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

Отчёт

по лабораторной работе №4

Выполнила:
студентка группы ПО-9
Матюшик Е.П.

Проверил:
Крощенко А. А.

Брест 2024

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

Задание 1

Вариант 4

Создать класс Зачетная Книжка с внутренним классом, с помощью объектов которого можно хранить информацию о сессиях, зачетах, экзаменах.

Результат программы:

```
Sessions:
Course: 3, Session Type: Winter
Exams:
Subject: Biology, Grade: 9, Status: Passed
Zachets:
Subject: Languages, Grade: 9, Status: Passed
```

Код программы:

```
import java.util.ArrayList;
import java.util.Random;
public class ZachetnayaKnizhka {
    private ArrayList<Session> sessions;
    private ArrayList<Exam> exams;
    private ArrayList<Zachet> zachets;
    private Random random;

    public ZachetnayaKnizhka() {
        this.sessions = new ArrayList<>();
        this.exams = new ArrayList<>();
        this.zachets = new ArrayList<>();
        this.random = new Random();
    }

    public void addSession() {
        int course = random.nextInt(4) + 1; // Генерируем случайное число от 1 до 4
        String[] sessionTypes = {"Winter", "Summer"};
        String sessionType = sessionTypes[random.nextInt(sessionTypes.length)]; // Случайный
выбор из списка типов сессии
        sessions.add(new Session(course, sessionType));
    }

    public void addExam() {
        String[] subjects = {"Math", "Physics", "Chemistry", "Biology"};
        String subject = subjects[random.nextInt(subjects.length)]; // Случайный выбор из списка
предметов
        int grade = random.nextInt(11); // Генерируем случайную оценку от 0 до 10
```

```
        exams.add(new Exam(subject, grade));
    }

    public void addZachet() {
        String[] subjects = {"History", "Literature", "Geography", "Languages"};
        String subject = subjects[random.nextInt(subjects.length)]; // Случайный выбор из списка
предметов
        int grade = random.nextInt(11); // Генерируем случайную оценку от 0 до 10
        zachets.add(new Zachet(subject, grade));
    }
    public void printSessions() {
        System.out.println("Sessions:");
        for (Session session : sessions) {
            System.out.println(session);
        }
    }
    public void printExams() {
        System.out.println("Exams:");
        for (Exam exam : exams) {
            System.out.println(exam);
        }
    }

    public void printZachets() {
        System.out.println("Zachets:");
        for (Zachet zachet : zachets) {
            System.out.println(zachet);
        }
    }

    class Session {
        private int course;
        private String sessionType;

        public Session(int course, String sessionType) {
            this.course = course;
            this.sessionType = sessionType;
        }

        @Override
        public String toString() {
            return "Course: " + course + ", Session Type: " + sessionType;
        }
```

```
}

class Exam {
    private String subject;
    private int grade;

    public Exam(String subject, int grade) {
        this.subject = subject;
        this.grade = grade;
    }

    @Override
    public String toString() {
        String status = (grade >= 4) ? "Passed" : "Failed";
        return "Subject: " + subject + ", Grade: " + grade + ", Status: " + status;
    }
}

class Zachet {
    private String subject;
    private int grade;

    public Zachet(String subject, int grade) {
        this.subject = subject;
        this.grade = grade;
    }

    @Override
    public String toString() {
        String status = (grade >= 4) ? "Passed" : "Failed";
        return "Subject: " + subject + ", Grade: " + grade + ", Status: " + status;
    }
}

public static void main(String[] args) {
    ZachetnayaKnizhka book = new ZachetnayaKnizhka();
    book.addSession();
    book.addExam();
    book.addZachet();

    book.printSessions();
    book.printExams();
    book.printZachets();
```

```
    }
}
```

Задание 2

Вариант 1

Реализовать агрегирование. Создать класс Строка, используя классы Слово, Символ.

