何海江 撰写

《软件质量保证与测试》

《软件测试》

实验指导书

**二零一七年 八月**《软件质量保证与测试》介绍软件质量与软件测试的基础知识，软件测试的过程、技术、工具，以及软件质量管理的活动框架。通过教学，引导学生掌握软件测试的基本理论和基本方法，能熟练使用一些常用的测试工具，了解软件项目中软件测试的实施策略和实施过程。理论教学和实践教学相结合，学生能够根据需求规格说明决定合适的测试策略，具有综合运用软件测试方法、技术和工具的实践能力。

要求学生深刻理解软件质量的重要性，了解软件质量工程的活动框架。要求学生学习优秀测试员应具备的素质，掌握黑盒测试方法和白盒测试方法。要求学生理解软件测试的过程，理解兼容性测试、配置测试和安全性测试等非功能性测试内容，能使用自动化测试工具，并能编写具备较多工作量的测试脚本。要求学生能独立设计测试用例，能独立设计测试方案和撰写测试报告。

目 录

[“挡风玻璃雨刷系统”需求文档 4](#_Toc485277953)

[“挡风玻璃雨刷系统”代码 6](#_Toc485277954)

[实验：配置管理器的安装与使用 11](#_Toc485277955)

[实验：软件评审 24](#_Toc485277956)

[实验：编码规则与代码优化 30](#_Toc485277957)

[实验：程序的调试(Eclipse or MyEclipse) 35](#_Toc485277958)

[实验：面向对象测试技术 41](#_Toc485277959)

[测试计划模板 43](#_Toc485277960)

[测试用例模板 53](#_Toc485277961)

[航班管理设计及代码 57](#_Toc485277962)

[实验：黑盒测试 82](#_Toc485277963)

[实验：白盒测试 87](#_Toc485277964)

[实验：JUnit的安装和使用 90](#_Toc485277965)

[实验：FEST-Swing结合JUnit自动化测试 98](#_Toc485277966)

[实验：QTP的脚本录制 103](#_Toc485277967)

[实验：QTP自动化测试进阶 106](#_Toc485277968)

[实验彩蛋 107](#_Toc485277969)

# “挡风玻璃雨刷系统”需求文档

某品牌汽车的挡风玻璃雨刷是由带刻度盘的操纵杆控制的。这种操纵杆有四个位置，分别是停止、间歇、低速和高速。在断电的情况下，操纵杆的位置不能变化；加电时，操纵杆的位置依照下图变化：

操纵杆抬起

操纵杆放下

操纵杆放下

操纵杆放下

**高速**

**低速**

操纵杆抬起

操纵杆抬起

**间歇**

**停止**

刻度盘有三个位置，分别是数字1、2和3，只有当操纵杆在间歇位置上，刻度盘的位置才有意义，刻度盘位置指示三种间歇速度。同样，在断电的情况下，刻度盘的位置不能变化；加电时，刻度盘的位置依照下图变化：

刻度盘减小

刻度盘增加

刻度盘增加

**1**

刻度盘减小

**2**

**3**

下表给出了挡风玻璃雨刷对应操纵杆和刻度盘的工作速度（每分钟摇摆次数）。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 操纵杆 | 停止 | 间歇 | | | 低速 | 高速 |
| 刻度盘 | ----- | 1 | 2 | 3 | ----- | ----- |
| 雨刷速度 | 0 | 4 | 6 | 12 | 30 | 60 |

1，以类的方法为单元，单元测试例子，应用分支覆盖技术。

操纵杆“抬起”事件的测试用例

2，以类为单元，单元测试例子，应用场景测试技术。

3，集成测试，应用场景测试技术。

挡风玻璃雨刷的测试用例场景：

挡风玻璃雨刷操纵杆在“停止”位置，刻度盘在位置1；加电，用户将操纵杆推到“间歇”，然后将刻度盘从位置1主导位置2，再转到位置3；然后将操纵杆推到“低速”；用户将操纵杆推到“间歇”；然后推到“停止”。

4, 自测试功能

# “挡风玻璃雨刷系统”代码

工程名：WindshieldWiper

DialSense.java

/\*刻度盘类 作者： 何海江 \*/

public class DialSense {

private int nPointer; //刻度盘指针

private int nStatus;//电源状态

//构造函数

public DialSense()

{

nPointer = 1; //刻度盘指针起始位置： 1

nStatus = PowerStatus.OFF; //电源是关闭状态

}

//构造带参数的操纵杆类

public DialSense(int nPower,int nPos)

{

nPointer = nPos; //刻度盘指针

nStatus = nPower; //电源状态

}

//增加刻度盘,只有带电才起作用。

public void Increment()

{

if( PowerStatus.OFF==nStatus )

return;

if( 1==nPointer ) nPointer = 2;

else if( 2==nPointer ) nPointer = 3;

else nPointer = 3;

}

//调小刻度盘,只有带电才起作用。

public void Decrement()

{

if( PowerStatus.OFF==nStatus )

return;

if( 3==nPointer ) nPointer = 2;

else if( 2==nPointer ) nPointer = 1;

else nPointer = 1;

}

//获得刻度盘指针

public int GetPointer()

{

return nPointer;

}

//设定刻度盘指针。

public void SetPointer(int nPos)

{

nPointer = nPos;

}

//改变电源状态

public void SetPowerStatus( int nps )

{

nStatus = nps;

}

}

LeverPosition.java

//枚举：控制杆的位置

public final class LeverPosition

{

static final int OFF = 1; //关闭控制杆

static final int INT = 2; //间歇

static final int LOW = 3; //低速

static final int HIGH =4; //高速

}

PowerStatus.java

//枚举：电源状态

public class PowerStatus {

static final int OFF = 1; //关闭

static final int ON = 2; //开启

}

LeverSense.java

/\*操纵杆类 作者： 何海江 \*/

public class LeverSense {

private int nPosition; //操纵杆的位置

private int nStatus; //电源状态

//构造函数

public LeverSense()

{

nPosition = LeverPosition.OFF; //操纵杆处于“关”

nStatus = PowerStatus.OFF; //电源是关闭状态

}

//构造带参数的操纵杆类

public LeverSense(int nPower,int nPos)

{

nPosition = nPos; //操纵杆位置

nStatus = nPower; //电源状态

}

//操纵杆抬起,只有带电才起作用。

public void Up()

{

if( PowerStatus.OFF==nStatus )

return;

switch( nPosition )

{

case LeverPosition.OFF:

nPosition = LeverPosition.INT; break;

case LeverPosition.INT:

nPosition = LeverPosition.LOW; break;

default:

nPosition = LeverPosition.HIGH; break;

}

}

(接前一页)

public void Down()

{//操纵杆放下,只有带电才起作用。

if( PowerStatus.OFF==nStatus )

return;

switch( nPosition )

{

case LeverPosition.HIGH:

nPosition = LeverPosition.LOW; break;

case LeverPosition. LOW:

nPosition = LeverPosition.INT; break;

default:

nPosition = LeverPosition.OFF; break;

}

}

public int GetPosition()

{//获得操纵杆的位置

return nPosition;

}

public void SetPosition(int nPos)

{//设定操纵杆的位置。

nPosition = nPos;

}

public void SetPowerStatus(int nps )

{//改变电源状态

nStatus = nps;

}

//测试我自己的Up()

//返回0，所有测试用例执行结果符合预期；否则，在第（返回值）个测试用例，与预期不符。

public int TestMyUp()

{

//第一个测试用例，断电后不能改变状态。

SetPowerStatus(PowerStatus.OFF);

SetPosition(LeverPosition.LOW);

Up();

if( LeverPosition.LOW!=GetPosition() ) //起始状态为LOW，抬起操纵杆后状态仍为LOW。

return 1;

//第二个测试用例，加电和抬起操纵杆后状态从LOW->HIGH。

SetPowerStatus(PowerStatus.ON);

SetPosition(LeverPosition.LOW);

Up();

if( LeverPosition.HIGH!=GetPosition() )

return 2;

//第三个测试用例，加电和抬起操纵杆后状态从INT->LOW。

SetPowerStatus(PowerStatus.ON);

SetPosition(LeverPosition.INT);

Up();

if( LeverPosition.LOW!=GetPosition() )

return 3;

//第四个测试用例，加电和抬起操纵杆后状态从OFF->INT。

SetPowerStatus(PowerStatus.ON);

SetPosition(LeverPosition.OFF);

Up();

if( LeverPosition.INT!=GetPosition() )

return 4;

return 0;

}

}

WindshieldWiper.java

/\*挡风玻璃雨刷类\*/

public class WindshieldWiper {

private LeverSense lever; //操纵杆

private DialSense dial; //刻度盘

private int nPowerStatus; //电源状态

//构造函数

public WindshieldWiper()

{

nPowerStatus = PowerStatus.OFF;

}

//构造带参数的挡风玻璃雨刷类

public WindshieldWiper(int nPower,int nLeverPos,int nDialPot)

{

nPowerStatus = nPower;

lever.SetPowerStatus(nPower);

lever.SetPosition(nLeverPos);

dial.SetPowerStatus(nPower);

dial.SetPointer(nDialPot);

}

//设置操纵杆和刻度盘的位置

public void SetLeverDial(int nLeverPos,int nDialPot)

{

lever.SetPosition(nLeverPos);

dial.SetPointer(nDialPot);

}

//加电

public void PowerOn()

{

nPowerStatus = PowerStatus.ON;

lever.SetPowerStatus(PowerStatus.ON);

dial.SetPowerStatus(PowerStatus.ON);

}

//断电

public void PowerOff()

{

nPowerStatus = PowerStatus.OFF;

lever.SetPowerStatus(PowerStatus.OFF);

dial.SetPowerStatus(PowerStatus.OFF);

}

//获得雨刷的速度

public int GetSpeed()

{

int nSpeed = 0;

if( PowerStatus.OFF==nPowerStatus )

return nSpeed = 0; //断电的状态，速度为零。

switch(lever.GetPosition())

{

case LeverPosition.OFF:

nSpeed = 0; break;

(接前一页)

case LeverPosition.INT:

int ndp = dial.GetPointer();

if(ndp==1)

nSpeed = 4;

else if(ndp==2)

nSpeed = 6;

else

nSpeed = 12;

break;

case LeverPosition.LOW:

nSpeed = 30; break;

case LeverPosition.HIGH:

nSpeed = 60; break;

}

return nSpeed;

}

}

主程序OOTestCase.java

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 测试挡风玻璃雨刷系统。

\* 作者： 何海江

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class OOTestCase {

public static void main(String args[])

{

//操纵杆"抬起"事件的测试用例

System.out.println("操纵杆抬起事件的测试用例");

LeverSense lSense = new LeverSense(LeverPosition.OFF,PowerStatus.ON);

/\*System.out.println("操纵杆的位置在关闭状态，加电后，抬起，状态预期为：间歇");

lSense.Up();

if( LeverPosition.INT==lSense.GetPosition() )

System.out.println("操纵杆的位置当前为：间歇");

else

System.out.println("错误：其它位置！");\*/

//LeverSense的自测试

System.out.println("LeverSense的自测试开始......");

int nErr = lSense.TestMyUp();

if( nErr>0 )

{

System.out.print(nErr);

System.out.println(" 开始出错。");

}

else

System.out.println("所有测试用例成功。");

}

}

# 实验：配置管理器的安装与使用

1. 实验名称：配置管理器的安装与使用
2. 实验目的及要求：

**实验目的** 使用版本控制的基本功能, 更好地理解软件配置管理。

**实验要求** 验证所有实验步骤，完成测试报告。

1. 实验环境

Subversion Tortoise

1. 实验内容

安装Subversion和Tortoise，并按照实验指导书的要求验证它们的基本功能。实验详细内容参照“实验指导书- **subversion**配置管理器的安装与使用”。

1. 实验步骤：

(写出实验步骤)

六、安装与使用过程及实验结果

与实验步骤合并。

**Subversion** 配置管理器的安装与使用

Subversion是一个自由、开源的版本控制系统。在Subversion管理下，文件和目录可以超越时空。Subversion将文件存放在中心版本库里。这个版本库很像一个普通的文件服务器，不同的是，它可以记录每一次文件和目录的修改情况。这样就可以藉此将数据恢复到以前的版本，并可以查看数据的更改细节。

**简介**

Subversion(简称svn)是一个版本控制系统，它的版本库可以通过网络访问，从而使用户可以在不同的电脑上进行操作。从某种程度上来说，允许用户在各自的空间里修改和管理同一组数据，可以促进团队协作。此外，由于所有的工作都已版本化，也就不必担心由于错误的更改而影响软件质量——如果出现不正确的更改，只要撤销那一次更改操作即可。

**功能**

·包含绝大部分CVS的功能

·目录的版本化

Subversion将目录名以版本号的形式体现。

·基于版本的复制，删除和重命名

无论复制，删除还是重命名，都会被打上版本号。

·自由的版本化元数据操作

　Subversion允许任何元数据附加在文件或目录中。这些属性是键/值对，并且被版本化。subversion也提供对修订版附加任何键/值属性的方法，这些属性不会被版本化，因为他们会自动将元数据附加到版本空间中，但他们可以随时被更改。

·混合追踪

Subversion 1.5开始加入了混合追踪功能。

·文件锁

　　支持文件锁定，当多个用户试图编辑同一个文件时会收到警告。

·Apache网络服务的支持，基于WebDAV/DeltaV协议

使用基于HTTP的WebDAV/DeltaV协议进行网络通信，而Apache网络服务器提供网络存储的站点服务。

·可执行的标签

　　当一个文件是可执行的时候，Subversion会提示，并且当这个可执行的文件被放在版本控制中时，Subversion会防止该程序检查其他目录。

·独立进程模式

Subversion可以运行在独立模式下。

·一个只读的存储镜像

　　Subversion提供一个工具，SVNsync， 用于同步主服务器上的文件到一个子存储服务器上，并且标为只读的属性。

**实验准备**

Subversion有服务器和客户端。工作状态，服务器单独安装在一台机器，可不装客户端。而客户端安装在开发团队每个成员的计算机中。这里装在一起，便于实验而已。

客户端：Subversion的客户端有两类，一类是webSVN等基于web的，一种是以TortoiseSVN为代表的客户端软件。前者需要web服务器的支持，后者需要用户在本地安装客户端，两种都有免费开源软件供使用。

服务器：Subversion支持linux和windows，更多是安装在linux下。SVN服务器有2种运行方式：独立服务器和借助Apache。2种方式各有利弊。

SVN存储版本数据也有2种方式：BDB和FSFS。因为BDB方式在服务器中断时，有可能锁住数据，所以还是FSFS方式更安全一点。

服务器程序：http:// sourceforge.net/projects/win32svn/files/

（Setup-Subversion-1.7.6.msi）。

客户端程序：http://sourceforge.net/projects/tortoisesvn/files

**( TortoiseSVN-1.7.9.23248-win32-svn-1.7.6.msi ）**

(**注意：**服务器程序和客户端程序用win32非X64的版本，你懂的。点击Application可找到)

中文包http://sourceforge.net/projects/tortoisesvn/files/1.7.9/Language Packs

（LanguagePack\_1.7.9.23248-win32-zh\_CN.msi）

在E盘创建目录Subversion，假定服务器安装目录目录为：G:\Subversion

在E盘创建目录TortoiseSVN，假定客户端安装目录目录为：G:\ TortoiseSVN

在E盘创建目录**G:\MySVN\Repository,**作为版本库（Repository）。

**实验步骤**

**（一）下载及安装**

**1，软件下载**

下载Subversion服务器程序。到官方网站的下载二进制安装文件。

下载Subversion的Windows客户端TortoiseSVN。TortoiseSVN是扩展Windows Shell的一套工具，TortoiseSVN提供了图形界面以及和windows系统菜单的集成，可以看作Windows资源管理器的插件，安装之后Windows就可以识别Subversion的工作目录。

**2，服务器和客户端安装**

服务器安装，直接运行Setup-Subversion-1.7.6.msi，根据提示安装即可(修改服务器安装目录为：G:\Subversion)，这样我们就有了一套服务器可以运行的环境。

安 装TortoiseSVN，同样直接运行**TortoiseSVN-1.7.9.23248-win32- svn-1.7.6.msi，**按照提示安装即可(修改客户端安装目录为：G:\ TortoiseSVN)，不过最后完成后会提示是否重启，其实重启只是使SVN工作拷贝在windows中的特殊样式生效，与所有的实际功能无关，为了立刻看到好的效果，还是重新启动机器。**注意学校的机器不能重启。**

重启完毕后安装简体中文语言包, 然后再随便一个目录右击, 就会发现多出了一些SVN相关菜单, 选择其中的TortoiseSVN, 再选择子菜单"Settings", 设置Language为"中文(简体)"。下图描述了中文(简体)包的设置菜单。



**（二）独立开发者的使用**

对于一个实际的项目，往往需要在一个安全的地方创建版本库并设置 Subversion 服务器来控制它。而对于本步骤而言，我们将会使用 Subversion 的本机版本库功能，该特性使得用户可以直接访问本机硬盘上的版本库而不需要服务器。

**1，建立本机版本库（Repository）**

运行Subversion服务器需要首先要建立一个版本库（Repository），可以看作服务器上存放数据的数据库，在安装了Subversion服务器之后，我们可以使用TortoiseSVN图形化完成这一步：

在目录G:\MySVN\OwnReposit（注意不是Repository，Repository将在实验步骤三以后部分作为远程服务器的版本库）下"右键->TortoiseSVN->Create Repository here...“， 中文版是"右键->TortoiseSVN->在此创建文件库"。然后可以选择版本库模式，这里使用默认即可，然后就创建了一系列目录和文件。



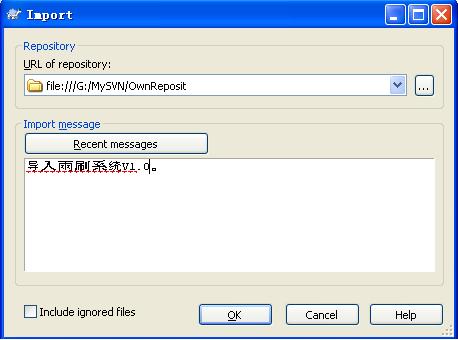
**2，初始化导入（最开始的代码，也就是基线）**

来到我们想要导入的项目根目录，在这个例子里是挡风玻璃雨刷系统G:\LocalWWiper(G:\WindWiperServer将在实验步骤三以后部分作为导入代码目录)。

右键->TortoiseSVN->Import...

URL of repository输入“svn://localhost/OwnReposit”

另一种格式，URL of repository输入“file:///G:/MySVN/OwnReposit”



此处特别注意，输入版本库的目录，非代码目录。

这个对话框中另一个重要的功能就是导入信息文本框，你可以输入一段信息来描述你的操作。当你查看项目的历史时，这些提交信息是记录了你修改什么和为什么修改的宝贵资料。在这个例子中，我们可以简单的写一下，例如“导入雨刷系统V1.0”。点击确定，文件夹就加入到版本库中了。

完成之后目录没有任何变化，如果没有报错，数据就已经全部导入到了我们刚才定义的版本库中。

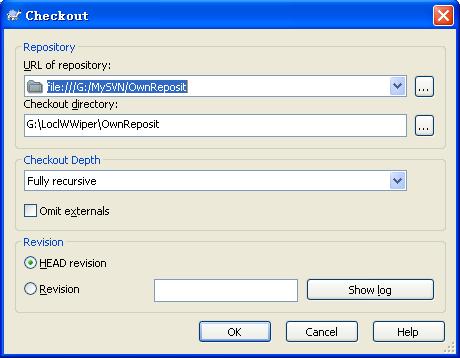
**3，检出工作副本(从版本库中找到最新代码)**

在G:\LocalWWiper下建立一个子目录OwnReposit，准备在G:\LocalWWiper\OwnReposit编辑、调试程序。当然，目录OwnReposit最好不要放在G:\LocalWWiper下，同学们搞混问题。

第一个输入框是版本库。

第一个输入框是工作目录。

注意：我这里做得不好，将OwnReposit放在G:\LocalWWiper下，并且目录名字取得不好，容易与版本库混淆。



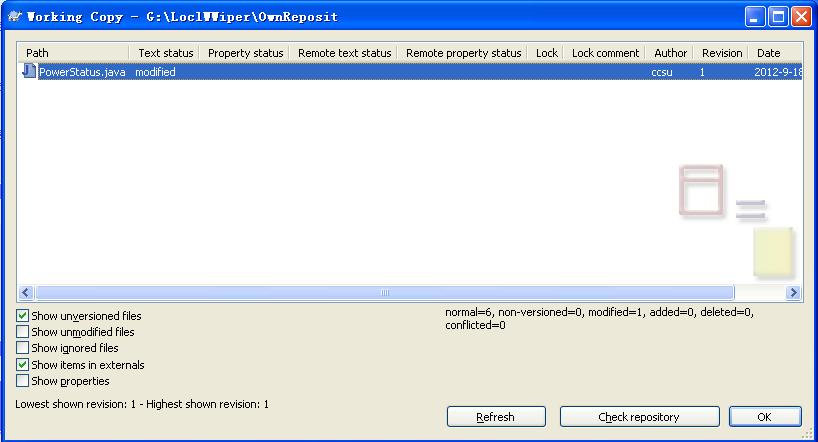
原始代码目录是LocalWWiper。此处应该输入调试程序的工作目录。

你会注意到（要机器重新启动），这个文件夹看起来与我们原来的文件夹不一样。每一个文件的左下角都有一个绿色的对钩。它们就是只出现在工作副本中的 TortoiseSVN 状态图标。绿色的图标表示文件未被修改，和版本库中的文件版本一致。

4，进行修改

工作时间到了。 在目录OwnReposit中，我们开始编辑文件：假设我们对PowerStatus.java 进行了修改。注意，这些文件的重载图标变成了红色，这说明本机文件被修改了。但是我们做了哪些更改？右键单击目录OwnReposit然后选择 TortoiseSVN → 比较差异（英文：Check for modifications）。启动TortoiseSVN 的目录比较工具，准确地显示哪些文件被修改了。如上图所示，在被修改的文件上鼠标双击，科查看详细的修改情况。

右键单击任意一个修改过的文件然后选择 TortoiseSVN → 比较差异。启动TortoiseSVN 的文件比较工具，准确地显示哪些行被修改了。



此处双击，可查看详细结果。

好的，我们对这些更改很满意，让我们更新版本库。这个动作叫“提交”更改。右键单击文件夹OwnReposit 然后选择 TortoiseSVN → 提交（英文：Commit…）。提交对话框列出了修改过的文件，每一个都有一个复选框。你可以选中列表中的部分文件，但在这个例子中我们将要提交全部修改过文件。输入一段信息来描述做了什么修改，然后单击确定。进度对话框显示被上传到版本库中的文件并且完成提交。

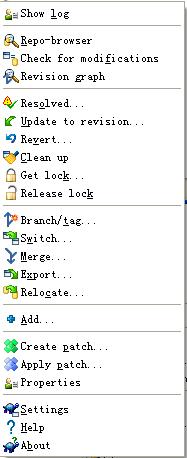
**5，添加更多的文件**

随着项目的发展，需要添加新文件 - 例如说你要添加新功能，有了新文件 Extras.java 还要在现有的文件WindshieldWiper.java 中加入起对该类的引用。右键单击文件夹然后选择 TortoiseSVN → 增加。 加入对话框显示了所有未被版本控制的文件，你可以选择哪些文件要被添加。另一个增加文件的方法是右键单击文件自身然后选择 TortoiseSVN → 加入。现在当你提交文件夹时，新文件会显示为增加，原有的文件显示为修改。注意你可以双击修改的文件查看做了哪些改动。

