**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Факультет инфокоммуникационных сетей и систем (ИКСС)

Лабораторная работа №6,7

по курсу

«Разработка Java-приложений управления телекоммуникациями»

Группа: ИКПИ-14

Выполнил студент: Хохлов Т. В.

Принял преподаватель: Березин А.Ю.

Место для подписи \_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2023 г.

# **Цель работы**

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

**Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание**

**Описание кода**

Программа состоит из 5 созданных классов:

* **Appliances.java – класс с/х техники**
* **Freezer.java – класс холодильника наследник Appliances**
* **Washer.java – класс стиральной машины наследник Appliances**
* **Dishwasher.java – класс посудомоечной машины наследник Appliances**
* **Jackson.java – класс для сериализации/десериализации списка созданных устройсв.**

**Основная логика:**

Программа представляет собою пример приложения «Умного дома», в котором можно «подключить» определенные устройства, указав их названия и модель.

У каждого устройства есть некоторые варианты взаимодействия. Например:

Для класса **Freezer:**

* Функция имитиации загрузки/разгрузки
* Включение/выключение устройства
* Смена режима работы

Для класса **Washer:**

* Функция имитиации стирки белья
* Включение/выключение устройства
* Смена режима работы

Для класса **Dishwasher:**

* Функция имитиации мойки посуды
* Включение/выключение устройства
* Смена режима работы

# **Листинг**

## **Appliances.java**

package org.PrescentV;

import com.fasterxml.jackson.annotation.JsonSubTypes;

import com.fasterxml.jackson.annotation.JsonTypeInfo;

import com.fasterxml.jackson.annotation.JsonTypeName;

import java.util.Random;

import java.util.Scanner;

import org.apache.logging.log4j.\*;

@JsonTypeInfo(use=JsonTypeInfo.Id.NAME, include=JsonTypeInfo.As.PROPERTY, property="type", visible = true)

@JsonSubTypes({

        @JsonSubTypes.Type(value = Freezer.class, name = "Freezer"),

        @JsonSubTypes.Type(value = Washer.class, name = "Washer"),

        @JsonSubTypes.Type(value = Dishwasher.class, name = "Dishwasher")

})

@JsonTypeName("type")

public abstract class Appliances {

    protected static final Logger logger = LogManager.getLogger(Main.class);

    private String UUID;

    protected String modelName;

    protected String brandName;

    protected int oi = 0;

    public Appliances() {

        this.UUID = null;

        this.modelName = null;

        this.brandName = null;

        this.oi = 0;

    }

    public Appliances(String UUID,String modelName, String brandName, int oi) {

        this.UUID = UUID;

        this.modelName = modelName;

        this.brandName = brandName;

        this.oi = oi;

    }

    public Appliances(String modelName, String brandName) {

        String characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

        String randomString = "";

        Random rand = new Random();

        char[] id = new char[8];

        for (char c : id) {

            c = characters.charAt(rand.nextInt(characters.length()));

            randomString += c;

        }

        this.UUID = randomString;

        this.modelName = modelName;

        this.brandName = brandName;

        this.oi = 0;

    }

    public void toggle() {

        if (oi == 1) {

            logger.info("Device turned OFF");

            oi = 0;

        } else if (oi == 0) {

            logger.info("Device turned ON");

            oi = 1;

        } else {

            logger.error("DEVICE OUT OF SERVICE!");

        }

    }

    abstract int changeMode(int choice);

    abstract void outputCurrentMode();

    abstract void outputCurrentStatus();

    abstract void outputAllModes();

    public String getModelName() {

        return this.modelName;

    }

    public String getBrandName() {

        return this.brandName;

    }

    public int getOi() {

        return this.oi;

    }

    public String getUUID() {

        return this.UUID;

    }

    public void setModelName(String modelName) {

        this.modelName = modelName;

    }

    public void setBrandName(String brandName) {

        this.brandName = brandName;

    }

    public void setOi(int mode) {

        this.oi = mode;

    }

    public void setUUID(String UUID) {this.UUID = UUID;}

    static void pressEnterToContinue(Scanner systemIn)

    {

        System.out.println("\nНажмите Enter чтобы продолжить...");

        try {

            System.in.read();

            systemIn.nextLine();

        } catch(Exception e) {

            e.getMessage();

        }

    }

    static void cleanScreen() {

        System.out.printf("\033[2J");

    }

}

## **Dishwasher.java**

package org.PrescentV;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonTypeInfo;

import com.fasterxml.jackson.annotation.JsonTypeName;

import java.util.Random;

import java.util.Scanner;

@JsonTypeInfo(use=JsonTypeInfo.Id.NAME, include=JsonTypeInfo.As.PROPERTY, property="type", visible = true)

@JsonTypeName("type")

@JsonIgnoreProperties(ignoreUnknown = true)

public class Dishwasher extends Appliances {

    long start;

    int loaded;

    int mode;

    int status;

