

МАГИСТЕРСКАЯ ПРОГРАММА 09.04.01/07 Интеллектуальные системы анализа,
обработки и интерпретации больших данных.

по лабораторной работе №4

Дисциплина: Языки программирования для работы с большими данными.

(И.О. Фамилия)

Москва, 2022

Лабораторная работа №4

Задание: Вариант 1

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

Ход работы: Код программы файла Mobile

```
import java.util.ArrayList;

public class Mobile {
    String firm;
    ArrayList<Model> models;

    public Mobile() {
        models = new ArrayList<>();
    }

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

    public void addModel(String name){
        models.add(new Model(name));
    }

    @Override
    public String toString() {
        return "Mobile \n{" +
            "Фирма: " + firm + "\" +
            "\nмодель: \n" + models +
            '}'';
    }

    class Model{
        String name;
        ArrayList<Attribute> attributes;

        public Model() {
            attributes = new ArrayList<>();
        }

        public Model(String name) {
            this.name = name;
            attributes = new ArrayList<>();
        }

        public void addAttribute(String name, int amount){
            Attribute attribute = new Attribute(name, amount);
            attributes.add(attribute);
        }
    }
}
```

```

public void addAttribute(String name){
    Attribute attribute = new Attribute(name);
    attributes.add(attribute);
}

@Override
public String toString() {
    return "\n Модель{" +
        "Название=" + name + "\" +
        "\n    Атрибуты=" + attributes + "\n' +
        " }";
}

class Attribute{
    String name;
    int amount;

    public Attribute() {
    }

    public Attribute(String name) {
        this.name = name;
        this.amount = -1;
    }

    public Attribute(String name, int amount) {
        this.name = name;
        this.amount = amount;
    }

    public String getName() {
        return name;
    }

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

    public int getAmount() {
        return amount;
    }

    public void setAmount(int amount) {
        this.amount = amount;
    }

    @Override
    public String toString() {
        if(amount != -1) {
            return "атрибут{" +
                "Название=" + name + "\" +

```

```
        ", сумма=" + amount +  
    }';  
    } else {  
        return "атрибут{" +  
            "Название=" + name +  
            '}';  
    }  
}  
  
}  
  
}
```

Код программы файла MainForMobile

```
public class MainForMobile {
    public static void main(String[] args) {

        Mobile mobile = new Mobile("Xiaomi");

        mobile.addModel("Mi 11");
        mobile.models.get(0).addAttribute("RAM", 8);
        mobile.models.get(0).addAttribute("ROM", 256);
        mobile.models.get(0).addAttribute("Accumulator", 4600);
        mobile.models.get(0).addAttribute("NFC");

        mobile.addModel("Mi 8");
        mobile.models.get(1).addAttribute("RAM", 6);
        mobile.models.get(1).addAttribute("ROM", 128);
        mobile.models.get(1).addAttribute("Accumulator", 3400);
        mobile.models.get(1).addAttribute("NFC");

        System.out.println(mobile);

    }
}
```

```
Mobile
{Фирма: 'Xiaomi'
модель:
[
  Модель{Название='Mi 11'
    Атрибуты=[атрибут{Название='RAM', сумма=8}, атрибут{Название='ROM', сумма=256}, атрибут{Название='Accumulator', сумма=4600}, атрибут{Название='NFC'}
  ],
  Модель{Название='Mi 8'
    Атрибуты=[атрибут{Название='RAM', сумма=6}, атрибут{Название='ROM', сумма=128}, атрибут{Название='Accumulator', сумма=3400}, атрибут{Название='NFC'}
  ]
}}
```

Рисунок 1. Результат работы программы

Задание: Вариант 1

4. Создать класс Художественная Выставка с внутренним классом, с помощью объектов которого можно хранить информацию о картинах, авторах и времени проведения выставок.

Ход работы: Код программы файла Exhibition

```
import java.util.ArrayList;

public class Exhibition {
    String title;
    ArrayList<Picture> picture;

    public Exhibition() {
        picture = new ArrayList<>();
    }

    public Exhibition(String title) {
        this.title = title;
        picture = new ArrayList<>();
    }

    public void addPicture(String name){
        picture.add(new Picture(name));
    }

    @Override
    public String toString() {
        return "Художественная выставка \n{" +
            "Название: '" + title + '\'' +
            "\nГалерея \n" + picture +
            '}'';
    }

    class Picture{
        String name;
        ArrayList<Attribute> attributes;

        public Picture() {
            attributes = new ArrayList<>();
        }

        public Picture(String name) {
            this.name = name;
            attributes = new ArrayList<>();
        }

        public void addAttribute(String name, int amount){
            Attribute attribute = new Attribute(name, amount);
            attributes.add(attribute);
        }

        public void addAttribute(String name){
            Attribute attribute = new Attribute(name);
            attributes.add(attribute);
        }

        @Override
        public String toString() {
            return "\n    Художник{" +
```

```

        "\"" + name + '\'' +
        "\n        Какртины=" + attributes + " }";
    }

    class Attribute{
        String name;
        int amount;

        public Attribute() {
        }

        public Attribute(String name) {
            this.name = name;
            this.amount = -1;
        }

        public Attribute(String name, int amount) {
            this.name = name;
            this.amount = amount;
        }

        public String getName() {
            return name;
        }

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

        public int getAmount() {
            return amount;
        }

        public void setAmount(int amount) {
            this.amount = amount;
        }

        @Override
        public String toString() {
            if(amount != -1) {
                return "{" +
                    "Название:" + name + '\'' +
                    ", Время показа=" + amount +
                    '}';
            } else {
                return "Зал № " + name;
            }
        }
    }
}

