МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

Практична робота (Lab-3)

з дисципліни «Об’єктно-орієнтоване програмування»

*назва дисципліни*

на тему: «Конструктори, перевантаження (overloading)»

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:

* створювати конструктор класу;
* перевантажувати конструктор класу;
* перевантажувати методи класу, які реалізують його поведінку.

Завдання для виконання роботи

На основі отриманого на лекції 3 теоретичного матеріалу скорегувати програму для практичної роботи Lab-2 наступним чином:

1. Додати до класу не менше трьох перевантажених конструкторів.

Примітка:

* 1. один з конструкторів має бути без параметрів (при його використанні застосувати ініціалізатори);
  2. має бути реалізація хоча б одного конструктора через виклик іншого власного конструктора.

1. Для деякого public-методу класу додати його перевантажені версії\*.
   1. пам’ятаємо про різну сигнатуру методів.
2. Структура меню залишається з практичної роботи Lab-2: 1 – Додати об’єкт 2 – Переглянути всі об’єкти 3 – Знайти об’єкт 4 – Продемонструвати поведінку 5 – Видалити об’єкт 0 – Вийти з програми

При додаванні об’єктів передбачити їх створення різними конструкторами, наприклад, за допомогою виклику випадково обраного конструктора. Користувач має отримувати повідомлення з інформацією про те, який саме конструктор справився. До програмної реалізації пункту меню «Продемонструвати поведінку» додати демонстрацію роботи перевантажених методів.

1. Детально протестувати програму. Мають бути протестовані всі пункти меню. Проводимо тестування не тільки на коректних значеннях, а також не забуваємо перевірити і некоректні значення, які може ввести користувач.
2. Оформити звіт.

**Короткий опис класу**

Характеристики (Властивості):

1. Назва (Name) - унікальна ідентифікаційна назва рослини
   * Валідація: 2-50 символів, тільки літери та пробіли
2. Тип (Type) - категорія рослини (дерево, кущ, квітка, трава, папороть, кактус, ліана)
3. Вік (Age) - кількість років рослини (від 0 до 5000)
   * Валідація: діапазон 0-5000 років
4. Висота (Height) - розмір рослини в метрах (від 0.01 до 115.7)
   * Валідація: діапазон 0.01-115.7 м (рекорд Hyperion)
5. Дата посадки (PlantingDate) - дата, коли було висаджено рослину
   * Валідація: не раніше 1900 року та не у майбутньому
6. Цвітуча (IsFlowering) - чи є рослина цвітучою (так/ні)
7. Категорія віку (AgeCategory) - автоматично розраховується на основі віку:
   * "Молода" - вік < 2 роки
   * "Доросла" - вік 2-9 років
   * "Стара" - вік ≥ 10 років
8. Час останнього поливу (LastWatered) - час, коли рослину востаннє поливали
   * Формат: "[dd.MM](https://dd.mm/" \t "_blank).yyyy HH:mm" або "Ніколи"

Конструктори (4 перевантажені версії):

1. Plant() - конструктор без параметрів
   * Ініціалізація: стандартні значення ("Без назви", Flower, 1 рік, 0.1 м, поточна дата)
2. Plant(string name, PlantType type) - через виклик основного конструктора
   * Автоматично: вік=1, висота=0.5, поточна дата посадки
3. Plant(string name, int age) - через виклик основного конструктора
   * Автоматично: тип=Tree, висота=1.0, дата=поточна дата мінус вік
4. Plant(string name, PlantType type, int age, double height, DateTime plantingDate) - основний конструктор

Поведінка (Методи):

Основні методи:

1. GetDescription() - надає текстовий опис рослини у форматі "Назва (Тип) - Вік років, Висота м"
2. GetPlantingInfo() - повертає інформацію про дату посадки у форматі "Назва було висаджено [dd.MM](https://dd.mm/" \t "_blank).yyyy"
3. IsMature() - перевіряє, чи є рослина дорослою (вік > 5 років)
4. ValidatePlantingDate() - перевіряє коректність дати посадки

Перевантажені методи:

Grow() - 3 версії:

* Grow(double growth) - збільшує висоту на задану величину (в метрах)
* Grow() - збільшує висоту на стандартну величину (0.1 м)
* Grow(int years) - збільшує висоту та вік на задану кількість років

WaterPlant() - 4 версії:

