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

Лабораторная работа №4

По дисциплине: «ССП»
Вариант - 12

Выполнил:
Студент 3 курса
Группы ПО-8
Иванюк М.С.
Проверил:
Крощенко А.А

Брест, 2024

Лабораторная работа №4

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

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

Код программы:

```
package Lab4;
import java.util.ArrayList;
import java.util.Date;

public class task1 {
    public static void main(String[] args){
        Account account = new Account(2232);
        account.makeDeposit(10000);
        account.makePayment(200);
        account.makeWithdrawal(2322);
        account.makeTransaction(2500, Account.TransactionType.DEPOSIT);
        account.makeDeposit(5555);
        account.makePayment(4344);
        account.makeWithdrawal(250);
        account.printTransactionsHistory();
    }
}

class Account{
    public enum TransactionType{
        WITHDRAWAL, // Снятие
        PAYMENT,    // Платеж
        DEPOSIT     // Поступление
    }
    private int accountNumber;
    private double balance;
    private ArrayList<Transaction> transactionsHistory;

    public Account(int cardNumber){
        this.accountNumber = cardNumber;
        this.balance = 0.0;
        transactionsHistory = new ArrayList<>();
    }

    public void makeDeposit(double deposit){
        if(deposit<=0){
            System.out.println("Сумма депозита должна быть больше 0.");
        }
        else{
            this.balance+=deposit;
            transactionsHistory.add(new
Transaction(deposit,TransactionType.DEPOSIT,this.balance));
        }
    }

    public void makeWithdrawal(double amount){
        if(amount>0 && amount<=balance){
            this.balance-=amount;
            transactionsHistory.add(new
Transaction(amount,TransactionType.WITHDRAWAL,this.balance));
```

```
        }
        else{
            System.out.println("Некорректная сумма снятия или недостаточно
средств на счете.");
        }
    }

    public void printTransactionsHistory(){
        System.out.println("История транзацикций:");
        for (Transaction transaction : this.transactionsHistory) {
            System.out.println(transaction);
        }
    }
    public void makePayment(double amount){
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            transactionsHistory.add(new Transaction(amount,
TransactionType.PAYMENT,this.balance));
        } else {
            System.out.println("Недостаточно средств для проведения платежа
или сумма некорректна.");
        }
    }

    public void makeTransaction(double amount,TransactionType type) {
        switch (type){
            case TransactionType.DEPOSIT:
                this.makeDeposit(amount);
                break;
            case TransactionType.PAYMENT:
                this.makePayment(amount);
                break;
            case TransactionType.WITHDRAWAL:
                this.makeWithdrawal(amount);
                break;
            default:
                System.out.println("error!");
        }
    }
    class Transaction{
        private Date transactionDate;
        private double amount;
        private double currentBalance;//остаток на счете после транзацкии
        private TransactionType type;

        public Transaction(double amount, TransactionType type , double
currentBalance){
            this.amount = amount;
            this.type = type;
            this.currentBalance = Account.this.balance;
            this.transactionDate = new Date();
        }

        public Date getTransactionDate() {
            return transactionDate;
        }

        public double getAmount() {
            return amount;
        }

        public TransactionType getType() {
            return type;
        }
```

```
        public void setTransactionDate(Date transactionDate) {
            this.transactionDate = transactionDate;
        }

        public void setAmount(double amount) {
            this.amount = amount;
        }

        public void setType(TransactionType type) {
            this.type = type;
        }

        @Override
        public String toString() {
            return "Тип транзации: " + this.type + ", Дата: " +
this.transactionDate +
                    ", Сумма: " + this.amount + " BYN, Остаток на счету: " +
this.currentBalance + " BYN;";
        }
    }

}
```

