МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

Практична робота (Lab-5)

з дисципліни «Об’єктно-орієнтоване програмування»

*назва дисципліни*

на тему: «Модульне тестування (Unit testing)»

Виконав: студент 2 курсу групи 622п напряму підготовки (спеціальності)

121 інженерія програмного забезпечення

(шифр і назва напряму підготовки (спеціальності))

Пьоришкін Т.Є.

(прізвище й ініціали студента)

Прийняв: доц. каф. 603, к.т.н., доцент

Шевченко І.В.

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала: Кількість балів: Оцінка ECTS:

Харків – 2025

**Завдання**

Мета роботи – навчитися застосовувати фреймворк MSTest для створення unit-тестів.

На основі отриманого на лекції 5 теоретичного матеріалу виконати модульне тестування (unit testing) розробленого класу, який був вдосконалений у практичній роботі Lab-4:

1. У Solution, в якому реалізовано основний проєкт (реалізація практичної роботи Lab-4), додати проєкт для побудови модульних тестів MSTest.

2. Для тест-класу MSTest-проєкта змінити назву за замовчуванням на відповідну назву (назва тест-класу має відповідати назві класу, який тестується).

3. Розробити тест-методи (test methods)\* для:

- повного тестування логіки роботи всіх нестатичних і статичних методів класу;

- тестування роботи секції set декількох властивостей (секція set повинна мати реалізовані перевірки).

\*Тест-метод (test method) повинен обов’язково мати назву, яка має відображати суть тесту, а також структуру, яка має відповідати шаблону AAA (Arrange, Act, Assert).

У тест-методах (test methods) має бути продемонстровано використання різних методів класу Assert: AreEqual, IsTrue/IsFalse, ThrowsException та інших\*\*, а також має бути продемонстровано використання різних атрибутів: [DataRow(…)], [ExpectedException(…)], [TestInitialize] і [TestCleanup].

4. Запустити виконання розроблених тестів. Якщо деякі тести будуть непройдені (fail), то необхідно підкорегувати програмний код, який тестується, і досягти повного проходження всіх розроблених тестів.

5. Оформити звіт.

**Class diagram**

### Основной проект (Plant, Program, PlantType)

classDiagram

class Plant {

-string \_name

-int \_age

-double \_height

-PlantType \_type

-DateTime \_plantingDate

-bool \_isFlowering

-string \_lastWatered

-static int \_count

-static double \_totalHeight

+string Name

+PlantType Type

+int Age

+double Height

+DateTime PlantingDate

+bool IsFlowering

+string AgeCategory

+string LastWatered

+static int Count

+static double AverageHeight

+Plant(string, PlantType, int, double, DateTime)

+Plant()

+Plant(string, PlantType)

+Plant(string, int)

+static void AddPlantToCollection(Plant)

+static void RemovePlantFromCollection(Plant)

+static string GetGlobalPlantInfo()

+static Plant Parse(string)

+static bool TryParse(string, out Plant)

+string ToString()

+string GetDescription()

+string GetPlantingInfo()

+bool IsMature()

+void Grow(double)

+void Grow()

+void Grow(int)

+void WaterPlant()

+void WaterPlant(string)

+void WaterPlant(int)

+void WaterPlant(string, int)

-void ValidatePlantingDate(DateTime)

-string FormatWateringTime(DateTime)

-string GetFormattedDescription()

}

class PlantType {

<<enumeration>>

Tree

Shrub

Flower

Grass

Fern

Succulent

Vine

}

class Program {

-static List~Plant~ plants

-static int maxPlants

-static Random random

+static void Main(string[])

+static void InitializeMaxPlants()

+static void ShowMenu()

+static void AddPlant()

+static Plant CreatePlantFromString()

+static Plant CreatePlantManually()

+static Plant CreatePlantRandom()

+static void ViewAllPlants()

+static void FindPlant()

+static void DisplaySearchResults(List~Plant~)

+static void DemonstrateBehavior()

+static void DeletePlant()

+static void DemonstrateStaticMethods()

+static int GetValidatedInput(string, int, int)

+static double GetValidatedDoubleInput(string, double, double)

+static DateTime GetValidatedDate()

}

Plant --> PlantType

### Тест-проект (PlantTests)

classDiagram

class PlantTests {

-Plant testPlant

-static Type plantType

-System.Reflection.FieldInfo countField

-System.Reflection.FieldInfo totalHeightField

+void TestInitialize()

+void TestCleanup()

+void PlantConstructor\_ValidParameters\_ObjectInitializedCorrectly()

+void NameSet\_EmptyString\_ThrowsArgumentException()

+void NameSet\_TooShort\_ThrowsArgumentException()

+void HeightSet\_NegativeValue\_ThrowsArgumentException()

+void TypeSet\_InvalidEnumValue\_ThrowsArgumentException()

+void GetDescription\_ValidPlant\_ReturnsCorrectDescription()

+void IsMature\_YoungPlant\_ReturnsFalse()

+void IsMature\_AdultPlant\_ReturnsTrue()

+void Grow\_WithPositiveGrowth\_IncreasesHeight()

+void Grow\_WithNegativeGrowth\_ThrowsArgumentException()

+void Grow\_WithoutParameters\_IncreasesHeightByDefault()

+void AgeCategory\_YoungPlant\_ReturnsYoung()

+void AgeCategory\_AdultPlant\_ReturnsAdult()

+void Parse\_ValidString\_ReturnsPlantObject()

+void Parse\_InvalidString\_ThrowsFormatException()

+void TryParse\_ValidString\_ReturnsTrueAndPlant()

+void TryParse\_InvalidString\_ReturnsFalseAndNull()

+void Count\_AfterAddingPlants\_ReturnsCorrectNumber()

+void AverageHeight\_WithMultiplePlants\_ReturnsCorrectAverage()

+void WaterPlant\_AfterWatering\_UpdatesLastWatered()

+void WaterPlant\_WithWaterType\_ExecutesSuccessfully()

+void Constructor\_NameAndType\_CreatesPlantWithDefaultValues()

+void Constructor\_NameAndAge\_CreatesPlantWithDefaultValues()

+void ToString\_ValidPlant\_ReturnsCorrectFormat()

+void PlantingDateSet\_FutureDate\_ThrowsArgumentException()

+void PlantingDateSet\_VeryOldDate\_ThrowsArgumentException()

