Модуль 2, практическое занятие 3 (5-6)

Классы

Объекты

Задача 1

Класс *Item,* моделирует одну покупку. У покупок есть название, цена и количество. Класс *ShoppingCart* реализует корзину покупок в виде массива элементов.

Дополните класс, реализующий корзину покупок в виде массива элементов.

Дополните *ShoppingCart* :

- Объявите переменную *_cart* и инициализируйте её массивом размера *capacity* в конструкторе
- Дополните метод *IncreaseSize ()* кодом. Размер массива дожен увеличиваться на 3 элемента.
- Дополните метод *AddToCart*() кодом. Этот метод должен добавлять элемент в корзину и обновлять переменную *_totalPrice*.

Напишите программу, имитирующую покупки. Программа должна содержать цикл, который повторяется до тех пор, пока пользователь хочет что-нибудь купить. Каждую итерацию цикла считывайте название, цену и количество вещей, которые хочет приобрести пользователь и добавляйте их в корзину. После добавления элемента в корзину выводите содержимое корзины. После выхода из цикла напишите “Пожалуйста, заплатите …”, подставив вместо троеточия сумму покупок.

http://www.hh.se/download/18.70cf2e49129168da0158000145444/uppgift5

Задача 1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

/*
 * Item.cs
 *
 * Представляет предмет в корзине.
 */

public class Item {
    /// <summary>
    /// Название предмета
    /// </summary>
    public string Name { get; }

    /// <summary>
    /// Цена предмета
    /// </summary>
    public double Price { get; }

    /// <summary>
    /// Количество предметов
    /// </summary>
    public int Quantity { get; }

    /// <summary>
    /// Создаёт новый предмет на основе переданных свойств
    /// </summary>
    /// <param name="itemName">Название предмета</param>
    /// <param name="itemPrice">Цена предмета</param>
    /// <param name="numPurchased">Количество предметов</param>
    public Item(string itemName, double itemPrice, int numPurchased) {
        Name = itemName;
        Price = itemPrice;
        Quantity = numPurchased;
    }

    /// <summary>
    /// Возвращает строку, представляющую текущий объект
    /// </summary>
    /// <returns></returns>
    public override string ToString() {
        return $"{Name}\t\t{Price:C}\t\t{Quantity}\t\t{Price * Quantity:C}";
    }
}
```

Задача 1

```
/*
 * ShoppingCart.cs
 *
 * Представляет корзину покупок
 */

public class ShoppingCart {
    private int _itemCount; // количество предметов в корзине
    private double _totalPrice; // цена всех предметов в корзине
    private int _capacity; // текущая вместимость корзины

    /// <summary>
    /// Создаёт новый экземпляр корзины с вместимостью в 5 элементов
    /// </summary>
    public ShoppingCart() {
        _capacity = 5;
        _itemCount = 0;
        _totalPrice = 0.0;
    }

    /// <summary>
    /// Добавляет предмет в корзину
    /// </summary>
    /// <param name="itemName">Название предмета</param>
    /// <param name="price">Цена предмета</param>
    /// <param name="quantity">Количество предметов</param>
    public void AddToCart(string itemName, double price, int quantity) { }

    /// <summary>
    /// Увеличивает вместимость корзины на 3
    /// </summary>
    private void IncreaseSize() { }

    /// <summary>
    /// Возвращает предметы в корзине с дополнительной информацией
    /// </summary>
    public override string ToString() {
        string contents = "\nShopping Cart\n";

        contents += "\nItem\t\tUnit Price\tQuantity\tTotal\n";

        for (int i = 0; i < _itemCount; i++)
            contents += _cart[i] + "\n";

        contents += $"\nTotal Price: {_totalPrice:C}\n";

