Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №5**

По дисциплине: «Современные платформы программирования»

**Выполнил:**
студент 3 курса
группы ПО-8
Бубен С.О.
**Проверил:**
Крощенко А.А

Брест, 2024

Цель работы: приобрести базовые навыки работы с файловой системой в Java.

## 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.

**Работа программы**:

```
Applying for admission to Faculty faculty
Student (18) is studying...
Taking exams for Faculty faculty
```

**Код:**

### Main

```java
public class Main
{
    public static void main(String[] args)
    {
        StudentOfFaculty student1 = new StudentOfFaculty("Student", 18, "Faculty");

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

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

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

Student

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

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

```
    public void study()
    {
        System.out.println(name + " (" + age + ") is studying...");
    }

    public abstract void takeExams();
}
```

StudentOfFaculty

```
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 applyForAdmission()
    {
        System.out.println("Applying for admission to " + faculty + " faculty");
    }

    @Override
    public void takeExams()
    {
        System.out.println("Taking exams for " + faculty + " faculty");
    }
}
```

**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.

**Работа программы**:

```
Students study:
School student John (15) is studying
College student Emma (20) is studying
School student Michael (14) is studying

School Students:
Name: John, Age: 15, Grade: 10
Name: Michael, Age: 14, Grade: 9

College Students:
Name: Emma, Age: 20, Major: Computer Science
```

**Код:**

CollegeStudent

```
public class CollegeStudent extends Student
{
    private String major;

    public CollegeStudent(String name, int age, String major)
    {
        super(name, age);
        this.major = major;
    }

    @Override
    public void study()
    {
        System.out.println("College student " + getName() + " (" + getAge() + ") is
studying");
    }

    public String getMajor()
    {
        return major;
    }
}
```

Main

```
public class Main
{
    public static void main(String[] args)
    {
        Student[] students = new Student[3];
        students[0] = new SchoolStudent("John", 15, 10);
        students[1] = new CollegeStudent("Emma", 20, "Computer Science");
        students[2] = new SchoolStudent("Michael", 14, 9);

        System.out.println("Students study:");
```

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

        System.out.println("\nSchool Students:");
        for (Student student : students)
        {
            if (student instanceof SchoolStudent schoolStudent)
            {
                System.out.println("Name: " + schoolStudent.getName()
                        + ", Age: " + schoolStudent.getAge()
                        + ", Grade: " + schoolStudent.getGrade()
                );
            }
        }

        System.out.println("\nCollege Students:");
        for (Student student : students)
        {
            if (student instanceof CollegeStudent collegeStudent)
            {
                System.out.println("Name: " + collegeStudent.getName()
                        + ", Age: " + collegeStudent.getAge()
                        + ", Major: " + collegeStudent.getMajor()
                );
            }
        }
    }
}
```

**SchoolStudent**

```
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("School student " + getName() + " (" + getAge() + ") is studying");
    }

    public int getGrade()
    {
        return grade;
    }
}
```

Student

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

    public Student(String name, int age)
```

```
    {
        this.name = name;
        this.age = age;
    }

    public abstract void study();

    public String getName()
    {
        return name;
    }

    public int getAge()
    {
        return age;
    }
}
```

**Задание 3.**

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

**Работа программы**:

```
Оплата заказа Z123 клиентом Иванов
Заказ №Z123 уже оплачен
Выполнение платежа на счет 0987654321 в размере 300.0
Нехватает баланса
Блокировка кредитной карты 9876543210 администратором Сергей
Карточка клиента Иванов заблокирована!
Блокировка кредитной карты клиента Иванов
Аннулирование счета клиента Иванов
У клиента Иванов нет счета
```

**Код:**

Accaunt

```
public class Account
{
    private String accountNumber;
    private double balance;

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

    public String getAccountNumber()
    {
        return accountNumber;
    }
```

```
    public double getBalance()
    {
        return balance;
    }

    public boolean pay(double amount)
    {
        if (this.balance - amount < 0)
        {
            return false;
        }

        this.balance -= amount;
        return true;
    }
}
```

Administrator

```
public class Administrator extends Human
{
    public Administrator(String name, int age, String address)
    {
        super(name, age, address);
    }

    public void blockCreditCard(CreditCard card)
    {
        card.setBlocked(true);
        System.out.println("Блокировка кредитной карты " + card.getCardNumber() + " администратором " +
name);
    }
}
```

Client

```
public class Client extends Human
{
    @SuppressWarnings("unused")
    private Account account;
    private CreditCard creditCard;

    public Client(String name, int age, String address)
    {
        super(name, age, address);
    }