**6，查看项目历史**

TortoiseSVN 最有用的功能之一便是日志对话框。它显示针对文件或文件夹所有的提交记录列表，并显示提交者输入的那些详细的提交信息。 ;-)

**查看项目历史**



**复原到此版本**

**撤消更改**

**添加更多的文件**

**（三）团队开发者的使用**

**1，建立版本库（Repository）**

运行Subversion服务器需要首先要建立一个版本库（Repository），可以看作服务器上存放数据的数据库，在安装了Subversion服务器之后，我们可以使用TortoiseSVN图形化完成这一步：

在目录G:\MySVN\Repository下"右键->TortoiseSVN->Create Repository here...“， 中文版是"右键->TortoiseSVN->在此创建文件库"。然后可以选择版本库模式，这里使用默认即可，然后就创建了一系列目录和文件。

（此部分前面已经有，操作菜单的图形就省略了。）

**2，配置用户和权限**

打开G:\MySVN\Repository, 你会发现已经多了一些目录和文件, 打开conf子目录, 打开svnserve.conf文件, 这里行前凡是有#的都等于是被注释忽略了, 你可以把#去掉让那一行生效, 或者自己新添加行. 里面的英文注释已经详细说明了各种设置的含义, 最后你设置 [general]小节中行前没有#号的内容为:

anon-access = none

auth-access = write

password-db = passwd

含义是：未验证用户无任何权限 (如果把none修改为read就是给予读权限)；已验证用户给予写权限 (当然也能读)；密码数据存放到passwd文件中。

然后打开同目录的passwd文件来设置帐户。同样, 设置[users]小节中行前没有#号的内容, 例如:

admin=123

含义是：用户admin的密码为123

**3，运行独立服务器**

在任意目录下运行：

**svnserve -d -r G:\MySVN\Repository** 我们的服务器程序就已经启动了。注意不要关闭命令行窗口，关闭窗口也会把svnserve停止。在其窗口中按Ctrl+C也可中止服务器运行。在客户端执行的操作都会保存到**G:\MySVN\Repository**。每位同学可以建立两个库，一个供自己用（G:\MySVN\OwnReposit，独立开发者模式）；另一个供别人用（**G:\MySVN\Repository，**团队开发者模式）。

d 是daemon的简写，r 是root的简写。

（下面的实验步骤不要求做，可作为课外自学部分。）

另外一种实用的方法，用Windows服务，完成其它实验任务的可操作该步骤。注意，该服务器包内未找到程序svnservice.exe。

sc config svnservice start= auto

net start svnservice

第一条语句将SVN变成windows系统的一个服务，并默认为自启动。

第二条语句启动Windows服务，也可以使用图形化的工具修改服务的属性，你可以在“开始->运行...”中执行"services.msc"，然后在界面中修改。

停止服务：使用"net stop svnservice"停止服务。

删除服务：如果服务安装的有问题，你可能需要删除服务。要删除前面添加的服务，只需要运行"sc delete svnservice"。

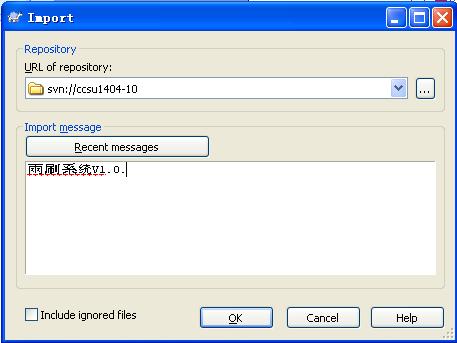
**4，初始化导入（最开始的代码，基线）**

来到我们想要导入的项目根目录，在这个例子里是挡风玻璃雨刷系统G:\WindWiperServer。

右键->TortoiseSVN->Import...

URL of repository输入“svn:// 133.96.121.22/” 或者“svn:// ccsu1404-10”。

运行SVNserve的主机的IP是133.96.121.22(机器名：ccsu1404-10)。



接下来会问你用户名和帐号, 你就填写前面搭建服务器端所设置的用户admin密码123.

完成之后目录没有任何变化，如果没有报错，数据就已经全部导入到了我们刚才定义的版本库中。基线数据保存到机器ccsu1404-10上。

**5，检出工作副本(从版本库中找到最新代码)**

创建一个准备用来存放版本控制工程的目录,就是存放了当前最新版本的代码，程序的编写和修改都在这个目录完成。例如G:\MyWiper, 然后在"我的电脑"中右击这个目录, 选择"SVN 检出..."( Checkout)。

6，进行修改

（此部分参考个人开发者内容，省略……）

**7，添加更多的文件**

（此部分参考个人开发者内容，省略……）

**8，查看项目历史**

（此部分参考个人开发者内容，省略……）

**9，撤消更改**

所有的版本控制系统都有的功能就是让你可以撤销之前做的更改。如你预料的一样，TortoiseSVN 非常容易做到这一点。如果你想抛弃还没有提交的更改并将文件复原到修改之前的状态，TortoiseSVN → SVN 还原（Revert）就是你的好伙伴。它抛弃了所做的更改(扔到回收站里)并复原到修改之前的版本。TortoiseSVN →Revert仅能撤消你本地的变更。他不能撤消已经提交的的变更。

**（四）**用来自别人的修改更新你的工作副本

假定别人就是服务器，你是客户端。

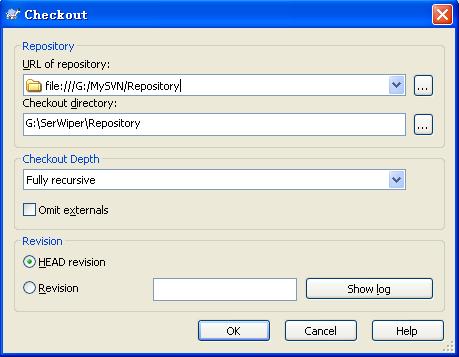
1，你加一个文件，叫aaa.txt。

2，用TortoiseSVN → 增加，将aaa.txt加入到你的工作副本。

3，用TortoiseSVN →提交，这样aaa.txt成为了项目的一部分。

4，在服务器上，别人执行下述操作：

（1）用SVN →CheckOut，将版本库的文件导出到别人的工作副本。比如在服务器建立了文件夹G:\SerWiper\Repository；在CheckOut对话框填入



填入团队开发的版本库。

别人的工作副本，此处可以修改成希望的文件夹。注意：此文件夹在服务器。

（2）检查看看是否符合预期；

（3）修改文件aaa.txt，再TortoiseSVN →提交。这样，就相当于开发团队的某位成员修改了代码，并且上传到版本库。

5，回到客户端。用TortoiseSVN →update，检查aaa.txt里面的内容是否修改，与别人的代码相同。

**(五)解决文件冲突**

假定别人就是服务器，你是客户端。

1，你加一个文件，叫bbb.txt。

2，用TortoiseSVN → 增加，将bbb.txt加入到你的工作副本。在bbb.txt键入一些字符。

3，用TortoiseSVN →提交，这样bbb.txt成为了项目的一部分。

4，在服务器上，别人执行下述操作：

（1）用SVN →CheckOut，将版本库的文件导出到别人的工作副本。比如在服务器建立了文件夹G:\TestConf；在CheckOut对话框填入

版本库 和 G:\TestConf

（2）检查看看是否符合预期；

（3）修改文件bbb.txt，再TortoiseSVN →提交。这样，就相当于开发团队的某位成员修改了代码，并且上传到版本库。

5，回到客户端。修改bbb.txt里的内容。再用TortoiseSVN →update，检查bbb.txt里面的内容是否如帮助文档描述。注意，此时产生了冲突。

6，你首先和团队人员商量，决定如何修改代码，消除冲突。你按照约定修改代码，然后用**TortoiseSVN→已解决。**注意观察G:\TestConf下冲突文件已经删除。再用**TortoiseSVN→提交，**将新的修改信息传到数据库。

7，团队成员，别人可以在服务器上**TortoiseSVN→Update.**

**(六)探索功能**

**TortoiseSVN→Update to reversion…**将该菜单的功能写进试验报告，并列出你的试验步骤。

下面是帮助文档，作为步骤（五）的参考资料。

偶尔，当你从版本库更新、合并文件时，或者切换工作副本至一个不同的 URL 时你会遇到冲突。有两种冲突:

**文件冲突** 当两名(或更多)开发人员修改了同一个文件中相邻或相同的行时就会发生文件冲突。

**树冲突** 当一名开发人员移动、重命名、删除一个文件或文件夹，而另一名开发人员也对它们进行了移动、重命名、删除或者仅仅是修改时就会发生树冲突。

本次试验只考虑文件冲突。

当两名或更多开发人员修改了同一个文件中相邻或相同的行时就会发生文件冲突。由于Subversion 不知道你的项目的具体情况，它把解决冲突的工作留给了开发人员。一旦出现冲突，你就应该打开有问题的文件，查找以字符串<<<<<<<开头的行。有冲突的区域用如下的方式

标记:

<<<<<<< 文件名

你的修改

=======

合并自版本库中的代码

>>>>>>> 版本

对于每个冲突的文件 Subversion 在你的目录下放置了三个文件:

文件名.ext.mine

这是你的文件，在你更新你的工作副本之前存在于你的的工作副本中——也就是说，没有冲突标志。这个文件除了你的最新修改外没有别的东西。

文件名.ext.r旧版本

这是在你更新你的工作副本之前的基础版本(BASE revision)文件。也就是说，它是在你做最后修改之前所检出的文件。

文件名.ext.r新版本

这个文件是当你更新你的工作副本时，你的 Subversion 客户端从服务器接收到的。这个文件对应于版本库中的最新版本。

你可以通过T**ortoiseSVN→ 编辑冲突**运行外部合并工具/冲突编辑器，或者你可以使用任何别的编辑器手动解决冲突。你需要冲定哪些代码是需要的，做一些必要的修改然后保存。然后，执行命令 **TortoiseSVN→已解决** 并提交人的修改到版本库。需要注意的是已解决命令并不是真正的解决了冲突，它只是删除了filename.ext.mine和filename.ext.r\*两个文件，允许你提交修改。如果你的二进制文件有冲突，Subversion不会试图合并文件。本地文件保持不变(完全是你最后修改时的样子)，但你会看到filename.ext.r\*文件。如果你要撤消你的修改，保留版本库中的版本，请使用还原(Revert)命令。如果你要保持你的版本覆盖版本库中的版本，使用已解决命令，然后提交你的版本。

你可以右击父文件夹，选择**TortoiseSVN→已解决**...，使用“已解决”命令来解决多个文件。这个操作会出现一个对话框，列出文件夹下所有有冲突的文件，你可以选择将哪些标记成已解决。

# 实验：软件评审

实验名称：软件评审

1. 实验目的：

熟悉软件评审的内容和流程。

1. 实验原理：

软件评审的角色和职能，软件评审的内容，软件评审的方法和技术。

1. 实验环境

Visual C++

1. 实验内容与步骤

查看使用手册、设计报告和源代码，检查运行情况；设计并填写缺陷检查表；从不同角色出发对设计报告、产品和使用手册评审。实验详细内容参照“实验指导书”。 查看工程WswProject的程序说明。

(写出实验步骤)

五、实验结果与分析。

**一，设计并填写缺陷检查表**

**检查表编写指南**

检查表（Checklist）是一种常用的质量保证手段。人们借助检查表以确认被检查对象的所有质量特征均得到满足。检查表归纳了所有检查要点，比起冗长的文档，使用检查表具有更高的工作效率。在正规技术评审中，检查表用来帮助评审员找出被审对象中可能的缺陷，评审过程由检查表驱动。因此，检查表是举行正规技术评审的必要工具。一份精心设计的检查表，对于提高评审效率、改进评审质量具有重要意义。

步骤：

（1）根据以往积累的经验收集同类评审对象的常见缺陷，或者从现有检查表中选择合适的作为基础。

（2）按缺陷的类型和子类型进行组织，并为每一个缺陷类型指定一个标识码。标识码将在评审中被用来分类缺陷类型。

（3）以简单问句的形式表达每一种缺陷。所谓简单问句，其答句为“是”或“否”。答句为“是”表示存在此缺陷；答句为“否”表示未发现此缺陷。

（4）按照各种缺陷对软件影响的严重性和（或）发生的可能性从大至小排列缺陷类型和子类型。各种缺陷发生的可能性可以基于以往的软件问题报告和个人经验。

（5）根据本次评审对象的质量要求和其他特性，对检查表中的问题作必要的增、删、修改和前后次序调整。

原则：

（1）不同类型的评审对象应编制不同的检查表。需求分析、概要设计、详细设计、程序代码、测试计划、测试案例、系统切换方案、用户操作手册等是不同类型的评审对象,应编制不同的检查表。C程序、JAVA、Visual Basic等也是不同类型评审对象，应使用不同的检查表。进而言之，被审对象采用不同的中间件和数据库管理系统，相应的检查表也可能不同。

（2）在检查表中，应排除文档编辑软件能够识别的拼写错误和编程语言编译器能够识别的语法错误。这类检查应由作者在提交被审对象前完成。正规技术评审的主持人负责检查被审文档是否经过文档编辑软件的拼写检查、被审程序代码是否通过编译。

（3）检查表应着重于“严重”（Major）缺陷。所谓“严重”缺陷，指会引起系统失败的错误、导致系统的功能不符合需求的错误，或者严重违反开发规范的错误。其余缺陷称为“次要”（Minor）缺陷。不要将次要缺陷包括在检查表中。

（4）仔细权衡检查表的长度与评审成本的关系。一般而言，检查表中列出的问题越多，从被审对象中找出问题的数量会越多，但总的评审花费也会越高。

（5）随着公司业务领域的拓展、软件开发质量的提高、开发人员技术水平的变化，检查表应经常加以更新，以反映最新的常见错误和质量目标，提高评审效率。

（6）检查表是评审的出发点，但在评审过程中不应局限于检查表中的问题。评审员应根据自己的经验和判断寻找其他可能的缺陷。

**设计、填写缺陷检查表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **评审内容** | **评审结果** | | |
| **是** | **否** | **不确定** |
| **1** | 用户界面是否与软件的功能相融洽？ |  |  |  |
| **2** | 用户界面是否适合于用户的应用环境？ |  |  |  |
| **3** | 界面元素有错别字，或者措词含糊、逻辑混乱。 |  |  |  |
| **4** | 消息框的提示文字和按钮在语义上不连贯。 |  |  |  |
| **5** | 提供联机帮助吗？ |  |  |  |
| **6** | 界面元素的尺寸是否合理？行、列的间距是否保持一致？ |  |  |  |
| **7** | 设计的所有功能都已经编码？ |  |  |  |
| **8** | 变量和参数的初始化都有吗？ |  |  |  |
| **9** | 《产品需求说明书》完备吗？ |  |  |  |
| **10** | 《产品需求说明书》容易理解吗？ |  |  |  |
| **11** | 《功能说明书》内容完整，满足一致性要求。 |  |  |  |
| **12** | 《测试用例》能保证测试到所有的不可见功能。 |  |  |  |
| **13** | 《详细设计说明书》易于理解，有电子文档。 |  |  |  |
| **14** | (学生设计) |  |  |  |
| **15** | (学生设计) |  |  |  |
| **16** | (学生设计) |  |  |  |
| **17** | (学生设计) |  |  |  |
| **18** | (学生设计) |  |  |  |
| **19** | **.......** |  |  |  |
| **20** | **.......** |  |  |  |
| **21** | **.......** |  |  |  |

评审人：

评审时间：

**缺陷检查表的解释**

在完成每个评审内容之后，在右边的评审结果中打个表示检查结果的符号（√）。是、否、不确定只能三选一。一般只能选择是、否，如果该评审内容没有做，就空在那里不要填。

**1,2项** 如果否定的话，意味着用户不能有效地使用这个软件，是不可原谅的缺陷是不可原谅的缺陷是不可原谅的缺陷是不可原谅的缺陷。这个缺陷是需求分析错误造成的。

**3,4项** 如果出现如此低级的缺陷，说明开发人员根本没有把用户界面放在心上，用户很反感这种不敬业的态度。是不可原谅的缺陷。

**7项** 代码的完整性检查。

**8项** 在程序的开始；在每个循环的开始；在函数/过程的入口。变量和参数的初始化检查这三个部分。此项评审需要特别多的时间，同学们可不做。

**9,10项** 参考教材的P330。

**12,13项** 参考教材的P216-P224。

**14-18项** 要求学生至少设计五个缺陷检查内容。

**19-XX项** 多做会加分。

**二 检查坏味的代码**

**坏味代码和软件重构**

软件在其生命周期内随着时间的变化而变化。由于需求的变更，软件也必须做出相应的改变。这种改变会打破软件原先的设计构架，使得软件结构不再清晰明了，软件的维护也变得越来越困难。软件重构是在不改变软件外部特性的情况下，通过调整软件内部结构以提高软件的可理解性、可维护性和可扩展性。软件重构使得软件结构不断得到优化并保持最佳的状态，以适应外界需求的变化。同时，随着软件结构更加清晰，软件中存在的问题和缺陷更容易暴露，可以及时得到修改和更正，从而提高了软件的整体开发效率。软件重构的效果已经在工业界和学术界获得广泛的认可。

代码坏味则是指由于设计缺陷或坏的编码习惯而引入程序的、影响软件结构的程序代码。它总结了一些常见的软件问题模式，指出了代码中存在的潜在问题。通过代码坏味，能够发现和定位软件中存在的问题，明确应该在何处实施软件重构、实施何种重构。

**典型代码坏味**

(1) 不合适的命名

程序员在编程中图省事，不经推敲，简单地命名软件中的元素(包括文件名、类名、方法名、变量名等)，造成软件代码的可阅读性和可理解性差。随着时间的推移，当程序员重新审视自己的软件时，根本想不起写代码时的思路。尤其当软件缺少必要的说明，维护人员面对一系列简单字符表示的名称，无从考证编程者的思路，很难对软件进行修改和维护。这种情况下，应该选择能够表达元素实际意义的名称进行命名，使得能够见名知意，增强软件的可阅读性和可理解性。

(2) 重复代码

程序员往往独立开发系统的不同部分，难免会出现一些相同或者相似的代码。更为常见的是，程序员在编写新的代码时发现，之前有相同或相似的代码可以直接使用，就进行拷贝和粘贴。重复代码使得软件更为复杂。当软件出现问题，或者功能需要调整时，必须对多处相同或相似的代码一并进行修改，容易遗漏或造成冲突。所以，应该对重复的部分进行合并，置于单独的类或方法中，使软件更为精简。

(3) Switch 语句

在软件中，经常会遇到使用Switch 语句的情况。并非所有的Switch 语句都是代码坏味，这里指的是以类型码为匹配对象进行选择的Switch 语句。并且，同样的Switch 语句会多次出现在不同的方法或类中。当需要增加新的或者修改CASE 分支内语句时，就必须找到所有相同的Switch 语句进行修改。.随着开发规模的扩大，软件内部多处会出现重复的代码，造成系统冗余。Switch 语句可以使用多态来处理。找到与类型码有关的方法或者类，抽取相关代码到适当的类中，用子类或者状态/策略替换类型码，建立继承体系，将条件式替换为多态。

(4) 特征依恋

在软件中，某个类的方法必须通过调用其他类中大量的数据才能完成其自身的工作。即，该方法对其他类中数据的依赖远远超过其宿主类中的数据。这导致类之间的耦合度过高，增加了软件维护的难度。此类代码坏味应该通过抽取方法分离出位置不当的部分，使用移动方法和移动成员变量将软件元素置于适当的类中加以处理。

(5) 平行继承体系

在为一个类添加子类的同时，需要给另一个与其平行的类添加子类。经过多次添加之后，系统会变得复杂并难以修改。这样，在对一个类进行修改的同时必须修改与其平行的类，增加了系统的复杂性，使得系统难以维护。平行继承体系是霰弹式修改的一种特殊情况，其继承体系以一种并行的方式发展。应该使用移动方法和移动成员变量来重新分配特性，去除并行的继承体系。

(6) 霰弹式修改

在软件中，多个类之间耦合度过高。当外界发生变化时，需要同时修改软件中的多个类。此时，因为需要修改的代码过于分散，难以找出要修改的部分，导致遗漏某处重要修改，给软件留下隐患。处理的方法是利用现有类或者创造新类，通过移动方法和移动成员变量，将需要同时发生变化的代码抽取到同一个类中来处理。

(7) 发散式变化

在开发过程中，程序员往往无意识地使类承担过多的责任。当出现不同决策时，都要对同一个类进行修改以适应条件的变化，增加了类的不稳定性。即，每一次外界变化都要对类修改，并且每次修改都需要对整个类进行重新测试，以保证修改不会影响到类的其他部分。这增加了不必要的工作量,尤其当多种条件需要同时实现时，造成类内部矛盾。处理此类代码坏味，应将针对某特定变化的所有相应修改都抽取出来放在单独类中，保证每次变化需要修改的部分都在单一的类中，并且这个类内所有的方法都与这个变化相关。

(8) 长方法

软件中往往存在一个方法中有大量代码行的情况。在代码编写过程中，程序员会无意识地在方法中写入更多的代码。这增加了方法理解和修改的难度，同时也不利于重用。所以，应该通过抽取方法将原方法分解为更小的部分，并恰当地命名，使得通过名称就能理解方法实现的功能，以减少代码重复，使系统更易于理解、扩展和复用。

(9) 大 类

和长方法类似，在软件设计时，赋予一个类太多的职责，在类的内部设置了过多的成员变量或方法，增加了类的理解难度。并且，如果类中存在重复代码或者死代码，也不容易发现。处理方法是，通过对类中的成员变量进行分类和抽取，提取到新类中，或者作为该类的子类，对类进行精简。

(10) 长参数列表

在软件设计过程中，程序员往往为了减少软件模块之间的耦合关系，或者为了使模块功能更加通用化，而使用多个参数来对方法传递所需的内容。本身使用多个参数并非不可以，但是带来的主要问题是，使用这些方法需要多个输入项，众多的参数不容易记忆和区分，增加了方法的使用难度。此类代码坏味应该通过以下方法处理：如果可以由对象得到参数则利用方法替换参数，如果可以将参数分类则通过生成参数对象来处理。

**坏味代码检查表**

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **坏味类型** | **代码位置** | **说明** |
| **1** |  |  |  |
| **2** |  |  |  |
| **3** |  |  |  |
| **4** |  |  |  |

**注释：**坏味类型从10个典型的代码坏味类型中选择，代码位置标注文件名，代码所在的行数。“说明”则写出同学们的认定代码坏味的依据。

**参考文献**

[1]百度文库．检查表编写指南[EB/OL]．http://wenku.baidu.com/view/

44009727af45b307e871976a.html，2012.10.20．

[2]百度文库．用户界面检查表[EB/OL]． http://wenku.baidu.com/view/

25dcf681e53a580216fcfec7.html，2012.10.20．

[3]CSDN．[个体软件过程]之缺陷管理--C++代码复查指南和检查表[EB/OL]．

http://blog.csdn.net/treewith/article/details/18469，2012.10.20．

[4]朱少民主编. 软件质量保证和管理[M].北京：清华大学出版社，2007年.