+void AgeCategory\_DifferentAges\_ReturnsCorrectCategory(int, string)

+void GetGlobalPlantInfo\_AfterAddingPlants\_ReturnsCorrectInfo()

+void GetPlantingInfo\_ValidPlant\_ReturnsCorrectString()

+void IsFlowering\_SetAndGet\_ReturnsCorrectValue()

+void AddPlantToCollection\_ValidPlant\_IncreasesCountAndTotalHeight()

+void AgeSet\_InvalidValue\_ThrowsArgumentException()

+void WaterPlant\_WithMilliliters\_ExecutesSuccessfully()

+void WaterPlant\_WithWaterTypeAndMilliliters\_ExecutesSuccessfully()

}

PlantTests --> Plant

## Реалізація тест-класу

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

[TestClass]

public class PlantTests

{

private Plant testPlant;

private static Type plantType;

private System.Reflection.FieldInfo countField;

private System.Reflection.FieldInfo totalHeightField;

[TestInitialize]

public void TestInitialize()

{

testPlant = new Plant("Тестова рослина", PlantType.Flower, 2, 0.5, new DateTime(2023, 5, 15));

plantType = typeof(Plant);

countField = plantType.GetField("\_count", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Static);

totalHeightField = plantType.GetField("\_totalHeight", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Static);

countField.SetValue(null, 0);

totalHeightField.SetValue(null, 0.0);

}

[TestCleanup]

public void TestCleanup()

{

testPlant = null;

}

[TestMethod]

public void PlantConstructor\_ValidParameters\_ObjectInitializedCorrectly()

{

// Arrange

string name = "Троянда";

PlantType type = PlantType.Flower;

int age = 2;

double height = 0.5;

DateTime plantingDate = new DateTime(2023, 5, 15);

// Act

Plant plant = new Plant(name, type, age, height, plantingDate);

// Assert

Assert.AreEqual(name, plant.Name);

Assert.AreEqual(type, plant.Type);

Assert.AreEqual(age, plant.Age);

Assert.AreEqual(height, plant.Height);

Assert.AreEqual(plantingDate, plant.PlantingDate);

}

[TestMethod]

public void NameSet\_EmptyString\_ThrowsArgumentException()

{

// Arrange

string invalidName = "";

// Act & Assert

try

{

testPlant.Name = invalidName;

Assert.Fail("Очікувалося виняток ArgumentException");

}

catch (ArgumentException)

{

Assert.IsTrue(true);

}

}

[TestMethod]

public void NameSet\_TooShort\_ThrowsArgumentException()

{

// Arrange

string invalidName = "А";

// Act & Assert

try

{

testPlant.Name = invalidName;

Assert.Fail("Очікувалося виняток ArgumentException");

}

catch (ArgumentException)

{

Assert.IsTrue(true);

}

}

[TestMethod]

public void HeightSet\_NegativeValue\_ThrowsArgumentException()

{

// Arrange

double invalidHeight = -1.0;

// Act & Assert

try

{

testPlant.Height = invalidHeight;

Assert.Fail("Очікувалося виняток ArgumentException");

}

catch (ArgumentException)

{

Assert.IsTrue(true);

}

}

[TestMethod]

public void TypeSet\_InvalidEnumValue\_ThrowsArgumentException()

{

// Arrange

PlantType invalidType = (PlantType)100;

// Act & Assert

try

{

testPlant.Type = invalidType;

Assert.Fail("Очікувалося виняток ArgumentException");

}

catch (ArgumentException)

{

Assert.IsTrue(true);

}

}

[TestMethod]

public void GetDescription\_ValidPlant\_ReturnsCorrectDescription()

{

// Arrange

string expectedDescription = "Тестова рослина (Flower) - 2 років, 0,5 м";

// Act

string actualDescription = testPlant.GetDescription();

// Assert

Assert.AreEqual(expectedDescription, actualDescription);

}

[TestMethod]

public void IsMature\_YoungPlant\_ReturnsFalse()

{

// Arrange

var youngPlant = new Plant("Молода рослина", PlantType.Tree, 1, 1.0, DateTime.Now);

// Act

bool result = youngPlant.IsMature();

// Assert

Assert.IsFalse(result);

}

[TestMethod]

public void IsMature\_AdultPlant\_ReturnsTrue()

{

// Arrange

var adultPlant = new Plant("Доросла рослина", PlantType.Tree, 10, 5.0, DateTime.Now.AddYears(-10));

// Act

bool result = adultPlant.IsMature();

// Assert

Assert.IsTrue(result);

}

[TestMethod]

public void Grow\_WithPositiveGrowth\_IncreasesHeight()

{

// Arrange

double initialHeight = testPlant.Height;

double growth = 0.3;

// Act

testPlant.Grow(growth);

// Assert

Assert.AreEqual(initialHeight + growth, testPlant.Height);

}

[TestMethod]

public void Grow\_WithNegativeGrowth\_ThrowsArgumentException()

{

// Arrange

double negativeGrowth = -0.1;

// Act & Assert

try

{

testPlant.Grow(negativeGrowth);

Assert.Fail("Очікувалося виняток ArgumentException");

}

catch (ArgumentException)

{

Assert.IsTrue(true);

}

}

[TestMethod]

public void Grow\_WithoutParameters\_IncreasesHeightByDefault()

{

// Arrange

double initialHeight = testPlant.Height;

// Act

testPlant.Grow();

// Assert

Assert.IsTrue(testPlant.Height > initialHeight);

}

[TestMethod]

public void AgeCategory\_YoungPlant\_ReturnsYoung()

{

// Arrange

var youngPlant = new Plant("Молода", PlantType.Flower, 1, 0.1, DateTime.Now);

// Act

string category = youngPlant.AgeCategory;

// Assert

Assert.AreEqual("Молода", category);

}

[TestMethod]

public void AgeCategory\_AdultPlant\_ReturnsAdult()

{

// Arrange

var adultPlant = new Plant("Доросла", PlantType.Tree, 5, 3.0, DateTime.Now.AddYears(-5));

// Act

string category = adultPlant.AgeCategory;

// Assert

Assert.AreEqual("Доросла", category);

}

[TestMethod]

public void Parse\_ValidString\_ReturnsPlantObject()

{

// Arrange

string validString = "Троянда,Flower,2,0.5,2023-05-15,True";

// Act

Plant plant = Plant.Parse(validString);

