# ПРАКТИЧЕСКАЯ РАБОТА № 2

**Тема:** Принципы объектно-ориентированного программирования

**Цель:** Создавать программы с использованием классов библиотеки FCL.

**Время выполнения:** 6 часов.

## 1 Входной контроль

1) Перечислите основные принципы объектно-ориентированного программирования.

Инкапсуляция, наследование, полиморфизм.

2) Опишите основные принципы объектно-ориентированного программирования.

Инкапсуляция – свойство придавать полям и методам различные уровни доступа.

Наследование есть свойство объектов порождать своих потомков. Объект-потомок автоматически наследует от родителя все поля и методы, может дополнять объекты новыми полями и заменять (перекрывать) методы родителя или дополнять их.

Полиморфизм - это свойство родственных объектов решать схожие по смыслу задачи разными способами.

## 2 Ход выполнения работы

## 2.1 Задание № 1. Разработка класса согласно варианту задания

В ходе выполнения данной работы необходимо разработать класс согласно индивидуальному варианту задания. А также класс MyApplication для работы с меню. Индивидуальные варианты заданий даны в таблице 1.

Таблица 1 – Индивидуальное задание

|  |  |  |
| --- | --- | --- |
| Вариант | Класс | Описание |
| 7 | Товар | Наименование – string  Количество – uint  Стоимость– double |

Реализация класса «Tovar» представлена в листинге 1.

Листинг 1 – Класс Tovar

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**namespace** Laba2

{

**class** Tovar

{

**public** **string** naimenovanie;

**public** **uint** kolichestvo;

**public** **double** price;

**public** Tovar()

{

**this**.naimenovanie = null;

**this**.kolichestvo = 1;

**this**.price = 0.0;

}

**public** Tovar(**string** naimenovanie, **uint** kolichestvo, **double** price)

{

**this**.naimenovanie = naimenovanie;

**this**.kolichestvo = kolichestvo;

**this**.price = price;

}

**public** **void** ElemAdd(List<Tovar> list, Tovar tovaros)

{

list.Add(tovaros);

}

**public** **void** ElemDelete(List<Tovar> list, Tovar tovaros)

{

list.Remove(tovaros);

}

**public** **void** Showlist(List<Tovar> list)

{

Console.WriteLine("Наименование Количество Стоимость");

**foreach** (var item in list)

{

Console.WriteLine(item.naimenovanie + " " + "{0,4}шт " +

"{1}руб", item.kolichestvo, item.price);

}

}

**public** **void** ShowCurrent(List<Tovar> list, int current)

{

Console.WriteLine(" Наименование" + " " + "Количество

Стоимость");

Console.WriteLine(list[current].naimenovanie + " " +

list[current].kolichestvo + "шт" + list[current].price + "rub ");

}

**public** **void** ElemChange(List<Tovar> list, int current, Tovar tovaros)

{

list[current] = tovaros;

}

**public** **void** ListSort(List<Tovar> list)

{

**int** sortEnd = 0;

**while** (sortEnd == 0)

{

sortEnd = 1;

**for** (**int** i = 0; i < list.Count - 1; i++)

{

**if** (list[i].price > list[i + 1].price)

{

sortEnd = 0;

Tovar temp = list[i + 1];

list[i + 1] = list[i];

list[i] = temp;

}

}

}

}

}

}

## 2.2 Задание № 2. Разработка класса для хранения списка объектов классов, используя коллекцию

Создать класс в Visual Studio для реализации хранения списка объектов классов посредством коллекции. Индивидуальное задание – реализация класса “Tovar ”.

Реализация класса представлена в листинге 1.

## 2.3 Задание № 3. Разработка класса MyApplication

Создать класс в Visual Studio, реализующий меню программы.

Реализация класса «MyApplication» представлена в листинге 2.

