**МЕЖГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»**

**Кафедра «Автоматизированные системы управления»**

**ВИЗУАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ ПРОГРАММИРАММНЫХ ПРИЛОЖЕНИЙ**

**Лабораторные работы**

**Объектно-ориентированное программирование**

Могилев 2022

**Лабораторная работа №1**

**Классы, свойства, индексаторы.**

**Цель работы:** получить практические навыки по разработке программ с использованием классов.

**1 Задание:**

Создать класс согласно своего варианта. Создать соответствующий тип-*перечисление* (enum).

Класс должен содержать:

* 4-5 скрытых полей;
* одно из полей должно быть типа *перечисление;*
* на каждое поле должно быть свойство;
* хотя бы одно свойство должно иметь проверку вводимых данных;
* конструктор по умолчанию, 2 конструктора с параметрами;
* переопределенный метод ToString().

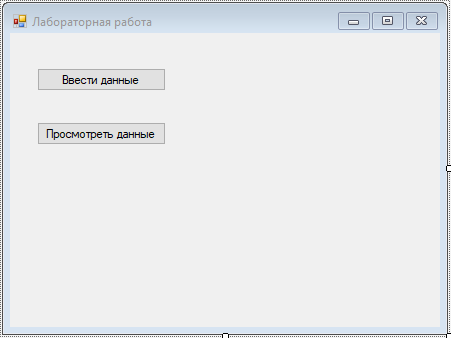
**2 Порядок выполнения работы**

1. Создайте новый проект Windows Forms.

2. Задайте имя форме и измените текст в названии окна.

3. Добавьте на форму две кнопки.

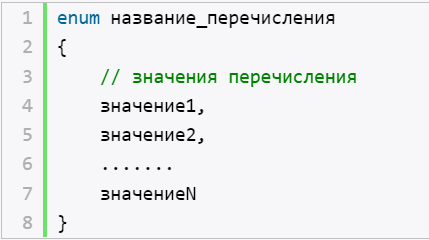
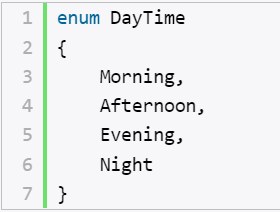
4. Задайте им соответствующие названия и текст.



5. Добавьте в проект класс. Сразу задайте ему необходимое имя. Объявите класс как **public**.

6. Создайте соответствующие поля.

7. Одно из полей должно быть типа перечисление. Сделайте его тоже **public**. Перечисление создается вне класса следующим образом:

8. Создайте конструктор с параметрами.

9. Создайте конструктор без параметров, который будет вызывать конструктор с параметрами и устанавливать какие-то «дежурные» значения.

10. Создайте конструктор с произвольным количеством параметров.

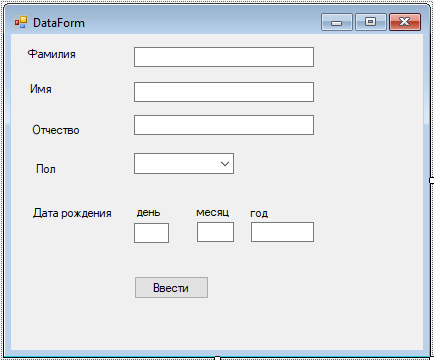
11. На все поля напишите свойства.

12. Свойство для работы с полем типа **enum**, должно принимать и возвращать тип String.

14. Добавьте в проект еще одну форму для ввода и просмотра данных.

(Для пункта, где будет использоваться перечисление добавьте **ComboBox**)

*Например:*



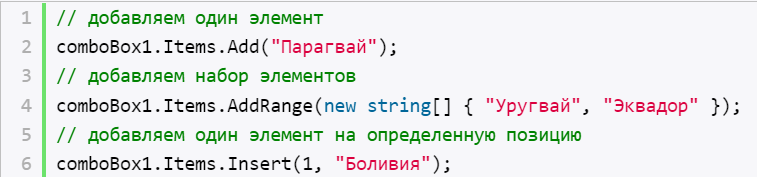
15. В классе главной формы создайте поле типа вашего класса (по заданию).

