**Туристическа агенция**

**Курсова работа по Шаблони за софтуерен дизайн I част**

1. **Описание:**

Конзолно приложение на .Net Core, което служи за преглед на екскурзии, предлагани от туристическа агенция, и резервация. Приложението се състои от 4 имплементирани шаблона – за създаване и преглед на екскурзии, за избор на екскурзия и добавяне на брой туристи, за резервация.

1. **Реализация:**
   1. **Abstract Factory** – в него ще се създават екскурзиите и по въведено от конзолата име на континент ще се принтира списък от екскурзии:
2. **Интерфейси:**

* ContinentFactory, абстрактен клас, който служи за създаване на абстрактни продукти. Той съдържа метод, който трябва да връща списък от екскурзии:

public abstract class ContinentFactory

{

public abstract List<Excursion> GetExcursions();

}

* Excursion, абстрактен клас, който играе ролята на модел, и определя типът на продукта, който ще се създаде, съдържа поле за ID, дестинация, дата на тръгване и връщане, име на екскурзовод, транспорт и цена:

public abstract class Excursion

{

public int ID { get; set; }

public string Destination { get; set; }

public DateTime DepartureDate { get; set; }

public DateTime ReturnDate { get; set; }

public string Transport { get; set; }

public double Price { get; set; }

public string TouristGuide { get; set; }

}

**Б) Класове:**

- Клас за отделните континенти, който наследява ContinentFactory, и в него ще се създават продуктите:

public class Asia : ContinentFactory

{

public override List<Excursion> GetExcursions()

{

return new List<Excursion>()

{

new India()

{

ID = 1,

Destination = "New Delhi",

DepartureDate = new DateTime(2021, 11, 15),

ReturnDate = new DateTime(2021, 11, 30),

Price = 4355.65,

TouristGuide = "George",

Transport = "Plane"

}

};

}

}

* Класове на отделните продукти, които наследяват абстрактния клас Excursion:

public class India : Excursion

{

}

* Клас Office, който използва абстрактните класове, за да изведе списък от екскурзии на конзолата. Той съдържа метод конструктор и метод, в който се обработва списъкът от екскурзии и се принтира на конзолата:

public Office(ContinentFactory continentFactory)

{

this.\_continent = continentFactory;

\_excursion = continentFactory.GetExcursions();

state = new ExcursionState();

}

public void GetExcursionsInfo()

{

Console.WriteLine("Travel around the World in: {0}", \_continent.GetType().Name);

Console.WriteLine();

for (int i = 0; i < \_excursion.Count; i++)

{

Console.WriteLine(\_excursion[i].ID + ". " + \_excursion[i].Destination);

if (\_excursion[i].DepartureDate > new DateTime(2021, 5, 1) || \_excursion[i].ReturnDate > new DateTime(2021, 5, 1))

{

if (\_excursion[i].DepartureDate < new DateTime(2021, 6, 30) || \_excursion[i].ReturnDate < new DateTime(2021, 6, 30))

{

state.Postpone();

state.PrintState();

Console.WriteLine(\_excursion[i].DepartureDate);

Console.WriteLine("To When: {0}", \_excursion[i].ReturnDate);

}

else

{

state.Organize();

state.PrintState();

Console.WriteLine(\_excursion[i].DepartureDate);

Console.WriteLine("To When: {0}", \_excursion[i].ReturnDate);

}

}

Console.WriteLine("Price: {0}", \_excursion[i].Price + "$");

Console.WriteLine("Tourist Guide: {0}", \_excursion[i].TouristGuide);

Console.WriteLine("Transport: {0}", \_excursion[i].Transport);

Console.WriteLine();

}

}

* 1. **FlyWeight –** той ще служи за допълнителна информация към резервацията след избор на екскурзия:

**А) Интерфейси:**

- IInformation – съдържа метод, който връща String с допълнителна информация за туристите:

public interface IInformation

{

public string GetInformation();