Результат работы программы:

```
"D:\Porgramming instruments\JDK2023\Java\jdk-21\bin\java.exe" "-javaagent:D:\Porgramming instruments\IntelliJ IDEA
История транзацикций:
Тип транзации: DEPOSIT, Дата: Wed Mar 13 14:10:13 MSK 2024, Сумма: 10000.0 BYN, Остаток на счету: 10000.0 BYN;
Тип транзации: PAYMENT, Дата: Wed Mar 13 14:10:13 MSK 2024, Сумма: 200.0 BYN, Остаток на счету: 9800.0 BYN;
Тип транзации: WITHDRAWAL, Дата: Wed Mar 13 14:10:13 MSK 2024, Сумма: 2322.0 BYN, Остаток на счету: 7478.0 BYN;
Тип транзации: DEPOSIT, Дата: Wed Mar 13 14:10:13 MSK 2024, Сумма: 2500.0 BYN, Остаток на счету: 9978.0 BYN;
Тип транзации: DEPOSIT, Дата: Wed Mar 13 14:10:13 MSK 2024, Сумма: 5555.0 BYN, Остаток на счету: 15533.0 BYN;
Тип транзации: PAYMENT, Дата: Wed Mar 13 14:10:13 MSK 2024, Сумма: 4344.0 BYN, Остаток на счету: 11189.0 BYN;
Тип транзации: WITHDRAWAL, Дата: Wed Mar 13 14:10:13 MSK 2024, Сумма: 250.0 BYN, Остаток на счету: 10939.0 BYN;
```

Задание 2: Создать объект класса Компьютер, используя классы Материнская Плата, Дисковод, ОЗУ.

Код программы:

```
package Lab4.task2;

public class task2 {
    public static void main(String[] args){
        Motherboard motherboard = new Motherboard("ASUS ROG Strix Z490-E
Gaming", "LGA1200", "Intel Z490", 128);
        Drive drive = new Drive("Samsung 970 EVO Plus", Drive.DriveType.SSD,
500);
        RAM ram = new RAM("Corsair Vengeance LPX", RAM.MemoryType.DDR4, 32,
3200);

        Computer myComputer = new Computer();
        myComputer.setMotherboard(motherboard);
        myComputer.setDrive(drive);
        myComputer.setRam(ram);
```

```
        System.out.println(myComputer);
    }
}

class Computer{
    private Motherboard motherboard;
    private Drive drive;
    private RAM ram;

    public Computer(){
    }
    public Computer(Motherboard motherboard,Drive drive,RAM ram){
        this.motherboard = motherboard;
        this.drive = drive;
        this.setRam(ram);
    }

    public void setDrive(Drive drive) {
        this.drive = drive;
    }
    public void setMotherboard(Motherboard motherboard){
        this.motherboard = motherboard;
    }
    public void setRam(RAM ram){
        if(ram.capacity <= this.motherboard.getMaxRAM()){
            this.ram = ram;
        }
        else{
            System.out.println("Материнская плата " +
this.motherboard.getTitle() + "поддерживает максимальный объем оперативной
памяти " + this.motherboard.getMaxRAM());
        }
    }

    @Override
    public String toString() {
        return "Computer specs:" + this.motherboard + this.drive + this.ram;
    }
}

class Motherboard{
    private String title;
    private String socket;
    private  String chipset;
    private int maxRAM;

    public Motherboard(String title,String socket,String chipset,int maxRAM){
        this.title = title;
        this.socket = socket;
        this.chipset = chipset;
        this.maxRAM = maxRAM;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public void setSocket(String socket) {
        this.socket = socket;
    }

    public void setChipset(String chipset) {
        this.chipset = chipset;
    }
```

```
    public void setMaxRAM(int maxRAM) {
        this.maxRAM = maxRAM;
    }

    public String getTitle() {
        return title;
    }

    public String getSocket() {
        return socket;
    }

    public String getChipset() {
        return chipset;
    }

    public int getMaxRAM() {
        return maxRAM;
    }

    @Override
    public String toString() {
        return "\nMorherboard specs: name = " + this.title + ", socket = " +
this.socket + ", chipset = " +
                this.chipset + ", max RAM capacity = " + this.maxRAM + " GB;";
    }
}

class Drive{
    public enum DriveType{
        SSD,
        HDD
    }
    String name;
    DriveType type;
    int capacity;

    public Drive(String name,DriveType type, int memory){
        this.name = name;
        this.type = type;
        this.capacity = memory;
    }

    public DriveType getType() {
        return type;
    }

    public int getCapacity() {
        return capacity;
    }

    public void setType(DriveType type) {
        this.type = type;
    }

    public void setCapacity(int capacity) {
        this.capacity = capacity;
    }

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "\nDrive specs: name = " + this.name + ", type = " + type + ",
capacity = " + capacity + " GB;";
    }
}

class RAM{
    public enum MemoryType{
        DDR2,
        DDR3,
        DDR4,
        DDR5
    }
    String name;
    MemoryType type;
    int capacity;
    int speed;

    public RAM(String title, MemoryType type, int capacity, int speed) {
        this.name = title;
        this.type = type;
        this.capacity = capacity;
        this.speed = speed;
    }

    public int getCapacity() {
        return capacity;
    }

    public String getName() {
        return name;
    }

    public int getSpeed() {
        return speed;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setCapacity(int capacity) {
        this.capacity = capacity;
    }

    public void setSpeed(int speed) {
        this.speed = speed;
    }

    @Override
    public String toString() {
        return "\nRAM specs: name = " + name + ", capacity = " + capacity + "
GB, RAM type = " +type+ ", speed = " + speed + " MHz;";
    }
}
```

