Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №6

з дисципліни

«Основи розробки програмного забезпечення на платформі

Microsoft.NET»

«Шаблони проектування»

Виконала ІП-21 Голованьов Г.О.

Київ 2024

**Комп‘ютерний практикум № 6**

**Шаблони проектування**

**Мета:** ознайомитися з основними шаблонами проектування, навчитися застосовувати їх при проектуванні і розробці ПЗ.

**Варіант: 2**) Реалізувати алгоритм гри «хрестики-нулики». Реалізувати можливість «взяти назад хід»

# Обгрунтування обраного паттерну:

Патерн Command було обрано для реалізації гри «Хрестики-нулики» з можливістю "взяти назад хід" через його явні переваги у цьому контексті:

1. **Інкапсуляція дій:** Патерн Command дозволяє інкапсулювати кожну дію (хід) в окремому об'єкті команди. Це робить код більш модульним і спрощує додавання нових функціональностей.
2. **Зберігання історії дій:** Збереження послідовності команд дозволяє легко реалізувати функціональність скасування та повторного виконання дій.
3. **Розширюваність:** Команди легко розширювати і додавати нові типи дій без зміни існуючого коду.

**Опис архітектури проекту**

**Player:**

* char Symbol { get; private set; } — символ гравця ('X' або 'O').
* Конструктор Player(char symbol) — встановлює символ гравця.

**Board:**

* char[] cells — масив, що представляє стан дошки.
* Конструктор Board() — ініціалізує порожню дошку.
* Методи char Get(int position), void Set(int position, char symbol) — отримання та встановлення значення в клітинку.
* Метод bool CheckWin() — перевірка виграшних комбінацій.
* Метод bool IsFull() — перевірка на заповненість дошки.
* Метод void Display() — відображення стану дошки.

**ICommand:**

* Інтерфейс, що визначає методи void Execute() і void Undo().

**MoveCommand:**

* Інкапсулює хід гравця.
* Методи void Execute() і void Undo() — реалізація виконання та скасування ходу.

**GameHistory:**

* Stack<ICommand> commands — стек команд для зберігання історії.
* Методи void AddCommand(ICommand command), ICommand Undo() — додавання команди до історії та скасування останньої команди.

**TicTacToe:**

* Основний клас гри.
* Методи void Move(int position), void Undo(), void Display(), bool CheckWin(), bool IsFull(), char CurrentPlayerSymbol() — логіка гри та взаємодія з гравцем.

**UML-діаграмма:**

**A screenshot of a computer

Description automatically generated**

**Особливості реалізації обраного патерну:**

**Інкапсуляція команд**: Кожен хід гравця інкапсулюється у об'єкті MoveCommand, що забезпечує чітке розділення між логікою гри та діями.

**Зберігання історії**: Об'єкти команд зберігаються у стеку в GameHistory, що дозволяє легко скасовувати останні дії.

**Undo**: Реалізація методу Undo у класі TicTacToe дозволяє відновити стан гри до попереднього стану, використовуючи збережені команди.

# Текст програми

using System;

using System.Collections.Generic;

class Player

{

public char Symbol { get; private set; }

public Player(char symbol)

{

Symbol = symbol;

}

}

class Board

{

private char[] cells = new char[9];

public Board()

{

for (int i = 0; i < cells.Length; i++)

{

cells[i] = ' ';

}

}

public char Get(int position)

{

return cells[position];

}

public void Set(int position, char symbol)

{

cells[position] = symbol;

}

public bool CheckWin()

{

int[][] winCombinations = new int[][]

{

new int[] { 0, 1, 2 }, new int[] { 3, 4, 5 }, new int[] { 6, 7, 8 },

new int[] { 0, 3, 6 }, new int[] { 1, 4, 7 }, new int[] { 2, 5, 8 },

new int[] { 0, 4, 8 }, new int[] { 2, 4, 6 }

};

foreach (var combo in winCombinations)

{

if (cells[combo[0]] == cells[combo[1]] && cells[combo[1]] == cells[combo[2]] && cells[combo[0]] != ' ')

{

return true;

}

}

return false;

}

public bool IsFull()

{

foreach (var cell in cells)

{

if (cell == ' ')

{

return false;

}

}

return true;

}

public void Display()

{

for (int i = 0; i < cells.Length; i += 3)

{

Console.WriteLine($"{cells[i]} | {cells[i + 1]} | {cells[i + 2]}");

if (i < 6)

{

Console.WriteLine("---------");

}

}

}

}

interface ICommand

{

void Execute();

void Undo();

}

class MoveCommand : ICommand

{

private Board board;

private Player player;

private int position;

public MoveCommand(Board board, Player player, int position)

{

this.board = board;

this.player = player;

this.position = position;

}

public void Execute()

{

board.Set(position, player.Symbol);

}

public void Undo()

{

board.Set(position, ' ');

}

}

class GameHistory

{

private Stack<ICommand> commands = new Stack<ICommand>();

public void AddCommand(ICommand command)

{

commands.Push(command);

}

public ICommand Undo()

{

if (commands.Count > 0)

{

return commands.Pop();

}

return null;

}

}

class TicTacToe

{

private Board board;

private GameHistory history;

private Player currentPlayer;

public TicTacToe()

{

board = new Board();

history = new GameHistory();

currentPlayer = new Player('X');

}

private void SwitchPlayer()

{

currentPlayer = currentPlayer.Symbol == 'X' ? new Player('O') : new Player('X');

}

public void Move(int position)

{

if (board.Get(position) == ' ')

{

var command = new MoveCommand(board, currentPlayer, position);

command.Execute();

history.AddCommand(command);

if (!board.CheckWin() && !board.IsFull())

{

SwitchPlayer();

}

}

}

public void Undo()

{

ICommand command = history.Undo();

if (command != null)

{

command.Undo();

SwitchPlayer();

}

}

public void Display()

{

board.Display();

}

public bool CheckWin()

{

return board.CheckWin();

}

public bool IsFull()

{

return board.IsFull();

}

public char CurrentPlayerSymbol()

{

return currentPlayer.Symbol;

}

}

class Program

{

static void Main(string[] args)

{

TicTacToe game = new TicTacToe();

while (!game.CheckWin() && !game.IsFull())

{

game.Display();

Console.WriteLine($"Гравець {game.CurrentPlayerSymbol()}, введіть позицію (1-9):");

int position = (int.Parse(Console.ReadLine()))-1;

game.Move(position);

}

game.Display();

if (game.CheckWin())

{

Console.WriteLine($"Гравець {game.CurrentPlayerSymbol()} виграв!");

}

else

{

Console.WriteLine("Нічия!");

}

Console.WriteLine("\nВідмінити останній хід:");

game.Undo();

game.Display();

Console.WriteLine("\nВідмінити ще один хід:");

game.Undo();

game.Display();

}

}

# Скріншоти виконання

A black screen with white text

Description automatically generated