Московский государственный технический
университет имени Н.Э. Баумана

Парадигмы и конструкции языков программирования
Отчёт по лабораторной работе №3

Работу выполнил
Студент группы ИУ5-35Б
Шиленок Д.А.

2023 г.

Задание

Разработать программу, реализующую работу с коллекциями.

1. Программа должна быть разработана в виде консольного приложения на языке C#.

2. Создать объекты классов «Прямоугольник», «Квадрат», «Круг».

3. Для реализации возможности сортировки геометрических фигур для класса «Геометрическая фигура» добавить реализацию интерфейса IComparable. Сортировка производится по площади фигуры.

4. Создать коллекцию класса ArrayList. Сохранить объекты в коллекцию. Отсортировать коллекцию. Вывести в цикле содержимое коллекции.

5. Создать коллекцию класса List. Сохранить объекты в коллекцию. Отсортировать коллекцию. Вывести в цикле содержимое коллекции.

6. Модифицировать класс разреженной матрицы (проект SparseMatrix) для работы с тремя измерениями – x,y,z. Вывод элементов в методе ToString() осуществлять в том виде, который Вы считаете наиболее удобным. Разработать пример использования разреженной матрицы для геометрических фигур.

7. Реализовать класс «SimpleStack» на основе односвязного списка. Класс SimpleStack наследуется от класса SimpleList (проект SimpleListProject). Необходимо добавить в класс методы:

- public void Push(T element) – добавление в стек;
- public T Pop() – чтение с удалением из стека.

8. Пример работы класса SimpleStack реализовать на основе геометрических фигур.

Текст программы

Delegates.cs (Работа с делегатом)

```
namespace Lab5_Delegates
{
   delegate double MathFunction(double p1, int p2);
   internal class Delegates
   {
      static double Pow(double x, int n) { return Math.Pow(x, n); }
      static double Root(double x, int n) { return Math.Pow(x, 1.0d / n); }

      static void MathFunctionMethodGeneric(double x, int n, Func<double, int, double>
MathFunctionParam)
      {
         double Result = MathFunctionParam(x, n);
         Console.WriteLine(Result);
      }
      static void MathFunctionMethod(double x, int n, MathFunction MathFunctionParam)
      {
         double Result = MathFunctionParam(x, n);
         Console.WriteLine(Result);
      }
      static void Main(string[] args)
      {
         Console.WriteLine("Возведение 4.5 в степень 3:");
         MathFunctionMethod(4.5, 3, Pow);
         Console.WriteLine("Корень 3 степени из числа 15.625:");
         MathFunctionMethod(15.625, 3, Root);
         Console.WriteLine("Возведение 2.5 в степень 3, аргумент – лямбда-выражение:");
         MathFunctionMethod(2.5, 3, (x, n) => Math.Pow(x, n));

         Console.WriteLine("\n\t\tИспользование обобщенного делегата Func<double, int,
double>\n");

         Console.WriteLine("Возведение 4.5 в степень 3:");
         MathFunctionMethodGeneric(4.5, 3, Pow);
         Console.WriteLine("Корень 3 степени из числа 15.625:");
         MathFunctionMethodGeneric(15.625, 3, Root);
         Console.WriteLine("Возведение 2.5 в степень 3, аргумент – лямбда-выражение:");
         MathFunctionMethodGeneric(2.5, 3, (x, n) => Math.Pow(x, n));
      }
   }
}
```

Program.cs (Работа с рефлексией)

```
using System.ComponentModel.DataAnnotations;
using System.Reflection;

namespace Lab5_Reflection
{
   internal class Program
   {
      public static bool GetPropertyAttribute(PropertyInfo checkType, Type attributeType, out object?
attribute)
      {
         bool Result = false;
         attribute = null;

         //Поиск атрибутов с заданным типом
         var isAttribute = checkType.GetCustomAttributes(attributeType, false);
         if (isAttribute.Length > 0)
         {
            Result = true;
            attribute = isAttribute[0];
         }

         return Result;
      }
      static void Main(string[] args)
      {
         Class ClassObject = new Class();
         Type t = ClassObject.GetType();

         Console.WriteLine("\nКонструкторы:");
         foreach (var constructor in t.GetConstructors())
         {
            Console.WriteLine(constructor);
         }

         Console.WriteLine("\nМетоды:");
         foreach (var method in t.GetMethods())
         {
            Console.WriteLine(method);
         }

         Console.WriteLine("\nСвойства:");
         foreach (var property in t.GetProperties())
         {
            Console.WriteLine(property);
         }

         Console.WriteLine("\nПоля:");
         foreach (var field in t.GetFields())
         {
            Console.WriteLine(field);
```

```
        }

        Console.WriteLine("\nСвойства, помеченные атрибутом:");
        foreach (var x in t.GetProperties())
        {
           if (GetPropertyAttribute(x, typeof(MyAttribute), out object? attributeObject))
           {
              MyAttribute? attribute = attributeObject as MyAttribute;
              Console.WriteLine(x.Name + " - " + attribute?.Description);
           }
        }

        Console.WriteLine("\nВызов метода:");
        Class? fi = (Class?)t.InvokeMember(string.Empty, BindingFlags.CreateInstance, null, null,
Array.Empty<object>());
        object[] parameters = new object[] { 15, 4 };
        object? Result = t.InvokeMember("Minus", BindingFlags.InvokeMethod, null, fi, parameters);
        Console.WriteLine("Minus(15,4) = " + Result);
     }
  }
}
```

MyAttribute.cs

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

        namespace Lab5_Reflection
        {
           [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = false)]
           public class MyAttribute : Attribute
           {
              public MyAttribute()
              {
                 Description = string.Empty;
              }
              public MyAttribute(string DescriptionParam)
              {
                 Description = DescriptionParam;
              }

              public string Description { get; set; }
           }
        }
```

Class.cs

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

namespace Lab5_Reflection
{
   internal class Class
   {
      public Class() { }
      public Class(int i) { }
      public Class(double d) { }

      public int Plus(int x, int y) { return x + y; }
      public int Minus(int x, int y) { return x - y; }

      [MyAttribute("Property1 attribute description")]
      public string Property1 { get; set; }
      public int Property2 { get; set; }

      [MyAttribute("Property3 attribute description")]
      public double Property3 { get; private set; }

      public int field1;
      public float field2;
   }
}
```

Вывод программы

```
Возведение 4.5 в степень 3:
91,125
Корень 3 степени из числа 15.625:
2,5
Возведение 2.5 в степень 3, аргумент - лямбда-выражение:
15,625

                Использование обобщенного делегата Func<double, int, double>

Возведение 4.5 в степень 3:
91,125
Корень 3 степени из числа 15.625:
2,5
Возведение 2.5 в степень 3, аргумент - лямбда-выражение:
15,625
```

```
Конструкторы:
Void .ctor()
Void .ctor(Int32)
Void .ctor(Double)

Методы:
Int32 Plus(Int32, Int32)
Int32 Minus(Int32, Int32)
System.String get_Property1()
Void set_Property1(System.String)
Int32 get_Property2()
Void set_Property2(Int32)
Double get_Property3()
System.Type GetType()
System.String ToString()
Boolean Equals(System.Object)
Int32 GetHashCode()

Свойства:
System.String Property1
Int32 Property2
Double Property3

Поля:
Int32 field1
Single field2

Свойства, помеченные атрибутом:
Property1 - Property1 attribute description
Property3 - Property3 attribute description

Вызов метода:
Minus(15,4) = 11
```