    String[] statuses = {"Ожидает","Моет","Смывает"};

    String[] modes = {"Автоматическая","Обычная","Деликатная","Экспресс","Спокойной ночи","Ополаскивание"};

    public Dishwasher() {

    }

    public Dishwasher(String modelName, String brandName) {

        super(modelName, brandName);

        Random rand = new Random();

        this.loaded = rand.nextInt(3);

        this.status = 0;

        logger.info("Dishwasher device connected");

    }

    public void outputCurrentStatus() {

        if (this.oi == 1) {

            System.out.println("Устройство включено.");

            System.out.println("Текущий статус: " + statuses[this.status]);

            outputCurrentLoad();

        } else {

            System.out.println("Устройство выключено.");

        }

        logger.info("CurrentStatus showed for Dishwasher");

    }

    public void outputCurrentMode() {

        System.out.println("Текущий режим: " + modes[this.mode]);

    }

    public void outputCurrentLoad() {

        System.out.println("Текущее количество ресурсов: " + this.loaded);

    }

    public void outputAllModes() {

        for (int i = 0; i < modes.length; i++) {

            System.out.println(i+1 + ". " + modes[i]);

        }

        logger.info("AllModes showed for Dishwasher");

    }

    public int changeMode(int choice) {

        if (this.status == 0) {

            if (choice > this.modes.length || choice <= 0) {

                logger.error("Trying to choose unexisting mode!");

                return -1;

            } else {

                logger.info("Mode switched for Dishwasher");

                this.mode = choice-1;

                return 0;

            }

        } else {

            logger.error("Trying to change mode while cleaning!");

            System.out.println("Устройство в работе!");

            return -1;

        }

    }

    public void startCleaning(long currentTimeSeconds, Scanner systemIn) {

        if (this.loaded >= 0) {

            if (this.status == 0 && this.loaded > 0) {

                logger.info("CLEAN IMITATION STARTED");

                this.start = System.currentTimeMillis()/1000;

                this.status = 1;

                this.loaded -= 1;

            } else if (currentTimeSeconds - this.start < 10) {

                logger.info("CLEAN IMITATION SYNC");

                this.status = 2;

            } else if (currentTimeSeconds - this.start < 20) {

                logger.info("CLEAN IMITATION SYNC/FINISHED");

                this.status = 0;

                this.start = 0;

            } else {

                logger.error("Trying to start clean imitation without resources");

                cleanScreen();

                System.out.println("Нет ресурсов для начала мойки!");

                pressEnterToContinue(systemIn);

            }

        }

    }

    public long getStart() {

        return start;

    }

    public int getLoaded() {

        return loaded;

    }

    public int getMode() {

        return mode;

    }

    public int getStatus() {

        return status;

    }

    public String[] getStatuses() {

        return statuses;

    }

    public String[] getModes() {

        return modes;

    }

}

## **Washer.java**

package org.PrescentV;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonTypeInfo;

import com.fasterxml.jackson.annotation.JsonTypeName;

import java.util.Random;

import java.util.Scanner;

@JsonTypeInfo(use=JsonTypeInfo.Id.NAME, include=JsonTypeInfo.As.PROPERTY, property="type", visible = true)

@JsonTypeName("type")

@JsonIgnoreProperties(ignoreUnknown = true)

public class Washer extends Appliances {

    long start;

    int loaded;

    int mode;

    int status;

    String[] statuses = {"Ожидает","Моет","Отжимает"};

    String[] modes = {"Смешанная","Детская","Спортивная одежда","Спортивная обувь","Быстрая"};

    public Washer() {

    }

    public Washer(String modelName, String brandName) {

        super(modelName, brandName);

        Random rand = new Random();

        this.loaded = rand.nextInt(3);

        this.status = 0;

        logger.info("Washer device connected");

    }

    public void outputCurrentStatus() {

        if (this.oi == 1) {

            System.out.println("Устройство включено.");

            System.out.println("Текущий статус: " + statuses[this.status]);

            outputCurrentLoad();

        } else {

            System.out.println("Устройство выключено.");

        }

        logger.info("CurrentStatus showed for Washer");

    }

    public void outputCurrentMode() {

        System.out.println("Текущий режим: " + modes[this.mode]);

        logger.info("CurrentMode showed for Washer");

    }

    public void outputCurrentLoad() {

        System.out.println("Текущее количество ресурсов: " + this.loaded);

        logger.info("CurrentLoad showed for Washer");

    }

    public void outputAllModes() {

        for (int i = 0; i < modes.length; i++) {

            System.out.println(i+1 + ". " + modes[i]);

        }

        logger.info("AllModes showed for Washer");

    }

    public int changeMode(int choice) {

        if (this.status == 0) {

            if (choice > this.modes.length || choice <= 0) {

                logger.error("Trying to choose unexisting mode!");

                return -1;

