МГТУ им. Н.Э. Баумана

Факультет «Информатика и системы управления»

ДИСЦИПЛИНА:

«Базовые компоненты интернет технологий»

Отчёт по лабораторной работе №3

|  |
| --- |
| Выполнил:  Студент 2 курса  Факультет ИУ  Группа  ИУ5-32  Громов В.В.  Преподаватель  Гапанюк Ю.Е. |

**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<Figure>. 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.

**Код программы.**

Program.cs

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Collections; |
|  |  |
|  | namespace Lab3 |
|  | { |
|  | class Program |
|  | { |
|  | static void Main\_menu() |
|  | { |
|  | Console.WriteLine(); |
|  | Console.WriteLine("MENU"); |
|  | Console.WriteLine(); |
|  | Console.WriteLine("1.Work with ArrayList"); |
|  | Console.WriteLine("2.Work with List"); |
|  | Console.WriteLine("3.Work with Sparse Matrix"); |
|  | Console.WriteLine("4.Work with Simple Stack"); |
|  | Console.WriteLine("5.Exit"); |
|  | Console.WriteLine(); |
|  |  |
|  | } |
|  |  |
|  | static int Main(string[] args) |
|  | { |
|  | #region |
|  | int n = 0; |
|  | ArrayList arli = new ArrayList(); |
|  | List <Geometric\_figures> li = new List<Geometric\_figures>(); |
|  |  |
|  | double len; |
|  |  |
|  | Rectangle rect = new Rectangle(0, 0); |
|  | Console.WriteLine("Creating rectangle"); |
|  | Console.WriteLine("Please put in your value"); |
|  | Console.Write("Length 1 "); |
|  | len = Double.Parse(Console.ReadLine()); |
|  | rect.length1 = len; |
|  | Console.Write("Length 2 "); |
|  | len = Double.Parse(Console.ReadLine()); |
|  | rect.length2 = len; |
|  |  |
|  | Square scv = new Square(0); |
|  | Console.WriteLine("Please put in your value"); |
|  | Console.Write("Length "); |
|  | len = Double.Parse(Console.ReadLine()); |
|  | scv.length1 = len; |
|  | scv.length2 = len; |
|  |  |
|  | Circle cir = new Circle(0); |
|  | Console.WriteLine("Please put in your value"); |
|  | Console.Write("Radius "); |
|  | len = Double.Parse(Console.ReadLine()); |
|  | cir.radius = len; |
|  |  |
|  | arli.Add(rect); |
|  | li.Add(rect); |
|  | arli.Add(scv); |
|  | li.Add(scv); |
|  | arli.Add(cir); |
|  | li.Add(cir); |
|  |  |
|  | #endregion |
|  |  |
|  | while (n != 5) |
|  | { |
|  |  |
|  | Main\_menu(); |
|  | n = int.Parse(Console.ReadLine()); |
|  | switch (n) |
|  | { |
|  | case 1: |
|  | { |
|  | int yeah; |
|  |  |
|  | Console.WriteLine("How do you want to sort this collection?"); |
|  | Console.WriteLine(" 1. Ascending"); |
|  | Console.WriteLine(" 2. Descending"); |
|  | yeah = int.Parse(Console.ReadLine()); |
|  | if (yeah == 1) |
|  | for (int j=0; j< arli.Count - 1; j++) |
|  | for (int i=0; i<arli.Count-1-j;i++) |
|  | { |
|  | if (((Geometric\_figures)arli[i]).CompareTo(arli[i+1])==1) |
|  | { |
|  | Object spec = arli[i]; |
|  | arli[i] = arli[i + 1]; |
|  | arli[i + 1] = spec; |
|  | } |
|  | } |
|  | else |
|  | for (int j = 0; j < arli.Count - 1; j++) |
|  | for (int i = 0; i < arli.Count - 1 - j; i++) |
|  | { |
|  | if (((Geometric\_figures)arli[i]).CompareTo(arli[i + 1]) == 0) |
|  | { |