// Assert

Assert.IsNotNull(plant);

Assert.AreEqual("Троянда", plant.Name);

Assert.AreEqual(PlantType.Flower, plant.Type);

Assert.AreEqual(2, plant.Age);

}

[TestMethod]

public void Parse\_InvalidString\_ThrowsFormatException()

{

// Arrange

string invalidString = "Неправильний,рядок";

// Act & Assert

try

{

Plant.Parse(invalidString);

Assert.Fail("Очікувалося виняток FormatException");

}

catch (FormatException)

{

Assert.IsTrue(true);

}

}

[TestMethod]

public void TryParse\_ValidString\_ReturnsTrueAndPlant()

{

// Arrange

string validString = "Кактус,Succulent,5,0.3,2020-10-10,False";

// Act

bool result = Plant.TryParse(validString, out Plant plant);

// Assert

Assert.IsTrue(result);

Assert.IsNotNull(plant);

Assert.AreEqual("Кактус", plant.Name);

}

[TestMethod]

public void TryParse\_InvalidString\_ReturnsFalseAndNull()

{

// Arrange

string invalidString = "Неправильний,рядок";

// Act

bool result = Plant.TryParse(invalidString, out Plant plant);

// Assert

Assert.IsFalse(result);

Assert.IsNull(plant);

}

[TestMethod]

public void Count\_AfterAddingPlants\_ReturnsCorrectNumber()

{

// Arrange

var plant1 = new Plant("Рослина1", PlantType.Tree, 1, 1.0, DateTime.Now);

var plant2 = new Plant("Рослина2", PlantType.Flower, 2, 0.5, DateTime.Now);

// Act

Plant.AddPlantToCollection(plant1);

Plant.AddPlantToCollection(plant2);

// Assert

Assert.AreEqual(2, Plant.Count);

}

[TestMethod]

public void AverageHeight\_WithMultiplePlants\_ReturnsCorrectAverage()

{

// Arrange

var plant1 = new Plant("Висока", PlantType.Tree, 1, 10.0, DateTime.Now);

var plant2 = new Plant("Низька", PlantType.Flower, 2, 2.0, DateTime.Now);

// Act

Plant.AddPlantToCollection(plant1);

Plant.AddPlantToCollection(plant2);

// Assert

Assert.AreEqual(6.0, Plant.AverageHeight);

}

[TestMethod]

public void WaterPlant\_AfterWatering\_UpdatesLastWatered()

{

// Arrange

string initialLastWatered = testPlant.LastWatered;

// Act

testPlant.WaterPlant();

// Assert

Assert.AreNotEqual(initialLastWatered, testPlant.LastWatered);

Assert.IsTrue(testPlant.LastWatered != "Ніколи");

}

[TestMethod]

public void WaterPlant\_WithWaterType\_ExecutesSuccessfully()

{

// Arrange

string waterType = "дощова";

// Act & Assert

testPlant.WaterPlant(waterType);

Assert.IsTrue(true);

}

[TestMethod]

public void Constructor\_NameAndType\_CreatesPlantWithDefaultValues()

{

// Arrange

string name = "Нова рослина";

PlantType type = PlantType.Shrub;

// Act

Plant plant = new Plant(name, type);

// Assert

Assert.AreEqual(name, plant.Name);

Assert.AreEqual(type, plant.Type);

Assert.AreEqual(1, plant.Age);

}

[TestMethod]

public void Constructor\_NameAndAge\_CreatesPlantWithDefaultValues()

{

// Arrange

string name = "Стара рослина";

int age = 10;

// Act

Plant plant = new Plant(name, age);

// Assert

Assert.AreEqual(name, plant.Name);

Assert.AreEqual(age, plant.Age);

Assert.AreEqual(PlantType.Tree, plant.Type);

}

[TestMethod]

public void ToString\_ValidPlant\_ReturnsCorrectFormat()

{

// Arrange

var plant = new Plant("Тест", PlantType.Grass, 3, 0.7, new DateTime(2022, 6, 1));

plant.IsFlowering = false;

// Act

string result = plant.ToString();

// Assert

StringAssert.Contains(result, "Тест");

StringAssert.Contains(result, "Grass");

StringAssert.Contains(result, "3");

StringAssert.Contains(result, "0.7");

}

[TestMethod]

public void PlantingDateSet\_FutureDate\_ThrowsArgumentException()

{

// Arrange

DateTime futureDate = DateTime.Now.AddDays(1);

// Act & Assert

try

{

testPlant.PlantingDate = futureDate;

Assert.Fail("Очікувалося виняток ArgumentException");

}

catch (ArgumentException)

{

Assert.IsTrue(true);

}

}

[TestMethod]

public void PlantingDateSet\_VeryOldDate\_ThrowsArgumentException()

{

// Arrange

DateTime oldDate = new DateTime(1899, 1, 1);

// Act & Assert

try

{

testPlant.PlantingDate = oldDate;

Assert.Fail("Очікувалося виняток ArgumentException");

}

catch (ArgumentException)

{

Assert.IsTrue(true);

}

}

[DataTestMethod]

[DataRow(1, "Молода")]

[DataRow(5, "Доросла")]

[DataRow(15, "Стара")]

public void AgeCategory\_DifferentAges\_ReturnsCorrectCategory(int age, string expectedCategory)

{

// Arrange

var plant = new Plant("Тест", PlantType.Tree, age, 1.0, DateTime.Now.AddYears(-age));

// Act

string actualCategory = plant.AgeCategory;

// Assert

Assert.AreEqual(expectedCategory, actualCategory);

}

[TestMethod]

public void GetGlobalPlantInfo\_AfterAddingPlants\_ReturnsCorrectInfo()

{

// Arrange

var plant1 = new Plant("Рослина1", PlantType.Tree, 1, 5.0, DateTime.Now);

var plant2 = new Plant("Рослина2", PlantType.Flower, 2, 3.0, DateTime.Now);

Plant.AddPlantToCollection(plant1);

Plant.AddPlantToCollection(plant2);

// Act

string info = Plant.GetGlobalPlantInfo();

// Assert

StringAssert.Contains(info, "2");

StringAssert.Contains(info, "4,00");

}

[TestMethod]

public void GetPlantingInfo\_ValidPlant\_ReturnsCorrectString()

{

// Arrange

DateTime plantingDate = new DateTime(2023, 5, 15);

var plant = new Plant("Тест", PlantType.Flower, 1, 0.5, plantingDate);