Листинг 2 – Код класса «MyApplication»

**using** Laba2;

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**namespace** Laba2

{

**class** MyApplication

{

**public** **void** Menu(Tovar tovar1, List<Tovar> tovarList)

{

**int** current = -1;

**int** n = 0;

**int** littleMenu = 0;

**do**

{

Console.Clear();

Console.WriteLine("1. Добавить элемент в коллекцию.");

Console.WriteLine("2. Удаление текущего элемента из списка");

Console.WriteLine("3. Перемещение по списку");  
 Console.WriteLine("4. Установка нового значения для текущего+

элемента");

Console.WriteLine("5. Вывести элементы списка");

Console.WriteLine("6. Показать текущий элемент списка");

Console.WriteLine("7. Сортировка");

Console.WriteLine("8. Выход");

n = Convert.ToInt32(Console.ReadLine());

**switch** (n)

{

**case** 1:

Console.WriteLine("Введите наименование товара");

**string** naimenovanie = Console.ReadLine();

Console.WriteLine("Введите количество");

**uint** kolichestvo = Convert.ToUInt32(Console.ReadLine());

Console.WriteLine("Введите стоимость");

**double** price = Convert.ToDouble(Console.ReadLine());

tovar1.ElemAdd(tovarList, new Tovar(naimenovanie,

kolichestvo, price));

current++;

**break**;

**case** 2:

tovar1.ElemDelete(tovarList, tovarList[current]);

current--;

**break**;

**case** 3:

**do**

{

Console.Clear();

Console.WriteLine("Выберите позицию текущего

элемента списка");

Console.WriteLine("1. Первый");

Console.WriteLine("2. Следующий");

Console.WriteLine("3. Предыдущий");

Console.WriteLine("4. Последний");

Console.WriteLine("5. Показать текущий элемент

списка");

Console.WriteLine("6. Назад");

littleMenu = Convert.ToInt32(Console.ReadLine());

**switch** (littleMenu)

{

**case** 4:

current = tovarList.Count - 1;

**break**;

**case** 3:

**if** (current > 0)

{

current--;

}

**break**;

**case** 2:

**if** (current < tovarList.Count)

{

current++;

}

**break**;

**case** 1:

current = 0;

**break**;

**case** 5:

Console.WriteLine();

tovar1.ShowCurrent(tovarList, current);

Console.ReadKey();

**break**;

}

} **while** (littleMenu < 6);

**break**;

**case** 4:

Console.WriteLine("Введите наименование товара");

**string** newNaimenovanie = Console.ReadLine();

Console.WriteLine("Введите количество");

**uint** newKolichestvo = Convert.ToUInt32(Console.ReadLine());

Console.WriteLine("Введите стоимость");

**double** newPrice = Convert.ToDouble(Console.ReadLine());

tovar1.ElemChange(tovarList, current, new

Tovar(newNaimenovanie, newKolichestvo, newPrice));

**break**;

**case** 5:

Console.Clear();

tovar1.Showlist(tovarList);

Console.WriteLine("Для продолжения нажмите любую

клавишу");

Console.ReadKey();

**break**;

**case** 6:

Console.Clear();

tovar1.ShowCurrent(tovarList, current);

Console.WriteLine("Для продолжения нажмите любую

клавишу");

Console.ReadKey();

**break**;

**case** 7:

Console.Clear();

Console.WriteLine("Выберите сортировку");

Console.WriteLine("1. По возрастанию");

Console.WriteLine("2. По убыванию");

Console.WriteLine("3. Назад");

littleMenu = Convert.ToInt32(Console.ReadLine());

**switch** (littleMenu)

{

**case** 1:

tovar1.ListSort(tovarList);

**break**;

**case** 2:

tovar1.ListSort(tovarList);

tovarList.Reverse();

**break**;

}

**break**;

}

} **while** (n < 8);

}

}

}

## 2.4 Задание № 4. Тестирование разработанных классов

Создать консольное приложение для тестирования разработанных классов.

Реализация класса «Program» представлена в листинге 3.

