**Цель работы:** научить применять структурные шаблоны для создания программной системы.

**Вариант 25**

**Выполнение заданий:**

Шаблон «Состояние». Проект «Телефон». Операции: позвонить, ответить на звонок, завершить разговор, пополнить баланс. Предусмотреть: вероятность поступления звонка, звонок, разговор, заблокирован (баланс отрицательный).

using System;

namespace MultySidedSolution

{

class Program

{

static class ConsoleHelper

{

static public int ShowAndRetunCountVariantSelection(params string[] variants)

{

ShowListVariants(variants);

return ReturnNumberFromConsole(variants.Length);

}

static public void ShowListVariants(params string[] variants)

{

int lineNumber = 1;

foreach (string concretVariant in variants)

{

Console.WriteLine(lineNumber + ". " + concretVariant);

lineNumber++;

}

}

static public int ReturnNumberFromConsole(int maximumValue = int.MaxValue)

{

int choice;

try

{

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

if (choice > 0 && choice <= maximumValue)

{

return choice;

}

else

{

Console.WriteLine("Нет такого варианта, попробуйте еще раз:");

return ReturnNumberFromConsole(maximumValue);

}

}

catch

{

Console.WriteLine("Некоректный ввод, ведите число:");

return ReturnNumberFromConsole(maximumValue);

}

}

}

interface IPhonStatus

{

void Handle(Phone sm);

}

class CallState : IPhonStatus

{

public void Handle(Phone sm)

{

ConsoleHelper.ShowListVariants("Продолжить разговор", "Закончить разговор");

switch (ConsoleHelper.ReturnNumberFromConsole(2))

{

case 1:

Console.WriteLine("Раговор.");

Console.WriteLine("Раговор..");

Console.WriteLine("Раговор...");

sm.Balance--;

break;

case 2:

sm.State = new StandartState();

sm.Reqest();

break;

}

Handle(sm);

}

}

class BalanceState : IPhonStatus

{

public void Handle(Phone sm)

{

Console.WriteLine("Ваш Баланс: " + sm.Balance);

ConsoleHelper.ShowListVariants("Пополнить баланс на", "Назад");

switch (ConsoleHelper.ReturnNumberFromConsole(2))

{

case 1:

Console.WriteLine("Введите сумму на которую хотите пополнить баланс");

sm.Balance += ConsoleHelper.ReturnNumberFromConsole();

Console.WriteLine("Баланс поплнен");

break;

case 2:

sm.State = new StandartState();

sm.Reqest();

break;

}

Handle(sm);

}

}

class NegativeBalanceState : IPhonStatus

{

public void Handle(Phone sm)

{

Console.WriteLine("Ваш баланс отрицательный:" + sm.Balance + ", поэтому вы больше неможете совершать звонки");

ConsoleHelper.ShowListVariants("Перейти к балансу", "Выйти");

switch (ConsoleHelper.ReturnNumberFromConsole(2))

{

case 1:

sm.State = new BalanceState();

sm.Reqest();

break;

case 2:

sm.State = new StandartState();

sm.Reqest();

break;

}

}

}

class StandartState : IPhonStatus

{

public void Handle(Phone sm)

{

ConsoleHelper.ShowListVariants("Позвонить", "Ответить на звонок", "Пополнить баланс");

switch (ConsoleHelper.ReturnNumberFromConsole(3))

{

case 1:

if (sm.Balance < 0) sm.State = new NegativeBalanceState();

else sm.State = new CallState();

sm.Reqest();

break;

case 2:

sm.State = new CallState();

sm.Reqest();

break;

case 3:

sm.State = new BalanceState();

sm.Reqest();

break;

}

}

}

class Phone

{

public int Balance;

public Phone(int Balance)

{

this.Balance = Balance;

}

public IPhonStatus State { get; set; }

public void Reqest()

{

State.Handle(this);

}

}

static void Main()

{

Phone sm = new Phone(1);

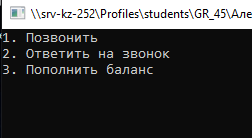
sm.State = new StandartState();

sm.Reqest();

}

}

}



[Ссылка на файл](MultySidedSolution/bin/Debug/netcoreapp3.1/MultySidedSolution.exe)

*Ответы на контрольные вопросы:*

1. Каково назначение поведенческих шаблонов проектирования?

Ответ: Поведенческие шаблоны определяют взаимодействие между объектами, увеличивая таким образом его гибкость.

2. Для чего предназначен поведенческий шаблон Состояние?

Ответ: Стратегия — поведенческий шаблон проектирования, предназначенный для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости.

3. Какое решение предлагается в шаблоне Состояние?

Ответ: Определить для каждого состояния отдельный класс со стандартным интерфейсом.

4. Что понимают под стратегией в шаблоне Стратегия?

Ответ: понимают то, что семейство алгоритмов может взаимозаменяться то есть меняться как стратегия.

5. Какое решение предлагается в шаблоне Стратегия?

Ответ: Определить для каждого алгоритма (стратегии) отдельный класс со стандартным интерфейсом.

6. Какую проблему позволяет решить шаблон Шаблонный метод?

Ответ: Как определить алгоритм и реализовать возможность переопределения некоторых шагов алгоритма в производных классах без изменения общей структуры алгоритма?