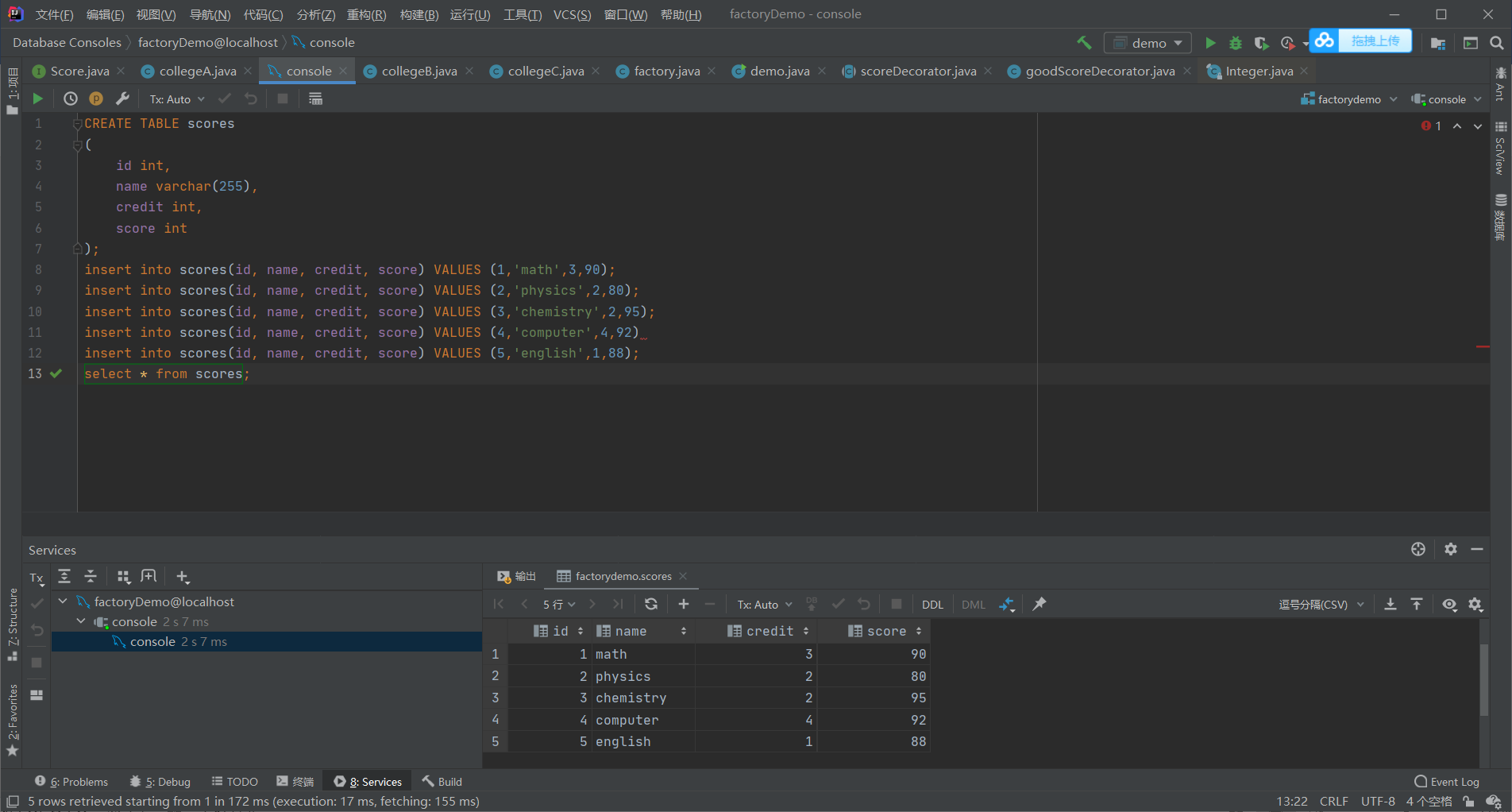
# Java运用工厂模式以及装饰器模式实现不同学院规则下同一成绩单的综测成绩计算：