[5]高原,刘辉,樊孝忠,牛振东,邵维忠.代码坏味的处理顺序[J].软件学报,2012,23(8):1965-1977．

# 实验：编码规则与代码优化

实验名称：编码规则与代码优化

一、实验目的：

通过FindBugs学习静态分析工具的使用。

1. 实验原理：

软件设计的优化；编程规则，代码审查，提高程序质量的技术。

1. 实验环境

FindBugs

1. 实验内容与步骤

安装FindBugs，并按照实验指导书的要求验证它们的基本功能。针对一款开源软件，检查它的编码规范，并优化软件的部分模块。实验详细内容参照“实验指导书”。

(写出实验步骤)

五、实验结果与分析。

**FindBugs**

**FindBugs 是什么？**

FindBugs 是一个静态分析工具，它检查类或者 JAR 文件，将字节码与一组缺陷模式进行对比以发现可能的问题。有了静态分析工具，就可以在不实际运行程序的情况对软件进行分析。不是通过分析类文件的形式或结构来确定程序的意图，而是通常使用 Visitor 模式。在FindBugs的GUI中，需要先选择待扫描的.class文件(FindBugs其实就是对编译后的class进行扫描，藉以发现一些隐藏的bug。)。如果你拥有这些.class档对应的源文件，可把这些.java文件再选上，这样便可以从稍后得出的报告中快捷的定位到出问题的代码上面。此外，还可以选上工程所使用的library，这样似乎可以帮助FindBugs做一些高阶的检查，藉以发现一些更深层的bug。

选定了以上各项后，便可以开始检测了。检测的过程可能会花好几分钟，具体视工程的规模而定。检测完毕可生成一份详细的报告，藉由这份报告，可以发现许多代码中间潜在的bug。比较典型的，如引用了空指针(null pointer dereference), 特定的资源(db connection)未关闭，等等。如果用人工检查的方式，这些bug可能很难才会被发现，或许永远也无法发现，直到运行时发作…当除掉了这些典型的bug后，可以确信的是，我们的系统稳定度将会上一个新的台阶。

以目前遇到的状况来看，FindBugs可以有两种使用时机。

开发阶段 　　当Developer完成了某一部分功能模块开发的时候(这通常是指代码撰写完成，并已debug通过之后)，可藉由FindBugs对该模块涉及的java文件进行一次扫描，以发现一些不易察觉的bug或是效能问题。交付新版的时候，开发团队可以跑一下FindBugs，除掉一些隐藏的Bug。FindBugs得出的报告可以作为该版本的一个参考文档一并交付给测试团队留档待查。在开发阶段使用FindBugs，一方面开发人员可以对新版的品质更有信心，另一方面，测试人员藉此可以把更多的精力放在业务逻辑的确认上面，而不是花大量精力去进一些要在特殊状况下才可能出现的BUG(典型的如空指针)。从而可以提高测试的效率。

维护阶段 　　这里指的是系统已经上线，却发现因为代码中的某一个bug导致系统崩溃。在除掉这个已暴露的bug之后，为了快速的找出类似的但还未暴露的 bug，可以使用FindBugs对该版的代码进行扫描。当然，在维护阶段使用FindBugs往往是无奈之举，且时间紧迫。此外，如果本来在新版交付的时候就使用过FindBugs的话，往往意味着这种bug是FindBugs还无法检测出的。这也是FindBugs局限的地方。

FindBugs出到目前的版本，功能已经相当强大，不过也有待完善的地方。从实际使用来看，有一些隐藏的bug并不能靠FindBugs直接发现。FindBugs也不能发现非java的Bug。对于非java撰写的代码，如javascript，SQL等等，要找出其中可能的bug，FindBugs是无能为力的。当然，javascript中的bug似乎还不至于使系统崩溃，而SQL中的bug往往又跟业务逻辑相关，只要测试仔细一些应该是可以发现的。

FindBugs不过是一个工具。作为开发人员，当然首先要在编程的时候努力避免引入bug，而不要依赖于某个工具来为自己把关。不过由于代码的复杂性，一些隐藏的bug确实很难靠咱们的肉眼发现。这时，应用一些好的工具或许就可以帮你发现这样的bug。这便是FingBug存在的价值。

**FindBugs 可以做什么？**

FindBugs 提供了35 个检测器来检测字节码中可能的缺陷。可以做的事情主要有：

**（1）找出 hash equals 不匹配** 找与 equals() 和 hashCode() 的实现相关的几个问题。这两个方法非常重要，因为几乎所有基于集合的类---List 、Map 、Set 等都调用它们。一般来说，这个检测器寻找两种不同类型的问题: ①当一个类重写对象的 equals() 方法，但是没有重写它的 hashCode 方法，或者相反的情况时。 ②定义一个 co-variant 版本的 equals() 或 compareTo() 方法。例如， Bob 类定义其 equals() 方法为布尔 equals(Bob) ，它覆盖了对象中定义的 equals() 方法。因为 Java 代码在编译时解析重载方法的方式，在运行时使用的几乎总是在对象中定义的这个版本的方法，而不是在 Bob 中定义的那一个（除非显式将 equals() 方法的参数强制转换为 Bob 类型）。因此，当这个类的一个实例放入到类集合中的任何一个中时，使用的是 Object.equals() 版本的方法，而不是在 Bob 中定义的版本。在这种情况下， Bob 类应当定义一个接受类型为 Object 的参数的 equals() 方法。

关于equals() 方法 和 hashCode()方法，这两个方法一般是连体儿。某个类继承自Object类，假如重写这两个方法的其中一个时，“ 必须 ”重写另一个方法 （一般情况下是重写 equals() 方法，此时就要注意重写 hashCode()方法了 ） 。

**（2）检测：忽略方法返回值**  这个检测器查找代码中忽略了不应该忽略的方法返回值的地方。这种情况的一个常见例子是在调用 String 方法时，例如: **Java代码**

1. String aString = "bob";
2. aString.replace('b', 'p');
3. if(aString.equals("pop"))

这个错误很常见。在第 2 行，程序员认为他已经用 p 替换了字符串中的所有 b 。确实是这样，但是他忘记了字符串是不可变的。所有这类方法都返回一个新字符串，而从来不会改变消息的接收者。

注意：执行到第三句的时候，此时有两个字符串，一个是aString，此时的值还是“bob”；还有一个，就是replace方法返回的新字符串“pop”。

**（3）检测：Null 指针对 null 的解引用（dereference ）和冗余比较**

这个检测器查找两类问题。它查找代码路径将会或者可能造成 null 指针异常的情况，它还查找对 null 的冗余比较的情况。例如，如果两个比较值都为 null ，那么它们就是冗余的并可能表明代码错误。FindBugs 在可以确定一个值为 null 而另一个值不为 null 时，检测类似的错误，例如：

**Java代码**

1. Person person = aMap.get("bob");
2. if (person != null) {
3. person.updateAccessTime();
4. }
5. String name = person.getName();

在这个例子中，如果第 1 行的 Map 不包括一个名为“bob ”的人，那么在第 5 行询问 person 的名字时就会出现 null 指针异常。因为 FindBugs 不知道 map 是否包含“bob ”，所以它将第 5 行标记为可能 null 指针异常。 空指针异常，绝大多数情况都是null.\*\*\*，想对为null的对象进行某种操作，必然会导致空指针异常。这种异常应该是Java编码中出现最频繁的异常了，应该引起足够的重视（在代码中多进行非空check）。

**（4）检测：初始化之前读取字段** 这个检测器寻找在构造函数中初始化之前被读取的字段。这个错误通常是由使用字段名而不是构造函数参数引起的，例如在构造函数中读取未初始化的字段:

**Java代码**

1. public class Thing {

2. private List actions;

3. public Thing(String startingActions) {

4. StringTokenizer tokenizer = new StringTokenizer(startingActions);

1. while (tokenizer.hasMoreTokens()) {
2. actions.add(tokenizer.nextToken());
3. }
4. }

9. }

在这个例子中，第 6 行将产生一个 null 指针异常，因为变量 actions 还没有初始化。这个问题，新版本的Eclipse应该不存在吧，出现这个情况会自动报错的，编译通不过，此时不需要劳驾FindBugs出马。

**（5）命名检查** 对标准 Java 命令规范的测试：变量名称不应太短；方法名称不应过长；类名称应当以小写字母开头；方法和字段名应当以小写字母开头，等等。

**（6）未使用的代码检查** 查找从未使用的私有字段和本地变量、执行不到的语句、从未调用的私有方法，等等。对多余代码的检查。私有字段或者私有方法，别人不能直接使用，假如在本类中都没有使用到，那还放在这边干嘛呢。

**（7）嵌套检查** 例如： switch 语句应当有 default 块，应当避免深度嵌套的 if 块，不应当给参数重新赋值，不应该对 double 值进行相等比较。 1.switch 语句的各个case中，应该不能没有break语句，否则会发生case穿透现象； 2.至于default块，肯定也是要有的，这个是兜底保本的。 3.应当避免深度嵌套的 if 块：深度太深，程序的逻辑结构就看得不太清楚了，此时可以考虑重新组织逻辑，或者拆分成多个方法。 4.不应当给参数重新赋值：这个当然了，你既然使用了接受参数，在方法内又把人家传过来的值给重新赋值了，这叫什么事呀。 5.不应该对 double 值进行相等比较：嗯，这个是要注意，double不像整数，它是不精确的（小数部分是由0101模拟的），所以无法直接对两个double数进行等值比较，一般都是比较两者的差是否小于某个值（比如0.0001，就相当于是精确到小数点后面第4位），若成立，则可以认为两者是”相等“的。

**（8）导入语句检查** 检查 import 语句的问题，比如同一个类被导入两次或者被导入 java.lang 的类中。

**（9）JUnit 测试检查**  查找测试用例和测试方法的特定问题，例如方法名称的正确拼写，以及 suite() 方法是不是 static 和 public 。

**（10）字符串检查** 找出处理字符串时遇到的常见问题，例如重复的字符串标量，调用 String 构造函数，对 String 变量调用 toString() 方法。

**（11）括号检查** 检查 for 、 if 、 while 和 else 语句是否使用了括号。 这个问题也是要注意一下，开始时只有一句语句时，从执行效果上来讲，加不加这个括号都是没有问题的。 但是不加括号是个很不好的习惯，有隐藏的问题：将来追加代码，直接加在原先一句的后面，目的是想将代码加在语句块内，但是因为前面没有加括号，假如此时也忘记加括号，这样的话，代码的逻辑显然就错了。 所以啊，哪怕此时代码段中的语句只有一句，也要加上括号，避免隐患。

**（12）代码尺寸检查** 测试过长的方法、有太多方法的类以及重构方面的类似问题。

**（13）终结函数检查** 因为在 Java 语言中， finalize() 方法不是那么普遍，它们的使用规则虽然很详细，但是人们对它们相对不是很熟悉。这类检查查找finalize() 方法的各种问题，例如空的终结函数，调用其他方法的 finalize() 方法，对 finalize() 的显式调用，等等。

**（14）克隆检查**  用于 clone() 方法的新规则。凡是重写 clone() 方法的类都必须实现 Cloneable ， clone() 方法应该调用 super.clone() ，而 clone() 方法应该声明抛出 CloneNotSupportedException 异常，即使实际上没有抛出异常，也要如此。

**（15）异常检查** 针对异常的检查：不应该声明该方法而抛出 java.lang.Exception 异常，不应当将异常用于流控制，不应该捕获 Throwable ，等等。

**（16)日志检查** 查找 java.util.logging.Logger 的不当使用，包括非终状态（nonfinal ）、非静态的记录器，以及在一个类中有多个记录器。

**(17)Open—Close 检查** 检查文件或通讯方面，是否忘记Close 的情况。

**（18）其它检查**

**（19）构建自己的规则集** 可以构建自己的规则集。

**参考文献**

[1]百度百科．FindBugs [EB/OL]．http://baike.baidu.com/view/2367937.htm，

2012.10.20．

[2]百度文库．findbugs介绍[EB/OL]． http://wenku.baidu.com/view/

ad628c5d3b3567ec102d8a2c.html，2012.10.20．

[3] www.sourceforge.net．FindBugs downloads[EB/OL]．

http://findbugs.sourceforge.net/downloads. html，2012.10.20．

[4]朱少民主编. 软件质量保证和管理[M].北京：清华大学出版社，2007年.

# 实验：程序的调试(Eclipse or MyEclipse)

1. 实验名称：程序的调试
2. 实验目的及要求：

**实验目的** 掌握Eclipse或者MyEclipse的调试方法，如断点、单步执行、观察变量值等。学习基本的调试技巧。

**实验要求** 独立完成给定程序的调试，修改代码使得程序正确执行。

1. 实验环境

MyEclipse 或者 Eclipse

1. 实验内容

给定一个Java程序， 能独立定位程序的错误代码，并且能修改代码。

1. 实验步骤：

(写出实验步骤)

六、调试过程及实验结果

与实验步骤合并。

工程名：DebugSample

ValueProcess.java

//类的定义 作者： 何海江

public class ValueProcess {

private int nValue[]; //存放数据的数组。

private int nTotal;// 数组内数据个数

public ValueProcess()

{

nValue = null ;

nTotal = 0;

}

//往数组内添加一个数

public void AddValue(int v)

{

int nIndex = nTotal-1;

nValue[nIndex] = v;

nTotal ++;

}

//获取数组内数据的最大值

public int GetMax()

{

int max = nValue[0];

for( int i=1;i<nTotal;i++ )

if( max<nValue[i] )

max = nValue[i];

return max;

}

(接上页)

//获取数组内数据的最小值

public int GetMin()

{

int min = nValue[0];

for( int i=1;i<nTotal;i++ )

if( min>nValue[i] )

min = nValue[i];

return min;

}

//从小到大排序

public String AfterSort()

{

for( int i=0;i<nTotal;i++ )

{

int first = nValue[i];

for( int j=i+1;j<nTotal; j++ )

if( first>nValue[j])

{

first = nValue[j];

nValue[j] = first;

}

}

StringBuffer buffer = new StringBuffer();

for( int i=0;i<nTotal;i++ )

{

buffer.append(nValue[i]);

buffer.append(",");

}

return buffer.toString();

}

}

DebugMain.java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

//类的定义 作者： 何海江

public class DebugMain {

public static void main(String[] args) {

// TODO Auto-generated method stub

ValueProcess vp = new ValueProcess();

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

try {

while(true)

{

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*数字小系统\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("a：添加一个数");

System.out.println("b：找出已有数列的最大值最小值");

System.out.println("c：对已有数列排序");

System.out.println("x：退出系统");

String operType=br.readLine();

(接上页)

// 旅客管理

if(operType.equals("a"))

{

System.*out*.println("请输入数字：");

BufferedReader bVT=new BufferedReader(new InputStreamReader(System.*in*));

int num = bVT.read();

vp.AddValue(num);

}

else if(operType.equals("b"))

{

int nMax = vp.GetMax();

int nMin = vp.GetMin();

System.*out*.println("最大值:"+nMax+" 最小值:"+nMin);

}

else if(operType.equals("c"))

{

String strInfo = vp.AfterSort();

System.*out*.println(strInfo);

}

else

{

System.*out*.println("再见！");

break;

}

}//end of while

}

catch(IOException e)

{

System.*err*.println(e.toString());

}

}

}

下面的内容先不看，老师讲解后再阅读。

步骤1，输入24，选取菜单a，执行。

观察执行结果，程序出错，抛出异常。

使用定位技术，在AddValue里设置断点。

int nIndex = nTotal-1 ---> int nIndex = nTotal

步骤2，输入24，选取菜单a，执行。

观察执行结果，程序继续出错，抛出异常。

在AddValue里设置断点。再运行Debug ，观察num。

可观察到nValue没有申请空间。

在构造函数加入 nValue = new int [100];

步骤3，再执行

输入25会变成50 ，36会变成51

定位错误，int num = bVT.read()

修改成string str = bVT.readLine()

int num = Integer.ParseInt(str)

步骤4，输入5 20 4 30 6

排序后出错。

AfterSort有错误。

步骤5，告知：实际还隐藏着一个错误。不经过严格测试是找不出来的，当数组里未加入任何数据时，仍然输出最大值0，最小值0.实际没有数据。

这也说明，要设计好的测试用例。比如排序也是如此，如果输入4 5 6 20 30这样的测试用例，很可能测不出错误。

步骤6，当调试程序时，频繁输入数据，效率比较低。

解决方法：在if(operType.equals("a")) 里直接使用vp.AddValue(num);输入数据，

vp.AddValue(4);

vp.AddValue(15);

vp.AddValue(20);

vp.AddValue(6);

**调试综合实验**

拷贝工程FlightDebugSample

FlightDebugSample调试程序。里面人工植入了几处错误代码，让学生通过前面学习到的调试技能找出错误代码。

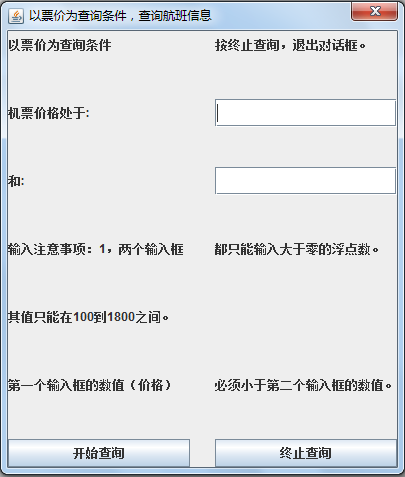
DebugSample是一个简单的程序，学习基本的调试技能。

FlightDebugSample的代码相对复杂许多。

1，在主菜单选择“文件”—〉“增加一个航班”时，程序出错。输入航空公司名称时，要求以H开头，后接5-19个任意字符、程序会验证航空公司名称是否输入正确，如果达不到前述要求，软件提示“录入数据无法通过验证，请重新录入，或取消”。实际情况是，当航班号、航班名称、机票价格、座位数目输入正确，而航空公司名称输入错误的数据，如”5H4657”时，程序并不报错。查找程序中对应的错误代码，并调试，正确完成此项功能。

2，在主菜单选择“文件”—〉“从文件读入数据”时，程序出错。文件FlightInfo.txt中实际有三条航班记录，可只能读到两条。查找程序中对应的错误代码，并调试，正确完成此项功能。

3，在主菜单选择“查询统计”—〉“以票价为条件检索”，出现如下所示的界面。



软件此部分的需求为：

在两个文本框输入机票价格，有效输入要求为数值处于100-1800的浮点数，并且第一个价格小于第二个价格。点击“终止查询”或者关闭对话框按钮时系统不判断输入是否有效，随后进入主界面。而点击“开始查询”时系统判断输入是否有效，如果无效，给出警示对话框，提示“录入数据无法通过验证”；如果有效，进入查询阶段。在查询阶段，如果当前航班记录中存在此价格区间的航班，全部显示出来，没有的话，提示“没有找到符合条件的航班”。

此部分代码出错。当输入最小价格300，最大价格1800时，文件中存在符合条件的航班，可是却提示“没有找到符合条件的航班”。查找程序中对应的错误代码，并调试，正确完成此项功能。

4，在主菜单选择“文件”—〉“删除航空公司的所有航班”时，程序出错。文件FlightInfo.txt中实际有多条某航空公司的航班记录，可只能删除一条。查找程序中对应的错误代码，并调试，正确完成此项功能。

# 实验：面向对象测试技术

1. 实验名称：面向对象测试技术
2. 实验目的及要求：

**实验目的** 掌握逻辑覆盖、场景测试等测试技术，理解类（面向对象）的单元测试、集成测试策略。

**实验要求** 独立设计测试用例，完成测试报告。

1. 实验环境

MyEclipse or Eclipse

1. 实验内容

给定一个类，设计测试用例，进行该类的随机测试和类层次的分割测试。详细内容参照“实验指导书实验一”。

1. 实验步骤：

(测试用例按照如下格式填写测试用例即可:)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 测试名称 | 描述 | 输入 | 预期输出 | 实际输出 |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| ... |  |  |  |  |  |

(自测试功能的代码则工整地写出来。)

六、调试过程及实验结果

与实验步骤合并。

**挡风玻璃雨刷系统 ：参见实验指导书的相关内容。**

实现上述系统的代码**：参见实验指导书的相关内容**。

具体的实验要求：

1，以类的方法为单元，单元测试，应用分支覆盖技术。

操纵杆“放下”事件的测试用例。

2，以类为单元，单元测试例子，应用场景测试技术。

设计一个场景，测试操纵杆类。

3，集成测试，应用场景测试技术。

设计一个场景，测试挡风玻璃雨刷系统。

挡风玻璃雨刷的一个测试用例场景：

挡风玻璃雨刷操纵杆在“停止”位置，刻度盘在位置1；加电，用户将操纵杆推到“间歇”，然后将刻度盘从位置1主导位置2，再转到位置3；然后将操纵杆推到“低速”；用户将操纵杆推到“间歇”；然后推到“停止”。

4，自测试功能

编写代码，完成上述三个要求的自测试功能。

# 测试计划模板

**简介**

**目的**

该《测试计划》文档有助于实现以下目标：确定现有项目的信息和应测试的软件构件；列出推荐的测试需求（高级需求）；推荐可采用的测试策略，并对这些策略加以说明；确定所需的资源，并对测试的工作量进行估计；列出测试项目的可交付元素。

**背景**

对测试对象（构件、应用程序、系统等）及其目标进行简要说明，包括的信息有：主要的功能和性能、测试对象的构架以及项目的简史。

**范围**

描述测试的各个阶段(例如，单元测试、集成测试或系统测试)，并且说明本计划所针对的测试类型（如功能测试或性能测试）。简要地列出测试对象中将接受测试或将不接受测试的那些性能和功能。如果在编写此文档的过程中做出的某些假设可能会影响测试设计、开发或实施，则列出所有这些假设。列出可能会影响测试设计、开发、或实施的所有风险或意外事件。列出可能会影响测试设计、开发或实施的所有约束。

**测试参考文档和测试提交文档**

**测试参考文档**

下表列出了制定测试计划时所使用的文档，并标明了各文档的可用性。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文档（版本／日期 | 已创建或可用 | 已被接受或已经过复审 | 来源 | 备注 |
| 可行性分析报告 | 是□　否□ | 是□　否□ |  |  |
| 软件需求定义 | 是□　否□ | 是□　否□ |  |  |
| 软件系统分析 | 是□　否□ | 是□　否□ |  |  |
| 软件概要设计 | 是□　否□ | 是□　否□ |  |  |
| 软件详细设计 | 是□　否□ | 是□　否□ |  |  |
| 软件测试需求 | 是□　否□ | 是□　否□ |  |  |
| 模块开发手册 | 是□　否□ | 是□　否□ |  |  |
| 用户操作手册 | 是□　否□ | 是□　否□ |  |  |

**将要提交的文档**

下表列出来后面的将要用到的文档，测试用例根据项目进度逐步完成。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文档（版本／日期 | 已创建或可用 | 已被接受或已经过复审 | 来源 | 备注 |
| 测试时间表／人员安排 | 是□　否□ | 是□　否□ |  |  |
| 测试总体计划 | 是□　否□ | 是□　否□ |  |  |
| 测试方案 | 是□　否□ | 是□　否□ |  |  |
| 测试分析报告 | 是□　否□ | 是□　否□ |  |  |
| 测试用例模板 | 是□　否□ | 是□　否□ |  |  |
| 测试报告单模板 | 是□　否□ | 是□　否□ |  |  |
| 测试用例通过情况统计表 | 是□　否□ | 是□　否□ |  |  |
| 各模块的测试用例 | 是□　否□ | 是□　否□ |  |  |

**测试提交文档**

下面应当列出在测试阶段结束后，所有可提交的文档。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文档（版本／日期 | 已创建或可用 | 已被接受或已经过复审 | 来源 | 备注 |
| 测试报告 | 是□　否□ | 是□　否□ |  |  |
| 测试记录 | 是□　否□ | 是□　否□ |  |  |

