

Технически Университет Габрово-гр. Габрово
Факултет ЕЕ катедра КСТ

КУРСОВА РАБОТА

Тема: Създаване на клас Workout за проследяване
на фитнес активност

Разработил: Ваня Тодорова Ванева

ф. номер: 22472128

курс: Първи

специалност: СКИ

Дата:

1. Теоретична част

1.1. Разлика между методи и конструктори в C#

Метод - блок от код, който изпълнява определена задача. Той се дефинира вътре в клас и може да приема параметри, да връща стойности и да бъде извикван многократно. Ползва се за обработка на данни, изпълнение на действия, организиране на кода в по-малки логически части.

Конструктор - специален метод, който се извиква автоматично при създаване на нов обект от клас. Има същото име като класа. Няма тип на връщане. Може да има параметри. Основната му задача е да инициализира полетата на обекта.

Разлики:	Метод	Конструктор
Име	Произволно	Същото име като класа
Тип на връщане	Има (напр. void, int, ...)	Няма
Извикване	Ръчно	Автоматично
Предназначение	Изпълнява действия или изчисления	Инициализира стойностите на полетата
Многократно извикване	Да	Да, но само при създаване на нов обект

Таблица 1

1.2. Обработка на масиви в C#

Масив в C# е структура от данни, която съдържа фиксиран брой елементи от един и същ тип. Масивите се използват за съхранение и обработка на множество стойности под общо име. Масивите в C# започват от индекс 0, т.е. първият елемент е достъпен чрез индекс 0, вторият чрез 1 и т.н.

1.2.1. Деклариране и инициализиране на масив

```
int[] numbers = new int[5]; // масив с 5 елемента (стойности по подразбиране)
```

```
string[] names = { "Иван", "Мария", "Петър" }; // инициализиран с конкретни стойности
```

1.2.2. Обхождане на масив

C for цикъл:

```
for (int i = 0; i < numbers.Length; i++)  
{  
    Console.WriteLine(numbers[i]);  
}
```

C foreach:

```
foreach (string name in names)  
{  
    Console.WriteLine(name);  
}
```

1.2.3. Типични операции върху масиви

Присвояване на стойност	<code>numbers[0] = 42;</code>
Достъп до елемент	<code>Console.WriteLine(numbers[1]);</code>
Дължина на масив	<code>int len = numbers.Length;</code>
Сумиране на елементи	<code>int sum = numbers.Sum();</code>
Сортиране	<code>Array.Sort(numbers);</code>
Обръщане на реда	<code>Array.Reverse(numbers);</code>

Таблица 2

1.3. Използвана литература

Светлин Наков и колектив. Основи на програмирането със C#. Faber Publishing, София, май 2017 г.

Светлин Наков, Веселин Колев и колектив. Принципи на програмирането със C#. Фабер, Велико Търново, 2018 г.

Конструктор (обектно ориентирано програмиране). Уикипедия: Свободната енциклопедия. Последна редакция на 31 март 2025 г. Дата на достъп: 2 май 2025 г. Достъпно на:

[https://bg.wikipedia.org/wiki/Конструктор_\(обектно_ориентирано_програмиране\)](https://bg.wikipedia.org/wiki/Конструктор_(обектно_ориентирано_програмиране))

