Міністерство освіти і науки України

Національний лісотехнічний університет України

Кафедра інформаційних технологій

**Звіт до лабораторної роботи**

з навчальної дисципліни

**«Методи та засоби ООАП»**

**Виконав:**

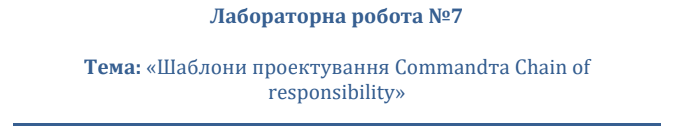
студент групи КНС-21

Марущак С. І.

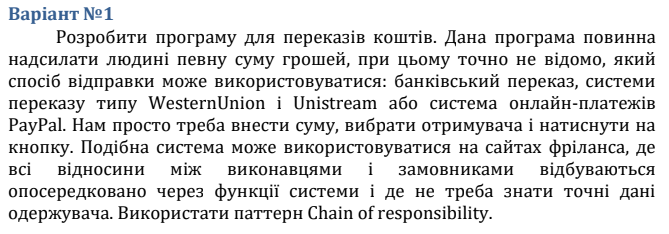
**Перевірив:**

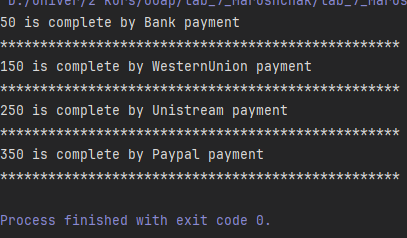
Волинець Є.О.

Львів – 2024

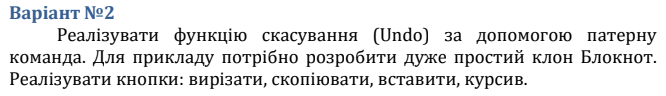


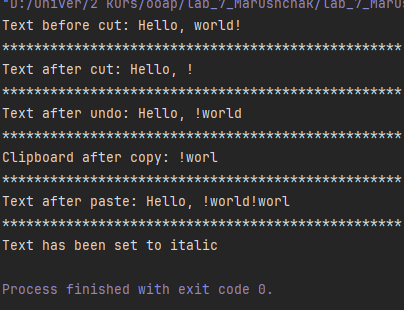






using System;  
  
namespace lab\_7\_ChainOfResponsibility  
{  
 abstract class PaymentHandler // common handler  
 {  
 protected PaymentHandler \_nextHandler;  
  
 public void SetNextHandler(PaymentHandler nextHandler)  
 {  
 \_nextHandler = nextHandler;  
 }  
  
 public abstract void CompletePayment(double amount);  
 }  
   
 class BankPaymentHandler : PaymentHandler // concreate handler  
 {  
 public override void CompletePayment(double amount)  
 {  
 if (amount <= 100)  
 {  
 Console.WriteLine($"{amount} is complete by Bank payment");  
 }  
 else if (\_nextHandler != null)  
 {  
 \_nextHandler.CompletePayment(amount);  
 }  
 }  
 }  
   
 class WesternUnionPaymentHandler : PaymentHandler  
 {  
 public override void CompletePayment(double amount)  
 {  
 if (amount > 100 && amount <= 200)  
 {  
 Console.WriteLine($"{amount} is complete by WesternUnion payment");  
 }  
 else if (\_nextHandler != null)  
 {  
 \_nextHandler.CompletePayment(amount);  
 }  
 }  
 }  
   
 class UnistreamPaymentHandler : PaymentHandler  
 {  
 public override void CompletePayment(double amount)  
 {  
 if (amount > 200 && amount <= 300)  
 {  
 Console.WriteLine($"{amount} is complete by Unistream payment");  
 }  
 else if (\_nextHandler != null)  
 {  
 \_nextHandler.CompletePayment(amount);  
 }  
 }  
 }  
   
 class PaypalPaymentHandler : PaymentHandler  
 {  
 public override void CompletePayment(double amount)  
 {  
 if (amount > 300 && amount <= 400)  
 {  
 Console.WriteLine($"{amount} is complete by Paypal payment");  
 }  
 else if (\_nextHandler != null)  
 {  
 \_nextHandler.CompletePayment(amount);  
 }  
 }  
 }  
  
 class Client // user  
 {  
 private PaymentHandler \_handlerChain;  
  
 public Client()  
 {  
 \_handlerChain = new BankPaymentHandler();  
 PaymentHandler westernUnionHandler = new WesternUnionPaymentHandler();  
 PaymentHandler unistreamHandler = new UnistreamPaymentHandler();  
 PaymentHandler payPalHandler = new PaypalPaymentHandler();  
   