**测试进度**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试活动 | 计划开始日期 | 实际开始日期 | 结束日期 |
| 制定测试计划 |  |  |  |
| 设计测试 |  |  |  |
| 集成测试 |  |  |  |
| 系统测试 |  |  |  |
| 性能测试 |  |  |  |
| 安装测试 |  |  |  |
| 用户验收测试 |  |  |  |
| 对测试进行评估 |  |  |  |
| 产品发布 |  |  |  |

**测试资源**

**人力资源**

下表列出了在此项目的人员配备方面所在的各种假定。

|  |  |  |
| --- | --- | --- |
| 角色 | 所推荐的最少资源（所分配的专职角色数量） | 具体职责或注释 |
|  |  |  |
|  |  |  |

注：可适当的删除或添加角色项。

**测试环境**

软件环境（相关软件、操作系统等）

硬件环境（网络、设备等）

**测试工具**

此项目列出测试使用的工具。

|  |  |  |  |
| --- | --- | --- | --- |
| 用途 | 工具 | 生产厂商/自产 | 版本 |
|  |  |  |  |

**系统风险、优先级**

简要描述测试阶段的风险和处理优先级

**测试策略**

测试策略提供了对测试对象进行测试的推荐方法。对于每种测试，都应提供测试说明，并解释其实施原因。制定测试策略时所考虑的主要事项有：将要使用的技术以及判断测试何时完成的标准。下面列出了在进行每项测试时需考虑的事项。注意：不实施某种测试，则应该用一句话加以说明，并陈述这样的理由。例如：“将不实施该测试。该测试本项目不适用。”

**数据和数据库完整性测试**

在“项目名称”中，数据库和数据库进程应作为一个子系统来进行测试。在测试这些子系统时，不应将测试对象的用户界面用做数据的接口。对于数据库管理系统（ＤＢＭＳ），还需要进行深入的研究，以确定可以支持以下测试的工具和技术。

* 测试目标：确保数据库访问方法和进程正常运行，数据不会遭到损坏。
* 测试范围：
* 技术：调用各个数据库访问方法和进程，并在其中填充有效的和无效的数据（或对数据的要求）。检查数据库，确保数据已按预期的方式填充，并且所有的数据库事件已正常发生；或者检查所返回的数据，确保正当的理由检索到了正确的数据
* 开始标准：
* 完成标准：所有的数据库访问方法和进程都按照设计的方式运行，数据没有遭到损坏。
* 测试重点和优先级：
* 需考虑的特殊事项：测试可能需要ＤＢＭＳ开发环境或驱动程序在数据库中直接输入或修改数据。进程应该以手工方式调用。应使用小型或最小的数据库（记录的数量有限）来使所有无法接受的事件具有更大的可视度。

**接口测试**

* 测试目标：确保接口调用的正确性。
* 测试范围：所有软件（硬件）接口，记录输入输出数据。
* 技术：
* 开始标准：
* 完成标准：
* 测试重点和优先级：
* 需考虑的特殊事项：接口的限制条件。

**集成测试**

集成测试主要目的是检测系统是否达到需求对业务流程及数据流的处理是否符合标准，检测系统对业务流程处理是否存在逻辑不严谨及错误，检测需求是否存在不合理的标准及要求。此阶段测试基于功能完成的测试。

* 测试目标：检测需求中业务流程，数据流的正确性。
* 测试范围：需求中明确的业务流程，或组合不同功能模块而形成一个大的功能。
* 技术：利用有效的和无效的数据来执行各个用例、用例流或功能，已核实在使用有效数据时得到预期的结果，在使用无效数据时显示相应的错误消息或警告信息，各业务规则都得到了正确的应用。
* 开始标准：在完成某个集成测试时必须达到的标准。
* 完成标准：所计划的测试已全部执行。所发现的缺陷已全部解决。
* 测试重点和优先级：测试重点指在测试过程中需要着重测试的地方，优先级可以根据需求及严重程度来定。
* 需要考虑的特殊事项：确定或说明那些将对功能测试的实施和执行造成影响的事项和因素（内部的或外部的）。

**功能测试**

对测试对象的功能测试应侧重于所有可直接追踪到用例或业务功能和业务规则的测试需求。这种测试的目标是核实数据的接受、处理和检索是否正确，以及业务规则的实施是否恰当。此类测试基于黑盒技术，该技术通过图形用户界面（GUI）与应用程序进行交互，并对交互的输出或结果进行分析，以此来核实应用程序及其内部进程。以下为各种应用程序列出了推荐使用的测试概要。

* 测试目标：确保测试的功能正常，其中包括导航、数据输入、处理和检索等功能。
* 测试范围：
* 技术：利用有效的和无效的数据来执行各个用例、用例流或功能，以核实在使用有效数据时得到预期的结果，在使用无效数据时显示相应的错误信息或警告信息，各业务规则都得到了正确的应用。
* 开始标准：
* 完成标准：
* 测试重点和优先级：
* 需要考虑的特殊事项：确定或说明那些将对功能测试的实施和执行造成影响的事项或因素（内部的或外部的）。

**用户界面测试**

用户界面（UI）测试用于核实用户与软件之间的交互。UI测试的目标是确保用户界面通过测试对象的功能来为用户提供相应的访问或浏览功能。另外，UI测试还可确保UI中的对象按照预期的方式运行，并符合公司或行业的标准。

* 测试目标：核实通过测试进行的浏览可正确的反映业务的功能和需求，这种浏览包括窗口与窗口之间、字段与字段之间的浏览，以及各种访问方法（TAB键、鼠标移动、和快捷键）的使用窗口的对象和特征（例如，菜单、大小、位置、状态和中心）都符合标准。
* 测试范围：
* 技术：为每个窗口创建或修改测试，以核实各个应用程序窗口和对象都可正确的进行浏览，并处于正常的对象状态。
* 开始标准：
* 完成标准：
* 测试重点和优先级：
* 需要考虑的特殊事项：并不是所有定制或第三方对象的特征都可以访问。

**性能评测**

性能评测是一种性能测试，它对响应时间、事务处理速率和其他与时间相关的需求进行评测和评估。性能评测的目标是核实性能需求是否都以满足。实施和执行性能评测的目的是将测试对象的性能行为当作条件（例如工作量或硬件配置）的一种函数来进行评测和微调。（注：以下提到的事务都是指逻辑业务事务）

* 测试目标：核实所指定的事务或业务功能在以下情况下的性能行为，正常的预期工作量预期的最繁重的工作量。
* 测试范围：
* 技术：使用为功能或业务周期测试制定的测试过程。通过修改数据文件来增加事务数量，或通过修改脚本来增加每项事物的迭代数量。脚本应该在一台计算机上运行（最好是以单个用户、单个事务为基准），并在多个客户机（虚拟的或实际的客户机，请参见下面的“需要考虑的特殊事项”）上重复。
* 开始标准：
* 完成标准：单个事务或单个用户时，每个事务所预期时间范围内成功地完成测试脚本，没有发生任何故障。多个事务或多个用户时，在可接受的时间范围内成功地完成测试脚本，没有发生任何故障。
* 测试重点和优先级：
* 需要考虑的特殊事项：综合的性能测试还包括在服务器上添加后台的工作量。可采用多种方法来执行此操作，其中包括：直接将“事务强行分配到”服务器上，这通常以“结构化语言”（SQL）调用的形式来实现。通过创建“虚拟的”用户负载来模拟多个（通常为数百个）客户机。此负载可通过“远程终端仿真”（REMOTE TERMINAL EMULATION）工具来实现。此技术还可用于在网络中加载“流量”。使用多台实际客户机（每台客户机都运行测试脚本）在系统上添加负载。性能测试应该在专用的计算机上或在专用的机时内执行，以便实现完全的控制和精确的评测。性能测试所用的数据库应该是实际大小或相同缩放比例的数据库。

**负载测试**

负载测试是一种性能测试。在这种测试中，将使测试对象承担不同的工作量，以评测和评估测试对象在不同工作量条件下的性能行为，以及持续正常运行的能力。负载测试的目标是确定并确保系统在超出最大预期工作量的情况下仍能正常运行。此外，负载测试还要评估性能特征，例如，响应时间、事务处理速率和其他与时间相关的方面。（注：以下提到的事务是指逻辑业务事务）

* 测试目标：核实所指定的事务或商业理由在不同的工作量条件下的性能行为时间。
* 测试范围：
* 技术：使用为功能或业务周期测试制定的测试。通过修改数据文件来增加事务数量，或通过修改脚本来增加每项事务发生的次数。
* 开始标准：
* 完成标准：多个事务或多个用户；在可接受的时间范围内成功地完成测试，没有发生任何故障。
* 测试重点和优先级：
* 需要考虑的特殊事项：负载测试应该在专用的计算机上或专用的机时内执行，一边实现完全的控制和精确的评测。负载测试所用的数据库应该是实际大小或相同缩放比例的数据库。

**强度测试**

强度测试是一种性能测试，实施和执行此类测试的目的是找出因资源不足或资源争用而导致的错误。如果内存或磁盘空间不足，测试对象就可能会表现出一些在正常条件下并不明显的缺陷。而其他缺陷则可能由于争用共享资源（如数据库锁或网络带宽）而造成的。强度测试还可用于确定测试对象能够处理的最大工作量。（注：以下提到的事务都是指逻辑业务事务）

* 测试目标：核实测试对象能够在以下强度条件下正常运行，不会出现任何错误：服务器上几乎没有或根本没有可用的内存（RAM和DASD）连接后模拟了最大实际（实际允许）数量的客户机多个用户对相同的数据或帐户执行相同的事务最繁重的事务量或最差的事务组合（请参见上面的（“性能测试”）。（注：强度测试的目标可表述为确定和记录那些使系统无法继续正常运行的情况或条件。客户机的强度测试在“配置测试”中进行的说明）
* 测试范围：
* 技术：使用为性能测试或负载测试制定的测试。要对有限的资源进行测试，就应该在一台计算机上运行测试，而且应该减少或限制服务器上的RAM和DASD。对其他强度测试，应该使用多台客户机来运行相同的测试或互补的测试，以产生最繁重的事务量或最差的事务组合。
* 开始标准：
* 完成标准：所计划的测试已全部执行，并且在达到或超出指定的系统限制时没有出现任何软件故障，或者导致系统出现故障条件的并不在指定的条件范围之内。
* 测试重点和优先级：
* 需考虑的特殊事项：如果要增加网络工作强度，可能会需要使用网络工具来给网络加载消息或信息包。应该暂时减少用于系统的DASD，以限制数据库可用空间的增长。使多个客户机对相同的记录或数据帐户同时进行访问达到同步。

**容量测试**

容量测试使测试对象处理大量的数据，以确定是否达到了将使软件发生故障的权限。容量测试还将确定测试对象在给定时间内能够持续处理的最大负载或工作量。例如，如果测试对象正在为生成一份报表而处理一组数据库记录，那么容量测试就会使用一个大型的测试数据库。检验该软件是否正常运行并生成了正确的报表。

* 测试目标：核实测试对象在以下高容量条件下能否正常运行；连接或模拟量最大（实际或实际允许）数量的客户机，所有客户机在长时间内执行相同的、并且情况（性能）最坏的业务功能。已达到最大的数据库大小（实际的或按比例缩放的），而且同时执行多个查询或报表事务。
* 测试范围：
* 技术：使用为性能评测或负载测试指定的测试。应该使用多台客户机来运行相同的测试或互补的测试,以便在长时间内产生最繁重的事务量或最差的事务所组合（请参见上面的“强度测试”），创建最大的数据库大小（实际的、按比例缩放的、或填充了代表数据的数据库），并使用多台客户机在长时间内同时运行查询和报表事务。
* 开始标准：
* 完成标准：所计划的测试已全部执行，而且达到或超出指定的系统限制时没有出现任何软件故障。
* 测试重点和优先级：
* 需要考虑的特殊事项：对于上述高容量条件，哪个时间段是可以接受的时间。

**安全性和访问控制测试**

安全性和访问控制测试侧重于安全性的两个关键方面：应用程序级别的安全性，包括对数据或业务功能的访问。系统级别的安全性，包括系统的登陆或远程访问。应用程序级别的安全性可确保：在预期的安全性情况下，ＡＣＴＯＲ只能访问特定的功能或用例，或者只能访问有限的数据。例如，可能会允许所有人输入数据、创建新账户，但只有管理员才能删除这些数据或账户。具有数据级别的安全性测试就可确保“用户类型一”能够看到所有客户消息（包括财务数据），而“用户二”看见同一客户的统计数据。系统级别的安全性可确保只有具备系统访问权限的用户才能访问应用程序，而且只能通过相应的网关来访问。

* 测试目标：应用程序级别的安全性：核实ACTOR只能访问其所属用户类型已被授权访问的那些功能或数据。系统级别的安全性：核实只有具备系统和应用程序访问权限的ACTOR才能访问系统和应用程序。
* 测试范围：
* 技术：应用程序级别的安全性：确定并列出各用户类型及其被授权访问的功能或数据。为各用户类型创建测试，并通过创建各用户类型所特有的事务来核实其权限。修改用户类型并为相同的用户重新运行测试。对于每种用户类型，确保正确地提供或拒绝了这些附加的功能或数据。关于系统级别的访问请参见以下的“需要考虑的特殊事项”。
* 开始标准：
* 完成标准：各种已知的ACTOR类型都可访问相应的功能或数据，而且所有事务都按照预期的方式运行，并在先前的应用程序功能测试中运行所有的事务。
* 测试重点和优先级：
* 需要考虑的特殊事项：必须与相应的网络或系统管理员一直对系统的访问权限进行检查和讨论。由于此测试可能是网络管理或系统管理的职能，可能会不需要执行此测试。

**故障转移和恢复测试**

故障转移和恢复测试可确保测试对象能成功完成转移，并能从导致意外数据损失或数据完整性破环的各种硬件、软件、网络故障中恢复数据。故障转移测试可确保：对于必须持续运行的系统，一旦发生故障，备用系统就将不失时机地“顶替”发生故障的系统，以避免丢失任何数据或事务。恢复测试是一种对抗性的测试过程。在这种测试中，将把应用程序或系统至于极端的条件下（或者是模拟的极端条件下），以产生故障（例如设备输入/输出（I/O）故障或无效的数据库指针和关键字）。然后调用恢复进程并检测和检查应用程序和系统，核实应用程序或系统和数据已得到了正确的恢复。

* 测试目标：确保恢复进程（手工或自动）将数据库、应用程序和系统正确地恢复到预期的已知状态。测试中将包括以下各种情况：客户机断电服务器断电，通过网络服务器产生的通信终端DASD和/或DASD控制器被中断、断电或与DASD和/或DASD控制器的通信中断周期未完成（数据过滤进程被中断，数据同步进程被中断）。数据库指针或关键字无效数据库中的数据元素无效或遭到破坏。
* 测试范围：
* 技术：应该使用为功能和业务周期测试创建的测试来创建一系列的事务。一旦达到预期的测试起点，就应该分别执行或模拟以下操作：（1）客户机断电：关闭PC机的电源；（2）服务器断电：模拟或启动服务器的断电过程；（3）通过网络服务器产生的中断：模拟或启动网络的通信中断（实际断开通信线路的连接或关闭网络服务器或路由器的电源）；（4）DASD和DASD控制器被中断、断电或与DASD和DASD控制器的通信中断：模拟与一个或多个DASD控制器或设备的通信，或实际取消这种通信。一旦实现了上述情况（或模拟情况），就应该执行其他事务。而且一旦达到第二个测试点状态，就应调用恢复过程。在测试不完整的周期时，所使用的技术与上述技术相同，只不过应异常终止或提前终止数据库进行本身。对以下情况的测试需要达到1个已知的数据库状态。当破坏若干个数据库字段指针和关键字时，应该以手工方式在数据库中（通过数据库工具）直接进行。其他事务应该通过使用“应用程序功能测试”和“业务周期测试”中的测试来执行，并且应执行完整的周期。
* 开始标准：
* 完成标准：在所有上述情况中，应用程序、数据库和系统应该在恢复过程完成时立即返回一个已知的预期状态。此状态包括仅限于已知损坏的字段、指针或关键字范围内的数据损坏，以及表明进程或事物因中断面未被完成的报表。
* 测试重点和优先级：
* 需要考虑的特殊事项：恢复测试会给其他操作带来许多麻烦。断开缆线连接方法（模拟断电或通信中断）可能并不可取或不可行。所以，可能会需要采用其他方法，例如诊断性软件工具。需要系统（或计算机操作）、数据库和网络组中的资源。这些测试应该在工作时间之外或在1台独立的计算机上运行。

**配置测试**

配置测试用来核实测试对象在不同的软件和硬件配置中的运行情况。在大多数生产环境中，客户机工作站、网络连接和数据库服务器的具体硬件规格会有所不同。客户机工作站可能会安装不同的软件，例如，应用程序、驱动程序等，而且在任何时候，都可能运行许多不同的软件组合，从而占用不同的资源。

* 测试目标：核实测试可否在所需的硬件和软件配置中正常运行。
* 测试范围：
* 技术：使用功能测试脚本。在测试过程中或在测试开始之前，打开各种与非测试对象相关的软件（例如Microsoft应用程序；Excel和Word），然后将其关闭。执行所选的事务，以模拟ACTOR于测试对象软件和非测试对象软件之间的交互。重复上述步骤，尽量减少客户机工作站上的常规可用内存。
* 开始标准：
* 完成标准：对于测试对象软件和非测试对象软件的各种组合，所有事务都成功完成，没有出现任何故障。
* 测试重点和优先级：
* 需要考虑的特殊事项：需要、可以使用并可以通过桌面访问哪些非测试对象软件？通常使用的是哪些应用程序？应用程序正在运行什么数据？例如，在Excel中打开的大型电子表格，或是在Word中打开的100页的文档。作为此测试的一部分，应将整修系统、Netware、网络服务器、数据库等都记录下来。

**安装测试**

安装测试有两个目的：第一个目的是确保该软件在正常情况和异常情况的不同条件下都能进行安装。例如，进行首次安装、升级、完整的或自定义的安装。异常情况包括磁盘空间不足、缺少目录创建权限等。第二个目标是核实软件在安装后可立即正常运行。这通常是指为运行大量功能测试制定的测试。

* 测试目标：核实在以下情况下，测试对象可正确地安装到各种所需的硬件配置中。首次安装——从前从未安装过“项目名称”的新计算机；更新——以前安装过相同版本的“项目名称”的计算机更新。
* 测试范围：
* 技术：手工开发脚本或开发自动脚本，以验证目标计算机的状况。“项目名称”从未安装过；“项目名称”安装过程相同或较早的版本。启动或执行安装，使用预先确定的功能测试脚本子集来运行事务。
* 开始标准：
* 完成标准：“项目名称”事务成功执行，没有出现任何故障。
* 测试重点和优先级：
* 需要考虑的特殊事项：应该选择“项目名称”的哪些事务才能准确地测试出“项目名称”应用程序已经成功安装，而且没有遗漏主要的软件构件？

**问题严重程度描述**

|  |  |  |
| --- | --- | --- |
| 问题严重程度 | 描述 | 响应时间 |
| 高 | 例如使系统崩溃 | 程序员多长时间内改正次问题 |
| 中 |  |  |
| 地 |  |  |

**附录**

**项目任务**

以下是一些于测试有关的任务。

**制定测试计划**

1. 确定测试需求
2. 评估风险
3. 制定测试策略
4. 确定测试策略
5. 创建时间表
6. 生成测试计划

**设计测试**

1. 准备工作量分析文档
2. 确定并说明测试用例
3. 确定测试过程，并创建测试过程的结果

**复审和评估测试覆盖**

**实施测试**

1. 记录或通过变成创建测试脚本
2. 确定设计与实施模型中的测试专用功能
3. 建立外部数据库

**执行测试**

**执行测试过程**

**评估测试执行情况**

**恢复暂停的测试**

**核实结果**

**调查意外结果**

**记录缺陷**

**对测试进行评估**

**评估测试用例覆盖率**

**评估代码覆盖率**

**分析缺陷**

**确定是否达到了测试完成标准于成功标准**

# 测试用例模板

每个用例应说明下列内容：

(1)测试用例名称

(2)测试项：规定并简要说明本测试用例所要涉及的项和特性、对于每一项、可考虑引用需求说明书和设计说明书。

(3)输入说明：规定执行测试用例所需的各个输入。有些输入可以用值（允许适当的误差）来规定。而另一些输入，如常数表或事务文件可以用名来规定。规定所有合适的数据库、文件、终端信息、内存常驻区域和由操作系统传送的值。规定各输入间所需的所有关系（如时序关系等）。

(4)输出说明：规定测试项的所有输出和特性（如：响应时间）。提供各个输出或特性的正确值（在适当的误差范围内）。

(5)环境要求：规定执行本测试用例所需的硬件特征和配置、系统软件和应用软件。

(6)规程说明：描述对执行本测试用例的测试规程的一切特殊限制。这些限制可以包括特定的准备、操作人员干预、确定特殊的输出和清除过程。

(7)用例间的依赖关系：列出必须在本测试用例之前执行的测试用例名称，归纳依赖性质。

建议采用表格描述每个用例。

航空公司为随时掌握各个航班飞机的乘载情况，需要定期进行查询统计，以便适当调整。航空公司能够统计一段时间内售出飞往某地的机票，需要测试该功能，我们采用等价类划分方法测试改功能。程序界面如下图所示：

起始日期：

结束日期：

目的地：

确 定

时间段售票统计等价类a测试合理输入的结果，详细说明如表1所示：

表1 时间段售票统计等价类a测试用例

|  |  |
| --- | --- |
| 测试用例编号： | HKST-001 |
| 测试用例名称： | 时间段售票统计等价类a |
| 测试技术 | 等价类划分 |
| 测试项： | 起始日期从2010年1月1日开始到当前日期；  结束日期从2010年1月1日开始到当前日期，结束日期不能早于起始日期；  目的地只能从数据库中选择，若为空，则不限制地点。 |
| 前置测试用例： | 航空公司管理员身份验证 |
| 输入说明： | 1.起始日期输入2010.10.1；  2.结束日期输入2010.12.1；  3.目的地输入空；  4.点击确定按钮（字符界面程序通常没有这样的步骤）。 |
| 输出说明(预期结果)： | 系统输出共有10张票售出。 |
| 测试结果： | 成功 |

时间段售票统计等价类b测试非法输入的结果，起始日期不符合要求，详细说明如表2所示：

表2 时间段售票统计等价类b测试用例

|  |  |
| --- | --- |
| 测试用例编号： | HKST-03 |
| 测试用例名称： | 时间段售票统计等价类b |
| 测试项： | 起始日期从2010年1月1日开始到当前日期；  结束日期从2010年1月1日开始到当前日期，结束日期不能早于起始日期；  目的地只能从数据库中选择，若为空，则不限制地点。 |
| 前置测试用例： | 航空公司管理员身份验证 |
| 输入说明： | 1.起始日期输入2009.9.15；  2.结束日期输入2011.12.1；  3.目的地输入“长沙”；  4.点击确定按钮（字符界面程序通常没有这样的步骤）。 |
| 输出说明： | 系统提示：起始日期输入不合理。 |
| 测试结果： | 成功 |

