МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

Практична робота (Lab-1)

з дисципліни «Об’єктно-орієнтоване програмування»

*назва дисципліни*

на тему: «Класи, об’єкти»

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:

- аналізувати предметну область і виявляти класи;

- використовувати принцип абстракції виділення характеристик і поведінки класів;

- реалізовувати класи мовою С#;

- створювати і працювати з об’єктами класів.

На основі отриманого на лекції 1 теоретичного матеріалу розробити консольну (або обрати інший тип) програму, яка:

1. Має опис деякого класу (предметну область обрати самостійно, будьте креативними!), клас повинен мати закриті (private) і загальнодоступні (public) поля\*, які описують характеристики класу, та методи, які задають поведінку класу\*.

\*“хоча б одне поле має бути типу \*\*enum\*\*, також для характеристик класу треба використовувати не менше 4-5 різних типів даних.\*

\*\*реалізувати enum і клас в окремих файлах \*.cs.\*

2. Для кожної характеристики класу сформувати обмеження. Перевірка зазначених обмежень має відбуватися в основній програмі (файл Program.cs). Характеристикам об’єктів можна привласнювати тільки коректні значення.

Наприклад, для класу \*Student обмеження для характеристики string name\* можуть бути наступними:

- довжина імені має бути від 3 до 12 символів;

- ім’я може містити тільки букви латинського алфавіту.

Обмеження для характеристики \*Date birthDate\* можуть бути наступними:

- формат дати dd.MM.yyyy;

- значення не раніше за 01.01.1960;

- значення не пізніше за поточну дату.

Обмеження для характеристики \*double GPA\* (середній бал) можуть бути наступними:

- GPA – число в діапазоні [0..100].

3. У програмі має бути передбачена можливість зберігати до N об'єктів

класу (N>0 і задається користувачем, наприклад, при старті

програми\*\*\*).

\*\*\* має бути реалізована перевірка введення коректного значення N.

\*\*\_Примітка: для зберігання використання N об’єктів можна використовувати масив\_\*\*

або колекцію List<T>.

4. В програмі має бути реалізоване наступне меню:

1 – Додати об’єкт

2 – Переглянути всі об’єкти

3 – Знайти об’єкт

4 – Продемонструвати поведінку

5 – Видалити об’єкт

0 – Вийти з програми

Під час \*\*додавання об’єкта користувачем всі його поля мають заповнюватися\*\*

обов’язково в ручному режимі, а також можуть заповнюватися і в автоматичному

режимі. Передбачити неможливість введення більше ніж N об’єктів.

При \*\*перегляді всіх доданих об’єктів детальну інформацію про них треба\*\*

виводити у табличній формі. Передбачити нумерацію об’єктів в таблиці виводу.

\*\*Пошук реалізувати по будь-яким двом характеристикам, у програмі передбачити\*\*

вибір користувачем характеристики для пошуку (наприклад, шукаємо або за кольору,

або за ціною). Якщо введеному значенню характеристики відповідають декілька

об’єктів, то всі вони мають бути виведені як результат пошуку у табличній формі.

При \*\*демонстрації поведінки необхідно показати виконання всіх реалізованих\*\*

public методів класу; одним з можливих варіантів реалізації є використання підменю.

\*\*Видалення об’єкта реалізувати за порядковим номером у таблиці виводу, а\*\*

також за однією з характеристик. При видаленні за значенням характеристики мають

видалятися всі об’єкти, які мають введене користувачем значення.

Якщо при перегляді, пошуку, видаленні не знайдено \_жодного об’єкта, то\_

користувач повинен отримати відповідне повідомлення!

Програма має завершувати свою роботу тільки після вибору користувачем

пункту меню «Вийти з програми».

5. Детально протестувати програму. Мають бути протестовані всі пункти

меню, а також пам’ятаємо про перевірку значень, які вводить

користувач, на відповідність обмеженням. Проводимо тестування не

тільки на коректних значеннях, а також не забуваємо перевірити і

некоректні значення, які може ввести користувач.