工厂模式

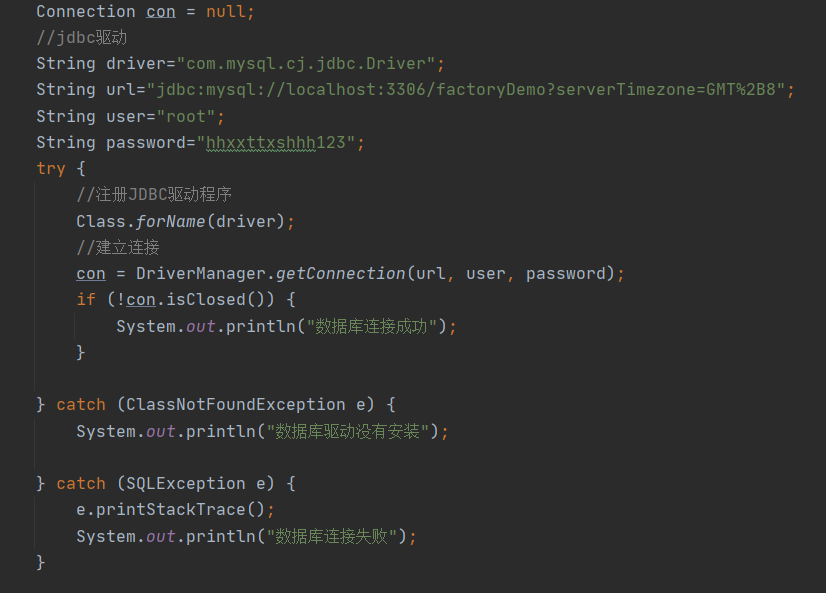
工厂模式（Factory Pattern）是 Java 中最常用的设计模式之一。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。

在工厂模式中，我们在创建对象时不会对客户端暴露创建逻辑，并且是通过使用一个共同的接口来指向新创建的对象。

在正式开始工厂模式设计之前。首先连接数据库，这里采用mysql数据库，并且创建表格，此处只生成一个学生的成绩单，因此只需要创建一个表，并且插入数据



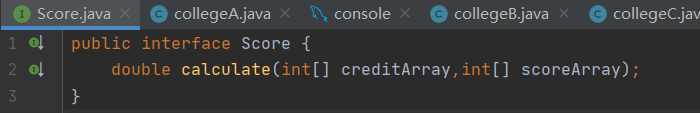
创建好成绩单之后，在主函数中利用jdbc的方式连接到数据库



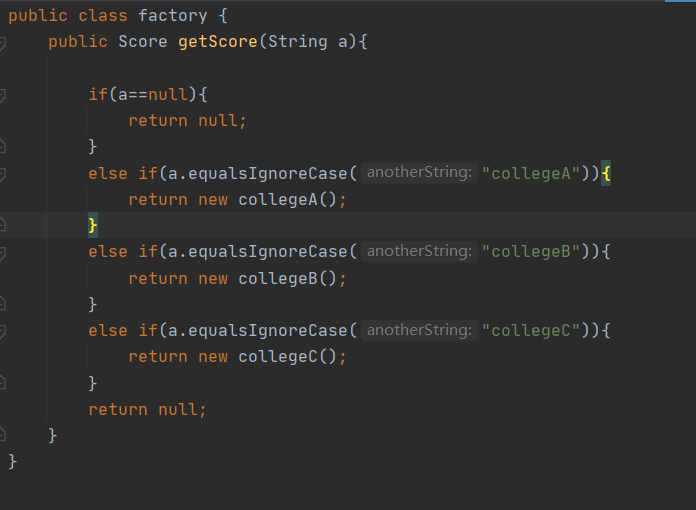
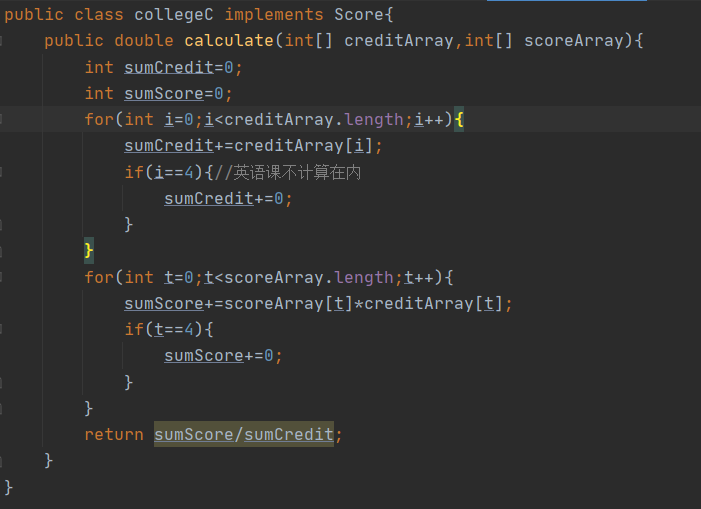
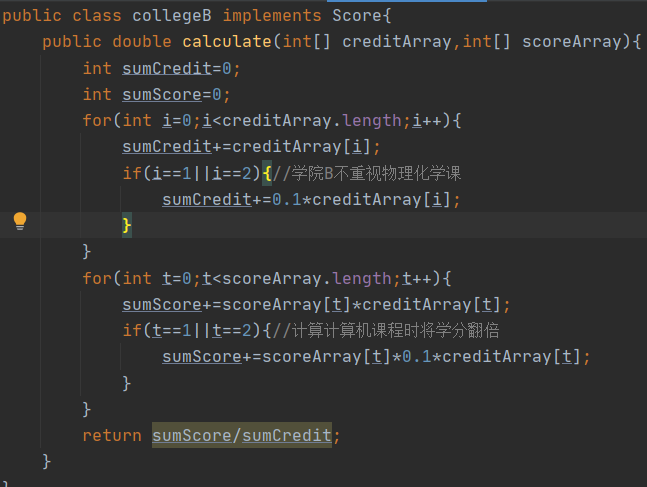
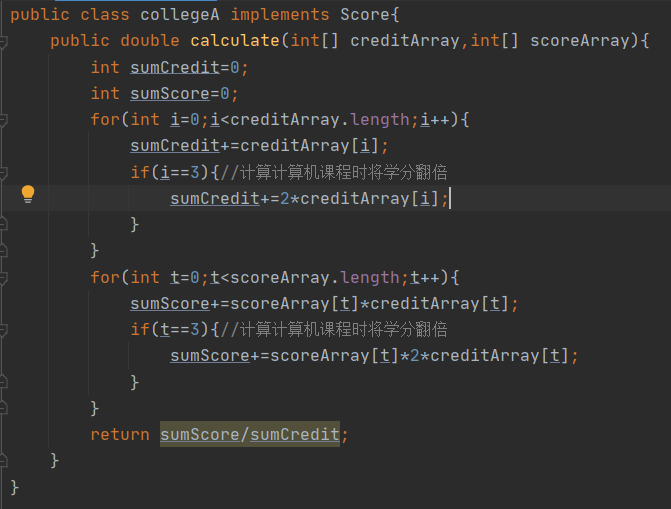
并且通过select语句和ResultSet类中的方法将数组中的学分和成绩两列数据提取到两个数组中：creditArrray和scoreArray。打印数据库中的成绩单使更加直观。



