next, let's create the necessary classes we will need to get a minimal functioning eCommerce application. We will need the following classes:
    a. Platform class
        i. This class represents the eCommerce platform where most of the action will take place. It is the most top-level object in your application.
        ii. Member variables:
            1. List<Product> inventory
    b. Consumer class
        i. This class represents the consumer who wants to search products and purchase products. We will need to keep track of their information to better assist them.
        ii. Member variables:
            1. string firstName
            2. string lastName
            3. ShoppingCart shoppingCart
    c. Product class

        i. This class represents the product that is able to be purchased. Each product will have the ability to be rated and reviewed (we use a List data structure so that we can keep track of multiple reviews for each product)
        ii. Member variables:
            1. int uniqueIdentifier
            2. string name
            3. string category
            4. double price
            5. double averageRating
            6. List<Review> reviews
    d. ShoppingCart class
        i. This class represents the consumer's shopping cart. When a consumer decides on a product they want to purchase, the product will be placed in the shopping cart.
        ii. Member Variables
            1. List<Product> products
            2. double totalCostProducts
    e. Review class
        i. This class represents a rating and review for a product.
        ii. Member Variables
            1. double rating
            2. string text
3. Now that we have classes with clearly defined member variables, it is time to create the class methods. Not only is it important for our objects to be able to hold on to state (the values assigned to the variables), but it also is important to give the ability for our objects to perform actions.
4. Platform class
    a. Create a constructor that instantiates the List<Product> inventory
    b. Create a method to create a new Product

c. Create a method to populate the product inventory. Take a look at the below example for inspiration:

```
public class Platform
{
    //member variables
    List<Product> inventory;

    //constructor
    0 references
    public Platform()
    {
        inventory = new List<Product>();
    }

    //methods
    0 references
    private Product CreateProduct(int uniqueIdentifier, string name, string category, double price)
    {
        Product product = new Product(uniqueIdentifier, name, category, price);

        return product;
    }
    0 references
    private void PopulateInventory(Product product)
    {
        //put product in inventory
    }
}
```

i. The inventory is a list of product objects
ii. The constructor is instantiating a new list of products
iii. The CreateProduct method will take in a unique identifier, a product name, a product category, and a product price. These are known as the method parameters, which basically tell the developer what this method will need in order to work properly. When the CreateProduct method is called, it will require arguments (the actual values) of unique identifier, product name, product category, and product price. The CreateProduct method instantiates a new Product, which itself requires a unique identifier, a product name, a product category, and a product price to be created properly. The CreateProduct method then returns the newly created Product object so that it can be used elsewhere in the application. Perhaps the PopulateInventory method can use it?
iv. The PopulateInventory method takes in a Product object as a parameter. This method's responsibility is to put the product into the Platform's inventory.

5. Consumer class
    a. Create a constructor that instantiates the ShoppingCart shoppingCart
    b. Create a method to set the first name of the consumer
    c. Create a method to set the last name of the consumer
    d. Create a method to search for a product by name
    e. Create a method that shows (prints to the console) a product's name, price, and average rating
    f. Create a method to add a product to the shopping cart
    g. Create a method to give a product a rating and review. This method must return the Review object from the method (this will be similar to what you did with the CreateProduct method)
6. Product class

a. Create a constructor that takes in as a parameter a unique identifier, a name, a category, and a price. The constructor must initialize the following member variables as so:

```
public class Product
{
    int uniqueIdentifier;
    string name;
    string category;
    double price;
    double averageRating;
    List<Review> reviews;
    1 reference
    public Product(int uniqueIdentifier, string name, string category, double price)
    {
        this.uniqueIdentifier = uniqueIdentifier;
        this.name = name;
        this.category = category;
        this.price = price;
        averageRating = 0;
        reviews = new List<Review>();
    }
}
```

b. Create a method that takes in a rating as a parameter and updates the averageRating of the product
c. Create a method that takes in a Review object as a parameter and adds the Review object to the list of reviews (this will be similar to what you did with the PopulateInventory method)

7. ShoppingCart class
    a. Create a constructor that instantiates the List<Product> products
    b. Create a method that updates the total cost of the products in the shopping cart
    c. Create a method that removes an item from the shopping cart and decreases that product's price from the total cost of the products
8. Nice work! You have now successfully created classes with member variables and methods.
    a. Each class will be a template for the multiple objects that can be instantiated
    b. Each object will be able to keep track of its own data thanks to the member variables
    c. Each object will have the ability to update its own member variables, the member variables of other objects, perform calculations, and many more functionalities thanks to the methods
9. Lastly, let's test out our code to make sure everything works in the logical order. To do this, think about the workflow of a consumer visiting Amazon and walking away with a shopping cart full of products. The workflow consists of the following:
    a. Search for product
    b. See information about product
    c. Add product to shopping cart
    d. Give product a rating
    e. Give product a review
10. It is up to you to create a method in the Platform called UsePlatform() in which you will call the necessary methods to perform the above workflow. Remember, these methods will be called

from different objects.

```
public void UsePlatform(Consumer consumer)
{
    //consumer workflow

}
```

11. There is so much more we can do with our eCommerce application, such as making purchases. This is known as an application being open-close. The functionality is open to be extended upon while not needing to modify too much of the existing code to make it happen.
12. Great job! Take everything you learned this assignment and continue building on it with future projects.