НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО"

Кафедра програмного забезпечення комп'ютерних систем

Лабораторна робота №2

із дисципліни «Програмування»

на тему **«C# .Net. Розширені можливості реалізації ООП у мові C#. Події.»**

Виконав:

студент 2 курсу ФПМ групи КП-93

Торговських Олександр Олегович

*(ПІБ)*

Прийняв/ла:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
|  | Бали |
| Якість виконання |  |
| Термін здачі |  |
| Сумарний бал |  |

КИЇВ — 2020

**Мета роботи**

Ознайомитися з можливостями мови C# щодо обробки колекцій даних, їх серіалізації, а також з анонімними методами, методами-розширеннями.

**Постановка задачі**

1. Перетворити код, який забезпечує роботу з подіями та обробниками подій в лабораторній роботі №2, на код, що використовує (\*):
   1. анонімні методи;
   2. lambda-вирази;
   3. типи Action та Func (кожен з них).

2. Ввести до класів (розроблених у попередніх лабораторних роботах) колекції об’єктів (використати уже існуючі типи даних), продемонструвати роботу з цими даними (додавання, видалення, пошук).

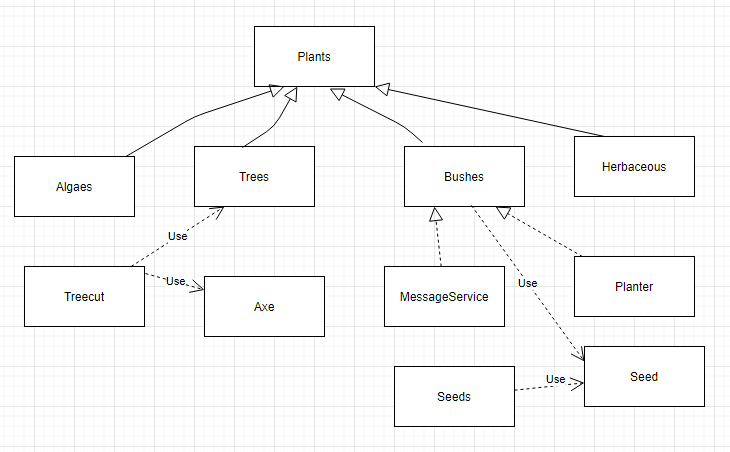
3. Створити власний тип – колекцію об’єктів. Розробити індексатори для звертання до необхідних елементів (з індексом типу не int!).

4. Реалізувати IComparable, IEnumerable, IEnumerator інтерфейси для забезпечення можливостей сортування елементів та їх обходу.

5. Створити метод-розширення класу-колекції.

6. Реалізувати у коді можливість запису поточного стану об’єктів та масивів об’єктів у файл та читання його з файлу (серіалізація). Продемонструвати роботу **двох** видів серіалізаторів.

**Діаграма класів**



Код програми

|  |
| --- |
| Program.cs |
| using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Xml.Serialization;  namespace ConsoleApp2\_Lab2  {  class Program  {  static void Main(string[] args)  {    Trees trees = new Trees(false, "green", "bigTrees" , 90);  Bigtree bt = new Bigtree { id = 2, heigth = 99, name = "OAK" };  trees.showBig();  trees.addBig(bt);  Bigtree bt\_ = trees.findBig(2);  Console.WriteLine("Look what we found in list \n "+bt\_.id + " " + bt\_.heigth + " " + bt\_.name);  trees.deleteBig(bt);  trees.showBig();  Seeds seeds\_\_ = new Seeds (6);  Seeds seeds\_\_2 = new Seeds(8);  foreach (Seed\_ w in seeds\_\_)  {  Console.WriteLine(w.Get\_plant\_name());  }    seeds\_\_.CharCount('s');//!!!  seeds\_\_.Sort();  seeds\_\_.XmlSerialize();  seeds\_\_.BinarySerialize();  seeds\_\_2.XmlSerializeGet();  seeds\_\_2.BinaryDeserialize();  foreach (Seed\_ w in seeds\_\_2)  {  Console.WriteLine(w.id);  }  Console.WriteLine("-----------");  foreach (Seed\_ w in seeds\_\_)  {  Console.WriteLine(w.Get\_plant\_name());  }  Seed\_ w1 = new Seed\_() { Name = "Blueberry", Id = 20 };  Console.WriteLine("-----------");  Bushes bush = Bushes.GetBush("USA");  Bushes.PLantABush qwe = Bushes.TryPLantABush(bush.max\_heigth\_meters);  string res = qwe(bush.type\_of\_fruit);  Console.WriteLine(res);  Console.WriteLine("-----------");  bush.type\_of\_fruit = "blueberry";  Bushes.PLantABush qwe2 = Bushes.lambdaTryPLantABush(bush.max\_heigth\_meters);  string res2 = qwe2(bush.type\_of\_fruit);  Console.WriteLine(res2);    Bushes.PLantABush qwe3 = Bushes.FuncTryPLantABush(bush.max\_heigth\_meters);  qwe3(bush.type\_of\_fruit);  //Console.WriteLine(res3);  Console.ReadKey();  }  }  } |