    @SuppressWarnings("unused")
    public String getAddress()
    {
        return address;
    }

    public void setAccount(Account account)
    {
        this.account = account;
    }

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


    public void payOrder(Order order)
    {
        Account acc;
```

```
        if (this.creditCard != null)
        {
            acc = this.creditCard.getAccount();
        } else if (this.account != null)
        {
            acc = this.account;
        } else
        {
            System.out.println("У клиента " + name + " нет счета");
            return;
        }

        if (acc == null)
        {
            System.out.println("Карточка клиента " + this.name + " заблокирована!");
            return;
        }

        if (order.isPayed())
        {
            System.out.println("Заказ №" + order.getOrderNumber() + " уже оплачен");
            return;
        }

        if (!acc.pay(order.getOrderAmount()))
        {
            System.out.println("Нехватает баланса");
            return;
        }
        order.ConfirmOrder();

        System.out.println("Оплата заказа " + order.getOrderNumber() + " клиентом " + this.name);
    }
    public void makePayment(Account targetAccount, double amount)
    {
        Account acc;
        if (this.creditCard != null)
        {
            acc = this.creditCard.getAccount();
        } else if (this.account != null)
        {
            acc = this.account;
        } else
        {
            System.out.println("У клиента " + name + " нет счета");
            return;
        }

        if (acc == null)
        {
            System.out.println("Карточка клиента " + this.name + " заблокирована!");
            return;
        }

        if (!acc.pay(amount))
        {
            System.out.println("Нехватает баланса");
            return;
        }
        targetAccount.pay(-amount);

        System.out.println("Выполнение платежа на счет " + targetAccount.getAccountNumber() + " в размере
" + amount);
    }
```

```
    public void blockCreditCard()
    {
        this.creditCard.setBlocked(true);
        System.out.println("Блокировка кредитной карты клиента " + name);
    }


    public void cancelAccount()
    {
        this.account = null;
        this.creditCard = null;

        System.out.println("Аннулирование счета клиента " + name);
    }
}
```

CreditCard

```
public class CreditCard
{
    private String cardNumber;
    private double creditLimit;
    private Account account;
    private boolean isBlocked;

    public CreditCard(String cardNumber, Account account, double creditLimit)
    {
        this.cardNumber = cardNumber;
        this.account = account;
        this.creditLimit = creditLimit;
        this.isBlocked = false;
    }

    public String getCardNumber()
    {
        return cardNumber;
    }

    @SuppressWarnings("unused")
    public double getCreditLimit()
    {
        return creditLimit;
    }

    public Account getAccount()
    {
        return (!this.isBlocked) ? account : null;
    }

    public void setBlocked(boolean blocked)
    {
        isBlocked = blocked;
    }
}
```

Human

```
public class Human
{
    protected String name;
    protected int age;
    protected String address;

    public Human(String name, int age, String address)
    {
        this.name = name;
        this.age = age;
        this.address = address;
```

```
    }
}
```

Main

```
public class Main
{
    public static void main(String[] args)
    {
        Task3.exec();
    }
}
```

Order

```
public class Order
{
    private String orderNumber;
    private double orderAmount;
    private boolean isPayed;

    public Order(String orderNumber, double orderAmount)
    {
        this.orderNumber = orderNumber;
        this.orderAmount = orderAmount;
        this.isPayed = false;
    }

    public String getOrderNumber()
    {
        return orderNumber;
    }

    public double getOrderAmount()
    {
        return orderAmount;
    }

    public void ConfirmOrder()
    {
        this.isPayed = true;
    }

    public boolean isPayed()
    {
        return isPayed;
    }
}
```

Task3

```
public class Task3
{
    public static void exec()
    {
        Client client = new Client("Иванов", 30, "ул. Пушкина, д.10");

        Account account = new Account("1234567890", 1000.0);
        client.setAccount(account);

        CreditCard creditCard = new CreditCard("9876543210", account, 5000.0);
        client.setCreditCard(creditCard);

        Order order = new Order("Z123", 500.0);

        client.payOrder(order);
        client.payOrder(order);
```

```
        Account otherAccount = new Account("0987654321", 2000.0);
        client.makePayment(otherAccount, 300.0);
        client.makePayment(otherAccount, 300.0);

        Administrator administrator = new Administrator("Сергей", 28, "ул. Пушкина, д.16");
        administrator.blockCreditCard(creditCard);

        client.payOrder(order);

        client.blockCreditCard();

        client.cancelAccount();
        client.payOrder(order);
    }
}
```

**Вывод:** исследовал создание классов и объектное-ориентированное программирование на языке программирования Java, приобрел практические навыки в области объектно-ориентированного проектирования в использовании суперклассов и интерфейсов.