

## Задание №2.

### Перечисления

#### Описание задания:

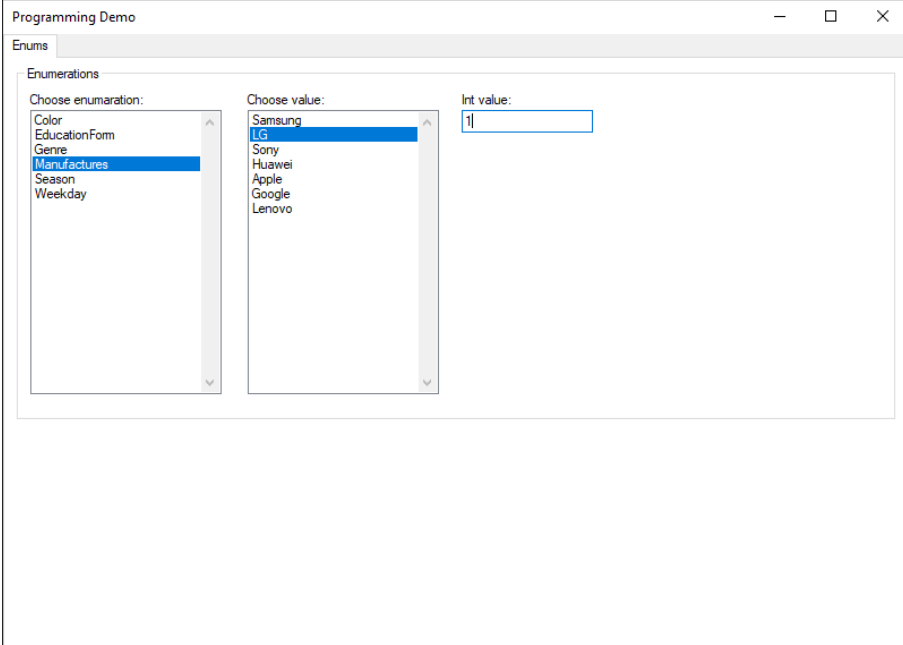
1. Создайте проект под названием Programming в репозитории. Тип проекта Windows Forms Application.
2. В новом проекте создайте две подпапки Model и View. Папка Model в дальнейшем будет хранить классы с логикой предметной области (так называемую бизнес-логику). Папка View будет хранить классы, отвечающие за работу пользовательского интерфейса (формы и сопроводительные классы).
3. Переименуйте класс Form1 в MainForm и переместите в подпапку View.
4. Добавьте в папку Model перечисление Weekday. Для этого добавьте новый файл, название файла должно совпадать с названием перечисления:

```
public enum Weekday
{
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday,
    Sunday
}
```

5. Создайте в папке Model новый файл и добавьте еще одно перечисление Genre (жанр кино) со значениями комедии, драмы, триллера, боевика, ужасов, блокбастера.
6. Создайте следующие перечисления, значения определите самостоятельно:
  - Цвет
  - Форма обучения студента (очная, заочная, вечернее, дистанционное)
  - Производители смартфонов
  - Время года

7. На главном окне разместите элемент TabControl, на котором должна быть одна вкладка с названием Enums. TabControl разместите во весь размер окна (свойство Dock в значение Fill).

8. На вкладке Enums разместите два списка (ListBox) и текстовое поле, как показано на макете:



9. При добавлении любых элементов на форму, не забываете их переименовывать в корректные названия – EnumsListBox вместо listBox1, ValuesListBox вместо listBox2. Это касается **всех** элементов интерфейса, включая элементы, которые будут добавлены позже.

10. Первый список должен содержать перечень всех реализованных перечислений. По умолчанию выбран первый элемент списка. При выборе перечисления, во втором списке должны отобразиться все значения выбранного перечисления. Используйте Enum.GetValues() для инициализации второго списка значениями.

11. При выборе любого значения перечисления во втором списке, в соответствующем текстовом поле должно отобразиться его числовое значение (числовое значение должно определяться явным преобразованием перечисления в целое число).

12. Протестируйте реализованную логику.

