**每日作业卷**

**Java基础第二阶段**

**第16天**

**基础加强\_注解\_类加载器\_动态代理**

传智播客.黑马程序员

# 关卡1

## 训练案例1

### 训练描述：【讲义1.1：JUnit介绍】

1. 请使用一句话说明JUnit的作用

答：

### 操作步骤描述

（参考讲义1.1）

## 训练案例2

### 训练描述：【讲义1.2：JUnit的使用】

1. 请写出使用第三方工具（jar包）的步骤：

答：

### 操作步骤描述

（参考讲义1.2）

## 训练案例3

### 训练描述：【讲义1.2：JUnit的使用\_@Test】

1. 请定义类ArrayTools，包含一个方法getMax，此方法可以接收一个int[]数组，并返回这个数组中的最大值。
2. 在ArrayTools中定义方法test()，并使用JUnit测试getMax方法；

### 操作步骤描述

1. 创建项目：day16作业\_Test1\_3
2. 参考讲义1.2，按上述要求编写代码；

## 训练案例4

### 训练描述：【讲义1.2：JUnit的使用\_@Before和@After】

1. 请定义类ArrayTools，并定义以下方法：
2. 方法printArray，此方法可以接收一个int[]数组，并打印这个数组中的所有元素（多个元素之间用逗号隔开）。
3. 方法：check，在方法内打印：”之前执行......”
4. 方法：log，在方法内打印：”之后执行......”
5. 方法test()，并使用JUnit测试printArray方法。要求在执行前先执行check()方法，在执行后执行log()方法。

### 操作步骤描述

1. 创建项目：day16作业\_Test1\_4
2. 参考讲义1.2，按上述要求编写代码；

## 训练案例5

### 训练描述：【讲义2.1：注解概述】

1. 请回答以下问题：
2. “注解”与“注释”的区别：

答：

1. 注解的作用：

答：

### 操作步骤描述

（参考讲义2.1）

## 训练案例6

### 训练描述：【讲义2.2：JDK提供的注解】

1. 请先写出以下注解的作用。并创建新项目，写出可以应用以下注解的代码：
2. @Deprecated

**作用：**

1. @Override

**作用：**

1. @SuppressWarnings(“deprecation”)

**作用：**

1. @SuppressWarnings(“rawtypes”)

**作用：**

1. @SuppressWarnings(“unused”)

**作用：**

1. @SuppressWarnings(“null”)

**作用：**

1. @SuppressWarnings(“serial”)

**作用：**

### 操作步骤描述

1. 先参考讲义2.2，了解并写出每种注解的作用；
2. 创建项目day16作业\_Test1\_6
3. 对应于7道题，编写Test1—Test7七个类，分别实现使用上述7个注解的代码应用场景。

## 训练案例7

### 训练描述：【讲义2.3-2.6：注解的定义和使用】

1. 创建新项目，按以下要求定义，并使用注解：
2. 请定义一个最简单的注解@MyAnno1(参考讲义2.3)：
3. 不需要任何属性。
4. 此注解只能修饰“类”和接口(参考讲义2.6)。
5. 此注解要出现在源码和字节码中(参考讲义2.6)
6. 定义测试类：Test1，并使用此注解修饰(参考讲义2.4)
7. 请定义注解@MyAnno2：
8. 包含一个String类型的属性“type”，并且定义默认值“java”。
9. 此注解只能修饰“字段”。
10. 此注解只需要能够在源码中使用。
11. 定义测试类：Test2，随意定义一个成员属性，并使用此注解；
12. 请定义注解@MyAnno3：
13. 包含一个String类型的属性“type”，不定义默认值。
14. 包含一个int[]数组类型的属性“intArr”，不定义默认值。
15. 此注解只能修饰“方法”。
16. 此注解要出现在源码和字节码中。
17. 定义测试类：Test3，随意定义一个成员方法，并使用此注解；

