МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЕТ

по лабораторной работе №1

по дисциплине

Шаблоны проектирования программного обеспечения

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Жевнерчук Д.В,\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Игнаков К. М.\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Фомин Е. Н.\_\_\_

(подпись) (фамилия, и.,о.)

\_\_\_19-ИВТ-3\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

Вариант 12.

Задание:

Разработайте объектно-ориентированную модель телефона.

Телефон обладает атрибутами: номер, баланс, вероятность поступления звонка.

С телефоном можно выполнить следующие операции: позвонить, ответить на звонок, завершить разговор, пополнить баланс.

Телефон может находиться в следующих состояниях: Ожидание, Звонок, Разговор, Заблокирован (баланс отрицательный).

Реализуйте приложение, демонстрирующее переход телефона между состояниями.

# Проектное решение

**Обоснование выбора паттернов**

Основная идея в том, что телефон может находиться в одном из нескольких состояний, которые всё время сменяют друг друга. Набор этих состояний, а также переходов между ними, предопределён и конечен. Находясь в разных состояниях, программа может по-разному реагировать на одни и те же события, которые происходят с ней.

Паттерн Состояние предлагает создать отдельные классы для каждого состояния, в котором может пребывать объект, а затем вынести туда поведения, соответствующие этим состояниям.

Вместо того, чтобы хранить код всех состояний, первоначальный объект, называемый контекстом, будет содержать ссылку на один из объектов-состояний и делегировать ему работу, зависящую от состояния.

Благодаря тому, что объекты состояний будут иметь общий интерфейс, контекст сможет делегировать работу состоянию, не привязываясь к его классу. Поведение контекста можно будет изменить в любой момент, подключив к нему другой объект-состояние.

Производными классами для класса State (Состояние) являются:

1. BlockedState – телефон не может совершать исходящие вызовы. Выйти из этого состояния можно только пополнив баланс.
2. WaitingState – телефон находится в ожидании команды. Из этого состояния возможно перейти в состояние разговора (TalkingState), приняв входящий, или в состояние вызова (CallingState), позвонив на какой-то номер.
3. CallingState – телефон находится в состоянии исходящего вызова. В последствии (после ответа собеседника) переход в состояние разговора (TalkingState).
4. TalkingState – телефон находится в состоянии разговора. Сбросив вызов, можно перейти в состояние ожидания (WaitingState).

Для удобного перехода между состояниями телефон получает команду в консоли от пользователя.

Класс Phone создан для хранения информации о телефоне: баланс, номер телефона, состояние телефона.

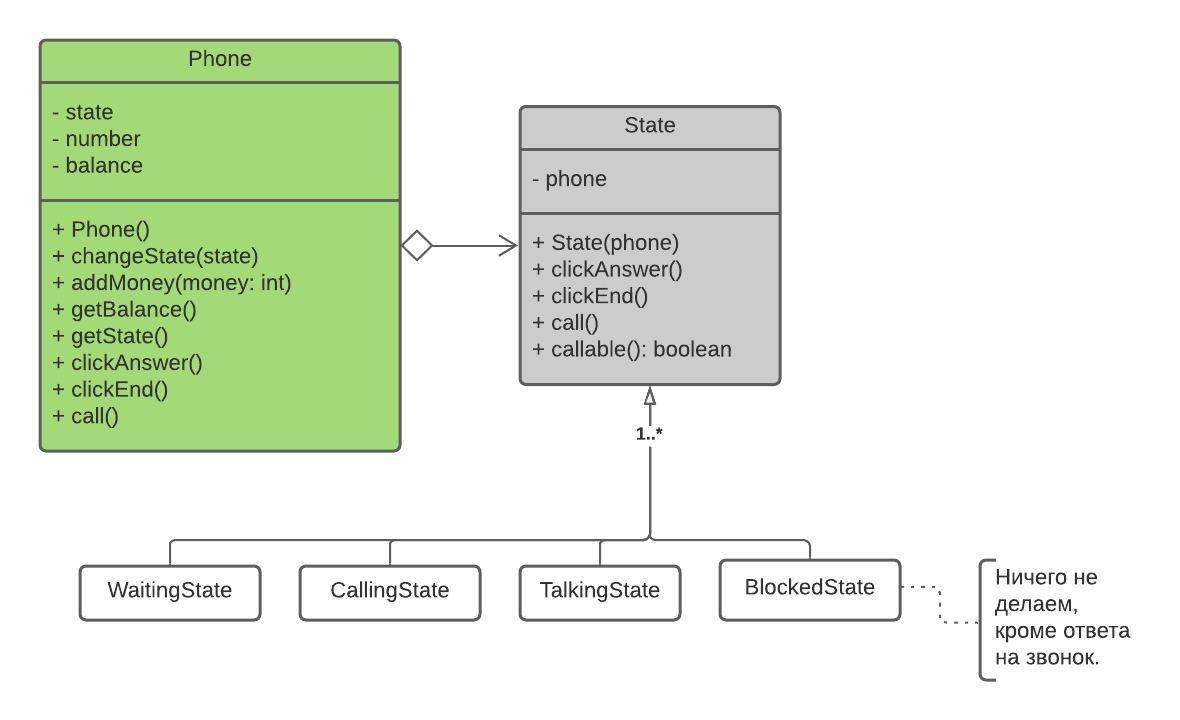


Рис. 1 – Диаграмма классов

# Приложение 1

**Программный код**

Phone.java

package phone;

import state.State;

import state.States.\*;

public class Phone {

    private State state;

    private String number;

    private float balance;

    public Phone () {

        this.state = new WaitingState(this);