时间段售票统计等价类c测试非法输入的结果，起始日期晚于结束日期不符合要求，详细说明如表3所示：

表3 时间段售票统计等价类c测试用例

|  |  |
| --- | --- |
| 测试用例编号： | HKST-04 |
| 测试用例名称： | 时间段售票统计等价类c |
| 测试项： | 起始日期从2010年1月1日开始到当前日期；  结束日期从2010年1月1日开始到当前日期，结束日期不能早于起始日期；  目的地只能从数据库中选择，若为空，则不限制地点。 |
| 前置测试用例： | 航空公司管理员身份验证 |
| 输入说明： | 1.起始日期输入2011.1.15；  2.结束日期输入2010.12.1；  3.目的地输入“北京”；  4.点击确定按钮（字符界面程序通常没有这样的步骤）。 |
| 输出说明： | 系统输出共有1张票售出。 |
| 测试结果： | 失败 |

前述三个测试用例的测试内容，可合并成一个测试用例。时间段售票统计等价类划分测试用例，测试内容包括四个无效等价类（起始日期无效，结束日期无效，结束日期早于起始日期，目的地无效），两个有效等价类（起始日期和结束日期有效，目的地为“空”或“从数据库中选择”），详细说明如表4所示。

表4 时间段售票统计等价类划分测试用例

|  |  |
| --- | --- |
| 测试用例编号： | HKST-05 |
| 测试用例名称： | 时间段售票统计等价类划分 |
| 测试项： | 起始日期从2010年1月1日开始到当前日期；  结束日期从2010年1月1日开始到当前日期，结束日期不能早于起始日期；  目的地只能从数据库中选择，若为空，则不限制地点。 |
| 前置测试用例： | 航空公司管理员身份验证 |
| 输入说明： | 1.起始日期输入2010.10.1，结束日期输入2010.12.1，目的地输入“”；  2.起始日期输入2010.10.1，结束日期输入2010.12.1，目的地输入“长沙”；  3.起始日期输入2009.9.15，结束日期输入2011.12.1；目的地输入“长沙”；  4.起始日期输入2011.1.15，结束日期输入2010.12.1，目的地输入“北京”；  5.起始日期输入2011.1.15，结束日期输入2015.12.1，目的地输入“北京”；  6.起始日期输入2010.2.1，结束日期输入2011.5.1，目的地输入“洪山桥”。  六个测试项输入结束后，都点击确定按钮。 |
| 输出说明： | 1.系统输出共有10张票售出。  2.系统输出共有2张票售出。  3.系统提示：起始日期输入不合理；  4.系统提示：结束日期不能早于起始日期；  5.系统提示：结束日期输入不合理；  6.系统提示：找不到目的地。 |
| 测试结果： | 1.成功 2.成功 3. 成功 4. 成功 5. 成功 6. 成功 |

# 航班管理设计及代码

实现了一个简单的航班管理程序，可以增加一个航班记录，可以删除航班记录，所有航班记录保存在文件中。另外还提供了两个查询功能：以票价来查询航班，以航空公司名称来查询航班。整个工程包括如下类：

MainProcess，主程序。显示主窗口，创建程序的菜单。

HFlightInfo，单个航班记录。

HManaFlight，航班管理类，里面维护了一个数组，所有航班记录都在数组中。它是其它类和HFlightInfo打交道的中介。

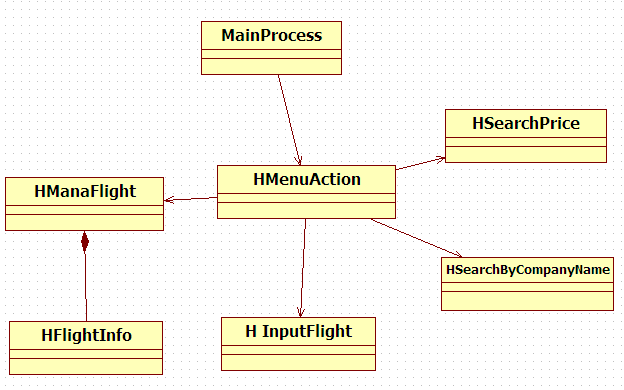
HInputFlight，提供了一个对话框，当要添加航班时，此对话框提示输入航班的各项信息。对话框由Java.swing技术实现。

HMenuAction，菜单类。它是整个工程的中枢，大部分类都和它产生关联，这样减少类之间的耦合。当然，HMenuAction也可以合并到MainProcess。之所以没有这样做，是考虑到，若合并，合并后的类承担的东西太多，不好维护。所有菜单的响应都是在此类中。

HSearchPrice，以票价来查询航班。提供了一个对话框，用户可以输入票价的范围，点击查询后，软件显示满足查询条件的所有航班。

HSearchByCompanyName，以航空公司名称来查询航班。提供了一个对话框，用户可以输入航空公司名称，点击查询后，软件显示满足查询条件的所有航班。

软件的类图如下所示：



所有类之间的关系都是单向的，并且除HMenuAction外，其它类都只与一个类产生关联。软件的整个架构比较简单，容易维护，大大减少了不同类之间耦合。这种设计思路值得同学们借鉴。**1，类：MainProcess**

（接上一页：MainProcess类）

import java.awt.FlowLayout;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JTextArea;

import javax.swing.WindowConstants;

public class MainProcess extends JFrame{

/\*\*

\* 这是我的主程序，创建一个Java的Frame,使用Swing技术。（作者：何海江）

\*/

private static final long serialVersionUID = 1L;

private static MainProcess m\_mProc = null;

public static void main(String[] args) {

//现在创建了一个对象，不过什么都显示不出来

m\_mProc = new MainProcess();

//添加菜单栏，不能设定位置，会自动放在最上部

//创建菜单处理类

HMenuAction hmaMenuAction = new HMenuAction();//菜单响应事件

JMenuBar jmb = AddMenu(hmaMenuAction);

m\_mProc.setJMenuBar(jmb);

//加上这一句就可以显示一个仅有关闭，最小化，最大化的按钮的Frame了

m\_mProc.setVisible(true);

m\_mProc.setTitle("白盒测试教学案例");

//再加上这一句就可以显示一个在左上角，拥有指定大小的Frame了

m\_mProc.setSize(800,600);

//再加上这一句就可以把Frame放在最中间了

m\_mProc.setLocationRelativeTo(null);

//如果没有这一句，在点击关闭Frame的时候程序其实还是在执行状态中的，加上这一句才算是真正的把资源释放掉了

m\_mProc.setDefaultCloseOperation(WindowConstants.DISPOSE\_ON\_CLOSE);

//显示所有航班记录

m\_mProc.setLayout(new FlowLayout());

JLabel txtPrompt = new JLabel("<html><body><h1>所有航班记录</h1></body></html>");

m\_mProc.add(txtPrompt);

JTextArea txtAllFlight = null; //显示所有航班记录

txtAllFlight = new JTextArea("");

txtAllFlight.setLineWrap(true);// 如果内容过长。自动换行

//txtAllFlight.setRows(100);

txtAllFlight.setColumns(100);

txtAllFlight.setEditable(false);

hmaMenuAction.SetShowArea(txtAllFlight);//关联显示区域，减少耦合。

m\_mProc.add(txtAllFlight);

m\_mProc.setLayout(new FlowLayout(FlowLayout.LEFT));

//mProc.setLayout(new GridLayout());//每行两个组件

m\_mProc.validate();

}

（接上一页：MainProcess类）

//添加菜单条。

//菜单处理类 hmaAction

protected static JMenuBar AddMenu(HMenuAction hmaAction)

{

JMenuBar jmbWork; //主菜单条组件

JMenu mnuFile, mnuOperator, mnuHelp;//菜单

JMenuItem jmiReadFile, jmiWriteFile, jmiAddTick,jmiDelTick,jmiRemoveAll;//mnuFile文件的菜单项

JMenuItem jmiSearchPrice, jmiSearchCompany;//mnuOperator查询统计的菜单项

JMenuItem jmiIntroduce, jmiAboutMe;//mnuHelp帮助的菜单项

//创建菜单

jmbWork = new JMenuBar();

//文件操作

mnuFile = new JMenu("文件(F)");

mnuFile.setMnemonic('F'); //助记符

jmiReadFile = new JMenuItem(HMenuAction.strReadFile);

jmiReadFile.addActionListener(hmaAction );// 菜单添加事件

jmiWriteFile = new JMenuItem(HMenuAction.strWriteFile);

jmiWriteFile.addActionListener(hmaAction );// 菜单添加事件

jmiAddTick = new JMenuItem(HMenuAction.strAddFlight);

jmiAddTick.addActionListener(hmaAction );// 菜单添加事件

jmiDelTick = new JMenuItem(HMenuAction.strDelFlight);

jmiDelTick.addActionListener(hmaAction );// 菜单添加事件

jmiRemoveAll = new JMenuItem(HMenuAction.strRemoveAllFlight);

jmiRemoveAll.addActionListener(hmaAction );// 菜单添加事件

//添加菜单项至菜单上

mnuFile.add(jmiReadFile);

mnuFile.add(jmiWriteFile);

mnuFile.insertSeparator(2);

mnuFile.add(jmiAddTick);

mnuFile.add(jmiDelTick);

mnuFile.add(jmiRemoveAll);

//将菜单加入至菜单栏

jmbWork.add(mnuFile);

//查询统计

mnuOperator = new JMenu("查询统计(D)");

mnuOperator.setMnemonic('D'); //助记符

jmiSearchPrice = new JMenuItem(HMenuAction.strPriceSearch);

jmiSearchPrice.addActionListener(hmaAction );// 菜单添加事件

jmiSearchCompany = new JMenuItem(HMenuAction.strAirlineNameSearch);

jmiSearchCompany.addActionListener(hmaAction );// 菜单添加事件

//添加菜单项至菜单上

mnuOperator.add(jmiSearchPrice);

mnuOperator.add(jmiSearchCompany);

//将菜单加入至菜单栏

jmbWork.add(mnuOperator);

//查询统计

mnuHelp = new JMenu("帮助(H)");

mnuHelp.setMnemonic('H'); //助记符

jmiIntroduce = new JMenuItem(HMenuAction.strHelp);

jmiIntroduce.addActionListener(hmaAction );// 菜单添加事件

jmiAboutMe = new JMenuItem(HMenuAction.strAboutMe);

jmiAboutMe.addActionListener(hmaAction );// 菜单添加事件

//添加菜单项至菜单上

mnuHelp.add(jmiIntroduce);

mnuHelp.add(jmiAboutMe);

//将菜单加入至菜单栏

jmbWork.add(mnuHelp);

//建立主菜单

return jmbWork;

}

}

2，类：HFlightInfo

（接上一页：HFlightInfo类）

public class HFlightInfo {

private String m\_strID; //航班号

private String m\_strName; //航班名称

private String m\_strCompany; //航空公司名称

private float m\_fPrice; //价格

private int m\_nSeat; //座位数目

//构造函数

public HFlightInfo()

{

setStrID("");

setStrName("");

setStrCompany("");

setfPrice(0.0f);

setSeat(0);

}

//允许所有属性设置的构造函数

public HFlightInfo(String sid, String sname, String scmy, float price, int seat)

{

setStrID(sid);

setStrName(sname);

setStrCompany(scmy);

setfPrice(price);

setSeat(seat);

}

//获取航空公司名称

public String getStrCompany() {

return m\_strCompany;

}

//设置航空公司名称

public void setStrCompany(String strCompany) {

this.m\_strCompany = strCompany;

}

//获取航班名称

public String getStrName() {

return m\_strName;

}

**3，类：HManaFlight**

//设置航班名称

public void setStrName(String strName) {

this.m\_strName = strName;

}

//获取航班号

public String getStrID() {

return m\_strID;

}

//设置航班号

public void setStrID(String strID) {

this.m\_strID = strID;

}

//获取价格

public float getfPrice() {

return m\_fPrice;

}

//设置价格

public void setfPrice(float fPrice) {

this.m\_fPrice = fPrice;

}

//获取座位数目

public int getSeat() {

return m\_nSeat;//by hehaijiang

}

//设置座位数目

public void setSeat(int nSeat) {

this.m\_nSeat = nSeat;

}

//获取航班号信息,打包为字符串

//航班信息包括：航班号，航班名称，航空公司名称，价格，座位数目

public String GetInfo()

{

StringBuffer sbInfo = new StringBuffer();

sbInfo.append("航班号:"+ m\_strID+", ");

sbInfo.append("航班名称:"+m\_strName+", ");

sbInfo.append("航空公司名称:"+m\_strCompany+", ");

sbInfo.append("价格:"+String.valueOf(m\_fPrice)+", ");

sbInfo.append("座位数目:"+String.valueOf(m\_nSeat));

return sbInfo.toString();

}

}

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.ArrayList;

import javax.swing.JOptionPane;

/\*\* \* 这是航班管理类。（作者：何海江）

\* 增加一个航班，删除一个航班，删除所有航班。

\* 从文件读航班信息，将航班信息写入文件 \*/

public class HManaFlight {

private ArrayList<HFlightInfo> m\_listFlight = null; //保存所有航班。

//构造函数，初始化数组。ArrayList

public HManaFlight()

{

m\_listFlight = new ArrayList<HFlightInfo>();

}

//析构函数

protected void finalize()

{

m\_listFlight.removeAll(null);

}

/\*增加一个航班到数列m\_listFlight

航班号strID 航班名称strName

航空公司名称strCompany fPrice; 价格 nSeat; 座位数目 \*/

public void AddFlight(String strID,String strName,String strCompany,

float fPrice,int nSeat)

{

HFlightInfo hfiFlight = new HFlightInfo(strID,strName,strCompany,fPrice,nSeat);

m\_listFlight.add(hfiFlight);

}

// 根据航班名称，获得以换行符结尾的航班信息

//返回该航班的所有信息，包括：航班号，航班名称，航空公司名称，价格，座位数目

private String GetFlightInfo(String strName)

{

int nSize = m\_listFlight.size();

StringBuffer sbInfo = null;

// 遍历整个ArrayList

for(int i=0;i<nSize;i++)

{

// 取出HFlightInfo对象

HFlightInfo hfiFlight=(HFlightInfo)m\_listFlight.get(i);

// 比较姓名

if(hfiFlight.getStrName().equals(strName))

{

sbInfo = new StringBuffer();

sbInfo.append(hfiFlight.GetInfo()+"\r\n");

break;

}

} //end of for

return sbInfo.toString();

}

（接上一页：HManaFlight 类）

//可以将航班从列表中删除

//strName航班名称

//如果找到对应航班，并删除成功，则返回true，否则返回false

public boolean DelFlight(String strName)

{

boolean flag = false;

int nSize = m\_listFlight.size();

for(int i=0;i<nSize;i++)

{

HFlightInfo hfiFlight=(HFlightInfo)m\_listFlight.get(i);

if(hfiFlight.getStrName().equals(strName))

{

m\_listFlight.remove(i);

flag = true;

break;

}

}//end of for

return flag;

}

//删除所有航班

public void RemoveAllFlight()

{

m\_listFlight.clear();//删除所有航班

}

//从文件FlightInfo.txt中读入航班信息

//如果找到文件，并读入成功，则返回true，否则返回false

private boolean ReadFlightFile()

{

boolean bFlag = true;

m\_listFlight.clear();//清除旧的

try{

DataInputStream fm = new DataInputStream(new BufferedInputStream(

new FileInputStream("FlightInfo.txt")));

//航班个数

int number = fm.readInt();

for( int i=0;i<number;i++ )

{

String strID = fm.readUTF();//航班号

String strName = fm.readUTF();//航班名称

String strCompany = fm.readUTF();//航空公司名称

float fPrice = fm.readFloat();//价格

int nSeat = fm.readInt();//座位数目

AddFlight(strID,strName,strCompany,fPrice,nSeat);

}

fm.close();

}//end of for...

catch(IOException e)

{

//System.err.println("文件FlightInfo.txt读错误。"+e.toString());

JOptionPane.showMessageDialog(null,

"<html><body><h1>文件FlightInfo.txt读错误。</h1><h3>检查该文件是否存在</h3></body></html>",

（接上一页：HManaFlight 类）

"报警信息",JOptionPane.WARNING\_MESSAGE);

bFlag = false;

}

return bFlag;

}

//向文件FlightInfo.txt写入航班信息

//如果写入成功，则返回true，否则返回false

public boolean WriteFlightFile()

{

boolean bFlag = true;

try{

DataOutputStream fm = new DataOutputStream(new BufferedOutputStream(

new FileOutputStream("FlightInfo.txt")));

//航班个数

int number = m\_listFlight.size();

fm.writeInt(number);

for( int i=0;i<number;i++ )

{

// 取出hfiFlight对象

HFlightInfo hfiFlight=(HFlightInfo)m\_listFlight.get(i);

fm.writeUTF(hfiFlight.getStrID());//航班号

fm.writeUTF(hfiFlight.getStrName());//航班名称

fm.writeUTF(hfiFlight.getStrCompany());//航空公司名称

fm.writeFloat(hfiFlight.getfPrice());//价格

fm.writeInt(hfiFlight.getSeat());//座位数目

}

fm.close();

}

catch(IOException e)

{

//System.err.println("文件passenger.txt写错误。"+e.toString());

JOptionPane.showMessageDialog(null,

"<html><body><h1>文件FlightInfo.txt写入错误。</h1><h3>检查该文件是否存在</h3></body></html>",

"程序出错",JOptionPane.ERROR\_MESSAGE);

bFlag = false;

}

return bFlag;

}

//返回该航班的所有信息，包括：航班号，航班名称，航空公司名称，价格，座位数目

public String GetAllFlightInfo()

{

int nSize = m\_listFlight.size();

StringBuffer sbInfo = new StringBuffer();

// 遍历整个ArrayList

for(int i=0;i<nSize;i++)

{

// 取出HFlightInfo对象

HFlightInfo hfiFlight=(HFlightInfo)m\_listFlight.get(i);

if( i!=(nSize-1) )

sbInfo.append(hfiFlight.GetInfo()+"\r\n");

（接上一页：HManaFlight 类）

else//最后一个不换行

sbInfo.append(hfiFlight.GetInfo());

} //end of for

return sbInfo.toString();

}

//从文件读入所有航班信息，并打包为字符串

//如果读入成功，则返回true，strAllInfo装入所有航班记录；否则返回false，

//注意：调用前，保证sbAllInfo开辟了空间。

public boolean ReadFileToString(StringBuffer sbAllInfo)

{

boolean bFlag = ReadFlightFile();

if( bFlag==true )

sbAllInfo.append(GetAllFlightInfo());

return bFlag;

}

//从当前的航班记录内搜索票价在一个范围内的航班

//fMin最小票价

public String SearchPriceRange(float fMin,float fMax)

{

int nSize = m\_listFlight.size();

StringBuffer sbInfo = new StringBuffer();

// 遍历整个ArrayList

for(int i=0;i<nSize;i++)

{

// 取出HFlightInfo对象

HFlightInfo hfiFlight=(HFlightInfo)m\_listFlight.get(i);

String strTmp;

if( hfiFlight.getfPrice()>=fMin && hfiFlight.getfPrice()<=fMax )

{

strTmp = hfiFlight.GetInfo();

sbInfo.append(strTmp+"\r\n");

}

} //end of for

return sbInfo.toString();

}

//从当前的航班记录内搜索所属航空公司所有的航班

//strCompany航空公司名称

public String SearchByCompanyName(String strCompany)

{

int nSize = m\_listFlight.size();

StringBuffer sbInfo = new StringBuffer();

// 遍历整个ArrayList

for(int i=0;i<nSize;i++)

{

// 取出HFlightInfo对象

HFlightInfo hfiFlight=(HFlightInfo)m\_listFlight.get(i);

String strTmp;

if( true==hfiFlight.getStrCompany().equals(strCompany) )

{

strTmp = hfiFlight.GetInfo();

sbInfo.append(strTmp+"\r\n");

}

} //end of for

return sbInfo.toString();

}

}

**4，类：HInputFlight**

（接上一页：HInputFlight 类）

import java.awt.Container;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

/\*\*

\* 这是航班信息录入类。（作者：何海江）

\*/

public class HInputFlight extends JDialog implements ActionListener{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

JLabel lblID;//航班号

JTextField txtID;//航班号

JLabel lblFname;//航班名称

JTextField txtFname;//航班名称

JLabel lblCompany;//航空公司名称

JTextField txtCompany;//航空公司名称

JLabel lblPrice;//价格

JTextField txtPrice;//价格

JLabel lblSeat;//座位数目

JTextField txtSeat;//座位数目

JButton btnOK,btnCancel;//确定、取消按钮

boolean bDoOK = false; //如果点击确定按钮，则是true，有效航班数据；否则false

//构造函数

HInputFlight(JFrame jFrame,String strTitle)

{

super(jFrame,strTitle,true);//模态

//this.setBounds(100, 100, 400, 400);

Container con = this.getContentPane();

//航班号

lblID = new JLabel("输入航班号（3-8位的数字字符）:");

txtID = new JTextField(8);

txtID.setEditable(true);

con.add(lblID);

con.add(txtID);

//航班名称

lblFname = new JLabel("输入航班名称（5-10位的数字及英文大写字符）:");

txtFname = new JTextField(10);

txtFname.setEditable(true);

con.add(lblFname);

con.add(txtFname);

//航空公司名称

lblCompany = new JLabel("输入航空公司名称（以H开头，后接5-19个任意字符）:");

txtCompany = new JTextField(20);

txtCompany.setEditable(true);

con.add(lblCompany);

con.add(txtCompany);

//价格

lblPrice = new JLabel("输入机票价格（允许小数，100<=价格<=1800）:");

txtPrice = new JTextField(8);

txtPrice.setEditable(true);

con.add(lblPrice);

con.add(txtPrice);

//座位数目

lblSeat = new JLabel("输入座位数目（整数，100到200之间）:");

txtSeat = new JTextField(6);

txtSeat.setEditable(true);

con.add(lblSeat);

con.add(txtSeat);

//setModalityType(false);

btnOK = new JButton("确定");

btnOK.addActionListener(this);

btnCancel = new JButton("取消");

btnCancel.addActionListener(this);

con.add(btnOK);

con.add(btnCancel);

con.setVisible(true);

con.setLayout(new FlowLayout(FlowLayout.LEFT));

con.setLayout(new GridLayout(6,2,25,40));//每行两个组件

con.setSize(200,100);

this.pack();

this.setLocationRelativeTo(null);

}

//"确定""取消"按钮的响应

public void actionPerformed(ActionEvent e)

{

if( e.getActionCommand()=="确定")

{

boolean bResult = ValidFlightInfo();

if( false==bResult )

JOptionPane.showMessageDialog(this,

"录入数据无法通过验证，请重新录入，或取消。",

"警告信息",JOptionPane.PLAIN\_MESSAGE);

else

{

setVisible(false);

dispose();

bDoOK = true;

//System.out.println("确定");

}

}

（接上一页：HInputFlight 类）

else if( e.getActionCommand()=="取消")

{

setVisible(false);

dispose();

bDoOK = false;

}//end of else if

}

//检查用户输入数据是否有效。

private boolean ValidFlightInfo()

{

if( ValidID()==false )

return false;

if( ValidFname()==false )

return false;

if( ValidCompany()==false )

return false;

if( ValidPrice()==false )

return false;

if( ValidSeat()==false )

return false;

//by hehaijiang

return true;

}

//验证航班名称是否输入正确。

//输入航班名称（5-10位的数字及英文大写字符）

//有效，返回true;否则，返回false.

private boolean ValidFname()

{

boolean flag = true;

String str = txtFname.getText();

int len = str.length();

if( len<5 || len>10 )

flag = false;

else

{

for( int i=0;i<len;i++ )

{

char ch = str.charAt(i);

if( (ch>='0'&& ch<='9') ||

(ch>='A'&& ch<='Z') )

continue;

else

{

flag = false;

break;