2. Практическо изпълнение

2.1. Програмен код

```
using System;
namespace courseWork
{
    public class Workout
    {
        private string name, type;
        private int duration, calories;
        public Workout(string name, string type, int duration, int
            calories)
        {
            this.name = name;
            this.type = type;
            this.duration = duration;
            this.calories = calories;
        }
        public string Name { get { return name; } set { name = value;
        } }
        public int Calories { get { return calories; } set { calories
            = value; } }
        public void Output()
        {
            Console.WriteLine($"{name} - {type} - {duration} mins -
                {calories} Kcals");
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter number of workouts for the week: ");
            int n = int.Parse(Console.ReadLine());
            Workout[] workouts = new Workout[n];
            for(int i = 0; i < n; i++)
            {
                Console.WriteLine($"{n}\nEnter details for workout {i +
                    1}");
                Console.Write("\tName: ");
                string name = Console.ReadLine();
                Console.Write("\tType: ");
                string type = Console.ReadLine();
            }
        }
    }
}
```

```
        Console.Write("\tDuration(in mins): ");
        int duration = int.Parse(Console.ReadLine());
        Console.Write("\tKcals burned: ");
        int calories = int.Parse(Console.ReadLine());
        workouts[i] = new Workout(name, type, duration,
        calories);
        Console.WriteLine();
    }
    CalsAsc(workouts);
    Workout maxCals = workouts[workouts.Length - 1];
    double avgCals = AvgCals(workouts);
    Console.WriteLine($"Burned most Kcals during {maxCals.Name} -
    {maxCals.Calories}!");
    Console.WriteLine($"Average Kcals burned - {avgCals:F2}\n");
    Console.WriteLine("List of this week's workouts");
    Console.WriteLine("\tname - type - duration - calories");
    foreach(var w in workouts)
    {
        w.Output();
    }
}
static void CalsAsc(Workout[] workouts)
{
    Array.Sort(workouts, (w1, w2) =>
        w1.Calories.CompareTo(w2.Calories));
}
static double AvgCals(Workout[] workouts)
{
    double total = 0;
    foreach(var w in workouts)
    {
        total += w.Calories;
    }
    return total / workouts.Length;
}
}
```

2.2. Описание на решението

Клас `Workout` съдържа частните полета име, тип, продължителност и калории. Използва конструктор, който инициализира всички полета при създаване на нов обект. Има свойства само за име и калории и метод за извеждане на информация за тренировка.

В главния метод се създава масив от обекти `Workout`, като размерът се определя чрез вход от конзолата. За всяка тренировка се въвеждат необходимите данни (име, тип, продължителност и калории), които се използват за създаване на нов обект от тип `Workout`. Обектът се съхранява в масива.

Методът `CalsAsc()` сортира масива по броя изгорени калории във възходящ ред, използвайки `Array.Sort()` с `lambda` израз:

```
Array.Sort(workouts, (w1, w2) => w1.Calories.CompareTo(w2.Calories));
```