然后开始简单工厂模式的设计，创建接口Score，接口中定义方法calculate，参数为两个整形数组（分别存放数据库中的学分数组和成绩数组）。



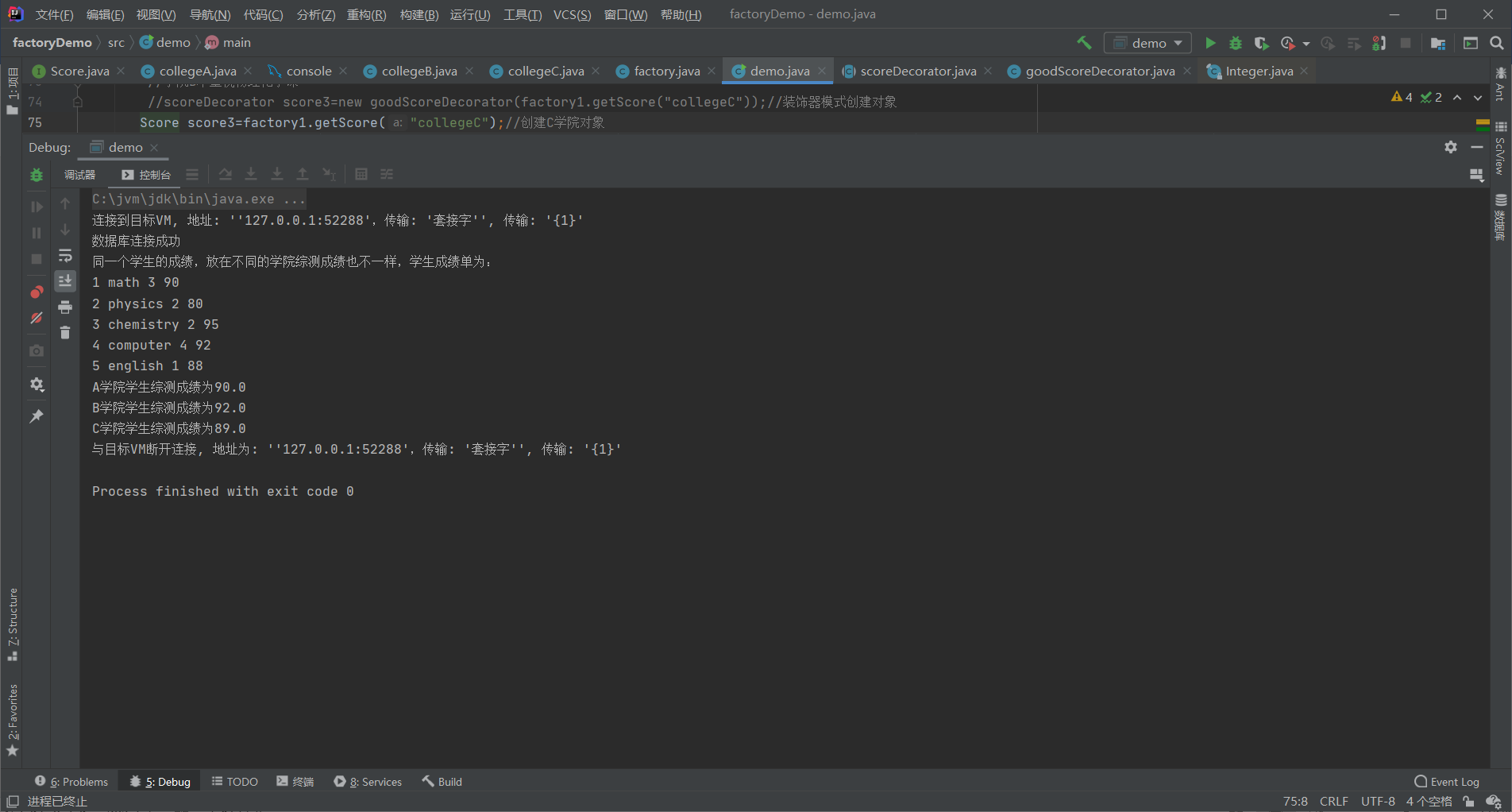
然后创建三个类：collegeA-C分别代表三个学院，三个学院计算标准各不一样。A学院看重计算机，B学院不太重视物理化学， C学院不算英语。并且创建一个类factory通过三个学院（类）的名字创建相应的对象。



