## Практическая работа № 23. Задание 2

## *Разработка приложений по работе с простыми классами с использованием GUI*

**Цель работы**: Изучение правил разработки классов и использования GUI.

### Часть 1. Постановка задачи

*Имеются сведения об автомобилях и их владельцах:*

* *Государственный регистрационный номер.*
* *Модель.*
* *Цвет.*
* *ФИО владельца.*

*Требуется программа, позволяющая вводить указанные данные, хранить введенную информацию и удалять сведения об автомобиле и его владельце.*

#### Интерфейс пользователя

Список госномеров хранится в объекте ***comboBox1***. Поля ввода используются соответственно ***textBox1*** – ***textBox4***, кнопки – соответственно ***button1*** – ***button3***.

***Задание для самостоятельного выполнения***

*Постройте интерфейс пользователя по рис. 1.*

#### Сценарий работы пользователя и программы

*Пользователь запускает программу и видит на экране только список госномеров и кнопку* ***«УДАЛИТЬ»*** *– объекты группы данных не показываются. При выборе госномера из списка раскрываются данные. При вводе нового госномера также открываются поля данных, но они пустые. Пользователь вводит новые значения (или изменяет имеющиеся) и сохраняет изменения (или отказывается от сохранения изменений). Данные о каждом автомобиле – это отдельный объект. Сохраненные значения записываются в массив объектов, а в списке отображаются только госномера. Госномер можно удалить из списка – в этом случае из массива удаляются и данные об объекте.*

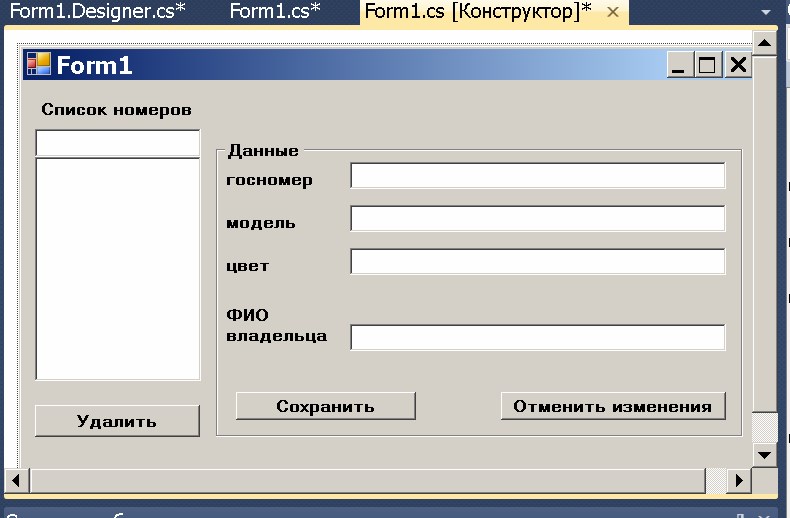


Рис. 1. Окно задачи

### Часть 2. Ввод и сохранение данных в массиве

#### Разработка некоторых элементов класса данных

Решение любой задачи начинается с описания используемых данных. В нашем случае данные имеют сложную структуру. Сведения о каждом транспортном средстве содержат четыре показателя, а самих автомобилей может быть любое количество. Фактически речь идёт о новом типе данных, содержащем в себе все перечисленные показатели. Оформим новый тип данных как класс ***Авто***. Поместим заготовку описания этого класса в модуль ***Form1*** в пространство имён ***Задача5\_ЛР9***, но обязательно после класса ***Form1***:

***public class Авто***

***{ }***

В составе класса Госномер, Модель, Цвет и ФИО определим как строки. Окончательно описание данных в классе будет таким:

***public class Авто***

***{ string Госномер; string Модель; string Цвет; string ФИО; }***

После нашей вставки в программе появятся ошибки, но они понятны: мы еще никак не используем данные. Продолжаем разрабатывать класс. Теперь необходимо описать конструктор объектов. Его назначение: при создании объекта дать исходные значения его полям. У нас четыре поля. Пусть при создании объекта они получат пустые значения. Тело конструктора будет содержать 4 оператора присваивания. Окончательно (фрагмент):

***string ФИО;***

***public Авто() {***

***Госномер = ""; Модель = ""; Цвет = ""; ФИО = "";***

***}***

Заметим: как только мы добавили конструктор, так ошибки сразу пропали – появились операторы, задающие значения полям объекта. Убедитесь, что запуск программы стал возможен.