// Act

string info = plant.GetPlantingInfo();

// Assert

StringAssert.Contains(info, "Тест");

StringAssert.Contains(info, "15.05.2023");

}

[TestMethod]

public void IsFlowering\_SetAndGet\_ReturnsCorrectValue()

{

// Arrange

bool expectedValue = true;

// Act

testPlant.IsFlowering = expectedValue;

// Assert

Assert.AreEqual(expectedValue, testPlant.IsFlowering);

}

[TestMethod]

public void AddPlantToCollection\_ValidPlant\_IncreasesCountAndTotalHeight()

{

// Arrange

var plant = new Plant("Нова", PlantType.Tree, 1, 2.0, DateTime.Now);

int initialCount = Plant.Count;

double initialTotalHeight = (double)totalHeightField.GetValue(null);

// Act

Plant.AddPlantToCollection(plant);

// Assert

Assert.AreEqual(initialCount + 1, Plant.Count);

Assert.AreEqual(initialTotalHeight + plant.Height, (double)totalHeightField.GetValue(null));

}

[TestMethod]

public void AgeSet\_InvalidValue\_ThrowsArgumentException()

{

// Arrange

int invalidAge = -1;

// Act & Assert

try

{

testPlant.Age = invalidAge;

Assert.Fail("Очікувалося виняток ArgumentException");

}

catch (ArgumentException)

{

Assert.IsTrue(true);

}

}

[TestMethod]

public void WaterPlant\_WithMilliliters\_ExecutesSuccessfully()

{

// Arrange

int milliliters = 500;

// Act & Assert

testPlant.WaterPlant(milliliters);

Assert.IsTrue(true);

}

[TestMethod]

public void WaterPlant\_WithWaterTypeAndMilliliters\_ExecutesSuccessfully()

{

// Arrange

string waterType = "мінеральна";

int milliliters = 300;

// Act & Assert

testPlant.WaterPlant(waterType, milliliters);

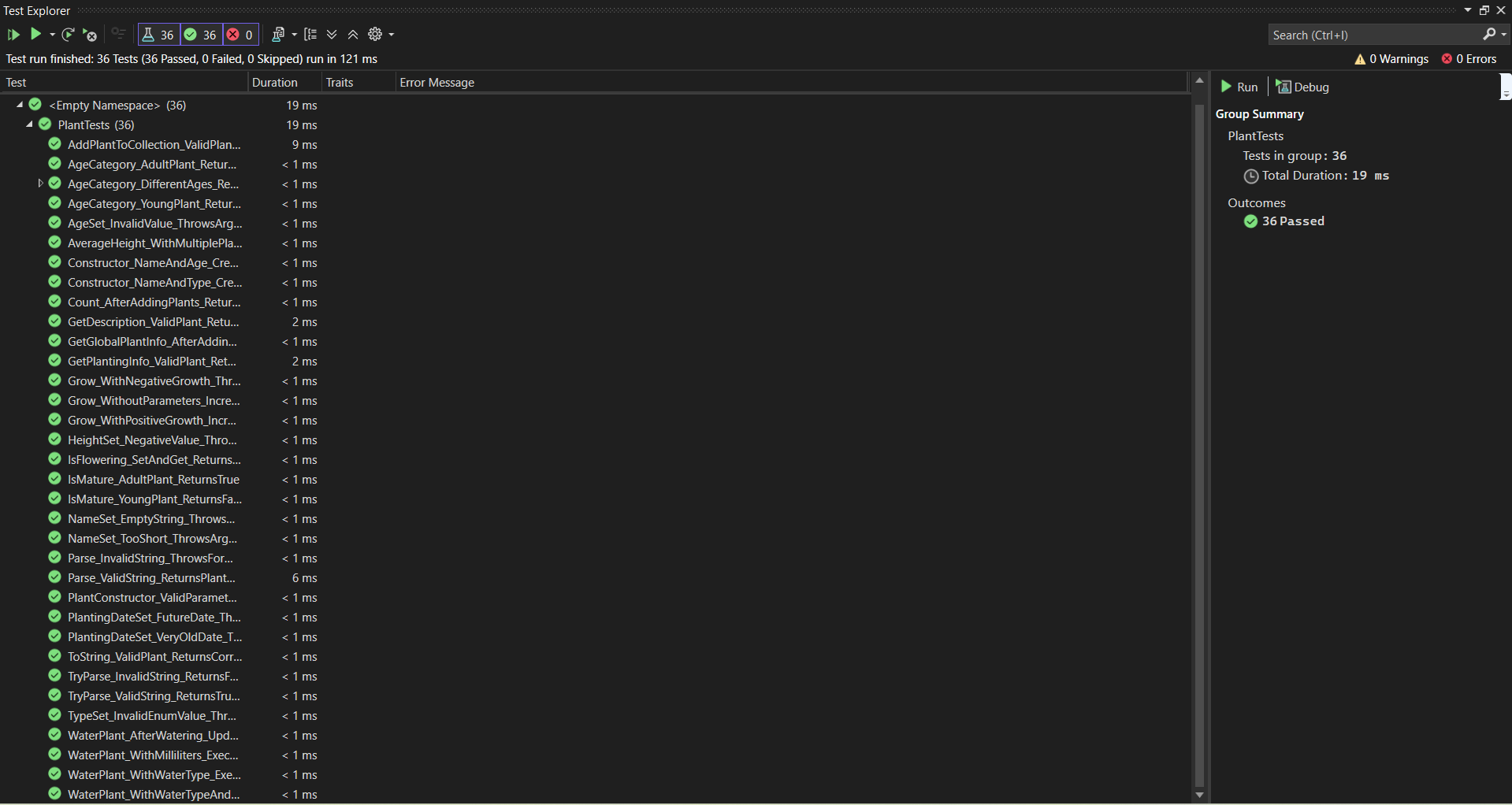
Assert.IsTrue(true);

