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

Отчёт
по лабораторной работе №5

Выполнил:
студент группы ПО-9
Солышко Дмитрий Андреевич

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:

Task_01.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 SamsungMobile implements Mobile {
    String model;

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

    @Override
    public void makeCall(String phoneNumber) {
        System.out.println("Making call from " + model + " to " + phoneNumber);
    }

    @Override
    public void sendMessage(String phoneNumber, String message) {
        System.out.println("Sending message from " + model + " to " + phoneNumber +
": " + message);
    }

    abstract void installApp(String appName);
}

class Model extends SamsungMobile {
    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("Installing app " + appName + " on " + model);
    }

    public void uninstallApp(String appName) {
        if (installedApps.contains(appName)) {
            installedApps.remove(appName);
            System.out.println("Uninstalling app " + appName + " from " + model);
        } else {
            System.out.println("App " + appName + " is not installed on " + model);
```

```
        }
    }

    @Override
    public void playMusic(String song) {
        System.out.println("Playing music " + song + " on " + model);
    }
}

public class Task_01 {
    public static void main(String[] args) {
        Model samsungGalaxy = new Model("Samsung Galaxy", 32);
        System.out.println("StorageCapacity: " + samsungGalaxy.getStorageCapacity() +
" GB");
        samsungGalaxy.makeCall("+375333283757");
        samsungGalaxy.sendMessage("+375333283757", "Hello!");
        samsungGalaxy.installApp("Play Market");
        samsungGalaxy.installApp("YouTube");
        samsungGalaxy.uninstallApp("Play Market");
        samsungGalaxy.playMusic("MySong");
    }
}
```

Результат работы программы:

```
StorageCapacity: 32 GB
Making call from Samsung Galaxy to +375333283757
Sending message from Samsung Galaxy to +375333283757: Hello!
Installing app Play Market on Samsung Galaxy
Installing app YouTube on Samsung Galaxy
Uninstalling app Play Market from Samsung Galaxy
Playing music MySong on Samsung Galaxy
```

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

Task_02.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 + " has been fed. Hunger level: " + 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("The dog " + name + "is playing with a toy bone");
    }

    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("Cat " + name + " is playing with a toy mouse.");
    }

    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("Parrot " + name + " is mimicking human speech.");
    }

    public String getColor() {
        return color;
    }
}

public class Task_02 {
    public static void main(String[] args) {
        Animal[] pets = new Animal[3];

        pets[0] = new Dog("Sharik", "Pomeranian spitz");
        pets[1] = new Cat("Barsik", "no breed");
        pets[2] = new Parrot("kesha", "Green");

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

Результат работы программы:

```
Name: Sharik
The dog Sharikis playing with a toy bone
Sharik has been fed. Hunger level: 30
Health: 100
Name: Barsik
Cat Barsik is playing with a toy mouse.
Barsik has been fed. Hunger level: 20
Health: 90
Name: kesha
Parrot kesha is mimicking human speech.
kesha has been fed. Hunger level: 40
Health: 95
```

Задание 3

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

Задание 3 ЛР No4: Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

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

Task_03.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 payBill(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 payBill(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 + " обратно подключен после уплаты
задолженности");
        }
    }

    public void accountAmount()
    {
        if(checkUnpaidBill())
        {
            System.out.println
                    ("у абонента " + this.phoneNumber + " до сих пор есть
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 + " временно
отключен за неуплату.");
        }
    }
}
```

```java
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 Task_03 {
    public static void main(String[] args) {
        TelephoneStation telephoneStation = new TelephoneStation();

        Service service1 = new Service("Интернет", 30.0);
        Service service2 = new Service("Кабельное телевидение", 50.0);
        Service service3 = new Service("Музыка", 40.0);
        TelephoneSubscriber subscriber1 = new TelephoneSubscriber("123456789");

        telephoneStation.addSubscriber(subscriber1);

        TelephoneAdministrator administrator = new TelephoneAdministrator();

        subscriber1.requestPhoneNumberChange(administrator, "987654321");

        subscriber1.payBill(50);

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

        subscriber1.accountAmount();

        subscriber1.cancelService(administrator, service2);

        administrator.temporarilyDisableSubscriber(subscriber1);

        subscriber1.payBill(80);
    }
}
```

Результат работы программы:

```
Абонент с номером 123456789 сменил номер телефона на 987654321
Абонент 987654321 положил 50.0 на счет
Абонент 987654321 подписался на услугу: Интернет
Абонент 987654321 подписался на услугу: Кабельное телевидение
Абонент 987654321 подписался на услугу: Музыка
у абонента 987654321 до сих пор есть задолженность суммой 70.0
Абонент 987654321 отказался от услуги: Кабельное телевидение
абонент 987654321 временно отключен за неуплату.
Абонент 987654321 положил 80.0 на счет
абонент 987654321 обратно подключен после уплаты задолженности
```

Вывод: я приобрёл практические навыки в области объектно-ориентированного проектирования