

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”  
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ  
ТЕХНОЛОГИЙ**

**Лабораторная работа №5**

**По дисциплине:** «Современные платформы программирования»

**Тема:** «Объектно-ориентированного проектирования в Java»

**Выполнил:**

Студент 3 курса

Группы ПО-8

Бувин Д.А.

**Проверил:**

А. А. Крощенко

**Брест, 2024**

## Лабораторная работа №5

### Вариант 3

**Цель работы:** приобрести практические навыки в области объектно-ориентированного проектирования в Java.

#### **Задание №1:**

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

interface Сотрудник ← class Инженер ← class Руководитель.

#### **Код программы:**

```
interface Employee {
    void work();
}

class Engineer implements Employee {
    @Override
    public void work() {
        System.out.println("Инженер работает");
    }
}

class Manager implements Employee {
    @Override
    public void work() {
        System.out.println("Руководитель работает");
    }
}

public class Task_1 {
    public static void main(String[] args) {
        Engineer engineer = new Engineer();
        Manager manager = new Manager();

        Employee emp1 = engineer;
        Employee emp2 = manager;

        emp1.work();
        emp2.work();
    }
}
```

## Результат работы:

```
Инженер работает
Руководитель работает

Process finished with exit code 0
```

## Задание №2:

В следующих заданиях требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру. Использовать объекты подклассов для моделирования реальных ситуаций и объектов.

Создать суперкласс Музыкальный инструмент и классы Ударный, Струнный, Духовой. Создать массив объектов Оркестр. Осуществить вывод состава оркестра.

## Код программы:

```
abstract class MusicalInstrument {
    private String name;

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

    public String getName() {
        return name;
    }

    public abstract void play();
}

class Percussion extends MusicalInstrument {
    public Percussion(String name) {
        super(name);
    }

    @Override
```

```

    public void play() {
        System.out.println(getName() + " играет ударными");
    }
}

class Stringed extends MusicalInstrument {
    public Stringed(String name) {
        super(name);
    }

    @Override
    public void play() {
        System.out.println(getName() + " играет струнными");
    }
}

class Wind extends MusicalInstrument {
    public Wind(String name) {
        super(name);
    }

    @Override
    public void play() {
        System.out.println(getName() + " играет духовыми");
    }
}

class Orchestra {
    private MusicalInstrument[] instruments;

    public Orchestra(int size) {
        instruments = new MusicalInstrument[size];
    }

    public void addInstrument(int index, MusicalInstrument instrument) {
        instruments[index] = instrument;
    }

    public void listInstruments() {
        System.out.println("Состав оркестра:");
        for (MusicalInstrument instrument : instruments) {
            if (instrument != null) {
                instrument.play();
            }
        }
    }
}

```

```

    }
}
}

public class Task_2 {
    public static void main(String[] args) {
        Percussion drums = new Percussion("Барабаны");
        Stringed violin = new Stringed("Скрипка");
        Wind trumpet = new Wind("Труба");

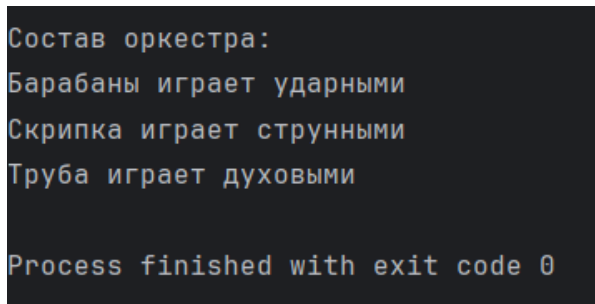
        Orchestra orchestra = new Orchestra(3);

        orchestra.addInstrument(0, drums);
        orchestra.addInstrument(1, violin);
        orchestra.addInstrument(2, trumpet);

        orchestra.listInstruments();
    }
}

```

### Результат работы:



```

Состав оркестра:
Барабаны играет ударными
Скрипка играет струнными
Труба играет духовыми

Process finished with exit code 0

```

### Задание №3:

В задании 3 ЛР №4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

Система Больница. Пациенту назначается лечащий Врач. Врач может сделать назначение Пациенту (процедуры, лекарства, операции). Медсестра или другой Врач выполняют назначение. Пациент может быть

выписан из Больницы по окончании лечения, при нарушении режима или иных обстоятельствах.

### **Код программы:**

```
abstract class Patient {
    protected String name;
    protected int age;
    protected String diagnosis;
    protected String condition;
    protected Doctor attendingDoctor;

    public Patient(String name, int age, String diagnosis) {
        this.name = name;
        this.age = age;
        this.diagnosis = diagnosis;
        this.condition = "In Hospital";
    }

    public void setAttendingDoctor(Doctor doctor) {
        this.attendingDoctor = doctor;
    }

    public String getName() {
        return name;
    }

    public abstract void changeCondition(String newCondition);
}

abstract class MedicalStaff {
    protected String name;
    protected String specialization;

    public MedicalStaff(String name, String specialization) {
        this.name = name;
        this.specialization = specialization;
    }
}

class Doctor extends MedicalStaff {
    public Doctor(String name, String specialization) {
        super(name, specialization);
    }
}
```

```

        public void prescribeTreatment(Patient patient, String treatment) {
            System.out.println(name + " prescribes " + treatment + " for patient " +
patient.getName());
        }
    }

class Nurse extends MedicalStaff {
    public Nurse(String name, String specialization) {
        super(name, specialization);
    }

    public void performTreatment(Patient patient, String treatment) {
        System.out.println(name + " performs " + treatment + " for patient " +
patient.getName());
    }
}

public class Task_3 {
    public static void main(String[] args) {
        Doctor doctor = new Doctor("Dr. Smith", "Cardiologist");
        Patient patient = new Patient("John Doe", 45, "Heart Disease") {
            @Override
            public void changeCondition(String newCondition) {
                this.condition = newCondition;
            }
        };
        patient.setAttendingDoctor(doctor);

        doctor.prescribeTreatment(patient, "Heart medication");

        Nurse nurse = new Nurse("Nurse Jane", "Registered Nurse");
        nurse.performTreatment(patient, "Heart medication");

        patient.changeCondition("Recovered");

        System.out.println("Patient " + patient.getName() + " is " + patient.condition);
    }
}

```

**Результат работы:**

```
Dr. Smith prescribes Heart medication for patient John Doe  
Nurse Jane performs Heart medication for patient John Doe  
Patient John Doe is Recovered|  
  
Process finished with exit code 0
```

**Вывод:** По итогу выполнения лабораторной работы, я приобрел практические навыки в области объектно-ориентированного проектирования в Java.