МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №5

Специальность ПО

Выполнил
Войтюк Е.О.
студент группы ПО-8
Проверил
А. А. Крощенко,
ст. преп. кафедры ИИТ,
«___k_____2024 г.

Брест 2024

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

Задание 1. Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

6) interface Mobile ← abstract class Samsung Mobile ← class Model.

Выполнение:

Код программы

```
interface Mobile {
    void call(String number);
    void sendMessage(String number, String message);
}

abstract class SamsungMobile implements Mobile {
    String model;

    public SamsungMobile(String model) {
        this.model = model;
    }

    @Override
    public void call(String number) {
        System.out.println("Звоним на номер " + number + " с телефона Samsung " + model);
    }

    @Override
    public void sendMessage(String number, String message) {
        System.out.println("Отправляем сообщение '" + message + "' на номер " + number + " с телефона
Samsung " + model);
    }

    abstract void useSamsungPay();
}

class Model extends SamsungMobile {
    public Model(String model) {
        super(model);
    }

    @Override
    void useSamsungPay() {
        System.out.println("Оплата через Samsung Pay с телефона Samsung " + model);
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Model myPhone = new Model("Galaxy S20");

        myPhone.call("1234567890");
        myPhone.sendMessage("1234567890", "Привет!");

        myPhone.useSamsungPay();
    }
}
```

Рисунки с результатами работы программы

```
Звоним на номер 1234567890 с телефона Samsung Galaxy S20
Отправляем сообщение 'Привет!' на номер 1234567890 с телефона Samsung Galaxy S20
Оплата через Samsung Pay с телефона Samsung Galaxy S20

Process finished with exit code 0
```

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. 6) Создать суперкласс Домашнее животное и подклассы Собака, Кошка, Попугай. С помощью конструктора установить имя каждого животного и его характеристики.

Выполнение:

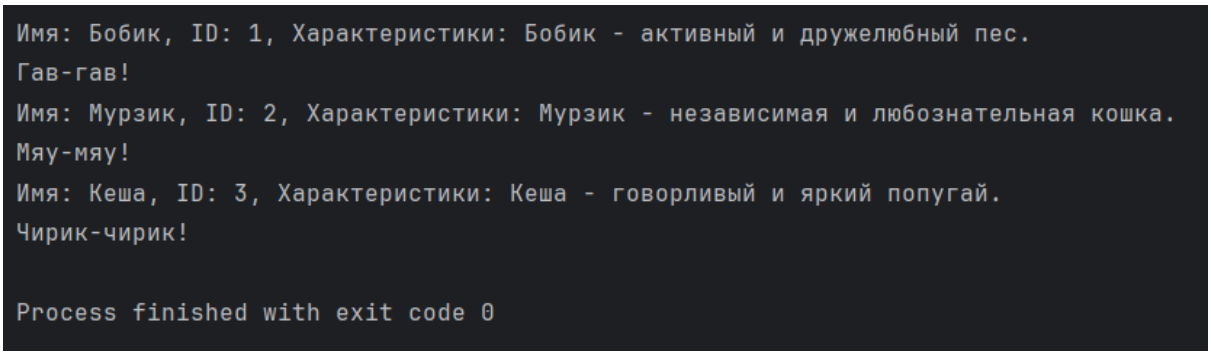
Код программы

```
abstract class DomesticAnimal {
    String name;
    String id;
    String characteristics;
```

```
    public DomesticAnimal(String name, String id, String characteristics) {
        this.name = name;
        this.id = id;
        this.characteristics = characteristics;
    }

    abstract void sound();
}

class Dog extends DomesticAnimal {
    public Dog(String name, String id, String characteristics) {
        super(name, id, characteristics);
    }

    @Override
    void sound() {
        System.out.println("Гав-гав!");
    }
}

class Cat extends DomesticAnimal {
    public Cat(String name, String id, String characteristics) {
        super(name, id, characteristics);
    }

    @Override
    void sound() {
        System.out.println("Мяу-мяу!");
    }
}

class Parrot extends DomesticAnimal {
    public Parrot(String name, String id, String characteristics) {
        super(name, id, characteristics);
    }

    @Override
    void sound() {
        System.out.println("Чирик-чирик!");
    }
}

public class Main {
    public static void main(String[] args) {
        DomesticAnimal[] pets = new DomesticAnimal[3];

        pets[0] = new Dog("Бобик", "1", "Бобик - активный и дружелюбный пес.");
        pets[1] = new Cat("Мурзик", "2", "Мурзик - независимая и любознательная кошка.");
        pets[2] = new Parrot("Кеша", "3", "Кеша - говорливый и яркий попугай.");

        for (DomesticAnimal pet : pets) {
```

```
            System.out.println("Имя: " + pet.name + ", ID: " + pet.id + ", Характеристики: " +
pet.characteristics);
            pet.sound();
        }
    }
}
```