}

}//end of for

}//end of else

return flag;

}

（接上一页：HInputFlight 类）

//验证航班号是否输入正确。

//输入航班号（3-8位的数字字符）

//有效，返回true;否则，返回false.

private boolean ValidID()

{

boolean flag = true;

String str = txtID.getText();

int len = str.length();

if( len<3 || len>8 )

flag = false;

else

{

for( int i=0;i<len;i++ )

{

char ch = str.charAt(i);

if( ch<'0'|| ch>'9' )

{

flag = false;

break;

}

}//end of for

}//end of else

return flag;

}

//验证航空公司名称是否输入正确。

//输入航空公司名称（以H开头，后接5-19个任意字符）

//有效，返回true;否则，返回false.

private boolean ValidCompany()

{

boolean flag = true;

String str = txtCompany.getText();

int len = str.length();

if( len<6 || len>20 )

flag = false;

else

{

char ch = str.charAt(0);

if( ch!='H')

flag = false;

}

return flag;

}

//验证机票价格是否输入正确。

//输入机票价格（允许小数，100<=价格<=1800

//有效，返回true;否则，返回false.

private boolean ValidPrice()

{

boolean flag = true;

String str = txtPrice.getText();

float price = 0.0f;

try

{//txtPrice可能为空字符串或者其它非法浮点数字符串，所以要出错处理

price = Float.parseFloat(str);

}

（接上一页：HInputFlight 类）

catch(Exception e)

{

flag = false;

}

if( (true==flag) && (price>=100.0f) && (price<=1800.0f) )

flag = true;

return flag;

}

//验证座位数目是否输入正确。

//输入座位数目（整数，100到200之间）`

//有效，返回true;否则，返回false.

private boolean ValidSeat()

{

boolean flag = true;

String str = txtSeat.getText();

int seat = 0;

try

{//txtSeat可能为空字符串或者其它非法整数字符串，所以要出错处理

seat = Integer.parseInt(str);

}

catch(Exception e)

{

flag = false;

}

if( true==flag && seat>=100 && seat<=200 )

flag = true;

return flag;

}

//如果点击确定按钮，则返回true；

//否则返回false

public boolean GetValidFlight()

{

return bDoOK;

}

//将输入的航班记录保存到各项参数中

//调用前，保证sbSid，sbName,sbCompany开辟了空间。

//注意：此处宁愿麻烦地用五个参数，而不简单地用一个类HFlightInfo来传递，是为了消除HFlightInfo和HMenuAction之间的耦合。

//同学们在以后的代码中可以学习这样的方法。

//做大一点的项目，一定要遵循软件工程的基本原则：低耦合；高内聚。

public void GetInputFlightRecord(StringBuffer sbSid, StringBuffer sbName, StringBuffer sbCompany, float[] price, int[] seat)

{

sbSid.append(this.txtID.getText());//航班号

sbName.append(this.txtFname.getText());//航班名称

sbCompany.append(this.txtCompany.getText());//航空公司名称

String str = this.txtPrice.getText();

price[0] = Float.parseFloat(str);//价格

str = this.txtSeat.getText();

seat[0] = Integer.parseInt(str);//座位数目

}

}

**5，类：HMenuAction**

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JOptionPane;

import javax.swing.JTextArea;

public class HMenuAction implements ActionListener{

public static final String strReadFile = "从文件读入数据";

public static final String strWriteFile = "保存数据到文件";

public static final String strAddFlight = "增加一个航班";

public static final String strDelFlight = "删除一个航班";

public static final String strRemoveAllFlight = "删除所有航班";

public static final String strPriceSearch = "以票价为条件检索";

public static final String strAirlineNameSearch = "以航空公司名称为条件检索";

public static final String strHelp ="软件功能介绍";

public static final String strAboutMe ="关于......";

private HManaFlight m\_hmfDmana; //航班管理类

private JTextArea m\_jtaShow; //关联显示区域，减少耦合。

//构造函数

HMenuAction()

{

m\_hmfDmana = new HManaFlight();

}

//显示所有航班信息。

public void SetShowArea(JTextArea jxa)

{

m\_jtaShow = jxa;

}

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

String strMenu = e.getActionCommand();

if( strHelp==strMenu )

DoHelp();

else if( strAboutMe==strMenu )

DoAboutMe();

else if( strReadFile==strMenu )

{//从文件读入数据：航班记录

StringBuffer strAllFlightInfo = new StringBuffer();

boolean bResult = m\_hmfDmana.ReadFileToString(strAllFlightInfo);

if( bResult==true )

ShowAllFlight(strAllFlightInfo.toString());

}

else if( strWriteFile==strMenu )

{//保存数据：航班记录到文件

boolean bResult = m\_hmfDmana.WriteFlightFile();

if( bResult==true )

JOptionPane.showMessageDialog(null,

"<html><body><h1>所有航班记录成功保存到文件。</h1></body></html>",

"保存文件",JOptionPane.PLAIN\_MESSAGE);

}

（接上一页：HMenuAction类）

else if( strAddFlight==strMenu )//增加一个航班

{

AddOneFlight();

}

else if( strDelFlight==strMenu )//删除一个航班

{

DelOneFlight();

}//end of else if ( strDelFlight==strMenu )//删除一个航班

else if( strRemoveAllFlight==strMenu )//删除所有航班

{

m\_hmfDmana.RemoveAllFlight();

ShowAllFlight("");

}

else if( strPriceSearch==strMenu )//以票价为条件检索

{

SearchFlightByPrice();

}

else if( strAirlineNameSearch==strMenu )//以航空公司名称为条件检索

{

SearchFlightByCompanyName();

}

else{}

}//end of actionPerformed

//以票价查询航班记录

private void SearchFlightByPrice()

{

HSearchPrice hsp = new HSearchPrice(null);

hsp.setVisible(true);

boolean bResult = hsp.GetValidSearch();

if( true==bResult )

{

float fMin = hsp.GetMinPrice();

float fMax = hsp.GetMaxPrice();

String strInfo = m\_hmfDmana.SearchPriceRange(fMin, fMax);

if( strInfo.length()<2 )

JOptionPane.showMessageDialog(null,

"<html><body><h1>没有找到符合条件的航班</h1></body></html>",

"查询结果",JOptionPane.PLAIN\_MESSAGE);

else

JOptionPane.showMessageDialog(null, strInfo,

"查询结果",JOptionPane.PLAIN\_MESSAGE);

}

}

//以票价查询航班记录

private void SearchFlightByCompanyName()

{

HSearchByCompanyName hsbc = new HSearchByCompanyName(null);

hsbc.setVisible(true);

boolean bResult = hsbc.GetValidSearch();

if( true==bResult )

{

String strCompany = hsbc.GetCompanyTitle();

String strInfo = m\_hmfDmana.SearchByCompanyName(strCompany);

（接上一页：HMenuAction类）

if( strInfo.length()<2 )

JOptionPane.showMessageDialog(null,

"<html><body><h1>没有找到符合条件的航班</h1></body></html>",

"查询结果",JOptionPane.PLAIN\_MESSAGE);

else

JOptionPane.showMessageDialog(null, strInfo,

"查询结果",JOptionPane.PLAIN\_MESSAGE);

}

}

//增加一个航班

private void AddOneFlight()

{

HInputFlight hif = new HInputFlight(/\*mpFrame\*/null,"输入航班的相关数据");

hif.setVisible(true);

//HFlightInfo hfiInfo=null;//这么做，会产生HFlightInfo和HMenuAction之间的耦合。

boolean bResult = hif.GetValidFlight();

if( true==bResult )

{

StringBuffer strSid = new StringBuffer();

StringBuffer strName = new StringBuffer();

StringBuffer strCompany = new StringBuffer();

float[] fPrice = {0.0f};

int[] nSeat = {0};

hif.GetInputFlightRecord(strSid,strName,strCompany,fPrice,nSeat);

m\_hmfDmana.AddFlight(strSid.toString(), strName.toString(), strCompany.toString(), fPrice[0], nSeat[0]);

String strInfo = m\_hmfDmana.GetAllFlightInfo();

ShowAllFlight(strInfo);

}

}

//删除一个航班

private void DelOneFlight()

{

String strName = JOptionPane.showInputDialog(null, "请输入要删除航班的航班名称","删除一个航班" , JOptionPane.INFORMATION\_MESSAGE);

if( null!=strName )

{ //strName=null按的取消按钮。

boolean bResult = m\_hmfDmana.DelFlight(strName);

if( true==bResult )//成功删除

ShowAllFlight(m\_hmfDmana.GetAllFlightInfo());

else

JOptionPane.showMessageDialog(null,

"<html><body><h1>该航班名称不存在，注意与航班号和航班公司名称区分开来！</h1></body></html>",

"删除一个航班",JOptionPane.PLAIN\_MESSAGE);

}

}

//about me关于......

private void DoAboutMe()

{

JOptionPane.showMessageDialog(null,

"<html><body><h1>白盒测试教学程序</h1><h3>参看实验指导书</h3></body></html>",

"关于......",JOptionPane.PLAIN\_MESSAGE);

}

（接上一页：HMenuAction类）

//help软件功能介绍

private void DoHelp()

{

JOptionPane.showMessageDialog(null,

"<html><body><h1>软件功能介绍</h1><h3>1,操作数据前，先读文件；使用结束，保存文件。</h3>"

+ "<h3>2,测试过程中，可通过菜单控制航班记录。</h3>"

+ "<h3>3,要有数据，才能查询统计。</h3></body></html>",

"软件功能介绍",JOptionPane.PLAIN\_MESSAGE);

}

//显示当前所有的航班记录

//航班记录放在strInfo中。

//供HMenuAction调用

public void ShowAllFlight(String strInfo)

{

m\_jtaShow.setText(strInfo);

}

//在当前所有的航班记录基础上，再添加一条航班记录

//航班记录放在strInfo中。

//供HMenuAction调用

public void IncrementShowOneFlight(String strInfo)

{

//String strAllFlight = m\_jtaShow.getText();

//m\_jtaShow.setText(strAllFlight+strInfo);

//mProc.validate();

}

}

**6，类：HSearchByCompanyName**

import java.awt.Container;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

（接上一页：HSearchByCompanyName类）

public class HSearchByCompanyName extends JDialog implements ActionListener

{

/\*\*

\* "以航空公司名称为条件检索"

\*/

private static final long serialVersionUID = 1L;

public static final String strSearch = "开始查询";

public static final String strCancel = "终止查询";

JTextField txtCompanyName;//航空公司名称输入框

String m\_strCompany;//航空公司名称

boolean bDoOK = false; //如果有效查询，则是true，；否则false

//构造函数

HSearchByCompanyName(JFrame jFrame)

{

super(jFrame,"以航空公司名称为查询条件，查询航班信息",true);//模态

m\_strCompany = "";

//this.setBounds(100, 100, 400, 400);

Container con = this.getContentPane();

JLabel lblTitle1 = new JLabel("以航空公司名称为查询条件，查询航班信息");

JLabel lblTitle2 = new JLabel("航空公司名称为6-20个任意字符。 ");

con.add(lblTitle1);

con.add(lblTitle2);

//航空公司名称

JLabel lblPrice1 = new JLabel("航空公司名称:");

txtCompanyName = new JTextField(10);

txtCompanyName.setEditable(true);

con.add(lblPrice1);

con.add(txtCompanyName);

//setModalityType(false);

JButton btnSearch = new JButton(strSearch);

btnSearch.addActionListener(this);

JButton btnCancel = new JButton(strCancel);

btnCancel.addActionListener(this);

con.add(btnSearch);

con.add(btnCancel);

con.setVisible(true);

con.setLayout(new FlowLayout(FlowLayout.LEFT));

con.setLayout(new GridLayout(3,2,25,40));//每行两个组件

con.setSize(200,100);

this.pack();

this.setLocationRelativeTo(null);

}

//"确定""取消"按钮的响应

public void actionPerformed(ActionEvent e)

{

if( e.getActionCommand()==strSearch)

{

boolean bResult = ValidCompanyName();

if( false==bResult )

JOptionPane.showMessageDialog(this,

"录入数据无法通过验证，请重新录入，或取消。",

"警告信息",JOptionPane.PLAIN\_MESSAGE);

（接上一页：HSearchByCompanyName类）

else

{//有效的输入价格，显示查询结果。

setVisible(false);

dispose();

bDoOK = true;

}

}

else if( e.getActionCommand()==strCancel)

{

setVisible(false);

dispose();

bDoOK = false;

}//end of else if

}//end of actionPerformed

//航空公司名称为6-20个任意字符。

private boolean ValidCompanyName()

{

boolean flag = true;

String str = txtCompanyName.getText();

int len = str.length();

if (len>=6 && len<=20 )

{

flag = true;

m\_strCompany = str;

}

else //非法的价格

flag = false;

return flag;

}

//获取航空公司名称

public String GetCompanyTitle()

{

return this.m\_strCompany;

}

//如果有效查询，则返回true；

//否则返回false

public boolean GetValidSearch()

{

return bDoOK;

}

}

**7，类：HSearchPrice**

import java.awt.Container;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

/\*\*

\* 这是以价格来查询航班的类。（作者：何海江）

\*/

public class HSearchPrice extends JDialog implements ActionListener

{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

public static final String strSearch = "开始查询";

public static final String strCancel = "终止查询";

JTextField txtMinPrice;//小的那个票价输入框

JTextField txtMaxPrice;//大的那个票价输入框

float fMinPrice,fMaxPrice;//小的那个票价,fMaxPrice大的那个票价

boolean bDoOK = false; //如果有效查询，则是true，；否则false

//构造函数

HSearchPrice(JFrame jFrame)

{

super(jFrame,"以票价为查询条件，查询航班信息",true);//模态

fMinPrice = 0.0f;

fMaxPrice = 0.0f;

//this.setBounds(100, 100, 400, 400);

Container con = this.getContentPane();

JLabel lblTitle1 = new JLabel("以票价为查询条件");

JLabel lblTitle2 = new JLabel("按终止查询，退出对话框。 ");

con.add(lblTitle1);

con.add(lblTitle2);

//小的那个票价

JLabel lblPrice1 = new JLabel("机票价格处于:");

txtMinPrice = new JTextField(8);

txtMinPrice.setEditable(true);

con.add(lblPrice1);

con.add(txtMinPrice);

//大的那个票价

JLabel lblPrice2 = new JLabel("和:");

txtMaxPrice = new JTextField(10);

txtMaxPrice.setEditable(true);

con.add(lblPrice2);

con.add(txtMaxPrice);

（接上一页：HSearchPrice类）

//提示信息

JLabel lblPrompt1 = new JLabel("输入注意事项：1，两个输入框");

JLabel lblPrompt2 = new JLabel("都只能输入大于零的浮点数。");

JLabel lblPrompt3 = new JLabel("其值只能在100到1800之间。");

JLabel lblPrompt4 = new JLabel(" ");

JLabel lblPrompt5 = new JLabel("第一个输入框的数值（价格）");

JLabel lblPrompt6 = new JLabel("必须小于第二个输入框的数值。");

con.add(lblPrompt1);

con.add(lblPrompt2);

con.add(lblPrompt3);

con.add(lblPrompt4);

con.add(lblPrompt5);

con.add(lblPrompt6);

//setModalityType(false);

JButton btnSearch = new JButton(strSearch);

btnSearch.addActionListener(this);

JButton btnCancel = new JButton(strCancel);

btnCancel.addActionListener(this);

con.add(btnSearch);

con.add(btnCancel);

con.setVisible(true);

con.setLayout(new FlowLayout(FlowLayout.LEFT));

con.setLayout(new GridLayout(7,2,25,40));//每行两个组件

con.setSize(200,100);

this.pack();

this.setLocationRelativeTo(null);

}

//"确定""取消"按钮的响应

public void actionPerformed(ActionEvent e)

{

if( e.getActionCommand()==strSearch)

{

boolean bResult = ValidPrice();

if( false==bResult )

JOptionPane.showMessageDialog(this,

"录入数据无法通过验证，请重新录入，或取消。",

"警告信息",JOptionPane.PLAIN\_MESSAGE);

else

{//有效的输入价格，显示查询结果。

setVisible(false);

dispose();

bDoOK = true;

}

}

else if( e.getActionCommand()==strCancel)

{

setVisible(false);

dispose();

bDoOK = false;

}//end of else if

}//end of actionPerformed

（接上一页：HSearchPrice类）

//验证机票价格是否输入正确。

//输入机票价格（允许小数，100<=价格<=1800 第一个价格必须小于第二个价格

//有效，返回true;否则，返回false.

private boolean ValidPrice()

{

boolean flag = true;

String str = txtMinPrice.getText();

float price = 0.0f;

try

{//txtMinPrice可能为空字符串或者其它非法浮点数字符串，所以要出错处理

price = Float.parseFloat(str);

}

catch(Exception e)

{

flag = false;

}

if( true==flag )

{

if (price>=100.0f && price<=1800.0f )

flag = true;

else //非法的价格

flag = false;

}

if( true==flag)

{

fMinPrice = price;

str = txtMaxPrice.getText();

try

{//txtMinPrice可能为空字符串或者其它非法浮点数字符串，所以要出错处理

price = Float.parseFloat(str);

}

catch(Exception e)

{

flag = false;

}

if( true==flag )

{

if( price>=100.0f && price<=1800.0f )

flag = true;

else

flag = false;

}

if( true==flag)

{

fMaxPrice = price;

if( fMinPrice<fMaxPrice )

flag = true;

else

flag = false;

}

}

return flag;

}

（接上一页：HSearchPrice类）

//获取查询票价的最小值

public float GetMinPrice()

{

return this.fMinPrice;

}

//获取查询票价的最大值

public float GetMaxPrice()

{

return this.fMaxPrice;

}

//如果有效查询，则返回true；

//否则返回false

public boolean GetValidSearch()

{

return bDoOK;

}

}

注意：总共有七个类。

# 实验：黑盒测试

实验名称：黑盒测试

1. 实验目的：

学习黑盒测试技术：等价类划分，边界值分析，因果图法等黑盒测试技术。

1. 实验原理：

（１）等价类划分

将输入数据域按有效的或无效的（也称合理的或不合理的）划分成若干个等价类，测试每个等价类的代表值就等于对该类其他值的测试。也就是说，如果从某个等价类中任选一个测试用例未发现程序错误，该类中其他测试用例也不会发现程序的错误。

（２）边界值分析

实践经验表明，程序往往在处理边界情况时发生错误。边界情况指输入等价类和输出等价类边界上的情况。

（３）错误推测

在测试程序时，人们可能根据经验或直觉推测程序中可能存在的各种错误，从而有针对性地编写检查这些错误的测试用例。错误推测法没有确定的步骤，凭经验进行。它的基本思想是列出程序中可能发生错误的情况，根据这些情况选择测试用例。

（４）因果图

因果图能有效地检测输入条件的各种组合可能会引起的错误。因果图的基本原理是通过画因果图，把用自然语言描述的功能说明转换为判定表，最后为判定表的每一列设计一个测试用例。

1. 实验环境

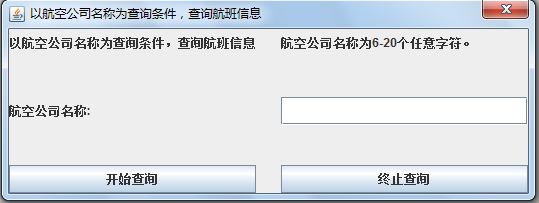
Eclipse 或者 MyEclipse

1. 实验内容与步骤
2. 从老师处拷贝工程BlackBoxTestSample，熟悉其代码及功能。
3. 老师讲解测试要求。
4. 编写测试用例，完成老师指定模块的测试，要求使用特定的黑盒测试技术。
5. 编写测试报告，上传到实验教学平台。

五、实验结果与分析。

BlackBoxTestSample实现了一个简单的航班管理程序，该软件可以增加一个航班记录，可以删除航班记录，所有航班记录保存在文件中。另外还提供了两个查询功能：以票价来查询航班，以航空公司名称来查询航班。

在主菜单选择“查询统计”—〉“以航空公司名称为条件检索”，出现如下所示的界面。



软件此部分的需求为：

在文本框输入航空公司名称，有效输入要求为6-20个任意字符。点击“终止查询”或者关闭对话框按钮时系统不判断输入是否有效，随后进入主界面。而点击“开始查询”时系统判断输入是否有效，如果无效，给出警示对话框，提示“录入数据无法通过验证”，如果有效，进入查询阶段。在查询阶段，如果当前航班记录中存在此航空公司的航班，全部显示出来，没有的话，提示“没有找到符合条件的航班”。

使用等价类划分方法测试该功能。

步骤1，划分等价类，并编号

|  |  |
| --- | --- |
| 有效等价类 | 无效等价类 |
| 文本框输入 | |
| 1, 字符数目6…20个 | 2, 字符数目少于6个  3, 字符数目多于20个 |
| 选择点击按钮 | |
| 4，点击“开始查询”  5，点击“终止查询”  6，点击关闭对话框按钮  右上角Ｘ |  |
| 输出查询结果 | |
| 7,输出一条航班记录  8,输出两条航班记录,(多于两条，更好的测试)。 | 9, 没有对应的航班记录 |

步骤２，为有效等价类设计测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 输入项 | 覆盖 | 预期输出 | 实际输出 |
| 1 | 航空公司名称输入Hr6234；  点击“开始查询”；（航班记录中存在一个） | 1,4,7 | 输出Hr6234的一条相关记录 | OK |
| 2 | 航空公司名称输入Hr6234；  点击“开始查询”；（航班记录中存在二个） | 1,4,8 | 输出Hr6234的两条相关记录 | OK |
| 3 | 航空公司名称输入Hr6234；  点击“终止查询” | 1,5 | 进入主界面 | OK |
| 4 | 航空公司名称输入Hr6234；  点击关闭对话框按钮 | 1,6 | 进入主界面 | OK |

步骤3，为无效等价类设计测试用例

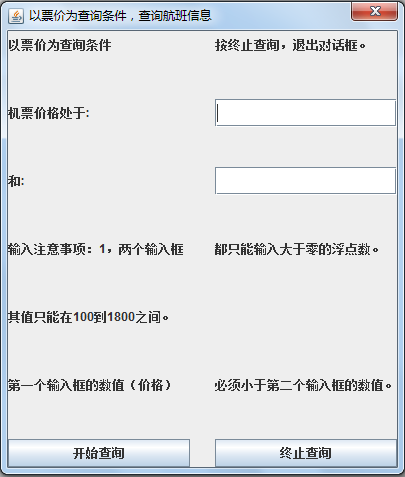
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 输入项 | 覆盖 | 预期输出 | 实际输出 |
| 5 | 航空公司名称输入hr234  点击“开始查询” | 2 | 提示“录入数据无法通过验证” | OK |
| 6 | 航空公司名称输入  h01234567890123456789  点击“开始查询” | 3 | 提示“录入数据无法通过验证” | OK |
| 7 | 航空公司名称输入hr8234；  点击“开始查询”；（航班记录中不存在对应的航班） | 9 | 提示“没有找到符合条件的航班” | OK |

步骤4，开始测试，并记录测试结果。

实际测试结果显示，所有测试用例都成功通过。

同学们动手做：

A，在主菜单选择“查询统计”—〉“以票价为条件检索”，出现如下所示的界面。



软件此部分的需求为：

在两个文本框输入机票价格，有效输入要求为数值处于100-1800的浮点数，并且第一个价格小于第二个价格。点击“终止查询”或者关闭对话框按钮时系统不判断输入是否有效，随后进入主界面。而点击“开始查询”时系统判断输入是否有效，如果无效，给出警示对话框，提示“录入数据无法通过验证”；如果有效，进入查询阶段。在查询阶段，如果当前航班记录中存在此价格区间的航班，全部显示出来，没有的话，提示“没有找到符合条件的航班”。

