МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Національний технічний університет України

Факультет інформатики та обчислювальної техніки

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра інформаційних систем та технологій

Звіт

з лабораторної  роботи № 1

з дисципліни «Програмування – 2.

Основи програмування»

Варіант № 5

Виконав: Валіваха Андрій

     Студент гр. ІС-11 , ФІОТ

1 курс

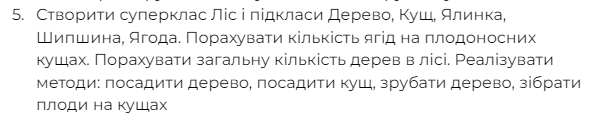
Київ 2022

**Лабораторна робота #4**

Інтерфейси. Абстрактні класи

**Завдання**

* Проаналізувати завдання, виділити інформаційні об’єкти та дії. Визначити правильну ієрархію об’єктів
* Створити базовий суперклас (абстрактний клас або інтерфейс) і визначити загальні методи для даного класу. Створити підкласи, в які додати специфічні властивості та методи. Частину методів перевизначити.
* Розробити програму з використанням абстрактних класів та інтерфейсів. Чітко розуміти, де доцільно використати суперклас, а де звичайний.
* При розробці використовувати наслідування та поліморфізм
* У всіх класах повинні бути реалізовані доцільні для класу методи, навіть якщо це не вказано у завданні
* Використовувати об’єкти підкласів для моделювання реальних ситуацій на об’єктів

using System;

using System.Collections.Generic;

namespace OPLab\_4

{

public class HelloWorld

{

public static void Main(string[] args)

{

Forest forest = new Forest();

IBerryCollectible berryForest = forest;

// trees

Spruce spruce = new Spruce();

Oak oak = new Oak();

// bushes

RedRose redRose1 = new RedRose();

RedRose redRose2 = new RedRose();

forest.Plant(spruce);

forest.Plant(oak);

forest.Plant(redRose1);

forest.Plant(redRose2);

var berriesCount = berryForest.BerriesCount();

Console.WriteLine($"Total berries in the forest: {berriesCount}");

var berries = berryForest.CollectBerries();

forest.CutTree();

}

public abstract class Plant

{

public abstract void Grow();

}

public class Berry

{

}

public interface IBerryCollectible

{

public List<Berry> Berries { get; }

public int BerriesCount()

{

return Berries.Count;

}

public List<Berry> CollectBerries();

}

public class Tree : Plant

{

public override void Grow()

{

Console.WriteLine("Tree is growing...");

}

public void CutDown()

{

Console.WriteLine("Tree is cut down(");

}

}

class Bush : Plant

{

public override void Grow()

{

Console.WriteLine("Bush is growing...");

}

}

class Spruce : Tree

{

}

class Oak : Tree

{

public override void Grow()

{

Console.WriteLine("Tree is growing... And this is Oak.");

}

}

class RedRose : Bush, IBerryCollectible

{

private List<Berry> \_berries = new List<Berry>() { new Berry(), new Berry(), new Berry() };

public List<Berry> Berries => \_berries;

public List<Berry> CollectBerries()

{

Console.WriteLine("Collecting red-rose berries...");

var collectedBerries = \_berries;

\_berries.Clear();

return collectedBerries;

}

}

public class Forest : IBerryCollectible

{

List<Plant> plants = new List<Plant>();

public List<Berry> Berries

{

get

{

List<Berry> listOfBerries = new List<Berry>();

foreach (Plant pln in plants)

{

if (pln is IBerryCollectible)

{

var bushBerries = ((IBerryCollectible)pln).Berries;

listOfBerries.AddRange(bushBerries);

}

}

return listOfBerries;

}

}

public void Plant(Plant p)

{

plants.Add(p);

p.Grow();

}

public List<Berry> CollectBerries()

{

List<Berry> listOfBerries = new List<Berry>();

foreach (Plant pln in plants)

{

if (pln is IBerryCollectible)

{

var bushBerries = ((IBerryCollectible)pln).CollectBerries();

listOfBerries.AddRange(bushBerries);

}

}

return listOfBerries;

}

public void CutTree()

{

for (int i = 0; i < plants.Count; i++)

{

var plant = plants[i];

if (plant is Tree)

{

((Tree)plant).CutDown();

plants.RemoveAt(i);

return;

}

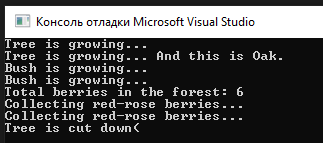
}

}

}

}

}



Висновок: виконавши дану лр я навчився використовувати Абстрактні класи й інтерфейси і визначив яка в них різниця.