Харківський університет радіоелектроніки

Факультет комп'ютерних наук

Кафедра програмної інженерії

ЗВІТ

до практичного заняття з дисципліни

"Архітектура програмного забезпечення"

на тему: "Патерни проєктування"

Виконав ст. гр. ПЗПІ-22-5

Разіньков Євгеній Олександрович

Перевірив

ст. викладач кафедри ПІ

Сокорчук Ігор Петрович

Харків 2024

ІСТОРІЯ ЗМІН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Дата | Версія звіту | Опис змін та виправлень |
| 1 | 16.03.2025 | 0.1 | Додано титульну сторінку та історію змін |
| 2 | 16.03.2025 | 0.3 | Створено розділ «Завдання» |
| 3 | 17.03.2025 | 0.7 | Додано мету, хід роботи та висновки |
| 4 | 17.03.2025 | 1 | Додано додатки |

ЗАВДАННЯ

У рамках практичного заняття 1 (ПЗ1) необхідно виконати такі завдання:

1. Підготувати доповідь на тему: «Шаблон (патерн) проєктування ПЗ», використовуючи матеріали з книги Ґамма, “ООП. Патерни проєктування” (GoF), за винятком патерна Singleton.
2. Створити презентацію доповіді, оформивши слайди відповідно до правил оформлення навчальних матеріалів.
3. Записати відеопрезентацію доповіді та опублікувати її на YouTube.
4. Оформити та завантажити звіт до виконаної роботи на dl.nure.ua.
5. При необхідності створити GitHub-акаунт з обліковим записом у домені @nure.ua.
6. Створити GitHub-репозиторій із вказаною назвою для зберігання матеріалів з дисципліни.
7. Створити та заповнити файл README.md у корені створеного репозиторію, відповідно до вимог завдання.
8. Створити у репозиторії директорію Pract1 для завантаження звіту та вихідного коду.
9. Експортувати звіт у текстовому форматі та завантажити його у директорію Pract1 у GitHub-репозиторії.
10. Завантажити у піддиректорію з кодом увесь програмний код, який розглядається у доповіді ПЗ1.

Мета: Ознайомитися з основними принципами патерна **"Легковес" (Flyweight)**, зрозуміти його застосування для оптимізації використання пам’яті та навчитися реалізовувати його на практиці.

Хід роботи

У рамках цього завдання я підготував презентацію на тему "Патерн Легковес", у якій розглянув ключові особливості цього підходу, його переваги та недоліки, а також відмінності від інших патернів.

Я обрав цю тему, оскільки Легковес є ефективним інструментом для оптимізації програм, що працюють із великою кількістю об’єктів, таких як ігри, графічні редактори чи системи кешування.

У презентації я розглянув такі аспекти:

1. Призначення та ідея патерна
   * Пояснив, що Легковес дозволяє зменшити використання пам’яті шляхом розділення внутрішнього та зовнішнього стану об'єктів.
   * Навів приклади, коли варто застосовувати цей підхід.
2. Структура та принцип роботи
   * Розповів, як патерн ділить стан об'єкта на внутрішній (спільний) і зовнішній (унікальний).
   * Показав класичну реалізацію Легковеса на прикладі дерев у графічній системі.
3. Фабрика Легковесів
   * Описав підхід до створення та повторного використання об'єктів через фабрику, яка уникає дублювання однакових екземплярів.
   * Показав, як реалізувати цей механізм у коді.
4. Застосування та ефективність
   * Пояснив, у яких ситуаціях Легковес найбільш ефективний, зокрема:  
      1. Коли система працює з великою кількістю об'єктів.  
      2. Коли пам’ять витрачається через дублювання однакових даних.  
     3. Коли об'єкти можуть ділити частину свого стану між собою.
5. Переваги та недоліки
   * Розглянув плюси та мінуси патерна, зокрема економію пам'яті, але водночас ускладнення коду.
   * Пояснив, як правильно використовувати цей підхід, щоб уникнути зайвих накладних витрат.
6. Взаємозв'язок із іншими патернами
   * Порівняв Легковес із Одиночкою (Singleton), Компоновщиком (Composite) та Фасадом (Facade).
   * Пояснив, коли ці патерни можуть використовуватися разом.

ВИСНОВКИ

У результаті виконання роботи я розібрався з принципами патерна Flyweight, його використанням та реалізацією. Цей підхід дозволяє значно оптимізувати витрати пам’яті, особливо у великих системах, що працюють із безліччю однотипних об'єктів.  
Презентація містить практичні приклади та пояснення, які допоможуть зрозуміти, як застосовувати цей патерн у реальних проєктах.

