МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

Практична робота (Lab-2)

з дисципліни «Об’єктно-орієнтоване програмування»

*назва дисципліни*

на тему: «Інкапсуляція, властивості»

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:  
– використовувати принцип інкапсуляції ООП;  
– створювати і використовувати властивості для забезпечення встановлення коректних значень і зчитування закритих полів класів;  
– створювати і використовувати закриті і загальнодоступні методи в класі.

**Завдання для виконання роботи**

На основі отриманого на лекції 2 теоретичного матеріалу скорегувати програму для практичної роботи Lab-1 наступним чином:

1. Всі поля класу повинні бути інкапсульовані за допомогою модифікатора доступу private.
2. Для кожного private-поля, яке необхідне для зовнішньої взаємодії, в класі мають бути додані відповідні public-властивості (public-properties)\*.

\*якщо предметна область вимагає, то секція set властивості обов’язково повинна мати відповідні перевірки. При спробі встановлення некоректного значення має генеруватися відповідний за типом exception з інформативним повідомленням.

1. До класу додати хоча б одну автовластивість, для якої передбачите значення за замовчуванням.
2. До класу додати хоча б одну обчислювальну властивість, яка не буде пов’язана з відповідним private-полем класу і буде використовувати інші властивості та/або методи.
3. Хоча б одна властивість повинна мати різний рівень доступу для секцій get і set\*\*\*.

\*\*\*пам’ятаємо, що модифікатор доступу секції має бути більш обмежувальним, ніж модифікатор самої властивості.

1. У public-методах використати private-методи для приховування деталей реалізації public-методів, що також буде забезпечувати принцип інкапсуляції.
2. Встановлення та зчитування значень полів об’єктів в основній програмі реалізувати через додані у клас властивості.

Примітка: при встановленні значення властивості може бути згенерований exception, а значить необхідне його оброблення.

1. Меню в програмі залишається з практичної роботи Lab-1:
   1. Додати об’єкт
   2. Переглянути всі об’єкти
   3. Знайти об’єкт
   4. Продемонструвати поведінку
   5. Видалити об’єкт
   6. Вийти з програми
2. 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.
3. Оформити звіт.

**Короткий опис класу**

**Клас: Plant (Рослина)**

**Характеристики:**

1. **Назва (Name)** - унікальна ідентифікаційна назва рослини
2. **Тип (Type)** - категорія рослини (дерево, кущ, квітка тощо)
3. **Вік (Age)** - кількість років рослини (від 0 до 5000)
4. **Висота (Height)** - розмір рослини в метрах (від 0 до 115.7)
5. **Дата посадки (PlantingDate)** - дата, коли було висаджено рослину
6. **Цвітуча (IsFlowering)** - чи є рослина цвітучою (так/ні) [НОВЕ]
7. **Категорія віку (AgeCategory)** - автоматично розраховується на основі віку (Молода, Доросла, Стара) [НОВЕ]
8. **Час останнього поливу (LastWatered)** - час, коли рослину востаннє поливали [НОВЕ]

**Поведінка:**

1. **GetDescription()** - надає текстовий опис рослини
2. **Grow()** - збільшує висоту рослини на задану величину
3. **GetPlantingInfo()** - повертає інформацію про дату посадки
4. **IsMature()**- перевіряє, чи є рослина дорослою (вік > 5 років)
5. **WaterPlant()** - поливає рослину, встановлюючи час останнього поливу [НОВЕ]
6. **ValidatePlantingDate()** - перевіряє коректність дати посадки [НОВЕ]

**Class diagram**

**Клас: Plant**

**Приватні поля:**

1. \_name: string
2. \_age: int
3. \_height: double
4. \_type: PlantType [НОВЕ]
5. \_plantingDate: DateTime [НОВЕ]

**Публічні властивості:**

1. Name: string {get; set;} (з валідацією)
2. Type: PlantType {get; set;} (з валідацією) [НОВЕ]
3. Age: int {get; set;} (з валідацією)
4. Height: double {get; set;} (з валідацією)
5. PlantingDate: DateTime {get; set;} (з валідацією) [НОВЕ]
6. IsFlowering: bool {get; set;} [НОВЕ]
7. AgeCategory: string {get;} (read-only) [НОВЕ]
8. LastWatered: string {get; private set;} [НОВЕ]

