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

Отчёт
по лабораторной работе №5

Выполнил:
студент группы ПО-9
Зеленков К. И.

Проверил:
Крощенко А. А.

Брест 2024

Вариант 6

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.

Код программы:

Main1.java:

```
import java.util.ArrayList;
import java.util.List;
interface Mobile {
    void makeCall(String phoneNumber);
    void sendMessage(String phoneNumber, String message);
    void playMusic(String song);
}
abstract class IphoneMobile implements Mobile {
    String model;
    public IphoneMobile(String model) {
        this.model = model;
    }
    @Override
    public void makeCall(String phoneNumber) {
        System.out.println("Звонок с " + model + " на " + phoneNumber);
    }
    @Override
    public void sendMessage(String phoneNumber, String message) {
        System.out.println("Отправка сообщения с " + model + " на " +
phoneNumber + ": " + message);
    }
    abstract void installApp(String appName);
}
class Model extends IphoneMobile {
    private final int storageCapacity;
    private List<String> installedApps;
    public Model(String model, int storageCapacity) {
        super(model);
        this.storageCapacity = storageCapacity;
        this.installedApps = new ArrayList<>();
    }
    public int getStorageCapacity() {
        return storageCapacity;
    }
    public void installApp(String appName) {
        installedApps.add(appName);
        System.out.println("Установка приложения " + appName + " на " +
model);
    }
    public void uninstallApp(String appName) {
        if (installedApps.contains(appName)) {
            installedApps.remove(appName);
            System.out.println("Удаление приложения " + appName + " с " +
model);
```

```
        } else {
            System.out.println("Приложение " + appName + " не установлено на
устройстве " + model);
        }
    }
    @Override
    public void playMusic(String song) {
        System.out.println("Воспроизведение музыки " + song + " на " +
model);
    }
}
public class Main1 {
    public static void main(String[] args) {
        Model iphone11 = new Model("Iphone 11 pro", 32);
        System.out.println("Объём памяти: " + iphone11.getStorageCapacity()
+ " GB");
        iphone11.makeCall("+375297228696");
        iphone11.sendMessage("+375297941646", "Как дела?");
        iphone11.installApp("ITunes");
        iphone11.installApp("YouTube");
        iphone11.uninstallApp("ITunes");
        iphone11.uninstallApp("Google Chrome");
        iphone11.playMusic("Beethoven - 3 Piano Sonata");
    }
}
```

Результат работы программы:

```
Объём памяти: 32 GB
Звонок с Iphone 11 pro на +375297228696
Отправка сообщения с Iphone 11 pro на +375297941646: Как дела?
Установка приложения ITunes на Iphone 11 pro
Установка приложения YouTube на Iphone 11 pro
Удаление приложения ITunes с Iphone 11 pro
Приложение Google Chrome не установлено на устройстве Iphone 11 pro
Воспроизведение музыки Beethoven - 3 Piano Sonata на Iphone 11 pro
```

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.

Код программы

Main2.java:

```
interface Pet {
    void setName(String name);
    String getName();
    void play();
    void feed(int foodAmount);
    void setHealth(int health);
    int getHealth();
}

abstract class Animal implements Pet {
    protected String name;
    protected int hunger;
    protected int health;

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

    public String getName() {
        return name;
    }

    public void feed(int foodAmount) {
        hunger -= foodAmount;
        if (hunger < 0) {
            hunger = 0;
        }
        System.out.println(name + " был накормлен. Уровень голода: " +
hunger);
    }

    public void setHealth(int health) {
        this.health = health;
    }

    public int getHealth() {
        return health;
    }

    public abstract void play();
}

class Dog extends Animal {
    private final String breed;

    public Dog(String name, String breed) {
        this.name = name;
        this.breed = breed;
        this.hunger = 50;
        this.health = 100;
    }

    @Override
    public void play() {
        System.out.println("Собака " + name + " играет с игрушечным
котом.");
    }

    public String getBreed() {
        return breed;
    }
}
```

```
class Cat extends Animal {
    private final String breed;
    public Cat(String name, String breed) {
        this.name = name;
        this.breed = breed;
        this.hunger = 40;
        this.health = 90;
    }

    @Override
    public void play() {
        System.out.println("Кот " + name + " играет с игрушечной мышью.");
    }

    public String getBreed() {
        return breed;
    }
}

class Parrot extends Animal {
    private final String color;

    public Parrot(String name, String color) {
        this.name = name;
        this.color = color;
        this.hunger = 60;
        this.health = 95;
    }

    @Override
    public void play() {
        System.out.println("Попугай " + name + " играет, разговаривая
человеческим голосом.");
    }

    public String getColor() {
        return color;
    }
}

public class Main2 {
    public static void main(String[] args) {
        Animal[] pets = new Animal[3];

        pets[0] = new Dog("Чарли", "Кокер-спаниель");
        pets[1] = new Cat("Оскар", "Британец");
        pets[2] = new Parrot("Кеша", "Желто-зеленый");

        for (Animal pet : pets) {
            System.out.println("Имя: " + pet.getName());
            pet.play();
            pet.feed(20);
            System.out.println("Здоровье: " + pet.getHealth());
        }
    }
}
```

