## Q3

## JUnit介绍

JUnit是一个Java语言的单元测试框架，正如其名字所包含的那样。

JUnit提供了注释来识别和标定测试方法，在一个java文件中标定多个代码块，运行此程序时，可以同时运行所有被标定的代码块（函数），并统计返回测试结果。

在JetBrains IDEA 开发环境中，IDE还支持用侧边栏的不同颜色来直观地显示测试的运行结果。

## JUnit使用方法

对于Maven项目，只需要在项目的配置文件中(pom.xml)加入

<dependency>  
   <groupId>junit</groupId>  
   <artifactId>junit</artifactId>  
   <version>版本号</version>  
</dependency>

import com.geasse.dao.CharactersMapper;  
import com.geasse.pojo.Characters;  
import com.geasse.utils.MyBatisUtils;  
import org.apache.ibatis.session.SqlSession;  
import org.apache.log4j.Logger;  
import org.junit.Test;  
​  
public class LogFactoryTest {  
   static Logger logger = Logger.getLogger(LogFactoryTest.class);  
​  
   @Test  
   public void LogFactoryTest1(){  
       SqlSession sqlSession = MyBatisUtils.getSqlSession();  
       CharactersMapper mapper = sqlSession.getMapper(CharactersMapper.class);  
​  
       Characters zack = mapper.getCharactersByName("Zack");  
       System.out.println(zack);  
​  
       sqlSession.close();  
  }  
​  
   @Test  
   public void Log4jTest(){  
       logger.info(" info : entered Log4jTest()");  
       logger.debug("debug: entered Log4jTest()");  
       logger.error("error: entered Log4jTest()");  
       logger.info("info: 这是一段中文");  
  }  
}  
​

对于软件工程而言，以JUnit为代表的测试方法的演进大大扭转了程序设计工业的流程。因为JUnit使得单元测试的代码编写的速度大大提升，成本随之降低。当下流行的敏捷开发过程模型中，往往会先编写测试程序，在编写主程序的过程中，目的被转换为通过测试。这是因为JUnit套件提供了断言等多种机制，可以很好地规范待测程序的输出结果。

对于软件测试而言，单元测试是测试工作实行的第一步。只有通过了单元测试，后续的集成测试和系统测试才有进行下去的可能。JUnit为Java项目的单元测试提供了方便简洁的方法。

## JUnit单元测试的意义

以上述代码为例，其中设置了两个测试子代码段，测试了Mybatis运行过程中的日志模块，日志工厂和Log4j是互斥的两个组件，此处分别简单测试了它们的功能。

Junit使用范例程序已经包含在作业中，此处截取部分代码和测试结果用于展示说明：

## Junit使用案例

导入Junit后，可以在测试程序中通过@Test注释标注待测代码段，一般来说，一个待测代码段对应着一个组件的测试。

一般而言，我们习惯于构建一个和项目实体目录相同的测试目录，在对应的组件目录中新建测试程序，添加测试代码。

对于一般的Java项目，则需要在网络上下载Junit的jar包，手动在项目的库中导入jar包。

就可以通过maven导入JUnit的jar包。

## Q8

1）语句覆盖

| **sale** | **day** | **money** | **路径** |
| --- | --- | --- | --- |
| 300 | 9 | 50% | abd |
| 300 | 9 | 70% | abc |
| 100 | 10 | 80% | eg |
| 100 | 10 | 90% | adh |

2）判断覆盖

| **Sale** | **Day** | **Money** | **路径** | **覆盖判断** |
| --- | --- | --- | --- | --- |
| 300 | 9 | 50% | abc | abc |
| 300 | 9 | 70% | abd | abd |
| 300 | 12 | 50% | afg | afg |
| 300 | 12 | 70% | afh | afh |
| 100 | 10 | 40% | eg | eg |
| 100 | 10 | 72% | eh | eh |

3）条件覆盖

A:销售额大于200万

B:销售额小于200万

C:请假天数<10

D:请假天数>=10

E:现金到账>=60%

F:现金到账<=60%

G:现金到账<=85%

H:现金到账>85%

| **Sale** | **Day** | **Money** | **路径** | **覆盖条件** |
| --- | --- | --- | --- | --- |
| 300 | 9 | 50% | abc | ACF |
| 300 | 9 | 70% | abd | ACE |
| 300 | 12 | 50% | afg | ADF |
| 300 | 12 | 70% | afh | ADE |
| 100 | 10 | 40% | eg | BG |
| 100 | 10 | 72% | eh | BH |

4）判断—条件覆盖

| **Sale** | **Day** | **Money** | **路径** | **覆盖条件** | **覆盖判断** |
| --- | --- | --- | --- | --- | --- |
| 300 | 9 | 50% | abc | ACF | abc |
| 300 | 9 | 70% | abd | ACE | abd |
| 300 | 12 | 50% | afg | ADF | afg |
| 300 | 12 | 70% | afh | ADE | afh |
| 100 | 10 | 40% | eg | BG | eg |
| 100 | 10 | 72% | eh | BH | eh |

5）条件组合覆盖

本题中条件组合覆盖等效于条件覆盖

| **Sale** | **Day** | **Money** | **路径** | **覆盖条件** | **覆盖判断** |
| --- | --- | --- | --- | --- | --- |
| 300 | 9 | 50% | abc | ACF | abc |
| 300 | 9 | 70% | abd | ACE | abd |
| 300 | 12 | 50% | afg | ADF | afg |
| 300 | 12 | 70% | afh | ADE | afh |
| 100 | 10 | 40% | eg | BG | eg |
| 100 | 10 | 72% | eh | BH | eh |

## Q9