}

**Б) Класове:**

- Клас Children, който наследява интерфейса IInformation, за добавяне на брой деца и техните имена:

public class Children : IInformation

{

public string GetInformation()

{

Console.Write("Enter a child's name: ");

string children = Console.ReadLine();

Console.WriteLine();

return children;

}

}

* Клас Tourist, който наследява интерфейса IInformation, за добавяне на брой туристи и допълнителна информация за тях, която се валидира при въвеждане в конзолата от потребителя:

public class TouristInfo : IInformation

{

public string GetInformation()

{

Console.Write("Enter a tourist's name: ");

string tourist = Console.ReadLine();

Console.Write("Enter the tourist's phone number: ");

string phoneNumber = Console.ReadLine();

bool flag = false;

Regex regexPhone = new Regex(@"^0[8-9][7-9][0-9]{7}$");

Match matchPhone = regexPhone.Match(phoneNumber);

while (flag == false)

{

if (!matchPhone.Success)

{

Console.WriteLine("Invalid phone number! Try Again: ");

phoneNumber = Console.ReadLine();

matchPhone = regexPhone.Match(phoneNumber);

}

else

{

flag = true;

}

}

Console.Write("Enter an email address: ");

string email = Console.ReadLine();

bool isEmailValid = false;

Regex regex = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$");

Match match = regex.Match(email);

while (isEmailValid == false)

{

if (!match.Success)

{

Console.WriteLine("Invalid email address! Try Again: ");

email = Console.ReadLine();

match = regex.Match(email);

}

else

{

isEmailValid = true;

}

}

return tourist + " " + phoneNumber + " " + email;

}

}

* Клас AdditionalInformation, който съдържа конструктор, метод за допълнителна информация и метод, в който се пресмята общата сума и се принтира на конзолата избраната екскурзия заедно с информацията за туристите:

public class AdditionalInformation

{

private readonly Dictionary<string, IInformation> additionalInfos = new Dictionary<string, IInformation>();

private readonly List<Excursion> \_excursions;

private readonly List<string> touristsInfo = new List<string>();

string touristInfo;

public AdditionalInformation(List<Excursion> excursions)

{

\_excursions = excursions;

}

public IInformation GetAdditionalInfo(string infoType)

{

IInformation information = null;

if (additionalInfos.ContainsKey(infoType))

{

information = additionalInfos[infoType];

touristInfo = information.GetInformation();

touristsInfo.Add(touristInfo);

}

else

{

if (infoType == "Children")

{

information = new Children();

touristInfo = information.GetInformation();

touristsInfo.Add(touristInfo);

additionalInfos.Add(infoType, information);

}

else

if (infoType == "TouristInfo")

{

information = new TouristInfo();

touristInfo = information.GetInformation();

touristsInfo.Add(touristInfo);

additionalInfos.Add(infoType, information);

}

}

return information;

}

public void GetExcursionTotalPrice(int ID, int childrenCount, int touristsCount)

{

double totalExcursionSum = 0;

for (int i = 0; i < \_excursions.Count; i++)

{

if (\_excursions[i].ID == ID)

{

totalExcursionSum = (\_excursions[i].Price - (\_excursions[i].Price \* 0.6)) \* childrenCount + (touristsCount \* \_excursions[i].Price);

Console.WriteLine("Where: {0}", \_excursions[i].Destination);

Console.WriteLine("When: {0}", \_excursions[i].DepartureDate);

Console.WriteLine("To When: {0}", \_excursions[i].ReturnDate);

Console.WriteLine("Price: {0}", \_excursions[i].Price + "$");

Console.WriteLine("Tourist Guide: {0}", \_excursions[i].TouristGuide);

Console.WriteLine("Transport: {0}", \_excursions[i].Transport);

}

}

Console.WriteLine();

for (int i = 0; i < touristsInfo.Count; i++)

{

Console.WriteLine("Tourist Information: {0}", touristsInfo[i]);

}

Console.WriteLine();

