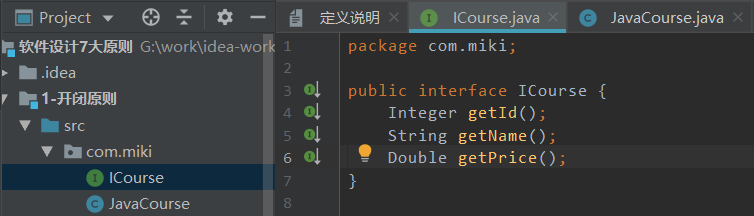
## 一：开闭原则

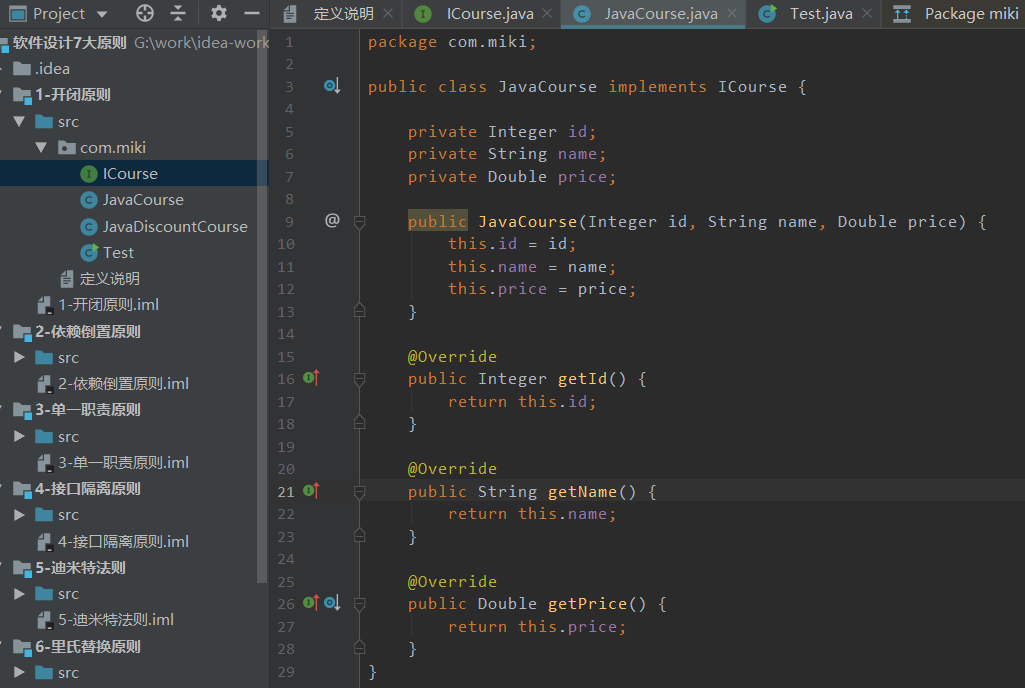
定义: 一个软件实体如类，模块和函数应该对扩展开放，对修改关闭  
 #核心思想：用抽象构建框架，用实现扩展细节  
 #优点：提高软件系统的可复用性及可维护性  
  
 举例：  
 对于很多互联网公司，工作时间弹性制，每天工作8小时，对于8小时时间是关闭的，不可修改的，但是对于何时来是开放的，早点来上班8个小时，可以早点下班，  
 晚点来上班8个小时，就要晚点走。总之要满足工作8小时。