ДОДАТОК А

Відеозапис доповіді на YouTube: https://youtu.be/TJMebcF-mds

ДОДАТОК Б



Рисунок Б.1 – Титульна сторінка

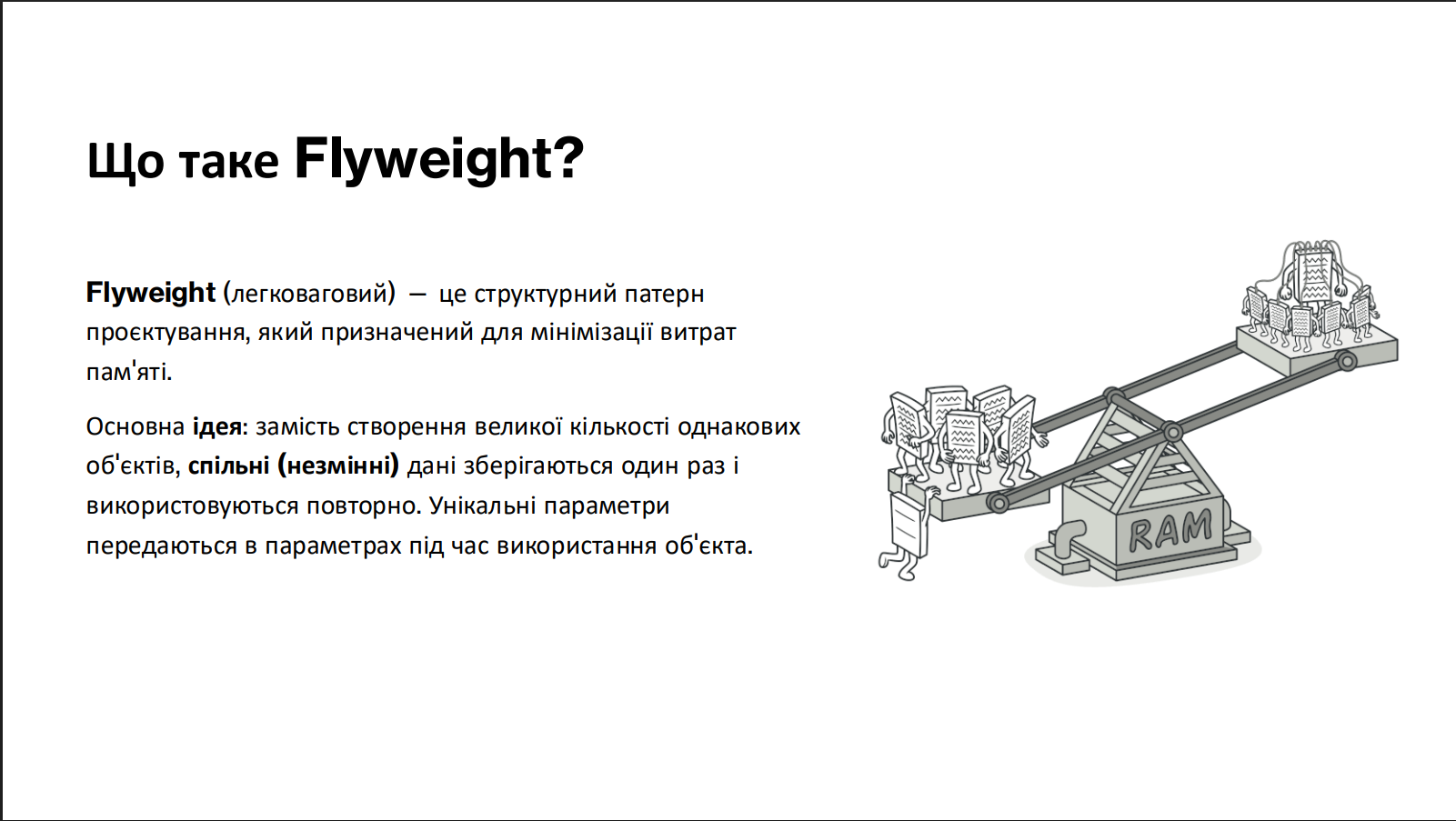


Рисунок Б.2 – Патерн Flyweight

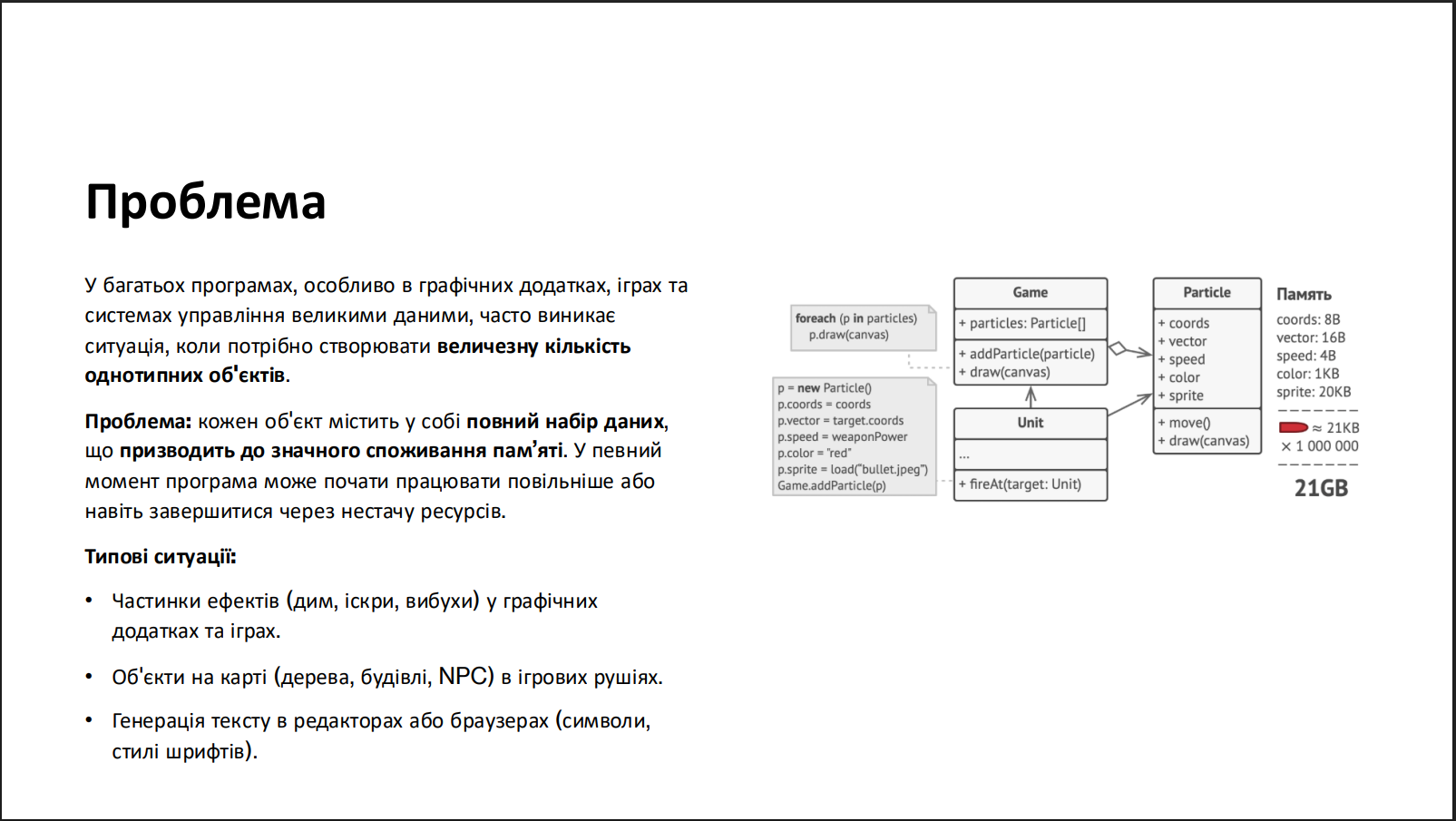


Рисунок Б.3 – Приклад проблеми коли треба використовувати цей патерн

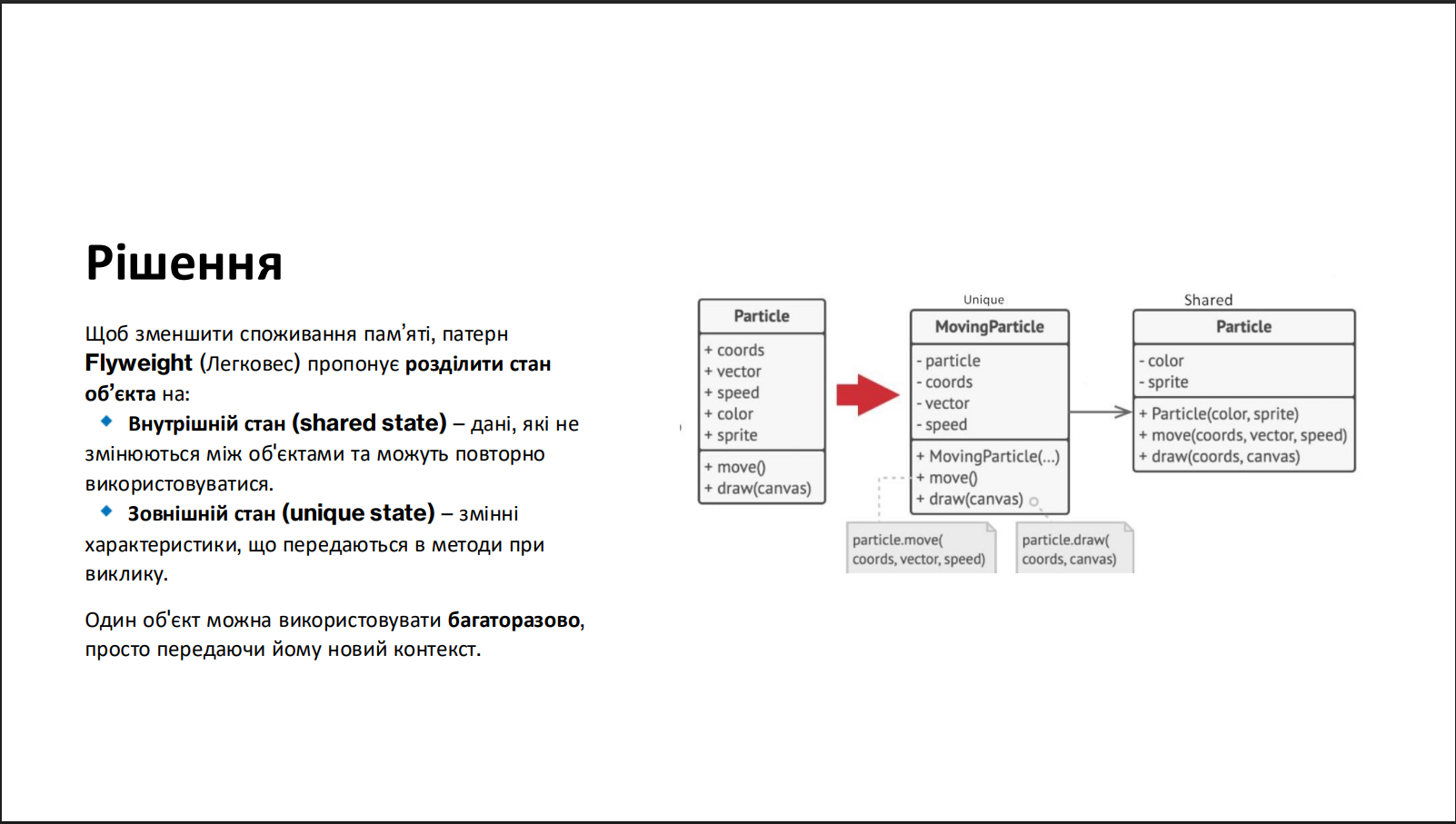


Рисунок Б.4 – Детальний опис Flyweight