|  | Object spec = arli[i]; |
|  | arli[i] = arli[i + 1]; |
|  | arli[i + 1] = spec; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | Console.WriteLine(); |
|  |  |
|  | foreach (object i in arli) |
|  | { |
|  | if (i.GetType().Name == "Rectangle") |
|  | { |
|  | Console.WriteLine(i.GetType().Name + ":"); |
|  | ((Rectangle)i).Print(); |
|  | } |
|  | else |
|  | if (i.GetType().Name == "Square") |
|  | { |
|  | Console.WriteLine(i.GetType().Name + ":"); |
|  | ((Square)i).Print(); |
|  | } |
|  | else |
|  | if (i.GetType().Name == "Circle") |
|  | { |
|  | Console.WriteLine(i.GetType().Name + ":"); |
|  | ((Circle)i).Print(); |
|  | } |
|  | } |
|  |  |
|  | break; |
|  | } |
|  | case 2: |
|  | { |
|  | int yeah; |
|  |  |
|  | Console.WriteLine("How do you want to sort this collection?"); |
|  | Console.WriteLine(" 1. Ascending"); |
|  | Console.WriteLine(" 2. Descending"); |
|  | yeah = int.Parse(Console.ReadLine()); |
|  | if (yeah == 1) |
|  | for (int j = 0; j < li.Count - 1; j++) |
|  | for (int i = 0; i < li.Count - 1 - j; i++) |
|  | { |
|  | if (((Geometric\_figures)li[i]).CompareTo(li[i + 1]) == 0) |
|  | { |
|  | Object spec = li[i]; |
|  | li[i] = li[i + 1]; |
|  | li[i + 1] = (Geometric\_figures)spec; |
|  | } |
|  | } |
|  | else |
|  | for (int j = 0; j < li.Count - 1; j++) |
|  | for (int i = 0; i < li.Count - 1 - j; i++) |
|  | { |
|  | if (((Geometric\_figures)li[i]).CompareTo(li[i + 1]) == 1) |
|  | { |
|  | Object spec = li[i]; |
|  | li[i] = li[i + 1]; |
|  | li[i + 1] = (Geometric\_figures)spec; |
|  | } |
|  | } |
|  |  |
|  | foreach (object i in li) |
|  | { |
|  | if (i.GetType().Name == "Rectangle") |
|  | { |
|  | Console.WriteLine(i.GetType().Name + ":"); |
|  | ((Rectangle)i).Print(); |
|  | } |
|  | else |
|  | if (i.GetType().Name == "Square") |
|  | { |
|  | Console.WriteLine(i.GetType().Name + ":"); |
|  | ((Square)i).Print(); |
|  | } |
|  | else |
|  | if (i.GetType().Name == "Circle") |
|  | { |
|  | Console.WriteLine(i.GetType().Name + ":"); |
|  | ((Circle)i).Print(); |
|  | } |
|  | } |
|  |  |
|  | break; |
|  | } |
|  |  |
|  | case 3: |
|  | { |
|  | Console.WriteLine("\nMatrix"); |
|  | Matrix<Geometric\_figures> matrix = new Matrix<Geometric\_figures>(3, 3, 3, new FigureMatrixCheckEmpty()); |
|  | matrix[0, 0, 0] = rect; |
|  | matrix[1, 1, 1] = scv; |
|  | matrix[2, 2, 2] = cir; |
|  | Console.WriteLine(matrix.ToString()); |
|  | break; |
|  | } |
|  | case 4: |
|  | { |
|  | SimpleStack<Geometric\_figures> stack = new SimpleStack<Geometric\_figures>(); |
|  | stack.Push(rect); |
|  | stack.Push(scv); |
|  | stack.Push(cir); |
|  |  |
|  | while (stack.Count > 0) |
|  | { |
|  | Geometric\_figures f = stack.Pop(); |
|  | Console.WriteLine(f); |
|  | } |
|  | break; |
|  | } |
|  | case 5: |
|  | { |
|  | Console.WriteLine("Thank you for using this very program"); |
|  | Console.ReadKey(); |
|  | break; |
|  | } |
|  | default: |
|  | { |
|  | Console.WriteLine("ERROR"); |
|  | } |
|  | break; |
|  | } |
|  |  |
|  | } |
|  | return 0; |
|  | } |
|  |  |
|  | } |
|  |  |
|  | } |