    }

    public float getBalance() {

        return balance;

    }

    public void changeState (State state) {

        this.state = state;

    }

    public boolean addMoney (float money) {

        if (money > 0) {

            this.balance += money;

            return true;

        }

        else

            return false;

    }

    public void payMoney (long time) {

        balance -= 0.03f \* time;

        return;

    }

    public String clickAnswer () {

        return this.state.clickAnswer();

    }

    public String clickEnd () {

        return this.state.clickEnd();

    }

    public String call (String num) {

        if (state.callable()) {

            return this.state.call();

        }

        return null;

    }

}

State.java

package state;

import phone.Phone;

public abstract class State {

    public Phone phone;

    public State(Phone phone) {

        this.phone = phone;

    }

    public abstract boolean callable();

    public abstract String clickAnswer();

    public abstract String clickEnd();

    public abstract String call();

}

TalkingState.java

package state.States;

import state.State;

import phone.Phone;

public class TalkingState extends State {

    public TalkingState(Phone phone) {

        super(phone);

    }

    /\*\*

     \* Добавлено для удобства работы совершения звонков

     \* Чтобы не обращаться к названию состояния и балансу, используем метод

\* callable

     \* Он проверяет возможность совершить звонок в данном состоянии телефона

     \* Во время разговора мы не можем кому-то позвонить

     \* @return boolean

     \*/

    @Override

    public boolean callable() {

        return false;

    }

    /\*\*

     \* Из состояния разговора нельзя ответить на ещё один звонок

     \* @return String

     \*/

    @Override

    public String clickAnswer() {

        return null;

    }

    /\*\*

     \* Завершение звонка

     \* @return String

     \*/

    @Override

    public String clickEnd() {

        phone.changeState(new WaitingState(phone));

        return this.getClass().getName() + " -> clickEnd" + " -> WaitingState";

    }

    /\*\*

     \* Во время разговора мы не можем позвонить другому человеку

     \* @return String

     \*/

    @Override

    public String call() {

        return null;

    }

}

WaitingState.java

package state.States;

import state.State;

import phone.Phone;

public class WaitingState extends State {

    public WaitingState(Phone phone) {

        super(phone);

    }

    /\*\*

     \* Добавлено для удобства работы совершения звонков

     \* Чтобы не обращаться к названию состояния и балансу, используем метод

\* callable

     \* Он проверяет возможность совершить звонок в данном состоянии телефона

     \* Если баланс положительный, то мы можем позвонить из состояния ожидания

     \* @return boolean

     \*/

    @Override

    public boolean callable() {

        if (phone.getBalance() > 0) {

            return true;

        }

        return false;

    }

    /\*\*

     \* Мы можем ответить на входящий звонок в режиме ожидания

     \* @return String

     \*/

    @Override

    public String clickAnswer() {

        phone.changeState(new CallingState(phone));

        return this.getClass().getName() + " -> clickAnswer"

+ " -> CallingState";

    }

    /\*\*

     \* Проверяем баланс телефона

     \* Если он положительный, то переходим к состоянию звонка,

     \* Если он отрицательный, то переходим в режиму блокировки.

     \* @return String

     \*/

    @Override

    public String call() {

        if (phone.getBalance() > 0) {

            phone.changeState(new CallingState(phone));

            return this.getClass().getName() + " -> call" + " -> CallingState";

        } else {

            phone.changeState(new BlockedState(phone));

            return this.getClass().getName() + " -> call" + " -> BlockedState";

        }

    }

    /\*\*

     \* В режиме ожидания мы не можем сбросить звонок

     \* @return null

     \*/

    @Override

    public String clickEnd() {

        return null;

    }

}

BlockedState.java

package state.States;

import phone.Phone;

import state.State;

public class BlockedState extends State {

    public BlockedState(Phone phone) {

        super(phone);

    }

    /\*\*

     \* Добавлено для удобства работы совершения звонков

     \* Чтобы не обращаться к названию состояния и балансу, используем метод

\* callable

     \* Он проверяет возможность совершить звонок в данном состоянии телефона

     \* Если баланс положительный, то мы можем выйти из состояния блокировки и позвонить

     \* @return boolean

     \*/

    @Override

    public boolean callable() {

        if (phone.getBalance() > 0) {

            return true;

        }

        return false;

    }

    /\*\*

     \* Мы можем принять звонок от другого человека с отрицательным балансом

     \* @return String

     \*/

    @Override

    public String clickAnswer() {

        phone.changeState(new CallingState(phone));

        return this.getClass().getName() + " -> clickAnswer"

+  " -> CallingState";

    }

    /\*\*

     \* Мы ни с кем не разговариваем - не за чем завершать разговор.