最后在主函数中分别创建对象，调用简单工厂模式并且计算各学院标准下的成绩



结果如图：

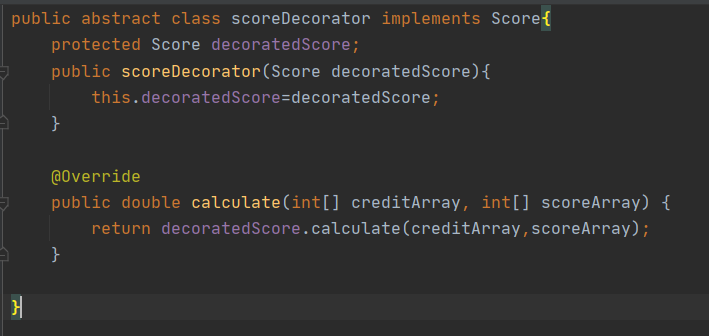


装饰器模式

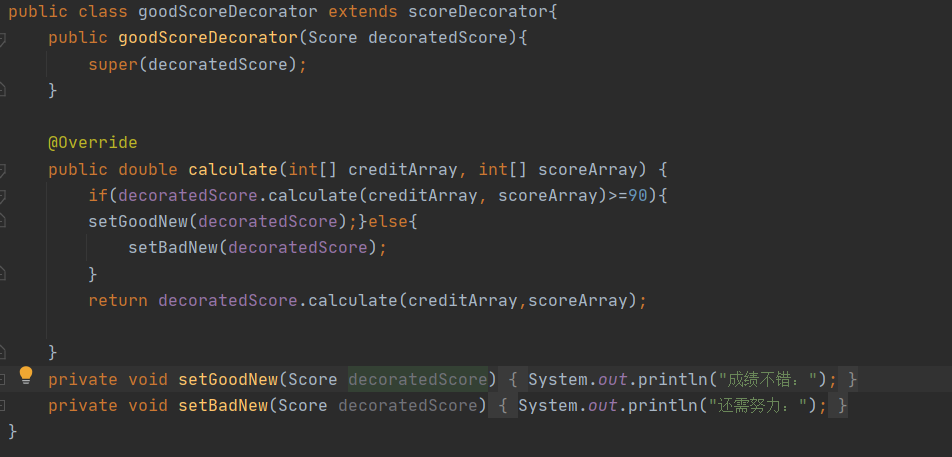
装饰器模式（Decorator Pattern）允许向一个现有的对象添加新的功能，同时又不改变其结构。这种类型的设计模式属于结构型模式，它是作为现有的类的一个包装。

这种模式创建了一个装饰类，用来包装原有的类，并在保持类方法签名完整性的前提下，提供了额外的功能。

我们的项目中，首先创建一个抽象装饰类scoreDecorator实现Score接口，并且使用Score对象作为实例变量



接着创建扩展了scoreDecorator的实体装饰类goodScoreDecorator用于实现额外的功能，这里涉及的功能是当计算出综测成绩大于90时打印成绩不错，当不到90时打印还需努力。



最后在主函数中利用goodScoreDecorator来装饰Score对象，用装饰器模式创建对象，实现额外的功能



结果：