**Публічні методи:**

1. Plant(string, PlantType, int, double, DateTime) - конструктор
2. Grow(double): void
3. GetDescription(): string
4. GetPlantingInfo(): string
5. IsMature(): bool
6. WaterPlant(): void [НОВЕ]

**Приватні методи:** [НОВІ]

1. ValidatePlantingDate(DateTime): void
2. FormatWateringTime(DateTime): string
3. GetFormattedDescription(): string

**Перелік PlantType:**

1. Tree
2. Shrub
3. Flower
4. Grass
5. Fern
6. Succulent
7. Vine

**Програмна реалізація класу**

**Файл PlantType.cs:**

**public enum PlantType**

**{**

**Tree,**

**Shrub,**

**Flower,**

**Grass,**

**Fern,**

**Succulent,**

**Vine**

**}**

**Файл Plant.cs:**

public class Plant

{

private string \_name;

private int \_age;

private double \_height;

private PlantType \_type; *// [НОВЕ] Приватне поле для типу*

private DateTime \_plantingDate; *// [НОВЕ] Приватне поле для дати посадки*

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;

}

}

*// [НОВЕ] Властивість Type з валідацією*

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;

}

}

*// [НОВЕ] Властивість PlantingDate з валідацією*

public DateTime PlantingDate

{

get => \_plantingDate;

set

{

ValidatePlantingDate(value); *// [НОВЕ] Виклик методу валідації*

\_plantingDate = value;

}

}

*// [НОВЕ] Нові властивості*

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)

{

Name = name;

Type = type;

Age = age;

Height = height;

PlantingDate = plantingDate;

}

*// [НОВЕ] Приватний метод валідації дати посадки*

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.");

}

*// [НОВЕ] Приватний метод форматування часу поливу*

private string FormatWateringTime(DateTime time)

{

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

}

*// [НОВЕ] Публічний метод поливу рослини*

public void WaterPlant()

{

DateTime wateringTime = DateTime.Now;

LastWatered = FormatWateringTime(wateringTime);

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

}

*// [НОВЕ] Приватний метод для форматування опису*

private string GetFormattedDescription()

{

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

}

public string GetDescription()

{

return GetFormattedDescription(); *// [НОВЕ] Використання приватного методу*

}

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 string GetPlantingInfo()

{

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

}

public bool IsMature()