}

}

**Результати запуску unit-тестів**

Рис. 1 результат тестування



**Код програми**

GitHub - https://github.com/TPoryshkin/OOP\_Lab5\_Poryshkin

**Файл Program.cs:**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**class Program**

**{**

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

**static int maxPlants;**

**static Random random = new Random();**

**static void Main(string[] args)**

**{**

**Console.OutputEncoding = System.Text.Encoding.UTF8;**

**InitializeMaxPlants();**

**ShowMenu();**

**}**

**static void InitializeMaxPlants()**

**{**

**while (true)**

**{**

**Console.Write("Введіть максимальну кількість рослин (N > 0): ");**

**if (int.TryParse(Console.ReadLine(), out maxPlants) && maxPlants > 0)**

**break;**

**Console.WriteLine("Помилка: Введіть коректне ціле число більше 0.");**

**}**

**}**

**static void ShowMenu()**

**{**

**while (true)**

**{**

**Console.WriteLine("\n--- Меню управління колекцією рослин ---");**

**Console.WriteLine("1 - Додати рослину");**

**Console.WriteLine("2 - Переглянути всі рослини");**

**Console.WriteLine("3 - Знайти рослину");**

**Console.WriteLine("4 - Продемонструвати поведінку");**

**Console.WriteLine("5 - Видалити рослину");**

**Console.WriteLine("6 - Продемонструвати static-методи");**

**Console.WriteLine("0 - Вийти з програми");**

**Console.Write("Ваш вибір: ");**

**string choice = Console.ReadLine();**

**switch (choice)**

**{**

**case "1": AddPlant(); break;**

**case "2": ViewAllPlants(); break;**

**case "3": FindPlant(); break;**

**case "4": DemonstrateBehavior(); break;**

**case "5": DeletePlant(); break;**

**case "6": DemonstrateStaticMethods(); break;**

**case "0": return;**

**default: Console.WriteLine("Некоректний вибір. Спробуйте ще."); break;**

**}**

**}**

**}**

**static void AddPlant()**

**{**

**if (plants.Count >= maxPlants)**

**{**

**Console.WriteLine($"Досягнуто максимум ({maxPlants}) рослин. Додавання неможливе.");**

**return;**

**}**

**Console.WriteLine("\n--- Додавання нової рослини ---");**

**Console.WriteLine("Оберіть спосіб створення:");**

**Console.WriteLine("1 - Вручну (основний конструктор)");**

**Console.WriteLine("2 - Випадковий конструктор");**

**Console.WriteLine("3 - Через рядок (парсинг)");**

**Console.Write("Ваш вибір: ");**

**string choice = Console.ReadLine();**

**try**

**{**

**Plant newPlant;**

**if (choice == "1")**

**{**

**newPlant = CreatePlantManually();**

**}**

**else if (choice == "2")**

**{**

**newPlant = CreatePlantRandom();**

**}**

**else if (choice == "3")**

**{**

**newPlant = CreatePlantFromString();**

**}**

**else**

**{**

**Console.WriteLine("Некоректний вибір.");**

**return;**

**}**

**plants.Add(newPlant);**

**Plant.AddPlantToCollection(newPlant);**

**Console.WriteLine($"Рослина '{newPlant.Name}' успішно додана!");**

**}**

**catch (Exception ex)**

**{**

**Console.WriteLine($"Помилка при додаванні: {ex.Message}");**

**}**

**}**

**static Plant CreatePlantFromString()**

**{**

**Console.Write("Введіть дані рослини у форматі: Назва,Тип,Вік,Висота,Дата посадки (рррр-мм-дд),Цвітуча (True/False): ");**