     \* Проверяем не стал ли баланс положительным для возможного перехода

     \* к другому состоянию.

     \* @return String

     \*/

    @Override

    public String clickEnd() {

        if (phone.getBalance() > 0) {

            phone.changeState(new WaitingState(phone));

            return this.getClass().getName() + " -> clickEnd"

+  " -> WaitingState";

        }

        return null;

    }

    /\*\*

     \* Проверяем не стал ли баланс положительным.

     \* Если да, то переходим к состоянию звонка,

     \* если нет, то ничего не делаем.

     \* @return String

     \*/

    @Override

    public String call() {

        if (phone.getBalance() > 0) {

            phone.changeState(new CallingState(phone));

            return this.getClass().getName() + " -> call" +  " -> CallingState";

        }

        return null;

    }

}

CallingState.java

package state.States;

import state.State;

import phone.Phone;

public class CallingState extends State {

    public CallingState(Phone phone) {

        super(phone);

    }

    /\*\*

     \* Добавлено для удобства работы совершения звонков

     \* Чтобы не обращаться к названию состояния и балансу, используем метод

* callable

     \* Он проверяет возможность совершить звонок в данном состоянии телефона

     \* Во время вызова мы не можем позвонить ещё раз

     \* @return boolean

     \*/

    @Override

    public boolean callable() {

        return false;

    }

    /\*\*

     \* В режиме разговора мы не можем ответить на звонок

     \* @return null

     \*/

    @Override

    public String clickAnswer() {

        return null;

    }

    /\*\*

     \* В режиме разговора мы не можем позвонить

     \* @return null

     \*/

    @Override

    public String call() {

        return null;

    }

    /\*\*

     \* Мы можем сбросить звонок при разговоре и перейти в режим ожидания

     \* @return String

     \*/

    @Override

    public String clickEnd() {

        phone.changeState(new WaitingState(phone));

        return this.getClass().getName() + " -> clickEnd" + " -> WaitingState";

    }

}

main.java

import phone.Phone;

import java.time.Instant;

import java.time.Duration;

import java.util.Scanner;

public class main {

    public static void main(String[] args) {

        Phone phone = new Phone();

        Scanner sc = new Scanner(System.in);

        String command;

        String state;

        String help =

                "\nhelp\t\t\t\tshow this text\n" +

                "addMoney <float>\tadd money to the phone`s balance\n" +

                "showBalance\t\t\tshow your phone`s balance\n" +

                "answerCall\t\t\tanswer to incoming call\n" +

                "call <number>\t\tcall to number

(only if balance more then 0.5)\n" +

                "endCall\t\t\t\tend talking\n" +

                "turnOff\t\t\t\tturn off your phone\n";

        String telephoneNumber;

        float deposit = 0;

        Instant start = Instant.now();

        Instant finish = Instant.now();

        boolean didYouCall = false;

        System.out.println(help);

        System.out.println("Phone is turn on!");

        while (true)

        {

            System.out.print("Enter command: ");

            command = sc.next();

            state = "";

            switch (command)

            {

                case "showBalance":

                    System.out.println("Your balance: " + phone.getBalance());

                    break;

                case "turnOff":

                    System.out.println("Turning off... Good bye!");

                    return;

                case "addMoney":

                    deposit = sc.nextFloat();

                    if (phone.addMoney(deposit)) {

                        System.out.print("The balance is replenished!\n");

                    }

                    else {

                        System.out.print("Something went wrong! " +

                                "May be you entered a negative amount or

a zero?\n");

                    }

                    break;

                case "call":

                    telephoneNumber = sc.next();

                    state = phone.call(telephoneNumber);

                    didYouCall = true;

                    if (state == null){

                        System.out.println("Can not call!");

                    } else {

                        start = Instant.now();

                        System.out.println("Calling on number: "

+ telephoneNumber);

// System.out.println(state);

                    }

                    break;

                case "answerCall":

                    state = phone.clickAnswer();

                    didYouCall = false;

                    if (state == null) {

                        System.out.println("Can not answer!");

                    } else {

                        System.out.println("Answer...");

                        phone.clickAnswer();

                        start = Instant.now();

// System.out.println(state);

                    }

                    break;

                case "endCall":

                    state = phone.clickEnd();

                    if (state == null) {

                        System.out.println("Can not end call!");

                    } else {

                        finish = Instant.now();

                        long elapsed =

(Duration.between(start, finish).toMillis())/1000;

                        System.out.println("Your phone conversation lasted: "

+ elapsed + " seconds");

                        if (didYouCall)

                            phone.payMoney(elapsed);

                        System.out.println("Call is end.");

// System.out.println(state);

                    }

                    break;

                case "help":

                    System.out.println(help);

                    break;

                default:

                    System.out.println("You entered the wrong command\n" +

                            "Enter 'help' to get information about the program.");

                    break;

            }

        }

    }

}

# Приложение 2

**Результаты тестирования**