Входные данные:

Символ1 = ‘Ю’

Символ2 = ‘Д’

Слово1 = ‘Привет’

Слово2 = ‘Мир’

Результат программы:

```
Количество символов в строке символов: 2
Общая длина слов в строке: 9
Первое слово начинается с гласной буквы: false
Символ 1 является гласной: true
```

Код программы:

```
import java.util.regex.Pattern;
class Symbol {
    private char symbol;
    private static final Pattern RUSSIAN_VOWELS =
Pattern.compile("[аеёиоуыэюяАЕЁИОУЫЭЮЯ]");
    private static final Pattern RUSSIAN_CONSONANTS =
Pattern.compile("[бвгджзйклмнпрстфхцчшщБВГДЖЗЙКЛМНПРСТФХЦЧШЩ]");

    public Symbol(char symbol) {
        this.symbol = symbol;
    }

    public char getSymbol() {
        return symbol;
    }

    public void setSymbol(char symbol) {
        this.symbol = symbol;
    }

    public boolean isVowel() {
        return RUSSIAN_VOWELS.matcher(Character.toString(symbol)).matches();
    }
}
```

```
class Word {
    private String word;

    public Word(String word) {
        this.word = word;
    }

    public String getWord() {
        return word;
    }

    public void setWord(String word) {
        this.word = word;
    }

    public int getLength() {
        return word.length();
    }

    public boolean startsWithVowel() {
        return new Symbol(word.charAt(0)).isVowel();
    }
}

class Sentence {
    private Word[] words;

    public Sentence(Word[] words) {
        this.words = words;
    }

    public Word[] getWords() {
        return words;
    }

    public void setWords(Word[] words) {
        this.words = words;
    }

    public int wordCount() {
        return words.length;
    }
```

```
    public int totalWordLength() {
        int totalLength = 0;
        for (Word word : words) {
            totalLength += word.getLength();
        }
        return totalLength;
    }
}

public class Main {
    public static void main(String[] args) {
        Symbol symbol1 = new Symbol('Ю');
        Symbol symbol2 = new Symbol('Д');

        Word word1 = new Word("Привет");
        Word word2 = new Word("Мир");

        Word[] words = {word1, word2};
        Sentence sentence = new Sentence(words);

        System.out.println("Количество букв в строке: " + sentence.wordCount());
        System.out.println("Общая длина слов в строке: " + sentence.totalWordLength());

        // Additional usage of class methods
        System.out.println("Первое слово начинается с гласной буквы: " +
word1.startsWithVowel());
        System.out.println("Символ 1 является гласной: " + symbol1.isVowel());
    }
}
```

Задание 3

Вариант 2

Система Платежи. Клиент имеет Счет в банке и Кредитную Карту (КК). Клиент может оплатить Заказ, сделать платеж на другой Счет, заблокировать КК и аннулировать Счет. Администратор может заблокировать КК за превышение кредита.

Результат программы:

```
Имя клиента: John Doe
Номер телефона: +123456789
Счет в банке: 123456, Баланс: 1000.0
Кредитная карта: 789012345678, Доступный кредит: 500.0
Клиент John Doe оплачивает заказ на сумму 200.0 с помощью кредитной карты.
```

```
Имя клиента: John Doe
Номер телефона: +123456789
Счет в банке: 123456, Баланс: 1000.0
Кредитная карта: 789012345678, Доступный кредит: 500.0
Клиент John Doe делает платеж на другой счет на сумму 100.0 с помощью счета в банке.
Администратор заблокировал карту клиента John Doe за превышение кредита.
```