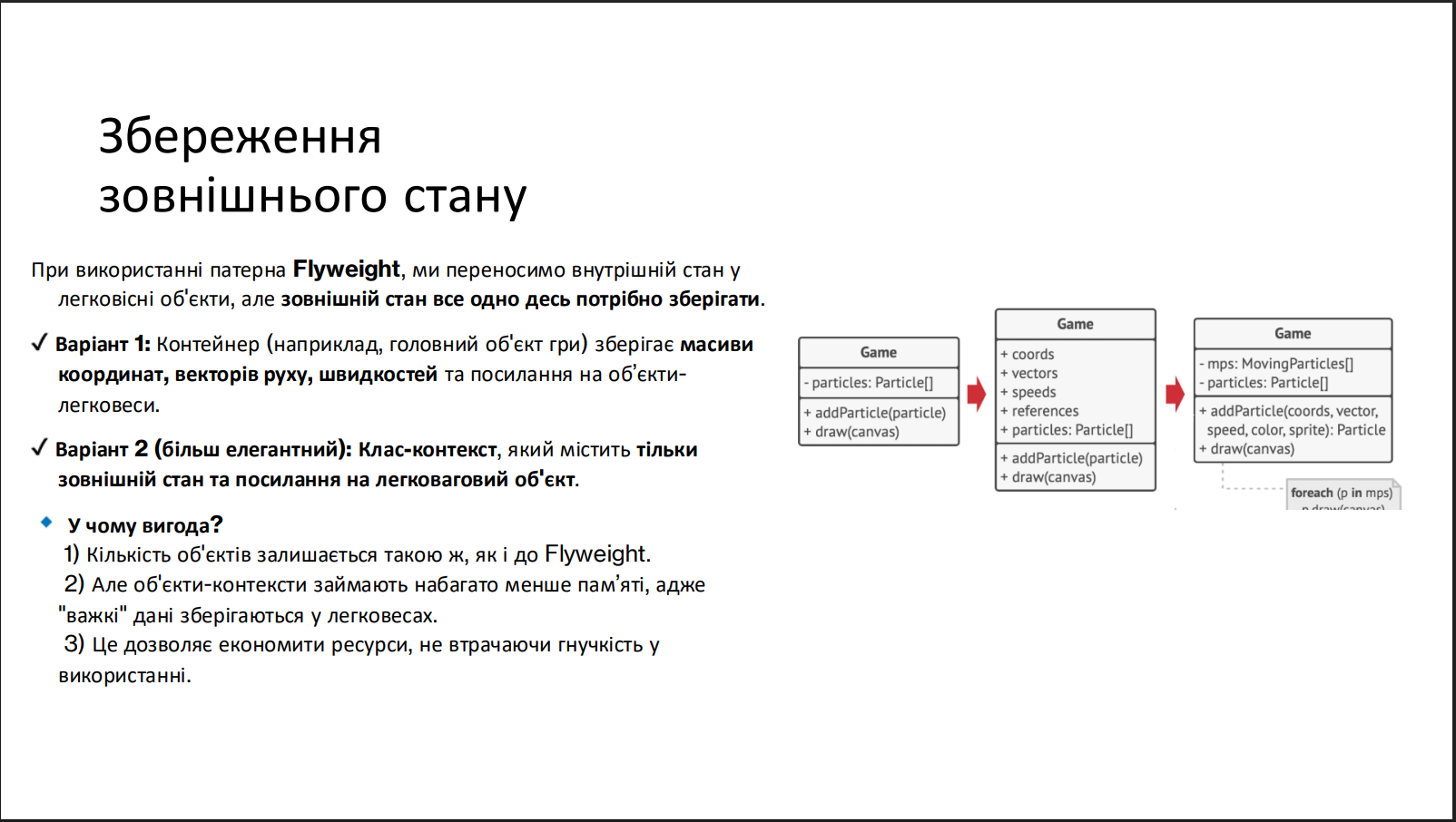


Рисунок Б.5 – Варіанти зберігання об’єктів у патерні

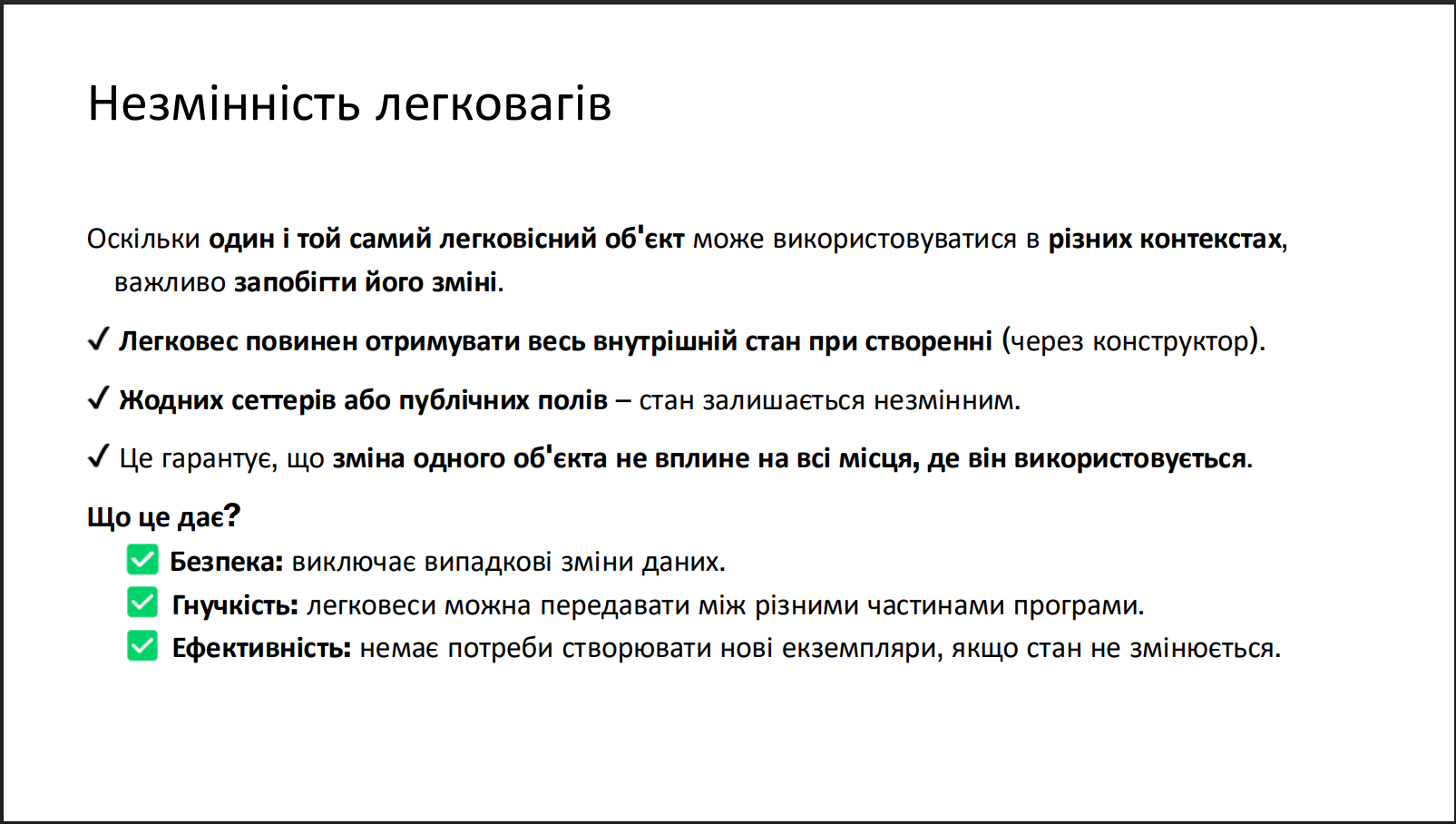


Рисунок Б.6 – Інкапсуляція та безпека легковагів

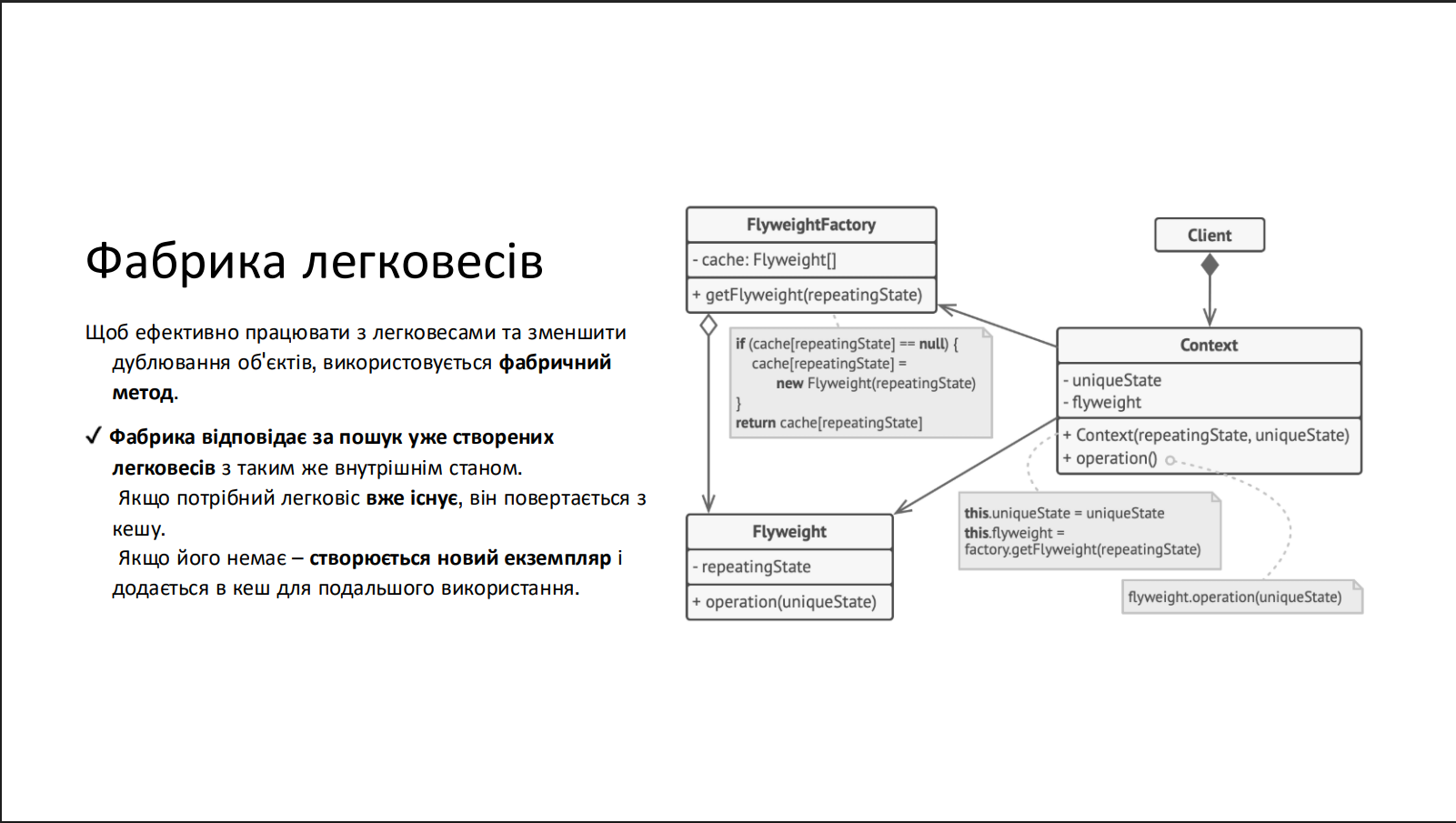


Рисунок Б.7 – Використання фабрики

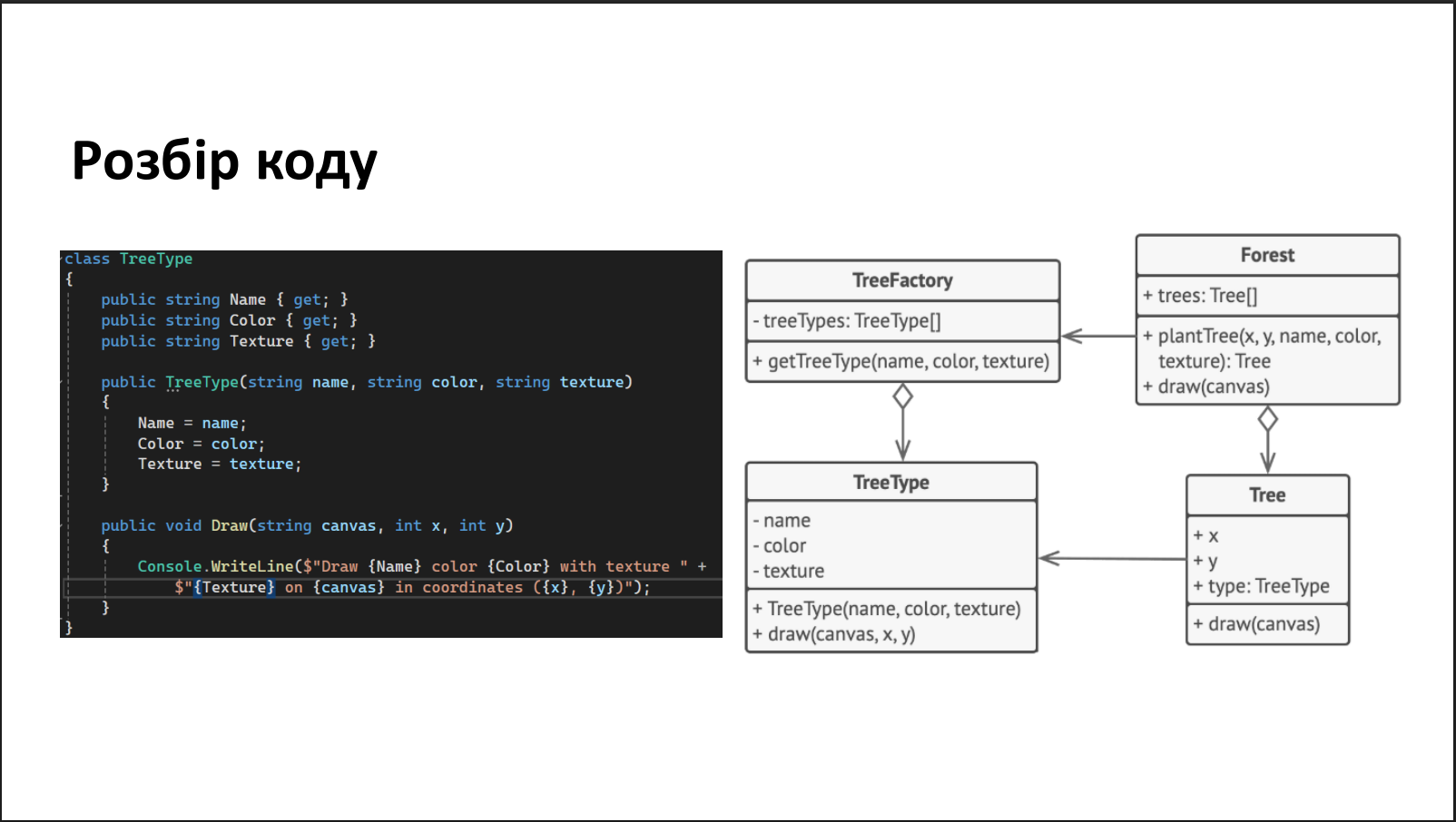


Рисунок Б.8 – Приклади програмного коду

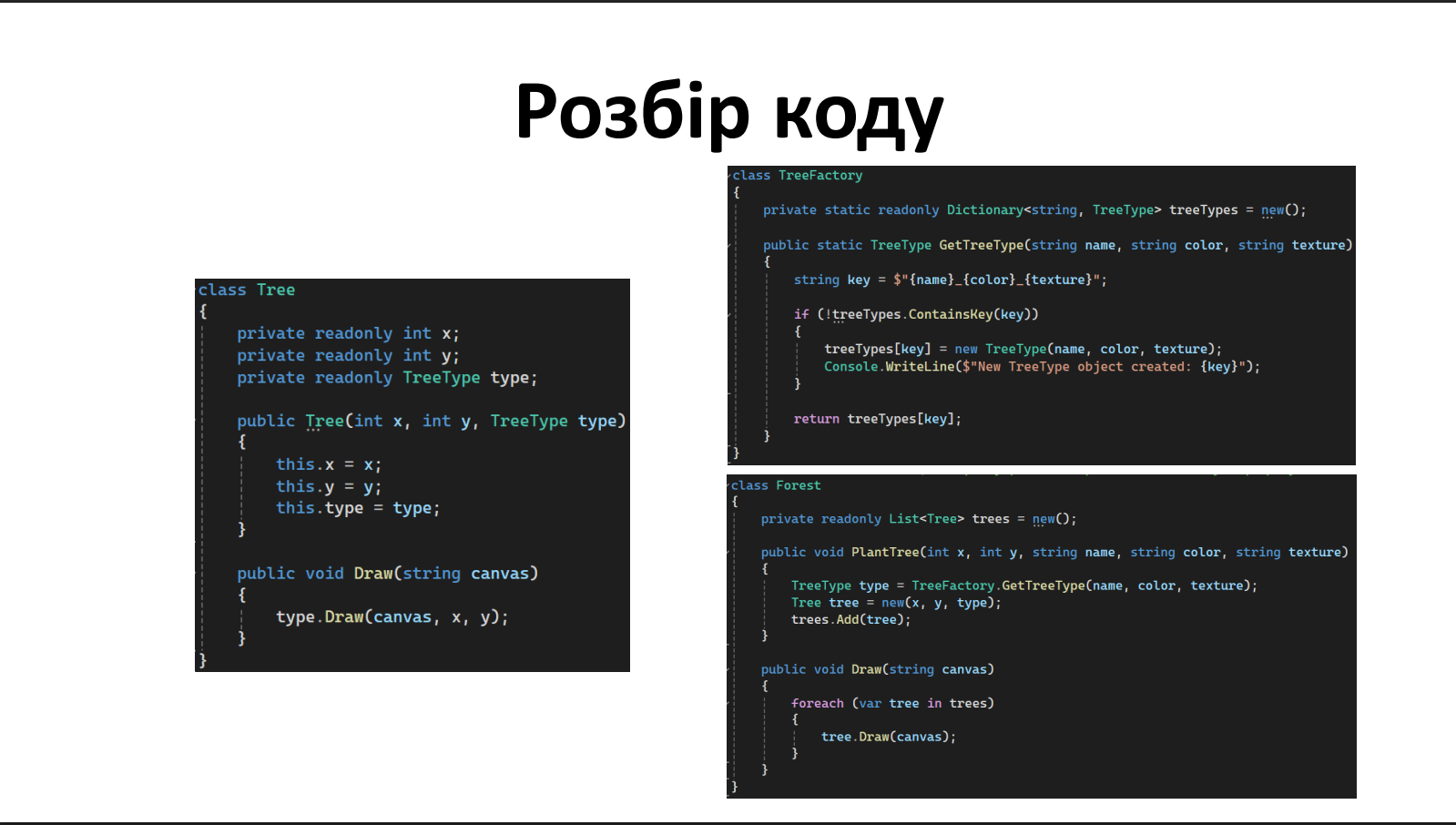