### 需求示例1：

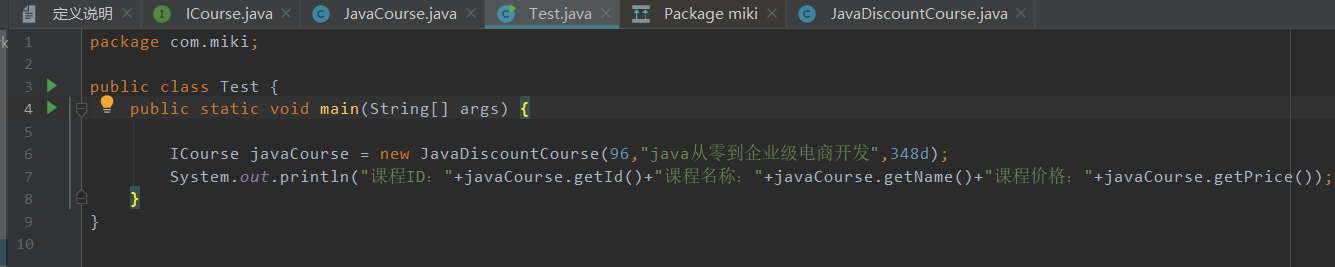
1. 以java培训课程为例，创建课程接口



1. 然后创建java课程类，



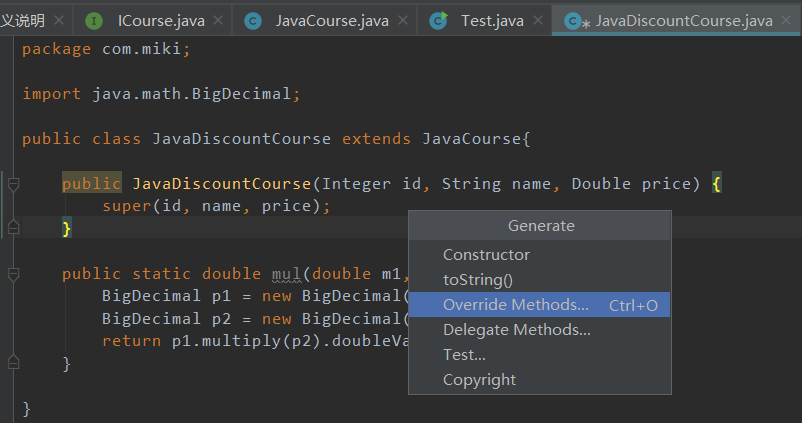
1. 测试结果：

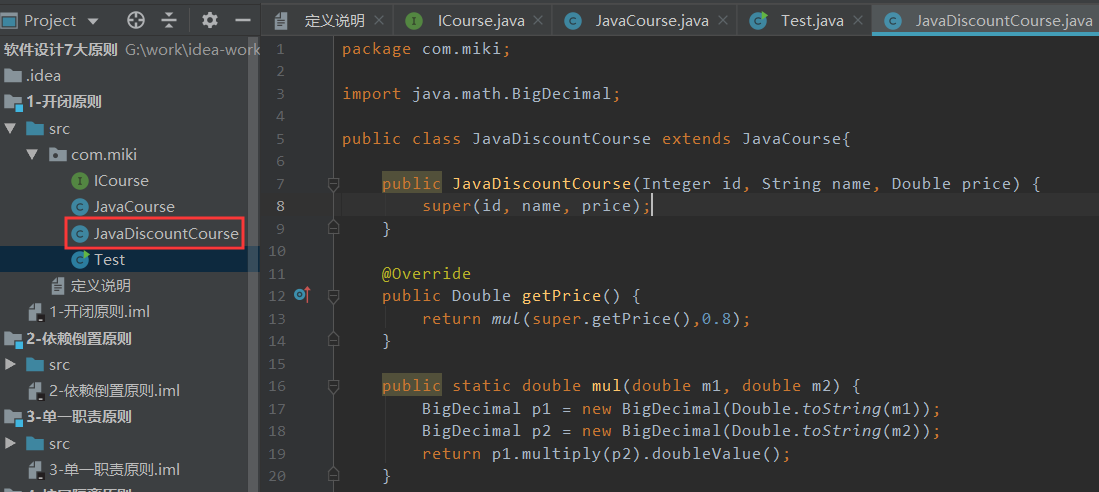


### 需求示例2：

获取打折后的课程价钱，利用开闭原则实现如下：

1. 创建java课程打折类，继承java课程类，重写父类的获取价格的方法：Alt + Insert





1. 运行结果如下：



### 需求示例3：

获取课程原价：