Код программы:

```
 class Client {
  private String name;
  private String phoneNumber;
  private BankAccount bankAccount;
  private CreditCard creditCard;

  public Client(String name, String phoneNumber) {
    this.name = name;
    this.phoneNumber = phoneNumber;
  }

  public String getName() {
    return name;
  }

  public String getPhoneNumber() {
    return phoneNumber;
  }

  public void setBankAccount(BankAccount bankAccount) {
    this.bankAccount = bankAccount;
  }

  public void setCreditCard(CreditCard creditCard) {
    this.creditCard = creditCard;
  }

  public void payOrder(Order order) {
    System.out.println("Имя клиента: " + name);
    System.out.println("Номер телефона: " + phoneNumber);
    System.out.println("Счет в банке: " + bankAccount.getAccountNumber() + ", Баланс: " +
bankAccount.getBalance());
    System.out.println("Кредитная карта: " + creditCard.getCardNumber() + ", Доступный
кредит: " + creditCard.getAvailableCredit());
    System.out.println("Клиент " + name + " оплачивает заказ на сумму " +
order.getAmount() + " с помощью кредитной карты.");
```

```
    }

    public void makePaymentToAccount(BankAccount recipientAccount, double amount) {
        System.out.println("Имя клиента: " + name);
        System.out.println("Номер телефона: " + phoneNumber);
        System.out.println("Счет в банке: " + bankAccount.getAccountNumber() + ", Баланс: " +
bankAccount.getBalance());
        System.out.println("Кредитная карта: " + creditCard.getCardNumber() + ", Доступный
кредит: " + creditCard.getAvailableCredit());
        System.out.println("Клиент " + name + " делает платеж на другой счет на сумму " +
amount + " с помощью счета в банке.");
    }

    public void blockCreditCard() {
        if (creditCard != null) {
            creditCard.blockCard();
            System.out.println("Кредитная карта клиента " + name + " была заблокирована.");
        }
    }

    public void cancelBankAccount() {
        if (bankAccount != null) {
            bankAccount.cancelAccount();
            System.out.println("Счет клиента " + name + " был аннулирован.");
        }
    }
}

class BankAccount {
    private String accountNumber;
    private double balance;

    public BankAccount(String accountNumber, double balance) {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    public void topUp(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
```

```
            balance -= amount;
        } else {
            System.out.println("Недостаточно средств на счете");
        }
    }

    public void cancelAccount() {
        // Логика аннулирования счета
    }

    public double getBalance() {
        return balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }
}

class CreditCard {
    private String cardNumber;
    private double creditLimit;
    private double availableCredit;
    private boolean blocked;

    public CreditCard(String cardNumber, double creditLimit) {
        this.cardNumber = cardNumber;
        this.creditLimit = creditLimit;
        this.availableCredit = creditLimit;
    }

    public boolean checkCreditExceed(double amount) {
        return (availableCredit - amount) < 0;
    }

    public void blockCard() {
        blocked = true;
    }

    public String getCardNumber() {
        return cardNumber;
    }
```

```
    public double getAvailableCredit() {
        return availableCredit;
    }
}

class Order {
    private String orderNumber;
    private double amount;

    public Order(String orderNumber, double amount) {
        this.orderNumber = orderNumber;
        this.amount = amount;
    }

    public double getAmount() {
        return amount;
    }
}

class Administrator {
    public void blockCardForOverdraft(Client client) {
        System.out.println("Администратор заблокировал карту клиента " + client.getName() + "
за превышение кредита.");
    }
}

public class Main3 {
    public static void main(String[] args) {
        // Пример использования системы
        Client client = new Client("John Doe", "+123456789");
        BankAccount bankAccount = new BankAccount("123456", 1000.0);
        CreditCard creditCard = new CreditCard("789012345678", 500.0);

        client.setBankAccount(bankAccount);
        client.setCreditCard(creditCard);

        Order order = new Order("0001", 200.0);
        client.payOrder(order);

        BankAccount recipientAccount = new BankAccount("654321", 0.0);
        client.makePaymentToAccount(recipientAccount, 100.0);

        Administrator administrator = new Administrator();
```

```
        administrator.blockCardForOverdraft(client);
    }
}
```

Вывод: в ходе выполнения лабораторной были приобретены практические навыки в области объектно-ориентированного проектирования.