ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ГАБРОВО

Катедра „Компютърни системи и технологии“

КУРСОВА РАБОТА

Тема:.....

Разработил:.....

Фак. №:..... Курс:..... Специалност:.....

Проверил:.....

Подпис:..... Дата:.....

Теоретична част:

Деклариране на класове в C#

В C#, класовете се използват за дефиниране на обекти, които могат да съдържат данни (полета) и функционалности (методи). Декларирането на клас става по следния начин:

```
public class ExampleClass
{
    // Полета (променливи) на класа
    public int intVar;

    // Методи на класа
    public void ExampleMethod()
    {
        // Тяло на метода
    }
}
```

- ✓ *public* е модификатор за достъп, който указва, че класът е достъпен от всички части на програмата. В C# има и други модификатори за достъп като *private, protected,* и *internal* които определят видимостта на класа.
- ✓ *class* е ключова дума, която обозначава, че следващият блок код е дефиниция на клас.
- ✓ *ExampleClass* е името на класа. Ще достъпваме класа чрез това име.

Деклариране на методи в C#

Методите в C# представляват функции, които изпълняват определени операции. Те могат да бъдат дефинирани в рамките на клас или структура. Декларирането на метод става по следния начин:

```
public void ExampleMethod()
{
   // Тяло на метода
}
```

- ✓ *public* отново е модификатор за достъп. Тук той определя, че методът е достъпен извън класа.
- ✓ *void* е типът на връщане на метода. В случая *void* означава, че методът не връща стойност. Има определени методи, които връщат стойност но според типа метод. Примерно има типове методи които връщат данни от тип *int*, *double*, *string* и др.
- ✓ *ExampleMethod* е името на метода.

Реализация на методи и класове

Реализацията на методите и класовете в C# се извършва чрез написване на код в тялото на методите и класовете съответно. Всяка функционалност, която искаме да имплементираме, трябва да бъде описана в тялото на съответния метод/клас.

Обработка на грешки в C#

За обработка на грешки в C# се използва механизмът на изключенията (*exceptions*). Изключенията се хвърлят, когато възникне някаква неочаквана ситуация по време на изпълнение на програмата. За да се хвърли изключение, използваме ключовата дума *throw*. Например:

```
public void MyMethod()
{
    if (anyCondition)
    {
        throw new Exception("Something went wrong!");
    }
}
```

И за да се улови изключение, използваме блока *try-catch*:

```
try
{
    // Изпълнява се код, който може да създаде изключение
}
catch (Exception e)
{
    // Тяло на изключението
    // Тук обикновено се извежда в точност грешката
}
```

Има още един блок от операции *finally*, който се използва за дефиниране на код, който трябва да се изпълни винаги, независимо дали е хвърлено изключение или не. Важно е да се отбележи, че блокът *finally* е опционален и може да бъде използван самостоятелно без *catch*. Общия вид е:

```
try
{
    // Код, който може да хвърли изключение
}
catch (Exception ex)
{
    // Тяло на изключението
}
finally
{
    // Код, който се изпълнява винаги, независимо от това дали е създадено
изключение или не
}
```

Обработка на масиви в C#

В C# масивите са структури, които позволяват съхранение на няколко елемента от един и същи тип в една променлива. Масивите в C# се декларират по следния начин:

```
int[] myArray = new int[5];
```

