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

ОТЧЁТ

по лабораторной работе №5

Выполнила:
студентка 3 курса
группы ПО-9
Бердникова В.А.

Проверил:
Крощенко А.А.

Брест 2024

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

Вариант 2

Задание 1

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:
interface Abiturient ← abstract class Student ← class Student Of Faculty.

Входные данные:

```
Abiturient abiturient1 = new StudentOfFaculty( name: "Студент1",  age: 18,  faculty: "Факультет1");
Student student1 = new StudentOfFaculty( name: "Студент2",  age: 20,  faculty: "Факультет2");
```

Выходные данные:

```
Студент1 подал заявку на поступление на факультет Факультет1
Студент2 изучает предметы на факультете Факультет2
Студент: Студент2, возраст: 20, факультет: Факультет2
```

Код программы:

Abiturient.java

```
public interface Abiturient {
    void apply();
}
```

StudentOfFaculty.java

```
public class StudentOfFaculty extends Student {
    private String faculty;

    public StudentOfFaculty(String name, int age, String faculty) {
        super(name, age);
        this.faculty = faculty;
    }

    @Override
    public void apply() {
        System.out.println(name + " подал заявку на поступление на факультет
" + faculty);
    }

    @Override
    public void study() {
        System.out.println(name + " изучает предметы на факультете " +
faculty);
    }

    @Override
    public void showInfo() {
        System.out.println("Студент: " + name + ", возраст: " + age + ",
факультет: " + faculty);
    }
}
```

Student.java

```
abstract public class Student implements Abiturient {
    protected String name;
    protected int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public abstract void study();

    public abstract void showInfo();
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        Abiturient abiturient1 = new StudentOfFaculty("Студент1", 18,
"Факультет1");
        Student student1 = new StudentOfFaculty("Студент2", 20,
"Факультет2");

        Abiturient[] abiturients = {abiturient1};
        Student[] students = {student1};

        for (Abiturient abiturient : abiturients) {
            abiturient.apply();
        }

        for (Student student : students) {
            student.study();
            student.showInfo();
        }
    }
}
```

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 Учащийся и подклассы Школьник и Студент. Создать массив объектов суперкласса и заполнить этот массив объектами. Показать отдельно студентов и школьников.

Входные данные:

```
students[0] = new SchoolStudent( name: "John", age: 15, grade: 9);
students[1] = new SchoolStudent( name: "Mary", age: 14, grade: 8);
students[2] = new UniversityStudent( name: "Peter", age: 20, university: "MSU", faculty: "Computer Science");
```

Выходные данные:

```
School Students:
John, age 15, Grade 9
SchoolStudent study in grade 9.
Mary, age 14, Grade 8
SchoolStudent study in grade 8.

University Students:
Peter, age 20, University: MSU
UniStudent study at MSU University, at faculty of Computer Science
```

Код программы:

Main.java

```
public class Main {
    public static void main(String[] args) {
        Student[] students = new Student[3];

        students[0] = new SchoolStudent("John", 15, 9);
        students[1] = new SchoolStudent("Mary", 14, 8);
        students[2] = new UniversityStudent("Peter", 20, "MSU", "Computer
Science");

        System.out.println("School Students:");
        for (Student student : students) {
            if (student instanceof SchoolStudent) {
                System.out.println(student.getName() + ", age " +
student.getAge() + ", " + student.getIdentification());
                student.study();
            }
        }

        System.out.println("\nUniversity Students:");
        for (Student student : students) {
            if (student instanceof UniversityStudent) {
                System.out.println(student.getName() + ", age " +
student.getAge() + ", " + student.getIdentification());
                student.study();
            }
        }
    }
}
```

Student.java

```
abstract public class Student {
    private String name;
    private int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
```

```
        return age;
    }

    public abstract void study();

    public abstract String getIdentification();
}
```

SchoolStudent.java

```
public class SchoolStudent extends Student {
    private int grade;

    public SchoolStudent(String name, int age, int grade) {
        super(name, age);
        this.grade = grade;
    }

    @Override
    public void study() {
        System.out.println("SchoolStudent study in grade " + grade + ".");
    }

    @Override
    public String getIdentification() {
        return "Grade " + grade;
    }
}
```