```

Код программы файла Exhibition

```
public class MainForExhibition {
    public static void main(String[] args) {

        Exhibition exhibition = new Exhibition("Искусство в нас ");

        exhibition.addPicture("Винсент Ван Гог");
        exhibition.picture.get(0).addAttribute("Звездная ночь", 12);
        exhibition.picture.get(0).addAttribute("Подсолнухи", 13);
        exhibition.picture.get(0).addAttribute("Ирисы", 14);
        exhibition.picture.get(0).addAttribute("10");

        exhibition.addPicture("Сальвадор Дали");
        exhibition.picture.get(1).addAttribute("Постоянство памяти", 15);
        exhibition.picture.get(1).addAttribute("Жираф в огне", 16);
        exhibition.picture.get(1).addAttribute("Метаморфозы Нарцисса", 17);
        exhibition.picture.get(1).addAttribute("15");

        System.out.println(exhibition);

    }
}
```

```
Художественная выставка
{Название: 'Искусство в нас '
Галерея
[
    Художник{'Винсент Ван Гог'
        Картины={Название:'Звездная ночь', Время показа=12}, {Название:'Подсолнухи', Время показа=13}, {Название:'Ирисы', Время показа=14}, Зал № 10} },
    Художник{'Сальвадор Дали'
        Картины={Название:'Постоянство памяти', Время показа=15}, {Название:'Жираф в огне', Время показа=16}, {Название:'Метаморфозы Нарцисса', Время показа=17}, Зал № 15} ]}]
```

Рисунок 2. Результат работы программы

Задание: Вариант 2. Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

3. interface Сотрудник <- class Инженер <- class Руководитель.

Ход работы: Код программы файла Employee

```
public interface Employee {
    String introduce();
}
```

Код программы файла Engineer

```
public class Engineer implements Employee{
    private String name;
    private String position;
    private String organization;
    private String department;

    public Engineer() {
```

```

    }

    public Engineer(String name, String position, String organization, String
department) {
        this.name = name;
        this.position = position;
        this.organization = organization;
        this.department = department;
    }

    @Override
    public String introduce() {
        return "Форма служебного письма\nЗдравствуйте, \nменя зовут " + name
+ ",\nЯ являюсь " + position + " в " + organization + " / " + department + "
департамент";
    }

    public String getName() {
        return name;
    }

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

    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
    }

    public String getOrganization() {
        return organization;
    }

    public void setOrganization(String organization) {
        this.organization = organization;
    }

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }

    @Override
    public String toString() {
        return "Engineer{" +
            "name='" + name + '\'' +
            ", position='" + position + '\'' +
            ", organization='" + organization + '\'' +
            ", department='" + department + '\'' +
            '}';
    }
}

```


Код программы файла Head

```
public class Head extends Engineer{

    public Head() {
    }

    public Head(String name, String position, String organization, String
department) {
        super(name, position, organization, department);
    }

    @Override
    public String introduce() {
        return super.introduce() + "\nЯ возглавляю этот отдел";
    }

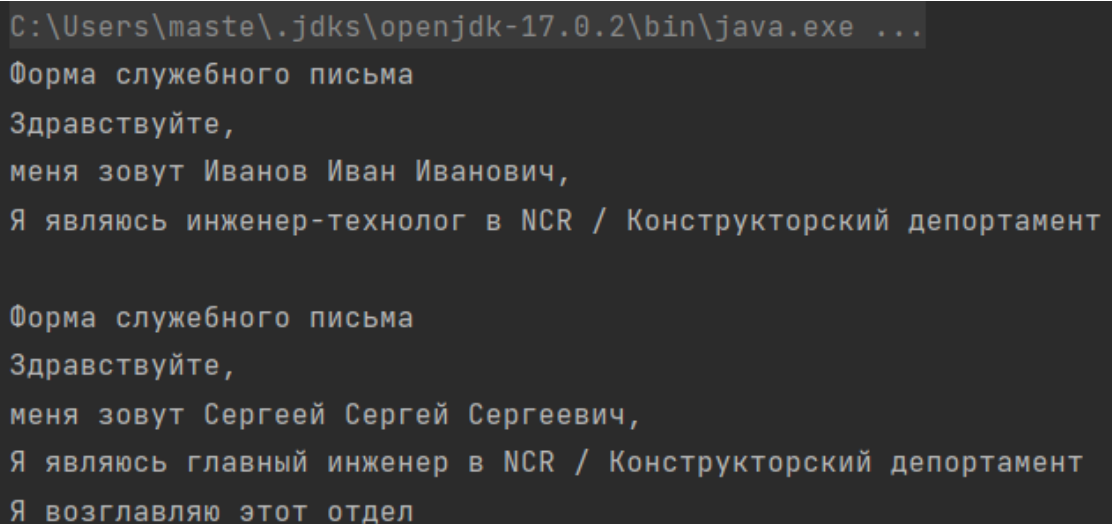
}
```