**Console.WriteLine("\nПриклад: Троянда,Flower,2,0.5,2023-05-15,True");**

**Console.Write("Введіть дані: ");**

**string input = Console.ReadLine();**

**if (Plant.TryParse(input, out Plant plant))**

**{**

**return plant;**

**}**

**else**

**{**

**throw new Exception("Не вдалося створити рослину з введеного рядка.");**

**}**

**}**

**static Plant CreatePlantManually()**

**{**

**Console.Write("Назва: ");**

**string name = Console.ReadLine();**

**Console.WriteLine("Тип (0-Дерево, 1-Кущ, 2-Квітка, 3-Трава, 4-Папороть, 5-Кактус, 6-Ліана): ");**

**PlantType type = (PlantType)GetValidatedInput("тип", 0, 6);**

**Console.Write("Вік (роки): ");**

**int age = GetValidatedInput("вік", 0, 5000);**

**Console.Write("Висота (метри): ");**

**double height = GetValidatedDoubleInput("висота", 0.01, 115.7);**

**Console.Write("Дата посадки (рррр-мм-дд): ");**

**DateTime plantingDate = GetValidatedDate();**

**Console.Write("Цвітуча рослина (так/ні): ");**

**bool isFlowering = Console.ReadLine().ToLower() == "так";**

**Plant plant = new Plant(name, type, age, height, plantingDate);**

**plant.IsFlowering = isFlowering;**

**return plant;**

**}**

**static Plant CreatePlantRandom()**

**{**

**int constructorType = random.Next(0, 4);**

**Plant newPlant;**

**string[] plantNames = { "Дуб", "Троянда", "Кактус", "Береза", "Лаванда", "Папороть", "Плющ", "Ялина" };**

**string randomName = plantNames[random.Next(plantNames.Length)];**

**switch (constructorType)**

**{**

**case 0:**

**Console.WriteLine("Створення за допомогою конструктора без параметрів...");**

**newPlant = new Plant();**

**newPlant.Name = randomName;**

**break;**

**case 1:**

**Console.WriteLine("Створення за допомогою конструктора з 2 параметрами (ім'я та тип)...");**

**PlantType randomType = (PlantType)random.Next(0, 7);**

**newPlant = new Plant(randomName, randomType);**

**break;**

**case 2:**

**Console.WriteLine("Створення за допомогою конструктора з 2 параметрами (ім'я та вік)...");**

**int randomAge = random.Next(1, 20);**

**newPlant = new Plant(randomName, randomAge);**

**break;**

**case 3:**

**Console.WriteLine("Створення за допомогою основного конструктора...");**

**PlantType type = (PlantType)random.Next(0, 7);**

**int age = random.Next(1, 50);**

**double height = Math.Round(random.NextDouble() \* 10 + 0.1, 2);**

**DateTime plantingDate = DateTime.Now.AddYears(-age).AddDays(-random.Next(0, 365));**

**newPlant = new Plant(randomName, type, age, height, plantingDate);**

**break;**

**default:**

**newPlant = new Plant();**

**break;**

**}**

**newPlant.IsFlowering = random.Next(0, 2) == 1;**

**return newPlant;**

**}**

**static int GetValidatedInput(string fieldName, int min, int max)**

**{**

**while (true)**

**{**

**if (int.TryParse(Console.ReadLine(), out int value) && value >= min && value <= max)**

**return value;**

**Console.Write($"Помилка: {fieldName} повинен бути від {min} до {max}. Спробуйте ще: ");**

**}**

**}**

**static double GetValidatedDoubleInput(string fieldName, double min, double max)**

**{**

**while (true)**

**{**

**if (double.TryParse(Console.ReadLine(), out double value) && value >= min && value <= max)**

**return value;**

**Console.Write($"Помилка: {fieldName} повинна бути від {min} до {max}. Спробуйте ще: ");**

**}**

**}**

**static DateTime GetValidatedDate()**

**{**

**while (true)**

**{**

**if (DateTime.TryParse(Console.ReadLine(), out DateTime date))**

**{**

**try**

**{**

**Plant testPlant = new Plant("Test", PlantType.Tree, 1, 1.0, date);**

**return date;**

**}**

**catch (ArgumentException ex)**

**{**

**Console.WriteLine($"Помилка: {ex.Message}");**

**}**

**}**

**else**

**{**

**Console.Write("Помилка: Некоректний формат дати. Використовуйте рррр-мм-дд: ");**

**}**

**}**

**}**

**static void ViewAllPlants()**

**{**

**if (plants.Count == 0)**

**{**

**Console.WriteLine("Колекція рослин порожня.");**

**return;**

**}**

**Console.WriteLine("\n--- Всі рослини ---");**

**Console.WriteLine("{0,-3} {1,-20} {2,-10} {3,-8} {4,-12} {5,-12} {6,-10} {7,-15}",**

**"#", "Назва", "Тип", "Вік", "Висота", "Посаджено", "Цвітуча", "Категорія");**

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

**{**

**var p = plants[i];**

**Console.WriteLine("{0,-3} {1,-20} {2,-10} {3,-8} {4,-12:F2} {5,-12:yyyy-MM-dd} {6,-10} {7,-15}",**

**i + 1, p.Name, p.Type, p.Age, p.Height, p.PlantingDate,**

**p.IsFlowering ? "Так" : "Ні", p.AgeCategory);**

**}**

**Console.WriteLine($"\nЗагальна кількість рослин: {Plant.Count}");**

**Console.WriteLine($"Середня висота рослин: {Plant.AverageHeight:F2} м");**

**}**

**static void FindPlant()**

**{**

**if (plants.Count == 0)**

**{**

**Console.WriteLine("Колекція рослин порожня.");**

**return;**

**}**

**Console.WriteLine("\n--- Пошук рослин ---");**

**Console.WriteLine("1 - По назві");**

**Console.WriteLine("2 - По типу");**

**Console.WriteLine("3 - По категорії віку");**

**Console.Write("Виберіть критерій пошуку: ");**

**string choice = Console.ReadLine();**

**List<Plant> results = new List<Plant>();**

**switch (choice)**

**{**

**case "1":**

**Console.Write("Введіть назву для пошуку: ");**

**string nameQuery = Console.ReadLine();**

**results = plants.Where(p => p.Name.Contains(nameQuery, StringComparison.OrdinalIgnoreCase)).ToList();**

**break;**

**case "2":**

**Console.WriteLine("Тип (0-Дерево, 1-Кущ, 2-Квітка, 3-Трава, 4-Папороть, 5-Кактус, 6-Ліана): ");**

**PlantType typeQuery = (PlantType)GetValidatedInput("тип", 0, 6);**

**results = plants.Where(p => p.Type == typeQuery).ToList();**

**break;**

**case "3":**

**Console.WriteLine("Категорія віку (1-Молода, 2-Доросла, 3-Стара): ");**

**int categoryChoice = GetValidatedInput("категорія", 1, 3);**

**string category = categoryChoice == 1 ? "Молода" : (categoryChoice == 2 ? "Доросла" : "Стара");**

**results = plants.Where(p => p.AgeCategory == category).ToList();**

**break;**

**default:**

**Console.WriteLine("Некоректний вибір.");**

**return;**

**}**

**DisplaySearchResults(results);**

**}**

**static void DisplaySearchResults(List<Plant> results)**

**{**

**if (results.Count == 0)**

**{**

**Console.WriteLine("Рослин не знайдено.");**

**return;**

**}**

**Console.WriteLine("\n--- Результати пошуку ---");**

**Console.WriteLine("{0,-3} {1,-20} {2,-10} {3,-8} {4,-12} {5,-12} {6,-10} {7,-15}",**

**"#", "Назва", "Тип", "Вік", "Висота", "Посаджено", "Цвітуча", "Категорія");**

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

