



**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 Информатика и вычислительная техника

**МАГИСТЕРСКАЯ ПРОГРАММА 09.04.01/12 Интеллектуальный анализ больших
данных в системах поддержки принятия решений.**

О Т Ч Е Т

по лабораторной работе № 4

Вариант № 5

Название: Внутренние классы и интерфейсы

Дисциплина: Языки программирования для работы с большими данными

Студент

ИУ6-23М
(Группа)

(Подпись, дата)

А.О.Крейденко
(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов
(И.О. Фамилия)

Москва, 2024

Цель: изучить работу внутренних классов и интерфейсов в java.

Задание 1: создать класс Календарь с внутренним классом, с помощью объектов которого можно хранить информацию о выходных и праздничных днях.

Код класса Calendar:

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

public class Calendar {
    private List<Holiday> holidays_info;

    public Calendar(HashMap<String, String> info) {
        this.holidays_info = new ArrayList<Holiday>();
        for(Map.Entry<String, String> holiday: info.entrySet()) {
            this.holidays_info.add(new Holiday(holiday.getKey(),
holiday.getValue()));
        }
    }

    public List<Holiday> getHolidays_info() {
        return holidays_info;
    }

    public void showHolidays() {
        for (Holiday h: this.holidays_info) {
            System.out.println(h.getDate() + ": " + h.getTitle());
        }
    }

    static class Holiday {
        private String title;
        private String date;

        public Holiday(String title, String date) {
            this.title = title;
            this.date = date;
        }

        public String getTitle() {
            return title;
        }

        public void setTitle(String title) {
            this.title = title;
        }

        public String getDate() {
```

```

        return date;
    }

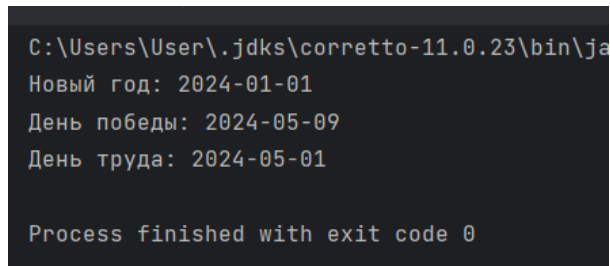
    public void setDate(String date) {
        this.date = date;
    }
}

public static void main(String[] args) {
    HashMap<String, String> holidays = new HashMap<>();
    holidays.put("2024-01-01", "Новый год");
    holidays.put("2024-05-01", "День труда");
    holidays.put("2024-05-09", "День победы");

    Calendar calendar = new Calendar(holidays);
    calendar.showHolidays();
}
}

```

Работа программы показана на рисунке 1.



The screenshot shows a terminal window with the following output:

```

C:\Users\User\.jdk\corretto-11.0.23\bin\java
Новый год: 2024-01-01
День победы: 2024-05-09
День труда: 2024-05-01

Process finished with exit code 0

```

Рисунок 1 – Работа программы 1

Задание 2: создать класс Shop (магазин) с внутренним классом, с помощью объектов которого можно хранить информацию об отделах, товарах и услуг.

Код класса Shop:

```

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

public class Shop {

    private List<SectionInfo> sections_info;

    public Shop() {
        this.sections_info = new ArrayList<>();
    }

    public void addSection(String name, List<String> products,
List<String> services) {

```

```

        SectionInfo new_section = new SectionInfo(name, products,
services);
        sections_info.add(new_section);
    }

    public void showSections() {
        for (SectionInfo s: this.sections_info) {
            System.out.println("Отдел: " + s.getSection_name() +
", продукты: " +
s.getProducts() + ", услуги: " +
s.getServices());
        }
    }
    static class SectionInfo {
        private String section_name;
        private List<String> products;
        private List<String> services;

        public SectionInfo(String section_name, List<String>
products, List<String> services) {
            this.section_name = section_name;
            this.products = products;
            this.services = services;
        }

        public String getSection_name() {
            return section_name;
        }

        public void setSection_name(String section_name) {
            this.section_name = section_name;
        }

        public List<String> getProducts() {
            return products;
        }

        public void setProducts(List<String> products) {
            this.products = products;
        }

        public List<String> getServices() {
            return services;
        }

        public void setServices(List<String> services) {
            this.services = services;
        }
    }
    public static void main(String[] args) {
        String section_name = "Фрукты";

        List<String> products = new ArrayList<>();

```

```

        products.add("Апельсин");
        products.add("Банан");
        products.add("Яблоко");
        products.add("Апельсин");

        List<String> services = new ArrayList<>();
        services.add("Взвешивание");

        Shop shop = new Shop();
        shop.addSection(section_name, products, services);

        shop.showSections();
    }
}

```

Работа программы показана на рисунке 2.

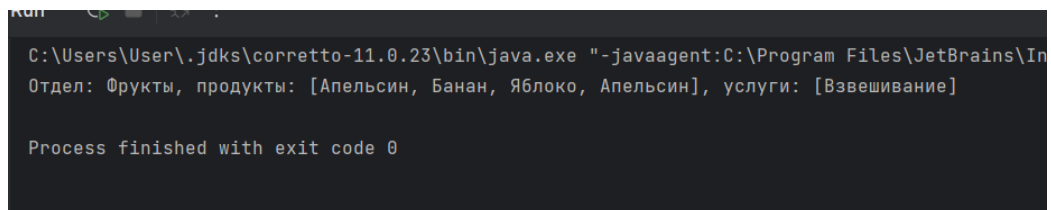


Рисунок 2 – Работа программы 2

Задание 3: реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов. interface Mobile <- abstract class Siemens Mobile <- class Model.

Код интерфейса Mobile:

