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

Лабораторная работа №4
по дисциплине: **СПП**
Тема: Отношения классов в Java

Выполнил
студент 3 курса
Корнасевич И. Д.

Проверил
Крощенко А. А.

Брест 2021

Цель работы приобрести практические навыки в области ООП

Задание 1 Создать класс Account (счет) с внутренним классом, с помощью объектов которого можно хранить информацию обо всех операциях со счетом (снятие, платежи, поступления).

App1.java

```java
package main4.task1;

public class App1 {

    public static void main(String[] args) {
        var zero = new Account(1000, 0);
        var one = new Account(2000, 1);
        zero.pay(one, 100);
        one.pay(zero, 1000);
        zero.withdraw(170);
        System.out.println(zero);
        System.out.println(one);
    }
}
```

Account.java

```java
package main4.task1;

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


public class Account {

    private final List<Payment> payments;

    private final List<Income> incomes;

    private final List<Withdrawal> withdrawals;

    private final long id;

    private long money;

    public Account(List<Payment> payments, List<Income> incomes, List<Withdrawal> withdrawals, long id, long money) {
        this.payments = payments;
        this.incomes = incomes;
        this.withdrawals = withdrawals;
        this.id = id;
        this.money = money;
    }

    public Account(long money, long id) {
        this(new ArrayList<>(), new ArrayList<>(), new ArrayList<>(), id, money);
    }

    public void withdraw(long money) {
        withdrawals.add(new Withdrawal(money));
        this.money -= money;
    }

    public void pay(Account destination, long payment) {
        payments.add(new Payment(destination, payment));
        destination.incomes.add(new Income(this, payment));
        this.money -= payment;
        this.money += payment;
    }

    @Override
    public String toString() {
        return "Account{" +
                "payments=" + payments +
                ", incomes=" + incomes +
                ", withdrawals=" + withdrawals +
                ", money=" + money +
                '}';
    }

```

```
53      private record Payment(Account destination, long money) {
54
55          @Override
56          public String toString() {
57              return "Payment{" +
58                      "destination=" + destination.id +
59                      ", money=" + money +
60                      '}';
61          }
62      }
63
64      private record Income(Account source, long money) {
65
66          @Override
67          public String toString() {
68              return "Income{" +
69                      "source=" + source.id +
70                      ", money=" + money +
71                      '}';
72          }
73      }
74
75      private record Withdrawal(long money) {
76
77      }
78  }
```

Задание 2 Реализовать агрегирование. При создании класса агрегируемый класс объявляется как атрибут (локальная переменная, параметр метода). Включить в каждый класс 2-3 метода на выбор. Продемонстрировать использование разработанных классов.

Создать класс Страница, используя класс Абзац.

App2.java

```
1   package main4.task2;
2
3   public class App2 {
4
5       public static void main(String[] args) {
6           var page = new Page(new Paragraph("Hello there"), new Paragraph("there"));
7           var found = page.findAll("there");
8           System.out.println(found);
9           System.out.println(page.read());
10      }
11  }
```

Page.java

```
1   package main4.task2;
2
3   import java.util.ArrayList;
4   import java.util.List;
5   import java.util.regex.Pattern;
6   import java.util.stream.Collectors;
7
8
9   public record Page(List<Paragraph> paragraphs) {
10
11      public Page(Paragraph... paragraph) {
12          this(List.of(paragraph));
13      }
14
15      public String read() {
16          return paragraphs.stream().map(Paragraph::text).collect(Collectors.joining("\n"));
17      }
18
19      public List<Integer> findAll(String regex) {
20          int offset = 0;
21          var starts = new ArrayList<Integer>();
22          var pattern = Pattern.compile(regex);
23          for (var paragraph : paragraphs) {
24              var matcher = pattern.matcher(paragraph.text());
25              while (matcher.find()) {
26                  starts.add(matcher.start() + offset);
```

```
            }
            offset += paragraph.text().length();
        }
        return starts;
    }
}
```

Paragraph.java

```
package main4.task2;

public record Paragraph(String text) {

}
```

Задание 3 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 на- значение. Пациент может быть выписан из Больницы по окончании лечения, при нарушении режима или иных обстоятельствах.

Main.java

```
package main4.task3;

import main5.task3.Clinic;
import main5.task3.Doctor;
import main5.task3.Nurse;
import main5.task3.Patient;
import java.util.ArrayList;


public class Main {

    public static void main(String[] args) {
        var clinic = new Clinic(
                new ArrayList<>() {{
                    add(new main5.task3.Doctor());
                    add(new Doctor());
                }},
                new ArrayList<>() {{
                    add(new Nurse());
                }}
        );
        var patient = new Patient();
        clinic.registerPatient(patient);
        clinic.cureAll();
        clinic.dischargeAll();
    }
}
```

Clinic.java

```
package main4.task3;

import java.util.List;
import java.util.Random;
import java.util.function.BiConsumer;
import java.util.stream.Collectors;


public class Clinic {

    private final List<Doctor> doctors;

    private final List<Nurse> nurses;

    public Clinic(List<Doctor> doctors, List<Nurse> nurses) {
        this.doctors = doctors;
```

```
        this.nurses = nurses;
    }

    public void registerPatient(Patient patient) {
        var doctor = List.copyOf(doctors).get(new Random().nextInt(doctors.size()));
        doctor.prescribeTreatment(patient);
    }

    public void cureAll() {
        applyToAllPatients((d, p) -> {
            if (Math.random() > 0.5) {
                var docs = doctors.stream().filter(doc -> !doc.equals(d)).collect(Collectors.toList
    ());
                var doc = docs.get(new Random().nextInt(docs.size()));
                doc.cure(p);
            } else {
                var nurse = nurses.get(new Random().nextInt(nurses.size()));
                nurse.cure(p);
            }
        });
    }

    public void dischargeAll() {
        applyToAllPatients((d, p) -> {
            if (p.getStatus().equals(Status.HEALTHY) || p.getStatus().equals(Status.RULES_VIOLATED))
     {
                d.patients.remove(p);
            }
        });
    }

    public void applyToAllPatients(BiConsumer<Doctor, Patient> doctorPatientBiConsumer) {
        for (var doc : doctors) {
            for (var pat : doc.patients) {
                doctorPatientBiConsumer.accept(doc, pat);
            }
        }
    }
}
```

Doctor.java

```
package main4.task3;

import main5.task3.Medic;
import main5.task3.Patient;
import main5.task3.Treatment;
import java.util.*;


public class Doctor extends Medic {

    public final Set<Patient> patients;

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

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

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

Medic.java

```
package main4.task3;

import main5.task3.Patient;


public abstract class Medic {

    public void cure(Patient patient) {
```

```
        patient.cure();
    }
}
```

Patient.java

```
package main4.task3;

import main5.task3.Status;
import main5.task3.Treatment;


public class Patient {

    private main5.task3.Treatment treatment;

    private main5.task3.Status status;

    public Patient() {
        this(main5.task3.Treatment.PILLS, main5.task3.Status.SICK);
    }

    public Patient(main5.task3.Treatment treatment, main5.task3.Status status) {
        this.treatment = treatment;
        this.status = status;
    }

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

    public void cure() {
        status = main5.task3.Status.HEALTHY;
    }

    public void violateRules() {
        status = main5.task3.Status.RULES_VIOLATED;
    }

    public Status getStatus() {
        return status;
    }
}
```

Вывод Я написал несколько систем классов. Использовал агрегирование, наследование.