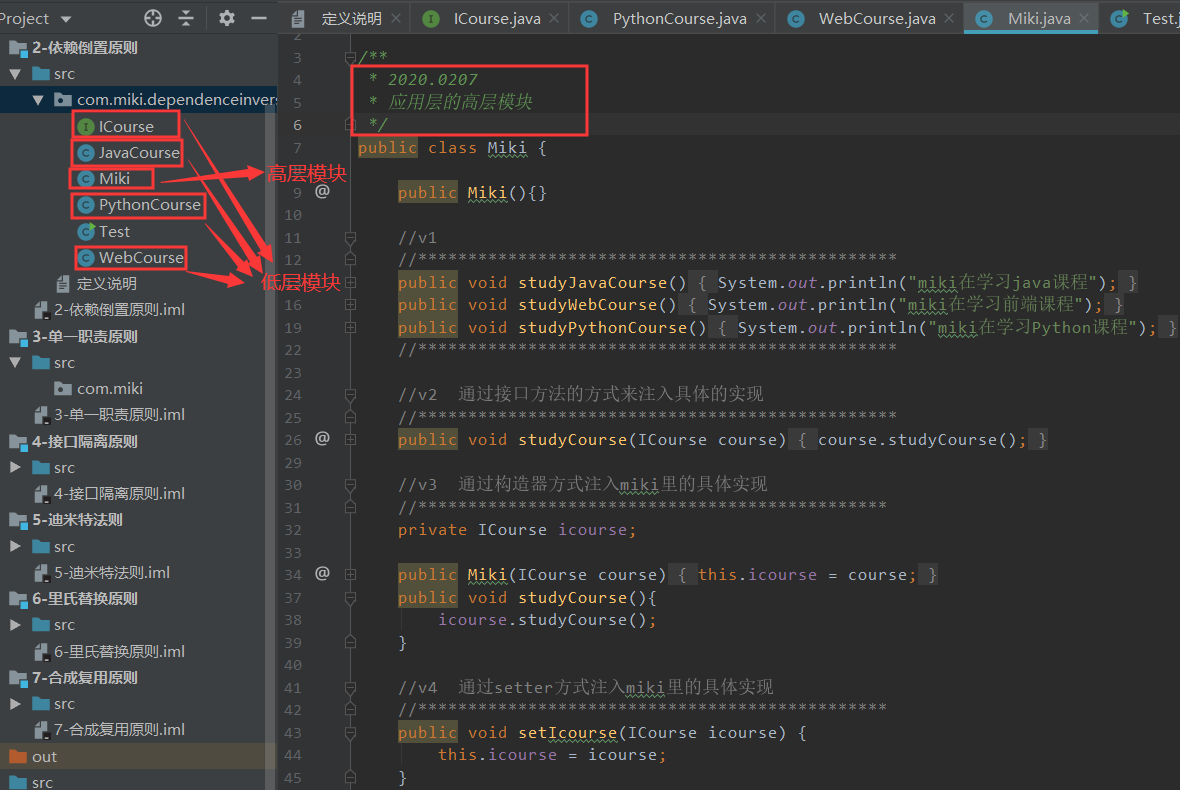
## 二：依赖倒置原则

定义: 高层模块不应该依赖低层模块，二者都应该依赖其抽象  
 #核心思想：抽象不应该依赖细节；细节应该依赖抽象；针对接口编程，不应该针对实现编程  
 #优点：可以减少类间的耦合性，提高系统的稳定性，提高代码的可读性和可维护性，可降低修改程序所造成的风险

### 需求示例1：

高层模块不依赖低层模块的三种方式：

1. 通过接口方法来注入高层模块具体实现类
2. 通过构造器方式来注入高层模块具体实现类（spring的service注解实现原理）
3. 通过setter方式驻日高层模块miki具体实现类中

