Лабораторно упражнение № 13

Шаблонни типове в език C#

I. Теоретична част

1. Същност и деклариране на шаблонен тип в C#

Много често възниква проблемът програмистът да създава класове, които са сходни по функционалност, а се различават само по типа на обектите, с които работят. Например, необходимо е да се изгради списък като елементите на списъка са цели числа. В този случай полетата в класа ще са от целочислен тип. Методите, който биха били включени в подобен клас като добавяне, изтриване и търсене на елемент в списъка, биха работили с променливи от целочислен тип. Ако същата задача бъде поставена за изграждане на списък от дробно-десетични стойности или на низове, структурата и функционалността на следващите два класа ще бъдат идентични с тези на първия клас като единствената разлика ще бъде в типовете на данните.
В подобни случаи възможностите на език C# е предлагат създаването на шаблонни типове (generics). Чрез шаблонните типове функционалността на един клас може да бъде приложена към множество обекти от различни типове. Шаблонните типове са класове, структури, интерфейси и методи, на които се предават като параметри типове. В последствие, при инстанцирането (създаването на програмни обекти) на класове и структури, ще бъдат използвани конкретните типове, които се предават като параметри.

Декларацията на типизиран (шаблонен) клас в C# има следния общ вид:

```
class ИмеТипизиранКлас <Type>
{

}
```

Type се явява идентификатор, който съответства на параметър на тип. Този тип е неизвестен по време на декларацията на класа.

На практика, типизирането на клас т.е. създаването на шаблонен клас е чрез добавянето на параметър към декларацията, съответстващ на неизвестен тип. Едва при инстанцирането на класа този параметър ще бъде заменен с конкретен тип.

Пример е декларацията на шаблонния клас Point, описващ точка в равнината.

```
public class Point<T>
  {
      private T x, y;
      public T X
      {
          get { return x; }
          set {x=value;}
      }
      public T Y
      {
          get { return y; }
          set { y = value; }
      }
```

```
        public void Info()
        {
            Console.WriteLine("Generic Type: {0}", this.x.GetType());
            Console.WriteLine("x={0}, y={1}", x, y);
        }
    }
```

При създаването на обекти, шаблонният клас се инстанцира. В примера са създадени два обекта от класа Point, тако координатите на първата точка са цели числа, а на втората – стойности от тип double.

```
Point<int> p1 = new Point<int>();
p1.X = 100;
p1.Y = 100;
p1.Info();

Point<double> p2 = new Point<double>();
p2.X = 10.01;
p2.Y = 100.12;
p2.Info();
```

2. Типизиране на методи

Подобно на класовете методите също могат да бъдат типизирани. Типизирането на метод се налага когато не е известно от какъв тип ще са параметрите на метода и означава, че в него ще бъдат използвани неизвестни типове. Едва при извикването на метода се определя кои ще бъдат неизвестните типове като типовете параметри се заменят с конкретни.
Синтаксисът на типизирането на метод е следният:

Модификатор тип ИмеМетод <K> (списък параметри)
{
}

В случая K е параметър на тип, който при дефиницията на метода е неизвестен.

Пример:

```
public static void Swap<K>(ref K a, ref K b)
  {
      K c;
      c = a;
      a = b;
      b = c;
  }
```

Извикването на типизиран метод е като след името на метода се задава конкретния тип:

```
int x = 10, y = 20;
Swap <int>(ref x, ref y);

string s1 = "Hello";
string s2 = "World";
Swap<string>(ref s1, ref s2);
```

Компилаторът може да определи вида на типа по конкретните стойности на параметрите и затова не е необходимо при извикването на типизиран метод да се

определя вида на типа. Съответно, програмните редове могат да се запишат и по този начин:

```
int x = 10, y = 20;
Swap(ref x, ref y);

string s1 = "Hello";
string s2 = "World";
Swap(ref s1, ref s2);
```

II. Задачи за изпълнение

1. Да се създаде типизиран клас Rectangle, описващ правоъгълна област. Класът да съдържа полета, описващи размерите на областта и свойства за четене и запис за връзка с тези полета. Класът да съдържа метод, който намира лицето на правоъгълника. Да се създаде конзолно приложение за работа с класа, в което се дефиниат обекти, съответстващи на правоъгълни области, чиито размери са целочислени стойности и стойности от реален и реален тип с двойна точност. Да се добави конструктор с параметри, които дават стойности на полетата. Да се направят необходимите изменения в програмния код.
2. Да се създаде типизиран клас, описващ масив. Класът да съдържа конструктор с един параметър, съответстващ на размера на масива; метод за четене и метод за промяна стойността на елемент. Да се създаде демонстрационна програма за работа с класа.
3. Да се създаде типизиран клас, описващ стек. Класът да съдържа конструктор с един параметър, съответстващ на размера на стека; метод за четене на елемент от стека и метод за запис на елемент в стека; метод за проверка за пълен и метод за проверка за празен стек. Да се създаде демонстрационна програма за работа с класа.