Результат работы программы:

```
"D:\Porgramming instruments\JDK2023\Java\jdk-21\bin\java.exe" "-javaagent:D:\Porgramming instruments\IntelliJ IDEA Community
Computer specs:
Morherboard specs: name = ASUS ROG Strix Z490-E Gaming, socket = LGA1200, chipset = Intel Z490, max RAM capacity = 128 GB;
Drive specs: name = Samsung 970 EVO Plus, type = SSD, capacity = 500 GB;
RAM specs: name = Corsair Vengeance LPX, capacity = 32 GB, RAM type = DDR4, speed = 3200 MHz;
```

Задание 3: Система Факультатив. Преподаватель объявляет запись на Курс. Студент записывается на Курс, обучается и по окончании Преподаватель выставляет Оценку, которая сохраняется в Архиве. Студентов, Преподавателей и Курсов при обучении может быть несколько.

Код программы:

```
package Lab4;

import java.util.*;

public class task3 {
    public static void main(String[] args){
        Archive archive = new Archive();
        Student student1 = new Student("Бубен Станислав Олеговочи","ПО-8");
        Student student2 = new Student("Бувин Дмитрий Александрович","ПО-8");
        Student student3 = new Student("Назаров Кирилл Викторович","ПО-8");

        Course course1 = new Course("Компьютерные системы и сети");
        Course course2 = new Course("Проектирование интернет систем");
        Teacher teacher = new Teacher("Гречаник Татьяна Викторовна");

        teacher.addCourse(course1);
        teacher.announceCourseRegistration(course1);
        student1.enrollInCourse(course1);
        student2.enrollInCourse(course1);
        student3.enrollInCourse(course1);
        teacher.evaluateTheStudent(course1,student1,new Mark(5));
        teacher.evaluateTheStudent(course1,student2,new Mark(5));
        teacher.evaluateTheStudent(course1,student3,new Mark(4));
        teacher.removeCourse(course1);

        System.out.println();

        teacher.addCourse(course2);
        teacher.announceCourseRegistration(course2);
        student1.enrollInCourse(course2);
        student2.enrollInCourse(course2);
        student3.enrollInCourse(course2);
        teacher.evaluateTheStudent(course2,student1,new Mark(2));
        teacher.evaluateTheStudent(course2,student2,new Mark(2));
        teacher.evaluateTheStudent(course2,student3,new Mark(4));
        teacher.removeCourse(course2);

        archive.printArchive();
    }
}

class Teacher{
    private String name;
    ArrayList<Course> courses;

    public Teacher(String name) {
```

```
        this.name = name;
        this.courses = new ArrayList<>();
    }

    public void addCourse(Course course){
        course.setTeacher(this);
        this.courses.add(course);
    }

    public void announceCourseRegistration(Course course) {
        System.out.println(this.name + " объявил(а) запись на курс " +
course.getName()+".");
        course.openRegistration();
    }
    public void removeCourse(Course course){
        this.courses.remove(course);
        System.out.println(this.getName() + " завершил(а) курс  " +
course.getName() + ".");
        course.closeRegistration();
    }

    public void evaluateTheStudent(Course course, Student student, Mark
mark){
        System.out.println(this.name + " выставил(а) оценку " +
mark.getMark() + " студенту " + student.getName() + " за курс " +
course.getName() + ".");
        student.removeCourse(course);
        Archive.saveMark(course, student, mark);
    }
    public String getName() {
        return this.name;
    }
    public ArrayList<Course> getCourses() {
        return this.courses;
    }
}

class Student{
    private String name;
    private String group;
    private ArrayList<Course> courses;

    public Student(String name, String group) {
        this.name = name;
        this.group = group;
        this.courses = new ArrayList<>();
    }

    public void enrollInCourse(Course course) {
        if (course.isRegistrationOpen()) {
            course.addStudent(this);
            this.courses.add(course);
            System.out.println("Студент " + this.name +" группы " +
this.group + " начал изучать курс " + course.getName()+".");