        return contents;
    }
}
```

Задача 2

Класс *IntegerList*, представляет список целых чисел. Методы:

- IntegerList(int size) – создаёт новый список из *size* элементов. Элементы равны 0.
- void Randomize() – заполняет список целыми числами между 1 и 100 включительно.
- void Print() – выводит список элементов и их индексы

Скопируйте код к себе в проект, скомпилируйте, запустите, чтобы изучить как он работает.

Часто необходима возможность добавления или удаления элементов из списка. Когда список хранится в массиве, этого можно добиться созданием нового массива необходимого размера и копированием в него значений из старого. Однако, это довольно-таки неэффективно. Более распространённой считается стратегия изначального создания массива некоторого небольшого размера и увеличения его размера в два раза каждый раз, когда текущий заполняется.

- Дополните класс *IntegerList.* Необходимо добавить метод *IncreaseSize* и поля для отслеживания количества записанных в массив элементов. Так как пока у вас нет способа добавить новый элемент, метод *IncreaseSize* вам пока не понадобится.

Задача 2

- Добавьте метод *void AddElement(int newVal)* в класс *IntegerList.* Данный метод добавляет новый элемент в список. В начале мотода *AddElement* проверьте, не является ли массив полностью заполненным. Если является, вызовите метод *IncreaseSize* перед дальнейшими действиями.
- Добавьте опцию к классу *IntegerListTest* для проверки нового метода.
- Добавьте метод *void RemoveFirst(int val)* в класс *IntegerList.* Данный метод удаляет первое вхождение элемента *val* из списка. Если указанного элемента в списке нет, метод не должен делать ничего (и ошибку он бросать тоже не должен). Удаление элемента не должно изменять размер массива, но учтите, что массив должен быть последовательным, так что после удаления элемента, необходимо сдвинуть все элементы правее него на одну позицию влево. Так же не забудьте уменьшить значение переменной, отвечающей за количество элементов в списке.
- Добавьте опцию к классу *IntegerListTest* для проверки нового метода.
- Добавьте метод *void RemoveAll(int val)* в класс *IntegerList.* Данный метод удаляет все вхождения элемента *val* из списка. Если указанного элемента в списке нет, метод не должен делать ничего (и ошибку он бросать тоже не должен).
- Добавьте опцию к классу *IntegerListTest* для проверки нового метода.

Задача 2

```
/*
 * IntegerList.cs
 *
 * Определяет список целых чисел с возможностью его создания и заполнения
 */

using System;


    public class IntegerList {
        private static readonly Random Random = new Random();

        private int[] _list;

        /// <summary>
        /// Создаёт список указанного размера
        /// </summary>
        /// <param name="size">Расмер списка</param>
        public IntegerList(int size) {
            _list = new int[size];
        }

        /// <summary>
        /// Заполняет список числами между 1 и 100 включительно
        /// </summary>
        public void Randomize() {
            for (int i = 0; i < _list.Length; i++)
                _list[i] = Random.Next(101);
```

Задача 2

```
        }

        /// <summary>
        /// Печатает элементы списка с их индексами
        /// </summary>
        public void Print() {
            for (int i = 0; i < _list.Length; i++)
                Console.WriteLine(i + ":\t" + _list[i]);
        }
    }



// ReSharper disable AssignNullToNotNullAttribute

/*
 * IntegerListTest.cs
 *
 * Тестирует класс IntegerList
 */



public class IntegerListTest {
    private static IntegerList _list = new IntegerList(10);

    /// <summary>
```

Задача 2

```
/// Создаёт список и выполняет пользовательские операции,
/// пока пользователь не захочет выйти
/// </summary>
public static void Main() {
    PrintMenu();

    int choice = int.Parse(Console.ReadLine());

    while (choice != 0) {
        Dispatch(choice);
        PrintMenu();

        choice = int.Parse(Console.ReadLine());
    }
}

/// <summary>
/// Выполняет действия меню
/// </summary>
/// <param name="choice">Выбранный пункт меню</param>
public static void Dispatch(int choice) {
    switch (choice) {
        case 0:
            Console.WriteLine("Пока!");
            break;
        case 1:
            Console.WriteLine("Какой размер будет у списка?");
            int size = int.Parse(Console.ReadLine());
```

Задача 2

```
                _list = new IntegerList(size);
                _list.Randomize();
                break;
            case 2:
                _list.Print();
                break;
            default:
                Console.WriteLine("Извините, вы выбрали что-то не то");
                break;
        }
    }

