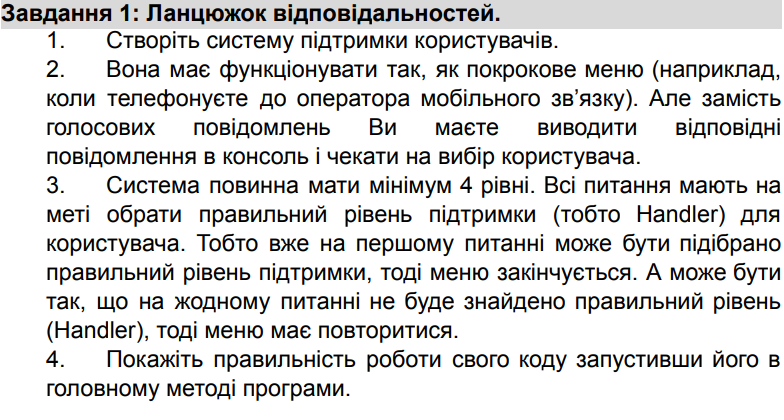
**Лабораторна робота № 6**

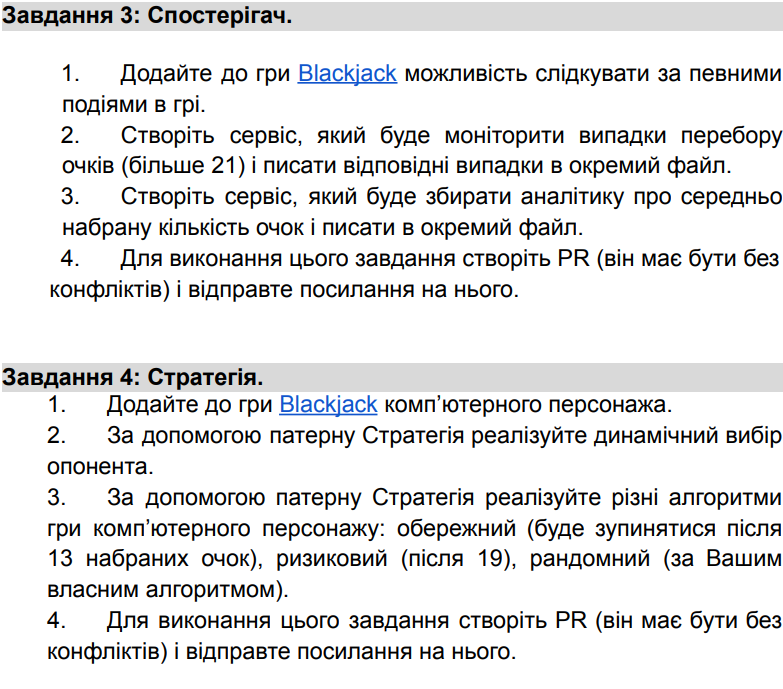
**ПОВЕДІНКОВІ ШАБЛОНИ**

**Мета:** навчитися реалізовувати структурні шоблони проєктування Ланцюжок відповідальностей, Посередник, Спостерігач, Стратегія

**Хід роботи:**

****

# 



**Завдання 1:** Ланцюжок відповідальностей.

Створення системи підтримки користувачів

Лістинг коду:

using System;

namespace BehavioralPatterns.ChainOfResponsibility.Handlers

{

public class AccountInfoHandler : ISupportRequestHandler

{

private RequestDelegate \_next;

public AccountInfoHandler(RequestDelegate next)

{

\_next = next;

}

public void Handle(SupportContext context)

{

if(context.Request!=null&&int.Parse(context.Request)==1)

{

System.Console.WriteLine("Name: Mike\nAge: 15\nContractId: 1231aseqwe\nAddress: str.Uk newasc");

}

\_next(context);

}

}

}

namespace BehavioralPatterns.ChainOfResponsibility.Handlers

{

public class AccountMoneyHandler : ISupportRequestHandler

{

private RequestDelegate \_next;

public AccountMoneyHandler(RequestDelegate next)

{

\_next = next;

}

public void Handle(SupportContext context)

{

if(context.RequestToInt()==4)

{

System.Console.Write("Enter sum:");

var res = Console.ReadLine();

System.Console.WriteLine($"your account is replenished by {res}");

}

\_next(context);

}

}

}

namespace BehavioralPatterns.ChainOfResponsibility.Handlers

{

public class ChangePasswordHandler : ISupportRequestHandler

{

private RequestDelegate \_next;

public ChangePasswordHandler(RequestDelegate next)

{

\_next = next;

}

public void Handle(SupportContext context)

{

if(context.RequestToInt()==2)

{

System.Console.Write("Enter you previous password: ");

string? res = Console.ReadLine();

if(IsPasswordMatch(res))

{

System.Console.WriteLine("Enter new password");

string newPass = Console.ReadLine();

System.Console.WriteLine("Password change sent for processing");

}else

{

System.Console.WriteLine("Password is not match, you can try to change password again");