Рисунок Б.9 – Приклади програмного коду



Рисунок Б.10 – Коли треба застосовувати патерн



Рисунок Б.11 – Переваги та недоліки патерну



Рисунок Б.12 – Зв’язок з іншими патернами

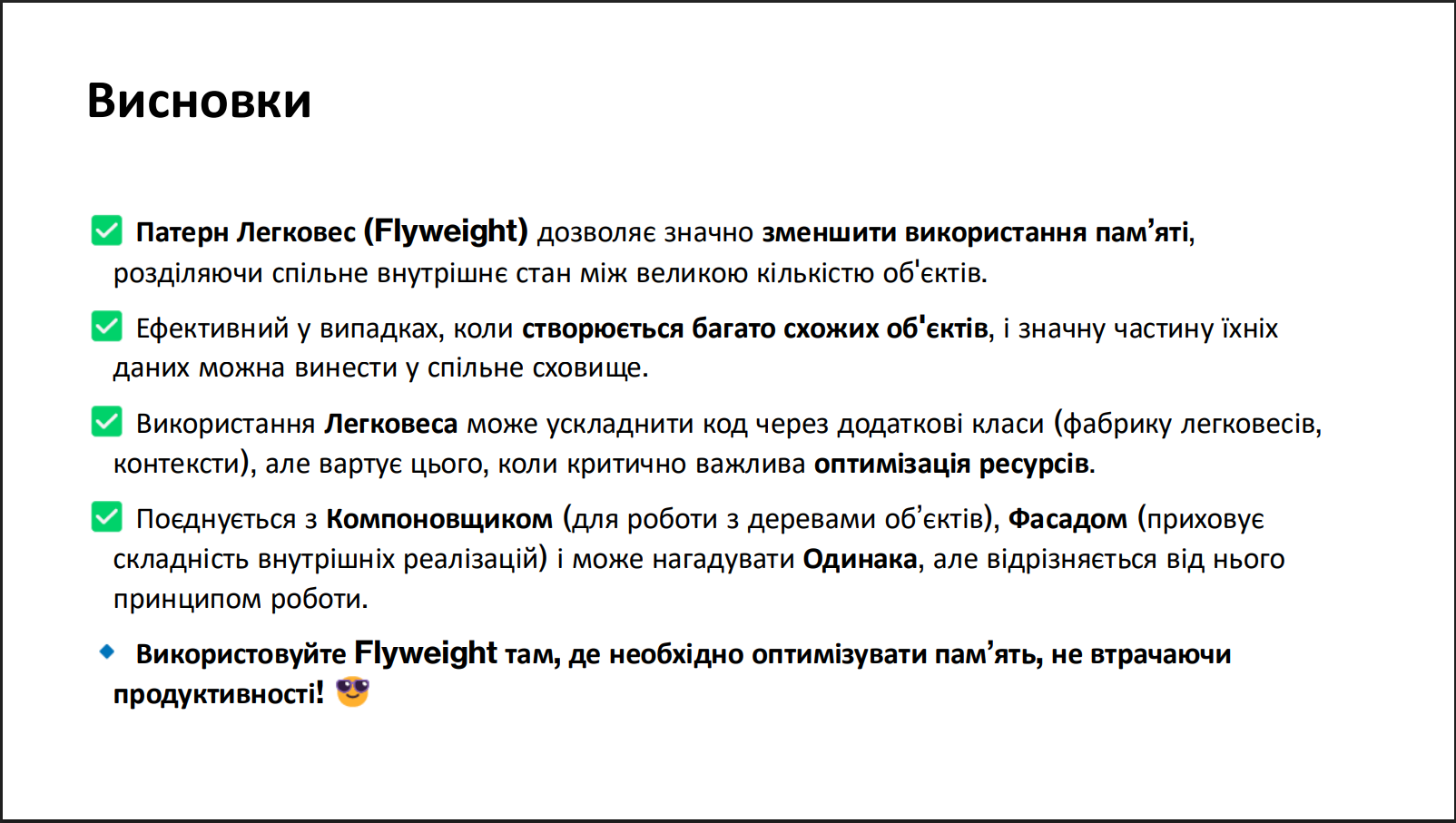


Рисунок Б.13 – Висновки

ДОДАТОК В

1 class TreeType

2 {

3 public string Name { get; }

4 public string Color { get; }

5 public string Texture { get; }

