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**

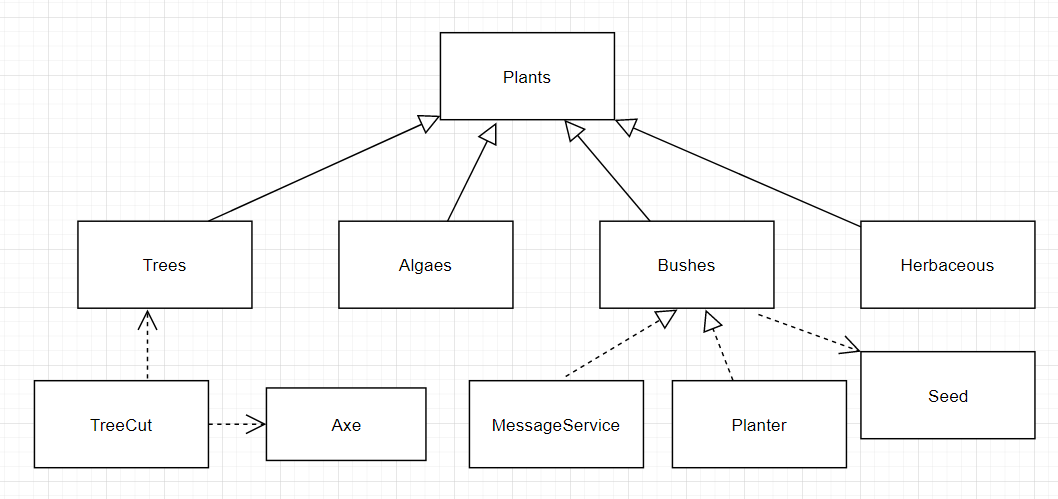
Ознайомитися з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати. Вивчити механізми оброблення подій у C#.

**Постановка задачі**

Для ієрархії класів, побудованої в лабораторній роботі №1, реалізувати:

1. Множину інтерфейсів. При чому один з класів повинен реалізовувати щонайменше 2 інтерфейси. Також продемонструвати реалізацію explicit implementation інтерфейса.
2. Абстрактний клас. Забезпечити його наслідування. Наявність в цьому класі абстрактних методів - обов’язкова.
3. Механізм «делегат – подія – обробник події»**.**
4. Механізм створення та обробки власних помилок:
   1. створити новий клас виключної ситуації;
   2. створити новий клас аргументів для передачі їх до обробника виключної ситуації;
   3. забезпечити ініціювання створеної виключної ситуації та продемонструвати, як працює обробник даної помилки.

**Діаграма класів**



Код програми

|  |
| --- |
| Program.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp2\_Lab2  {  class Program  {  static void Main(string[] args)  {  Trees oak = new Trees(false, "green", "oak", 50);  Console.WriteLine(oak.name1);  Console.WriteLine(Trees.add\_num\_of\_trees());  Console.WriteLine(Algaes.return\_num\_of\_trees());  Algaes alg = new Algaes("light-green", "Pacific ocean", "Undaria");  alg.plant\_smthing("Corals");  Herbaceous bush = new Herbaceous("yellow", false, "sunflower", 2, false, 25);  Bushes b = Bushes.GetBush("Africa"); //private ctor gotten  var seed = new Seed("StrawBerry") ;  var bush\_ = new Bushes("Blackberry", "purple"); //publisher  var planter = new Planter(); //subscriber  var messageService = new MessageService(); //subscriber  bush\_.PlantABush += planter.OnPlantABush;  bush\_.PlantABush += messageService.OnPlantABush;  bush\_.plantBush(seed);    Axe axe = new Axe("SmallAxe"); //catch an Exception  TreeCut t = new TreeCut ();  Console.WriteLine("\n");  try  {  t.CutATree(oak, axe);  } catch (TreeCutException ex)  {  Console.WriteLine($"Error: {ex.Message}");  Console.WriteLine($"Need a bigger axe, you have: {ex.Axe}");  }  }  }  } |

