



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 № 4**

**Название:** внутренние классы и интерфейсы

**Дисциплина:** языки программирования для работы с большими данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

А.А.Клушина

(И.О. Фамилия)

Преподаватель

П.В. Степанов

(Подпись, дата)

(И.О. Фамилия)

Москва, 2024

**Цель:** изучить работу внутренних классов и интерфейсов в java.

**Задание 1:** Создать класс Художественная Выставка с внутренним классом, с помощью объектов которого можно хранить информацию о картинах, авторах и времени проведения выставок.

**Код класса ArtExhibition:**

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

public class ArtExhibition {
    private String exhibitionName;
    private String exhibitionDate;
    private List<Painting> paintings;

    public ArtExhibition(String exhibitionName, String
exhibitionDate) {
        this.exhibitionName = exhibitionName;
        this.exhibitionDate = exhibitionDate;
        this.paintings = new ArrayList<>();
    }

    public void addPainting(String title, String artist) {
        Painting newPainting = new Painting(title, artist);
        paintings.add(newPainting);
    }

    public void displayPaintings() {
        System.out.println("Картины на выставке " +
exhibitionName + " " + exhibitionDate + ":");
        for (Painting painting : paintings) {
            System.out.println("Название: " +
painting.getTitle() + " | Художник: " + painting.getArtist());
        }
    }

    public class Painting {
        private String title;
        private String artist;

        public Painting(String title, String artist) {
            this.title = title;
            this.artist = artist;
        }

        public String getTitle() {
            return title;
        }

        public String getArtist() {
```

```

        return artist;
    }
}

public static void main(String[] args) {
    ArtExhibition exhibition = new
    ArtExhibition("Современное искусство", "29 марта, 2024");
    exhibition.addPainting("Звездная ночь", "Ван Гог");
    exhibition.addPainting("Мона Лиза", "Леонардо да
    Винчи");
    exhibition.addPainting("Постоянство памяти", "Сальвадор
    Дали");

    exhibition.displayPaintings();
}
}

```

Работа программы показана на рисунке 1.

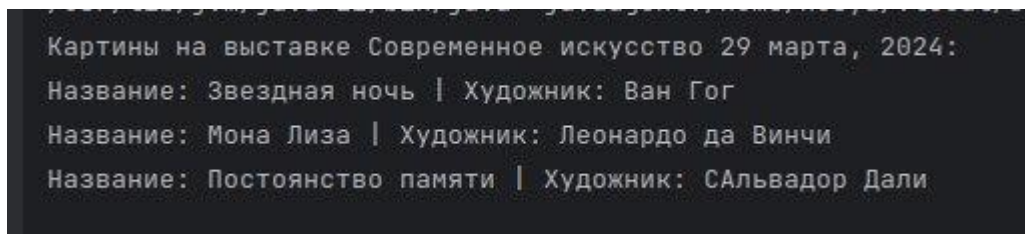


Рисунок 1 – Работа программы

**Задание 2:** 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 Map<String, String> holidays;

    public Calendar() {
        this.holidays = new HashMap<>();
    }

    public void addHoliday(String date, String name) {
        holidays.put(date, name);
    }
}

```

```

    }

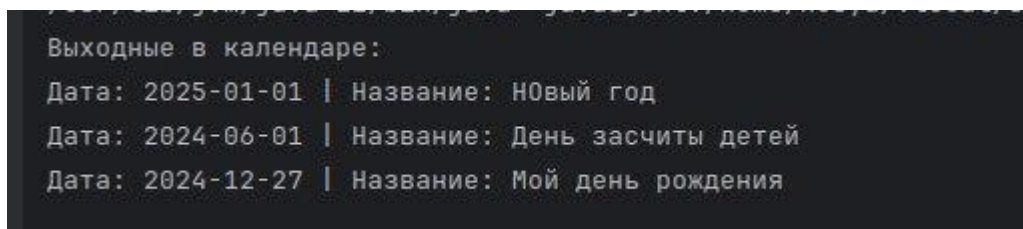
    public void displayHolidays() {
        System.out.println("Выходные в календаре:");
        for (Map.Entry<String, String> entry :
holidays.entrySet()) {
            System.out.println("Дата: " + entry.getKey() + "
| Название: " + entry.getValue());
        }
    }

    public static void main(String[] args) {
        Calendar calendar = new Calendar();
        calendar.addHoliday("2025-01-01", "НОВЫЙ год");
        calendar.addHoliday("2024-12-27", "Мой день
рождения");
        calendar.addHoliday("2024-06-01", "День засчиты
детей");

        calendar.displayHolidays();
    }
}

