## 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。

## Q4

传统的单元测试包括了驱动模块（driver） 和桩模块（stub）。驱动模块的目的很单纯，就是为了访问类库的属性和方法，来检测类库的功能是否正确；

驱动模块

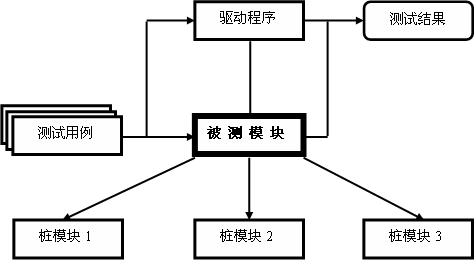
驱动模块是用来模拟被测试模块的上一级模块，相当于被测模块的主程序。它接收数据，将相关数据传送给被测模块，启用被测模块，并打印出相应的结果。

驱动模块（Driver）可以通过模拟一系列用户操作行为，比如选择用户界面上的某一个选项或者按下某个按钮等，自动调用被测试模块中的函数。驱动模块（Driver）设置，使对模块的测试不必与用户界面真正交互。

桩模块

桩模块是指模拟被测试的模块所调用的模块，而不是软件产品的组成的部分。主模块作为驱动模块,与之直接相连的模块用桩模块代替。在集成测试前要为被测模块编制一些模拟其下级模块功能的“替身”模块，以代替被测模块的接口，接受或传递被测模块的数据，这些专供测试用的“假”模块称为被测模块的桩模块。

如果被测试的单元模块需要调用其他模块中的功能或者函数（method），我们就应该设计一个和被调用模块名称相同的桩模块来模拟被调用模块。这个桩模块本身不执行任何功能仅在被调用时返回静态值来模拟被调用模块的行为。举例说明：如果被测试单元中需要调用另一个模块customer的函数 getCustomerAddress(customerID: Integer)，这个函数应该查询数据库后返回某一个客户的地址。我们设计的同名桩模块（Stub）中的同名函数并没有真正对数据库进行查询而仅模拟了这个行为，直接返回了一个静态的地址例如"123 Newton Street"。桩模块的设置使得单元测试的进行成为一个相对独立且简单的过程。



总结：

桩模块的使命除了使得程序能够编译通过之外，还需要模拟返回被代替的模块的各种可能返回值(什么时候返回什么值需要根据测试用例的情况来决定)。

驱动模块的使命就是根据测试用例的设计去调用被测试模块，并且判断被测试模块的返回值是否与测试用例的预期结果相符

## Q5

开源测试工具：

1. NoSQL的单元测试工具 NoSQLUnit

单元测试工具

授权协议:Apache

开发语言:Java

操作系统:跨平台

NoSQLUnit 详细介绍：

NoSQLUnit 是一个 JUnit 的扩展，用来为那些使用了 NoSQL 后端的应用提供单元测试和集成测试的工具。

使用示例：

@UsingDataSet(locations="my\_data\_set.json", loadStrategy=LoadStrategyEnum.REFRESH)

@ShouldMatchDataSet(location="my\_expected\_data\_set.json")

2.C++单元测试工具 ECUT

Eclipse插件 测试工具 单元测试工具

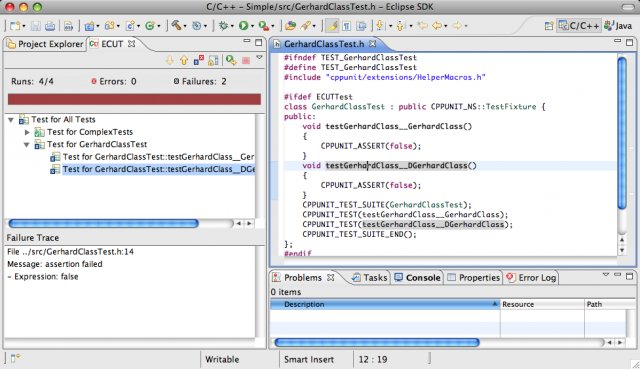
授权协议:EPL

开发语言:C/C++

操作系统:跨平台

ECUT 详细介绍：

ECUT (Eclipse C++ Unit Testing) 项目将 CppUnit 和 Eclipse CDT 开发工具集成起来，提供更好的无缝的单元测试支持，类似 JUnit 和 JDT 一样。该项目也允许跟其他的 C++单元测试框架进行集成。