|  |
| --- |
| Plants.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp2\_Lab2  {  public abstract class Plants  {  public int age; //int  public bool underwater; //boolean  private static int number\_of\_Trees\_species = 0; //int  private static int number\_of\_Bushes\_species = 0; //int  protected int number\_of\_leafes;  public static string addgrass;  public Plants()  {  age = 1;  underwater = false;  number\_of\_leafes = 0;  Console.WriteLine("default Plants ctor was called");  }  public Plants(int age, bool underwater, int number\_of\_leafes)  {  this.age = age;  this.underwater = underwater;  this.number\_of\_leafes = number\_of\_leafes;  Console.WriteLine("Plants ctor with prmetrs was called");  }  static Plants()  {  Console.WriteLine("static ctor for plant was called");  addgrass = CreateGrass();  }  public static string add\_num\_of\_trees()  {  ++number\_of\_Trees\_species;  return "added a tree species";  }  public static int return\_num\_of\_trees()  {  return number\_of\_Trees\_species;  }  private static string CreateGrass()  {  return "grass";  }  virtual public void plant\_smthing(string smthing)  {  Console.WriteLine("Planted " + smthing + " !!!");  }  public virtual string Create\_a\_plant { get; set; }  public abstract void SeeWhatPlant();  }  } |

|  |
| --- |
| Trees.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp2\_Lab2  {  interface BigTree  {  string printBigTree();  }  interface SmallTree  {  string printSmallTree();  }  class Trees : Plants, BigTree, SmallTree  {  const int max\_heigth\_meters = 100;  public string color;  public bool coniferous;  public readonly string name1;  public static string namee;  public int age;  static Trees()  {  Console.WriteLine("static ctor for trees was called");  namee = CreateOak();  }  public Trees(bool coniferous, string color, string name1, int age)  {  Console.WriteLine("default trees constructor called");  this.underwater = false;  this.coniferous = coniferous;  this.color = color;  this.name1 = name1;  this.age = age;  }  private static string CreateOak()  {  return "oak";  }  private static string CreateSequoia()  {  return "sequoia";  }  string BigTree.printBigTree()  {  Console.WriteLine("Big tree given(sequoia)");  return CreateSequoia();  }  string SmallTree.printSmallTree()  {  Console.WriteLine("Small tree given(oak)");  return CreateOak();  }  public override void SeeWhatPlant()  {  Console.WriteLine("This is a tree its called "+ name1);  }  }  } |

|  |
| --- |
| Bushes.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp2\_Lab2  {  public class BushEventArgs : EventArgs  {  public Seed Seed { get; set; }  }  class Bushes : Plants  {  const double max\_heigth\_meters = 3.5;  public string type\_of\_fruit { get; set; }  public string color { get; private set; }  public Bushes(string type\_of\_fruit, string color)  {  this.type\_of\_fruit = type\_of\_fruit;  this.color = color;  }  private Bushes()  {  Console.WriteLine("private ctor 4 bushes was called");  type\_of\_fruit = "raspberry";  color = "purple";  this.underwater = false;  }  public static Bushes GetBush(string LocationOfBush)  {  if (LocationOfBush != "Africa")  {  Console.WriteLine("Bush was planted in " + LocationOfBush);  return new Bushes();  }  else  {  Console.WriteLine("U can`t plant a bush in " + LocationOfBush);  return null;  }  }  public override void SeeWhatPlant()  {  Console.WriteLine("This is a bush its color is " + color);  }  //public delegate void PlantABushEventHandler(object source, BushEventArgs args);  public event EventHandler<BushEventArgs> PlantABush; // same as delegate  public void plantBush(Seed seed)  {  Console.WriteLine("\n\n Planting a bush " + seed.nameOfBush);  this.type\_of\_fruit = seed.nameOfBush;  OnPlantABush(seed);  }  protected virtual void OnPlantABush(Seed seed )  {  if (PlantABush != null)  PlantABush(this, new BushEventArgs() { Seed = seed});    }  }  } |