16. На главной форме щелкните по кнопке «Ввести данные» и в сгенерированном событии организуйте создание объекта класса, а так же создание и открытие второй формы.

17. Для того, чтобы формы могли обмениваться данными между собой, организуем передачу объекта класса через конструктор. Для этого напишем еще один конструктор для второй формы. В качестве параметра он будет получать объект класса и вызывать конструктор по умолчанию.

В классе второй формы объявите поле типа вашего класса. В конструкторе объект, полученный в качестве параметра, положите в поле класса. Теперь в него можно записывать данные или читать их.

18. **ComboBox** представляет собой простой список. Ключевым свойством этого элемента является свойство **Items**, которое как раз и хранит набор всех элементов списка. Для того чтобы в **ComboBox** добавить записи, которые должны отображаться для выбора воспользуйтесь методом **Add** (новые элементы помещаются в конец списка), **AddRange** (позволяет добавить сразу несколько элементов, например, массив) или **Insert** (вставка производится по определенному индексу списка):



Программным путем добавить список в **ComboBox** лучше всего в конструкторе формы.

19. В обработчик события нажатия кнопки «Ввести» добавьте проверки и ввод данных из TextBox-ов в поля класса. Для проверки выбора записи в ComboBox-е воспользуйтесь свойством **SelectedItem** (оно возвращает тип **Object**, так что не забудьте привести его к нужному вам типу). Добавьте в функцию логическую переменную, которая будет хранить результат проверок. Если хотя бы одно поле не заполнено или заполнено неверно отобразите на форме сообщение об ошибке (добавьте **Label** и управляйте его отображением с помощью свойства **Visible**). Если все данные введены верно, они присваиваются полям класса и вторая форма закрывается.

20. Теперь организуем просмотр данных объекта.

21. На главной форме на щелчок кнопки «Просмотр данных» напишите создание и открытие второй формы. Но, прежде чем передавать в конструктор объект класса необходимо проверить, не пустой ли он.

22. Можно создать еще одну форму для просмотра данных объекта, осуществляя вывод не в TextBox-ы, а на Label-ы.

Другим вариантом будет вывод данных в ту же форму, где мы их вводили, но сделать поля недоступными для изменения. Для этого можно воспользоваться свойством **ReadOnly** для **TextBox** (true) и свойством **Enabled** для **ComboBox** (false). Для того чтобы форма знала, для чего ее открыли (для ввода или для просмотра) можно добавить еще один параметр логического типа в конструктор второй формы. Если форма открыта для просмотра, то в TextBox-ы выводятся значения полей объекта и они закрываются от доступа. Можно сделать кнопку «Ввести» невидимой, а добавить кнопку «Закрыть», для закрытия формы.

**3 Варианты заданий:**

|  |  |
| --- | --- |
| **№ варианта** | **Класс** |
| 1 | Телефон |
| 2 | Человек |
| 3 | Животное |
| 4 | Транспортное средство |
| 5 | Текст |
| 6 | Жилое здание |
| 7 | Человек |
| 8 | Лекарственное средство |
| 9 | Компьютер |
| 10 | Товар |

**4 Контрольные вопросы**

1. Какие базовые концепции положены в основу объектно-ориентированного программирования? В чем они заключаются?
2. Дайте определение терминам «класс» и «объект». Как соотносятся эти понятия между собой?
3. Что такое инкапсуляция? В чем она заключается и как используется при работе с классами?
4. Что такое конструктор класса и для чего он используются в программе? Что такое конструктор копирования и конструктор по умолчанию?
5. В чём отличия свойств от полей?

**Лабораторная работа №2**

**Наследование. Перегрузка методов и операторов. Полиморфизм.**

**Цель работы:** получить навыки использования механизмов наследования и полиморфизма классов, закрепить навыки перегрузки методов и операторов.

**1 Задание**

Построить иерархию классов предметной области согласно варианту.