3.C单元测试工具 greatest

[单元测试工具](https://www.oschina.net/project/tag/222/unittest)

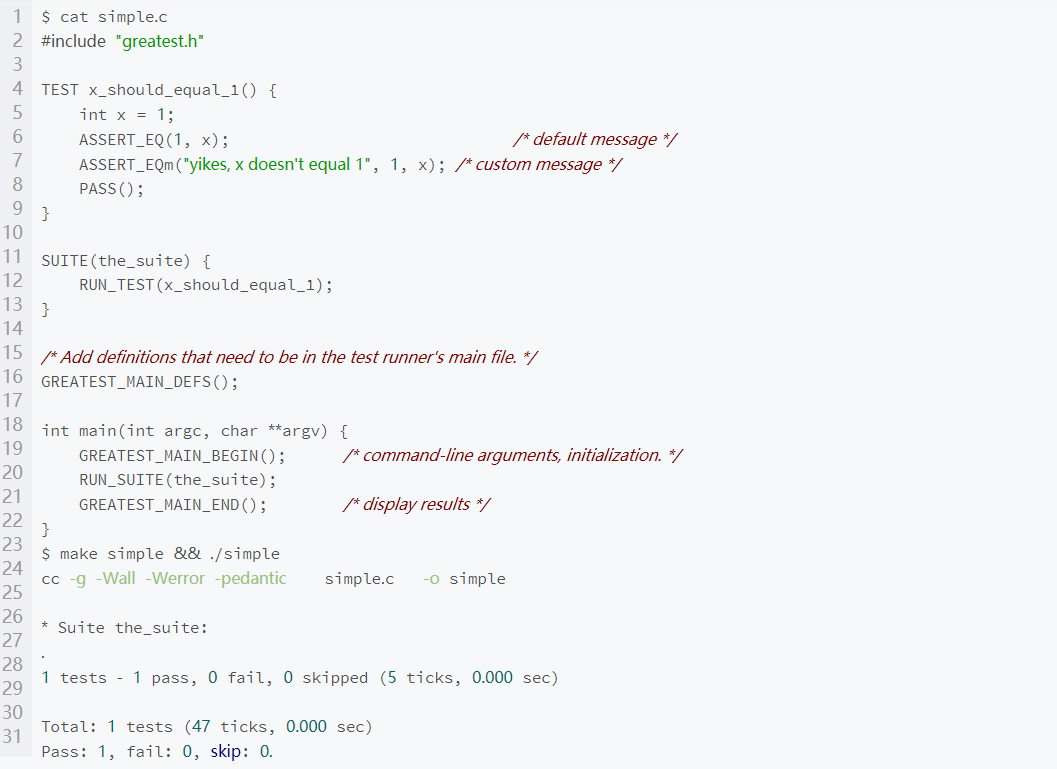
授权协议:未知

开发语言:[C/C++](https://www.oschina.net/project/lang/21/c)

操作系统:跨平台

greatest 详细介绍：

greatest 是一个 C 语言的单元测试系统，只需要一个文件支持。示例代码：



缺陷跟踪工具：

1、Redmine

简介：

Redmine 是一个开源的、基于Web的项目管理和缺陷跟踪工具。它用日历和甘特图辅助项目及进度可视化显示,同时它又支持多项目管理。Redmine是一个自由开放源码软件解决方案，它提供集成的项目管理功能，问题跟踪，并为多个版本控制选项的支持。

   虽说像IBM Rational Team Concert的商业项目调查工具已经很强大了，但想坚持一个自由和开放源码的解决方案，可能会发现Redmine是一个有用的Scrum和敏捷的选择。 由于Redmine的设计受到Rrac的较大影响，所以它们的软件包有很多相似的特征。

   Redmine建立在Ruby on Rails的框架之上，支持跨平台和多种数据库。

功能：

1、通过一套公共的基础数据和统一的权限控制信息共享，让项目干系人各司其职，各取所需。

2、项目负责人能够做计划、分任务、跟踪项目进展、处理项目问题、拟制项目报告;

3、研发工程师能够接受和处置任务、缺陷，管理和使用代码，查看需求甚至测试用例; （当然，如果想要更满足组织需求，你还需要做些其他的整合，如与testlink的整合等……）

4、测试工程师能够根据需求编写测试用例，查看需求覆盖情况，记录测试结果，输出测试报告，录入和跟踪缺陷;