* WaterPlant() - базовий полив, встановлює час останнього поливу
* WaterPlant(string waterType) - полив з вказівкою типу води
* WaterPlant(int milliliters) - полив з вказівкою об'єму води
* WaterPlant(string waterType, int milliliters) - полив з вказівкою типу та об'єму води

Додаткові особливості:

* Автоматичне розрахункове поле - AgeCategory обчислюється автоматично на основі віку
* Вбудована валідація - всі властивості мають перевірку на коректність значень
* Гнучкість створення - 4 різних конструктори для різних сценаріїв ініціалізації
* Перевантажена функціональність - методи Grow та WaterPlant мають кілька варіантів використання
* Форматування даних - спеціальне форматування для часу поливу та описів

Клас забезпечує повний цикл управління рослиною: від створення різними способами, через різноманітні операції догляду, до отримання детальної інформації про стан рослини.

**Class diagram**

### **Приватні поля:**

* \_name - зберігає назву рослини
* \_age - зберігає вік у роках
* \_height - зберігає висоту в метрах
* \_type - зберігає тип рослини з переліку PlantType
* \_plantingDate - зберігає дату посадки

### **Публічні властивості:**

* Name - з валідацією (2-50 символів, тільки літери та пробіли)
* Type - з валідацією (має належати до значень PlantType)
* Age - з валідацією (0-5000 років)
* Height - з валідацією (0.01-115.7 метрів)
* PlantingDate - з валідацією (не раніше 1900 року, не пізніше поточної дати)
* IsFlowering - чи є рослина цвітучою
* AgeCategory - тільки для читання (автоматично обчислюється: "Молода", "Доросла", "Стара")
* LastWatered - тільки для читання, встановлюється приватним методом

### **Конструктори (4 перевантажені версії):**

1. Без параметрів (з ініціалізаторами)
2. З 2 параметрами (назва, тип) - викликає основний конструктор
3. З 2 параметрами (назва, вік) - викликає основний конструктор
4. Основний з 5 параметрами

### **Публічні методи:**

* **Основні методи:** GetDescription(), GetPlantingInfo(), IsMature()
* **Перевантажені методи Grow:** 3 версії з різними параметрами
* **Перевантажені методи WaterPlant:** 4 версії з різними параметрами

### **Приватні методи:**

* ValidatePlantingDate() - перевіряє коректність дати посадки
* FormatWateringTime() - форматує час поливу у строку
* GetFormattedDescription() - допоміжний метод для форматування опису

### **Перелік PlantType:**

Містить 7 можливих типів рослин: Tree, Shrub, Flower, Grass, Fern, Succulent, Vine

**Програмна реалізація класу**

**Файл PlantType.cs:**

**public enum PlantType**

**{**

**Tree,**

**Shrub,**

**Flower,**

**Grass,**

**Fern,**

**Succulent,**

**Vine**

**}**

**Файл Plant.cs:**

using System;

public class Plant

{

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).");

\_height = value;

}

}

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)");

}

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.");

Height += growth;

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

}

public void Grow()

{

double defaultGrowth = 0.1;

Height += defaultGrowth;

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

}

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;

Height += totalGrowth;

Age += years;

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()