```

public interface Mobile {
    // метод для осуществления звонка
    void call();
}

```

Код класса Model:

```

public class Model extends SiemensMobile{
    private String model;

    public Model(String model) {
        super();
        this.model = model;
    }

    public void show_model() {
        System.out.println("Модель телефона: " + this.model);
    }

    public String getModel() {

```

```

        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public static void main(String[] args) {
        Model mobile_phone = new Model("SIEMENS 1920 PRO
MAX");

        mobile_phone.call();
        mobile_phone.show_model();
        System.out.println(mobile_phone.getCompany_name());
    }
}

```

Код класса SiemensMobile:

```

abstract class SiemensMobile implements Mobile {
    protected String company_name;

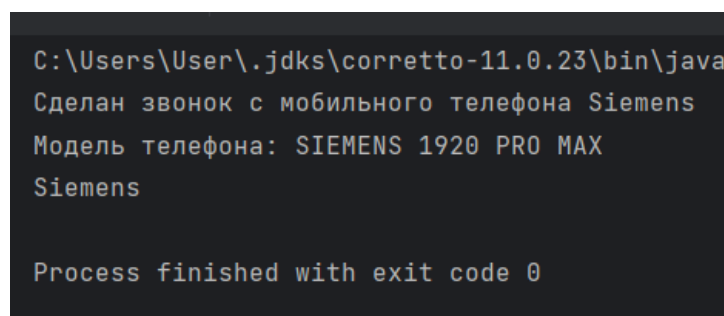
    public SiemensMobile() {
        this.company_name = "Siemens";
    }

    public String getCompany_name() {
        return company_name;
    }

    @Override
    public void call() {
        System.out.println("Сделан звонок с мобильного
телефона Siemens");
    }
}

```

Работа программы показана на рисунке 3.



```

C:\Users\User\.jdk\corretto-11.0.23\bin\java
Сделан звонок с мобильного телефона Siemens
Модель телефона: SIEMENS 1920 PRO MAX
Siemens

Process finished with exit code 0

```

Рисунок 3 – Работа программы 3

Задание 4: реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов. interface Корабль <- abstract class Военный Корабль <- class Авианосец.

Код интерфейса Ship:

```
public interface Ship {
    void sale();
    String getShipName();
    void setShipName(String new_name);
}
```

Код класса WarShip:

```
abstract class WarShip implements Ship{
    protected String ship_class;
    protected String name;

    public WarShip(String name) {
        this.name = name;
        this.ship_class = "Военный";
    }

    @Override
    public String getShipName() {
        return this.name;
    }

    @Override
    public void setShipName(String new_name) {
        this.name = new_name;
    }

    public String getShip_class() {
        return ship_class;
    }
}
```

Код класса Carrier:

```
public class Carrier extends WarShip{
    private String ship_type;

    public Carrier(String name) {
        super(name);
        this.ship_type = "Авианосец";
    }

    @Override
    public void sale() {
        System.out.println("Плывет авианосец");
    }
}
```

```

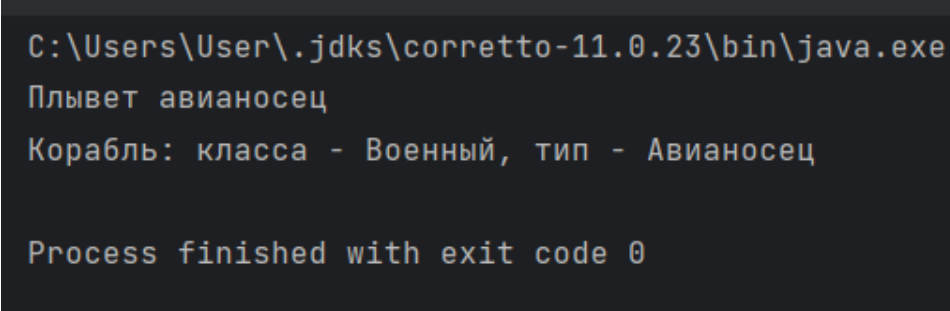
        public void show_ship_info() {
            System.out.println("Корабль:   класса   -   "   +
this.ship_class +
            ", тип - " + this.ship_type);
        }

        public String getShip_type() {
            return ship_type;
        }

        public static void main(String[] args) {
            Carrier carrier = new Carrier("Адмирал Кузнецов");
            carrier.sale();
            carrier.show_ship_info();
        }
    }
}

```

Работа программы показана на рисунке 4.



```

C:\Users\User\.jdk\corretto-11.0.23\bin\java.exe
Плывет авианосец
Корабль: класса - Военный, тип - Авианосец

Process finished with exit code 0

```

Рисунок 4 – Работа программы 4

Вывод: была изучена работа абстрактных классов и интерфейсов в java.