Код программы файла MainForEngineer

```
public class MainForEngineer {
    public static void main(String[] args) {

        Engineer engineer = new Engineer("Иванов Иван Иванович", "инженер-
технолог", "NCR", "Конструкторский");
        Head head = new Head("Сергеей Сергей Сергеевич", "главный инженер",
"NCR", "Конструкторский");

        System.out.println(engineer.introduce());
        System.out.println();
        System.out.println(head.introduce());
    }
}
```



```
C:\Users\maste\.jdk\openjdk-17.0.2\bin\java.exe ...
Форма служебного письма
Здравствуйте,
меня зовут Иванов Иван Иванович,
Я являюсь инженер-технолог в NCR / Конструкторский департамент

Форма служебного письма
Здравствуйте,
меня зовут Сергей Сергей Сергеевич,
Я являюсь главный инженер в NCR / Конструкторский департамент
Я возглавляю этот отдел
```

Рисунок 3. Результат работы программы

Задание: Вариант 2. Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

4. interface Здание <- abstract class Общественное Здание <- class Театр.

Ход работы: Код программы файла Building

```
public interface Building {  
    String enter();  
}
```

Код программы файла Public_Building

```
public class Public_Building implements Building{  
    private String name;  
    private String address;  
    private String website;  
    private String type_of_building;  
  
    public Public_Building() {  
    }  
  
    public Public_Building(String name, String address, String website,  
String type_of_building) {  
        this.name = name;  
        this.address = address;  
        this.website = website;  
        this.type_of_building = type_of_building;  
    }  
  
    @Override  
    public String enter() {  
        return "Описание здания\nНазвание: " + name + ";\nАдрес: " + address  
+";\nWeb-сайт: " + website + ";\nТип здания: " + type_of_building;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public String getAddress() {  
        return address;  
    }  
  
    public void setAddress(String address) {  
        this.address = address;  
    }  
  
    public String getWebsite() {  
        return website;  
    }  
  
    public void setWebsite(String website) {  
        this.website = website;  
    }  
}
```

```

public String getType_of_building() {
    return type_of_building;
}

public void setType_of_building(String type_of_building) {
    this.type_of_building = type_of_building;
}

@Override
public String toString() {
    return "Public_Building{" +
        "name='" + name + '\'' +
        ", address='" + address + '\'' +
        ", website='" + website + '\'' +
        ", type_of_building='" + type_of_building + '\'' +
        '}';
}
}

```

Код программы файла Theatre

```

public class Theatre extends Public_Building{

    public Theatre() {
    }

    public Theatre(String name, String address, String website, String
type_of_building) {
        super(name, address, website, type_of_building);
    }

    @Override
    public String enter() {
        return super.enter() + "\nГосударственный драматический театр";
    }
}

```

Код программы файла Theatre

```

public class MainForPublic_Building {
    public static void main(String[] args) {

        Public_Building public_building = new Public_Building("Планетарий",
"Москва, ул.Садовая-Кудринская, д. 5, стр. 1", "planetarium-moscow.ru",
"Общественное здание");
        Theatre theatre = new Theatre("Современник", "Москва, Чистопрудный
бульвар, 19, стр. 1", "sovremennik.ru", "Общественное здание");

        System.out.println(public_building.enter());
        System.out.println();
        System.out.println(theatre.enter());
    }
}

```

```
C:\Users\maste\.jdk\openjdk-17.0.2\bin\java.exe ...  
Описание здания  
Название: Планетарий;  
Адрес: Москва, ул.Садовая-Кудринская, д. 5, стр. 1;  
Web-сайт: planetarium-moscow.ru;  
Тип здания: Общественное здание  
  
Описание здания  
Название: Современник;  
Адрес: Москва, Чистопрудный бульвар, 19, стр. 1;  
Web-сайт: sovremennik.ru;  
Тип здания: Общественное здание  
Государственный драматический театр
```

Рисунок 4. Результат работы программы

Вывод: лабораторная работа была выполнена в соответствии с заданием и полученные верные результаты работ программ