Результат работы программы:

```
Имя: Чарли
Собака Чарли играет с игрушечным котом.
Чарли был накормлен. Уровень голода: 30
Здоровье: 100
Имя: Оскар
Кот Оскар играет с игрушечной мышью.
Оскар был накормлен. Уровень голода: 20
Здоровье: 90
Имя: Кеша
Попугай Кеша играет, разговаривая человеческим голосом.
Кеша был накормлен. Уровень голода: 40
Здоровье: 95
```

Задание 3

В задании 3 ЛР No4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

6) Система Телефонная станция. Абонент оплачивает Счет за разговоры и Услуги, может попросить Администратора сменить номер и отказаться от услуг. Администратор изменяет номер, Услуги и временно отключает Абонента за неуплату.

Код программы

Main3.java:

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

interface Subscriber {
    void requestPhoneNumberChange(TelephoneAdministrator administrator,
String newNumber);
    void requestService(TelephoneAdministrator administrator, Service
service);
    void cancelService(TelephoneAdministrator administrator, Service
service);
    void payArrears(double amount);
    void accountAmount();
    boolean checkUnpaidBill();
}

class Bill {
    private double amount;

    public Bill(double amount) {
        this.amount = amount;
    }

    public double getAmount() {
        return amount;
    }

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

class Service {
```

```
    private final String name;
    private final double price;

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

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }
}

class TelephoneSubscriber implements Subscriber {
    public String phoneNumber;
    public List<Service> services;
    public Bill bill;
    public boolean isActive;

    public TelephoneSubscriber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
        this.services = new ArrayList<>();
        this.bill = new Bill(0);
        this.isActive = true;
    }

    public void requestPhoneNumberChange(TelephoneAdministrator
administrator, String newNumber) {
        administrator.changePhoneNumber(this, newNumber);
    }

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

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

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

    public void accountAmount()
    {
        if(checkUnpaidBill())
        {
            System.out.println
                    ("У абонента " + this.phoneNumber + " имеется
задолженность суммой " + this.bill.getAmount());
        }
```

```
        else
        {
            System.out.println
                    ("У абонента " + this.phoneNumber + " имеется остаток на
счете суммой " + (-this.bill.getAmount()));
        }
    }

    public boolean checkUnpaidBill() {
        return this.bill.getAmount() > 0;
    }
}

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

class TelephoneAdministrator implements Administrator {
    public void changePhoneNumber(TelephoneSubscriber subscriber, String
newNumber) {
        System.out.println("Абонент с номером " + subscriber.phoneNumber + "
сменил номер телефона на " + newNumber);
        subscriber.phoneNumber = newNumber;
    }

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

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

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

class TelephoneStation {
    private List<TelephoneSubscriber> subscribers;

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

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

    public List<TelephoneSubscriber> 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);
        TelephoneSubscriber subscriber1 = new
TelephoneSubscriber("297228696");

        telephoneStation.addSubscriber(subscriber1);

        TelephoneAdministrator administrator = new TelephoneAdministrator();

        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();
    }
}
```

Результат работы программы:

```
Абонент с номером 297228696 сменил номер телефона на 336663432
Абонент 336663432 положил 15.0 на счет.
Абонент 336663432 подписался на услугу: Интернет обслуживание
Абонент 336663432 подписался на услугу: Подписка на музыку
Абонент 336663432 подписался на услугу: Подписка на анекдоты
У абонента 336663432 имеется задолженность суммой 123.0
Абонент 336663432 отказался от услуги: Подписка на музыку
Абонент 336663432 отключен за неуплату.
Абонент 336663432 положил 1000.0 на счет.
Абонент 336663432 вновь подключен после уплаты задолженности
У абонента 336663432 имеется остаток на счете суммой 966.0
```

Вывод: Приобрёл практические навыки в области объектно-ориентированного проектирования в языке Java.