|  |
| --- |
| Аніноімний метод, lambda, action,func |
| //anon  public delegate string PLantABush(string type\_of\_fruit);  public static PLantABush TryPLantABush(double max\_heigth)  {  if (max\_heigth == 3.5)  {  PLantABush can\_ = delegate (string type\_of\_fruit)  {  return "Grats! U can plant "+ type\_of\_fruit;  };  return can\_;  }  else  {  PLantABush cant\_ = delegate (string type\_of\_fruit)  {  return "Sry u can`t plan " + type\_of\_fruit;  };  return cant\_;  }  }  //lambda  public static PLantABush lambdaTryPLantABush(double max\_heigth)  {  if (max\_heigth == 3.5) { PLantABush can\_ = (string type\_of\_fruit) =>{ return "Grats! U can plant " + type\_of\_fruit; }; return can\_;}  else { PLantABush cant\_ = (string type\_of\_fruit) =>{ return "Sry u can`t plan " + type\_of\_fruit;}; return cant\_;}  }  //action  public static PLantABush ActionTryPLantABush(double max\_heigth)  {  if (max\_heigth == 3.5)  {  Action<string> tryPlant = TryPLantABush\_yes;  return null;  }  else  {  Action<string> cantPlant = TryPLantABush\_no;  return null;  }  }  static void TryPLantABush\_yes(string type\_of\_fruit)  {  Console.WriteLine("Grats! U can plant " + type\_of\_fruit);  }  static void TryPLantABush\_no(string type\_of\_fruit)  {  Console.WriteLine("Sry u can`t plan " + type\_of\_fruit);  }  //func  public static PLantABush FuncTryPLantABush(double max\_heigth)  {  if (max\_heigth == 3.5)  {  Func<string, string> tryPlant = TryPLantABush\_yes2;  string message = TryPLantABush\_yes2(max\_heigth.ToString());  Console.WriteLine(message);  return null;  }  else  {  Func<string, string> cantPlant = TryPLantABush\_no2;  string message = TryPLantABush\_no2(max\_heigth.ToString());  Console.WriteLine(message);  return null;  }  }  static string TryPLantABush\_yes2(string max\_heigth)  {  return ("FUNC:\nGrats! U can plant it is only" + max\_heigth + " meters high!");  }  static string TryPLantABush\_no2(string max\_heigth)  {  return ("FUNC:\nSry u can`t plan it is only" + max\_heigth + " meters high!");  } |

|  |
| --- |
| Колекція в класі Trees |
| public List<Bigtree> bigtrees = new List<Bigtree> { new Bigtree { id = 1 , heigth = 100, name = "sequoia" } };  public void showBig()  {  Console.WriteLine("Big trees u have in list");  foreach ( Bigtree tree\_ in bigtrees)  {  Console.WriteLine(tree\_.id+" "+ tree\_.heigth + " "+tree\_.name);  }  }  public void addBig(Bigtree tree)  {  bigtrees.Add(tree);  }  public void deleteBig(Bigtree tree)  {  bigtrees.Remove(tree);  }  public Bigtree findBig(int id)  {  Bigtree tree\_ = new Bigtree { };  foreach (Bigtree tree\_\_ in bigtrees)  {  if (tree\_\_.id == id)  {  tree\_ = tree\_\_;  return tree\_;  }  }  return null;  } |