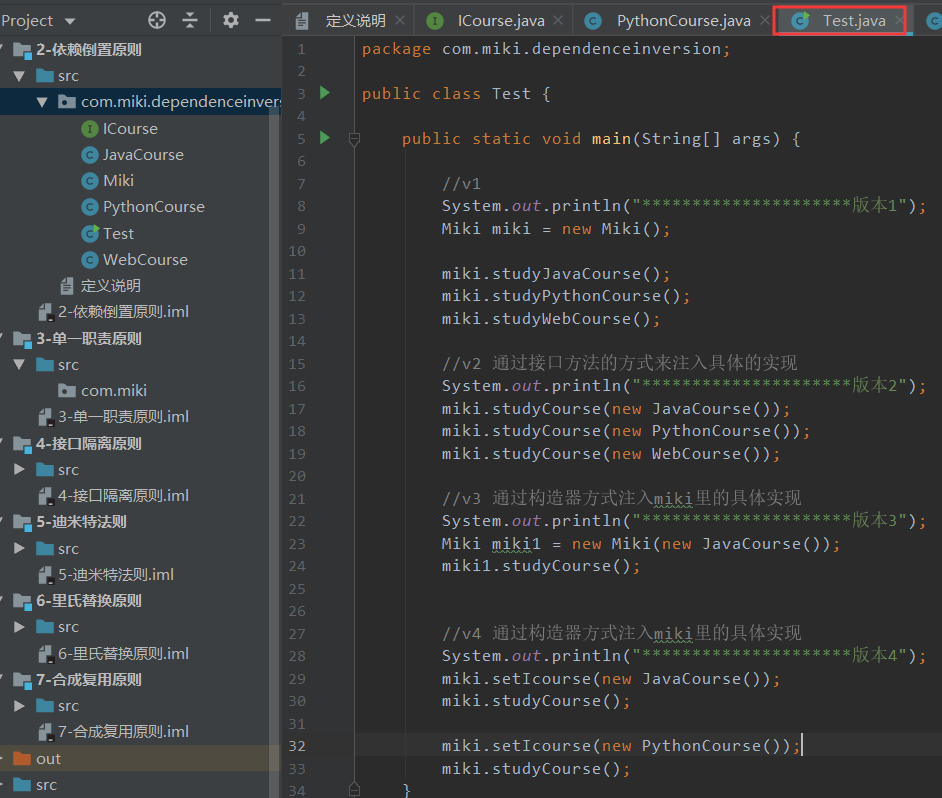


应用层代码调用高层模块代码的三种调用方式：

1.通过接口方法来注入高层模块具体实现类

2.通过构造器方式来注入高层模块具体实现类（spring的service注解实现原理）

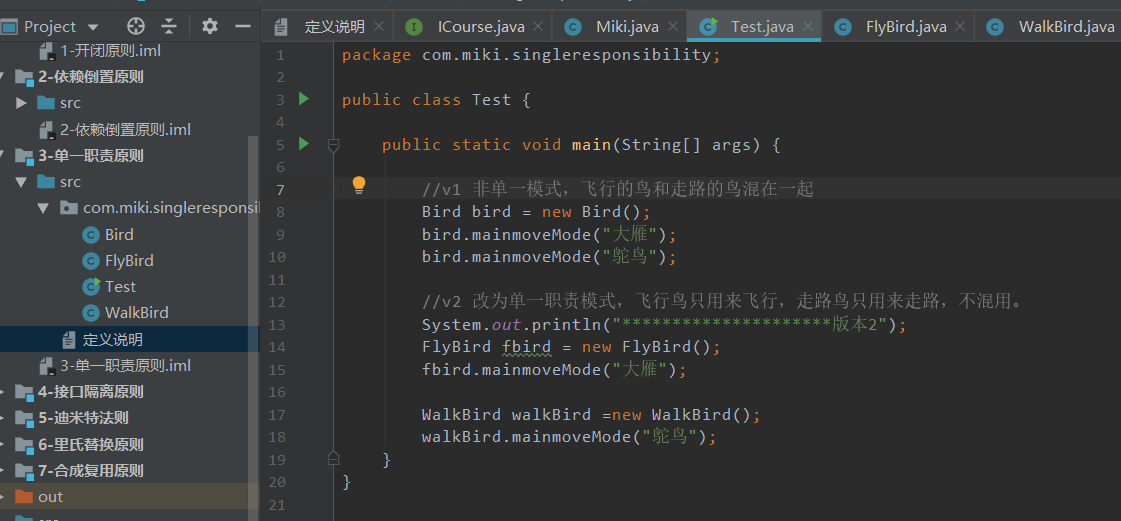
3.通过setter方式驻日高层模块miki具体实现类中



## 三：单一职责原则

定义: 不要存在多于一个导致类变更的原因  
 #核心思想：一个类/接口/方法 只负责一项职责  
 #优点：降低类的复杂度，提高类的可读性，提高系统的可维护性，降低变更引起的风险  
  