**{**

**var p = results[i];**

**Console.WriteLine("{0,-3} {1,-20} {2,-10} {3,-8} {4,-12:F2} {5,-12:yyyy-MM-dd} {6,-10} {7,-15}",**

**i + 1, p.Name, p.Type, p.Age, p.Height, p.PlantingDate,**

**p.IsFlowering ? "Так" : "Ні", p.AgeCategory);**

**}**

**}**

**static void DemonstrateBehavior()**

**{**

**if (plants.Count == 0)**

**{**

**Console.WriteLine("Колекція рослин порожня.");**

**return;**

**}**

**ViewAllPlants();**

**Console.Write("Виберіть номер рослини для демонстрації: ");**

**if (int.TryParse(Console.ReadLine(), out int index) && index > 0 && index <= plants.Count)**

**{**

**Plant plant = plants[index - 1];**

**Console.WriteLine($"\n--- Демонстрація поведінки для {plant.Name} ---");**

**Console.WriteLine(plant.GetDescription());**

**Console.WriteLine(plant.GetPlantingInfo());**

**Console.WriteLine($"Доросла рослина: {(plant.IsMature() ? "Так" : "Ні")}");**

**Console.WriteLine($"Категорія за віком: {plant.AgeCategory}");**

**Console.WriteLine($"Цвітуча рослина: {(plant.IsFlowering ? "Так" : "Ні")}");**

**Console.WriteLine($"Останній полив: {plant.LastWatered}");**

**Console.WriteLine("\n--- Демонстрація перевантажених методів ---");**

**Console.WriteLine("\n1. Демонстрація перевантажених методів Grow:");**

**Console.WriteLine("Базовий метод Grow:");**

**plant.Grow(0.5);**

**Console.WriteLine("\nПеревантажений метод Grow() без параметрів:");**

**plant.Grow();**

**Console.WriteLine("\nПеревантажений метод Grow(int years):");**

**plant.Grow(2);**

**Console.WriteLine("\n2. Демонстрація перевантажених методів WaterPlant:");**

**Console.WriteLine("Базовий метод WaterPlant():");**

**plant.WaterPlant();**

**Console.WriteLine("\nПеревантажений метод WaterPlant(string waterType):");**

**plant.WaterPlant("дощова");**

**Console.WriteLine("\nПеревантажений метод WaterPlant(int milliliters):");**

**plant.WaterPlant(500);**

**Console.WriteLine("\nПеревантажений метод WaterPlant(string waterType, int milliliters):");**

**plant.WaterPlant("мінеральна", 300);**

**}**

**else**

**{**

**Console.WriteLine("Некоректний номер рослини.");**

**}**

**}**

**static void DeletePlant()**

**{**

**if (plants.Count == 0)**

**{**

**Console.WriteLine("Колекція рослин порожня.");**

**return;**

**}**

**Console.WriteLine("\n--- Видалення рослин ---");**

**Console.WriteLine("1 - За порядковим номером");**

**Console.WriteLine("2 - За назвою");**

**Console.Write("Виберіть спосіб видалення: ");**

**string choice = Console.ReadLine();**

**switch (choice)**

**{**

**case "1":**

**ViewAllPlants();**

**Console.Write("Введіть номер рослини для видалення: ");**

**if (int.TryParse(Console.ReadLine(), out int index) && index > 0 && index <= plants.Count)**

**{**

**Plant removed = plants[index - 1];**

**plants.RemoveAt(index - 1);**

**Plant.RemovePlantFromCollection(removed);**

**Console.WriteLine($"Рослина '{removed.Name}' була успішно видалена.");**

**}**

**else**

**{**

**Console.WriteLine("Некоректний номер.");**

**}**

**break;**

**case "2":**

**Console.Write("Введіть назву для видалення: ");**

**string nameToDelete = Console.ReadLine();**

**var toRemove = plants.Where(p => p.Name.Equals(nameToDelete, StringComparison.OrdinalIgnoreCase)).ToList();**

**if (toRemove.Count > 0)**

**{**

**foreach (var plant in toRemove)**

**{**

**Plant.RemovePlantFromCollection(plant);**

**}**

**plants.RemoveAll(p => p.Name.Equals(nameToDelete, StringComparison.OrdinalIgnoreCase));**

**Console.WriteLine($"Видалено {toRemove.Count} рослин з назвою '{nameToDelete}'.");**

**}**

**else**

**{**

**Console.WriteLine("Рослин з такою назвою не знайдено.");**

**}**

**break;**

**default:**

**Console.WriteLine("Некоректний вибір.");**

**break;**

**}**

**}**

**static void DemonstrateStaticMethods()**

**{**

**Console.WriteLine("\n--- Демонстрація static-методів ---");**

**Console.WriteLine($"\n1. Статичні властивості:");**

**Console.WriteLine($"Загальна кількість рослин: {Plant.Count}");**

**Console.WriteLine($"Середня висота рослин: {Plant.AverageHeight:F2} м");**

**Console.WriteLine($"\n2. Статичний метод GetGlobalPlantInfo():");**

**Console.WriteLine(Plant.GetGlobalPlantInfo());**

**Console.WriteLine($"\n3. Демонстрація методів Parse та TryParse:");**

**string correctString = "Троянда,Flower,2,0.5,2023-05-15,True";**

**Console.WriteLine($"Коректний рядок: {correctString}");**

**try**

**{**

**Plant parsedPlant = Plant.Parse(correctString);**

**Console.WriteLine("Метод Parse успішно створив рослину:");**

**Console.WriteLine($"Назва: {parsedPlant.Name}, Тип: {parsedPlant.Type}");**

**}**

**catch (Exception ex)**

**{**

**Console.WriteLine($"Помилка Parse: {ex.Message}");**

**}**

**Console.WriteLine($"\nДемонстрація TryParse з коректним рядком:");**

**if (Plant.TryParse(correctString, out Plant tryParsedPlant))**

**{**

**Console.WriteLine("TryParse успішно створив рослину:");**

**Console.WriteLine($"Назва: {tryParsedPlant.Name}, Вік: {tryParsedPlant.Age}");**

**}**

**else**

**{**

**Console.WriteLine("TryParse не вдалось створити рослину");**

**}**

**string incorrectString = "Неправильний,рядок";**

**Console.WriteLine($"\nДемонстрація TryParse з некоректним рядком: {incorrectString}");**

**if (Plant.TryParse(incorrectString, out Plant invalidPlant))**

**{**

**Console.WriteLine("TryParse успішно створив рослину (неочікувано)");**

**}**

**else**

**{**

**Console.WriteLine("TryParse не вдалось створити рослину (очікувана поведінка)");**

**}**

**Console.WriteLine($"\n4. Демонстрація методу ToString():");**

**if (plants.Count > 0)**

**{**

**Plant samplePlant = plants[0];**

**string plantString = samplePlant.ToString();**

**Console.WriteLine($"Результат ToString(): {plantString}");**

**if (Plant.TryParse(plantString, out Plant recreatedPlant))**

**{**

**Console.WriteLine("Рядок з ToString() успішно перетворено назад у рослину!");**

**Console.WriteLine($"Перевірка: {recreatedPlant.Name}, {recreatedPlant.Type}, {recreatedPlant.Age} років");**

**}**

**}**

**Console.WriteLine("\nДемонстрація завершена. Лічильник не змінився, оскільки демонстраційні рослини не додавались до колекції.");**

**}**

**}**

**Файл Plant.cs:**

using System;

using System.Linq;

using System.Globalization;

public class Plant

{

private static int \_count = 0;

private static double \_totalHeight = 0;

public static int Count => \_count;

public static double AverageHeight => \_count > 0 ? \_totalHeight / \_count : 0;

private string \_name;

private int \_age;

private double \_height;

private PlantType \_type;

private DateTime \_plantingDate;

public string Name

{

get => \_name;

set

{

if (string.IsNullOrWhiteSpace(value))

throw new ArgumentException("Назва не може бути порожньою.");

if (value.Length < 2 || value.Length > 50)

throw new ArgumentException("Назва повинна містити від 2 до 50 символів.");

if (!value.All(c => char.IsLetter(c) || c == ' '))

throw new ArgumentException("Назва може містити лише літери та пробіли.");