```

Работа программы показана на рисунке 2.



```

Выходные в календаре:
Дата: 2025-01-01 | Название: НОВЫЙ год
Дата: 2024-06-01 | Название: День засчиты детей
Дата: 2024-12-27 | Название: Мой день рождения

```

Рисунок 2 – Работа программы

**Задание 3:** Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

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

Код классов и интерфейсов:

```

// Интерфейс "Здание"
interface Building {
    void open(); // открыть здание
    void close(); // закрыть здание
}

// Абстрактный класс "Общественное Здание" реализует интерфейс "Здание"
abstract class PublicBuilding implements Building {

```

```

String buildingName;

public PublicBuilding(String buildingName) {
    this.buildingName = buildingName;
}

public abstract void operate();

@Override
public void open() {
    System.out.println(buildingName + " открыт.");
}

@Override
public void close() {
    System.out.println(buildingName + " закрыт.");
}
}

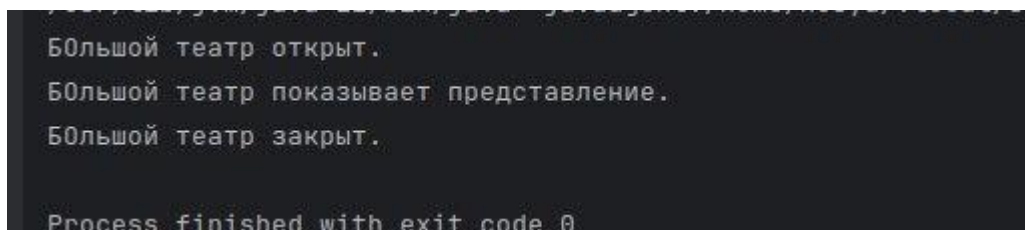
// Класс "Театр" наследует абстрактный класс "Общественное
Здание"
class Theater extends PublicBuilding {
    public Theater(String buildingName) {
        super(buildingName);
    }

    @Override
    public void operate() {
        System.out.println(buildingName + " показывает
представление.");
    }

    public static void main(String[] args) {
        Theater theater = new Theater("БОЛЬШОЙ театр");
        theater.open();
        theater.operate();
        theater.close();
    }
}

```

Работа программы показана на рисунке 3.



```

БОЛЬШОЙ театр открыт.
БОЛЬШОЙ театр показывает представление.
БОЛЬШОЙ театр закрыт.

Process finished with exit code 0

```

Рисунок 3 – Работа программы

**Задание 4:** Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

1. interface Здание <- abstract class Общественное Здание <- class Театр.
2. interface Mobile <- abstract class Siemens Mobile <- class Model.

Код классов и интерфейсов:

```
// Интерфейс "Mobile"
interface Mobile {
    void call(); // метод для совершения звонка
}

// Абстрактный класс "SiemensMobile" реализует интерфейс "Mobile"
abstract class SiemensMobile implements Mobile {
    protected String model;

    public SiemensMobile(String model) {
        this.model = model;
    }

    abstract void displayInfo(); // метод для отображения информации о модели

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

// Класс "Model" наследует абстрактный класс "SiemensMobile"
class Model extends SiemensMobile {
    public Model(String model) {
        super(model);
    }

    @Override
    void displayInfo() {
        System.out.println("Модель мобильного телефона Siemens: " + model);
    }

    public static void main(String[] args) {
        Model phone = new Model("S45");
        phone.displayInfo();
        phone.call();
    }
}
```

Работа программы показана на рисунке 4.

```
Модель мобильного телефона Siemens: S45  
Звонок с мобильного телефона Siemens.  
  
Process finished with exit code 0
```

Рисунок 4 – Работа программы

**Вывод:** была изучена работа внутренних классов и интерфейсов в java.