Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №5
по дисциплине: **СПП**
Тема: ООП в Java

Выполнил
студент 3 курса
Корнасевич И. Д.

Проверил
Крощенко А. А.

Брест 2021

Цель работы приобрести практические навыки в области объектно-ориентированного проектирования

Задание 1 Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

interface Сотрудник ← class Инженер ← class Руководитель.

Main.java

```java
package main5.task1;

import java.util.Arrays;


public class Main {

    public static void main(String[] args) {
        var m = new Manager();
        var e = new Engineer();
        var employees = Arrays.<Employee>asList(m, e);
        employees.forEach(Employee::work);
    }
}
```

Employee.java

```java
package main5.task1;

public interface Employee {

    void work();
}
```

Engineer.java

```java
package main5.task1;

public class Engineer implements Employee {

    @Override
    public void work() {
        System.out.println("Engineer works");
    }
}
```

Manager.java

```java
package main5.task1;

public class Manager extends Engineer {

    @Override
    public void work() {
        System.out.println("Manager works");
    }
}
```

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. Со- здать массив объектов Оркестр. Осуществить вывод состава оркестра.

Main.java

```java
package main5.task2;

public class Main {

    public static void main(String[] args) {
        var orchestra = new MusicalInstrument[]{new PercussionInstrument(), new StringedInstrument()
    , new WindInstrument()};
        for (var instrument : orchestra) {
            instrument.play();
        }
    }
}
```

MusicalInstrument.java

```java
package main5.task2;

public abstract class MusicalInstrument {

    public void play() {
        System.out.println("Generic instrument plays");
    }
}
```

PercussionInstrument.java

```java
package main5.task2;

public class PercussionInstrument extends MusicalInstrument {

    @Override
    public void play() {
        System.out.println("Percussion instrument plays");
    }
}
```

StringedInstrument.java

```java
package main5.task2;

public class StringedInstrument extends MusicalInstrument {

    @Override
    public void play() {
        System.out.println("Stringed instrument plays");
    }
}
```

WindInstrument.java

```java
package main5.task2;

public class WindInstrument extends MusicalInstrument {

    @Override
    public void play() {
        System.out.println("Wind instrument plays");
    }
}
```

Задание 3 В задании 3 ЛР No4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

Curable.java

```java
package main5.task3;

public interface Curable {

    void prescribeTreatment(Treatment treatment);

    void cure();

    Status getStatus();
}
```

Doctor.java

```java
package main5.task3;

import java.util.*;


public class Doctor extends Medic {

    public final Set<Curable> patients;

    public Doctor() {
        this(new HashSet<>());
    }

    public Doctor(Set<Curable> patients) {
        this.patients = patients;
    }

    public void prescribeTreatment(Curable patient) {
        var treatments = Treatment.values();
        var treatment = treatments[new Random().nextInt(treatments.length)];
        patient.prescribeTreatment(treatment);
        patients.add(patient);
    }
}
```

Patient.java

```java
package main5.task3;

public class Patient implements Curable {

    private Treatment treatment;

    private Status status;

    public Patient() {
        this(Treatment.PILLS, Status.SICK);
    }

    public Patient(Treatment treatment, Status status) {
        this.treatment = treatment;
        this.status = status;
    }

    @Override
    public void prescribeTreatment(Treatment treatment) {
        this.treatment = treatment;
    }

    @Override
    public void cure() {
        status = Status.HEALTHY;
    }

    @Override
    public Status getStatus() {
        return status;
    }

    public void violateRules() {
        status = Status.RULES_VIOLATED;
    }
}
```

Вывод Я смоделировал системы реального мира используя ООП.