|  |
| --- |
| Herbaceous.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp2\_Lab2  {  class Herbaceous : Plants  {  private string plant\_to\_create;  public override string Create\_a\_plant  {  get  {  return plant\_to\_create;  }  set  {  if (!string.IsNullOrEmpty(value))  {  plant\_to\_create = value;  Console.WriteLine(plant\_to\_create + " bush was created");  }  else  {  plant\_to\_create = "Please make sure u input everything correct";  }  }  }  const double max\_heigth\_meters = 1.5;  public string color;  public bool predatory;  public Herbaceous(string color, bool predatory, string plant\_to\_create, int age, bool underwater, int number\_of\_leafes) : base(age, underwater, number\_of\_leafes)  {  this.color = color;  this.predatory = predatory;  this.Create\_a\_plant = plant\_to\_create;  }  public void show\_created\_plant()  {  Console.WriteLine(" CREATED!! " + plant\_to\_create);  }  public override void SeeWhatPlant()  {  Console.WriteLine("This is a Herbaceous its color is " + color);  }  }  } |

|  |
| --- |
| Algaes.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp2\_Lab2  {  public class Algaes : Plants  {  private string plant\_to\_create;  public override string Create\_a\_plant  {  get  {  return plant\_to\_create;  }  set  {  if (!string.IsNullOrEmpty(value))  {  plant\_to\_create = value;  Console.WriteLine(plant\_to\_create + " algaes were created");  }  else  {  plant\_to\_create = "Please make sure u input everything correct";  }  }  }  const double max\_heigth\_meters = 2.5;  public string color;  public string oceanORsea\_where\_lives;  public Algaes(string color, string oceanORsea\_where\_lives, string plant\_to\_create) : base()  {  this.color = color;  this.oceanORsea\_where\_lives = oceanORsea\_where\_lives;  this.Create\_a\_plant = plant\_to\_create;  }  public void show\_created\_plant()  {  Console.WriteLine(" CREATED!! " + plant\_to\_create);  }  public override void plant\_smthing(string smthing)  {  Console.WriteLine("starting planting");  base.plant\_smthing(smthing);  }  public override void SeeWhatPlant()  {  Console.WriteLine("This is a algae its location is " + oceanORsea\_where\_lives);  }  }  } |

|  |
| --- |
| TreeCutException.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp2\_Lab2  {  class TreeCutException : Exception  {  public string Axe { get; }  public int InputAge { get; }  public TreeCutException(string message, string axe, int input\_age) : base(message)  {  Axe = axe;  InputAge = input\_age;  }  }  class Axe  {  public string axe\_type;  public Axe (string axe\_type)  {  this.axe\_type = axe\_type;  }  }  class TreeCut  {  public TreeCut() { }  public void CutATree(Trees tree, Axe axe)  {  if (tree.age >= 20 && axe.axe\_type == "SmallAxe") throw new TreeCutException("This axe doesn`t fit to cut this tree", axe.axe\_type, tree.age);  else Console.WriteLine("Tree was cut");  }  }  } |

|  |
| --- |
| Planter.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp2\_Lab2  {  public class Planter  {  public void OnPlantABush(object source, BushEventArgs e)  {  bool x = false;  int num;  while (x == false)  {  Console.WriteLine("Enter how many seeds should gardener plant");  try  {  num = Convert.ToInt32(Console.ReadLine());  x = true;  Console.WriteLine("Planter: gardener just planted a bush - " + e.Seed.nameOfBush +" "+num+ " times ");    }  catch (FormatException)  {  Console.WriteLine("Format Exeption, please input correct data (number)");  x = false;  }  }    }  }  }  public class Seed  {  public string nameOfBush;  public Seed (string nameOfBush)  {  this.nameOfBush = nameOfBush;  }  } |

|  |
| --- |
| MessageService.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp2\_Lab2  {  public class MessageService  {  public void OnPlantABush(object source, BushEventArgs args)  {  Console.WriteLine("MessageService: Sending an info about planted bush " + args.Seed.nameOfBush);  }  }  } |

Висновки

Виконуючи дану лабораторну роботу я ознайомився з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати. Вивчив механізми оброблення подій у C#