基本实训：使用等价类划分方法测试该功能。

B，在主菜单选择“查询统计”—〉“以票价为条件检索”，出现前述界面，需求同上。

基本实训：使用边界值分析方法测试该功能。

实训提高：结合边界值分析和等价类划分两种方法测试该功能。

C，在主菜单选择“文件”—〉“增加一个航班”，出现如下所示的界面。



软件此部分的需求为：

航班号要求3-8位数字字符；航班名称要求5-10位的数字及大写英文字符，小写英文字符会报错；航班名称，必须以H开头，后面接5-19个任意字符，比如：Hr4567；机票价格在100到1800之间，是浮点数；座位数目在100到200之间。点击“取消”或者关闭对话框按钮时系统不判断输入是否有效，随后进入主界面。而点击“确定”时系统判断输入是否有效，如果无效，给出警示对话框，提示“录入数据无法通过验证”；如果有效，进入主界面，在当前航班记录中添加此次输入的航班，并且在主窗口中显示所有航班记录。

基本实训：使用等价类划分方法测试该功能。

实训提高：结合边界值分析和等价类划分两种方法测试该功能。

# 实验：白盒测试

实验名称：白盒测试

1. 实验目的：

学习逻辑覆盖技术：判断（分支）覆盖，条件覆盖，条件组合覆盖。学习路径覆盖技术，学习循环测试技术

1. 实验原理：

判定覆盖。设计足够的测试用例，使得被测程序中每个判定表达式至少获得一次“真”值和“假”值，从而使程序的每一个分支至少都通过一次，因此判定覆盖也称为分支覆盖。

条件覆盖。设计足够的测试用例，使得判定表达式中每个条件的各种可能的值至少出现一次。

条件组合覆盖。比较强的覆盖标准，它是指设计足够的测试用例，使得每个判定表达式中条件的各种可能的值的组合都至少出现一次。

路径覆盖。设计足够的测试用例，覆盖被测程序中所有可能的路径。

循环测试原理，参考教材。

1. 实验环境

Eclipse 或者 MyEclipse

1. 实验内容与步骤
2. 从老师处拷贝工程WhiteBoxTestSample，熟悉其代码及功能。
3. 老师讲解测试要求。
4. 编写测试用例，完成老师指定模块的测试，要求使用特定的白盒测试技术。
5. 编写测试报告，上传到实验教学平台。

五、实验结果与分析。

WhiteBoxTestSample实现了一个简单的航班管理程序，该软件可以增加一个航班记录，可以删除航班记录，所有航班记录保存在文件中。另外还提供了两个查询功能：以票价来查询航班，以航空公司名称来查询航班。

整个工程包括如下类：

MainProcess，主程序。显示主窗口，创建程序的菜单。

HFlightInfo，单个航班记录。

HManaFlight，航班管理类，里面维护了一个数组，所有航班记录都在数组中。它是其它类和HFlightInfo打交道的中介。

HInputFlight，提供了一个对话框，当要添加航班时，此对话框提示输入航班的各项信息。对话框由Java.swing技术实现。

HMenuAction，菜单类。它是整个工程的中枢，大部分类都和它产生关联，这样减少类之间的耦合。当然，HMenuAction也可以合并到MainProcess。之所以没有这样做，是考虑到，若合并，合并后的类承担的东西太多，不好维护。所有菜单的响应都是在此类中。

HSearchPrice，以票价来查询航班。提供了一个对话框，用户可以输入票价的范围，点击查询后，软件显示满足查询条件的所有航班。

HSearchByCompanyName，以航空公司名称来查询航班。提供了一个对话框，用户可以输入航空公司名称，点击查询后，软件显示满足查询条件的所有航班。

讲解内容

在HSearchByCompanyName内找到下面的代码。

//航空公司名称为6-20个任意字符。

//当航空公司名称合法时，返回true；否则返回false.

//点击“开始查询”按钮时，调用此方法。如果方法返回false,则软件弹出告警信息；否则开始查询。

private boolean ValidCompanyName()

{

boolean flag = true;

String str = txtCompanyName.getText();

int len = str.length();

if (len>=6 || len<=20 )

{

flag = true;

m\_strCompany = str;

}

else //非法的航空公司名称

flag = false;

return flag;

}

我们使用判定覆盖技术测试ValidCompanyName 时，可设计如下测试用例：

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 输入项 | 预期输出 | 实际输出 |
| 1 | 航空公司名称输入Hty678 | flag =true  m\_strCompany=“ Hty678” | flag =true |
| 2 | 航空公司名称输入TY80 | flag =false  m\_strCompany的值不改变 | flag =true |

由此可知，第二个测试用例未通过，说明ValidCompanyName存在错误。

我们使用条件覆盖技术测试ValidCompanyName 时，可设计如下测试用例：

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 输入项 | 预期输出 | 实际输出 |
| 1 | 航空公司名称输入Hty8 | flag =false  m\_strCompany=“ Hty8” | flag =true |
| 2 | 航空公司名称输入TY8056 | flag = true  m\_strCompany=“TY8056” | flag =true |
| 3 | 航空公司名称输入  T01234567890123456789 | flag =false  m\_strCompany的值不改变 | flag =true |

由此可知，第一个和第三个测试用例未通过，说明ValidCompanyName存在错误。

错误原因，在if (len>=6 || len<=20 )中，||应该为&&。

同学们动手做：

1，在HSearchByCompanyName中，测试它的方法：

public void actionPerformed(ActionEvent e)

要求使用判定覆盖技术。

2，在HInputFlight中，测试它的方法：

private Boolean ValidCompany()

要求使用条件覆盖技术。

3，在HSearchPrice中，测试它的方法：

Private Boolean ValidPrice()

要求使用路径覆盖技术。注意：catch语句可看作条件语句。

4，在HManaFlight中，测试它的方法：

Public String SearchPriceRange(float fMin,float fMax)

要求使用循环测试技术。

# 实验：JUnit的安装和使用

1, JUnit的安装和使用 https://jingyan.baidu.com/article/380abd0a4d021a1d91192c40.html

2, JUnit官网：<http://junit.org/>

3, JUnit的jar包下载：<https://github.com/junit-team/junit/wiki/Download-and-Install>

4，JUnit4单元测试入门教程： http://www.jianshu.com/p/7088822e21a3

5, 单元测试利器 JUnit 4： https://www.ibm.com/developerworks/cn/java/j-lo-junit4/

JUnit是用于编写可复用测试集的简单框架，是xUnit的一个子集。xUnit是一套基于测试驱动开发的测试框架，有PythonUnit、CppUnit、JUnit等。Junit测试是程序员测试，即所谓白盒测试，因为程序员知道被测试的软件如何完成功能和完成什么样的功能。 多数Java的开发环境都已经集成了JUnit作为单元测试的工具，比如Eclipse。

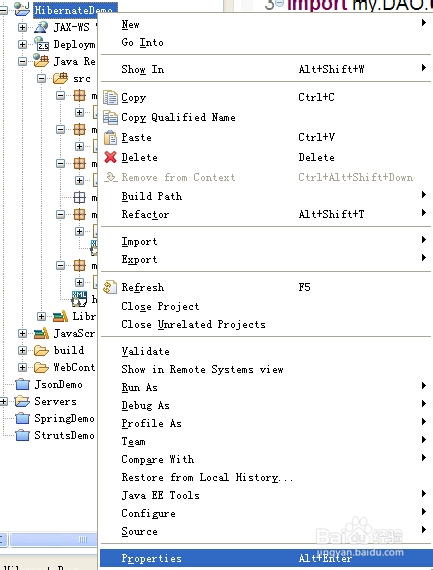
为什么要使用测试工具。测试框架可以帮助我们对编写的程序进行有目的地测试，帮助我们最大限度地避免代码中的bug，以保证系统的正确性和稳定性。 很多人对自己写的代码，测试时就简单写main，然后sysout输出控制台观察结果。这样非常枯燥繁琐，不规范。缺点：测试方法不能一起运行，测试结果要程序猿自己观察才可以判断程序逻辑是否正确。JUnit的断言机制，可以直接将我们的预期结果和程序运行的结果进行一个比对，确保对结果的可预知性。

**JUnit的安装**

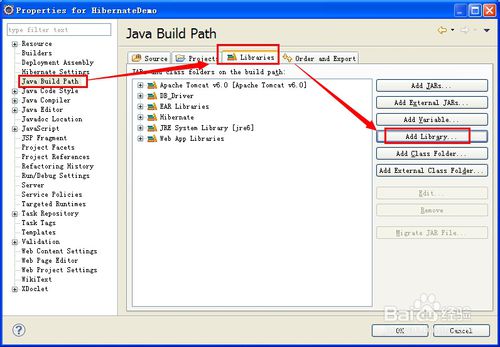
简单的说，Junit 是一个基于Java语言的回归单元测试框架。是白盒测试的一种技术。1

右键项目名称选择“Properties”，在弹出的窗体中选择“Java Build Path”-->选则“Libraries”选项卡-->“Add Library”,如下图所示：

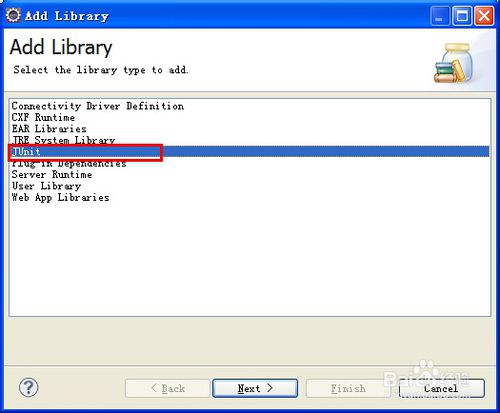
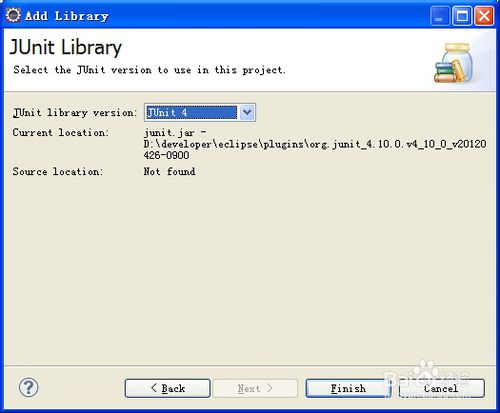
步骤阅读



步骤阅读

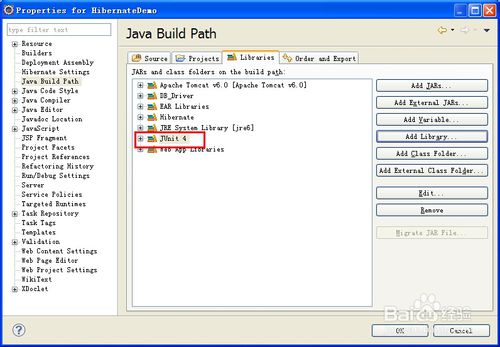


选择我们需要添加的“Junit”及版本号，添加成功确认即可。



步骤阅读

步骤阅读

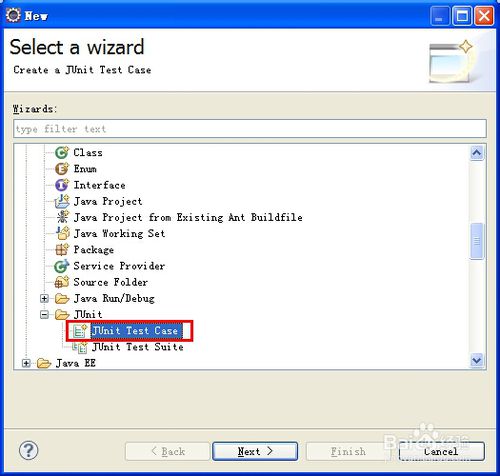
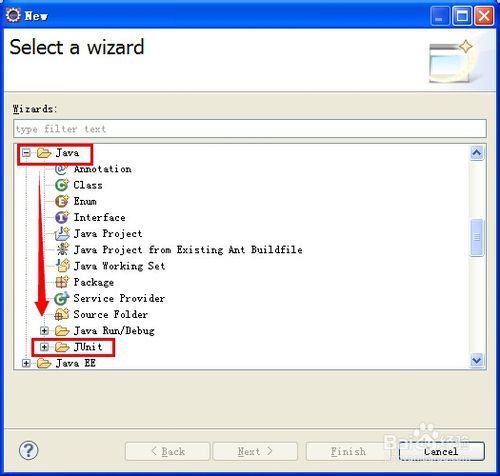


百度经验:jingyan.baidu.com

**JUnit的使用**1

选择项目右键“New”-->“Others”-->“Junit”(在Java目录下)-->“Junit Test Case”

步骤阅读



步骤阅读

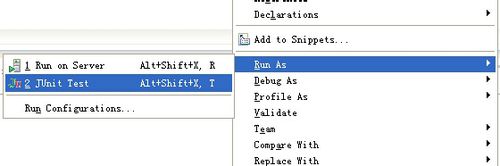
填写单元测试的名称，勾选“setUp” “tearDown”方法，选择被测试类以及被测试的方法，如图：

步骤阅读



1. 2

需要查看测试结果，只需要在Junit测试类上右键“Run as”-->“Junit Test”即可。下面就只需要观察控制台的输出信息看看有没有错。



实验名称：JUnit的安装和使用

1. 实验目的：

学习高级测试技术。

1. 实验原理：

JUnit是一个Java语言的单元测试框架。它由Kent Beck和Erich Gamma建立，逐渐成为源于Kent Beck的sUnit的xUnit家族中最为成功的一个。JUnit有它自己的JUnit扩展生态圈。多数Java的开发环境都已经集成了JUnit作为单元测试的工具。

JUnit是由 Erich Gamma 和 Kent Beck 编写的一个回归测试框架。Junit测试是程序员测试，即所谓白盒测试。Junit是一套框架，继承TestCase类，就可以用Junit进行自动测试了。

JUnit是一个开放源代码的Java测试框架，用于编写和运行可重复的测试。他是用于单元测试框架体系xUnit的一个实例（用于java语言）。它包括以下特性：1、用于测试期望结果的断言（Assertion）; 2、用于共享共同测试数据的测试工具; 3、用于方便的组织和运行测试的测试套件; 4、图形和文本的测试运行器。

1. 实验环境

Eclipse 或者 MyEclipse

1. 实验内容与步骤
2. 从老师处拷贝工程RmbMoney和FlightJUnitFestSwing，熟悉其代码及功能。
3. 老师讲解测试要求。
4. 编写测试用例，完成老师指定模块的测试，要求使用特定的白盒测试技术。
5. 编写测试报告，上传到实验教学平台。

五、实验结果与分析。

**工程RmbMoney代码**

Money.java

/\*一个简单的类，给学生介绍Junit的安装和使用.\*/

public class Money {

private float fAmount; //金额

private String strCurrency; //币种

//构造函数，何海江

public Money(float amount,String currency)

{

fAmount = amount;

strCurrency = currency;

}

public float GetAmount()

{

return fAmount;

}

public String GetCurrency()

{

return strCurrency;

}

//相加

public void AddAmount(float amount)

{

fAmount += amount;

}

//相乘

public void multiply(float factor)

{

fAmount \*= factor;

}

@Override

public String toString()

{

StringBuffer buffer = new StringBuffer();

buffer.append("["+GetAmount()+" "+GetCurrency()+"]");

return buffer.toString();

}

}

MainProc.java

public class MainProc {

public static void main(String[] args) {

// TODO Auto-generated method stub

Money mny = new Money(50,"RMB");

System.out.println(mny);

}

}

通过前面介绍的方法，为类Money添加JUnit测试代码，即测试类MoneyTest。勾选setup 和tearDown,在类MoneyTest中添加测试成员USD；并且添加AddAmount的测试方法，添加multiply的测试方法，添加toString的测试方法。于是，有了如下的MoneyTest代码。

MoneyTest.java

import static org.junit.Assert.\*;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import junit.framework.TestCase;

public class RmbTest {

private Money mnyUSD; //供后面的测试使用，在setUp总初始化

@Before

public void setUp() throws Exception {

mnyUSD = new Money(80,"USD");

}

@After

public void tearDown() throws Exception {

//此处可放置释放资源的代码。

}

@Test

public void test() {

//故意放置一个失败的测试用例。

fail("Not yet implemented");

}

//此测试用例：测试Money的方法AddAmount

//注意：有时候需要多个测试用例才能发现错误代码。

//整数的判断：TestCase.assertEquals(expected, actual);

//浮点数的判断：TestCase.assertEquals(expected, actual, delta);

@Test

**public** **void** testAddAmount()

{

mnyUSD.AddAmount(30);

TestCase.*assertEquals*(110, mnyUSD.GetAmount(),1e-5);

}

//此测试用例：测试Money的方法multiply

@Test

**public** **void** testMultiAmount()

{

mnyUSD.multiply(3);

TestCase.*assertEquals*(240, mnyUSD.GetAmount(),1e-5);

}

//此测试用例：测试Money的方法ToString

@Test

**public** **void** testToString()

{

TestCase.*assertEquals*("[80.0 USD]", mnyUSD.toString());

}

}

//此测试用例：测试Money的方法AddAmount

//注意：有时候需要多个测试用例才能发现错误代码。

//整数的判断：TestCase.assertEquals(expected, actual);

//浮点数的判断：TestCase.assertEquals(expected, actual, delta);

@Test

public void testAddAmount()

{

mnyUSD.AddAmount(30);

TestCase.assertEquals(110, mnyUSD.GetAmount(),1e-5);

}

//此测试用例：测试Money的方法multiply

@Test

public void testMultiAmount()

{

mnyUSD.multiply(3);

TestCase.assertEquals(240, mnyUSD.GetAmount(),1e-5);

}

//此测试用例：测试Money的方法ToString

@Test

public void testToString()

{

TestCase.assertEquals("[80.0 USD]", mnyUSD.toString());

}

}

特别注意整数的判断和浮点数的判断有差别。

整数的判断：TestCase.assertEquals(expected, actual);

浮点数的判断：TestCase.assertEquals(expected, actual, delta);

由于类Money的属性fAmount 是浮点数，上面的测试用例如果用代码TestCase.assertEquals(240, mnyUSD.GetAmount())，则达不到测试效果。

# 实验：FEST-Swing结合JUnit自动化测试

FEST-Swing是一个用于Swing GUI应用程序功能测试的java开源类库。支持模拟用户交互（鼠标操作和键盘输入）。支持JDK中的所有Swing组件。提供简洁、强大的API来创建和维护GUI功能测试。支持在HTML测试报告中嵌入GUI测试失败的截屏。能够与JUnit或TestNG一起使用。

首先到http://code.google.com/p/fest/downloads/list上面下载最新的FEST-Swing文件，一般文件名应该是fest-swing-x.x.zip。下载完成之后解压。 注意，需要将fest-swing-1.0.jar以及lib下面的所有文件都添加到工程路径下，同时需要添加JUnit支持。这里我们使用JUnit4，添加完成之后即可使用FEST-Swing了。

1, 使用FEST-Swing测试GUI http://devbean.blog.51cto.com/448512/126828/

2，FEST-Swing 简介信息 http://www.open-open.com/open205243.htm

需要注意的是，FEST-Swing是使用组件的name值来获取组件的，因此代码中setName方法的调用是必不可少的。因为不使用FEST-Swing，老师原来的程序都没有加setName。新建一个JUnit4 Test Case，使用FEST-Swing测试代码。首先需要有一个FrameFixture对象的属性。这里可以把FrameFixture理解成被测试的对象，因为我们想测试一个JFrame，所以使用FrameFixture。在FEST-Swing中，这些类与Swing的组件名字大体是一致的，只是后面多了一个Fixture。比如，JButton对应的类就是JButtonFixture。然后在@Before方法中对其进行实例化：

private FrameFixture frame;       
   @Before   
   public void setUp() {   
     frame = new FrameFixture(new MyFrame());   
      frame.show(); // 将frame显示出来   
   }

在@After方法中对其进行清理：

@After   
   public void tearDown() {   
     frame.cleanUp();   
   }

然后编写@Test方法：

@Test   
   public void testCopyTextToLabel() {   
      frame.textBox("input").enterText("Hello World!");   
     frame.button("copy").click();   
     frame.label("show").requireText("Hello World!");   
   }

实验名称：FEST-Swing结合JUnit自动化测试

1. 实验目的：

学习自动化测试框架FEST-Swing，图形界面测试工具。

1. 实验原理：

FEST-Swing 是一个 Java 库，为功能性 Swing GUI 测试提供 fluent interface。这个库提供了简单易用的 API，创建和维护GUI测试变得容易。

1. 实验环境

Eclipse 或者 MyEclipse

1. 实验内容与步骤
2. 从老师处拷贝工程FlightJUnitFestSwing，熟悉其代码及功能。拷贝文件夹Fest-Swing-JAR，其下有四个文件：fest-assert-1.4.jar， fest-reflect-1.4.1.jar， fest-swing-1.2.1.jar， fest-util-1.1.6.jar
3. 在工程FlightJUnitFestSwing，选择Project-> Properties->Java Build Path ->Libreares->Add External JARs，添加上述四个jar文件。另外，当提示添加JUnit时，同样需要添加。
4. 老师讲解测试要求。
5. 编写测试用例，完成老师指定模块的测试，要求使用特定的白盒测试技术。
6. 编写测试报告，上传到实验教学平台。

五、实验结果与分析。

拷贝工程FlightJUnitFestSwing。

1,从http://mvnrepository.com/artifact/org.easytesting/fest-swing下载

2，版本搭配有问题，目前的搭配如下：

fest-assert-1.4.jar

fest-reflect-1.4.1.jar

fest-swing-1.2.1.jar

fest-util-1.1.6.jar （搭配2.5就不行）

类HInputFlightTest专门为测试HInputFlight而写，目前能够测试座位的输入，也就是用条件覆盖技术测试 private boolean ValidSeat() ，它的需求为：

//验证座位数目是否输入正确。

//输入座位数目（整数，100到200之间）

//有效，返回true;否则，返回false.