Тук *int*[] указва, че става въпрос за масив от цели числа, а *new int*[*5*] заделя памет за масив с 5 елемента. За обработка на масивите се използват цикли, като *for*, *foreach* и други, както и стандартните операции за достъп до елементите на масива.

```
using System;
namespace KursRab
{
	public class Hotel
	{
		protected string name;
		protected string address;
		protected double[] gpsCoords = new double[2];
		protected string[] events = new string[5];
		protected string[] eventDates = new string[5];
		public Hotel()
		{
			name = "";
			address = "";
			gpsCoords[0] = 0.0;
			gpsCoords[1] = 0.0;
		}
		public Hotel(string n, string a, double gps1, double gps2)
		{
			this.name = n;
			this.address = a;
			this.gpsCoords[0] = gps1;
			this.gpsCoords[1] = gps2;
		}
		public string Name
		{
			get { return name; }
			set { name = value; }
		}
		public string Address {
			get { return address; }
			set {  address = value; }
		}
		public double[] GpsCoords
		{
			get { return gpsCoords; }
			set { gpsCoords = value; }
		}
		public void Input()
		{
			Console.WriteLine("Enter the hotel name:");
			Name = Console.ReadLine();
			Console.WriteLine("Enter the hotel address:");
			Address = Console.ReadLine();
			Console.WriteLine("Enter the hotel gps coordinate on x:");
			GpsCoords[0] = double.Parse(Console.ReadLine());
			Console.WriteLine("Enter the hotel gps coordinate on y:");
			GpsCoords[1] = double.Parse(Console.ReadLine());
		}
		public void EventInput(int arrNumb)
		{
			Console.WriteLine("Enter the event title:");
			events[arrNumb] = Console.ReadLine();
			Console.WriteLine("Enter the event date:");
			eventDates[arrNumb] = Console.ReadLine();
		}
```

```
            public string Output()
            {
                  return String.Join(" ", name, address, gpsCoords[0],
gpsCoords[1]);
            }
            public void EventOutput (int arrNumb)
            {
                  Console.WriteLine("Current events at " + arrNumb + " hotel.");
                  Console.WriteLine(String.Join(" ", events[arrNumb],
eventDates[arrNumb]));
            }
            public static string FindHotel(string hotelName, Hotel hotel)
            {
                  if (hotelName == hotel.Name)
                  {
                        return String.Join(" ", hotel.name, hotel.address,
hotel.GpsCoords[0], hotel.GpsCoords[1]);
                  } else
                  {
                        return " ";
                  }
            }

      }
}

using System;

namespace KursRab
{
      internal class Program {
            static void Main(string[] args)
            {
                  Hotel[] hotels = new Hotel[5];
                  Console.WriteLine("List of commands: \n Input \n Output \n
Event Input \n Event Output \n Check Hotel \n End");
                  string command = Console.ReadLine();
                  while (command != "End") {
                        switch (command) {
                              case "Input":
                                    Console.Clear();
                                    for (int i = 0; i < hotels.Length; i++) {
                                          hotels[i] = new Hotel();
                                          hotels[i].Input();
                                    }
                                    Console.WriteLine("List of commands: \n
Input \n Output \n Event Input \n Event Output \n Check Hotel \n End");
                                    command = Console.ReadLine();
                                    break;
                              case "Output":
                                    Console.Clear();
                                    for (int i = 0; i < hotels.Length; i++) {
                                         Console.WriteLine(hotels[i].Output());
                                    }
                                    Console.WriteLine("List of commands: \n
Input \n Output \n Event Input \n Event Output \n Check Hotel \n End");
                                    command = Console.ReadLine();
                                    break;
```

```
                                case "Event Input":
                                      Console.Clear();
                                      for (int i = 0; i < hotels.Length; i++)
                                      {
                                            hotels[i].EventInput(i);
                                      }
                                      Console.WriteLine("List of commands: \n
Input \n Output \n Event Input \n Event Output \n Check Hotel \n End");
                                      command = Console.ReadLine();
                                      break;
                                case "Event Output":
                                      Console.Clear();
                                      for (int i = 0; i < hotels.Length; i++) {

                                            hotels[i].EventOutput(i);
                                      }
                                      Console.WriteLine("List of commands: \n
Input \n Output \n Event Input \n Event Output \n Check Hotel \n End");
                                      command = Console.ReadLine();
                                      break;
                                case "Check Hotel":
                                      Console.Clear();
                                      string hotelName = "";
                                      Console.WriteLine("Enter the name of the
hotel:");
                                      hotelName = Console.ReadLine();
                                      for (int i = 0; i < hotels.Length; i++) {
                                            Console.WriteLine(Hotel.FindHotel(
hotelName, hotels[i]));
                                      }
                                      Console.WriteLine("List of commands: \n
Input \n Output \n Event Input \n Event Output \n Check Hotel \n End");
                                      command = Console.ReadLine();
                                      break;
                                case "End":
                                      return;
                                default:
                                      Console.Clear();
                                      Console.WriteLine("Command not found!");
                                      Console.WriteLine("List of commands: \n
Input \n Output \n Event Input \n Event Output \n Check Hotel \n End");
                                      command = Console.ReadLine();
                                      break;
                          }
                    }
              }
        }
}
```

Резултат от програмния код:

```
Enter the hotel name:
Western Cave Resort
Enter the hotel address:
4263 Cameron Road
Enter the hotel gps coordinate on x:
217,23
Enter the hotel gps coordinate on y:
35,28
Enter the hotel name:
Sunrise Shroud Hotel
Enter the hotel address:
755 Barrington Court
Enter the hotel gps coordinate on x:
259,27
Enter the hotel gps coordinate on y:
52,76
Enter the hotel name:
Ancient Shore Hotel & Spa
Enter the hotel address:
746 Marion Drive
Enter the hotel gps coordinate on x:
176,34
Enter the hotel gps coordinate on y:
43,27
Enter the hotel name:
Nimbus Resort
Enter the hotel address:
1873 Emma Street
Enter the hotel gps coordinate on x:
201,47
Enter the hotel gps coordinate on y:
32,14
Enter the hotel name:
Mirage Hotel
Enter the hotel address:
3348 Skinner Hollow Road
Enter the hotel gps coordinate on x:
304,17
Enter the hotel gps coordinate on y:
34,78
List of commands:
 Input
 Output
 Event Input
 Event Output
 Check Hotel
 End
```

```
Western Cave Resort 4263 Cameron Road 217,23 35,28
Sunrise Shroud Hotel 755 Barrington Court 259,27 52,76
Ancient Shore Hotel & Spa 746 Marion Drive 176,34 43,27
Nimbus Resort 1873 Emma Street 201,47 32,14
Mirage Hotel 3348 Skinner Hollow Road 304,17 34,78
List of commands:
 Input
 Output
 Event Input
 Event Output
 Check Hotel
 End
Enter the name of the hotel:
Ancient Shore Hotel & Spa


Ancient Shore Hotel & Spa 746 Marion Drive 176,34 43,27


List of commands:
 Input
 Output
 Event Input
 Event Output
 Check Hotel
 End
```