

WZORCE PROJEKTOWE

SPRAWOZDANIE

ZADANIE ADAPTER KWADRAT

Patryk Figas
Informatyka, programowanie
Grupa 34_Inf_P_NW_6

1. Cel

Celem tego dokumentu jest przedstawienie rozwiązania zadania polegającego na dostosowaniu istniejącej **klasy Square** do interfejsu oczekiwanego przez funkcję operującą na obiektach implementujących **metodę getArea()**.

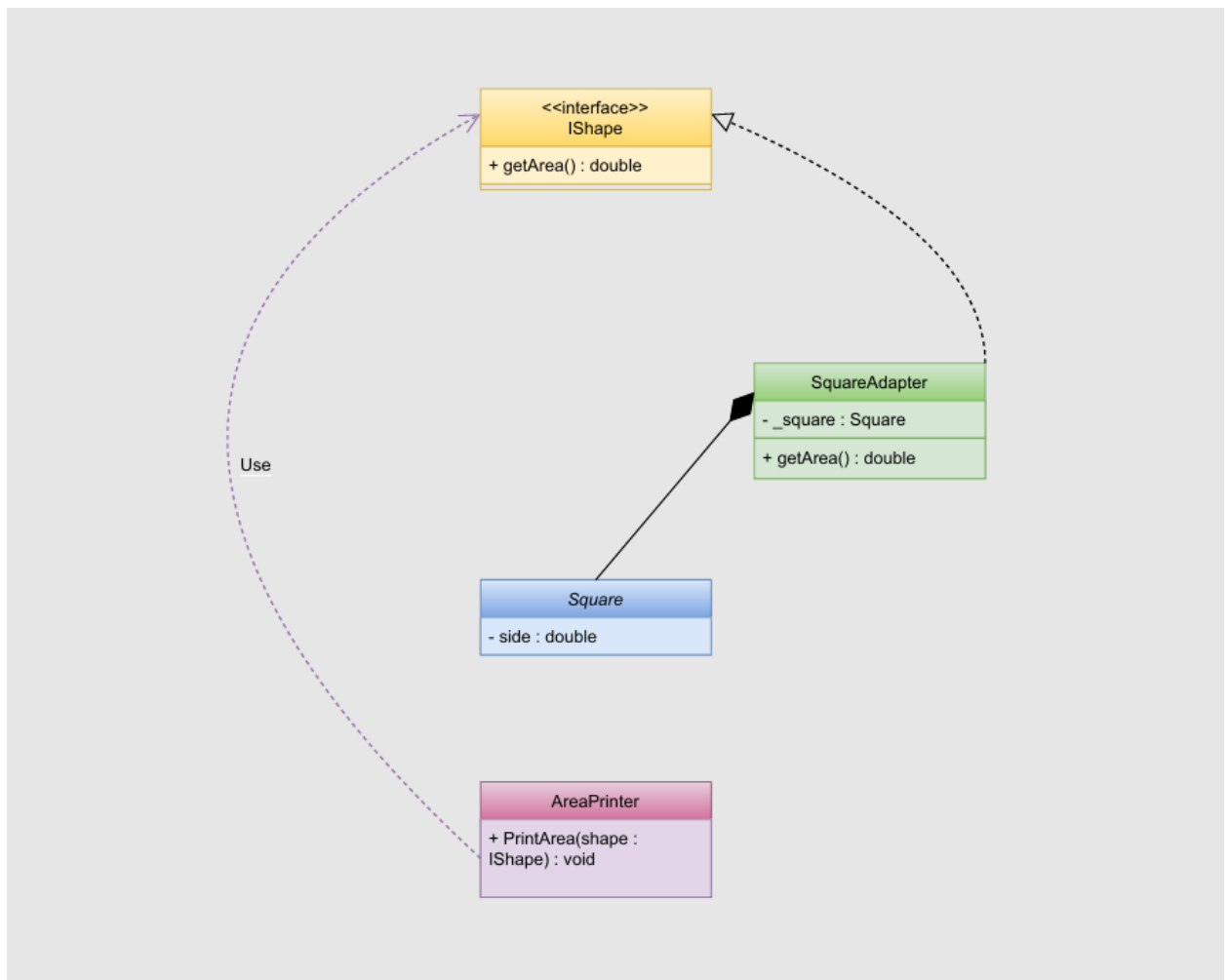
W ramach ćwiczenia zaprojektowano klasę za pomocą „pseudokodu”, diagramu UML i implementacji klasy do programu oraz użycie jej w programie Main.

