Punjab University College of Information Technology

BS (SE) Fall 2022 Morning

Web Engineering

Lab 2 - Introduction to C# Programming

Time Duration: 2 hr, Total Marks: 30

Objective:

In this lab, you will create a single task that covers all key concepts such as **Classes**, **Access Modifiers**, **String Concatenation**, **Properties (getters/setters)**, **Inheritance**, **Function Overriding**, and **File Handling**. You will build a small system that manages a list of **Products**, allows adding, retrieving, and displaying product details, and handles data using file operations.

Task Overview:

You will implement a **Product Management System** where:

- You create base classes and derived classes to manage different types of products.
- You will use string concatenation to display product information.
- File handling will be used to store and retrieve product data.

Step 1: Create a Class Structure with Properties (8 points)

1. **Create a Base Class Product:**
    - Private attributes:
        - ProductId
        - ProductName
        - Price
    - Public properties to get and set the values of the attributes using **getters** and **setters**.
    - Use **Access Modifiers** to control the visibility of the class members.

Step 2: Inheritance and Function Overriding (8 points)

1. **Create a Derived Class Electronics that Inherits from Product:**
    - Add a new attribute WarrantyPeriod.
    - Override the GetProductInfo method to display additional information about the warranty.
2. **Create another Derived Class Groceries that Inherits from Product:**
    - Add a new attribute ExpiryDate.
    - Override the GetProductInfo method to include the expiry date.

Step 3: String Concatenation (2 points)

1. **Demonstrate String Concatenation in the GetProductInfo Method**:
    - Use both the + operator and the String.Concat() method to concatenate the product details in the GetProductInfo method of each class.

Step 4: File Handling (Reading/Writing Data) (6 points)

1. **Create a Class ProductService to Handle File Operations:**
    - **Add Product to File:** Write the product details into a file named products.txt.
    - **Retrieve Product by ID:** Read through the file and retrieve the product based on its ID.
    - **List All Products:** Read and display all the products stored in the file.

Step 5: Main Program to Test the Functionality (4 points)

1. **In the Main method:**

- **Menu Prompt:** Added a simple menu system to allow the user to select the type of product to add (Electronics or Groceries) or exit the program.
- **Input Handling:** Prompts the user to input product details like Product ID, Product Name, Price, Warranty Period for Electronics, and Expiry Date for Groceries.
- **Exit Option:** An exit option (choice 3) is provided to end the program.
- **Product Retrieval:** After adding products, the program lists all products and allows the user to retrieve a product by its ID.

Readability and Documentation (2 points)

- A well organized and documented code is a blessing for software!
- So ensure your code is well-commented and follows C# coding conventions.
- Use meaningful variable and method names for better readability.
- Use PascalCase or camelCase for both Method names and Property names.

