Android第二次课--Java面向对象

主讲人：徐国林

话说在座的各位有对象了嘛，上次在群里说了这节课最好带个对象来上课才浅显易懂，有没有真的带了对象来的。（题外话）

哈哈哈娱乐一下，现在正式开始吧

1.面向对象的基本概念 Object Oriented Programming（oop）

> 面向过程，像C语言 就是面向过程语言，
>
> 面向对象，像java，c++等 面向对象语言，

> 下面就看看书上这个例子吧。
> 例如，现在有两位师傅“面向过程”和“面向对象”要设计-一个首饰盒:
> (1）“面向过程”师傅:用户提出哪些要求，师傅就针对用户的要求进行制作，直接制作‘
> 出一个完整的整体，本身也不准备好做首饰盒所需要的工具，而是需要什么再单独拿什么。
>
> (2）“面向对象”师傅:针对用户提出的要求进行分析，并将分析的结果设计成一张完整
> 的图纸，与需求的用户确认，然后将一切的准备工作全部处理完之后再分块制作，最后将
> 各个小的部分组装在一起。
>
> 从以上两个师傅的做法可以发现，“面向对象”师傅要比“面向过程”师傅更能适应用户
> 的变化，而一旦用户有变化之后，“面向过程”师傅基本上要推倒重做，而“面向对象”
> 师傅却可以适应变化。

2.类的定义和使用

类的定义：类实际上是表示一个**客观某类群体**的一些基本特征**抽象**，由属性和方法组成。

类的属性：定义类一个个的具体信息，实际上一个属性就是一个变量,也叫成员变量。

类的方法：一些操作的行为。

这里我们通过一个例子和代码来了解类的概念

> 这里我们用人来举例吧，因为人类是一个很大的概念，是一个群体；这里我们就可以把人类抽象成一个类来定义；因为你可以代表自己，不能说自己代表整个人类吧，所以人类就可以声明成一个类。
>
> 然后人类有很多**基本特征**（这些我们每个人都有，你不会说自己没有这些东西吧），像头发，眼睛，鼻子，皮肤颜色等。**这些基本特征就可以称为人类这个类的属性。**
>
> 接着作为一个人，你要吃饭，睡觉，学习吧，这些都是**人的行为，在类中它有称为方法。**

```
package person;

public class Person {
    String name;
    int age;
    String color;

    public void eat()
    {
        System.out.println("该吃饭了，好想吃黄焖鸡。");
    }

    public void sleep()
    {
        System.out.println("我去睡觉了。");
```

```
    }

    public void doHomework()
    {
        System.out.println("遭了高数作业还没做，赶紧的做作业去吧。");
    }
}
```

这里是一个简单的人类的类的定义。

3.this关键字

this：是对自身对象的一个地址引用。

> - 通过this调用另一个构造方法，用法是this(参数列表)，这个仅仅在类的构造方法中，别的地方不能这么用。
> - 函数参数或者函数中的局部变量和成员变量同名的情况下，成员变量被屏蔽，此时要访问成员变量则需要用“this.成员变量名”的方式来引用成员变量。当然，在没有同名的情况下，可以直接用成员变量的名字，而不用this，用了也不为错。
> - 在函数中，需要引用该函所属类的当前对象时候，直接用this。

看看demo代码

super： 是类似引用 但不是引用的调用。

- 在子类构造方法中要调用父类的构造方法，用“super(参数列表)”的方式调用，参数不是必须的。同时还要注意的一点是：“super(参数列表)”这条语句只能用在子类构造方法体中的第一行。
- 当子类方法中的局部变量或者子类的成员变量与父类成员变量同名时，也就是子类局部变量覆盖父类成员变量时，用“super.成员变量名”来引用父类成员变量。当然，如果父类的成员变量没有被覆盖，也可以用“super.成员变量名”来引用父类成员变量，不过这是不必要的。
- 当子类的成员方法覆盖了父类的成员方法时，也就是子类和父类有完全相同的方法定义（但方法体可以不同），此时，用“super.方法名(参数列表)”的方式访问父类的方法。

4.对象

hhh，讲完了类，现在开始讲对象吧。

讲了这个，以后大家都是有对象的人了吧。

定义：就是表示一个个具体的东西。类是一个抽象的东西，而对象就是表示类的一个实体。

所以我们刚才定义了一个person类，现在就可以声明几个对象（实体）了。例如张涛学长，天齐学长等。

```
        Person tq=new Person();

//思考思考对象存放在栈内存中还是堆内存中，tq为对象名称，存放在栈内存中，而对象的属性存放在堆内存中。
```

对象的用途：访问类中的某个属性和方法。

```
        tq.setName("天齐学长");
        tq.setAge(18);
        tq.setColor("yellow");
        tq.print();
        System.out.println(tq.getName()+tq.getAge()+tq.getColor());
```