{

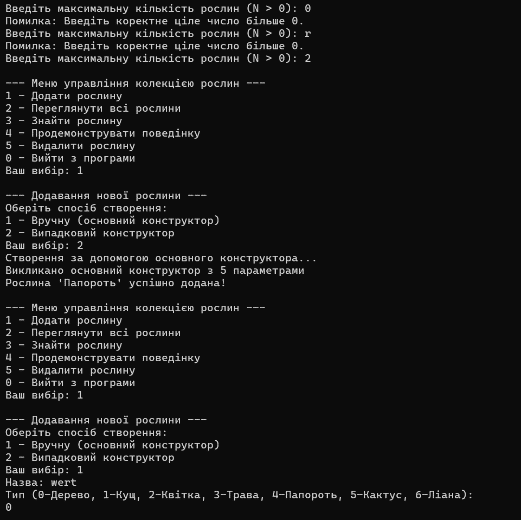
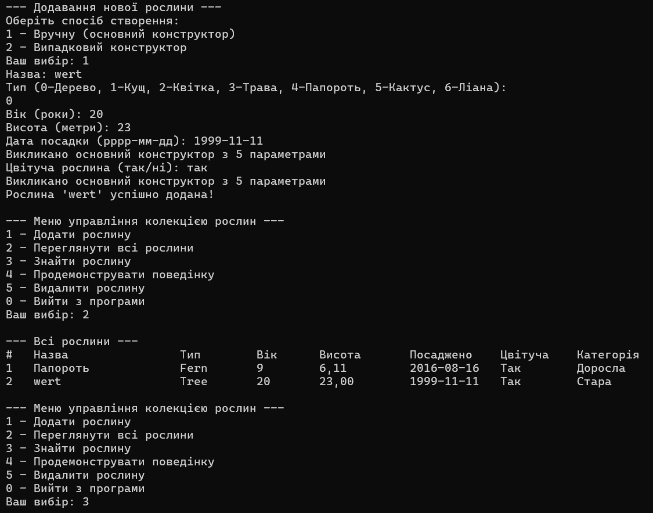
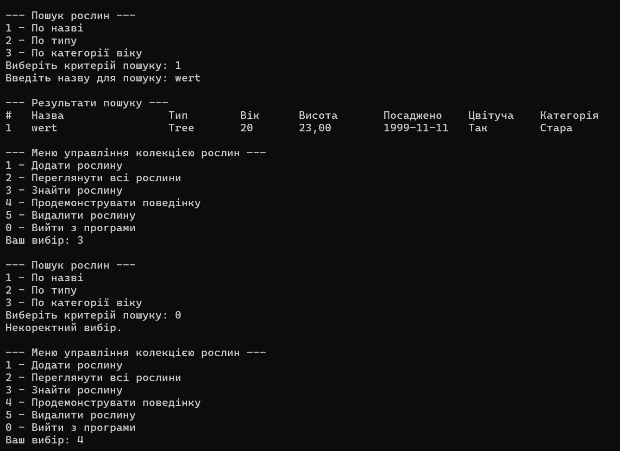
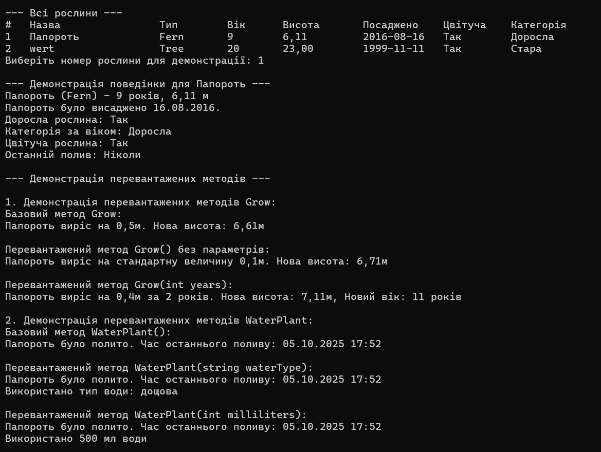
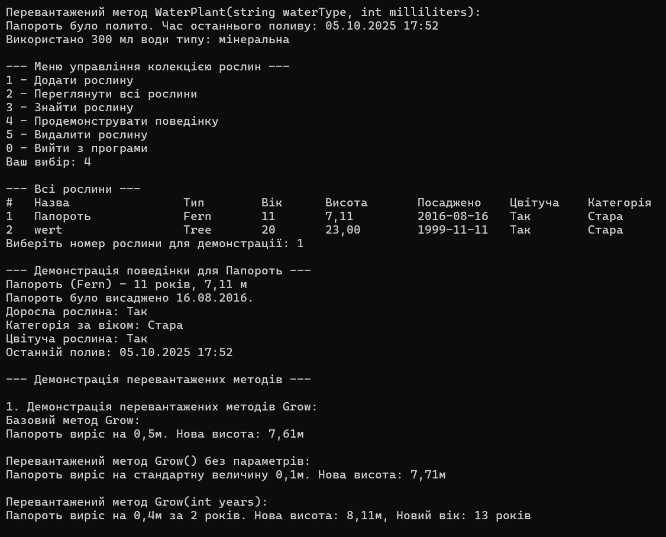
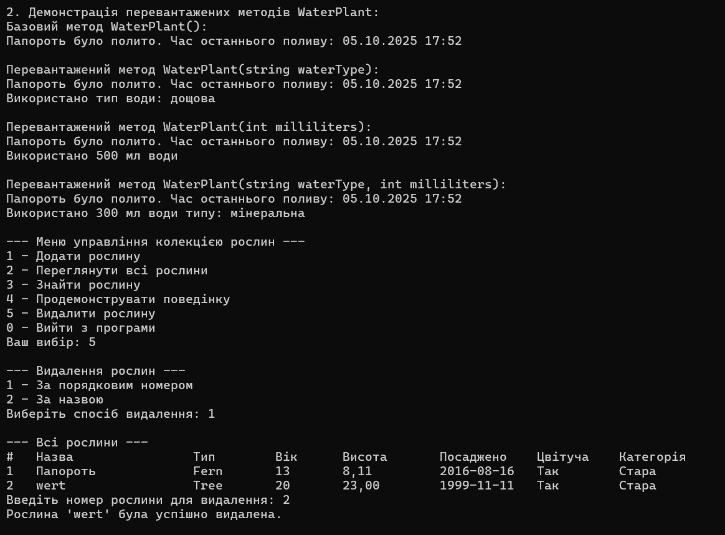
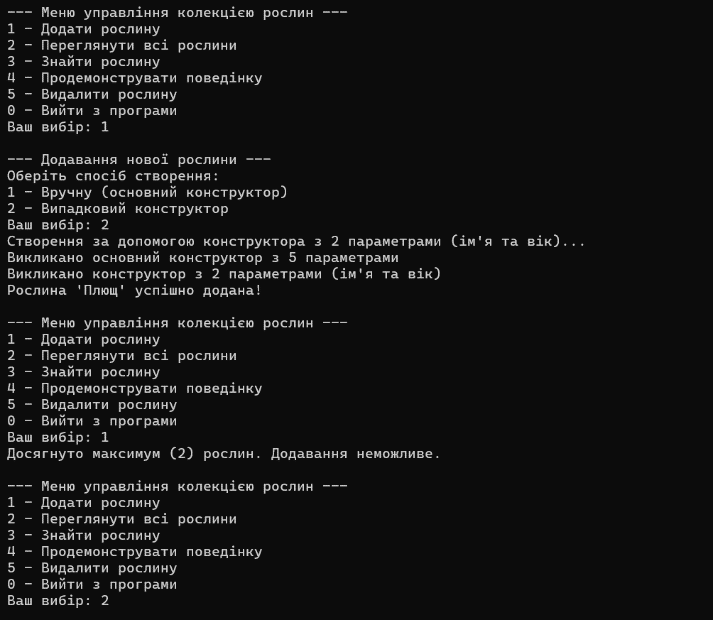
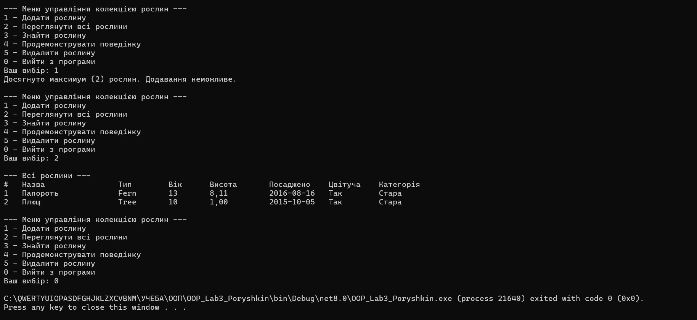
return Age > 5;

}

}

**Результати детального тестування програми**

Рис. 1-8 результати тестування



**Код програми**

GitHub - https://github.com/TPoryshkin/OOP\_Lab3\_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("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 "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.Write("Ваш вибір: ");**

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

**try**

**{**

**Plant newPlant;**

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

**{**

**newPlant = CreatePlantManually();**

**}**

**else**

**{**

**newPlant = CreatePlantRandom();**

**}**

**plants.Add(newPlant);**

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

**}**

**catch (Exception ex)**

**{**

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

**}**

**}**

**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);**

**}**

**}**

**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);**

**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)**

**{**

**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;**

**}**

**}**

**}**

**Файл Plant.cs:**

using System;

public class Plant

{

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).");

\_height = value;

}

}

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)");

}

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.");

Height += growth;

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

}

public void Grow()

{

double defaultGrowth = 0.1;

Height += defaultGrowth;

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

}

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;

Height += totalGrowth;

Age += years;

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

}