    /// <summary>
    /// Выводит варианты пользователю
    /// </summary>
    public static void PrintMenu() {
        Console.WriteLine("\n Меню ");
        Console.WriteLine(" ====");
        Console.WriteLine("0: Выйти");
        Console.WriteLine("1: Создать новый список (** сделайте это с самого
начала!! **)");
        Console.WriteLine("2: Напечатать список");
        Console.Write("\nВведите ваш выбор: ");
    }
}
```

Задача 3

Одно из интересных применений двухмерных массивов это *магические квадраты*. Магический квадрат – это матрица, в которой суммы элементов каждой из строк, каждого из столбцов и обоих диагоналей равны. Магические квадраты изучались многие годы, и мы знаем несколько наиболее известных таких квадратов. В этом задании вам предстоит написать код для определения, является ли квадрат магическим.

Файл *Square.cs* содержит заготовку для класса, представляющего квадратную матрицу. Он содержит заголовки для конструктора, задающего размер матрицы, и для метода, читающего значения в квадрат, печатающего квадрат, находящего сумму отдельной строки/столбца/диагонали и определяющего, является ли квадрат магическим. Метод для считывания вам дан, вам нужно написать все остальные. Учитывайте, что считывающий метод принимает объект типа *TextReader* на вход.

Файл *SquareTest.cs* содержит заготовку программы, читающей значения квадрата из файла *magicData* и сообщающей, является ли квадрат магическим. Заполните оставшиеся методы, поглядывая на комментарии к коду. Учитывайте, что главный метод считывает только размер квадрата, после создания квадрата указанного размера он вызывает метод *ReadSquare* для считывания значений элементов квадрата. Метод *ReadSquare* требует объект типа *TextReader* в качестве параметра.

Вы должны убедиться, что первые три квадрата в файле являются магическим, а остальные нет. Учтите, что значение -1 в конце файла сообщает программе об окончании чтения.

Задача 3

```
/*
 * Square.cs
 *
 * Определяет квадрат с методами по его созданию, заполнению
 * и подсчёту сумм элементов строк, столбцов и диагоналей,
 * и определению принадлежности квадрата к магическим
 */

using System;
using System.IO;


public class Square {
    private int[][] _square;

    /// <summary>
    /// Создаёт новый квадрат указанного размера
    /// </summary>
    /// <param name="size">Размер квадрата</param>
    public Square(int size) { }

    /// <summary>
    /// Возвращает сумму элементов указанной строки
    /// </summary>
    /// <param name="row">Номер строки</param>
    public int SumRow(int row) { }

    /// <summary>
    /// Возвращает сумму элементов указанного столбца
    /// </summary>
    /// <param name="col">Номер столбца</param>
    public int SumCol(int col) { }

    /// <summary>
```

Задача 3

```
        /// Возвращает сумму элементов главной диагонали
        public int SumMainDiag() {}
        /// Возвращает сумму элементов побочной диагонали
        public int SumOtherDiag() {}
        /// <summary>
        /// Возвращает, является ли текущий квадрат магическим
        /// </summary>
        public bool Magic() {}
        /// <summary>
        /// Считывает значения элементов квадрата из консоли
        /// </summary>
        public void ReadSquare(string[] lines, int lineIndex)
        {
            for (int row = 0; row < _square.Length; row++) {
                string[] line = lines[lineIndex+row]
                    .Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);
                if (line.Length != _square.Length)
                        Console.WriteLine($"Ошибка при чтении квадрата: строка должна содержать
{_square.Length} значений, а содержит {line.Length}");
                for (int i = 0; i < _square.Length; i++)
                    int.TryParse(line[i], out _square[row][i]);
            }
        }
        /// <summary>
        /// Выводит аккуратно отформатированное содержимое квадрата
        /// </summary>
        public void PrintSquare()
        { }
    }
```

Задача 3

```
static void Main(string[] args)  {
    string[] lines = System.IO.File.ReadAllLines("..\\..\\magicData.txt");    //
читаем все строки файла в массив
    int lineIndex = 0;  // на какой строке файла находимся
    int count = 0; // считаем, на каком мы сейчас квадрате
    while (lines.Length > lineIndex) {
        int size;   // размер квадрата
        if (!int.TryParse(lines[lineIndex], out size)) {
            Console.WriteLine($"Ошибка при чтении размера квадрата: {lines[lineIndex]}
- не число (строка {lineIndex+1})");
            return;
        }
        if (size == -1)    // в конце файла ожидается -1
            return;
        lineIndex++;
        // TODO: создаём новый квадрат размера size
        // TODO: вызываем метод считывания значений элементов квадрата
        Console.WriteLine($"\n******** Квадрат номер {++count} ********");
        // TODO: выводим квадрат
        // TODO: выводим суммы элеметов его строк
        // TODO: выводим суммы элементов его столбцов
        // TODO: выводим сумму элементов его главной диагонали
        // TODO: выводим сумму элементов его побочной диагонали
        // TODO: определяем и выводим, является ли квадрат магическим
    }
}
```

Задача 3