Console.WriteLine("Total Price: {0}", Math.Round(totalExcursionSum, 2) + "$");

Console.WriteLine();

}

}

* 1. **Mediator:**

1. **Интерфейси:**

* IServerMediator – съдържа метод Send:

public abstract class IServerMediator

{

public abstract void Send(string message, Participant participant);

}

* Participant:

public abstract class Participant

{

protected ServerMediator mediator;

public Participant(ServerMediator server)

{

this.mediator = server;

}

}

**Б) Класове:**

- Класове за участниците, които наследяват интерфейса Participant и съдържат конструктор, методите Send и Receive:

public class Client : Participant

{

public Client(ServerMediator mediator): base(mediator)

{

}

public void Send(string message)

{

mediator.Send(message, this);

}

public void Receive(string message)

{

Console.WriteLine(message);

}

}

* Клас ServerMediator, който наследява интерфейса IServerMediator:

public class ServerMediator : IServerMediator

{

private Staff staff;

private Client client;

public Staff Staff

{

set { staff = value; }

}

public Client Client

{

set { client = value; }

}

public override void Send(string message, Participant participant)

{

if (participant == staff)

{

client.Receive(message);

}

else

{

staff.Receive(message);

}

}

}

* 1. **State:**

1. **Интерфейси:**

* Интерфейс IExcursionState, който съдържа метод PrintState:

public interface IExcursionState

{

public void PrintState();

}

**Б) Класове:**

- Класове за различните състояния, които наследяват интерфейса IExcursionState:

public class OrganizedState : IExcursionState

{

public static OrganizedState state = null;

public OrganizedState()

{

}

public static OrganizedState GetState()

{

if (state == null)

{

state = new OrganizedState();

}

return state;

}

public void PrintState()

{

Console.WriteLine("The excursion will take place on: ");

}

}

* Клас ExcursionState, който съдържа конструктор, методите Postpone, Organize, PrintState за промяна на състоянието и принтирането му на конзолата:

public class ExcursionState

{

public static IExcursionState state;

public ExcursionState()

{

state = OrganizedState.GetState();

}

public void Postpone()

{

state = PostponedState.GetState();

}

public void Organize()

{

state = OrganizedState.GetState();

}

public void PrintState()

{

state.PrintState();

}

}

* 1. **Допълнителни класове:**
* Клас Continent – съдържа метод ChooseContinent, който връща списък от екскурзии при въведено коректно име на континент:

public class Continent

{

List<Excursion> excursions;

private readonly ContinentFactory europe = new Europe();

private readonly ContinentFactory africa = new Africa();

private readonly ContinentFactory asia = new Asia();

private readonly ContinentFactory northAmerica = new NorthAmerica();

private readonly ContinentFactory southAmerica = new SouthAmerica();

private readonly ContinentFactory australia = new Australia();

public List<Excursion> ChooseContinent(string continentName)

{

while (continentName != null)

{

if (continentName == europe.GetType().Name)

{

Office office = new Office(europe);

office.GetExcursionsInfo();

excursions = europe.GetExcursions();

break;

}

else if (continentName == africa.GetType().Name)

{

Office office = new Office(africa);

Office.GetExcursionsInfo();

excursions = africa.GetExcursions();

break;

}

else if (continentName == asia.GetType().Name)

{

Office office = new Office(asia);

office.GetExcursionsInfo();

excursions = asia.GetExcursions();

break;

}

else if (continentName == northAmerica.GetType().Name)

{

Office office = new Office(northAmerica);

office.GetExcursionsInfo();

excursions = northAmerica.GetExcursions();

break;

}

else if (continentName == southAmerica.GetType().Name)

{

Office office = new Office(southAmerica);

office.GetExcursionsInfo();

excursions = southAmerica.GetExcursions();

break;

}

else if (continentName == australia.GetType().Name)

{

Office office = new Office(australia);

office.GetExcursionsInfo();

excursions = australia.GetExcursions();

break;