#### Обработка ввода госномера

Логично представить себе, что пользователь при первом запуске программы захочет ввести данные о первом автомобиле. Он попробует набрать в верхней строке комбинированного списка госномер. Это событие – набор с клавиатуры. Точнее, нажатие символов на клавиатуре, ***KeyPress***. Такое событие связано с объектом ***СomboBox***, но оно требует ручного написания и подключения обработчика. Создадим обработчик, подключим его к событию.

Надо описать алгоритм, который будет работать при наборе госномера. В соответствии с сущностью события ***KeyPress*** обработчик будет выполняться каждый раз при любом нажатии на любую клавишу. Но госномер будет набран целиком только после нажатия клавиши ***«ENTER»***. Значит, все остальные нажатия метод должен проигнорировать.Но как определить, что нажата именно клавиша ***«ENTER»***?

Ответ на этот вопрос станет очевиден, если знать смысл параметров метода. Параметр ***sender*** – это источник события, т. е. объект ***comboBox1***. Мы об источнике и так знаем, доступ к нему по его имени имеем, а потому ***sender*** нам не пригодится. А вот второй параметр ***e*** – это контейнер, содержащий информацию о событии. Выражение ***e.KeyChar***позволяет узнать символ нажатой клавиши. Проверку можно осуществить так:

***if (e.KeyChar == (char) Keys.Enter) { }***

Вставим эту конструкцию в обработчик. Системное перечисление ***Keys*** включает в себя все символы клавиатуры, в том числе и ***Enter***. Только при сравнении его надо преобразовать в символ. Между фигурными скобками можно записать операторы, которые будут работать только при завершении набора госномера – все остальные клавиши никак не будут «замечены» обработчиком.

***Задание для самостоятельного выполнения***

*Убедитесь, что обработчик реагирует на нажатие клавиши* ***«ENTER»***. *Вставьте между фигурными скобками оператор выдачи на экран какого-либо сообщения (через messageBox) и запустите программу. Наберите в поле ввода что-нибудь и нажмите* ***«ENTER»***. *Убедившись – удалите оператор выдачи сообщения.*

Теперь определим, что делать, когда нажата клавиша ***«ENTER»***. Очевидно, нам придётся отобразить невидимые объекты окна – группу данных.

***Задание для самостоятельного выполнения***

*Добавьте оператор, делающий группу объектов видимой.*

#### Ввод данных

Теперь можно попробовать набирать данные в поля ввода. В общем-то всё нормально. Но! Ведь мы уже набрали госномер – почему же приходится набирать его ещё раз?

***Задание для самостоятельного выполнения***

*Добавьте ещё один оператор. Как только группа объектов стала видимой, перенесите в первое поле ввода значение набранного госномера.*

Есть и еще одно «но», значительно более существенное. Цвет лучше бы выбирать из какого-то списка, а не набирать. Пользователь вряд ли ошибётся с набором госномера и модели, а вот с цветом – запросто. Значит, придется запретить набор, т. е. установить у этого поля свойство ***Enable=false.*** А для выбора чуть-чуть изменим Интерфейс (рис. 2): добавим между строками цвета и ФИО объект ***Label*** с текстом «Выбор цвета» и объект ***comboBox2***, в котором и создадим список цветов. У последнего объекта установим запрет набора цвета – это свойство ***DropDownStyle***=***DropDownList***.

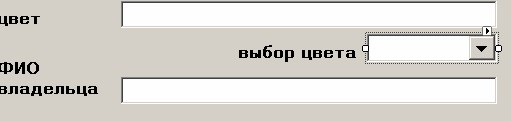


Рис. 2. Изменение интерфейса

Теперь решим вопрос с заполнением списка цветов. Создадим в пространстве имён проекта (у каждого свое имя) еще один тип данных – перечисление ***Color***. Включим в него шесть значений-констант (можно больше, но ограничимся этим):

***public enum Color { неопределенный, белый, красный, фиолетовый, серый, зелёный }***

Запишем этот оператор между классами ***Form1*** и ***Авто***. Затем создадим в ***comboBox2*** список цветов на основе перечисления ***Color***. Значения этого перечисления перепишем в коллекцию объекта ***comboBox2*** при запуске программы – перед открытием окна. Создадим обработчик события ***Load*** для окна ***Form1*** и вставим в него код:

***int i;***

***for (i = 0; i < 6; i++) comboBox2.Items.Add((Color)i);***

Запустите программу и убедитесь, что список цветов заполняется.

