Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Кафедра «Математической кибернетики и информационных технологий»

Информационные технологии и программирование

Лабораторная работа №2

Выполнил: студент группы

БВТ2306

Кесслер Алексей Сергеевич

Москва, 2024 г.

Цель работы: Изучение основных концепция ООП на примере языка программирования java, а именно: Абстракция, Инкапсуляция, Полиморфизм, Наследование.

Задача работы: Создание собственной иерархии классов в соответствии с вариантом.

Выполнение

Мой вариант: 15 Базовый класс: Компьютерная периферия Дочерние классы: Клавиатура, Наушники, Графический планшет.

Для начала создадим абстрактный класс для Компьютерной периферии:

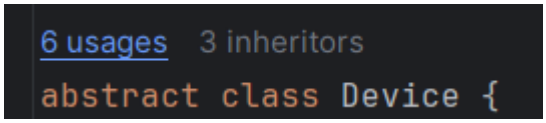

Рисунок 1 – Абстрактный класс

2. Теперь создадим классы-наследники, которые расширяют наш абстрактный класс.

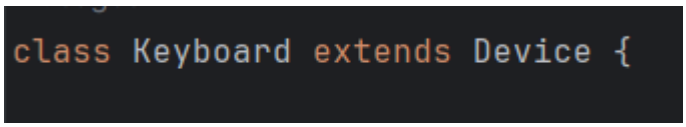

Рисунок 2 – Класс Клавиатуры

Рисунок 3 – Класс Наушников

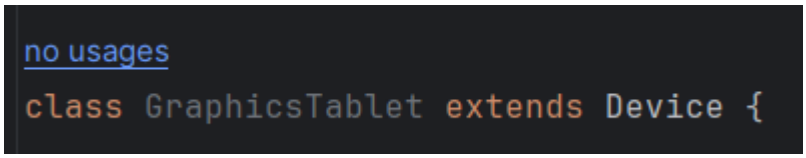

Рисунок 4 – Класс Графического планшета

3. Заполним все существующие классы полями и методами.

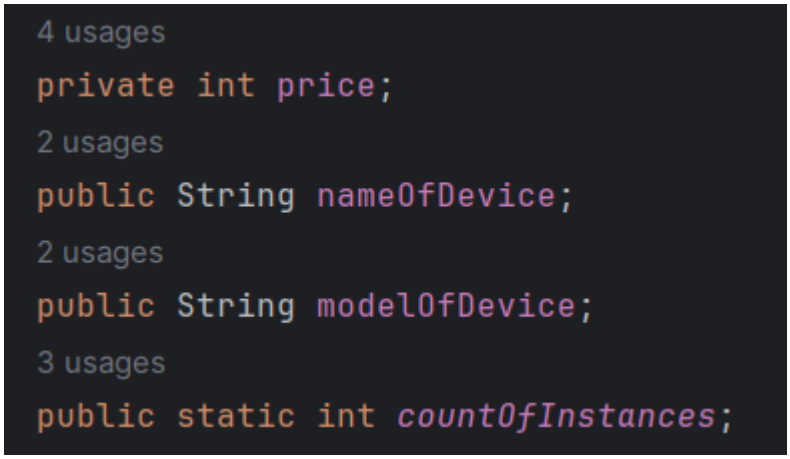

Рисунок 5 – Поля абстрактного класса

```
private boolean isLighted;

2 usages
public float width;
2 usages
public float height;
11 usages
private String printedLine;
```

Рисунок 6 – Поля класса Keyboard

```
2 usages
private boolean isWireless;

2 usages
public float frequencyRange;
2 usages
public float maxPower;
```

Рисунок 7 – Поля класса Headphones

```
2 usages
public long width;
2 usages
public long height;

2 usages
public float tabletResolution;
```

Рисунок 8 – Поля класса GraphicTablet

```
no usages  3 overrides
public void buyDevice() {
    System.out.println("You have bought this very useful device... but I`m not sure what it
}

no usages
public void showDetails() {
    System.out.println("As you buying this stuff you have been agreed with our rules," +
            " do not cross the line");
}
```

Рисунок 9 – Методы абстрактного класса

```
@Override
public void buyDevice() {
    System.out.println("You have bought this very useful device and it is a keyboard!");
}

no usages
public String getPrintedLine() {
    return this.printedLine;
}

no usages
public void editPrintedLine(String s) {
    if (!s.isEmpty()) {
        this.printedLine = this.printedLine.substring(0, printedLine.length()-1);
    }
    else {
        this.printedLine += s;
    }
}

no usages
public void editPrintedLine() {
    if (!this.printedLine.isEmpty()) {
        this.printedLine = this.printedLine.substring(0, printedLine.length()-1);
    }
}
```

