



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.04.01/07 Интеллектуальные системы анализа,
обработки и интерпретации больших данных

О Т Ч Е Т

по лабораторной работе № 4

Название: Арифметические операции

Дисциплина: Языки программирования для работы с большими
данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

А.А. Аветисян

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

(И.О. Фамилия)

Москва, 2022

Лабораторная работа № 4

Задание:

Создать класс CD (mp3-диск) с внутренним классом, с помощью объектов которого можно хранить информацию о каталогах, подкаталогах и записях.

Ход работы:

Код программы:

```
package com.company.Lab4;

import java.util.Objects;

public class Element {
    private String name;
    private String type;

    public Element(String name){
        this.name = name;
    }

    public void setType(String type) {
        if ((type=="Catalog") || (type=="Track")){
            this.type = type;
        } else{
            System.out.println("Ошибка, неверный тип элемента");
        }
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public boolean isTrack(){
        return Objects.equals(type, "Track");
    }

    public boolean isCatalog(){
        return Objects.equals(type, "Catalog");
    }

    @Override
    public String toString() {
        return "Element{" +
            "name=" + name + "\" +
            ", type=" + type + "\" +
            '";
    }
}
```

```

package com.company.Lab4;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class CD {
    private String name;

    public CD(String name){
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public static class Component{
        private HashMap<Element, ArrayList<Element>> catalogHashMap;
        private ArrayList<Element> rootCatalogs;

        public Component() {
            catalogHashMap = new HashMap<>();
            rootCatalogs = new ArrayList<>();
        }

        public void addCatalog(Element catalog){
            if (catalog.isCatalog()){
                rootCatalogs.add(catalog);
            } else {
                System.out.println("Type error");
            }
        }

        public void addCatalog(Element catalog, Element directory){
            if (catalog.isCatalog() && directory.isCatalog()){
                if (catalogHashMap.get(directory)==null){
                    ArrayList<Element> buf = new ArrayList<>();
                    buf.add(catalog);
                    catalogHashMap.put(directory, buf);
                } else{
                    catalogHashMap.get(directory).add(catalog);
                }
            } else{
                System.out.println("Type error");
            }
        }

        public void addTrack(Element track, Element catalog){
            if (catalog.isCatalog() && track.isTrack()){
                if (catalogHashMap.get(catalog)==null){
                    ArrayList<Element> buf = new ArrayList<>();
                    buf.add(track);
                    catalogHashMap.put(catalog, buf);
                } else{
                    catalogHashMap.get(catalog).add(track);
                }
            } else {
                System.out.println("Type error");
            }
        }
    }
}

```

```

    }
}

@Override
public String toString() {
    return "Component{" +
        "trackHashMap=" + trackHashMap +
        ", catalogHashMap=" + catalogHashMap +
        ", rootCatalogs=" + rootCatalogs +
        '}';
}

@Override
public String toString() {
    return "CD{" +
        "name=" + name + "\" +
        '}';
}
}

public static void main(String[] args) {
    CD cd = new CD("Disk");
    CD.Component component = new CD.Component();

    Element root = new Element("ROOT");
    root.setType("Catalog");
    Element first = new Element("first");
    first.setType("Catalog");
    Element second = new Element("second");
    second.setType("Catalog");
    Element track_1 = new Element("track_1");
    track_1.setType("Track");
    Element track_2 = new Element("track_2");
    track_2.setType("Track");
    Element track_3 = new Element("track_3");
    track_3.setType("Track");

    component.addCatalog(root);
    component.addCatalog(first, root);
    component.addCatalog(second, root);
    component.addTrack(track_1, first);
    component.addTrack(track_2, first);
    component.addTrack(track_3, second);

    System.out.println(component);
}

```

Задание:

Создать класс Mobile с внутренним классом, с помощью объектов которого можно хранить информацию о моделях телефонов и их свойствах.