### 操作步骤描述

1. 创建项目day16作业\_Test1\_7
2. 参考讲义2.3—2.6，按上述要求实现代码。

## 训练案例8

### 训练描述：【讲义2.7—自定义@Test注解】

1. 自定义注解@MyTest，模拟JUnit的@Test注解，可以执行一个无参，无返回值的方法。

### 操作步骤描述

1. 创建项目day16作业\_Test1\_8
2. 定义自定义注解@MyTest，不需要属性。但要求这个注解只能在方法上使用，并且定义注解的生命周期为”运行时”。
3. 编写类Student，随意定义一个方法，使用此注解；
4. 编写类ParseAnno，使用反射加载类及判断@MyTest注解，执行具有@MyTest注解的所有方法。

## 训练案例9

### 训练描述：【讲义4.1：动态代理】

1. 按以下要求编写代码：
2. 定义一个接口：Person，包含以下抽象方法：work()
3. 定义一个类Student，实现Person接口，实现work()方法，打印输出：”我做Java项目”；
4. 定义一个MyHandler，实现InvocationHandler接口，有如下要求：
5. 定义成员属性--被代理对象：
6. 定义构造方法，为被代理对象赋值；
7. 定义一个方法before()，打印输出：”项目设计”；
8. 定义一个方法after()，打印输出：”项目总结”；
9. 重写invoke()方法，要求在调用方法前执行before()方法，在调用方法后执行after()方法。
10. 定义一个测试类：Test，包含main()方法。要求用动态代理获取Student类的代理对象，并执行coding()方法。

### 操作步骤描述

1. 创建项目：day16作业\_Test1\_9
2. 参考讲义4.1，按要求编写代码

# 关卡2

## 训练案例1

### 训练描述：【JUnit、注解综合】

1. 自定义三个注解：@MyTest，@MyBefore，@MyAfter，分别模拟JUnit中的@Test，@Before，@After三个注解的作用，并测试使用。

### 操作步骤描述

1. 创建项目：day16作业\_Test2\_1；
2. 分别定义三个注解：@MyTest， @MyBefore，@MyAfter，无需任何属性，要求三个注解均只能用在方法上，而且作用域范围在运行时；
3. 定义一个Student类，随意定义三个方法，分别使用三个注解。
4. 定义一个注解解析类ParseMyAnno，并定义方法parse()，方法中：反射解析注解，先运行@MyBefore的方法，然后运行@MyTest的方法，最后运行@MyAfter的方法。
5. 使用JUnit的@Test注解运行parse()方法。

# 关卡3

## 训练案例1

### 训练描述：【文件、自定义注解综合】

1. 需求说明：我们来做一个“图书信息管理系统”，有以下要求：
2. 图书要求记录以下信息：

图书编号，作者，出版社，出版日期，单价，数量

1. 图书信息记录到文件：Test3\_1.txt中，每个图书信息存储一行，各属性值之间用逗号隔开。
2. 程序要求提供以下功能(不需要列菜单)：

添加图书，修改图书，删除图书，查询所有图书

1. 各功能具体要求：
2. 添加图书：要求验证图书编号不能重复。将新图书信息追加写入到文件：Test3\_1.txt中；
3. 修改图书：根据图书编号查找图书，允许修改图书的除图书编号外的所有信息。修改完成后，将修改后的数据再次写入到Test3\_1.txt中。
4. 删除图书：根据图书编号查找图书，找到后先提示是否删除，如果用户选择”是”，执行删除。同时文件中的数据也要被删除；
5. 查询所有图书：根据图书编号查询商品；
6. 代码要求：
7. 定义类BookDAO，此类中包含了上述所有的方法。
8. 为测试每个功能模块，不从main()方法启动，使用我们自定义的@MyTest注解分别测试每个方法。

### 操作步骤描述

## 训练案例2

### 训练描述：【文件、动态代理综合】

1. 需求说明：接上例，本例中，我们为BookDAO添加动态代理。
2. 要求为BookDAO中的每个方法进行增强，要求在执行每个方法前要进行”权限检查”，在执行每个方法后要进行”写日志”操作。（这两个操作只需要打印文字即可）

### 操作步骤描述

1. 创建新项目：day16作业\_Test3\_2
2. 将关卡3的案例1中的所有类和文件复制到本项目中；
3. 为BookDAO增加一个接口IBook，包含所有方法的声明；
4. 定义Test类，包含main()方法，获取BookDAO的动态代理对象，并调用每个方法测试。