6

7 public TreeType(string name, string color, string texture)

8 {

9 Name = name;

10 Color = color;

11 Texture = texture;

12 }

13

14 public void Draw(string canvas, int x, int y)

15 {

16 Console.WriteLine($"Draw {Name} color {Color} with texture " +

17 $"{Texture} on {canvas} in coordinates ({x}, {y})");

18 }

19 }

20

21 class TreeFactory

22 {

23 private static readonly Dictionary<string, TreeType> treeTypes = new();

24

25 public static TreeType GetTreeType(string name, string color, string texture)

26 {

27 string key = $"{name}\_{color}\_{texture}";

28

29 if (!treeTypes.ContainsKey(key))

30 {

31 treeTypes[key] = new TreeType(name, color, texture);

32 Console.WriteLine($"New TreeType object created: {key}");

33 }

34

35 return treeTypes[key];

36 }

37 }

38

39 class Tree

40 {

41 private readonly int x;

42 private readonly int y;

43 private readonly TreeType type;

44

45 public Tree(int x, int y, TreeType type)

46 {

47 this.x = x;

48 this.y = y;

49 this.type = type;

50 }

51

52 public void Draw(string canvas)

53 {

54 type.Draw(canvas, x, y);

55 }

56 }

57

58 class Forest

59 {

60 private readonly List<Tree> trees = new();

61

62 public void PlantTree(int x, int y, string name, string color, string texture)

63 {

64 TreeType type = TreeFactory.GetTreeType(name, color, texture);

65 Tree tree = new(x, y, type);

66 trees.Add(tree);

67 }

68

69 public void Draw(string canvas)

70 {

71 foreach (var tree in trees)

72 {

73 tree.Draw(canvas);

74 }

75 }

76 }