

Задание №3.

Классы

Описание задания:

1. Ниже приведены **классы**, описанные на формальном языке. Переведите нижеописанные классы с формального языка на язык C#. Все классы должны находиться в папке Model проекта Programming. Убедитесь, что программа компилируется.

```
// Класс Прямоугольник
// Начало описания класса
// Вещественное поле Длина
// Вещественное поле Ширина
// Строковое поле Цвет
// Конец описания класса
```

```
// Класс Рейс
// Начало описания класса
// Строковое поле Пункт Вылета
// Строковое поле Пункт назначения
// Целочисленное поле Время полета в минутах
// Конец описания класса
```

```
// Класс Фильм
// Начало описания класса
// Строковое поле Название
// Целочисленное поле Продолжительность в минутах
// Целочисленное поле Год выпуска (от 1900 до текущего года)
// Строковое поле Жанр
// Вещественное поле Рейтинг (от 0 до 10)
// Конец описания класса
```

```
// Класс Время
// Начало описания класса
// Целочисленное поле Часы (от 0 до 23)
// Целочисленное поле Минуты (от 0 до 60)
// Целочисленное поле Секунды (от 0 до 60)
```

// Конец описания класса

2. Для нижеперечисленных классов придумайте поля (3-4 поля), определите наиболее подходящие типы данных для полей, а затем напишите код этих классов на языке C#. Убедитесь, что программа компилируется:

- Контакт (в телефонной книжке на смартфоне)
- Песня (в плеере)
- Дисциплина (как запись в зачетной книжке)

3. Все поля созданных классов должны быть строго закрытыми (модификатор `private`). Ни одно из полей не должно быть статическим (модификатор `static` отсутствует).

4. Для каждого поля в каждом созданном классе создайте открытые **свойства** – методы-аксессоры, предоставляющие доступ к полям объектов в клиентском коде.

5. Если поле имеет ограничения на вводимые значения в своей предметной области, то сеттер свойства для этого поля должен реализовывать проверку на корректность вводимого значения. Например, поле *Ширина* в классе *Прямоугольник* не может быть отрицательным. Свойство должно проверять присваиваемое значение и, если оно отрицательное, должно выкидывать исключение типа `ArgumentException`.

6. Если поле не имеет никаких ограничений, замените поле на **автосвойство**.

7. Добавьте **конструкторы** в каждый созданный класс. Конструктор должен инициализировать все поля класса. Инициализация должна проходить через вызовы свойств, а не прямое обращение к полям, т.е. значения в поля должны присваиваться строго через проверки, реализованные в свойствах.

8. Добавьте в каждый класс конструктор без аргументов, для возможности создания объектов без инициализации полей.

9. В главном окне `MainForm` добавьте новую вкладку `Classes` в элемент `TabControl`. Добавьте на вкладку элементы согласно макету ниже:

The screenshot shows a Windows application window titled "Rectangles". On the left side, there is a list box containing five items: "Rectangle 1", "Rectangle 2", "Rectangle 3", "Rectangle 4", and "Rectangle 5". To the right of the list box, there are three text input fields. The first field is labeled "Lenght:" and contains the value "10". The second field is labeled "Width:" and contains the value "25". The third field is labeled "Color:" and contains the value "Orange". Below these three fields, there is a button labeled "Find".

10. В логике главного окна добавьте закрытое поле, хранящее массив прямоугольников. Именование закрытых полей должно подчиняться стилю именования camel с нижнего подчеркивания, например `_rectangles`.

11. В логике главного окна добавьте закрытое поле типа ранее созданного класса Прямоугольник. Назовите поле `_currentRectangle`.

12. В конструкторе главного окна проинициализируйте массив `_rectangles` массивом из пяти объектов прямоугольников. Значения длины и ширины должны определяться с помощью генератора случайных чисел.

13. Добавьте пять элементов в `ListBox`, как показано на макете.

14. Сделайте обработчик события `RectanglesListBox_SelectedIndexChanged()`, где при смене выбранного элемента в списке `ListBox`, будет меняться объект в поле `_currentRectangle` на один из объектов из массива `_rectangles` по текущему выбранному индексу.

15. При смене выбранного элемента в списке прямоугольников должно происходить автоматическое заполнение полей в соответствующих текстовых полях с шириной, длиной и цветом. То есть, когда пользователь выбирает в списке любой из прямоугольников, в текстовых полях должны показываться текущие значения полей выбранного прямоугольника.

16. Для каждого текстового поля создайте обработчик события `TextChanged`. При изменении текста в текстовом поле, новое значение текста должно присваиваться в соответствующее свойство объекта `_currentRectangle`. То есть, пользователь может вручную задать значения для полей текущего выбранного прямоугольника.

17. Для ввода длины и ширины нужно сделать обработку ситуации, когда в поля введены некорректные значения (текст не может быть преобразован в число или число выходит за допустимый диапазон значений). Обработка ошибки должна быть реализована через обработку исключений – конвертирование строки в число и присвоение числа в свойство должны быть помещены в блок `try`. Если в блоке `try` вылетит исключение, то фон текстового поля должен подсветиться красным цветом (используйте цвет `LightPink`). Если ошибок не произошло, тогда фон текстового поля должен стать обычным белым.

18. Добавьте в главное окно закрытый метод `FindRectangleWithMaxWidth()`. Метод должен принимать на вход массив прямоугольников и находить среди них прямоугольник с наибольшей шириной. Метод должен вернуть индекс найденного прямоугольника.

19. Добавьте обработчик для кнопки `Find`. По нажатию на кнопку в обработчике должен вызываться метод `FindRectangleWithMaxWidth()`, и найденный индекс должен задаваться в качестве `SelectedIndex` списка `ListBox`.

20. По аналогии с прямоугольниками, реализуйте подобную часть главного окна для класса Фильм. Кнопка поиска должна находить фильм с наибольшим рейтингом.