6. 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

**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)

**Class diagram**

Клас: Plant

Приватні поля (private):

1. \_name: string- назва рослини
2. \_age: int - вік рослини
3. \_height: double - висота рослини

Публічні властивості (public):

1. Name: string - назва рослини (з валідацією)
2. Type: PlantType - тип рослини (enum)
3. Age: int - вік рослини (з валідацією)
4. Height: double - висота рослини (з валідацією)
5. PlantingDate: DateTime - дата посадки

Публічні методи (public):

1. Plant(string, PlantType, int, double, DateTime) - конструктор
2. Grow(double) - збільшує висоту рослини
3. GetDescription(): string - повертає опис рослини
4. GetPlantingInfo(): string - повертає інформацію про посадку
5. IsMature(): bool - перевіряє, чи доросла рослина

Перелік: PlantType

Можливі значення: Tree, Shrub, Flower, Grass, Fern, Succulent, Vine

Типи даних:

1. string - текстовий рядок
2. int - ціле число
3. double - число з плаваючою точкою
4. DateTime - дата і час
5. PlantType - перелік типів рослин
6. bool - логічний тип (true/false)

**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;

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; set; }

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; set; }

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

{

Name = name;

Type = type;

Age = age;

Height = height;

PlantingDate = plantingDate;

}

public string GetDescription()

{

return $"{Name} — це {Type.ToString().ToLower()}, віком {Age} років.";

}

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()