Выбранное значение цвета попадает в свойство ***Text*** объекта ***comboBox2***. Его надо выбрать и поместить в текстовое поле ***textBox3***. Воспользуемся событием ***SelectedIndexChanged***, которое как раз и возникает при выборе. Двойным кликом по объекту ***comboBox2*** создаем обработчик этого события. Подключается он автоматически. Внутрь обработчика запишем оператор:

***textBox3.Text = comboBox2.Text;***

Запустите программу и проверьте работу.

Решим попутно и ещё один интересный вопрос. Сделаем так, чтобы при наборе данных пользователь мог работать не только мышкой. При наборе данных удобно перемещаться от поля к полю клавишей ***«TAB»***. Откроем модуль ***Form1.Dеsigner.cs***. В секциях-описаниях каждого объекта есть оператор, определяющий порядок обхода объектов окна с помощью клавиши ***«TAB»***. Установим индексы (TabIndex) следующим образом: 0 – comboBox1, 1 – textBox1, 2 – textBox2, 3 – comboBox2, 4 – textBox4, 5 – кнопка button2, 6 – кнопка button3, 7 – кнопка button1. Остальные не играют роли. Теперь запустим программу и поработаем клавишей ***«TAB»***. Курсор должен перемещаться в указанном порядке.

На этом обработка ввода данных завершена. Можно начинать обработку нажатия кнопок.

#### Кнопка отмены изменений

При отмене изменений (или отказе от сохранения набранных данных) надо сделать:

* очистить поля ***textBox1*** – ***textBox4***;
* убрать с экрана (скрыть) блок ввода данных;
* очистить поле ввода в объекте ***comboBox1***.

***Задание для самостоятельного выполнения***

*Внесите необходимые изменения в программу.*

#### Кнопка сохранения результатов ввода данных

Введенные данные должны сохраниться в элементе массива данных – таково условие задачи. Но мы еще не создали массив. Где именно он должен размещаться и в каком виде? В каком из классов: ***Form1***, ***Program***, ***Авто***? Какого типа должен быть массив? Массив должен хранить все данные о каждом автомобиле. Значит, элементы массива должны быть типа ***Авто***. Предположим, что мы создаем хранилище на 1000 автомобилей. Можно для объявления использовать следующий оператор:

***Авто [] МД = new Авто[1000];***

Запишем этот оператор в самом начале класса ***Form1***. Это гарантирует доступ к нему из любого метода этого класса, т. е. в пределах окна. Кроме того, также в начале класса создадим переменную-счетчик, которая будет содержать количество заполненных элементов массива: ***int КоличЭлем;***

Не забудем эту переменную обнулить при открытии окна.

Теперь вспомним, что при объявлении массива в памяти выделено место только под его элементы, а они являются ссылками на объекты – при создании массива эти ссылки будут равны значению ***null***. Выделим место под каждый объект также при открытии окна:

***for (i = 0; i < 1000; i++) // создание объектов***

***{ МД[i] = new Авто(); }***

Теперь массив готов к приему данных. Заметим, что первый свободный элемент имеет номер, совпадающий со значением ***КоличЭлем***. Можно было бы введенные значения записать в поля первого свободного элемента массива. Однако, просто так этого сделать не удастся: поля объекта являются закрытыми. Придётся включить в состав класса ***Авто*** метод, позволяющий обновить значения полей объекта. Например, такой:

***public void Обновить(string g, string m, string c, string fio)***

***{ Госномер = g; Модель = m; Цвет = c; ФИО = fio; }***

Этот метод – динамический. Он может быть вызван для работы с объектом массива МД. В обработчике нажатия на кнопку «Сохранить» запишем:

***МД[КоличЭлем].Обновить(textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text); КоличЭлем++;***

После увеличения значения ***КоличЭлем*** следующий набор данных запишется уже в следующий по порядку объект.