|  |
| --- |
| власний тип – колекцію об’єктів, метод-розширення класу-колекції |
| [Serializable]  public class Seeds : IEnumerable, IEnumerator, IDisposable  {  public Seed\_[] seeds;  [NonSerialized]  int pos = -1;  int count = 0;  public Seed\_ this[string index]  {  get  {  foreach (Seed\_ w in seeds)  {  if (w.Get\_plant\_name() == index) return w;  }  return null;  }  set { this[index] = value; }  }  public IEnumerator GetEnumerator() { return (IEnumerator)this; }  public bool MoveNext()  {  if (pos < seeds.Length - 1)  {  pos++;  return true;  }  else  return false;  }  public void Reset()  {  pos = -1;  }  public object Current  {  get { return seeds[pos]; }  }  public Seeds(int n)  {  seeds = new Seed\_[n];  this.count = n;  for (int i = 0; i < n; i++)  {  seeds[i] = new Seed\_("Seed #" + i.ToString());  seeds[i].Id = n - i;  Console.WriteLine("{0} seed is {1} , {2} years ", i, seeds[i].Get\_plant\_name(), seeds[i].Id);  }  }  public void XmlSerialize()  {  XmlSerializer serializer = new XmlSerializer(typeof(Seed\_[]));  using (FileStream stream = new FileStream("data.xml", FileMode.OpenOrCreate))  {  serializer.Serialize(stream, seeds);  Console.WriteLine(" XML serialized!");  }  }  public void XmlSerializeGet()  {  XmlSerializer serializer = new XmlSerializer(typeof(Seed\_[]));  // Declare an object variable of the type to be deserialized.  Seed\_[] i;  using (Stream reader = new FileStream("data.xml", FileMode.Open))  {  // Call the Deserialize method to restore the object's state.  i = (Seed\_[])serializer.Deserialize(reader);  Console.WriteLine("XML DEserialized!");  }    seeds = i;  }  public void BinarySerialize()  {  BinaryFormatter formatter = new BinaryFormatter();  using (FileStream fs = new FileStream("data.bin", FileMode.OpenOrCreate))  {  formatter.Serialize(fs, seeds);  Console.WriteLine("Binary Serialized!");  }  }  public void BinaryDeserialize()  {  BinaryFormatter formatter = new BinaryFormatter();  using (FileStream fs = new FileStream("data.bin", FileMode.OpenOrCreate))  {  seeds = (Seed\_[])formatter.Deserialize(fs);  Console.WriteLine("Binary DeSerialized!");  }  }  public void Dispose()  {  foreach (Seed\_ w in seeds)  w.Dispose();  }  ~Seeds()  {  Dispose();  }  }  public static class SeedsExtention  {  public static int CharCount(this Seeds seed, char c)  {  int counter = 0;  return counter;  }  public static Seeds Sort(this Seeds seed)  {  Array.Sort(seed.seeds);  seed.Reset();  return seed;  }  } |

|  |
| --- |
| XML & Binary серіалізація |
| public void XmlSerialize()  {  XmlSerializer serializer = new XmlSerializer(typeof(Seed\_[]));  using (FileStream stream = new FileStream("data.xml", FileMode.OpenOrCreate))  {  serializer.Serialize(stream, seeds);  Console.WriteLine(" XML serialized!");  }  }  public void XmlSerializeGet()  {  XmlSerializer serializer = new XmlSerializer(typeof(Seed\_[]));  // Declare an object variable of the type to be deserialized.  Seed\_[] i;  using (Stream reader = new FileStream("data.xml", FileMode.Open))  {  // Call the Deserialize method to restore the object's state.  i = (Seed\_[])serializer.Deserialize(reader);  Console.WriteLine("XML DEserialized!");  }    seeds = i;  }  public void BinarySerialize()  {  BinaryFormatter formatter = new BinaryFormatter();  using (FileStream fs = new FileStream("data.bin", FileMode.OpenOrCreate))  {  formatter.Serialize(fs, seeds);  Console.WriteLine("Binary Serialized!");  }  }  public void BinaryDeserialize()  {  BinaryFormatter formatter = new BinaryFormatter();  using (FileStream fs = new FileStream("data.bin", FileMode.OpenOrCreate))  {  seeds = (Seed\_[])formatter.Deserialize(fs);  Console.WriteLine("Binary DeSerialized!");  }  } |

Висновки

Виконуючи дану лабораторну роботу я ознайомився з можливостями мови C# щодо обробки колекцій даних, їх серіалізації, а також з анонімними методами, методами-розширеннями.