{

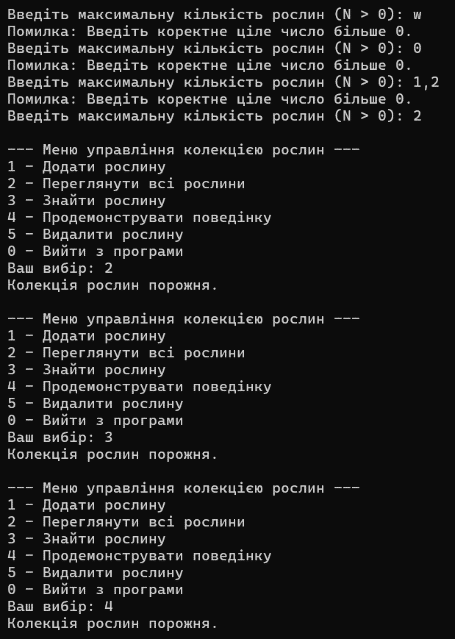
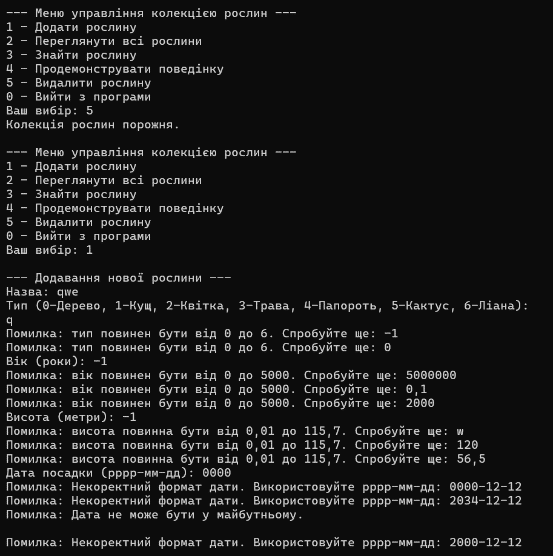
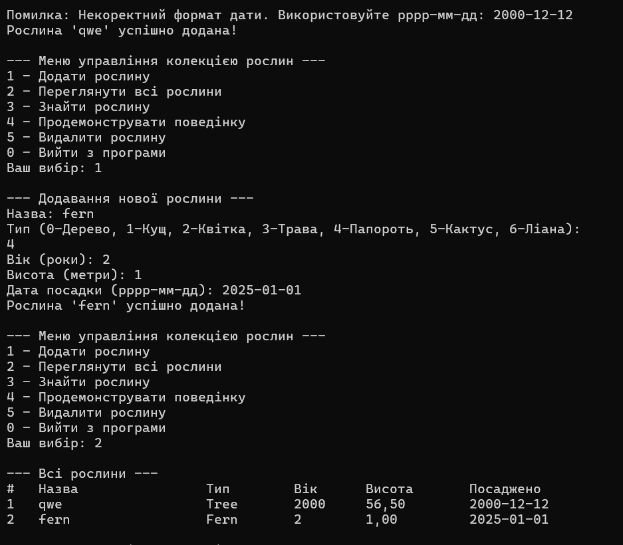
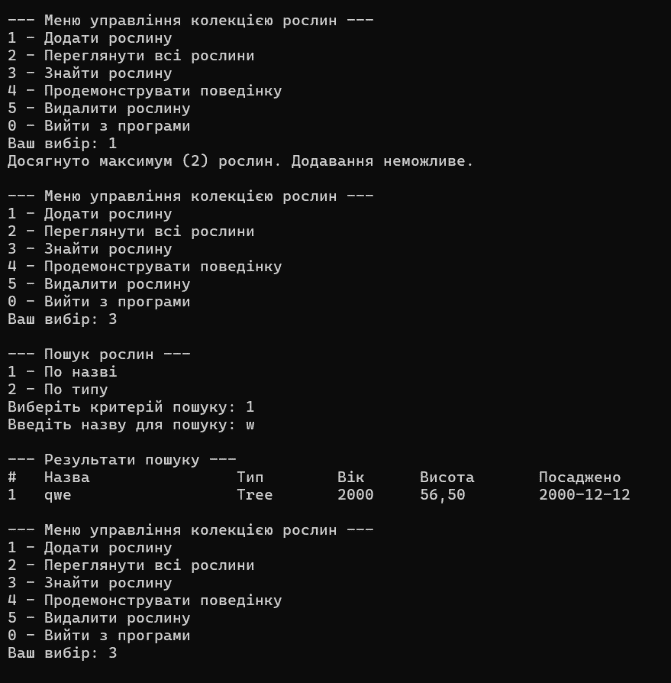
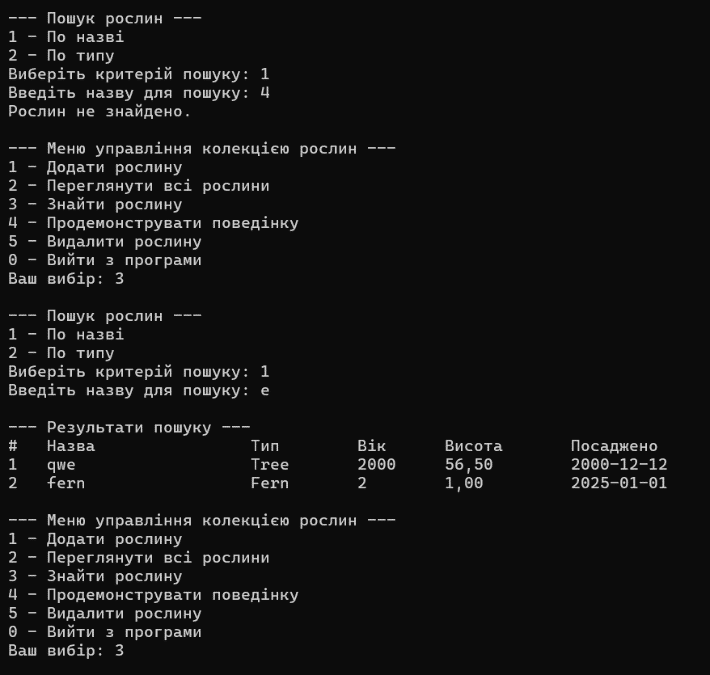
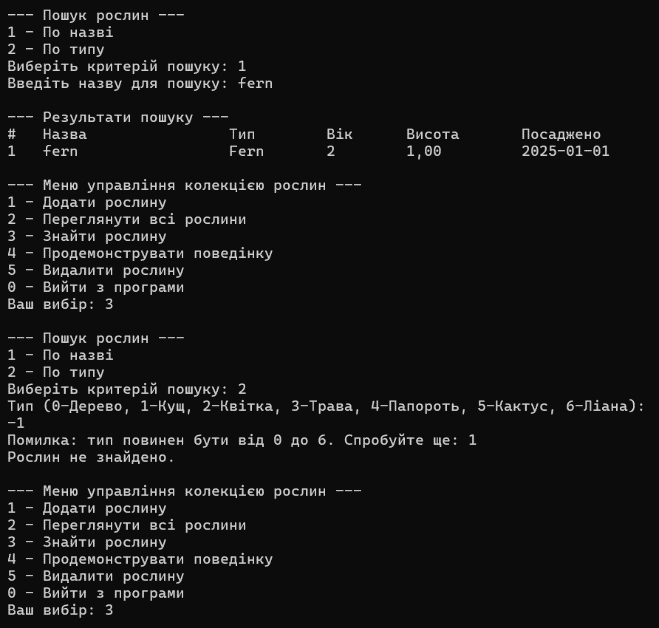
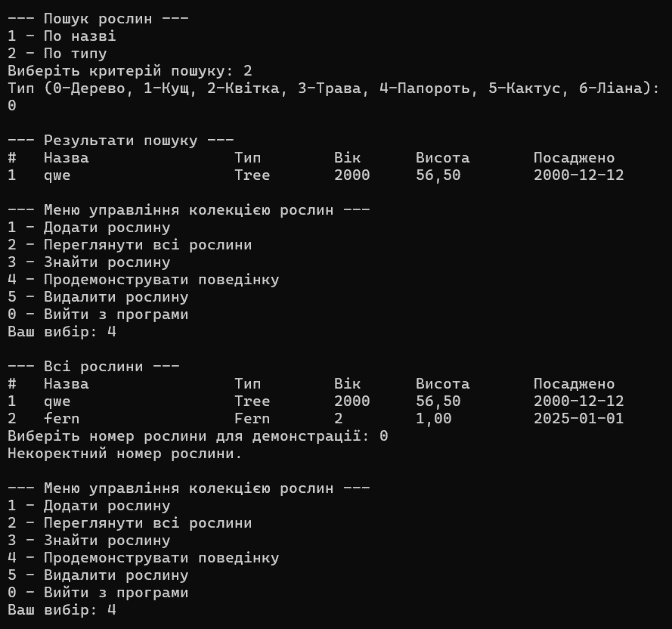
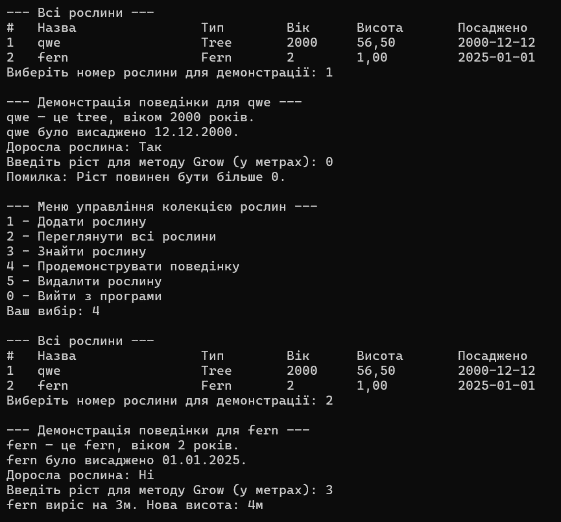
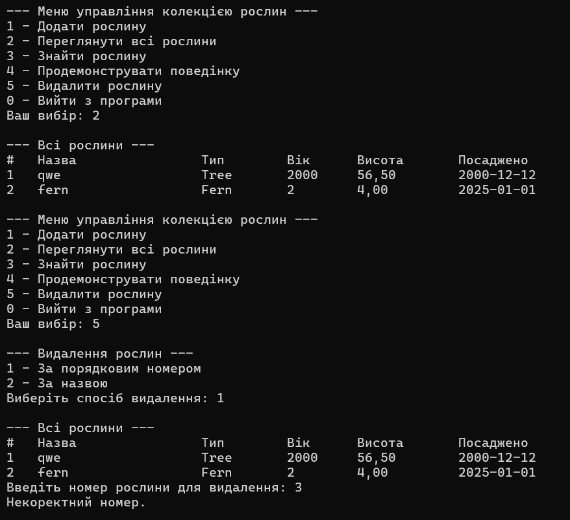
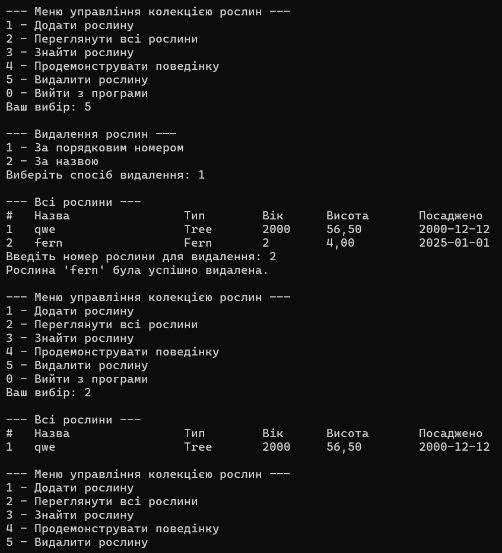
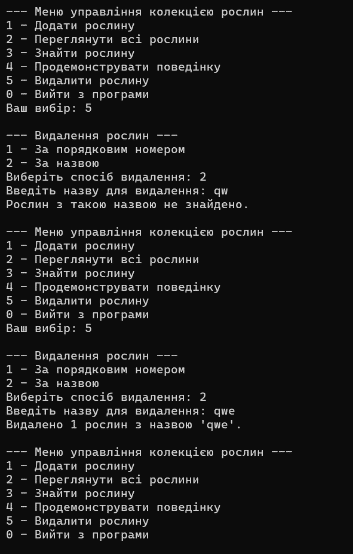
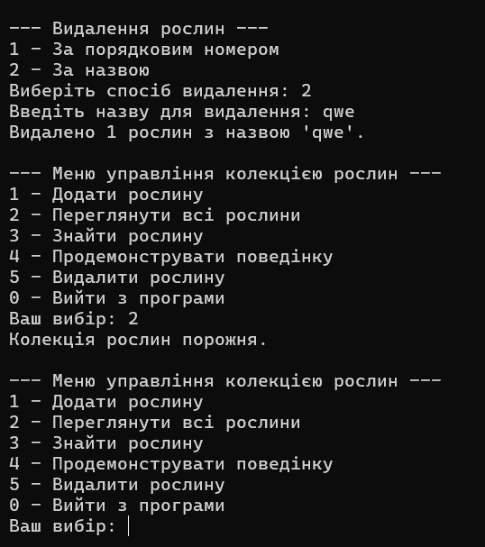
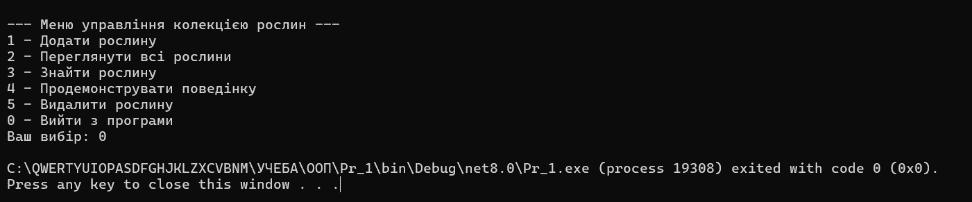
return Age > 5;

}

}

**Результати детального тестування програми**

Рис. 1-13 результати тестування



**Код програми**

GitHub - https://github.com/TPoryshkin/OOP\_Lab1\_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();**

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

**}**

**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))**

**{**

**if (date.Year < 1900)**

**Console.WriteLine("Помилка: Дата не може бути раніше 1900 року.");**

**else if (date > DateTime.Now)**

**Console.WriteLine("Помилка: Дата не може бути у майбутньому.");**

**else**

**return date;**

**}**

**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}",**

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

**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}",**

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

**}**

**}**

**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.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;**

**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}",**

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

**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}",**

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

**}**

**}**

**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.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.");**

**}**

**}**

**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;**

**}**

**}**

**}**