Листинг 3 – Код класса «Program»

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**namespace** Laba2

{

**class** Program

{

**static** **void** Main(**string**[] args)

{

MyApplication menu = **new** MyApplication();

menu.Menu(new Tovar(), **new** List<Tovar>());

}

}

}

Скриншоты реализованной программы представлены на рисунках 1 – 5.

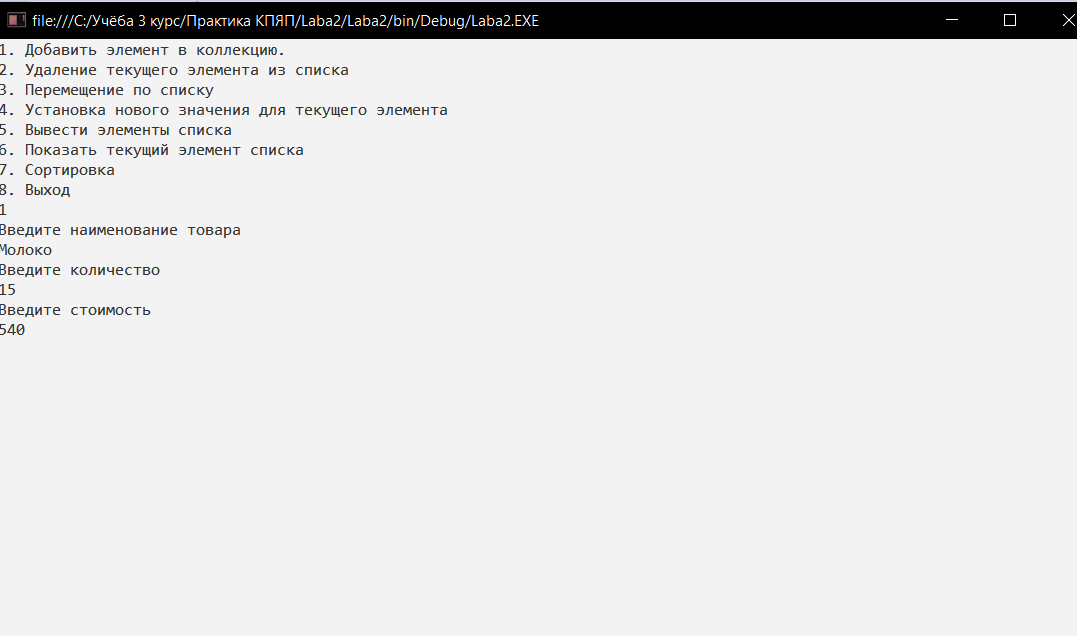


Рисунок 1 – Главное меню и добавление товара

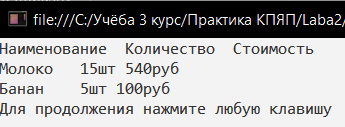


Рисунок 2 – Список товаров

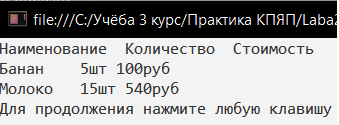


Рисунок 3 – Результат сортировки по возрастанию

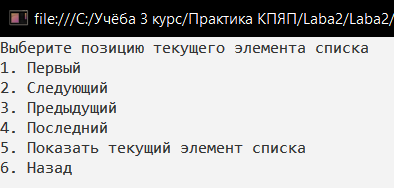


Рисунок 4 – Меню перемещения по списку

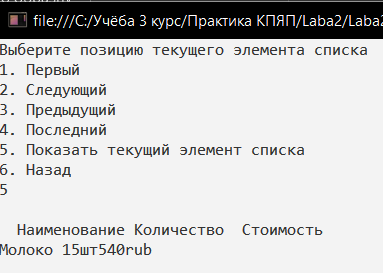


Рисунок 5 – Текущий элемент списка

**Вывод**

В ходе выполнения данной работы был разработан класс Tovar, Program, а так же класс MyApplication который хранит в себе меню программы и выполняет различные операции с данными класса Tovar.