

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

**Лабораторная работа №4**

**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 №4

### Вариант 3

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

#### **Задание №1:**

Реализовать указанный класс, включив в него вспомогательный внутренний класс или классы. Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.

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

#### **Код программы:**

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

public class Task_1 {
    private double balance;
    private List<Transaction> transactions;

    public Task_1() {
        this.balance = 0;
        this.transactions = new ArrayList<>();
    }

    public void deposit(double amount) {
        balance += amount;
        Transaction transaction = new Transaction("Deposit", amount, new Date());
        transactions.add(transaction);
    }

    public void withdraw(double amount) {
        if (amount > balance) {
            System.out.println("Insufficient funds");
        } else {
            balance -= amount;
            Transaction transaction = new Transaction("Withdrawal", amount, new Date());
            transactions.add(transaction);
        }
    }
}
```

```

    }
}

public void displayTransactions() {
    System.out.println("Transaction history:");
    for (Transaction transaction : transactions) {
        System.out.println(transaction);
    }
}

private class Transaction {
    private String type;
    private double amount;
    private Date timestamp;

    public Transaction(String type, double amount, Date timestamp) {
        this.type = type;
        this.amount = amount;
        this.timestamp = timestamp;
    }

    @Override
    public String toString() {
        return "Type: " + type + ", Amount: " + amount + ", Date: " + timestamp;
    }
}

public static void main(String[] args) {
    Task_1 account = new Task_1();
    account.deposit(1000);
    account.withdraw(500);
    account.deposit(200);
    account.displayTransactions();
}
}

```

## Результат работы:

```

Transaction history:
Type: Deposit, Amount: 1000.0, Date: Tue May 14 16:00:30 MSK 2024
Type: Withdrawal, Amount: 500.0, Date: Tue May 14 16:00:30 MSK 2024
Type: Deposit, Amount: 200.0, Date: Tue May 14 16:00:30 MSK 2024

Process finished with exit code 0

```

### **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.

### **Код программы:**

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

public class Task_2 {
    public static class Paragraph {
        private String content;

        public Paragraph(String content) {
            this.content = content;
        }

        public String getContent() {
            return content;
        }

        public void setContent(String content) {
            this.content = content;
        }

        public void addText(String text) {
            content += text;
        }

        public int wordCount() {
            String[] words = content.split("\\s+");
            return words.length;
        }
    }

    public static class Page {
        private List<Paragraph> paragraphs;

        public Page() {
            this.paragraphs = new ArrayList<>();
        }
    }
}
```

```

    }

    public void addParagraph(Paragraph paragraph) {
        paragraphs.add(paragraph);
    }

    public void removeParagraph(Paragraph paragraph) {
        paragraphs.remove(paragraph);
    }

    public void displayContent() {
        for (Paragraph paragraph : paragraphs) {
            System.out.println(paragraph.getContent());
        }
    }

    public int paragraphCount() {
        return paragraphs.size();
    }
}

public static void main(String[] args) {
    Paragraph paragraph1 = new Paragraph("This is the first paragraph. ");
    Paragraph paragraph2 = new Paragraph("This is the second paragraph. ");
    Paragraph paragraph3 = new Paragraph("This is the third paragraph. ");

    Page page = new Page();
    page.addParagraph(paragraph1);
    page.addParagraph(paragraph2);
    page.addParagraph(paragraph3);

    page.displayContent();
    System.out.println("Number of paragraphs: " + page.paragraphCount());
    System.out.println("Total words: " + totalWords(page));
}

public static int totalWords(Page page) {
    int totalWords = 0;
    for (Paragraph paragraph : page.paragraphs) {
        totalWords += paragraph.wordCount();
    }
    return totalWords;
}
}

```

### Результат работы:

```
This is the first paragraph.  
This is the second paragraph.  
This is the third paragraph.  
Number of paragraphs: 3  
Total words: 15  
  
Process finished with exit code 0
```

### 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 назначение. Пациент может быть выписан из Больницы по окончании лечения, при нарушении режима или иных обстоятельствах.

### Код программы:

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

```
public class Task_3 {  
    public static class Patient {  
        private String name;  
        private int age;  
        private String diagnosis;  
        private String condition;  
        private 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 void changeCondition(String newCondition) {
        this.condition = newCondition;
    }
}

public static class Doctor {
    private String name;
    private String specialization;

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

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

public static class Nurse {
    private String name;
    private String specialization;

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

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

public static void main(String[] args) {
    Doctor doctor = new Doctor("Dr. Smith", "Cardiologist");
    Patient patient = new Patient("John Doe", 45, "Heart Disease");

```

```
patient.setAttendingDoctor(doctor);

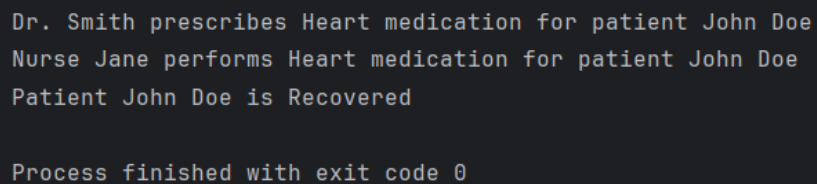
doctor.prescribeTreatment(patient, "Heart medication");

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:

A screenshot of a terminal window with a dark background and light-colored text. The output shows three lines of status messages: 'Dr. Smith prescribes Heart medication for patient John Doe', 'Nurse Jane performs Heart medication for patient John Doe', and 'Patient John Doe is Recovered'. Below these, after a blank line, it says 'Process finished with exit code 0'.

```
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.