UniversityStudent.java

```
public class UniversityStudent extends Student{
    private String university;
    private String faculty;
    public UniversityStudent(String name, int age, String university, String
faculty){
        super(name, age);
        this.university = university;
        this.faculty = faculty;
    }

    @Override
    public void study() {

        System.out.println("UniStudent study at " + university + "
University, at faculty of ” + faculty);
    }

    @Override
    public String getIdentification() {
        return "University: " + university;
    }
}
```

Задание 3

В задании 3 ЛР №4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов. 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.

Входные данные:

```
Client client = new Client( fullName: "Julie", account, creditCard);
Admin admin = new Admin( fullName: "John Smith");
```

Выходные данные:

```
Оплата заказа Julie прошла успешно
Текущий долг на карте: 500.0
Кредитная карта заблокирована Администратором John Smith
Карта разблокирована
Снято со счета: 200.0
Перевод денег со счета Julie прошел успешно: 200.0
Счет получателя: 200.0
Счет отправителя: 800.0
```

Код программы:

Admin.java

```
public class Admin extends Human {
    public Admin(String fullName){
        super(fullName);
    }

    public void blockCreditCardForExceedingCredit(CreditCard creditCard) {
            creditCard.block();
            System.out.println("Кредитная карта заблокирована Администратором
" + fullName);
    }
}
```

Client.java

```
public class Client extends Human{
    private Account account;
    private CreditCard creditCard;

    public Client(String fullName, Account account, CreditCard creditCard) {
        super(fullName);
        this.account = account;
        this.creditCard = creditCard;
    }

    public void makePayment(Order order) {
        double totalPrice = order.getTotalPrice();
        if (creditCard.isBlocked()){
            System.out.println("Карта заблокирована. Невозможно провести
операцию");
            return;
        }
        if( creditCard.getAvailableCredit() >= totalPrice) {
            creditCard.increaseDebt(totalPrice);
            order.setPaid(true);
            System.out.println("Оплата заказа " + fullName + " прошла
успешно");
```

```
        } else {
            System.out.println("Недостаточно средств на кредитной карте.");
        }
    }

    public void transferMoney(double amount, Account recipientAccount) {
        if (creditCard.isBlocked()){
            System.out.println("Карта заблокирована. Невозможно провести
операцию");
            return;
        }
        if (account.getBalance() >= amount) {
            account.withdraw(amount);
            recipientAccount.deposit(amount);
            System.out.println("Перевод денег со счета " +fullName + " прошел
успешно: " + amount);
        } else {
            System.out.println("Недостаточно средств на счете.");
        }
    }

    public void blockCreditCard() {
        creditCard.block();
        System.out.println("Кредитная карта заблокирована Клиентом " +
fullName);
    }

    public void cancelAccount() {
        account.cancel();
    }

}
```

Human.java

```
public class Human {
    protected String fullName;

    public Human(String fullName){
       this.fullName= fullName;
    }
}
```

Task3.java

```
public class Task3 {
    public static void main(String[] args) {
        Account account = new Account(1000);
        CreditCard creditCard = new CreditCard(2000);
        Client client = new Client("Julie", account, creditCard);
        Order order = new Order(500);
        client.makePayment(order);

        double debt = creditCard.getCurrentDebt();
        System.out.println("Текущий долг на карте: " + debt);

        Admin admin = new Admin("John Smith");
        admin.blockCreditCardForExceedingCredit(creditCard);

        creditCard.unblock();
```

```
        Account otherAcc = new Account(0);
        client.transferMoney(200,otherAcc);
        double otherBalance = otherAcc.getBalance();
        System.out.println("Счет получателя: " + otherBalance);
        double balance = account.getBalance();
        System.out.println("Счет отправителя: " + balance);

    }
}
```

Вывод: научилась создавать и использовать классы в программах на языке программирования Java.