Selected files

4 printable files

01_DATA_TYPES\10_PROPERTIES\Program.cs
01_DATA_TYPES\10_PROPERTIES\StudentAutoProp.cs
01_DATA_TYPES\10_PROPERTIES\StudentWithoutProperties.cs
01_DATA_TYPES\10_PROPERTIES\StudentWithProperties.cs

01_DATA_TYPES\10_PROPERTIES\Program.cs

```
1  using System;
2
3  namespace _10_PROPERTIES
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              #region 01. STUDENT WITHOUT PROPERTY EXAMPLE:
10             // Demonstration of StudentWithoutProperties (no properties for private fields)
11             Console.WriteLine("=== Student Without Properties ===");
12
13             // Create an instance and initialize via constructor
14             StudentWithoutProperties studentWithoutProps = new StudentWithoutProper-
   ties("Thillai", 50);
15
16             // Set public fields directly (no validation)
17             studentWithoutProps.StudentID = 20;
18             studentWithoutProps.StudentAddress = "Kumbakonam";
19
20             // Display information
21             studentWithoutProps.DisplayInfo();
22
23             // Attempting to access private fields directly will cause a compilation error
24             // Error: 'StudentWithoutProperties._studentName' is inaccessible due to its
   protection level
25             // studentWithoutProps._studentName = "THILLAI"; // Uncommenting this will
   cause an error
26             // studentWithoutProps._studentAge = 20; // Uncommenting this will cause an
   error
27
28             // Problem: No way to modify private fields outside the constructor, limiting
   flexibility
29             Console.WriteLine("Note: Cannot modify private fields (_studentName,
   _studentAge) without properties.\n");
30             #endregion
31
32             #region 02. STUDENT WITH PROPERTY WITH EXAMPLE:
33             // Demonstration of StudentWithProperties (using properties for private fields)
34             Console.WriteLine("=== Student With Properties ===");
35             // Create an instance
36             StudentWithProperties studentWithProps = new StudentWithProperties();
37
38             // Set public fields directly (no validation)
```

```
            studentWithProps.StudentID = 50;
            studentWithProps.StudentAddress = "Thanjavur";

            // Set private fields using properties (includes validation)
            try
            {
                studentWithProps.Name = "Thillai";
                studentWithProps.Age = 40;
                studentWithProps.DisplayInfo();

                // Try setting invalid values to demonstrate property validation
                studentWithProps.Name = ""; // Will throw an exception
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }

            // Show that valid updates work
            try
            {
                studentWithProps.Name = "Alice";
                studentWithProps.Age = 25;
                studentWithProps.DisplayInfo();
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
            #endregion

            #region 03. STUDENT WITH AUTO-IMPLEMENT PROPERTY:
            // Demonstration of StudentWithAutoProperties (auto-implemented properties)
            Console.WriteLine("\n=== Student with Auto-Implemented Properties ===");

            // Create a student with auto-implemented properties
            StudentWithAutoProperties studentAuto = new StudentWithAutoPrope-
rties("Thillai", 40, 2001, "Kumbakonam");
            studentAuto.DisplayInfo();

            // No validation, so empty or invalid values are allowed
            studentAuto.Name = ""; // No exception, but allows invalid data
            studentAuto.Age = -5;  // No exception, but allows invalid data
            studentAuto.DisplayInfo();

            // Update valid values
            studentAuto.Name = "Alice";
            studentAuto.Age = 25;
            studentAuto.DisplayInfo();
            #endregion

        }
    }
}
```

```
92
```

01_DATA_TYPES\10_PROPERTIES\StudentAutoProp.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4
5  namespace _10_PROPERTIES
6  {
7      // Class using auto-implemented properties
8      class StudentWithAutoProperties
9      {
10         // Auto-implemented properties: Compiler creates backing fields
11         public string Name { get; set; } // No validation, simple get/set
12         public int Age { get; set; }     // No validation, simple get/set
13         public int StudentID { get; set; } // No validation, simple get/set
14         public string Address { get; set; } // No validation, simple get/set
15
16         // Constructor to initialize properties
17         public StudentWithAutoProperties(string name, int age, int studentID, string
   address)
18         {
19             Name = name;
20             Age = age;
21             StudentID = studentID;
22             Address = address;
23         }
24
25         // Method to display student information
26         public void DisplayInfo()
27         {
28             Console.WriteLine($"Student (Auto Properties): ID={StudentID}, Name={Name},
   Age={Age}, Address={Address}");
29         }
30     }
31 }
32
```

01_DATA_TYPES\10_PROPERTIES\StudentWithoutProperties.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4
5  namespace _10_PROPERTIES
6  {
7      // Class to demonstrate accessing private fields without properties
8      class StudentWithoutProperties
9      {
10         // Public fields: Directly accessible from outside the class, no validation
11         public int StudentID;
12         public string StudentAddress;
13
14         // Private fields: Not accessible outside the class, no properties provided
```

```
        private string _studentName;
        private int _studentAge;

        // Constructor to initialize private fields
        // Parameters: name (student's name), age (student's age)
        public StudentWithoutProperties(string name, int age)
        {
            _studentName = name;
            _studentAge = age;
        }

        // Method to display student information
        public void DisplayInfo()
        {
            Console.WriteLine($"Student Info (Without Properties): ID={StudentID}, Name={_studentName}, Age={_studentAge}, Address={StudentAddress}");
        }
    }
}
```

01_DATA_TYPES\10_PROPERTIES\StudentWithProperties.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace _10_PROPERTIES
{
    // Class to demonstrate accessing private fields using properties
    class StudentWithProperties
    {
        // Public fields: Directly accessible, no validation (not recommended for sensitive data)
        public int StudentID;
        public string StudentAddress;

        // Private fields: Encapsulated, only accessible via properties
        private string _studentName;
        private int _studentAge;

        // Property for StudentName: Provides controlled access to _studentName
        public string Name
        {
            // Get accessor: Returns the private field's value
            // Set accessor: Sets the private field's value with validation

            set
            {
                if (string.IsNullOrEmpty(value))
                {
                    throw new ArgumentException("Student name cannot be null or empty.");
                }
                _studentName = value;
            }
            get
```

```
            {
                return _studentName;
            }
        }

        // Property for StudentAge: Provides controlled access to _studentAge
        public int Age
        {
            // Get accessor: Returns the private field's value
            // Set accessor: Sets the private field's value with validation

            set
            {
                _studentAge = value;
            }
            get
            {
                return _studentAge;
            }
        }

        // Constructor: Initializes the object with default values
        public StudentWithProperties()
        {
            _studentName = "Unknown";
            _studentAge = 0;
            StudentID = 0;
            StudentAddress = "Unknown";
        }

        // Method to display student information
        public void DisplayInfo()
        {
            Console.WriteLine($"Student Info (With Properties): ID={StudentID}, Name=
{Name}, Age={Age}, Address={StudentAddress}");
        }

    }
}

```