НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО"

Кафедра програмного забезпечення комп'ютерних систем

Лабораторна робота №1

із дисципліни «Програмування»

на тему **«C# .Net. Реалізація основних принципів ООП мовою C#»**

Виконав:

студент 2 курсу ФПМ групи КП-91

Торговських Олександр Олегович

*(ПІБ)*

Прийняв/ла:

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

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

КИЇВ — 2020

**Мета роботи**

Ознайомитися з основами об’єктно-орієнтованого підходу до створення ПЗ у мові С#, створенням класів, об’єктів, механізмами інкапсуляції, наслідування та поліморфізму.

**Постановка задачі**

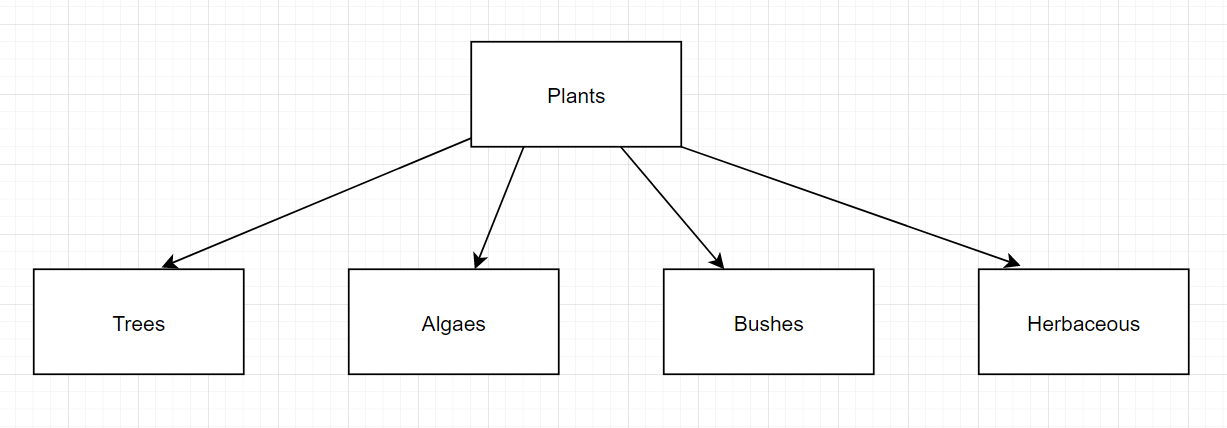
Побудувати ієрархію класів, що відтворюватимуть відношення наслідування між об’єктами реального світу. (Кількість класів >= 5). При цьому забезпечити реалізацію таких механізмів ООП:

1. Наявність у класах статичних даних. Забезпечити коректну ініціалізацію цих даних.
2. Наявність у класах змінних та методів з різними модифікаторами доступу.
3. Наявність у класах полів const та readonly.
4. Наявність у класах властивостей (складніше, ніж просто get;set;).
5. Конструктори. Створити для розроблюваних класів такі конструктори:
   * + перевизначений конструктор за замовчуванням;
     + перевизначений конструктор;
     + приватний конструктор;
     + статичний конструктор.

Продемонструвати, яким чином викликаються конструктори базового та дочірнього класів.

1. Використати віртуальні та перевизначені методи.
2. Продемонструвати поліморфізм методів.

**Діаграма класів**



Код програми

|  |
| --- |
| Program.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp1\_Lab1  {  class Program  {  static void Main(string[] args)  {  Trees oak = new Trees(false, "green", "oak");  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(); //private ctor gotten  }  }  } |

|  |
| --- |
| Plants.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp1\_Lab1  {  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; }  }  } |

|  |
| --- |
| Trees.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp1\_Lab1  {  class Trees : Plants  {  const int max\_heigth\_meters = 100;  public string color;  public bool coniferous;  public readonly string name1;  public static string namee;  static Trees()  {  Console.WriteLine("static ctor for trees was called");  namee = CreateOak();  }  public Trees(bool coniferous, string color, string name1)  {  Console.WriteLine("default trees constructor called");  this.underwater = false;  this.coniferous = coniferous;  this.color = color;  this.name1 = name1;  }  private static string CreateOak()  {  return "oak";  }  }    } |

|  |
| --- |
| Bushes.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp1\_Lab1  {  class Bushes : Plants  {  const double max\_heigth\_meters = 3.5;  public string type\_of\_fruit { get; set; }  public string color { get; private set; }    private Bushes()  {  Console.WriteLine("private ctor 4 bushes was called");  type\_of\_fruit = "raspberry";  color = "purple";  this.underwater = false;  }  public static Bushes getBush() => new Bushes(); //private ctor    }  } |

|  |
| --- |
| Herbaceous.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp1\_Lab1  {  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);  }  }  } |

|  |
| --- |
| Algaes.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ConsoleApp1\_Lab1  {  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);  }  }  } |

Висновки

У цій лабораторній роботі я ознайомився з основами об’єктно-орієнтованого підходу до створення ПЗ у мові С#, створенням класів, об’єктів, механізмами інкапсуляції, наслідування та поліморфізму.

Для процесу **інкапсуляції** я використав області видимості public, protected, private. **Поліморфізм** був використаний з абстрактним класом та звичайним наслідуванням віртуальних функцій класу наслідувача.