Assignment-FIRST

On following sections:[due-date-2/28/2024]

**Section-1:Constructor:**

1. **What is the purpose of a constructor in C#?**  Done
2. **What is the typical syntax for declaring a constructor?** Done
3. **What is a copy constructor, and how does it work?** (Answer: A constructor that takes an existing object as a parameter and copies its state to a new object) Done
4. **Imagine you have a class called**Order**that can have multiple**Product**items. How might you use constructors to ensure proper initialization of the**Order**object and its associated products?** Done
5. Create a class named BankAccount with the following properties:

* accountNumber (string)
* balance (double)
* ownerName (string)
* Define a constructor that takes the accountNumber and ownerName as arguments and initializes the corresponding properties. Set the initial balance to 0.
* Create a new BankAccount object using the constructor with appropriate arguments and print the details (account number, owner name, and balance) to the console. Done

**Section-2:properties:**

1. How can you use properties to design classes that follow good object-oriented principles such as encapsulation and information hiding? Done
2. How can you initialize properties with values during object creation, either through constructors or object initializers? Done
3. Explain how you can use custom logic within getters and setters of properties to perform specific actions when accessing or modifying the underlying data. Done
4. Create a class called Person with properties for FirstName and LastName. Implement a simple program that sets and retrieves these properties for an instance of the Person class. Done
5. Develop a class called Book with auto-implemented properties for Title and Author. Create an object of this class and set the properties to represent a book’s details. Done
6. Design a class called Circle with a read-only property/ **Write-Only Property** Radius. Initialize the radius during object creation and calculate the area of the circle using a method.Done
7. **Encapsulation and Validation:**

Create a class named Employee with the following fields:

* firstName (string)
* lastName (string)
* salary (double)

a) Modify the class by converting these fields into properties. Each property should have a getter and a setter. Implement validation logic in the setters:

- `firstName` and `lastName` should only accept non-empty strings. Throw an `ArgumentException` with an appropriate message if an empty string is provided.

- `salary` should only accept positive values. Throw an `ArgumentOutOfRangeException` with an informative message if a non-positive value is provided.

b) Create a new Employee object and try setting invalid values for each property. Observe the exceptions thrown by the validation logic. Done

**Section-3:array and string:**

1. What is the difference between a one-dimensional and a multi-dimensional array in C#? Done
2. What is the difference between for loop and for each loop in C# to access array values? Done
3. What is the Array class in C#? Done
4. How does C# handle string immutability? Done
5. How to search a string in a String array in C#? Done
6. Write a C# program that stores elements in an array and prints them. Done
7. Write a C# program to read n values in an array and display them in reverse order. Done
8. Write a C# program to find the sum of all array elements. Done
9. Write a C# program to copy the elements of one array into another array. Done
10. Write a C# program to find duplicate characters in a string. Done

11.Write a C# program to get all unique characters in a string. Done

**Section-4:indexers:**

1. What is an indexer in C#? How does it compare to an array? Done
2. How do you declare and define an indexer in a class? Done
3. What are the different types of parameters an indexer can accept? Done
4. Can an indexer have both a getter and a setter? Explain the functionalities. Done
5. Can an indexer be overloaded (have multiple definitions with different parameter types)? Done
6. Design a class with an indexer to represent a student's list of grades, allowing access and modification by index. Done
7. Imagine developing an e-commerce application. You could create a ShoppingCart class with an indexer to manage the list of items added by a user. This indexer would allow accessing and modifying individual items in the cart based on their position (index). Done
8. For a password management system, you could build a PasswordManager class using an indexer to store usernames as keys and their corresponding passwords as values. The indexer would act like a dictionary, enabling users to retrieve or update password entries by username. Done

HAPPY CODING

