Text

Description automatically generated with medium confidence

wzorce projektowe

Sprawozdanie

Ćwiczenie 10 - Observer

Adam Zajler

Informatyka, programowanie

34\_Inf\_P\_NW\_6

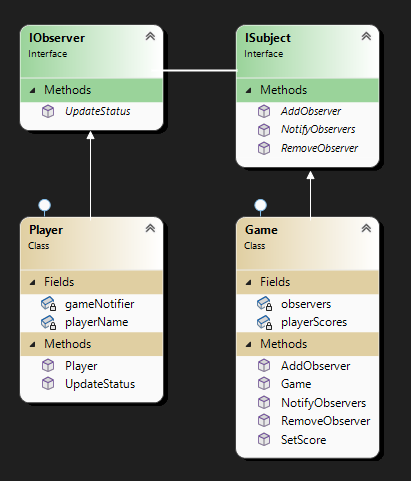
Rok akademicki 2023/24

# Cel

Celem tego sprawozdania jest zarejestrowanie postępu w zadaniu. W ramach tego został stworzony program w języku C#, który zarządza tabelą wyników dla różnych graczy. Całość opiera się na wzorcu projektowym Obserwator.

# Opis rozwiązania

Rozwiązaniem tego zadania jest program napisany w języku C#, prezentujący wykorzystanie wzorca projektowego Obserwator. W tym zaproponowanym rozwiązaniu została stworzona tabela wyników, do której przypisani są gracze, działający jednocześnie jako obserwatorzy. Po aktualizacji tabeli, wszyscy gracze są powiadamiani, a ich bieżące wyniki są wyświetlane.



# Implementacja

using System;

using System.Collections.Generic;

public interface IObserver

{

void UpdateStatus(Dictionary<string, int> scores);

}

public interface ISubject

{

void AddObserver(IObserver observer);

void RemoveObserver(IObserver observer);

void NotifyObservers();

}

public class Player : IObserver

{

private string playerName;

private ISubject gameNotifier;

public Player(string playerName, ISubject gameNotifier)

{

this.playerName = playerName;

this.gameNotifier = gameNotifier;

gameNotifier.AddObserver(this);

}

public void UpdateStatus(Dictionary<string, int> scores)

{

Console.WriteLine($"{playerName}'s updated scores:");

foreach (var score in scores)

{

Console.WriteLine($"Player: {score.Key}, Score: {score.Value}");

}

Console.WriteLine();

}

}

public class Game : ISubject

{

private List<IObserver> observers;

private Dictionary<string, int> playerScores;

public Game()

{

observers = new List<IObserver>();

playerScores = new Dictionary<string, int>();

}

public void AddObserver(IObserver observer)

{

observers.Add(observer);

}

public void RemoveObserver(IObserver observer)

{

observers.Remove(observer);

}

public void NotifyObservers()

{

foreach (var observer in observers)

{

observer.UpdateStatus(playerScores);

}

}

public void SetScore(string playerName, int score)

{

if (playerScores.ContainsKey(playerName))

{

playerScores[playerName] = score;

}

else

{

playerScores.Add(playerName, score);

}

NotifyObservers();

}

}

public class Program

{

public static void Main(string[] args)

{

Game game = new Game();

Player player1 = new Player("Adam", game);

Player player2 = new Player("John", game);

Player player3 = new Player("Michael", game);

game.SetScore("Adam", 10);

game.SetScore("John", 15);

game.SetScore("Michael", 8);

game.SetScore("Adam", 20);

Console.ReadKey();

}

}

# Podsumowanie

# W realizacji tego zadania zastosowano wzorzec projektowy Obserwator. Ten wzorzec projektowy jest doskonały do tego rodzaju zastosowań, gdyż pozwala na błyskawiczną aktualizację i prezentację danych zaraz po ich modyfikacji.

# Lista załączników

[AdamZajler/wzorce-projektowe (github.com)](https://github.com/AdamZajler/wzorce-projektowe/tree/main)