Рисунки с результатами работы программы



Задание 3.

В задании 3 ЛР No4, где возможно, заменить объявления суперклассов объявлениями абстрактных

классов или интерфейсов.

Выполнение:

Код программы

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

interface Payable {
    double getAmount();
    void setAmount(double amount);
}

class Arrears implements Payable {
    private double amount;

    public Arrears(double amount) {
        this.amount = amount;
    }
```

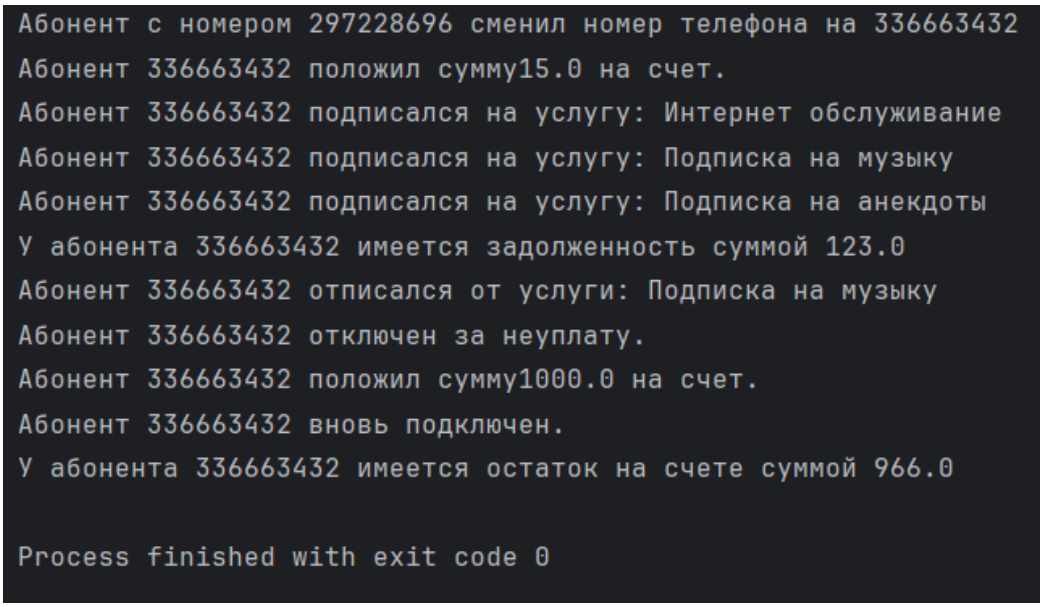
```
    public double getAmount() {
        return amount;
    }

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

interface Serviceable {
    String getName();
    double getPrice();
}

class Service implements Serviceable {
    private String name;
    private double price;

    public Service(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }
}

abstract class User {
    public abstract void accountAmount();
    public abstract boolean checkUnpaidArrears();
}

class Subscriber extends User {
    public String phoneNumber;
    public List<Service> services;
    public Arrears arrears;
    public boolean isActive;

    public Subscriber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
        this.services = new ArrayList<>();
        this.arrears = new Arrears(0);
        this.isActive = true;
    }

    public void requestPhoneNumberChange(Administrator administrator, String newNumber) {
```