***Задание для самостоятельного выполнения***

*Добавьте в обработчик button2 очистку полей ввода данных и сокрытие группы объектов набора данных, как это сделано в обработчике button3. Но!* *Не пишите повторно операторы, которые уже есть в обработчике для button3. Воспользуйтесь уже имеющимся методом.*

#### Отображение списка госномеров

Мы завершили обработку кнопок, связанных с вводом данных. Но если с отменой набора все очевидно, то с сохранением не ясно: то ли сохраняются данные, то ли нет. В списке-то новый госномер не появляется! Мы же его туда и не добавили.

Список госномеров – это список в коллекции ***comboBox1***. У нас уже есть опыт записи в коллекцию, т. к. мы уже составляли список цветов в ***comboBox2***. Только в том случае мы использовали перечисление, а здесь будем использовать массив объектов МД. Первое поле каждого объекта – это госномер.

***Задания для самостоятельного выполнения***

1. *В классе Авто создайте динамический метод, позволяющий получить госномер объекта Авто.*
2. *В обработчике кнопки* ***«СОХРАНИТЬ»*** *очистите коллекцию Items объекта comboBox1.*
3. *Просмотрите массив МД в цикле, выберите из каждого объекта поле Госномер с помощью созданного ранее метода и добавьте госномер в коллекцию Items объекта comboBox1. Алгоритм следует оформить в виде метода* ***СоздСписГос****. Метод вызвать в обработчике нажатием кнопки* ***«СОХРАНИТЬ»****.* ***Обработка выбора госномера из списка***

Первая особенность – это другое событие: не набор, а выбор значения из списка. Имя события – ***SelectedIndexChanged***. Обработчик этого события можно получить двойным кликом по объекту. Внутрь обработчика надо записать операторы, извлекающие значения из выбранного объекта и записывающие их в поля ввода ***textBox1***. А затем – отобразить группу. И всё.

Для заполнения поля ввода госномера ***textBox1*** сам объект не нужен. Ведь после выбора госномера он автоматически отображается в поле ввода объекта ***comboBox1*** (свойство ***Text***), откуда его можно перекинуть в поле ***textBox1***. Вместе с оператором отображения группы это выглядит так:

***textBox1.Text = comboBox1.Text; groupBox1.Visible = true;***

А вот дальше – две проблемы:

* получить номер выбранного объекта ***Авто***;
* извлечь из объекта значения модели, цвета и ФИО.

Первая проблема решается легко. У объекта ***comboBox1*** есть свойство ***SelectedIndex***, которое содержит номер выбранной строки списка (счёт с нуля). Этот номер совпадает с индексом объекта в массиве. Выбрать номер можно оператором:

***int nom=comboBox1.SelectedIndex;***

А вот извлечь значения сразу не получится – прямой доступ к полям ***Модель***, ***Цвет***, ***ФИО*** запрещен. Придётся ещё расширить функциональность класса ***Авто***. Например, добавить в него описания свойств, которые позволят получить значения закрытых полей. Итак:

***public string Мод { get { return Модель; } }***

***public string Цв { get { return Цвет; } }***

***public string Фио { get { return ФИО; } }***

Воспользуемся этими свойствами для получения значений:

***textBox2.Text = МД[nom].Мод; textBox3.Text = МД[nom].Цв; textBox4.Text = МД[nom].Фио;***

***Задание для самостоятельного выполнения***

*Наберите и сохраните два разных набора значений. Затем последовательно отобразите их, выбрав госномера. Обратите внимание, что в одном из случаев значение в поле цвета противоречит значению в списке цветов – там осталось последнее набиравшееся значение. Сделайте так, чтобы значение цвета в поле comboBox2 и textBox3 были одинаковыми.*

Теперь выберем какое-либо значение госномера и внесем изменение в данные. Например, заменим ФИО. Сохраним. В списке появится двойной номер, и это правильно: мы не установили контроля на повторную запись. Измененные данные должны записываться в тот же объект, если госномер остался прежним. Это значит, что алгоритм в обработчике ***button2*** должен различать, новый это госномер или существующий. Сейчас у нас обновляется свободный элемент массива, т.е. в массив пишется как бы новый элемент.

***Задания для самостоятельного решения***

1. *Внесите изменения в алгоритм обработчика кнопки button2, не допускающие двойных записей.*
2. *Разрешите пользователю набирать новые цвета и обеспечьте их сохранение в списке. Двойных записей в список цветов не делать. 3. Создайте обработчик кнопки* ***«УДАЛИТЬ»*** *и составьте алгоритм работы. Не забудьте, что удалять надо не только госномер из списка, но и данные из массива.*