        } else {
            System.out.println("Запись на курс " + course.getName() + "
закрыта.");
        }
    }
    public String getName() {
        return this.name;
    }
    public String getGroup() {
        return this.group;
```

```
    }
    public ArrayList<Course> getCourses() {
        return this.courses;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setGroup(String group) {
        this.group = group;
    }

    public void removeCourse(Course course){
        this.courses.remove(course);
        System.out.println("Студент группы "+this.getGroup() + " " +
this.getName() + " завершил изучение курса " + course.getName()+".");
    }

    @Override
    public String toString() {
        return "Student{" +
                "name='" + this.name + '\'' +
                ", group='" + this.group + '\'' +
                '}';
    }
}

class Course{
    private String name;
    private Teacher teacher;
    private ArrayList<Student> students;
    private boolean registrationOpen;
    public Course(String name){
        this.name = name;
        this.students = new ArrayList<>();
        this.registrationOpen = false;
    }
    public void openRegistration() {
        this.registrationOpen = true;
    }

    public void closeRegistration() {
        this.registrationOpen = false;
    }

    public boolean isRegistrationOpen() {
        return this.registrationOpen;
    }

    public void addStudent(Student student) {
        this.students.add(student);
    }

    public void setTeacher(Teacher teacher) {
        this.teacher = teacher;
    }

    public String getTeacherName() {
        return this.teacher.getName();
    }

    public List<Student> getStudents() {
        return this.students;
    }
    public String getName() {
```

```
        return this.name;
    }

}
class Archive {
    private static Map<Course, Map<Student, Mark>> gradesArchive;

    public Archive() {
        gradesArchive = new HashMap<>();
    }
    public static void saveMark(Course course, Student student, Mark mark) {
        Map<Student, Mark> courseGrades =
gradesArchive.computeIfAbsent(course, m -> new HashMap<>());
        courseGrades.put(student, mark);
    }

    public void printArchive(){
        StringBuilder result = new StringBuilder();
        for (Course course : gradesArchive.keySet()) {
            result.append("Course: ").append(course.getName()).append(",
Teacher: ").append(course.getTeacherName()).append("\n");
            Map<Student, Mark> courseGrades = gradesArchive.get(course);
            for (Student student : courseGrades.keySet()) {
                Mark grade = courseGrades.get(student);
                result.append("\t\tStudent:
").append(student.getName()).append(", Group: ").append(student.getGroup());
                result.append(", Mark:
").append(grade.getMark()).append("\n");
            }
        }
        System.out.println("\n"+result);
    }

    @Override
    public String toString() {
        StringBuilder result = new StringBuilder();
        for (Course course : gradesArchive.keySet()) {
            result.append("Course: ").append(course.getName()).append(",
Teacher: ").append(course.getTeacherName()).append("\n");
            Map<Student, Mark> courseGrades = gradesArchive.get(course);
            for (Student student : courseGrades.keySet()) {
                Mark grade = courseGrades.get(student);
                result.append("\tStudent:
").append(student.getName()).append(", Group: ").append(student.getGroup())
                        .append(", Mark:
").append(grade.getMark()).append("\n");
            }
        }
        return result.toString();
    }
}

class Mark {
    private int mark;

    public Mark(int mark) {
        this.mark = mark;
    }
    public int getMark() {
        return this.mark;
    }
    public void setMark(int mark) {
        this.mark = mark;
    }
```

```
    @Override
    public String toString() {
        return "Grade{" +
                "grade=" + this.mark +
                '}';
    }
}
```

