**Министерство образования и науки Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**

**МЕХАНИКИ И ОПТИКИ”**

**Лабораторная работа №4.**

ФИО студента: Бекмухаметов Владислав Робертович

Направление подготовки(специальность): 09.03.04

Группа: P3119

ФИО преподавателя: Письмак Алексей Евгеньевич

**Санкт-Петербург, 2021**

Оглавление

[Текст работы 3](#_Toc93006214)

[UML диаграмма 4](#_Toc93006215)

[Код 5](#_Toc93006216)

[Результат работы программы 16](#_Toc93006217)

# Текст работы

Фрекен Бок покорилась и послушно сказала, что она, конечно, это тоже способна понять, если как следует подумает. Но она явно не желала, чтобы эти сказочные существа мешали ей пить кофе с дядей Юлиусом, и тут же увлекла его назад в гостиную. И Карлсон и Малыш снова оказались перед закрытой дверью. "И смотреть в глазок было не очень-то весело", - думал Малыш. И Карлсон тоже так думал. Да, представьте себе, что и Карлсон так думал. Тут, к счастью, позвонил телефон. Малыш подошел. Какой-то женский голос попросил фрекен Бок. Малыш понял, что это Фрида с Фрейгатен, и очень обрадовался. Теперь он имел законное право помешать фрекен Бок, и хотя вообще-то он был милым мальчиком, он сделал это с радостью. - Фрекен Бок, вас к телефону! - крикнул он и громко постучал в дверь гостиной. Но безрезультатно. - Скажи, что я занята! - крикнула в ответ фрекен Бок из-за закрытой двери. Фрида была так же бессильна нарушить ее уединение с дядей Юлиусом, как и сказочные существа. Малыш вернулся к телефону и сказал, что фрекен Бок не может подойти. Но Фрида хотела во что бы то ни стало узнать, чем так занята ее сестра, что не может даже подойти к телефону, и засыпала Малыша вопросами. В конце концов он сказал: - Лучше всего спросите ее завтра сами. Он положил трубку и поглядел; что делает Карлсон. Но Карлсона нигде не было. Малыш искал его всюду, в конце концов прибежал на кухню. Подошел к открытому окну и увидел на подоконнике странную фигуру. Там стояло верхом на маминой любимой метле таинственное существо, готовое к отлету. По всей вероятности, это и был Карлсон, хотя по виду существо это было похоже на маленькую ведьму: лицо, вымазанное черным, на голове косынка, а на плечах развевался ведьминский плащ - мамин большой цветастый передник.

# UML диаграмма

# Код

FairtPerson.java

package src.*A*bs*C*lass;

import src.Location;

import src.Person;

public abstract class FairyPerson extends Person {

    public FairyPerson(final String name, final Location loc) {

        super(name, loc);

    }

}

Item.java

package src.*A*bs*C*lass;

import src.Location;

public abstract class Item extends Thing {

    abstract public String getStatus();

    public void setLocation(Location loc) {

        this.location = loc;

    }

    public Item(final String name, final Location loc) {

        this.name = name;

        try {

            loc.addThing(this);

            this.location = loc;

        } catch (Exception e) {

            System.out.println(e);

        }

    }

}

Thing.java

package src.*A*bs*C*lass;

import java.util.Objects;

import src.Location;

import src.Person;

public abstract class Thing {

    protected String name;

    protected Location location;

    @Override

    public boolean equals(final Object o) {

        if (this == o)

            return true;

        if (o == null || getClass() != o.getClass())

            return false;

        final Person person = (Person) o;

        return Objects.equals(name, person.name);

    }

    @Override

    public int hashCode() {

        return Objects.hash(name, getClass());

    }

    @Override

    public String toString() {

        return this.name + "|" + this.getClass();

    }

    public String getName() {

        return this.name;

    }

    public Location getLocation() {

        return this.location;

    }

}

Drinks.java

package src.enums;

import java.util.Arrays;

public enum Drinks {

    COLA("Кола"),

    BEER("Пиво"),

    TEA("Чай"),

    WATER("Вода"),

    COFE("Кофе");

    private String name;