5、产品经理、需求管理员能够规划产品版本和特性，条目化，版本化管理需求，跟踪产品问题和工作;

6、配置管理员进行开发库、受控库和产品库的管理，包括曰志记录、报表和授权；

7、质量人员能够跟踪会议评审记录，审计各类输出，监控项目关键过程;

8、项目管理办公室（PMO）能够对项目进度进行跟踪，查看单位人员组织矩阵、管理协作单位和进行资源规划;

9、部门领导能够查看成员的项目参与情况，在各项目中的工作任务和待处理问题，能够汇总统计员工的整体工作量、质量和延期情况，能够跟踪部门人员的主要工作;

10、甲方能够量化监控乙方情况，包括投入人力，项目测试和缺陷情况，节点通过情况，会议记录，甚至能够估算出乙方投入人天。（同样需要配合其他插件使用效果会更好）

2、TestDirector

简介：

TestDirector是Mercury Interactive公司推出的基于WEB的测试管理工具(现在已被HP公司收购)。它能够指导进行测试需求定义、测试计划、测试执行和缺陷跟踪，即整个测试过程的各个阶段。通 过整合所有任务到软件测试中，来使整个测试管理工作更有效，并确保客户收到更高质量的产品。

功能：

1、TestDirector能够让你系统地控制整个测试过程，并创建整个测试工作流的框架和基础，使整个测试管理过程变得更为简单和有组织。

2、TestDirector能够帮助你维护一个测试工程数据库，并且能够覆盖你的应用程序功能性的各个方面。在你的工程中的每一个测试点都对应着一个 指定的测试需求。To meet the various goals of a project, you organize the tests in your project into unique groups. TestDirector还为你提供了直观和有效的方式来计划和执行测试集、收集测试结果并分析数据。

3、TestDirector还专门提供了一个完善的缺陷跟踪系统，它能够让你跟踪缺陷从产生到最终解决的全过程。TestDirector通过与你的邮件系统相关联，缺陷跟踪的相关信息就可以被整个应用开发组，QA , 客户支持，负责信息系统的人员所共享。

4、TestDirector提供了与Mercury Interactive公司的测试工具（WinRunner, LoadRunner, QuickTest Professional, Astra QuickTest, QuickTest Professional for MySAP.com Windows Client, Astra LoadTest, XRunner, Visual APIand Visual API-XP）、第三方或者自主开发的测试工具、需求和配置管理工具、建模工具的整合功能。TestDirector能够与这些测试工具很好的无缝链接， 为你提供的全套解决方案选择来进行全部自动化的应用测试。

5、TestDirector会指导你进行需求定义、测试计划、测试执行和缺陷跟踪，即整个测试过程的各个阶段。通过整合所有的任务到应用程序测试中来确保你的客户收到更高质量的产品。