```
3
8 1 6
3 5 7
4 9 2
7
30 39 48  1 10 19 28
38 47  7  9 18 27 29
46  6  8 17 26 35 37
5  14 16 25 34 36 45
13 15 24 33 42 44  4
21 23 32 41 43  3 12
22 31 40 49  2 11 20
4
48  9  6 39
27 18 21 36
15 30 33 24
12 45 42  3
3
6 2 7
1 5 3
2 9 4
4
3  16  2 13
6   9  7 12
10  5 11  8
15  4 14  1
5
17 24 15  8  1
23  5 16 14  7
4   6 22 13 20
10 12  3 21 19
11 18  9  2 25
7
30 39 48  1 10 28 19
38 47  7  9 18 29 27
46  6  8 17 26 37 35
5  14 16 25 34 45 36
13 15 24 33 42  4 44
21 23 32 41 43 12  3
22 31 40 49  2 20 11
-1
```

Задача 4

Необходимо реализовать класс ArithmeticSequence, представляющий из себя арифметическую прогрессию.
Класс должен содержать следующие элементы (другие члены класса добавлять разрешено):

Поля:
double _start – начальное значение последовательности;
double _increment – разность прогрессии;

Конструкторы:
ArithmeticSequence() – назначает start равным нулю, а increment равным единице;
ArithmeticSequence(double start, double increment) – инициализирует поля класса значениями параметров;

Индексаторы:
double this[int index] – элемент последовательности с порядковым номером index (start имеет порядковый номер 1);

Методы:
string GetInfo() – возвращает строку с информацией о последовательности (разрешается переопределить ToString вместо этого метода);
double GetSum(int n) - находит сумму первых n-членов прогрессии.

В основной программе необходимо создать отдельный объект типа ArithmeticSequence и массив из N объектов типа ArithmeticSequence, где N – случайное число из интервала [5, 15]. Начальное значение последовательности генерировать случайно из диапазона [0, 1000], а разность прогрессии из диапазона [1, 10]. Сгенерировать случайное число step из диапазона [3, 15]. Вывести на экран информацию о последовательностях из массива, у которых элемент с номером step больше, чем у отдельной последовательности. Для каждой последовательности из массива вывести сумму первых step членов.

Соблюдение инкапсуляции и цикл повтора решения обязательны. Обязательно выводить промежуточные значения на экран.

Задача 5

Необходимо реализовать класс VideoFile, представляющий из себя видеофайл. Класс должен содержать следующие элементы (другие члены класса добавлять разрешено):

Поля:
string _name – наименование видеофайла;
int _duration – длительность в секундах;
int _quality – качество видеофайла;

Конструкторы:
VideoFile(string name, int duration, int quality) – инициализирует поля класса значениями параметров;

Свойства:
int Size – размер видеофайла (свойство доступа);
Будем считать размер как произведение длительности на качество.

Методы:
string GetInfo() – возвращает строку с информацией о видеофайле (разрешается переопределить ToString вместо этого метода);

В основной программе необходимо создать отдельный объект типа VideoFile и массив из N объектов типа VideoFile, где N – случайное число из интервала [5, 15]. Длительность генерировать случайно из диапазона [60, 360], а качество из диапазона [100, 1000], наименование каждого видеофайла определить самостоятельно (генерировать случайную строку латинских символов длины от 2 до 9). Вывести на экран информацию о видеофайлах из массива, размер которых больше, чем размер отдельного видеофайла.

Соблюдение инкапсуляции и цикл повтора решения обязательны. Обязательно выводить промежуточные значения на экран.

Задача 6

Реализовать класс Библиотека (Library), предоставляющий доступ к коллекции книг (класс Book) через индексатор.

Предусмотреть:
- возможность создания как пустой библиотеки (конструктор без параметров), так и из готовой коллекции книг.
- метод добавления книги в библиотеку (AddBook)
- свойство, возвращающее количество книг в библиотеке (BooksCount)
- метод с целым параметром n, возвращающий книги с количеством страниц меньшим, чем n (CountBooksWithTheLessAmountOfPages)
- метод string GetInfo(), который возвращает строку с информацией˘о библиотеке (разрешается переопределить ToString вместо этого метода);

Каждая книга имеет следующие поля:
- количество страниц (_countPages)
- номер секции в библиотеке (_sectionNumber)

Предусмотреть невозможность изменения полей книги после её создания.
Предусмотреть наличие конструктора в классе Book.
Предусмотреть метод string GetInfo(), который возвращает строку с информацией˘о библиотеке (разрешается переопределить ToString вместо этого метода);

В самой программе необходимо создать библиотеку и заполнить её N (от 10 до 20) случайно сгенерированными книгами (количество страниц может быть от 1 до 500, в библиотеке от 5 до 10 секций). Необходимо вывести все книги и книги с количеством страниц меньше 200.

Задача 7

Создать класс, описывающий точку в трехмерном пространстве (Point). Координаты точки – поля класса.

Описать конструктор без параметров и конструктор с параметрами – координаты точки.