            } else {

                logger.info("Mode switched for Washer");

                this.mode = choice-1;

                return 0;

            }

        } else {

            logger.error("Trying to change mode while cleaning!");

            System.out.println("Устройство в работе!");

            return -1;

        }

    }

    public void startCleaning(long currentTimeSeconds, Scanner systemIn) {

        if (this.loaded >= 0) {

            if (this.status == 0 && this.loaded > 0) {

                logger.info("CLEAN IMITATION STARTED");

                this.start = System.currentTimeMillis()/1000;

                this.status = 1;

                this.loaded -= 1;

            } else if (currentTimeSeconds - this.start < 10) {

                logger.info("CLEAN IMITATION SYNC");

                this.status = 2;

            } else if (currentTimeSeconds - this.start < 20) {

                logger.info("CLEAN IMITATION SYNC/FINISHED");

                this.status = 0;

                this.start = 0;

            } else {

                logger.error("Trying to start clean imitation without resources");

                cleanScreen();

                System.out.println("Нет ресурсов для начала стирки!");

                pressEnterToContinue(systemIn);

            }

        }

    }

    public long getStart() {

        return start;

    }

    public int getLoaded() {

        return loaded;

    }

    public int getMode() {

        return mode;

    }

    public int getStatus() {

        return status;

    }

    public String[] getStatuses() {

        return statuses;

    }

    public String[] getModes() {

        return modes;

    }

}

## **Freezer.java**

package org.PrescentV;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonTypeInfo;

import com.fasterxml.jackson.annotation.JsonTypeName;

import java.util.Random;

import java.util.Scanner;

@JsonTypeInfo(use=JsonTypeInfo.Id.NAME, include=JsonTypeInfo.As.PROPERTY, property="type", visible = true)

@JsonTypeName("type")

@JsonIgnoreProperties(ignoreUnknown = true)

public class Freezer extends Appliances {

    int loaded;

    int mode;

    int status;

    String[] statuses = {"Охлаждает","Замораживает","Размораживает"};

    String[] modes = {"Обычный","Заморозка","Разморозка"};

    public Freezer() {

    }

    public Freezer(String modelName, String brandName) {

        super(modelName, brandName);

        Random rand = new Random();

        this.loaded = rand.nextInt(20);

        logger.info("Freezer device connected");

    }

    public void outputCurrentStatus() {

        if (this.oi == 1) {

            System.out.println("Устройство включено.");

            System.out.println("Текущий статус: " + statuses[this.status]);

            outputCurrentLoad();

        } else {

            System.out.println("Устройство выключено.");

        }

        logger.info("CurrentStatus showed for Freezer");

    }

    public void outputCurrentMode() {

        System.out.println("Текущий режим: " + modes[this.mode]);

        logger.info("CurrentMode showed for Freezer");

    }

    public void outputCurrentLoad() {

        System.out.println("Текущее количество ресурсов: " + this.loaded);

        logger.info("CurrentLoad showed for Freezer");

    }

    public void outputAllModes() {

        for (int i = 0; i < modes.length; i++) {

            System.out.println(i+1 + ". " + modes[i]);

        }

        logger.info("AllModes showed for Freezer");

    }

    public int takeProducts(int amount) {

        if (amount < 0) {

            System.out.println("ERROR. Negative amount");

            return 2;

        }

        if (amount > this.loaded) {

            logger.warn("Tried to take more products than Freezer have");

            this.loaded = 0;

            return 1;

        } else {

            logger.info("Product taken from Freezer");

            this.loaded -= amount;

            return 0;

        }

    }

    public int giveProducts(int amount, Scanner systemIn) {

        if (amount < 0) {

            logger.error("Negative amount of products!");

            cleanScreen();

            System.out.println("Ошибка ввода!");

            pressEnterToContinue(systemIn);

            return -1;

        } else {

            if (this.loaded+amount > 50) {

                logger.error("Freezer limit reached!");

                System.out.println("Лимит продуктов достигнут");

                this.loaded = 50;

                return 1;

            }

            this.loaded += amount;

            return 0;

        }

    }

    public int changeMode(int choice) {

        if (choice > this.modes.length || choice <= 0) {

            logger.error("Trying to choose unexisting mode!");

            return -1;

        } else {

            logger.info("Mode switched for Freezer");

            this.mode = choice - 1;

            this.status = choice - 1;

            return 0;

        }

    }

    public int getLoaded() {

        return loaded;

    }

    public int getMode() {

        return mode;

    }

    public int getStatus() {

        return status;

    }

    public String[] getStatuses() {

        return statuses;

    }

    public String[] getModes() {

        return modes;

    }

}

# **Выводы**

В результате выполнения лабораторной работы была написана программа на языке Java, которая включает в себя несколько задач; изучена структура csv и json файлов; изучены такие коллекции как HashSet и HashMap.