Результат работы программы:

```
"D:\Porgramming instruments\JDK2023\Java\jdk-21\bin\java.exe" "-javaagent:D:\Porgramming instruments\IntelliJ IDEA Community
Гречаник Татьяна Викторовна объявил(а) запись на курс Компьютерные системы и сети.
Студент Бубен Станислав Олеговочи группы ПО-8 начал изучать курс Компьютерные системы и сети.
Студент Бувин Дмитрий Александрович группы ПО-8 начал изучать курс Компьютерные системы и сети.
Студент Назаров Кирилл Викторович группы ПО-8 начал изучать курс Компьютерные системы и сети.
Гречаник Татьяна Викторовна выставил(а) оценку 5 студенту Бубен Станислав Олеговочи за курс Компьютерные системы и сети.
Студент группы ПО-8 Бубен Станислав Олеговочи завершил изучение курса Компьютерные системы и сети
Гречаник Татьяна Викторовна выставил(а) оценку 5 студенту Бувин Дмитрий Александрович за курс Компьютерные системы и сети.
Студент группы ПО-8 Бувин Дмитрий Александрович завершил изучение курса Компьютерные системы и сети
Гречаник Татьяна Викторовна выставил(а) оценку 4 студенту Назаров Кирилл Викторович за курс Компьютерные системы и сети.
Студент группы ПО-8 Назаров Кирилл Викторович завершил изучение курса Компьютерные системы и сети
Гречаник Татьяна Викторовна завершил(а) курс  Компьютерные системы и сети.

Гречаник Татьяна Викторовна объявил(а) запись на курс Проектирование интернет систем.
Студент Бубен Станислав Олеговочи группы ПО-8 начал изучать курс Проектирование интернет систем.
Студент Бувин Дмитрий Александрович группы ПО-8 начал изучать курс Проектирование интернет систем.
Студент Назаров Кирилл Викторович группы ПО-8 начал изучать курс Проектирование интернет систем.
Гречаник Татьяна Викторовна выставил(а) оценку 2 студенту Бубен Станислав Олеговочи за курс Проектирование интернет систем.
Студент группы ПО-8 Бубен Станислав Олеговочи завершил изучение курса Проектирование интернет систем
Гречаник Татьяна Викторовна выставил(а) оценку 2 студенту Бувин Дмитрий Александрович за курс Проектирование интернет систем
Студент группы ПО-8 Бувин Дмитрий Александрович завершил изучение курса Проектирование интернет систем
Гречаник Татьяна Викторовна выставил(а) оценку 4 студенту Назаров Кирилл Викторович за курс Проектирование интернет систем.
Студент группы ПО-8 Назаров Кирилл Викторович завершил изучение курса Проектирование интернет систем

Course: Проектирование интернет систем, Teacher: Гречаник Татьяна Викторовна
        Student: Бубен Станислав Олеговочи, Group: ПО-8, Mark: 2
        Student: Бувин Дмитрий Александрович, Group: ПО-8, Mark: 2
        Student: Назаров Кирилл Викторович, Group: ПО-8, Mark: 4
Course: Компьютерные системы и сети, Teacher: Гречаник Татьяна Викторовна
        Student: Бубен Станислав Олеговочи, Group: ПО-8, Mark: 5
        Student: Бувин Дмитрий Александрович, Group: ПО-8, Mark: 5
        Student: Назаров Кирилл Викторович, Group: ПО-8, Mark: 4
```

Вывод: в ходе выполнения лабораторной работы я приобрел практические навыки в области объектно-ориентированного программирования.