Ход работы:

Код программы:

```
package com.company.Lab4;

import java.util.ArrayList;
import java.util.HashMap;

public class Mobile {
    private String brand;
    private ArrayList<Model> models;

    public Mobile(String brand){
        this.brand = brand;
        this.models = new ArrayList<>();
    }

    public void addModel(Model model){
        this.models.add(model);
    }

    @Override
    public String toString() {
        return "Mobile{" +
            "brand='" + brand + '\'' +
            ", models=" + models +
            '}';
    }

    public static class Model{
        private String model;
        private HashMap<String, String> params;

        public Model(String model){
            this.model = model;
            this.params = new HashMap<>();
        }

        public void addParam(String param_name, String value){
            this.params.put(param_name, value);
        }

        @Override
        public String toString() {
            return "Model{" +
                "model='" + model + '\'' +
                ", params=" + params +
                '}';
        }
    }
}
```

```

public static void main(String[] args){
    Mobile mobile = new Mobile("iphone");
    Mobile.Model model = new Mobile.Model("5s");

    model.addParam("Память", "32гб");

    System.out.println(model);
}

```

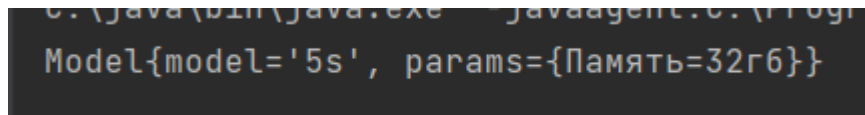


Рисунок 1 – Результат работы программы

Задание:

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов. 2. interface Абитуриент <- abstract class Студент <- class Студент-Заочник.

Ход работы:

Код программы:

```

interface Abiturient{
    String getName();
    void rateExam(String exam, String result);
}

public abstract class Student implements Abiturient {
    private String name;
    private HashMap<String, String> results;

    public Student(String name){
        this.name = name;
        this.results = new HashMap<>();
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public void rateExam(String exam, String result) {
        this.results.put(exam, result);
    }

    @Override
    public String toString() {
        return "Student{" +
            "name=" + name + "\" +
            ", results=" + results +
            '}'';
    }
}

```

```

public class Zaochnik extends Student{
    private String type;

    public Zaochnik(String name, String type){
        super(name);
        this.type = type;
    }

    @Override
    public String getName() {
        return super.getName() + " " + this.type;
    }

    @Override
    public String toString() {
        return "Zaochnik{" + super.toString() +
            "type=" + type + '\n' +
            '}';
    }
}

public static void main(String[] args){
    Zaochnik zaochnik = new Zaochnik("Антон Владимирович Путов", "Заочник");
    zaochnik.rateExam("Математика", "Отлично");
    zaochnik.rateExam("Физика", "Хорошо");
    zaochnik.rateExam("Программирование", "Хорошо");

    System.out.println(zaochnik);
}

```

```
Zaochnik{Student{name='Антон Владимирович Путов', results={Физика=Хорошо, Программирование=Хорошо, Математика=Отлично}};type='Заочник'}
```

Рисунок 2 – Результат работы программы

Задание:

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов. 3. interface Сотрудник <- class Инженер <- class Руководитель.

Ход работы:

Код программы:

```

public interface Employee {
    void getInfo();
}

public class Engineer implements Employee{
    private String name;
    private String org_name;
    private int salary;

    public Engineer(String name, String org_name, int salary){
        this.name = name;
        this.org_name = org_name;
        this.salary = salary;
    }
}

```

```

@Override
public void getInfo() {
    System.out.println(this);
}

@Override
public String toString() {
    return "Engineer{" +
        "name=" + name + "\" +
        ", org_name=" + org_name + "\" +
        ", salary=" + salary +
        '}'";
}
}

public class Supervisor extends Engineer{
    private int sub_number;

    public Supervisor(String name, String org_name, int salary, int sub_number){
        super(name, org_name, salary);
        this.sub_number = sub_number;
    }

    @Override
    public void getInfo() {
        System.out.println(this);
    }

    @Override
    public String toString() {
        return "Supervisor{" + super.toString() +
            "sub_number=" + sub_number +
            '}'";
    }
}

```

```
Supervisor{Engineer{name='Антон', org_name='Заправка', salary=34000}sub_number=12}
```

Рисунок 3 – Результат работы программы

Вывод: лабораторная работа выполнена в соответствии с заданием и вариантом.