}

else

{

Console.WriteLine("You've entered a wrong name!");

continentName = Console.ReadLine();

}

}

return excursions;

}

}

* Клас ExcursionReservation за резервация на екскурзии, който съдържа метод за избор на екскурзия по дадено ID, въведено от потребителя в конзолата, и метод за допълнителна информация и потвърждаване на резервацията:

public class ExcursionReservation

{

private readonly List<Excursion> \_excursions;

private AdditionalInformation information;

private readonly ServerMediator mediator;

private readonly Staff staff;

private readonly Client client;

public ExcursionReservation(List<Excursion> excursions)

{

\_excursions = excursions;

mediator = new ServerMediator();

staff = new Staff(mediator);

client = new Client(mediator);

}

public void ChooseDestinationById(int excursionID)

{

for (int i = 0; i < \_excursions.Count; i++)

{

if (\_excursions[i].ID == excursionID)

{

if (\_excursions[i].DepartureDate > new DateTime(2021, 5, 1) || \_excursions[i].ReturnDate > new DateTime(2021, 5, 1))

{

if (\_excursions[i].DepartureDate < new DateTime(2021, 6, 30) || \_excursions[i].ReturnDate < new DateTime(2021, 6, 30))

{

Console.WriteLine("This excursion is postponed. Choose another.");

excursionID = int.Parse(Console.ReadLine());

i = -1;

}

else

{

Console.Clear();

GetAdditionalInfo(excursionID);

break;

}

}

}

if (excursionID > \_excursions.Count || excursionID < 0)

{

Console.WriteLine("Wrong ID");

excursionID = int.Parse(Console.ReadLine());

i = -1;

}

}

}

public void GetAdditionalInfo(int excursionID)

{

information = new AdditionalInformation(\_excursions);

int childrenCount;

int touristsCount;

if (information != null)

{

Console.Write("Children under 18 years old: ");

childrenCount = int.Parse(Console.ReadLine());

while (childrenCount >= 0 || childrenCount < 0)

{

if (childrenCount >= 0)

{

for (int i = 0; i < childrenCount; i++)

{

information.GetAdditionalInfo("Children");

}

break;

}

else

{

Console.WriteLine("Try again...");

childrenCount = int.Parse(Console.ReadLine());

}

}

Console.Clear();

Console.Write("Tourists: ");

touristsCount = int.Parse(Console.ReadLine());

while (touristsCount > 0 || touristsCount <= 0)

{

if (touristsCount > 0)

{

for (int i = 0; i < touristsCount; i++)

{

information.GetAdditionalInfo("TouristInfo");

}

break;

}

else

{

Console.WriteLine("There must be at least 1 tourist!");

touristsCount = int.Parse(Console.ReadLine());

}

}

Console.Clear();

if (childrenCount >= 0 && touristsCount > 0)

{

information.GetExcursionTotalPrice(excursionID, childrenCount, touristsCount);

mediator.Staff = staff;

mediator.Client = client;

staff.Send("Are you sure you want to reserve this trip? Yes/ No: ");

var message = Console.ReadLine();

if (message == "Yes")

{

client.Send(message);

}

else if (message == "No")

{

Console.Clear();

Console.Write("Enter a continent name: ");

string continentName = Console.ReadLine();

Continent continent = new Continent();

List<Excursion> excursions = continent.ChooseContinent(continentName);

Console.WriteLine("Choose an excursion: ");

int newexcursionID = int.Parse(Console.ReadLine());

ExcursionReservation reservation = new ExcursionReservation(excursions);

reservation.ChooseDestinationById(newexcursionID);

}

else

{

Console.WriteLine("Something went wrong...");

}

}

else

{

Console.WriteLine("Something went wrong...");

}

}

else

{

Console.WriteLine("Something went wrong...");