W rozwiązaniu zaprojektowano i zaimplementowano wzorzec projektowy **Adapter**, umożliwiający przekształcenie klasy bez modyfikowania jej kodu źródłowego.

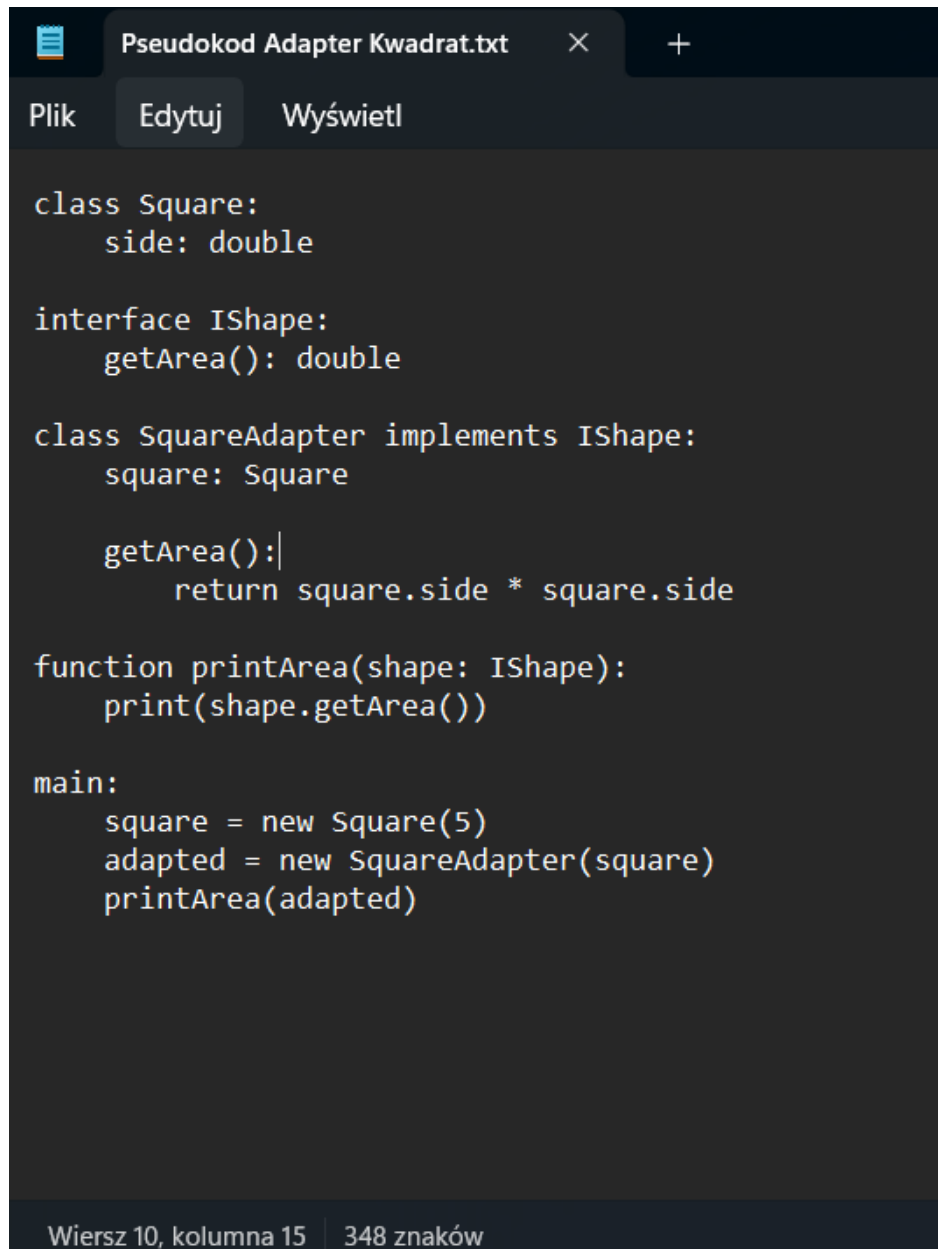
2. Opis rozwiązania

W zadaniu wykorzystano **wzorzec Adapter**, aby umożliwić wykorzystanie istniejącej klasy **Square**, która nie implementuje **interfejsu IShape**, gdzie wymagane jest wywoływanie metody `getArea()`. **Adapter** opakowuje klasę **Square** i udostępnia jej funkcjonalność zgodnie z wymaganym interfejsem, bez ingerencji w oryginalną klasę.

