继 承（上）

毕向东

4 继承（上）

4.1 继承的概述

4.2 继承的特点

4.3 super关键字

4.4 函数覆盖

4.5 子类的实例化过程

4.6 final关键字

4.1 继承的概述

- 多个类中存在相同属性和行为时，将这些内容抽取到单独一个类中，那么多个类无需再定义这些属性和行为，只要继承单独的那个类即可。
- 多个类可以称为子类，单独这个类称为父类或者超类。
- 子类可以直接访问父类中的非私有的属性和行为。
- 通过 extends 关键字让类与类之间产生继承关系。
  - `class SubDemo extends Demo{}`
- 继承的出现提高了代码的复用性。
- 继承的出现让类与类之间产生了关系，提供了多态的前提。

4.2 继承的特点

- Java只支持单继承，不支持多继承。
  - 一个类只能有一个父类，不可以有多个父类。
  - `class SubDemo extends Demo{} //ok`
  - `class SubDemo extends Demo1,Demo2...//error`
- Java支持多层继承(继承体系)
  - `class A{}`
  - `class B extends A{}`
  - `class C extends B{}`
- 定义继承需要注意：
  - 不要仅为了获取其他类中某个功能而去继承
  - 类与类之间要有所属("is a")关系，xx1是xx2的一种。

4.3 super关键字

- super和this的用法相同
- this代表本类应用
- super代表父类引用
- 当子父类出现同名成员时，可以用super进行区分
- 子类要调用父类构造函数时，可以使用super语句。

4.4 函数覆盖(Override)

- 子类中出现与父类一模一样的方法时，会出现覆盖操作，也称为重写或者复写。
- 父类中的私有方法不可以被覆盖。
- 在子类覆盖方法中，继续使用被覆盖的方法可以通过super.函数名获取。
- 覆盖注意事项：
  - 覆盖时，子类方法权限一定要大于等于父类方法权限
  - 静态只能覆盖静态。
- 覆盖的应用：
  - 当子类需要父类的功能，而功能主体子类有自己特有内容时，可以复写父类中的方法，这样，即沿袭了父类的功能，又定义了子类特有的内容。

4.5 子类的实例化过程

- 子类中所有的构造函数默认都会访问父类中空参数的构造函数
- 因为每一个构造函数的第一行都有一条默认的语句super();
- 子类会具备父类中的数据，所以要先明确父类是如何对这些数据初始化的。
- 当父类中没有空参数的构造函数时，子类的构造函数必须通过this或者super语句指定要访问的构造函数。

4.6 final关键字

- final可以修饰类，方法，变量。
- final修饰的类不可以被继承。
- final修饰的方法不可以被覆盖。
- final修饰的变量是一个常量。只能被赋值一次。
- 内部类只能访问被final修饰的局部变量。