**接口支持申明带实现的方法，使用default关键字。**

Java 8中加入默认方法主要是为了支持库设计师，让他们能够写出更容易改进的接口。这一方法很重要，因为你会在接口中遇到越来越多的默认方法，但由于真正需要编写默认方法的程序员相对较少，而且它们只是有助于程序改进，而不是用于编写任何具体的程序，举个例子，

List<Apple> heavyApples3 = inventory.stream()

.filter((Apple a) -> a.getWeight() > 150)  
 .collect(Collectors.*toList*());

List<Apple> heavyApples4 = inventory.parallelStream()  
 .filter((Apple a) -> a.getWeight() > 150)  
 .collect(Collectors.*toList*());

但这里有个问题：在Java 8之前，List<T>并没有stream或parallelStream方法，它实现的Collection<T>接口也没有，因为当初还没想到这些方法！可没有这些方法，这些代码就不能编译。换做你自己的接口的话，最简单的解决方案就是让Java 8的设计者把stream方法加入Collection接口，并加入ArrayList类的实现。

可要是这样做，对用户来说就是噩梦。有很多的替代集合框架都用Collection API实现了接口。但给接口加入一个新方法，意味着所有的实体类都必须为其提供一个实现，这下你就进退两难了：你如何改变已发布的接口而不破坏已有的实现呢？

Java 8的解决方法就是打破最后一环——接口如今可包含实现类没有提供实现的方法签名！那谁来实现它呢？缺失的方法主体随接口提供了（因此就有了默认实现），而不是由实现类提供。

这就给接口设计者提供了一个扩充接口的方式，而不会破坏现有的代码。Java 8在接口声明中使用新的default关键字来表示这一点。例如，在Java 8里，你现在可以直接对List调用sort方法。它是用Java 8List接口中如下所示的默认方法实现的，它会调用Collections.sort静态方法：

@SuppressWarnings({"unchecked", "rawtypes"})

default void sort(Comparator<? super E> c) {  
 Object[] a = this.toArray();  
 Arrays.*sort*(a, (Comparator) c);  
 ListIterator<E> i = this.listIterator();  
 for (Object e : a) {  
 i.next();  
 i.set((E) e);  
 }  
}

这意味着List的任何实现类都不需要显式实现sort，而在以前的Java版本中，除非提供了sort的实现，否则这些实体类在重新编译时都会失败。

不过慢着，一个类可以实现多个接口，不是吗？如果在好几个接口里有多个多个默认实现，在某种程度上意味着Java中有了某种形式的多重继承。