5.匿名对象

就是没有给出明确名字的对象，一般匿名对象只使用一次，而且匿名对象只在堆内存中开辟空间，不纯在栈内存的引用。

举个栗子

```
new Person("张涛学长",18,"yellow").print();
```

6.访问修饰符

> （1）public：可以被所有其他类所访问，不管是否在同一个包。
>
> （2）private：只能被自己访问和修改。
>
> （3）protected：自身，不同包的子类及同一个包中类可以访问。

（4）default（默认）：同一包中的其他类可以访问，声明时没有加修饰符。

- 常用的就是public和private。

	类内部	本包	子类	外部包
public	√	√	√	√
protected	√	√	√	×
default	√	√	×	×
private	√	×	×	×

7.方法的声明及使用

定义：方法就是一段可重复调用的代码段。在C语言中好像是被称为函数，不过在java中我们称为方法。

为什么要定义方法呢?

其实在一个项目中，会有很多代码会被重复利用，如果我们每次重新写一遍就会增加代码量，很浪费空间，更会降低你代码的质量，显的很混乱，所以我们可以讲这些代码定义为一个方法进行反复利用。

方法的命名及一些类的命名和包的命名：在定义类时，全部单词的首字母必须大写，那么在定义方法时也有命名规范要求。即第一个单词的首字母小写，之后每个单词的首字母大写，如printInfo()方法。希望你们在日后的开发中养成好的习惯。

方法的结构：

```
public static 返回值类型 方法名称（类型参数1,类型参数2, ...) {

    //程序语句;
    //[ return表达式];

}
//如果返回类型是void这里我们就不用return 数据回去
```

下面是一个方法的具体实例

```
public void doHomework() {
        System.out.println("遭了高数作业还没做，赶紧的做作业去吧。");
    }
```

方法的重载：就是方法名称相同，但**参数的类型和参数的个数**不同。通过传递参数的个数及类型的不同可以完成不同功能的方法调用。

```
     public static int add(int x,int y)
    {
        return x+y;
    }

    public static int add(int x,int y,int z)
    {
        return x+y+z;
    }

    public static float add（float x,float y)
    {
        return x+y;
    }
```

重载注意事项

```
public static int add(int x,int y)
    {
        return x+y;
    }

//    public static float add(int x,int y)
//    {
//        return x+y;
//    }
```

方法的重写：子类与父类方法的方法名，参数，返回值都要相同

> （1）父类与子类之间的多态性，对父类的函数进行重新定义。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。在java中，子类可继承父类中的方法，而不需要重新编写相同的方法。但有时子类并不想原封不动地继承父类的方法，而是想作一定的修改，这就需要采用方法的重写。方法重写又称方法覆盖;
>
> （2）若子类中的方法与父类中的某一方法具有相同的方法名、返回类型和参数表，则新方法将覆盖原有的方法。如需父类中原有的方法，可使用super关键字，该关键字引用了当前类的父类;
>
> （3）子类函数的访问修饰权限不能少于父类的.

结束一个方法：如果返回类型是void则调用方法就自动结束了，如果是int，或者是float，string，用return返回对应数据结束方法。

8.构造方法

从前面所讲解的代码可以发现，实例化一个类的对象后，如果要为这个对象中的属性赋值，则必须用setter方法，那么有没有一种简单的方法，可以在对象实例化时就直接把对象属性赋值呢?

```
        Person ts=new Person("张涛学长",18,"yellow");
        Person tq=new Person("天齐学长",18,"yellow");


//person类里的构造方法

    public Person(String name, int age, String color) {
        this.name = name;
        this.age = age;
        this.color = color;
    }
```

注意：

> 构造方法的名称必须与类名称一致。
> 构造方法的声明处不能有任何返回值类型的声明。
> 不能在构造方法中使用return返回一个值。

其实你会发现前面的类我没有写构造方法，也可以调用，这是为什么?

那是因为类本身就存在构造方法，需要说明的是类必须存在构造方法，在java中如果一个类中没有声明一个构造方法，编译时会直接生成一个无参数的，不发挥作用的构造方法。

```
//形式如下
class person {
    public Person ()
    }
```

当一个类中有多个构造方法时，编译时不再自动声明默认的构造方法，这里的构造方法也只会实现第一个。

```
public Person (String name)
{
    this.setName(name);
}

pulic Person (String name,int age){
    this.setName(name);
    this.setAge(age);
}
```