{

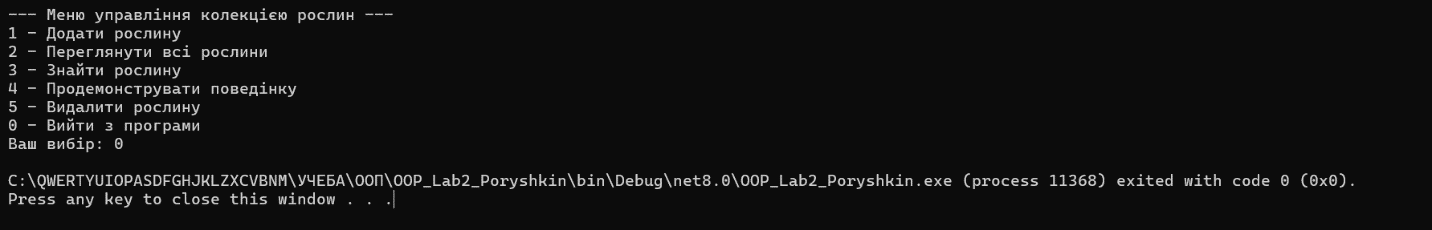
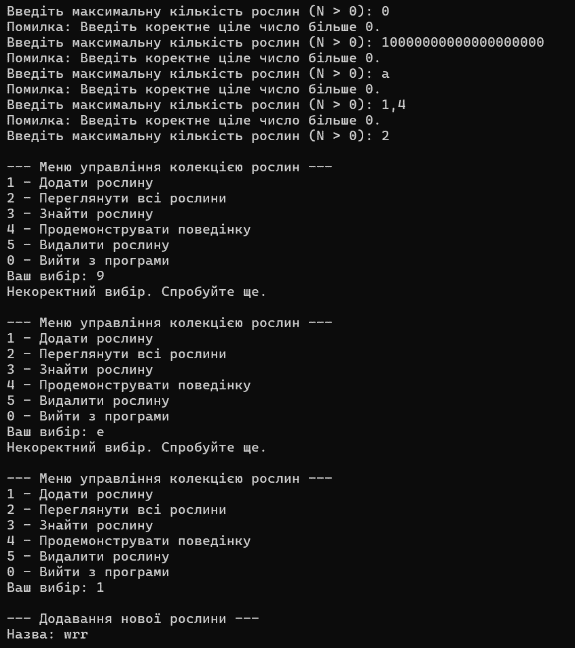
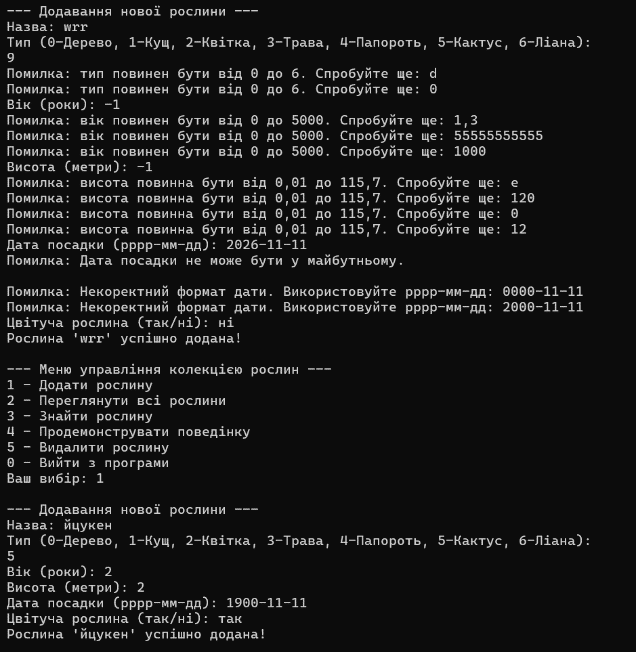
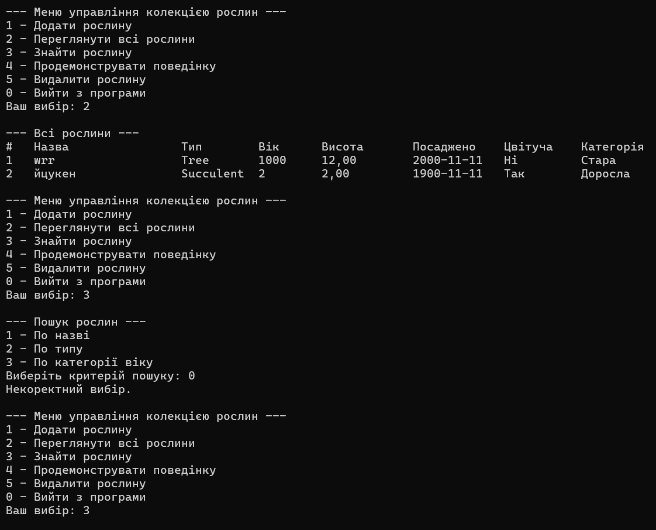
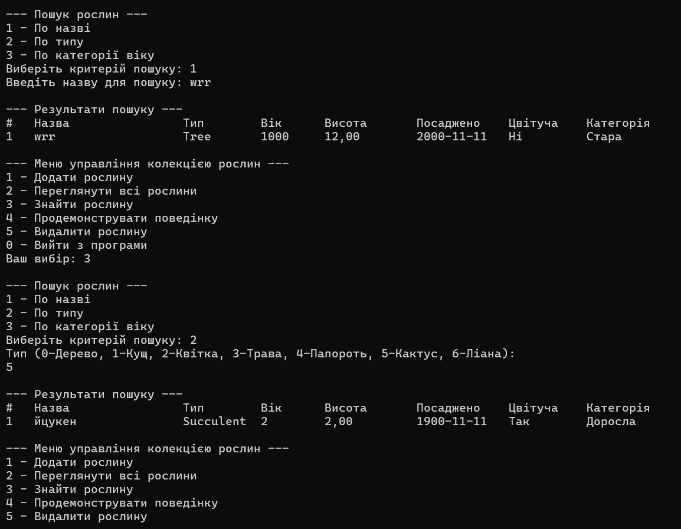
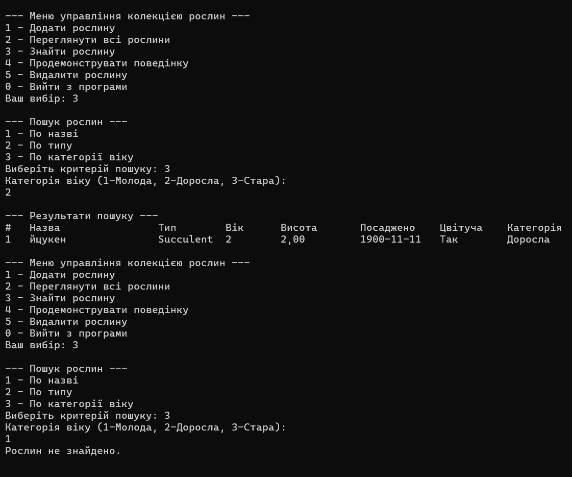
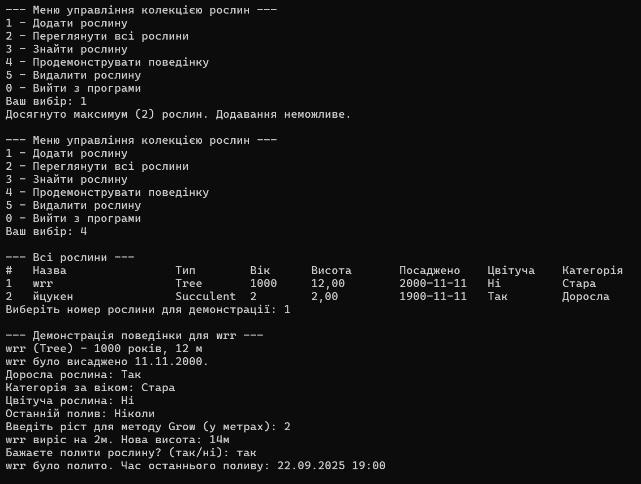
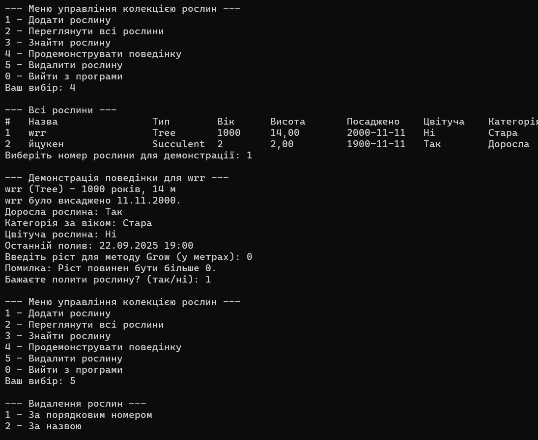
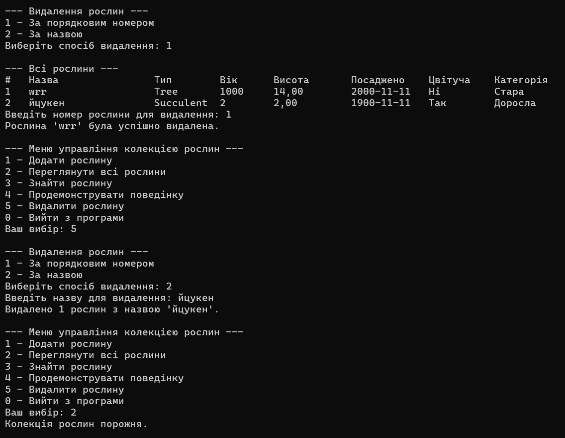
return Age > 5;

}

}

**Результати детального тестування програми**

Рис. 1-9 результати тестування



**Код програми**

GitHub - https://github.com/TPoryshkin/OOP\_Lab2\_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 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 ---");**

**try**

**{**

**Plant newPlant = CreatePlantManually();**

**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 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.Write("Введіть ріст для методу Grow (у метрах): ");**

**if (double.TryParse(Console.ReadLine(), out double growth))**

**{**

**try**

**{**

**plant.Grow(growth);**

**}**

**catch (Exception ex)**

**{**

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

**}**

**}**

**else**

**{**

**Console.WriteLine("Некоректне значення росту.");**

**}**

**Console.Write("Бажаєте полити рослину? (так/ні): ");**

**if (Console.ReadLine().ToLower() == "так")**

**{**

**plant.WaterPlant();**

**}**

**}**

**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:**

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)

{

Name = name;

Type = type;

Age = age;

Height = height;

PlantingDate = plantingDate;

}

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}");

}

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 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

}