    Drinks(String name) {

        this.name = name;

    }

    Drinks() {

    };

    public String getName() {

        return name;

    }

    @Override

    public String toString() {

        return name;

    }

    public static void menu() {

        System.out.println(Arrays.toString(Drinks.values()));

    }

}

Meal.java

package src.enums;

import java.util.Arrays;

public enum Meal {

    MEAT("Мясо"),

    BREAD("Хлеб"),

    SALAD("Салат");

    private String name;

    Meal(String name) {

        this.name = name;

    }

    public String getName() {

        return name;

    }

    public static void menu() {

        System.out.println(Arrays.toString(Meal.values()));

    }

    @Override

    public String toString() {

        return name;

    }

}

DoorInterface.java

package src.*I*nterfaces;

public interface DoorInterface {

    public void close();

    public void open();

    public String getDoorStatus();

}

LocationInterface.java

package src.*I*nterfaces;

import src.LocationOverFlow;

import src.AbsClass.\*;

public interface LocationInterface {

    abstract public void addThing(final Thing o) throws LocationOverFlow;

    public int getCount();

}

SayAble.java

package src.*I*nterfaces;

public interface SayAble {

    public void sayPhrase(String phrase);

}

ArrayListException.java

package src;

public class ArrayListException extends RuntimeException {

    public ArrayListException(String message) {

        super(message);

    }

}

InvalidPhoneNumberException.java

package src;

public class InvalidPhoneNumberException extends Exception {

    public InvalidPhoneNumberException(String message){

        super(message);

    }

}

Karlson.java

package src;

import src.AbsClass.FairyPerson;

public class Karlson extends FairyPerson {

    public Vint vint = new Vint();

    public Karlson(String name, Location loc) {

        super(name, loc);

    }

    public class Vint {

        private String vintStatus = "Off";

        public void setVintStatus(String vintStatus) {

            this.vintStatus = vintStatus;

        }

        public String getVintStatus() {

            return vintStatus;

        }

        public void turnOn() {

            this.vintStatus = "On";

            System.out.println("Vint on");

        }

        public void turnOff() {

            this.vintStatus = "Off";

            System.out.println("Vint off");

        }

    }

}

Location.java

package src;

import java.util.ArrayList;

import src.Interfaces.LocationInterface;

import src.AbsClass.\*;

public class Location implements LocationInterface {

    final String name;

    public int count = 0;

    final private static int maxPersonCount = 5;

    @Override

    public int getCount() {

        return this.count;

    }

    public String getName() {

        return this.name;

    }

    public ArrayList<Thing> listThing = new ArrayList<Thing>();

    public ArrayList<Thing> getAllThing() throws ArrayListException {

        if (this.listThing.size() == 0) {

            throw new ArrayListException(this.name + " is empty");

        }

        return this.listThing;

    }

    @Override

    public void addThing(final Thing o1){

        if (count != Location.maxPersonCount) {

            this.count += 1;

            this.listThing.add(o1);

        } else {

            throw new LocationOverFlow(o1.getName() + " не может быть добавлен в локацию " + this.getName());

        }

    }

    public Location(final String name) {

        this.name = name;

    }

    public Location(final String name, final int count) {

        this.name = name;

        this.count = count;

    }

    public static class LocationInfo {

        public static int getMaxCountPerson() {

            return maxPersonCount;

        }

    }

}

LocationOverFlow.java

package src;

public class LocationOverFlow extends Error {

    public LocationOverFlow(String message){

        super(message);

    }

}

Person.java

package src;

import src.AbsClass.Item;

import src.AbsClass.Thing;

import src.Interfaces.SayAble;

import java.util.Objects;

public class Person extends Thing implements SayAble {

    public Person(final String name, final Location loc) {

        this.name = name;

        try {

            loc.addThing(this);

            this.location = loc;

        } catch (Exception e) {

            System.out.println(e);

        }

    }

    public void setLocation(final Location location) {

        this.location = location;