AppState.RestartService();

return;

}

}

\_next(context);

}

private bool IsPasswordMatch(string password)

{

if(password!="")

{

return true;

}

return false;

}

}

}

namespace BehavioralPatterns.ChainOfResponsibility

{

public class ErrorrHandler : ISupportRequestHandler

{

private RequestDelegate \_next;

private int \_userInput;

public ErrorrHandler(RequestDelegate next)

{

\_next = next;

}

public void Handle(SupportContext context)

{

if(!int.TryParse(context.Request,out \_)||context.Request==null)

{

System.Console.WriteLine("Incorrect input !");

AppState.RestartService();

return;

}

\_userInput = int.Parse(context.Request);

if(!context.AvailableSerivices.Contains(\_userInput))

{

System.Console.WriteLine("This service is not available");

AppState.RestartService();

return;

}

\_next(context);

}

}

}

namespace BehavioralPatterns.ChainOfResponsibility.Handlers

{

public class StateHandler:ISupportRequestHandler

{

private RequestDelegate \_next;

public StateHandler(RequestDelegate next)

{

\_next = next;

}

public void Handle(SupportContext context)

{

if(context.RequestToInt()==5)

{

AppState.RestartService();

return;

}

if(context.RequestToInt()==6)

{

AppState.StopService();

return;

}

\_next(context);

}

}

}

# Результат виконання програми:

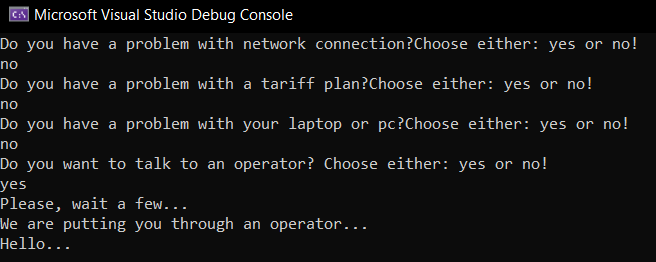


Рис.6.1. Реалізація завдання.

**Завдання 2:** Посередник.

Лістинг коду**:**

using System;

using BehavioralPatterns.Mediator.CustomEventArgs;

namespace BehavioralPatterns.Mediator.Classes

{

class Aircraft

{

public string Name;

private CommandCentre \_centre;

public bool IsTakingOff { get; set; }

private int \_currentRunway;

public Aircraft(string name, CommandCentre centre)

{

this.Name = name;

\_centre = centre;

}

public void LandRequest(int runwayId)

{

\_centre.OnAircraftLanded += OnLand;

\_centre.Notify(this, new AirEventArgs()

{

RunwayId = runwayId,

AirState = AirState.Land

});

\_centre.OnAircraftLanded -= OnLand;

\_currentRunway = runwayId;

}

private void OnLand(RunwayStateEventArgs e)

{

Console.WriteLine($"Aircraft {this.Name} is landing.");

Console.WriteLine($"Checking runway.");

if (e.LandingIsAllowed)

{

Console.WriteLine($"Aircraft {this.Name} has landed.");

}

else

{

Console.WriteLine($"Could not land, the runway is busy.");

}

}

public void TakeOff()

{

\_centre.OnAircraftTakeOff += OnTakeOff;

\_centre.Notify(this, new AirEventArgs() {AirState = AirState.TakeOff, RunwayId=\_currentRunway});

\_centre.OnAircraftTakeOff -= OnTakeOff;

}

private void OnTakeOff()

{

Console.WriteLine($"Aircraft {this.Name} is taking off.");

Console.WriteLine($"Aircraft {this.Name} has took off.");

}

}

}

namespace BehavioralPatterns.Mediator.Classes

{

public delegate void LandingEventHandler(RunwayStateEventArgs e);

public delegate void TakeOffEventHandler();

internal class CommandCentre

{

private List<Runway> \_runways = new List<Runway>();

private List<Aircraft> \_aircrafts = new List<Aircraft>();

public event LandingEventHandler OnAircraftLanded;

public event TakeOffEventHandler OnAircraftTakeOff;

public void SetUp(Runway[] runways, Aircraft[] aircrafts)

{

\_aircrafts.AddRange(aircrafts);

\_runways.AddRange(runways);

}

public void Notify(object sender, AirEventArgs e)

{

var runway = GetRunwayOrExcept(e.RunwayId);

if (sender is Aircraft)

{

if (e.AirState == AirState.Land)

{

TryLand(runway);

}

if (e.AirState == AirState.TakeOff)

{

TakeOff(runway);

}

}

if (sender is Runway)

{

runway.IsBusyWithAircraft = true;

runway.HighLightRed();

}

}

private Runway GetRunwayOrExcept(int id)

{

var runway = \_runways.FirstOrDefault(r => r.Id == id);

if (runway == null)

{

throw new Exception("exp!");

}

return runway;

}

private void TryLand(Runway runway)