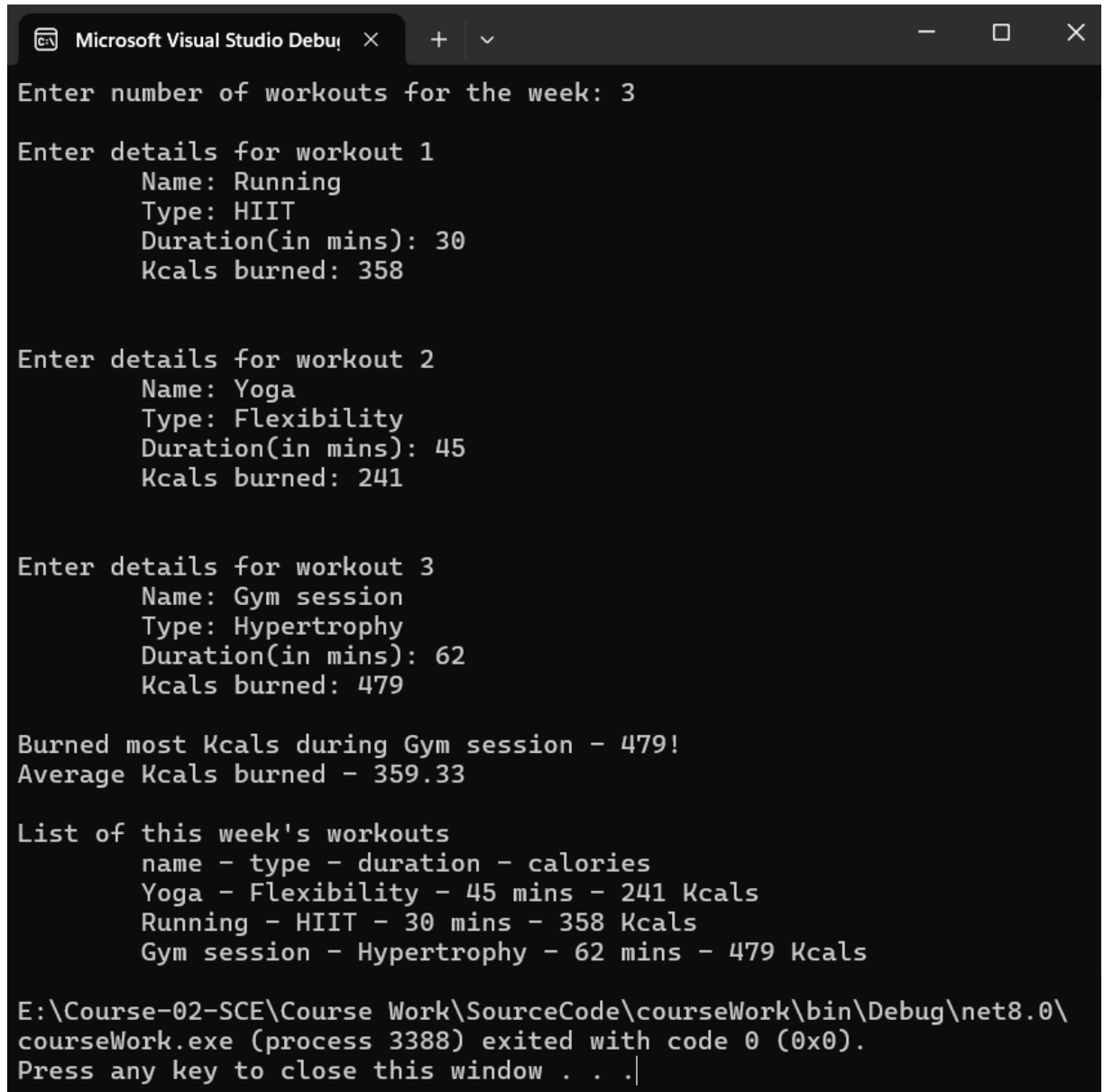
Този израз сравнява стойностите на калориите между два обекта `Workout` и ги подрежда по нарастваща стойност. Така тренировката с най-малко калории се оказва първа, а тази с най-много – последна.

Най-голямото количество изгорени калории се определя чрез последния елемент в сортирания вече масив (`workouts[workouts.Length - 1]`).

Методът `AvgCals` изчислява средния брой изгорени калории за седмицата. Той използва цикъл `foreach`, с който се обхождат всички елементи от масива и се събират стойностите на калориите. След като се изчисли общата сума, тя се разделя на броя тренировки, за да се получи средната стойност.

След сортирането се отпечатва тренировката с най-много изгорени калории, средните калории за седмицата и списък с всички въведени тренировки.

2.3. Екрани



```
Microsoft Visual Studio Debug Console
Enter number of workouts for the week: 3

Enter details for workout 1
  Name: Running
  Type: HIIT
  Duration(in mins): 30
  Kcals burned: 358

Enter details for workout 2
  Name: Yoga
  Type: Flexibility
  Duration(in mins): 45
  Kcals burned: 241

Enter details for workout 3
  Name: Gym session
  Type: Hypertrophy
  Duration(in mins): 62
  Kcals burned: 479

Burned most Kcals during Gym session - 479!
Average Kcals burned - 359.33

List of this week's workouts
  name - type - duration - calories
  Yoga - Flexibility - 45 mins - 241 Kcals
  Running - HIIT - 30 mins - 358 Kcals
  Gym session - Hypertrophy - 62 mins - 479 Kcals

E:\Course-02-SCE\Course Work\SourceCode\courseWork\bin\Debug\net8.0\
courseWork.exe (process 3388) exited with code 0 (0x0).
Press any key to close this window . . .|
```