https://developer.alexanderklimov.ru/android/java/class.php
ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ.
Осуществлять разработку кода программного модуля на современных языках программирования
Оформлять документацию на программные средства
Основные этапы разработки программного обеспечения
Основные принципы технологии структурного и объектно-ориентированного программирования

Цель: изучение понятия объектно-ориентированное программирование в java
Задача: Раскрыть содержание знаний и понятий объектно-ориентированного программирования.

Понятие	Знал	Узнал	Хочу узнать

Java это (определение записано в тетради)
Java - это объектно-ориентированный язык,
Поэтому? код в ваших программах будет состоять из объектов и классов.
Объектно-ориентированное программирование. Классы. Объекты
Объектно-ориентированное программирование (ООП) — это метод программирования, при использовании которого главными элементами программ являются объекты.
Классы и объекты
Java - это объектно-ориентированный язык, поэтому код в ваших программах будет состоять из объектов и классов.
Классы
Java позволяет создавать классы, которые представляют объекты из реального мира. Например, можно создать класс **Car** (автомобиль) или **Animal** (животное) и задать им различные свойства. Для класса **Car** логично создать такие свойства как двери, колёса, лобовое стекло и т.д. Имея класс **Car**, можно создать новые классы Легковушки, Грузовики, Автобусы, которые будут иметь все свойства класса **Car**, а также свои собственные свойства.
У класса **Animal** соответственно можно задать свойства Лапы, Хвост, а затем создать наш любимый класс **Cat**, у которого будет ещё дополнительное свойство Усы. Иными словами, классы могут наследовать свойства от других классов. Родительский класс называется суперклассом. Внутри классов могут быть объявлены поля и методы.
Для объявления класса служит ключевое слово **class**. Вспомним стандартную строчку кода из Android-проекта:

```
public class MainActivity extends Activity {
    // код внутри класса
}
```

Упрощённая общая форма для класса может иметь следующий вид:

```
class ИмяКласса {
    тип переменная_экземпляра1;

    тип имяМетода(список параметров){
        // тело метода
    }
}
```

В Java принято начинать имена класса с большой буквы. В классе могут быть несколько переменных и методов. Переменные, определённые внутри класса (не метода), называются переменными экземпляра или полями (fields). Код пишется внутри класса. Методы и переменные внутри класса являются членами класса.

Объекты

Новый объект (или экземпляр) создаётся из существующего класса при помощи ключевого слова **new**:

```
Cat barsik = new Cat(); // создали кота из класса Cat
```

В большинстве случаев вы будете использовать такой способ. Пусть вас не удивляет, что приходится дважды использовать слово **Cat**, оно имеет разный смысл.

Слева от оператора присваивания = определяется имя переменной и его тип **Cat**. В правой части выражения происходит выделение памяти для нового экземпляра класса **Cat** и инициализируется(создается) экземпляр. Оператор присваивания присваивает переменной ссылку на только что созданный объект. Имена объектов не нужно начинать с большой буквы, как у класса. Так вы будете различать, где класс, а где экземпляр класса. Если имя экземпляра класса состоит из нескольких слов, то используется верблюжья нотация, когда все первые буквы слов, кроме первой, пишутся с большой - superBlackCat.

Если вы помните, при объявлении примитивных типов мы указывали нужный тип в самом начале.

```
int catAge;
```

Поэтому код **Cat barsik** также определяет его тип. Он не всегда может совпадать с именем класса.

```
Pet barsik = new Cat();
```

В этом примере используется тип класса домашних любимцев **Pet**, а обращаемся к классу котов **Cat**.

Теперь подробнее.

Простой пример создания класса **Box** (коробка для кота):

```
class Box {
   int width; // ширина коробки
   int height; // высота коробки
   int depth; // глубина коробки
}
```

При таком варианте Java автоматически присвоит переменным значения по умолчанию. Например, для **int** это будет значение 0. Но не всегда значения по умолчанию подойдут в вашем классе. Если вы создали переменную для описания количества лап у кота, то логично сразу присвоить значение 4. Поэтому считается хорошей практикой сразу присваивать нужные значения полям класса, не полагаясь на систему.

Вам нужно создать отдельный файл **Box.java**, в который следует вставить код, описанный выше. О том, как создавать новый файл для класса я не буду здесь расписывать.

Сам класс - это просто шаблон, заготовка. Чтобы ваше приложение могло использовать данный шаблон, нужно создать на его основе объект при помощи ключевого слова **new**:

```
    Box catBox = new Box; // создали реальный объект с именем catBox на основе шаблона Box
```

Красивая получилась коробочка.

Объект **catBox**, объявленный в коде вашей программы, сразу займёт часть памяти на устройстве. При этом объект будет содержать собственные копии переменных экземпляра **width, height, depth**. Для доступа к этим переменным используется точка (.). Если мы хотим присвоить значение переменной **width**, то после создания объекта класса можете написать код:

```
catBox.width = 400; // ширина коробки для кота 400 миллиметров
```
Если мы хотим вычислить объём коробки, то нужно перемножить все значения размеров коробки:

```
Box catBox = new Box();

catBox.width = 400;
catBox.height = 200;
catBox.depth = 250;

int volume = catBox.width * catBox.height * catBox.depth;

mInfoTextView.setText("Объём коробки: " + volume);
```
Каждый объект содержит собственные копии переменных экземпляра. Вы можете создать несколько объектов на основе класса **Box** и присваивать разные значения для размеров коробки. При этом изменения переменных экземпляра одного объекта никак не влияют на переменные экземпляра другого объекта. Давайте объявим два объекта класса **Box**:

```
Box bigBox = new Box(); // большая коробка
Box smallBox = new Box(); // маленькая коробка

int volume;

// присвоим значения переменным для большой коробки
bigBox.width = 400;
bigBox.height = 200;
bigBox.depth = 250;

// присвоим значения переменным для маленькой коробки
smallBox.width = 200;
smallBox.height = 100;
smallBox.depth = 150;

// вычисляем объём первой коробки
volume = bigBox.width * bigBox.height * bigBox.depth;
mInfoTextView.setText("Объём большой коробки: " + volume + "\n");

// вычисляем объём маленькой коробки
volume = smallBox.width * smallBox.height * smallBox.depth;
mInfoTextView.append("Объём маленькой коробки: " + volume);
```
Когда мы используем конструкцию типа **Box bigBox = new Box();**, то в одной строке выполняем сразу два действия - объявляем переменную типа класса и резервируем память под объект. Можно разбить конструкцию на отдельные части:

```
Box bigBox; // объявляем ссылку на объект
bigBox = new Box(); // резервируем память для объекта Box
```
Обычно такую конструкцию из двух строк кода не используют на практике, если нет особых причин. Когда мы используем ключевое слово **new** и указываем имя класса, то после имени ставим круглые скобки, которые указывают на конструктор класса. О них поговорим позже.