13. Добавьте на вкладку Enums элементы согласно макету:

Programming Demo

Enums

Enumerations

Choose enumeration:

- Color
- EducationForm
- Genre
- Manufactures
- Season
- Weekday

Choose value:

- Samsung
- LG
- Sony
- Huawei
- Apple
- Google
- Lenovo

Int value:

1

Weekday Parsing

Type value for parsing:

Tuesday

Parse

Это день недели (Tuesday = 2)

14. Реализуйте проверку парсинга введенного значения – пользователь вводит текст в текстовое поле и нажимает кнопку «Parse». Если текст удастся распознать как значение перечисления Weekday, тогда во втором текстовом поле указывается распознанное значение, а в скобках указывается его числовой эквивалент. Например:

- «Это день недели (Monday = 1)»
- «Это день недели (Friday = 5)»
- «Нет такого дня недели»

15. Для распознавания текста используйте метод Enum.Parse() или Enum.TryParse(), для определения числа используйте явное преобразование к целому типу. Используйте интерполяцию строк для формирования текста во втором текстовом поле

16. Протестируйте реализованную логику.

17. Добавьте на вкладку Enums элементы согласно макету:

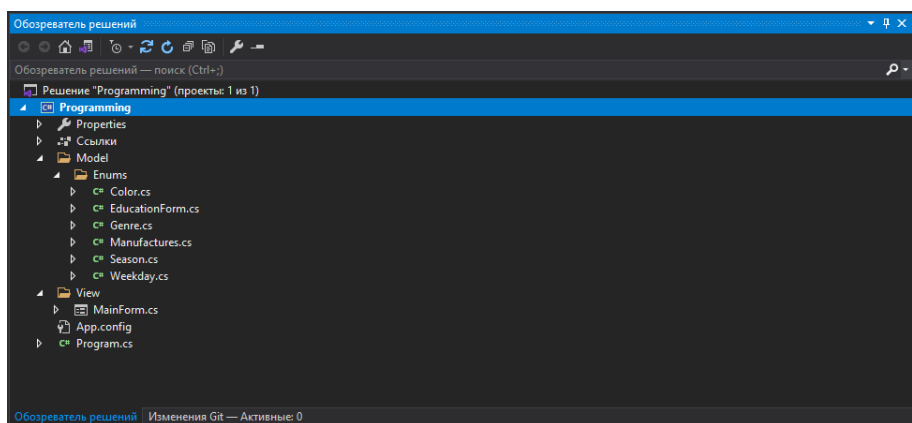
18. Реализуйте следующую логику – пользователь выбирает в выпадающем списке время года и нажимает кнопку «Go!». В зависимости от выбранного времени года должно произойти одно из действий:

- Если выбрано лето, показать всплывающее сообщение «Ура! Солнце!».
- Если выбрана осень, поменять цвет фона окна на оранжевый (например, можно использовать цвет #e29c45 в HEX-формате).
- Если выбрана зима, показать всплывающее сообщение «Бррр! Холодно!»
- Если выбрана весна, поменять цвет фона окна на зеленый (#559c45)

19. Действия на разные времена года могут быть произвольные, обязательно перечисленные выше. Главное, чтобы действия были разные. Реализовать данную логику необходимо с использованием оператора switch-case.

20. Протестируйте реализованную логику.

21. По завершению задания структура проекта должна выглядеть следующим образом:



22. По завершению задания сделайте коммит в репозиторий. По возможности, делайте коммиты чаще в ходе выполнения задания.

### Типовые ошибки:

1. Неправильные названия проекта и решения.
2. Нет папок Model и View, содержащих логику и пользовательский интерфейс соответственно.
3. Перечисления не находятся в отдельных файлах.
4. Названия файлов не соответствуют названиям перечислений.
5. Файлы перечислений не находятся в папке Model.
6. Перечисления оформлены неправильно – нарушения в табуляции, лишние пустые строки или отсутствие переносов строк в нужных местах, использование транслита или капса при именовании перечисления. Значения перечисления должны именоваться по-английски в стиле именованья Pascal.
7. Класс главного окна не переименован в MainForm или MainWindow и находится не в папке View.
8. Верстка главного окна выполнена не аккуратно, не соответствует макету, нет взаимного выравнивания элементов относительно друг друга.
9. Логика окна реализована неправильно, не выполняется исходное задание.
10. Логика окна реализована без использования указанных в задании методов.
11. Логика окна избыточна, дублируется.

**Защита задания:**

1. Во время защиты студент даёт ссылку на свой репозиторий на GitHub с выполненным заданием в ветке develop.
2. Преподаватель выполняет проверку задания на правильность (см. список типовых ошибок), оставляет замечания. После правильного исправления всех замечаний задание считается выполненным и принятым.