Text

Description automatically generated with medium confidence

wzorce projektowe

Sprawozdanie

Ćwiczenie 8 - Chain of Responsibility

Adam Zajler

Informatyka, programowanie

34\_Inf\_P\_NW\_6

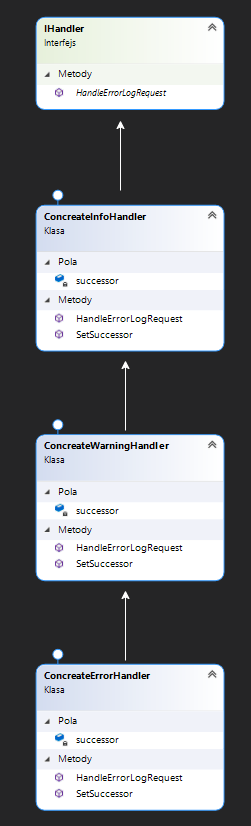
Rok akademicki 2023/24

# Cel

Celem sprawozdanie jest udokumentowanie wykonania ćwiczenia. W ramach zadania został utworzony program w języku c# który służy do obsługi systemu logowania błędów opartego na wzorcu projektowym chain of responsibility.

# Opis rozwiązania

Jako rozwiązanie zadania został napisany kod w języku c#. Używa on wzorca projektowego chain of responsibility na przykładzie systemu obsługi błędów. Użytkownik może podać typ błędu, a następnie system przydziela go do odpowiedniego handlera.



# Implementacja

using System;

namespace Ćwiczenie\_8\_\_\_Chain\_of\_Responsibility

{

interface IHandler

{

void HandleErrorLogRequest(string code);

}

class ConcreateInfoHandler : IHandler

{

private IHandler successor;

public void SetSuccessor(IHandler successor)

{

this.successor = successor;

}

public void HandleErrorLogRequest(string code)

{

if (code == "INFO")

{

Console.WriteLine("Obsługa błędu typu INFO\n");

}

else

{

successor.HandleErrorLogRequest(code);

}

}

}

class ConcreateWarningHandler : IHandler

{

private IHandler successor;

public void SetSuccessor(IHandler successor)

{

this.successor = successor;

}

public void HandleErrorLogRequest(string code)

{

if (code == "WARNING")

{

Console.WriteLine("Obsługa błędu typu WARNING\n");

}

else

{

successor.HandleErrorLogRequest(code);

}

}

}

class ConcreateErrorHandler : IHandler

{

private IHandler successor;

public void SetSuccessor(IHandler successor)

{

this.successor = successor;

}

public void HandleErrorLogRequest(string code)

{

Console.WriteLine("Obsługa błędu typu INFO\n");

}

}

class Program

{

static void Main(string[] args)

{

var InfoHandler = new ConcreateInfoHandler();

var WarningHandler = new ConcreateWarningHandler();

var ErrorHandler = new ConcreateErrorHandler();

InfoHandler.SetSuccessor(WarningHandler);

WarningHandler.SetSuccessor(ErrorHandler);

var errors = new string[] { "INFO", "WARNING", "ERROR" };

foreach (string error in errors)

{

Console.WriteLine("TYP BŁĘDU: {0}", error);

InfoHandler.HandleErrorLogRequest(error);

}

Console.ReadKey();

}

}

}

# Podsumowanie

# Zastosowano wzorzec łańcucha odpowiedzialności, aby umożliwić obsługę różnych typów błędów. Każdy handler w łańcuchu jest odpowiedzialny za obsługę określonego typu błędu. Jeśli handler nie jest w stanie obsłużyć błędu, przekazuje je dalej do następnego handlera w łańcuchu.

# Lista załączników

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