Предусмотреть свойства доступа к полям класса.

Описать метод вычисляющий расстояние от точки (объекта класса) до точки, координаты которой переданы в параметрах метода.

В основной программе создать три объекта класса и вывести расстояние от этих точек до начала координат.

Задача 7

Задание: Изменить основную программу!
В основной программе необходимо создать массив из N объектов типа Point, где N – случайное число из интервала [5, 15]. Координаты точек заполнить случайными числами в интервале [-10, 10]. Вывести на экран информацию обо всех объектах массива, а также расстояние от каждой точки до начала координат.
Найти точку, наиболее удаленную от начала координат и вывести информацию о ней на экран, а также расстояние до точки.

Соблюдение инкапсуляции и цикл повтора решения обязательны. Обязательно выводить промежуточные значения на экран.

Задача 8

Создать класс, описывающий треугольник на плоскости (Triangle).

Треугольник задаётся координатами вершин (поля класса типа Point. Point – точка в плоскости, по аналогии с задачей 7).

Описать конструктор без параметров и два конструктора с параметрами – координаты вершин или точки на плоскости.

Предусмотреть свойства доступа полям класса.
Описать свойства «периметр» и «площадь» треугольника.

В основной программе необходимо создать массив из N объектов типа Triangle, где N – случайное число из интервала [5, 15]. Координаты точек треугольника заполнить случайными числами в интервале [-10, 10]. Вывести на экран информацию обо всех объектах массива.

Отсортировать массив треугольников по убыванию их площади.

Соблюдение инкапсуляции и цикл повтора решения обязательны. Обязательно выводить промежуточные значения на экран.

Задача 9

Описать класс LinearEquation, соответствующую линейному уравнению.
Уравнение $ax + b = c$ задаётся коэффициентами a, b и c.
Определить метод, находящий корень линейного уравнения.

Разрешается добавлять члены классов, необходимые для реализации программы.

В основной программе необходимо создать массив из N объектов типа LinearEquation, где N – число, введенное пользователем с клавиатуры. Координаты коэффициентов a, b и c генерируются случайным образом в интервале [-10; 10].

Вывести на экран информацию обо всех объектах массива.

Отсортировать массив по возрастанию корней линейного уравнения.

Соблюдение инкапсуляции и цикл повтора решения обязательны. Обязательно выводить промежуточные значения на экран.

Задача 10

Описать класс Circle, соответствующую окружностям.
Окружность задаётся координатами центра и радиусом.

Определить функцию, проверяющую, пересекаются ли две окружности.

Разрешается добавлять члены классов, необходимые для реализации программы.

В основной программе необходимо создать массив из N объектов типа Circle, где N – число, введенное пользователем с клавиатуры. Координаты центра и радиуса генерируются случайным образом в интервале [1; 15].

Также создать отдельный объект класса Circle.

Вывести на экран информацию обо всех объектах массива.

Вывести информацию о тех объектах массива, которые пересекаются с отдельно созданным объектом класса Circle.

Соблюдение инкапсуляции и цикл повтора решения обязательны. Обязательно выводить промежуточные значения на экран.

Задача 11

Описать класс GeometricProgression, соответствующий геометрической прогрессии.

Класс должен содержать следующие элементы (другие члены класса добавлять разрешено):

Поля:
double _start – начальное значение последовательности;
double _increment – знаменатель прогрессии;

Конструкторы:
GeometricProgression() – назначает start равным нулю, а increment равным единице;
GeometricProgression(double start, double increment) – инициализирует поля класса значениями параметров;

Индексаторы:
double this[int index] – элемент последовательности с порядковым номером index (start имеет порядковый номер 1);

Методы:
string GetInfo() – возвращает строку с информацией о последовательности (разрешается переопределить ToString вместо этого метода);
double GetSum(int n) - находит сумму первых n-членов геометрической прогрессии.

В основной программе необходимо создать отдельный объект типа GeometricProgression и массив из N объектов типа GeometricProgression, где N – случайное число из интервала [5, 15]. Начальное значение последовательности генерировать случайно из диапазона [0, 10], а знаменатель прогрессии из диапазона (0, 5]. Сгенерировать случайное число step из диапазона [3, 15]. Вывести на экран информацию о последовательностях из массива, у которых элемент с номером step больше, чем у отдельной последовательности. Для каждой последовательности из массива вывести сумму первых step членов.

Соблюдение инкапсуляции и цикл повтора решения обязательны. Обязательно выводить промежуточные значения на экран.