    }

    @Override

    public void sayPhrase(final String phrase) {

        System.out.println(this.name + ": " + phrase);

    }

    @Override

    public boolean equals(final Object o) {

        if (this == o)

            return true;

        if (o == null || getClass() != o.getClass())

            return false;

        final Person person = (Person) o;

        return Objects.equals(name, person.name) && (Objects.equals(location, person.location));

    }

    @Override

    public int hashCode() {

        return Objects.hash(name, location, getClass());

    }

    @Override

    public String toString() {

        return this.name + "|" + this.location + this.getClass();

    }

    public void move(Location loc) {

        loc.addThing(this);

        System.out.println(loc.getCount());

        this.location = loc;

        System.out.println(this.getName() + " переместился в " + this.location.getName());

    }

    public void movePredmet(Item predmet, Location loc) {

        predmet.setLocation(loc);

        System.out.println(this.name + " переместил " + predmet.getName() + " в " + loc.getName());

    }

}

Telephone.java

package src;

import src.AbsClass.Thing;

public class Telephone extends Thing {

    String name;

    public Telephone(String name, Location loc) {

        this.name = name;

        this.location = loc;

    }

    public void call(String number) throws InvalidPhoneNumberException {

        class PhoneNumber {

            String phoneNumber = null;

            public PhoneNumber(String phoneNumber) throws InvalidPhoneNumberException {

                String currentNumber = phoneNumber.replaceAll("[^0-9]", "");

                if (currentNumber.length() == 11) {

                    this.phoneNumber = currentNumber;

                } else {

                    throw new InvalidPhoneNumberException("invalid phone number");

                }

            }

            public String getNumber() {

                return phoneNumber;

            }

        }

        PhoneNumber phoneNumber = new PhoneNumber(number);

        System.out.println("звонок абоненту +" + phoneNumber.getNumber());

    }

}

Main.java

import src.InvalidPhoneNumberException;

import src.Karlson;

import src.Location;

import src.Telephone;

import src.AbsClass.\*;

import src.Interfaces.DoorInterface;

public class Main {

    public static void main(String[] args) {

        Location room = new Location("Комната");

        Location kitchen = new Location("Кухня");

        System.out.println(Location.LocationInfo.getMaxCountPerson());

        // Person human = new Person("Иван", room);

        Karlson kar = new Karlson("Карлсон", room);

        System.out.println(kar.vint.getVintStatus());

        kar.move(kitchen);

        System.out.println(kar.getLocation().getName());

        DoorInterface door = new DoorInterface() {

            String doorStatus = "Closed";

            public void open() {

                doorStatus = "Open";

            }

            public void close() {

                doorStatus = "Closed";

            }

            public String getDoorStatus() {

                return doorStatus;

            }

        };

        door.open();

        System.out.println(door.getDoorStatus());

        door.close();

        try {

            for (Thing i : kitchen.getAllThing()) {

                System.out.print(i.getName() + " ");

            }

            System.out.println();

        } catch (Exception e) {

            System.out.println(e);

        }

        Telephone test = new Telephone("Сотовый", room);

        try {

            test.call("+79111274352");

        } catch (InvalidPhoneNumberException e) {

            System.out.println(e);

        }

    }

}

// Добавлено проверяемое исключение ArrayListException in Location.java in

// getAllPerson()

// Добавлено не проверяемое исключение LocationOverFlow in Location.java in

// addPerson()

// Переделать метод getAllPerson in Location.java +

// Попробовать придумать как перемещать в доступную локацию персонажей при +

// выбросе исключения LocationCountException +

// inner class: Vint in Karlson.java

// Anonymous class: Predmet door = new Predmet(){} in Main.java

// Static nested class: Location.java public static class LocationInfo

// Method-local Inner class: Telephone.java

# Результат работы программы

Изображение выглядит как текст

Автоматически созданное описание

# Вывод

В данной работе я познакомился с вложенными классами, а именно c Inner class, local inner classes и анонимными классами. Разобрался в работе исключений в java, научился проводить их обработку, а также написал свои проверяемые и непроверяемые исключения.