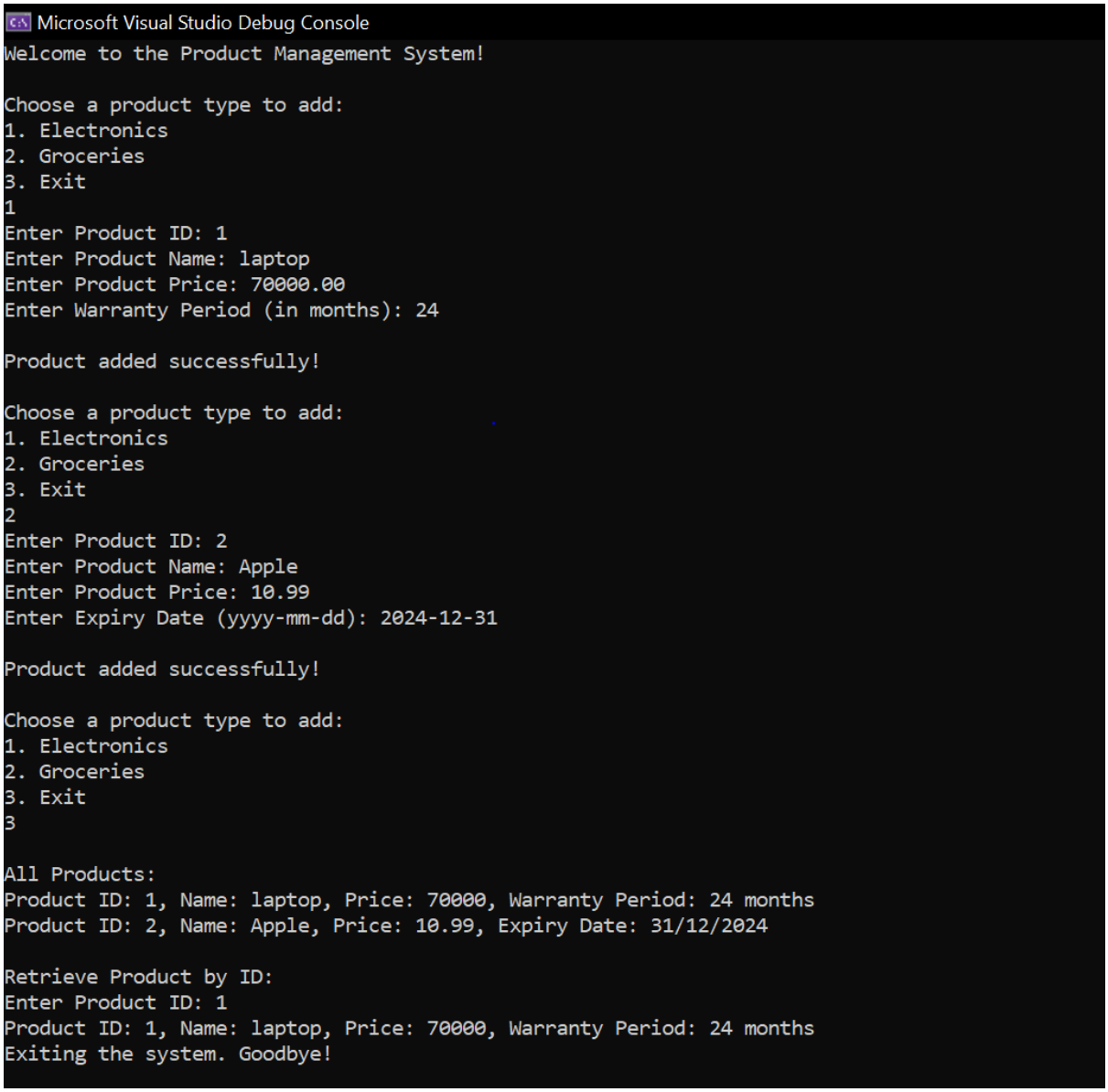
Here is the structure of the project:

```
├── Product.cs            (Base Class)
├── Electronics.cs        (Derived Class)
├── Groceries.cs          (Derived Class)
├── ProductService.cs     (File Handling Logic)
├── Program.cs            (Main entry point)
└── products.txt          (File for storing product data - will be created at runtime)
```

However, products.txt file location may depend on how the program is executed. Typically, for a .NET project, the file would be generated in the following directory:

ProductManagement\bin\Debug\net7.0\products.txt

Sample Output:



Summary of Lab:

1. **Product.cs**
   Contains the Product class with properties (ProductId, ProductName, Price) and the base method GetProductInfo().
2. **Electronics.cs**
   Contains the Electronics class, which inherits from Product, adds a WarrantyPeriod property, and overrides GetProductInfo().
3. **Groceries.cs**
   Contains the Groceries class, which inherits from Product, adds an ExpiryDate property, and overrides GetProductInfo().

4. **ProductService.cs**
   Implements file handling methods to add products, list all products, and get a product by ID from the products.txt file.
5. **Program.cs**
   It includes a menu for adding Electronics or Groceries, inputting product details (e.g., ID, Name, Price, Warranty for Electronics, Expiry Date for Groceries), an option to exit, and retrieval of products by ID after listing all added items.

6. **products.txt**
   This is the file that will be created dynamically when the program runs, storing product information. It may not appear in Solution Explorer by default but will be generated at runtime in the project's root directory.