## Q6

**复印机系统**

# 一、复印机工作步骤

1、初始化一个复印机，复印机处于闲置状态

2、下达复印命令，复印机开始复印

3、若发现卡纸，进入卡纸状态

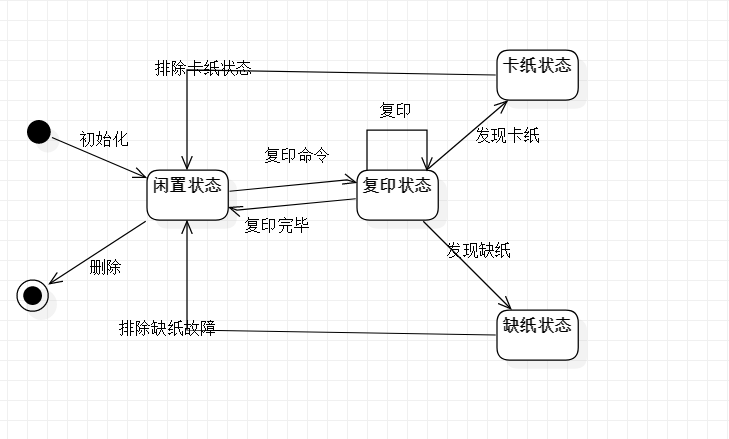
4、解决卡纸，回到闲置状态

5、若发现缺纸，进入缺纸状态

6、解决缺纸，回到闲置状态

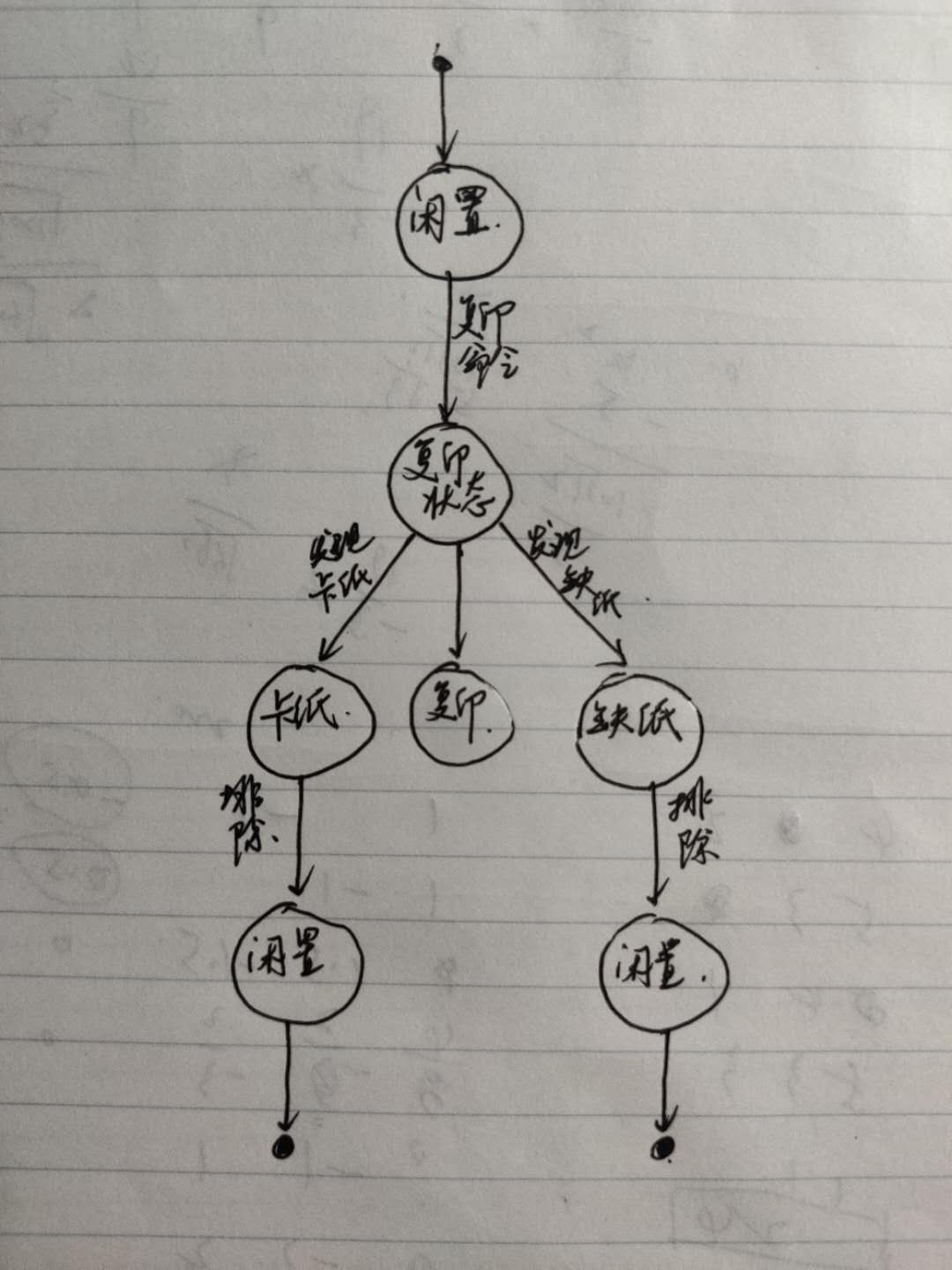
7、删除复印机

# 二、复印机工作状态图

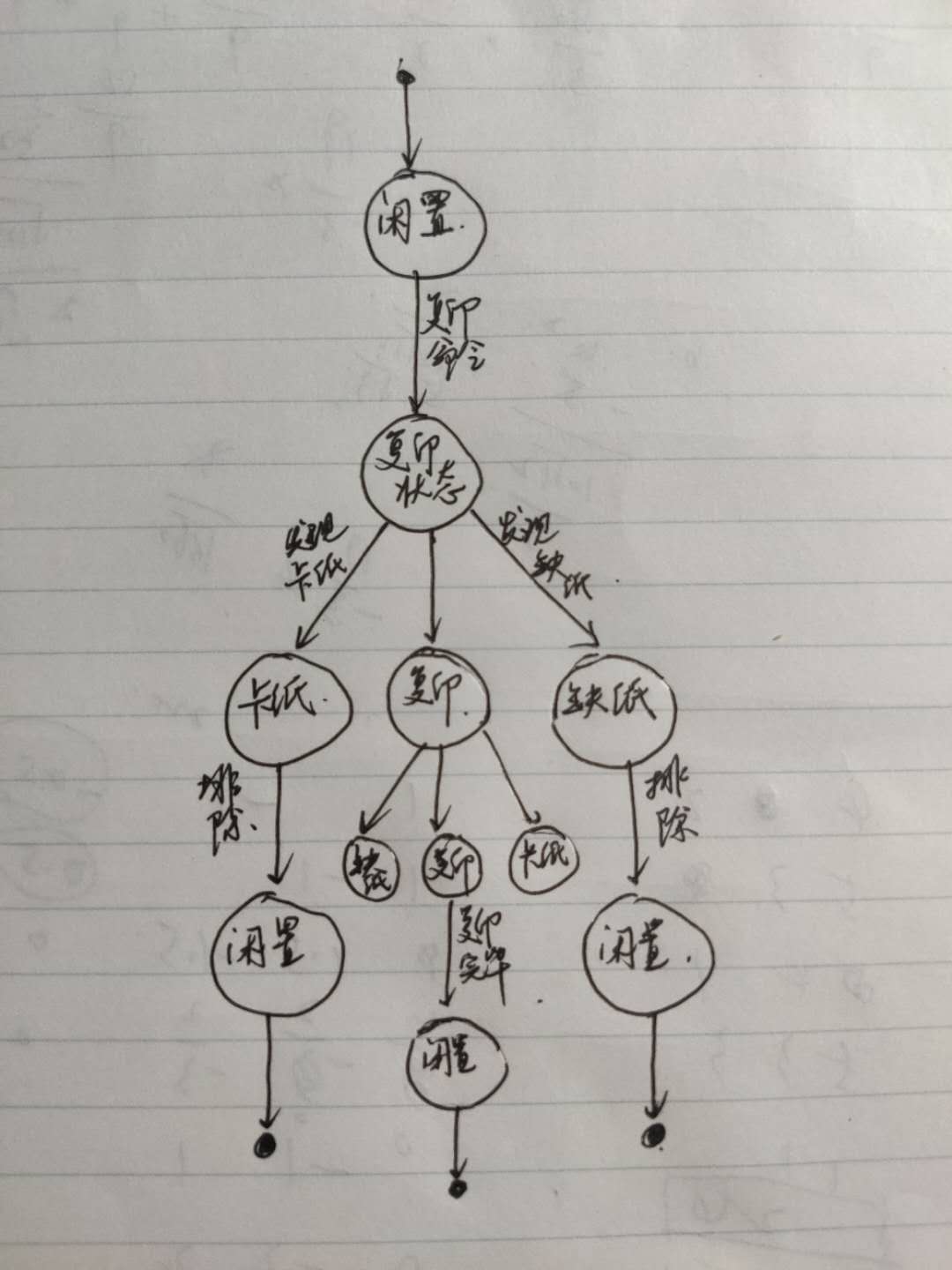


# 三、复印机状态转换树

**基本展开：**



**健壮展开：**

****

# 四、复印机测试用例

1、所有状态覆盖一次：

初始化[闲置状态]，复印命令[复印状态]，发现卡纸[卡纸状态]，解决卡纸[闲置状态]，复印命令[复印状态]，发现缺纸[缺纸状态]。

2、所有边覆盖一次：

初始化[闲置状态]，复印命令[复印状态]，复印命令[复印状态]，发现卡纸[卡纸状态]，解决卡纸[闲置状态]，复印命令[复印状态]，发现缺纸[缺纸状态]，解决缺纸[闲置状态]。

3、覆盖所有状态、事件和转换

初始化[闲置状态]，复印命令[复印状态]，复印命令[复印状态]，发现卡纸[卡纸状态]，解决卡纸[闲置状态]，复印命令[复印状态]，发现缺纸[缺纸状态]，解决缺纸[闲置状态]，复印命令[复印状态]，复印完毕[闲置状态]。

## 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