{

if (runway.IsBusyWithAircraft)

{

OnAircraftLanded.Invoke(new RunwayStateEventArgs() { LandingIsAllowed = false });

}

else

{

OnAircraftLanded.Invoke(new RunwayStateEventArgs() { LandingIsAllowed = true });

runway.HighLightRed();

runway.IsBusyWithAircraft = true;

}

}

private void TakeOff(Runway runway)

{

OnAircraftTakeOff.Invoke();

runway.HighLightGreen();

runway.IsBusyWithAircraft = false;

}

}

}

namespace BehavioralPatterns.Mediator.Classes

{

internal class Runway

{

private CommandCentre \_centre;

public int Id {get;set;}

public bool IsBusyWithAircraft{get;set;}

public Runway(CommandCentre centre)

{

\_centre = centre;

}

public void TechnicalWork()

{

\_centre.Notify(this,new AirEventArgs(){RunwayId=Id});

}

public void HighLightRed()

{

Console.WriteLine($"Runway {this.Id} is busy now");

}

public void HighLightGreen()

{

Console.WriteLine($"Runway {this.Id} is free!");

}

}

}

# Результат виконання програми:

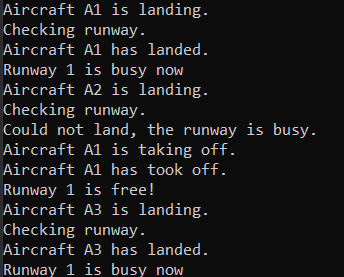


Рис.6.2. Реалізація завдання.

**Завдання 3:** Спостерігач.

Лістинг коду:

using blackjack.Game.EventHandlers;

using blackjack.Game.Handlers;

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace blackjack.Game.Finshers

{

public class FileStatiscticLogger

{

public string FileNameForOver21 { get; private set; } = "..\\..\\..\\Over21.txt";

public string FileNameForStatistics { get; private set; } = "..\\..\\..\\AvgStat.txt";

public FileStatiscticLogger()

{ }

public void WriteOver21Statistics(object? sender, Over21HandlerLog e)

{

this.Write(FileNameForOver21, e.ToString());

}

public void WriteAvgStatistics(object? sender, AvgStatHandler e)

{

this.Write(FileNameForStatistics, e.ToString());

}

private void Write(string FileName, string message)

{

using (StreamWriter file = new StreamWriter(FileName, true))

{

file.WriteLine(message);

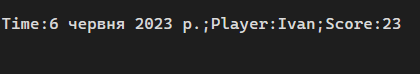
}

}

}

}

# Результат виконання програми:



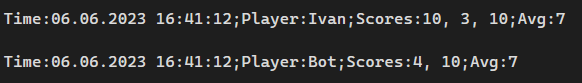


Рис.6.3-4. Реалізація завдання.

**Завдання 4:** Стратегія.

Лістинг коду:

using BlackJack;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace blackjack.Game.Strategy

{

public interface IStartegy

{

public List<int> HandlePlayer(Player player, CardsDeck Deck, List<int> scores);

}

}

namespace blackjack.Game.Strategy

{

public class StrategyOfEasyBot : IStartegy

{

private int POINT = 13;

public List<int> HandlePlayer(Player player, CardsDeck Deck, List<int> Scores)

{

while (PointsCounter.CountSum(player.DrawnCards) < this.POINT)

{

var card = player.DrawCard(Deck);

Scores.Add(PointsCounter.GetCardPower(card));

}

return Scores;

}

}

}

namespace blackjack.Game.Strategy

{

public class StrategyOfLooserBot : IStartegy

{

private int POINT = 0;

public List<int> HandlePlayer(Player player, CardsDeck Deck, List<int> Scores)

{

Console.WriteLine($"--{player.Name}: Я здаюсь...");

player.DrawnCards = new List<Card>();

return Scores;

}

}

}

namespace blackjack.Game.Strategy

{

public class StrategyOfPlayer : IStartegy

{

public List<int> HandlePlayer(Player player, CardsDeck Deck, List<int> Scores)

{

while (PointsCounter.CountSum(player.DrawnCards) < PointsCounter.MAX\_POINTS\_COUNT && player.ConfirmNextDraw())

{

var card = player.DrawCard(Deck);

Scores.Add(PointsCounter.GetCardPower(card));

}

return Scores;

}

}

}

namespace blackjack.Game.Strategy

{

public class StrategyOfRiskyBot : IStartegy

{

private int POINT = 19;

public List<int> HandlePlayer(Player player, CardsDeck Deck, List<int> Scores)

{

while (PointsCounter.CountSum(player.DrawnCards) < this.POINT)

{

var card = player.DrawCard(Deck);

Scores.Add(PointsCounter.GetCardPower(card));

}

return Scores;

}

}

}

**Висновок:** на лабораторній роботі навчився реалізовувати структурні шоблони проєктування Ланцюжок відповідальностей, Посередник, Спостерігач, Стратегія