 举例：各行各业中，每个人都负责自己专门的的工作，不会插手干预别人的工作，这样是单一职责的最好体现。

### 需求示例：



## 四：接口隔离原则

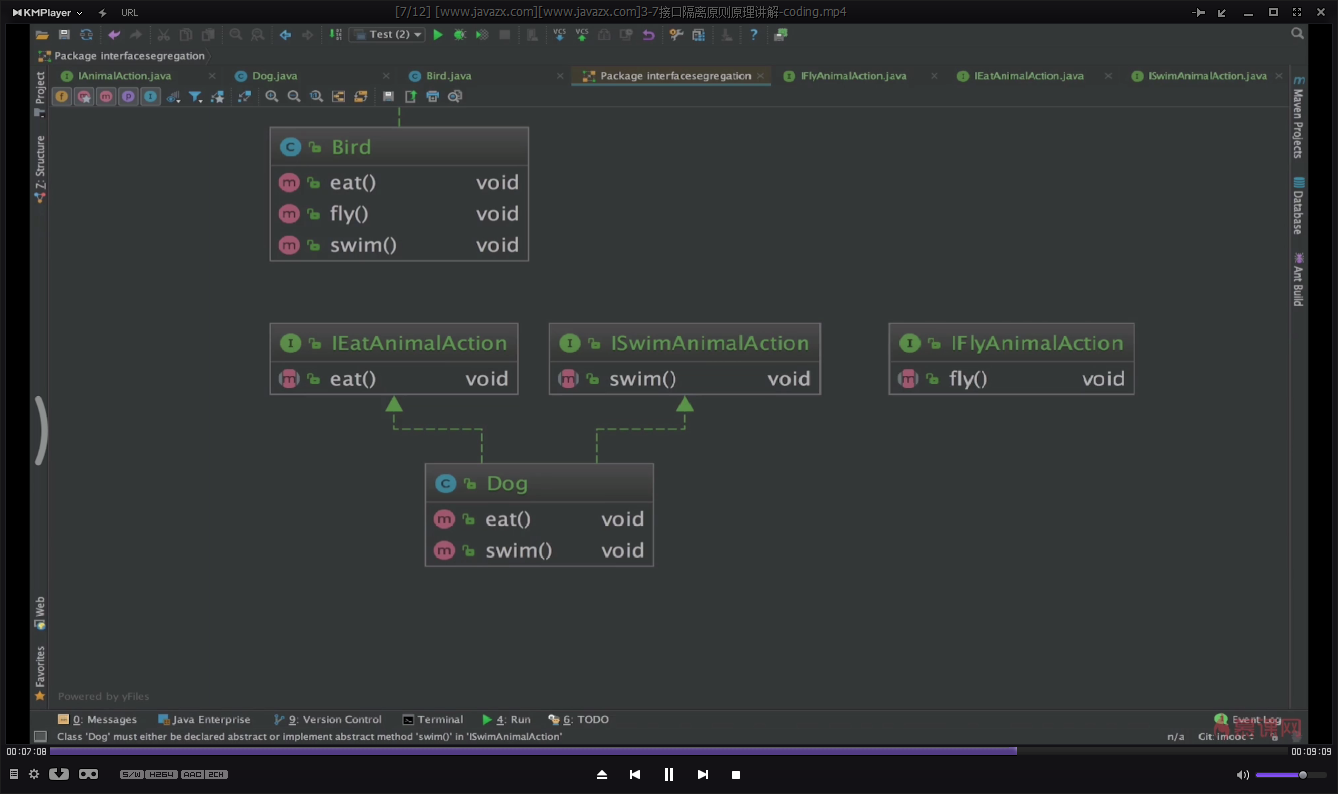
定义: 用多个专门的接口，而不使用单一的总接口，客户端不应该依赖它不需要的接口  
 #核心思想：1.一个类对一个类的依赖应建立在最小的接口上  
 2.建立单一接口，不要建立庞大臃肿的接口  
 3.尽量细化接口，接口中的方法尽量少  
 #注意事项：一定要适度，否则会建立过多庞大数量的接口  
  
 #优点：符合我们平时所说的高内聚，低耦合的设计思想，从而使得类具有很好的可读性，可扩展性，和可维护性。  
  
 举例：

# 和单一职责原则的区别：单一职责是针对接口负责某一个功能（一个功能可以衍生很多个细化功能，如消费功能：包含交房租，购物，水电费等）接口隔离原则则是将消费功能隔离出来，不用细化各种消费功能。

### 需求示例：

最初版的代码是一个Bird接口类包含eat方法，fly方法，swim方法

按照接口隔离原则，将bird接口类的三个方法拆分为三个行为接口类



## 五：迪米特(最少知道原则)原则

定义: 一个对象应该对其他对象保持最少的了解，又叫最少知道原则。  
  
 #核心思想：1.尽量降低类与类之间的耦合  
 2.强调只和朋友交流，不和陌生人说话  
 3.对外部引入的类越少越好  
 #朋友：出现在成员变量，方法的输入，输出参数中的类称为朋友类，而出现在方法体内部的类不属于朋友类。  
 #注意事项：每个类都尽量减少对其他类的依赖  
  
 #优点：降低类与类之间的耦合。  
  
 举例：

### 需求示例：