- Diagram klas programu



- Pseudokod



The image shows a screenshot of a code editor window titled "Pseudokod Adapter Kwadrat.txt". The editor has a menu bar with "Plik", "Edytuj", and "Wyświetl". The code is written in a light blue font on a dark background. It defines a class "Square" with a "side" attribute, an interface "IShape" with a "getArea()" method, and a class "SquareAdapter" that implements "IShape" by delegating the "getArea()" call to a "Square" object. A "main" function demonstrates the usage by creating a "Square" object, wrapping it in a "SquareAdapter", and calling "printArea()". The status bar at the bottom indicates "Wiersz 10, kolumna 15" and "348 znaków".

```
class Square:
    side: double

interface IShape:
    getArea(): double

class SquareAdapter implements IShape:
    square: Square

    getArea():
        return square.side * square.side

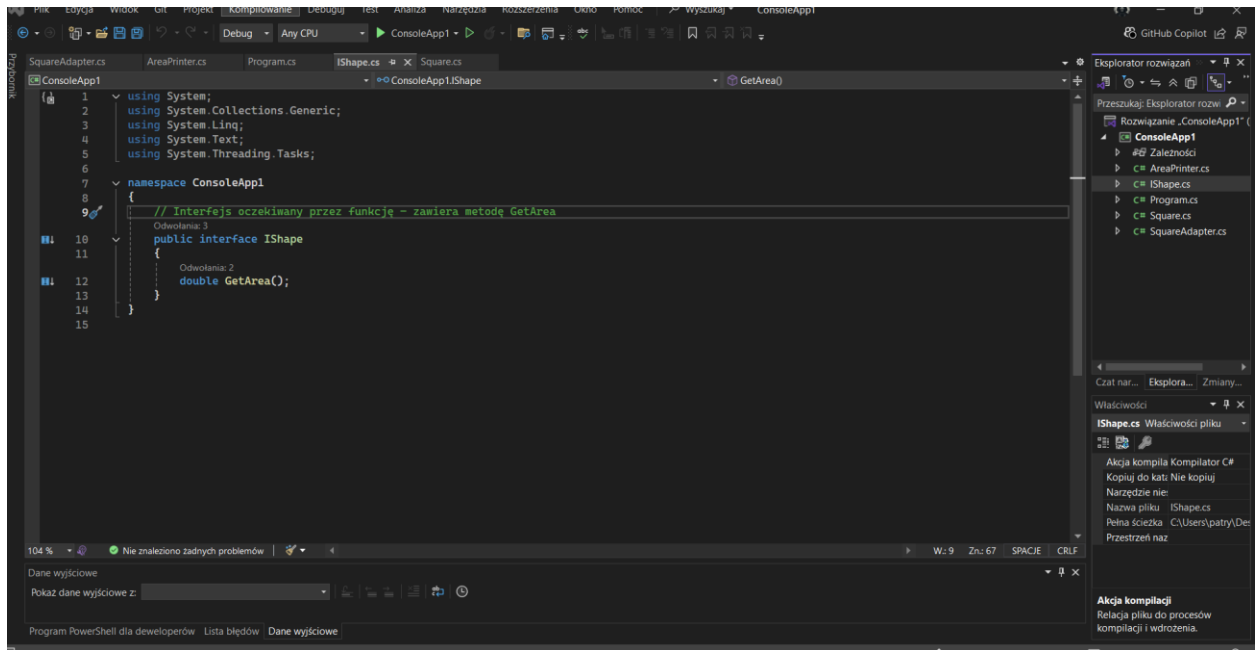
function printArea(shape: IShape):
    print(shape.getArea())

main:
    square = new Square(5)
    adapted = new SquareAdapter(square)
    printArea(adapted)
```

