在面向对象语言中，接口的多种不同的实现方式即为多态。

允许将子类类型的[指针](http://baike.baidu.com/item/%E6%8C%87%E9%92%88)赋值给父类类型的指针

把不同的子类对象都当作父类来看，可以屏蔽不同子类对象之间的差异

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | public interface Parent//父类接口  {      public void simpleCall();  }  public class Child\_A implements Parent  {      public void simpleCall();      {      //具体的实现细节；      }  }    public class Child\_B implements Parent  {      public void simpleCall();      {      //具体的实现细节；      }  } |

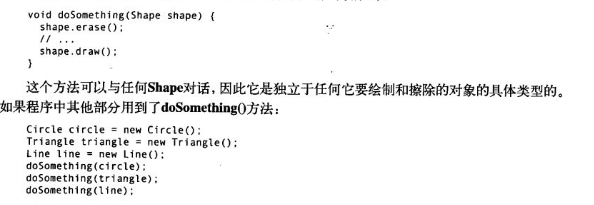
Parent pa = new Child\_A();

pa.simpleCall()则显然是调用Child\_A的方法；

Parent pa = new Child\_B();

pa.simpleCall()则是在调用Child\_B的方法。

子类向上转型



这里的有一个**后期绑定（绑定是方法调用和方法实体相关联）**的概念，编译器确保被绑定的方法存在，对参数的调用和返回进行检查（不同参数是不同的方法，就不算覆盖），但是并不知道确切执行的方法。Java使用一小段特殊代码替换绝对地址调用（不同于前期绑定）

Static和final是前期绑定



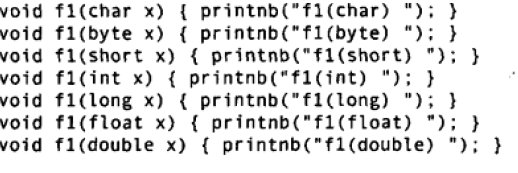
重载

重载（overload）：对于类的方法（包括从**父类中继承的方法**），**方法名相同，参数列表不同的方法之间就构成了重载关系**。这里有两个问题需要注意：

（1） 参数列表指：参数的类型，参数的个数，参数的顺序。这三者只要有一个不同就叫做参数列表不同。

（2）重载可发生在同一个类中，也可发生在子类与父类中

基本类型问题：



F1(5)，5被当作int处理，如果没有int（方法中的参数(long float doble)大于传入参数则会被提升）

Int>long>float>double怎么一个提升顺序

如果传入参数为char，但没有接受char的方法则char提升为int

覆盖

1、“**三同一不低**” 子类和父类的**方法名称**，**参数列表**，**返回类型**必须**完全相同**，而且子类方法的访问修饰符的权限不能比父类**低**。

2、子类方法不能抛出比父类方法**更多的异常**。即子类方法所抛出的异常必须和父类方法所抛出的**异常一致**，或者**是其子类**，或者**什么也不抛出**；

3、被覆盖的方法不能是**final**类型的。因为final修饰的方法是无法覆盖的。（如果子类存在与父类final方法同名的方法，不会覆盖只会新建一个同名的）

4、被覆盖的方法不能为**private**。否则在其子类中只是新定义了一个方法，并没有对其进行覆盖。

5、被覆盖的方法不能为**static**。所以如果父类中的方法为静态的，而子类中的方法不是静态的，但是两个方法除了这一点外其他都满足覆盖条件，那么会发生编译错误。反之亦然。即使父类和子类中的方法都是静态的，并且满足覆盖条件，但是仍然不会发生覆盖，因为静态方法是在**编译**的时候把**静态方法和类的引用类型**进行匹配。

方法覆盖要求参数列表（参数签名）必须一致，而方法重载要求参数列表必须不一致。

方法覆盖要求返回类型必须一致，方法重载对此没有要求。

方法覆盖只能用于子类覆盖父类的方法，方法重载用于同一个类中的所有方法（包括从父类中继承而来的方法）

方法覆盖对方法的访问权限和抛出的异常有特殊的要求，而方法重载在这方面没有任何限制。

父类的一个方法只能被子类覆盖一次，而一个方法可以在所有的类中可以被重载多次。

**无法以返回型作为重载函数的区分标准**

方法重载是指在一个类中定义多个同名的方法（和继承没有直接关系）