\_name = value;

}

}

public PlantType Type

{

get => \_type;

set

{

if (!Enum.IsDefined(typeof(PlantType), value))

throw new ArgumentException("Невірний тип рослини.");

\_type = value;

}

}

public int Age

{

get => \_age;

set

{

if (value < 0 || value > 5000)

throw new ArgumentException("Вік повинен бути в діапазоні від 0 до 5000 років.");

\_age = value;

}

}

public double Height

{

get => \_height;

set

{

if (value <= 0 || value > 115.7)

throw new ArgumentException("Висота повинна бути в діапазоні від 0 до 115.7 м (рекорд Hyperion).");

\_totalHeight -= \_height;

\_height = value;

\_totalHeight += \_height;

}

}

public DateTime PlantingDate

{

get => \_plantingDate;

set

{

ValidatePlantingDate(value);

\_plantingDate = value;

}

}

public bool IsFlowering { get; set; } = true;

public string AgeCategory

{

get

{

if (Age < 2) return "Молода";

if (Age < 10) return "Доросла";

return "Стара";

}

}

public string LastWatered { get; private set; } = "Ніколи";

public Plant(string name, PlantType type, int age, double height, DateTime plantingDate)

{

Console.WriteLine("Викликано основний конструктор з 5 параметрами");

\_name = name;

\_type = type;

\_age = age;

\_height = height;

\_plantingDate = plantingDate;

}

public Plant()

{

Console.WriteLine("Викликано конструктор без параметрів");

\_name = "Без назви";

\_type = PlantType.Flower;

\_age = 1;

\_height = 0.1;

\_plantingDate = DateTime.Now;

IsFlowering = true;

}

public Plant(string name, PlantType type) : this(name, type, 1, 0.5, DateTime.Now)

{

Console.WriteLine("Викликано конструктор з 2 параметрами (ім'я та тип)");

}

public Plant(string name, int age) : this(name, PlantType.Tree, age, 1.0, DateTime.Now.AddYears(-age))

{

Console.WriteLine("Викликано конструктор з 2 параметрами (ім'я та вік)");

}

public static void AddPlantToCollection(Plant plant)

{

if (plant != null)

{

\_count++;

\_totalHeight += plant.Height;

}

}

public static void RemovePlantFromCollection(Plant plant)

{

if (plant != null && \_count > 0)

{

\_count--;

\_totalHeight -= plant.Height;

}

}

public static string GetGlobalPlantInfo()

{

return $"Всього рослин: {Count}, Середня висота: {AverageHeight:F2} м";

}

public static Plant Parse(string s)

{

if (string.IsNullOrEmpty(s))

throw new ArgumentNullException(null, "Рядок не може бути порожнім або null");

string[] parts = s.Split(',');

if (parts.Length != 6)

throw new FormatException("Рядок повинен містити 6 значень розділених комами");

try

{

string name = parts[0];

PlantType type = (PlantType)Enum.Parse(typeof(PlantType), parts[1]);

int age = int.Parse(parts[2]);

double height = double.Parse(parts[3], CultureInfo.InvariantCulture);

DateTime plantingDate = DateTime.Parse(parts[4]);

bool isFlowering = bool.Parse(parts[5]);

return new Plant(name, type, age, height, plantingDate)

{

IsFlowering = isFlowering

};

}

catch (Exception ex)

{

throw new FormatException($"Помилка парсингу: {ex.Message}");

}

}

public static bool TryParse(string s, out Plant plant)

{

plant = null;

try

{

plant = Parse(s);

return true;

}

catch (Exception ex)

{

Console.WriteLine($"Помилка: {ex.Message}");

return false;

}

}

public override string ToString()

{

return $"{Name},{Type},{Age},{Height.ToString(CultureInfo.InvariantCulture)},{PlantingDate:yyyy-MM-dd},{IsFlowering}";

}

private void ValidatePlantingDate(DateTime date)

{

if (date.Year < 1900)

throw new ArgumentException("Дата посадки не може бути раніше 1900 року.");

if (date > DateTime.Now)

throw new ArgumentException("Дата посадки не може бути у майбутньому.");

}

private string FormatWateringTime(DateTime time)

{

return time.ToString("dd.MM.yyyy HH:mm");

}

public void WaterPlant()

{

DateTime wateringTime = DateTime.Now;

LastWatered = FormatWateringTime(wateringTime);

Console.WriteLine($"{Name} було полито. Час останнього поливу: {LastWatered}");

}

public void WaterPlant(string waterType)

{

WaterPlant();

Console.WriteLine($"Використано тип води: {waterType}");

}

public void WaterPlant(int milliliters)

{

WaterPlant();

Console.WriteLine($"Використано {milliliters} мл води");

}

public void WaterPlant(string waterType, int milliliters)

{

WaterPlant();

Console.WriteLine($"Використано {milliliters} мл води типу: {waterType}");

}

private string GetFormattedDescription()

{

return $"{Name} ({Type}) - {Age} років, {Height} м";

}

public string GetDescription()

{

return GetFormattedDescription();

}

public void Grow(double growth)

{

if (growth <= 0)

throw new ArgumentException("Ріст повинен бути більше 0.");

\_totalHeight -= \_height;

\_height += growth;

\_totalHeight += \_height;

Console.WriteLine($"{Name} виріс на {growth}м. Нова висота: {Height}м");

}

public void Grow()

{

double defaultGrowth = 0.1;

Grow(defaultGrowth);

}

public void Grow(int years)

{

if (years <= 0)

throw new ArgumentException("Кількість років повинна бути більше 0.");

double growthPerYear = 0.2;

double totalGrowth = years \* growthPerYear;

\_totalHeight -= \_height;

\_height += totalGrowth;

\_age += years;

\_totalHeight += \_height;

Console.WriteLine($"{Name} виріс на {totalGrowth}м за {years} років. Нова висота: {Height}м, Новий вік: {Age} років");

}

public string GetPlantingInfo()

{

return $"{Name} було висаджено {PlantingDate:dd.MM.yyyy}.";

}

public bool IsMature()

{

return Age > 5;

}

}

**Файл PlantType.cs:**

public enum PlantType

{

Tree,

Shrub,

Flower,

Grass,

Fern,

Succulent,

Vine

}