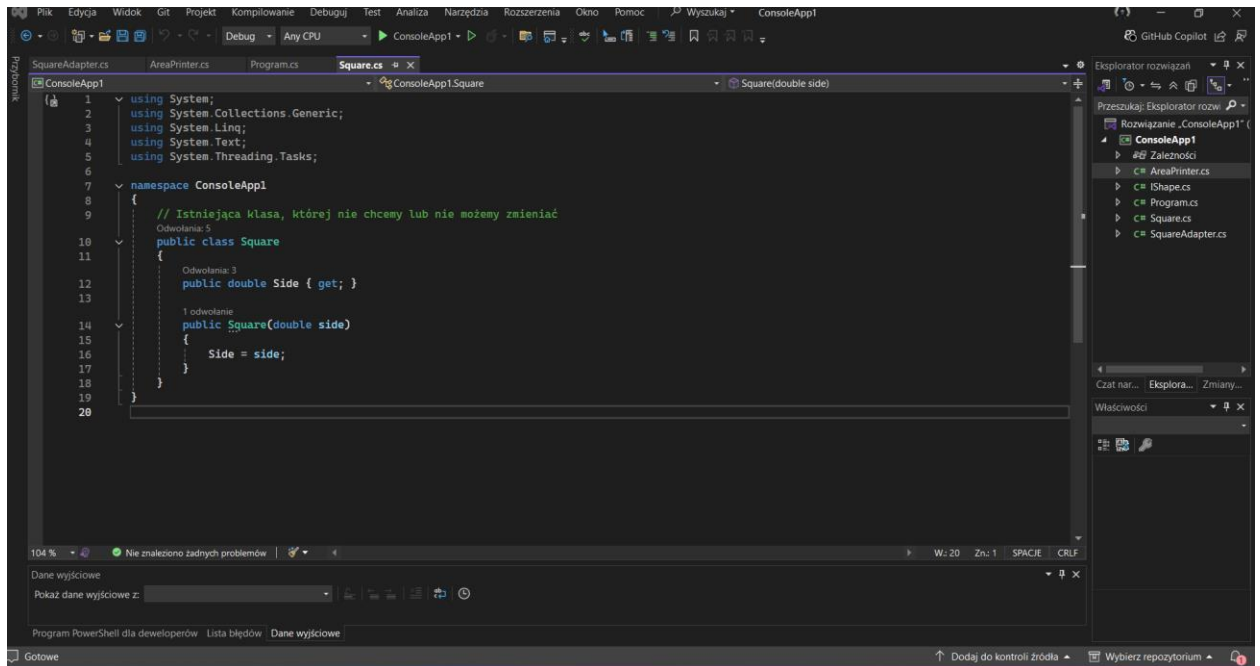
Wiersz 10, kolumna 15 | 348 znaków

3. Implementacja

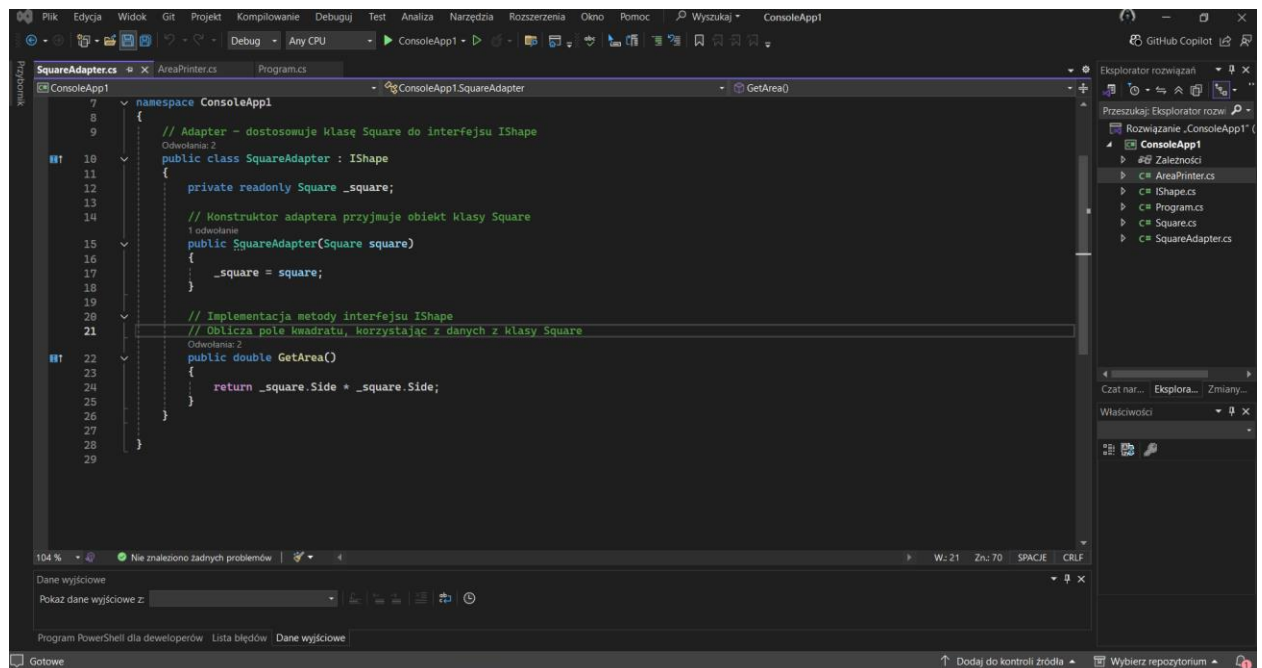
- Kod interfejsu **IShape.cs**



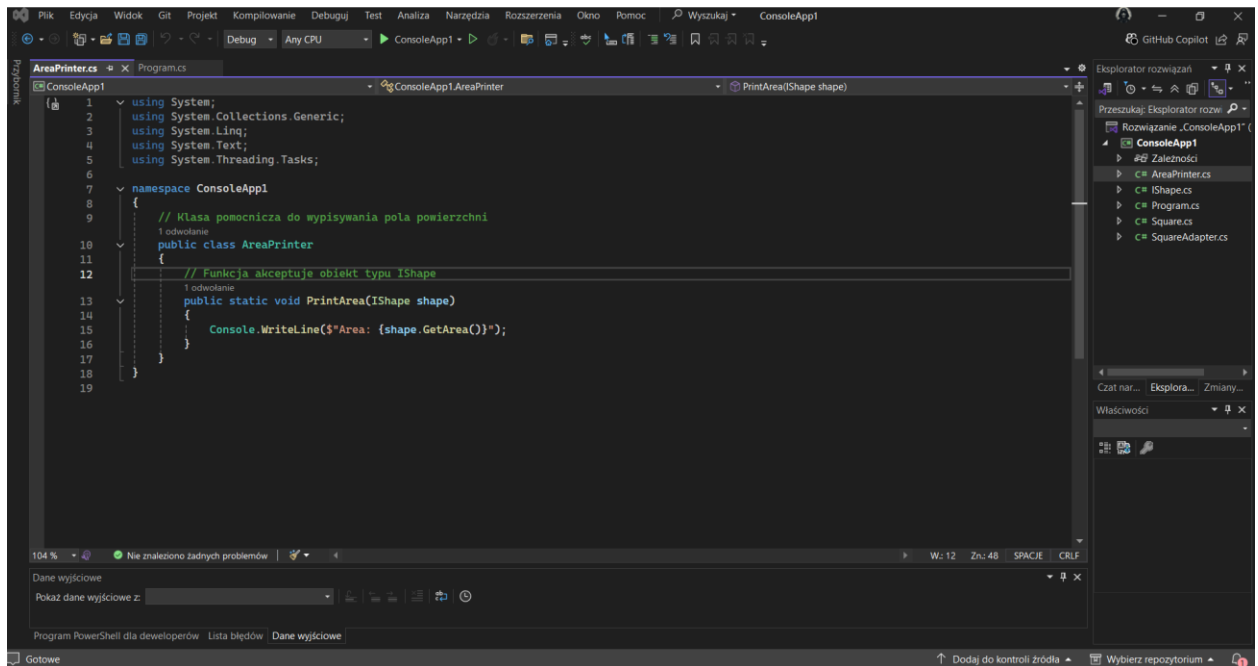
- Kod klasy **Square.cs**



- Kod klasy **SquareAdapter.cs**



- Kod klasy **AreaPrinter.cs**

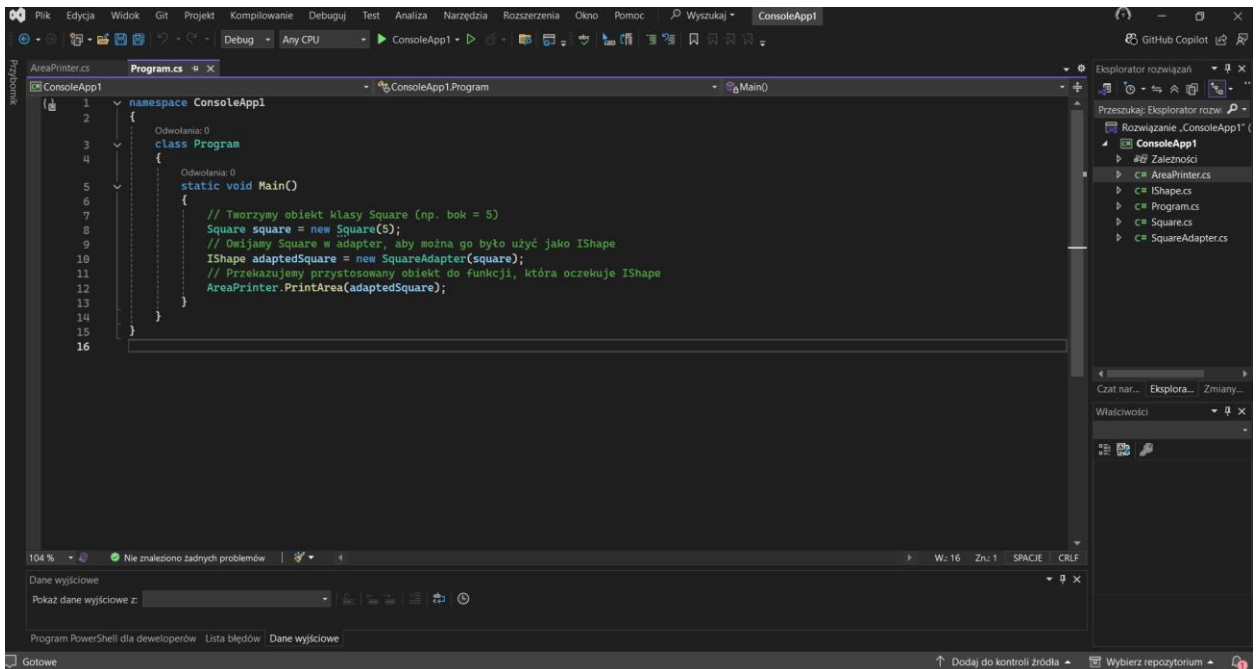


```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp1
8  {
9      // Klasa pomocnicza do wypisywania pola powierzchni
10     1 odwołanie
11     public class AreaPrinter
12     {
13         // Funkcja akceptuje obiekt typu IShape
14         1 odwołanie
15         public static void PrintArea(IShape shape)
16         {
17             Console.WriteLine($"Area: {shape.GetArea()}");
