Name: Koutou Richmond

Date: 02/23/2025

Estimated Development Time: 8h

Development Log:

02/21/2025 - 1 hour - I created a parent class called "Person" and two child classes "Student" and "Teacher". The child classes inherit from the parent class "Name" and "Old"

02/22/2025 - 2 hours - Create two add functions: for student and for teacher.

02/22/2025 - 1 hour - Check if the entered data is correct.

02/23/2025 - 1 hour - Display a message showing the validation of the entered fields

02/23/2025 - 1 hour - Testing, debugging, and final documentation.

02/27/2025 – 2 Hours Documentation remarks

Total Development Time: 8 hours

Student & Teacher Registration System - Documentation

1. Program Description

This C# application enables users to register **students and teachers** while implementing **Object-Oriented Programming (OOP)** principles. The program consists of a **Graphical User Interface (GUI)** with multiple forms for input collection and validation.

- **Key Features:**
  Object-Oriented Design with **inheritance**
  Student & Teacher registration
  Data validation to ensure correct input
  Multiple forms for navigation
  User-friendly **GUI**

---

2. Features & Functionality

Object-Oriented Structure (OOP)

- **Base Class (`Person`)**: Defines `Name` and `Old` properties.
- **Derived Class (`Student`)**: Inherits from `Person`, adds `Subject`.
- **Derived Class (`Teacher`)**: Inherits from `Person`, adds `Classroom`.
- **Encapsulation**: Uses **private fields with public getters** to protect data integrity.

Graphical User Interface (GUI)

- **Main Menu (Form1)**: Navigation buttons to open `Form2` and `Form3`.
- **Student Registration (Form2)**: Input fields for **Name, Age, and Subject**.
- **Teacher Registration (Form3)**: Input fields for **Name, Age, and Classroom**.

Data Validation

- **Ensures `Old` (age) is an integer.**
- **Validates `Name` format using regex.**
- **Prevents empty or incorrect entries.**

Navigation Between Forms

- `Form2` and `Form3` open from **Main Form** (`Form1`).
- Buttons `button1` and `button2` handle **form transitions**.

4. Remarks & Analysis

1 Estimated vs. Actual Time

- **Estimated Time: 8 hours**
- **Actual Time: 8 hours**
- The estimate was **accurate**, indicating proper planning.

2. Challenges Encountered

- **Data Validation Issues**: Ensuring correct name format and age input.
- **Form Navigation**: Ensuring smooth transitions between `Form1`, `Form2`, and `Form3`.
- **Encapsulation**: Structuring private fields with public getters for security.

3 Key Takeaways & Future Improvements

Enhance Error Handling:

- Use **try-catch blocks** to handle invalid inputs.
- Provide **real-time feedback** (e.g., change text color when input is incorrect).

Improve UI:

- Implement a **more structured layout with icons and tooltips**.
- Use **themes and colors** to make the interface user-friendly.

Data Storage Integration:

- Extend functionality to **store user data in a database or file**.

Testing Strategy:

- Allocate more time for testing to **cover edge cases and improve user experience**.

. Conclusion

This project successfully demonstrates **Object-Oriented Programming, GUI design, and data validation** in C#. Future improvements will focus on **UI refinements, better data handling, and enhanced error handling**.