9.面向对象的三大特性：封装，继承，多态

相信大家已经对对象和类有了一定的了解

接下来我们将要接触三个概念，这个东西将会在你后面的使用中经常出现。

封装：是指利用抽象数据类型将数据和基于数据的操作封装在一起，使其构成一个不可分割的独立实体，对外不可见。也可以使用上面的访问修饰符，限制访问权限.（数据隐藏）

```
    private String name;
    private int age;
    private String color;

//这种情况下，外部访问成员变量目前只能够通过getter来得到数据，外部改变目前也只能通过setter来访问。
```

简单的说就是，你设计一个类，类本身 对提供需求的人 只管我给 封装的类 需求，然后 人 拿到结果，人 不用去管 封装的类内部 是怎么去操作得到结果的。

封装的好处：

> - 良好的封装能够减少耦合。（简单理解，一个类和其他地方减少联系）
> - 类内部的结构可以自由修改。（因为内部怎么实现的不给外面知道，所以内部结构可以修改）
>   修改我们实现的代码而又不会破坏其他人使用我们的代码
> - 写好封装，应该先清楚什么变量，方法应该暴露，什么是不能暴露，或在只能通过一些接口（比如
>   Getter()， Setter()）暴露出去。
>
> 那么如果设置了访问权限，这样真就不能访问到私有变量（被private修饰的变量）了吗？（hhh 后面讲到反射，你们就知道了）

继承：拥有反映事物一般特性的类，然后在其基础上派生出反映特殊事物的类。

关键字： extends 表示正在构造的新类派生于一个已存在的类。被继承的类称为超类，基类或父类；
继承产生的类称为子类，派生类或孩子类。

> 举个例子：
>
> 学生，老师等等是人，所以学生类，可以继承 人 类，得到 人 类的一些属性，像年龄，肤色，姓名；
>
> 然后也会继承到人的一些方法，例如睡觉，吃饭。
>
> 而这里以学生举例，学生自己要做作业，所以学生类自己也有方法 做作业。
>
> 现在你们知道继承的好处了嘛，避免重复写很多代码，减少了很多代码量。

```
public class Student extends Person {

    //子类调用父类的构造方法
    public Student(String name, int age, String color) {
        super(name, age, color);
    }
    public void doHomework() {
        System.out.println("我要做作业了");
    }
}

//这里你会发现学生也会拥有人的属性和基本方法，这就是继承的好处
```

注意: 单继承，Java继承只能继承一个父类。

多态：java语言中含有对象有着多态和方法重载；对象可以在特定的情况下，表现不同的状态，从而对应着不同的属性和方法；我们可以在不同的调用下，选择不同的方法（但方法名是一样的，就是前面讲方法的重载）。

> 对象向上转型：

```
    Person stu = new Stdudent ()

//在这里我们这样理解，这里定义了一个Person 类型的stu，它指向Student对象实例。由于Student是继承与Person，所以Student可以自动向上转型为Person，所以stu是可以指向Student实例对象的。这样做的好处是，在继承中我们知道子类是父类的扩展，它可以提供比父类更加强大的功能，如果我们定义了一个指向子类的父类引用类型，那么它除了能够引用父类的共性外，还可以使用子类强大的功能。但是向上转型存在一些缺点，那就是它必定会导致一些方法和属性的丢失，而导致我们不能够获取它们。所以父类类型的引用可以调用父类中定义的所有属性和方法，对于只存在与子类中的方法和属性就不能调用了。
```

> 对象向下转型：对象先向上转型，在向下转型

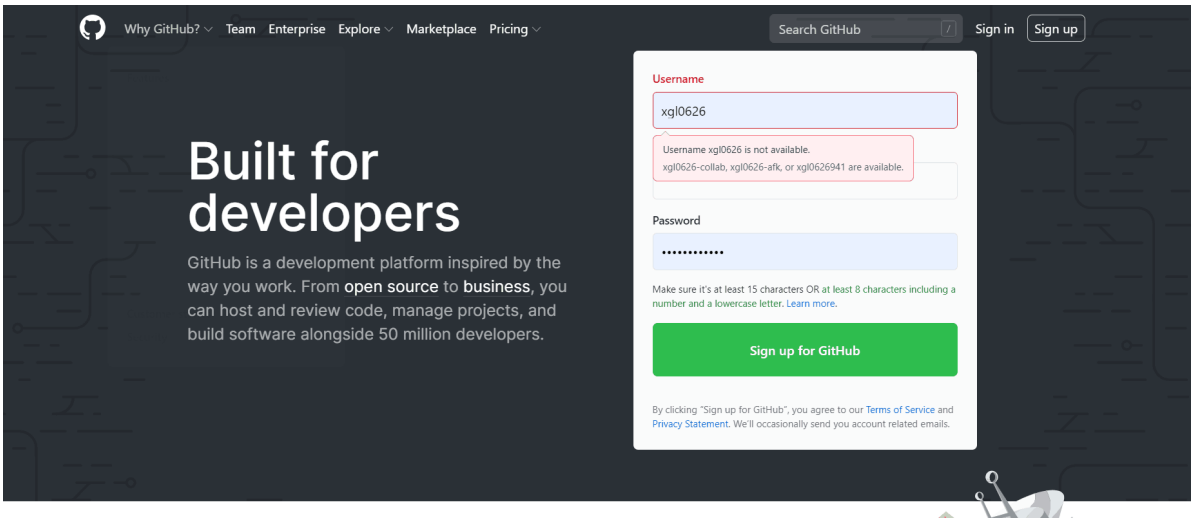
```
Person per = new Student();//向上转型

Student stu = (Student) per;//向下转型

//调用父类及子类所有方法，弥补向上转型的缺点。
```