 \_handlerChain.SetNextHandler(westernUnionHandler);  
 westernUnionHandler.SetNextHandler(unistreamHandler);  
 unistreamHandler.SetNextHandler(payPalHandler);  
 }  
  
 public void Pay(double amount)  
 {  
 \_handlerChain.CompletePayment(amount);  
 }  
 }  
  
 internal class Program  
 {  
 public static void Main(string[] args)  
 {  
 Client client = new Client();  
   
 client.Pay(50);  
 Console.WriteLine(new string('\*',50));  
   
 client.Pay(150);  
 Console.WriteLine(new string('\*',50));  
   
 client.Pay(250);  
 Console.WriteLine(new string('\*',50));  
   
 client.Pay(350);  
 Console.WriteLine(new string('\*',50));  
   
 client.Pay(450);  
  
 }  
 }  
}





using System;  
using System.Collections.Generic;  
  
namespace lab\_7\_Command  
{  
 class Note // receiver  
 {  
 public string Text { get; private set; }  
 public string Clipborad { get; private set; }  
  
 public void CutText(int startIndex, int length)  
 {  
 Text = Text.Remove(startIndex, length);  
 }  
  
 public void CopyText(int startIndex, int length)  
 {  
 Clipborad = Text.Substring(startIndex, length);  
 }  
  
 public void PasteText(string text)  
 {  
 Text += text;  
 }  
  
 public void SetItalicFont()  
 {  
 Console.WriteLine("Text has been set to italic ");  
 }  
   
 }  
   
 abstract class Command // common command's class  
 {  
 protected Note \_note;  
 protected string \_backup;  
 protected int \_startIndex;  
 protected int \_length;  
   
 public Command(Note note, int startIndex, int length)  
 {  
 \_note = note;  
 \_startIndex = startIndex;  
 \_length = length;  
 }  
 public abstract void Execute();  
 public abstract void Undo();  
 }  
   
 class CutCommand: Command  
 {  
 public CutCommand(Note note, int startIndex, int length) : base(note, startIndex, length)  
 {  
   
 }  
 public override void Execute()  
 {  
 \_backup = \_note.Text.Substring(\_startIndex, \_length);  
 \_note.CutText(\_startIndex,\_length);  
 }  
  
 public override void Undo()  
 {  
 \_note.PasteText(\_backup);  
 }  
 }  
   
 class CopyCommand : Command  
 {  
 public CopyCommand(Note note, int startIndex, int length) : base(note, startIndex, length)  
 {  
   
 }  
   
 public override void Execute()  
 {  
 \_note.CopyText(\_startIndex, \_length);  
 }  
  
 public override void Undo()  
 {  
 throw new NotImplementedException();  
 }  
   
 }  
   
 class PasteCommand : Command  
 {  
 private string \_textToPaste;  
   
 public PasteCommand(Note note, string textToPaste, int startIndex) : base(note, startIndex, textToPaste.Length)  
 {  
 \_textToPaste = textToPaste;  
 }  
   
 public override void Execute()  
 {  
 \_note.PasteText(\_textToPaste);  
 }  
  
 public override void Undo()  
 {  
 \_note.CutText(\_startIndex, \_textToPaste.Length);  
 }  
 }  
   
 class ItalicCommand : Command  
 {  
 public ItalicCommand(Note note, int startIndex, int length) : base(note, 0, 0)  
 {  
   
 }  
  
 public override void Execute()  
 {  
 \_backup = \_note.Text;  
 \_note.SetItalicFont();  
 }  
  
 public override void Undo()  
 {  
 \_note.PasteText(\_backup);  
 }  
 }  
  
 class Invoker  
 {  
 private Stack<Command> \_commands = new Stack<Command>();  
  
 public void StoreAndExecute(Command command)  
 {  
 command.Execute();  
 \_commands.Push(command);  
 }  
  
 public void UndoLast()  
 {  
 if (\_commands.Count > 0)  
 {  
 var lastCommand = \_commands.Pop();  
 lastCommand.Undo();  
 }  
 }  
   
 }  
  
  
 internal class Program  
 {  
 public static void Main(string[] args)  
 {  
 Note note = new Note();  
 Invoker invoker = new Invoker();  
   
 note.PasteText("Hello, world!");  
 Console.WriteLine("Text before cut: " + note.Text);  
 Console.WriteLine(new string('\*',50));  
   
 invoker.StoreAndExecute(new CutCommand(note, 7, 5));  
 Console.WriteLine("Text after cut: " + note.Text);  
 Console.WriteLine(new string('\*',50));  
   
 invoker.UndoLast();  
 Console.WriteLine("Text after undo: " + note.Text);  
 Console.WriteLine(new string('\*',50));  
  
 invoker.StoreAndExecute(new CopyCommand(note, 7, 5));  
 Console.WriteLine("Clipboard after copy: " + note.Clipborad);  
 Console.WriteLine(new string('\*',50));  
  
 invoker.StoreAndExecute(new PasteCommand(note, note.Clipborad, note.Text.Length));  
 Console.WriteLine("Text after paste: " + note.Text);  
 Console.WriteLine(new string('\*',50));  
  
 invoker.StoreAndExecute(new ItalicCommand(note, 0, 0));  
 }  
 }  
}

**Висновок:** на даній лабораторній роботі використовував шаблон проектування Command, Chain of responsibility.