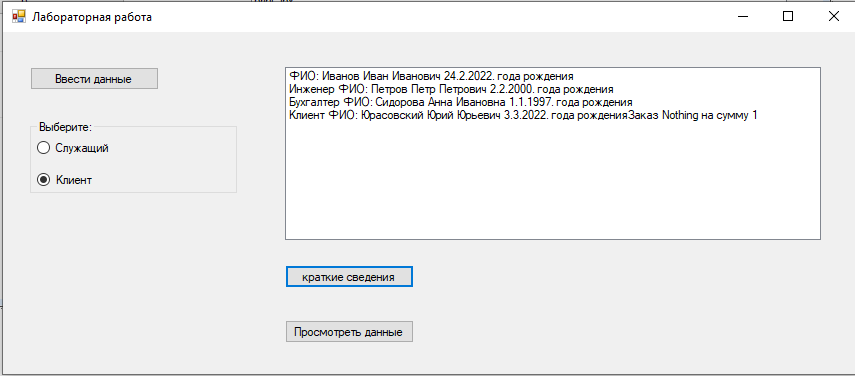
* В классах описать конструкторы с параметрами и конструкторы по умолчанию, свойства для установки и получения значений полей классов, методы для описания поведения объектов. Каждый из создаваемых классов должен иметь не менее двух методов, трех свойств, трех конструкторов.
* Для каждого созданного класса переопределить методы **ToString**() и **Equals**(). Метод **Equals** переопределить так, чтобы объекты считались равными, если равны значения полей объектов, а не ссылки объектов.
* Для каждого созданного класса переопределить один и тот же виртуальный метод родительского класса.
* Создать массив из объектов обоих производных классов и продемонстрировать полиморфизм в действии.

**2** **Порядок выполнения работы**

1. Создайте два производных класса, согласно своему варианту. В каждом должно быть несколько дополнительных полей и соответствующих свойств. Напишите конструктор по умолчанию, с параметрами и конструктор, принимающий в качестве параметра объект базового типа.
2. Для каждого созданного класса переопределить метод **Equals**().
3. В базовом классе реализуйте виртуальный метод (метод, который присущ всем объектам базового класса и его наследникам, но отличается для разных классов.)
4. Переопределите виртуальный метод определенный в базовом классе для каждого производного класса. Он может использовать метод базового класса и добавлять свой функционал или быть реализован совершенно по-другому, по вашему усмотрению.
5. На главной форме объявите массив объектов базового класса. Для этого удобней всего использовать **List**.
6. Добавьте на главную форму панель **GroupBox**. В нее вложите переключатели **RadioButton** для определения типа нового создаваемого объекта. Проверить, включен ли переключатель можно с помощью логического свойства **Checked**.
7. Далее, прежде чем создать новую форму для ввода новых объектов, сначала создайте объект нужного класса (базового или производного) и положите его в созданный ранее массив. А затем передавайте его в конструктор формы.
8. Продумайте ввод данных специфических для разных производных классов. Для проверки типа объекта используйте конструкцию:

person is Employee

1. На главную форму добавьте элемент управления **ListBox.**
2. Пример главной формы:



1. Организуйте обход в цикле всех объектов массива базового типа и вызов для них виртуального перегруженного метода. Результат отобразите в **ListBox**.
2. Чтобы очистить сразу весь список, примените метод **Clear**:

listBox1.Items.Clear();

|  |  |
| --- | --- |
|  |  |

1. Протестируйте все методы классов, а также метод **Equals**().

**3 Варианты заданий:**

|  |  |  |
| --- | --- | --- |
| **№варианта** | **Класс родитель** | **Производные классы** |
| 1 | Телефон | Смартфон, стационарный |
| 2 | Человек | Студент, преподаватель |
| 3 | Животное | Млекопитающее, птица |
| 4 | Транспортное средство | Грузовик, легковой автомобиль |
| 5 | Текст | Статья, заметка |
| 6 | Жилое здание | Коттедж, многоквартирный дом |
| 7 | Человек | Пациент, врач |
| 8 | Лекарственное средство | Таблетки, мазь |
| 9 | Компьютер | Ноутбук, десктоп |
| 10 | Товар | Продукт, домашняя техника |

**4 Контрольные вопросы**

1. Каким образом реализуется перегрузка конструкторов?
2. Можно ли перегрузкой операторов отменить очередность выполнения операции?
3. В чем заключена основная задача наследования?
4. Как осуществляется доступ к элементам производных и базовых классов?
5. Что такое полиморфизм? В чем он заключается и как используется при работе с классами?