```
        administrator.changePhoneNumber(this, newNumber);
    }

    public void requestService(Administrator administrator, Service service) {
        administrator.requestService(this, service);
    }

    public void cancelService(Administrator administrator, Service service) {
        administrator.cancelService(this, service);
    }

    public void payArrears(double amount) {
        this.arrears.setAmount(this.arrears.getAmount() - amount);
        System.out.println("Абонент " + this.phoneNumber + " положил сумму" + amount + " на счет.");
        if(!this.isActive && !checkUnpaidArrears())
        {
            this.isActive = true;
            System.out.println
                    ("Абонент " + this.phoneNumber + " вновь подключен.");
        }
    }

    public void accountAmount()
    {
        if(checkUnpaidArrears())
        {
            System.out.println
                    ("У абонента " + this.phoneNumber + " имеется задолженность суммой " +
this.arrears.getAmount());
        }
        else
        {
            System.out.println
                    ("У абонента " + this.phoneNumber + " имеется остаток на счете суммой " + (-
this.arrears.getAmount()));
        }
    }

    public boolean checkUnpaidArrears() {
        return this.arrears.getAmount() > 0;
    }
}

interface AdminActions {
    void changePhoneNumber(Subscriber subscriber, String newNumber);
    void requestService(Subscriber subscriber, Service service);
    void cancelService(Subscriber subscriber, Service service);
    void temporarilyDisableSubscriber(Subscriber subscriber);
}

class Administrator implements AdminActions {
    public void changePhoneNumber(Subscriber subscriber, String newNumber) {
```

```
        System.out.println("Абонент с номером " + subscriber.phoneNumber + " сменил номер телефона на " + newNumber);
        subscriber.phoneNumber = newNumber;
    }

    public void requestService(Subscriber subscriber, Service service) {
        subscriber.services.add(service);
        subscriber.arrears.setAmount(subscriber.arrears.getAmount() + service.getPrice());
        System.out.println("Абонент " + subscriber.phoneNumber + " подписался на услугу: " + service.getName());
    }

    public void cancelService(Subscriber subscriber, Service service) {
        if (subscriber.services.contains(service)) {
            subscriber.services.remove(service);
            subscriber.arrears.setAmount(subscriber.arrears.getAmount() - service.getPrice());
            System.out.println("Абонент " + subscriber.phoneNumber + " отписался от услуги: " + service.getName());
        }
    }

    public void temporarilyDisableSubscriber(Subscriber subscriber) {
        if (subscriber.checkUnpaidArrears()) {
            subscriber.isActive = false;
            System.out.println("Абонент " + subscriber.phoneNumber + " отключен за неуплату.");
        }
    }
}

class TelephoneStation {
    private List<Subscriber> subscribers;

    public TelephoneStation() {
        this.subscribers = new ArrayList<>();
    }

    public void addSubscriber(Subscriber subscriber) {
        this.subscribers.add(subscriber);
    }

    public List<Subscriber> getSubscribers() {
        return subscribers;
    }
}

public class Main3 {
    public static void main(String[] args) {
        TelephoneStation telephoneStation = new TelephoneStation();

        Service service1 = new Service("Интернет обслуживание", 34.0);
        Service service2 = new Service("Подписка на музыку", 89.0);
        Service service3 = new Service("Подписка на анекдоты", 15.0);
```

```
        Subscriber subscriber1 = new Subscriber("297228696");

        telephoneStation.addSubscriber(subscriber1);

        Administrator administrator = new Administrator();

        subscriber1.requestPhoneNumberChange(administrator, "336663432");

        subscriber1.payArrears(15);

        subscriber1.requestService(administrator, service1);
        subscriber1.requestService(administrator, service2);
        subscriber1.requestService(administrator, service3);

        subscriber1.accountAmount();

        subscriber1.cancelService(administrator, service2);

        administrator.temporarilyDisableSubscriber(subscriber1);

        subscriber1.payArrears(1000);

        subscriber1.accountAmount();
    }
}
```

Рисунки с результатами работы программы



Вывод: приобрел практические навыки в области объектно-ориентированного проектирования