Рисунок 10 – Методы класса Keyboard

Заметим, что в классе Keyboard мы уже пользуемся перегрузкой и переопределением методов java

```
no usages
@Override
public void buyDevice() {
    System.out.println("You have bought this very useful device and there are a headphones!");
}

no usages
public void listenMusic() {
    System.out.println("Sorry, bro, this headphones can only play songs of Metallica");
}
```

Рисунок 11 – Методы Headphones

```
@Override
public void buyDevice() {
    System.out.println("You have bought this very useful device and it is a graphic tablet!");
}

no usages
public void showNicePict() {
    String s = "..... (¯`v´¯)♥\n" +
            ".......•.¸.•´\n" +
            ".....¸.•´\n" +
            "... (\n" +
            "☻/\n" +
            "/▌♥♥\n" +
            "/ \\ ♥♥\n";
    System.out.println(s);
}

no usages
public void showNotNicePict() {
```

Рисунок 12 – Методы GraphicTablet

Метод showNotNicePict вам лучше не видеть…

4. Создадим конструкторы классов, в том числе и конструкторы по умолчанию.

```java
public Device(int price, String nameOfDevice, String modelOfDevice) {
    this.price = price;
    this.nameOfDevice = nameOfDevice;
    this.modelOfDevice = modelOfDevice;
    Device.countOfInstances++;
}

3 usages
public Device() {
    this.price = -1;
    this.nameOfDevice = "Not assigned";
    this.modelOfDevice = "Not assigned";
    Device.countOfInstances++;
}
```

Рисунок 13 – Конструкторы абстрактного класса

```java
1 usage
public Keyboard(boolean isLighted, float width, float height,
                int price, String nameOfDevice, String modelOfDevice) {
    super(price, nameOfDevice, modelOfDevice);
    this.isLighted = isLighted;
    this.width = width;
    this.height = height;
    printedLine = "";
}

1 usage
public Keyboard() {
    super();
    this.isLighted = false;
    this.width = 0F;
    this.height = 0F;
    printedLine = "";
}
```

Рисунок 14 – Конструкторы класса Keyboard

```java
public Headphones(boolean isWireless, float frequencyRange, float maxPower,
                  int price, String nameOfDevice, String modelOfDevice) {
    super(price, nameOfDevice, modelOfDevice);
    this.isWireless = isWireless;
    this.frequencyRange = frequencyRange;
    this.maxPower = maxPower;
}

no usages
public Headphones() {
    super();
    this.isWireless = false;
    this.frequencyRange = 0F;
    this.maxPower = 0F;
}
```

Рисунок 15 – Конструкторы класса Headphones

```java
public GraphicsTablet(long width, long height, float tabletResolution,
                      int price, String nameOfDevice, String modelOfDevice) {
    super(price, nameOfDevice, modelOfDevice);
    this.width = width;
    this.height = height;
    this.tabletResolution = tabletResolution;
}

no usages
public GraphicsTablet() {
    super();
    this.width = 0;
    this.height = 0;
    this.tabletResolution = 0F;
}
```

Рисунок 16 – Конструкторы класса GraphicTablet

5. Создадим геттер и сеттер для цены девайса

```java
public int getPrice() {
    return this.price;
}

no usages
public void setPrice(int price) {
    this.price = price;
}
```

Рисунок 17 – Геттер и сеттер класса Device

6. Создадим статическую переменную, которая будет нам показывать количество создаваемых объектов класса Device

```java
public static int countOfInstances;

3 usages
public Device(int price, String nameOfDevice, String modelOfDevice) {
    this.price = price;
    this.nameOfDevice = nameOfDevice;
    this.modelOfDevice = modelOfDevice;
    Device.countOfInstances++;
}

3 usages
public Device() {
    this.price = -1;
    this.nameOfDevice = "Not assigned";
    this.modelOfDevice = "Not assigned";
    Device.countOfInstances++;
}
```

Рисунок 18 – Статическое поле countOfInstances

Проверяем работу этой переменной

```java
Keyboard k = new Keyboard();
Keyboard K = new Keyboard( isLighted: false,  width: 100F,  height: 50F

System.out.println(Device.countOfInstances);
```

Рисунок 19 – countOfInstances в деле

Переменная вывела 2, результат верный.

GitHub репозиторий - https://github.com/LightninG8/MTUCI/tree/main/%D0%98%D0%A2%D0%98%D0%9F/%D0%9B%D0%B0%D0%B1%D1%8B/%D0%9B%D0%B0%D0%B1%D0%B0%202