18         }
19     }

```

- Kod klasy **Program.cs**



```

1  namespace ConsoleApp1
2  {
3      Odwołania: 0
4      class Program
5      {
6          Odwołania: 0
7          static void Main()
8          {
9              // Tworzymy obiekt klasy Square (np. bok = 5)
10             Square square = new Square(5);
11             // Dajemy Square w adapter, aby można go było użyć jako IShape
12             IShape adaptedSquare = new SquareAdapter(square);
13             // Przekazujemy przystosowany obiekt do funkcji, która oczekuje IShape
14             AreaPrinter.PrintArea(adaptedSquare);
15         }
16     }

```


4. Podsumowanie

W zadaniu zastosowano wzorzec projektowy **Adapter**, ponieważ był to najbardziej odpowiedni sposób na dostosowanie istniejącej **klasy Square** do wymagań funkcji oczekującej **interfejsu IShape**. Adapter pozwolił na użycie klasy bez jej modyfikowania, co jest zgodne z zasadami **otwarte/zamknięte** (SOLID).

Funkcja PrintArea mogła operować na obiekcie **klasy Square**, mimo że klasa ta pierwotnie nie implementowała wymaganej metody.

Alternatywne wzorce, takie jak **Dekorator** lub **Strategia**, nie byłyby tutaj odpowiednie – celem nie była zmiana zachowania klasy, lecz dostosowanie jej interfejsu. Dlatego **Adapter** jest najwłaściwszym wyborem w tym przypadku.

Lista załączników

Repozytorium GITHUB z projektem:

<https://github.com/PatrykFigas/Wzorce-projektowe.git>