这里两个转型如果理解不了，就先不管吧，后面慢慢用多了就知道了

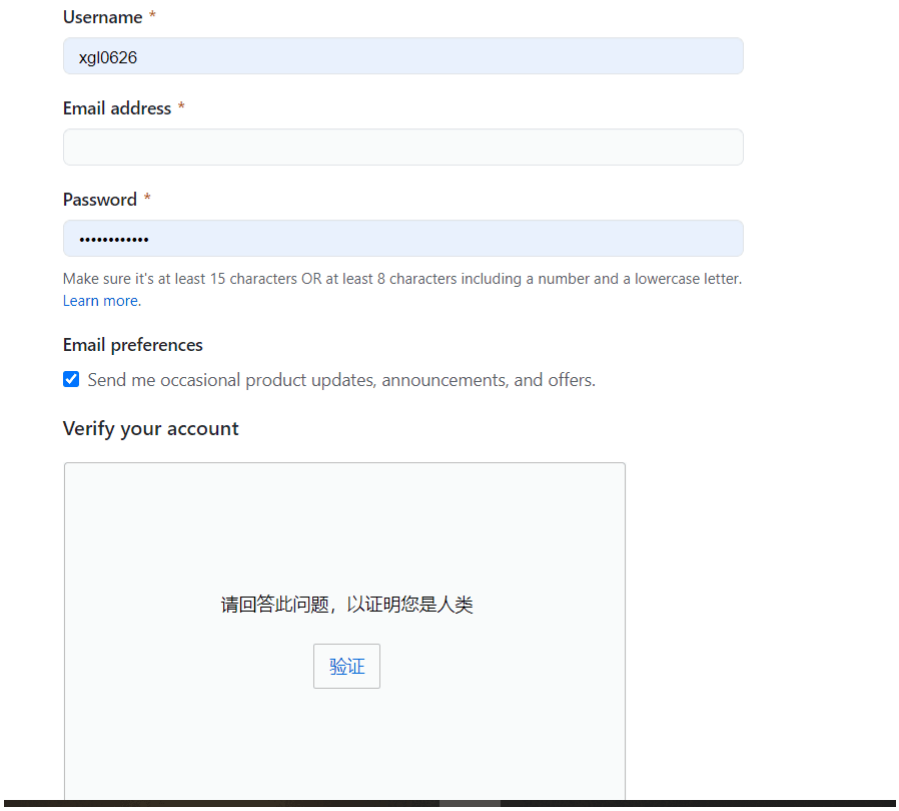
10. github与git

[github网站]

点击 sign up 进入注册



然后填好自己的个人信息就可以注册了



建议安装google浏览器。

git

这里是你们自行安装git，我相信你们都没有问题，如果有问题的话，就自己多百度和bing，google都可以，下面也有我给出的教程，也可以看看。

git使用教程:

> 廖雪峰git教程

安装git的教程

> git官网下载地址这里也有git的一些查阅资料
>
> git安装教程
>
> git上次本地代码到github远程仓库教程

git常用命令行

> git init　// 在你的项目文件里生成一个本地仓库
>
> git add . //添加你写的代码文件
>
> git commit -m "注释语句"　（引号里写关于你代码的注释语句） //提交你写的代码文件
>
> git push　//推送你的代码文件到远程仓库，这一步基于你已经关联远程仓库才可以推送代码，否则会失败
>
> git remote add origin 填写你的远程仓库地址　//本地仓库关联远程仓库
>
> git remote -v　//查看当前关联的远程仓库

> git push --set-upstream origin master //创建一个远程master分支并提交代码到远程仓库master分支上
>
> 这些是你们这次提交代码会用的一些 git 命令语句， 到时候课上我也会给你们演示一遍。

最后提一句

> 这节课的内容或许比前面一节课的内容要难上许多，如果没有在课上没有听懂也不要怕，课下多花点时间看看java实战经典开发或者其他书籍还是可以搞懂的。加油！！！