**一，接口**

1. 接口可以理解为一个**纯粹的抽象类**

最原始的接口——只含有**抽象方法**和**全局常量（主要原则）**

JDK1.8之后，接口中也可以定义**普通方法**和**静态方法（挽救原则）**

1. 接口中所有抽象方法的权限都是public （写与不写都是public）

void fun( ); public void fun( ); abstract void fun( ); 这三个都是public abstract

接口中的属性都是public static final（写于不写都是public static final）

1. 接口多继承：一个接口可以继承多个接口
2. 接口的使用一般有三种形式
   1. 进行标准设置
   2. 表示一种操作的能力
   3. 暴露远程方法视图，这个一般用在RPC分布式开发中

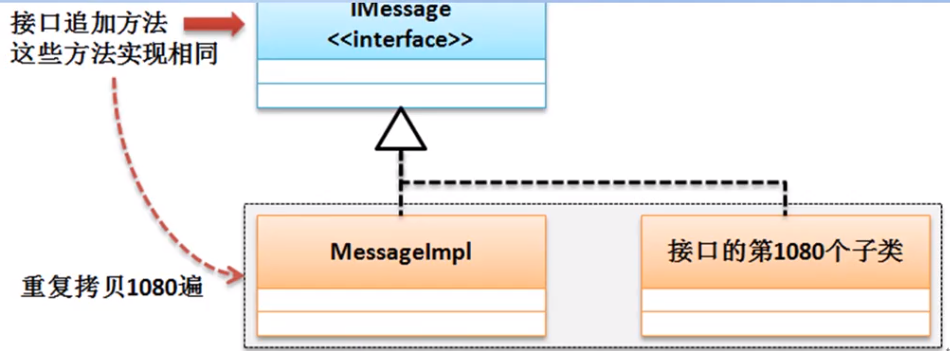
**二，接口加强定义（JDK1.8后可以定义普通方法，静态方法）**

**JDK1.8之前，接口中不可以定义普通方法，只能定义抽象方法和常量**

这就导致了一个问题

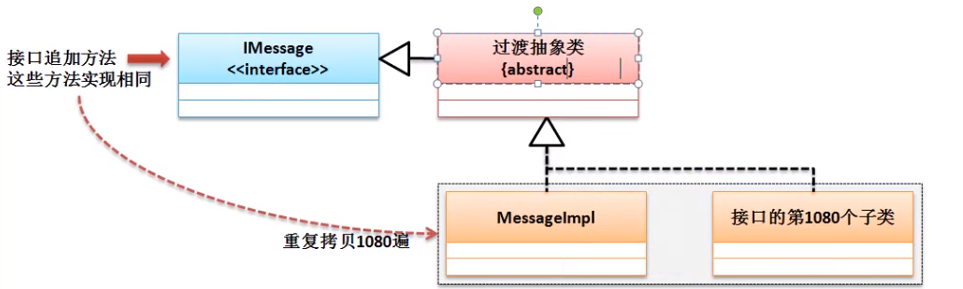
如果接口的结构设计不当，后期需要在接口中添加新的方法，这些方法实现都相同

这时候，如果上千个子类直接implements这个接口，那么我们在上千个子类中要覆写这个方法上万遍



**为了避免这种情况发生，我们不让子类直接implemnets接口，而是在子类和接口之间增加一个抽象类**

**这样我们直接在抽象类中追加这个方法，就可以解决这个问题**



**但是JDK1.8之后，为了解决接口设计的缺陷，所以在接口中我们可以定义普通方法和static方法**

**接口中的普通方法必须追加default声明，但是需要注意的是，这个功能属于挽救功能，不应该作为设计首选**

**public default 返回类型 方法名{ 方法体 }**

**综上所述，解决接口设计不当，在接口中添加新方法的问题 ，有两个措施**

1. **在子类和接口中添加一个过渡抽象类（主要考虑这个）**
2. **利用接口的新特性，在接口中定义普通方法（JDK1.8之后）**

**三，抽象类和接口的区别**

**抽象类 接口**

**组成 构造方法，普通方法，静态方法 抽象方法，全局常量**

**全局变量，普通变量 普通方法，static方法**

**权限 可以用各种权限 只有public**

**子类使用 子类只可以继承一个抽象类 子类可以有多个接口**

**两者关系 抽象类可以实现若干个接口 接口不可以继承类，**

**但可以继承多个接口**

**使用(同) 1，两个都必须有子类**

1. **两个的子类都必须覆写抽象方法**
2. **都通过子类向上转型实现对象实例化**

**在两个都可以用的时候，优先考虑接口**