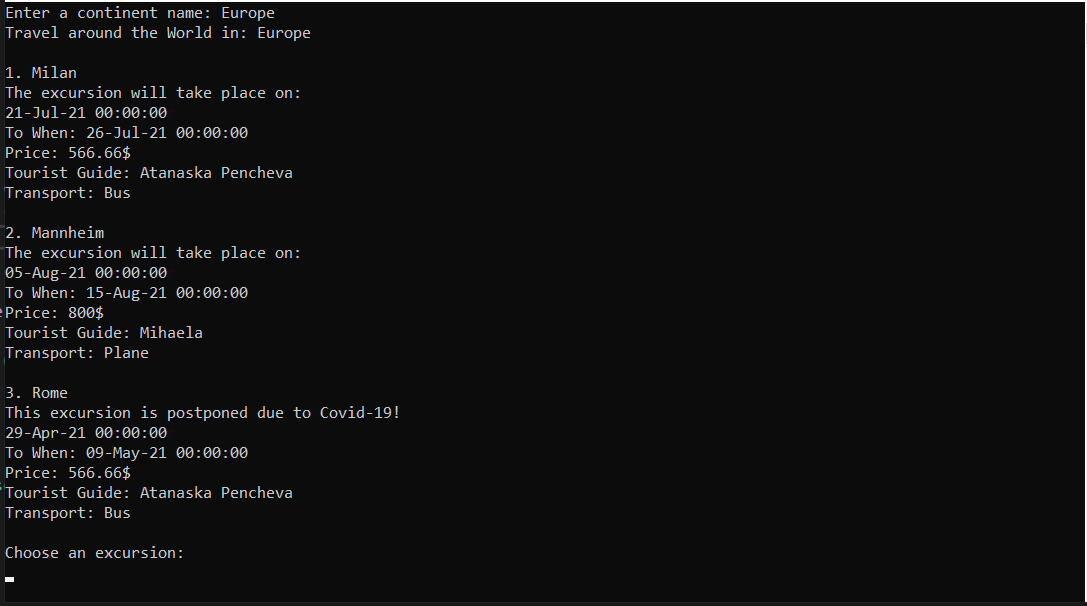
}

}

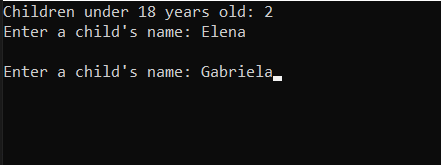
}

1. **Резултати:**

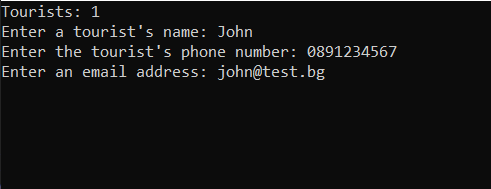
* Избор на континент:

****

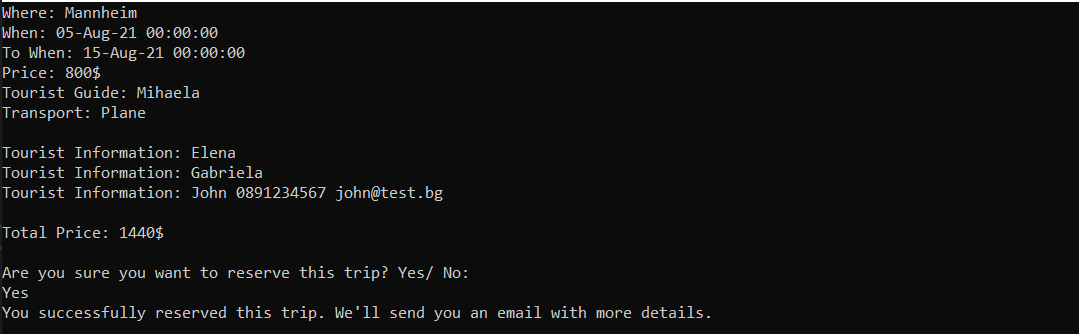
* Въвеждане на брой деца:

****

* Въвеждане на брой туристи:

****

* Резервация:

****