HInputFlightTest.java完整内容如下：

import static org.junit.Assert.\*;

import org.fest.swing.fixture.DialogFixture;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

public class HInputFlightTest {

DialogFixture jfDlg; //测试对应的对话框。

HInputFlight hif; //被测对象，能够获取该对象的值。

@Before

public void setUp() throws Exception {

hif = new HInputFlight(null,"测试：输入航班的相关数据");

jfDlg = new DialogFixture( hif );

jfDlg.show();

}

@After

public void tearDown() throws Exception {

hif.dispose();

jfDlg.cleanUp();

}

@Test

public void test() {

fail("Not yet implemented");

}

//测试增加一个航班的座位数目

//使用的测试技术：条件覆盖

@Test

public void testSeatForAddFlight() {

/\*第一个测试用例，测试：座位数不能有0-9以外的字符，也就是

\* 在方法private boolean ValidSeat()内

\* 可以跳到catch(Exception e)里面。

\*注意：其它输入项必须正确，否则，如何测试ValidSeat

\*/

jfDlg.textBox("txtIDFest").setText("93687");

jfDlg.textBox("txtFnameFest").setText("YE4002");

jfDlg.textBox("txtCompanyFest").setText("Hguohang");

jfDlg.textBox("txtPriceFest").setText("800.5");

jfDlg.textBox("txtSeatFest").setText("126.0");

jfDlg.button("btnOKFest").click();

assertEquals(false,hif.GetValidFlight());

/\*第二个测试用例，测试：座位数不能小于100。

\*注意：其它输入项必须正确，否则，如何测试ValidSeat

\*/

jfDlg.textBox("txtIDFest").setText("93687");

jfDlg.textBox("txtFnameFest").setText("YE4002");

jfDlg.textBox("txtCompanyFest").setText("Hguohang");

jfDlg.textBox("txtPriceFest").setText("800.5");

jfDlg.textBox("txtSeatFest").setText("98");

jfDlg.button("btnOKFest").click();

assertEquals(false,hif.GetValidFlight());

/\*第三个测试用例，测试：座位数不能大于100。

\*注意：其它输入项必须正确，否则，如何测试ValidSeat

\*/

jfDlg.textBox("txtIDFest").setText("93687");

jfDlg.textBox("txtFnameFest").setText("YE4002");

jfDlg.textBox("txtCompanyFest").setText("Hguohang");

jfDlg.textBox("txtPriceFest").setText("800.5");

jfDlg.textBox("txtSeatFest").setText("298");

jfDlg.button("btnOKFest").click();

assertEquals(false,hif.GetValidFlight());

/\*

\* 何海江 for JUnit Fest-Swing GUI测试

\*/

/\*第四个测试用例，测试：座位数不能大于100。

\*注意：其它输入项必须正确，否则，如何测试ValidSeat

\*/

jfDlg.textBox("txtIDFest").setText("93687");

jfDlg.textBox("txtFnameFest").setText("YE4002");

jfDlg.textBox("txtCompanyFest").setText("Hguohang");

jfDlg.textBox("txtPriceFest").setText("800.5");

jfDlg.textBox("txtSeatFest").setText("128");

jfDlg.button("btnOKFest").click();

assertEquals(true,hif.GetValidFlight());

}

}

类HInputFlight里面需要添加这样的代码：txtID.setName("txtIDFest");这是为了JUnit Fest-swing的测试。后面就可以使用jfDlg.textBox("txtIDFest").setText("93687");

有两种方式运行JUnit测试。（1）在Eclipse中选择工程FlightJUnitFestSwing后，右键点击，在菜单中选择Run as；（2）已经执行过测试后，在JUnit界面里，选择要重新执行的测试方法（带@test的方法），右键点击，在菜单中选择Run。

当@Test

public void testSeatForAddFlight() {

执行完第二个测试用例时，会停下来。报告测试用例通不过，也就是代码有问题。通过检查代码，发现在类HInputFlight的方法：

private boolean ValidSeat()中，

语句 if( true==flag && seat>=100 && seat<=200 )

flag = true;

应该为 if( true==flag && seat>=100 && seat<=200 )

flag = true;

else

flag = false;

修改代码后，继续执行测试用例，通过。说明类HInputFlight的方法ValidSeat()的bug已经消除。

要求同学们完成：

（1）使用条件覆盖技术，编写方法测试类HInputFlight的方法ValidPrice；使用路径覆盖技术，编写方法测试ValidCompany。

（2）使用合适的白盒测试技术，编写类及方法，测试类HSearchPrice 。

（3）使用合适的白盒测试技术，编写类及方法，测试类HSearchByCompanyName。

# 实验：QTP/UFT的脚本录制

1. 实验目的：

学习自动测试工具。

1. 实验原理：

QuickTest Professional简称QTP，是一种自动测试工具。使用QTP，用它来执行重复的手动测试，用于回归测试和测试同一软件的新版本。因此在测试前要考虑好如何对应用程序进行测试，例如要测试那些功能、操作步骤、输入数据和期望的输出数据等。Unified Functional Testing简称UFT。

1. 实验环境

Eclipse 或者 MyEclipse

1. 实验内容与步骤
2. 从老师处拷贝工程BlackBoxTestSample，熟悉其功能。BlackBoxTestSample打包成可执行的JAR文件, 类HManaFlight的方法GetFlightInfo从未被使用，属于警告信息，可将该方法注释掉。打包步骤：在Eclipse中，选择工程后，点击右键，选择Export->Java->Runable JAR file，确定各参数项，输入可执行JAR文件名，如bbst.jar。生成后，在资源管理器点击bbst.jar，可脱离Eclipse环境运行该程序。

**从官方网站下载软件：https://saas.hpe.com/en-us/products。UFT试用版，要求注册。安装时，可能要求操作系统打补丁。**

1. 老师讲解测试要求。
2. 编写测试用例，完成老师指定模块的测试，要求使用特定的黑盒测试技术。
3. 编写测试报告，上传到实验教学平台。

五、实验结果与分析。

1，QuickTest Professional，百度百科，

http://baike.baidu.com/link?url=tsmSh1z7PEBLeRpLKcLsi0N4VYekNGrOtkaUFRoFzWgXK9tnX5nGSSwNdECEEUkxWrosLHYk3pfA3UbIEpOWqp\_X3V-WWS1jPIKbbywRUSqkmvseBTbQ\_mtRaHk3tuF0

2，HP QuickTestProfessional（QTP，UFT）功能测试软件

http://www.sinocax.com/fuwuxiangmu/HPchanpin/83.html

3，UFT工具简介和使用

http://blog.csdn.net/ws0513/article/details/49226119

4， 【自动化测试工具】QTP/UFT入门

http://www.51testing.com/html/83/23583-3569614.html

**UFT简介**

UFT是一种自动化测试工具，以VBScirpt为内嵌语言。

UFT支持功能测试和回归测试自动化，可用于软件应用程序和环境的测试。UFT自动化测试的基本功能包括：

①创建测试

②检验数据

③增强测试

④运行测试脚本

⑤分析测试结果

⑥维护测试

UFT支持两种视图，一种是Keyword View(关键字视图)，另一种是Expert View(专家视图)。Keyword View(关键字视图)：是一种图形化的视图。

Expert View(专家视图)：对于关键字视图中的每个节点，在专家视图中都对应一行脚本。也叫脚本视图。测试对象

UFT中的对象有两个概念，一个是测试对象，一个是运行时对象。

①测试对象（Test Object，TO)：是UFT定义的一些类，用他们代表被测应用的各种对象。(对象库中的对象)

②运行时对象（Runtime Object,RO）：是实际的被测应用对象，是测试执行过程中，TO用来关联的对象。（运行时识别的对象）

**对象的属性**

1)To属性：获取的是对象的属性，可以理解为静态的属性，在运行时的输入东西与它无关。

测试对象的属性是UFT为了识别在测试执行过程中的运行时对象而保留在对象库中的属性。GetToProperty/GetToProperties和SetToProperty则分别用于读取和修改To的属性值。

GetToProperty用于取得测试对象的某个属性的值，

GetToProperties用于取得测试对象的所有属性的值，

SetToProperty用于设置测试对象的某个属性的值。

2) Ro属性：是变的（用户名框输入不同的值，这个Ro属性取出来就是不一样）

GetRoProperty方法用于取得运行时对象（实际对象）的某个属性的值。

**UFT对象库**

UFT用对象库对对象进行管理，记录的对象和这些对象的属性都存储在对象存储库中。

下面介绍使用Object Repository完成一些基本的功能操作.

HP QuickTestProfessional（QTP，UFT）

HP QuickTest Professional为每一个重要软件应用和环境提供功能和回归测试自动化的行业最佳解决方案，包括如WebServices、MacromediaFlex、.NET、J2EE和EPR/CRM应用等下一代开发技术。HPQuickTestProfessional为自动化测试提供了一种全新的方法，它使用关键词驱动（Keyword-Driven）的测试法，从而大大简化了测试的创建和维护工作。这种测试方法的独特之处是，测试者可以使用先进的截取技术（录音/重放）直接从应用屏幕上获取流程，从而选择建立什么样的测试实例。另外，测试自动化专家可以通过一个集成的脚本调试环境，拥有对基础测试脚本和对象属性的完全访问权限，这些脚本和调试环境与关键词视图（KeywordView）可以互为同步。有了该产品，您的QA机构就可以：

● 以最少的培训令整个小组具备进行复杂测试的能力；

● 确保跨所有环境、数据包和业务流程的正确功能点；

● 为开发人员全面记录和复制缺陷，使他们能更快地修复缺陷，满足最后上线期限；

● 对不断变化的应用和环境，轻松实施回归测试；

● 帮助整个机构实现高质量产品和服务、提高总收入。

工作方式

传统的脚本记录工具所生产的脚本不易修改。与此不同的是，QuickTestProfessional的关键词驱动方式能让您便捷地插入、修改、数据驱动和移除测试步骤。HPQuickTestProfessional可以处理新应用的构建。当一种测试中的应用发生改变，比如“Login”按钮被重新命名为“SignIn”，您可以让一个电脑档案更新变成以XML为基础的SharedObjectRepository（在新的ObjectRepositoryManager那里），这个更新就会出现在所有指向这个对象的测试中。您可以向惠普质量管理发布您的测试脚本，让其他QA小组成员再次使用这个脚本，从而避免重复工作。HPQuickTestProfessional支持所有企业环境的功能测试。

为小组合作而设计

HP QuickTest Professional 9为测试工作组之间的测试自动化合作而设计。它包括一项管理在新的Object Respository Manager里的应用定义或目标的新技术。Object Respository Manager使用公开的XML报告格式，让QuickTest Professional的使用者相互合作并分享应用目标定义。QuickTest Professional通过测试创建的努力让测试者之间保持目标水平的同步变化。并且，在质量保证部门得到应用之前，就能够确定存储库。另外，用户可以通过测试工作组分享功能库、应用资产定义和数据驱动的电子数据表程序。

惠普功能测试

惠普功能测试（HP Functional Testing）将HP QuickTest Professional和HP WinRunner结合成一种集成产品。功能测试是WinRunner和QuickTest Profes-sional的集成。在功能测试中，产品间可以相互调用脚本，测试结果可以在一个共有的报告界面上呈现。

惠普业务流程测试

惠普业务流程测试（HP Business Process Testing）是功能测试的一种完整的系统，既适合于自动测试，也适合于手动测试。业务流程测试使非技术性主题专家成为质量优化流程不可分割的一部分，自动化测试计划文档的创建并使应用变化的测试维护流水线化。惠普业务流程测试降低了自动化测试的维护费用并把测试自动化与文档连接到一起。它使得主题专家和业务分析家以业务流程测试框架下的抽象的业务定义标准来衡量应用交付的质量。通过业务分析家和质量工程师的有效合作，惠普业务流程测试也能提高生产力。用这种基于角色的方法，主题专家集中精力创建高水平的测试流程来反映实际测试流程，而质量工程师则专心于自动化的工作。

惠普质量中心的组成部分

HPQuickTestProfessional是惠普质量中心（HPQual-ityCenter）的组成部分之一，惠普质量中心集成了一整套软件、服务和最佳实践，用于自动化关键质量活动，包括需求管理、测试管理、缺陷管理、功能测试和业务流程测试。

特点和优势

● 具有行业领先的便于使用的特性，以及支持提前配置环境的功能，确保了快速的投资回报(ROI)。

● 可独立运行，也可以同惠普业务流程测试和惠普质量中心集成。

● 确保工作组之间相互协作，分享惠普质量中心的功能库、强大的目标管理和灵活的资产储存。

● 引进了新一代的“零配置”关键词驱动测试技术，从而实现了快速建立测试、测试脚本更易维护，和更强大的数据驱动能力。

● 在目标与UniqueSmartObjectRecognition之间建立联系，即使目标的构件不断变化，也能保证确定的却未被注意的脚本的执行。

● 在RecoveryManager的帮助下处理事先未预见的应用事件，简化一周24小时的测试工作以满足测试工程最后上线日期。

● 利用自动化文档技术把测试文档和测试创建合并成为一个步骤。

● 通过集成的数据表（DataTable），可数据驱动任意对象、方式、检查点和输出值等。

● 为QA工程师提供一个强大的高设置的IDE环境。

● 通过平衡HPQuickTestProfessional和WinRunner集成的TSL资产，把您的投资保留在惠普WinRunner的测试脚本中。

● 快速隔离并诊断缺陷，用一体化报告形式写出，并以XML和HTML的形式输出。

● 通过全面执行检查点获得完全合法的应用软件。

● 为不同语言的应用测试提供统一代码支持。

以下的实验指导步骤参考了HP Unified Functional Testing 的《GUI 测试教程》。

运行测试时， UFT 模拟人类用户， 在网页或应用程序窗口中移动光标、单击GUI 对象并执行键盘输入。但是， UFT 的操作速度比任何人类用户都要快。

自动测试的优点

快速 自动测试的速度比人类用户快得多。

可靠 每次运行测试时， 测试将执行完全相同的操作， 从而消除人为错误。

可重复 可以测试网站或应用程序在重复执行相同操作之后如何反应。

可编程 可通过编程设置显示隐藏信息的复杂测试。

全面 可以构建涵盖网站或应用程序中每个功能的一整套测试。

可重用 即使用户界面更改， 也可以在网站或应用程序的不同版本上重用测试。

如果安装了ALM， 则可以将发现的缺陷报告给数据库。ALM 是HP 测试管理解决方案。有关使用ALM 的详细信息， 请参见《HP Application Lifecycle Management 用户指南》。

1，《GUI 测试教程》中有Mercury Tours 网站的测试举例，学习该部分内容，学习Web页面的测试。

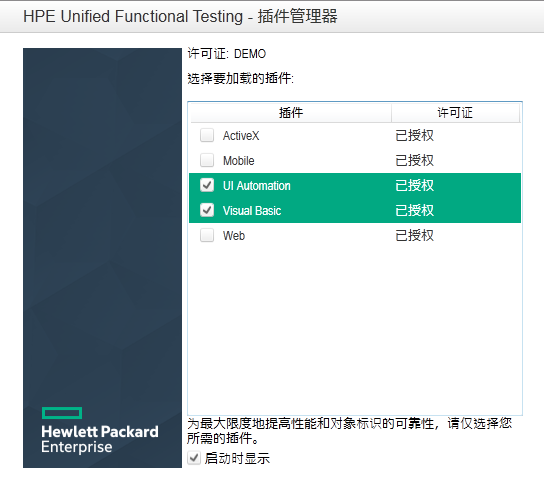
特别注意：确保在UFT 之后打开新的Internet Explorer 窗口。如果使用在打开UFT之前已打开的浏览器， 则UFT 可能无法正确识别该浏览器中的对象。

2，测试BlackBoxTestSample，已经打包成bbtsexe.jar。

下面是一个简单的例子，测试以票价为条件检索的对话框测试。具体步骤包括：

（1）测试了解应用程序 在开始创建测试之前，必须确定应用程序中要测试的确切内容。要这样做，需要根据业务流程(客户在应用程序中为完成特定任务而执行的不同活动)对应用程序进行分析。打开bbst.jar并稍作浏览，熟悉“以票价为查询条件，查询航班”的功能，输入输出情况。

（2）启动UFT。



注意：去掉Web插件，此处不需要。

（3）新建测试。单击新建按钮。默认情况下， UFT 假定您要新建测试，并将打开“新建测试”对话框。选择GUI 测试， 如下所示填充字段：

名称： 输入SearchPirceTest。

位置： 测试会自动保存到C:\%HOMEPATH%\My Documents\Unified

Functional Testing，选择一个自己建立的测试文件目录。

解决方案名称： 输入LearnUFTgui。

选择创建解决方案目录。单击创建。将打开一个空白测试，且测试流显示在画布中，空白操作选项卡在背景中打开。测试名称和路径显示在UFT 主窗口的标题栏中。在解决方案浏览器中，可以看到测试作为LearnUFTgui解决方案的一部分进行加载。

（4）重命名Action1， 以使它拥有逻辑名称。在画布中， 右键单击Action1 图像，然后选择操作属性。在“操作属性”对话框的名称框中，输入InputPrice 作为新名称。在打开的消息中，则单击是。

(5)除了InputPrice外，还可以新建其他操作。在画布的任意位置右键单击，然后选择调用新操作。将打开“插入对新操作的调用”对话框。在名称框中输入（同学们自己命名）， 并确认选择了可重用操作和位于测试结尾。单击确定。将关闭对话框，（同学们自己命名）操作将添加到测试结尾处。

(6）保存测试。单击保存按钮。浏览画布中的测试流。单击文档窗格中的对应选项卡以查看画布中的整个测试流。可以使用鼠标或画布上方的缩放栏放大或缩小测试流。当测试流放大到最大且超出画布范围时，也可以使用画布左下角的微型图浏览测试流。

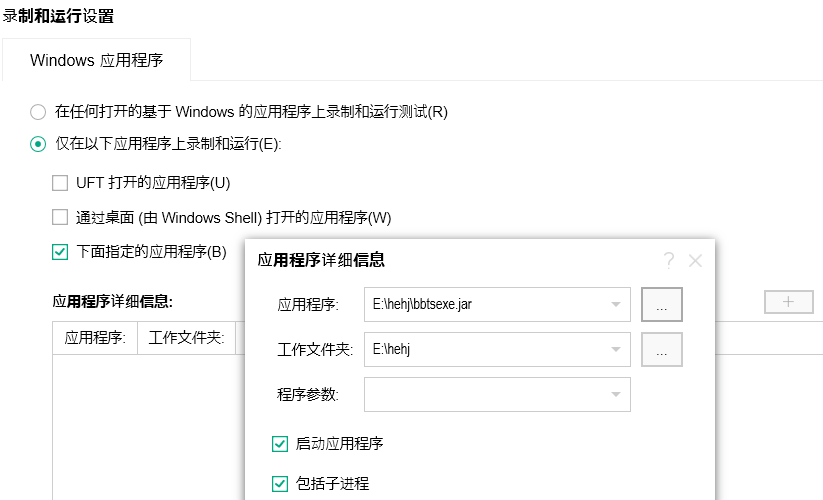
操作调用类型 可以将以下类型的操作调用插入到测试中：

调用新操作。在测试中插入空的新操作。

调用操作副本。在测试中插入现有操作的副本。操作不链接到复制操作的源测试或操作。

调用现有操作。在测试中插入对现有操作的调用。

(7)选择InputPrice栏，选择录制> 录制和运行设置。将打开“录制和运行设置”对话框。选择“仅在以下应用程序上录制和运行”->下面指定的应用程序->+按钮->选择bbtsexe.jar，出现如下图。



(8)在打包程序，按照输入要求录入数据。录入结束后，点击结束按钮。

(9)录入的动作，观察代码，可修改代码。可以重新运行。如果运行过程中，产生警告详细信息，如：

警告 - 步骤

以票价为查询条件，查询航班信息.Type

…

可手动点击“以票价为条件检索”菜单。

# 实验：QTP/UFT自动化测试进阶

1. 实验目的：

学习自动化测试框架FEST-Swing，图形界面测试工具。

1. 实验原理：

UFT文件组织结构说明

1. Autotest文件夹,整个工程的最高一级目录，名称可以修改。

2. driver文件夹,这个是整个框架的入口，内有Driver.vbs驱动程序、wyz.vbs辅助程序和UFT测试项目文件夹

3. testpro文件夹,用于记录有哪些项目，是否执行

4. testdata文件夹，用于设计测试用例，给testscript提供数据、期望值等信息

5. testscript文件夹，存放测试脚本，全部存储为vbs文件。类名对应项目名，对应文件名，一个函数对应一个用例（需和testdata中信息对应），也可添加其他公共函数

6. library文件夹，按项目名称分文件夹放置的对象库文件

注：testdata和testscript目录下的内容，是真正需要开发的。

1. 实验环境

Eclipse 或者 MyEclipse

1. 实验内容与步骤
2. 从老师处拷贝工程BlackBoxTestSample，熟悉其代码及功能。
3. 老师讲解测试要求。
4. 编写测试用例，完成老师指定模块的测试，要求使用特定的白盒测试技术。
5. 编写测试报告，上传到实验教学平台。

五、实验结果与分析。

《HP Unified Functional Testing 用户指南》描述了如何使用UFT 测试

应用程序。它逐步详细地说明了如何创建、调试和运行测试以及如

何报告测试过程中检测到的缺陷。

# 实验彩蛋

白盒测试实验中(何海江)

1，在HSearchByCompanyName中，测试它的方法：

public void actionPerformed(ActionEvent e)

要求使用判定覆盖技术。（代码没有问题）

2，在HInputFlight中，测试它的方法：

private boolean ValidCompany()

要求使用条件覆盖技术。

if( ch=='H')

flag = false;

正确代码应该为：

if( ch！='H')

flag = false;

3，在HSearchPrice中，测试它的方法：

Private boolean ValidPrice()

要求使用路径覆盖技术。

str = txtMaxPrice.getText();

try

{//txtMinPrice可能为空字符串或者其它非法浮点数字符串，所以要出错处理

price = Float.parseFloat(str);

}

catch(Exception e)

{

flag = true;

}

此段代码内，正确的语句flag=false。

4，在HManaFlight中，测试它的方法：

public String SearchPriceRange(float fMin,float fMax)

要求使用循环测试技术。

在代码HFlightInfo hfiFlight=(HFlightInfo)m\_listFlight.get(i+1); 中，i+1应该是i。

黑盒测试实验中(何海江)，

A和B测试同一个功能。等价类划分可能无法找到错误，而边界值分析可找到错误。在第二个文本框中，若输入1800，软件无法通过有效性验证。

C有三处错误，第一个，当机票价格是浮点数时，软件无法通过有效性验证；第二个，当航空公司以小些h开头，软件仍然通过有效性验证；第三个，当航班名称输入小写字符时，软件仍然通过有效性验证。

程序的调试(Eclipse or MyEclipse) 实验中(何海江)，

综合调试实验

第一个错误的位置：

**private** **boolean** ValidCompany()

{

**boolean** flag = **true**;

String str = txtCompany.getText();

**int** len = str.length();

**if**( len<6 || len>20 )

flag = **false**;

**else**

{

**char** ch = str.charAt(1); //此处正确代码为charAt(0)

**if**( ch!='H')

flag = **false**;

}

**return** flag;

}

第二个错误的位置：

private boolean ReadFlightFile()

{

boolean bFlag = true;

m\_listFlight.clear();//清除旧的

try{

DataInputStream fm = new DataInputStream(new BufferedInputStream(

new FileInputStream("FlightInfo.txt")));

//航班个数

int number = fm.readInt();

for( int i=1;i<number;i++ ) //此处正确代码为int i=0;

{

String strID = fm.readUTF();//航班号

String strName = fm.readUTF();//航班名称

String strCompany = fm.readUTF();//航空公司名称

float fPrice = fm.readFloat();//价格

int nSeat = fm.readInt();//座位数目

AddFlight(strID,strName,strCompany,fPrice,nSeat);

}

fm.close();

}//end of for...

catch(IOException e)

{

//System.err.println("文件FlightInfo.txt读错误。"+e.toString());

JOptionPane.showMessageDialog(null,

"<html><body><h1>文件FlightInfo.txt读错误。</h1><h3>检查该文件是否存在</h3></body></html>",

"报警信息",JOptionPane.WARNING\_MESSAGE);

bFlag = false;

}

return bFlag;

}

第三个错误的位置：

public String SearchPriceRange(float fMin,float fMax)

{

int nSize = m\_listFlight.size();

StringBuffer sbInfo = new StringBuffer();

// 遍历整个ArrayList

for(int i=0;i<nSize;i++)

{

// 取出HFlightInfo对象

HFlightInfo hfiFlight=(HFlightInfo)m\_listFlight.get(i);

String strTmp;

if( hfiFlight.getSeat()>=fMin && hfiFlight.getSeat()<=fMax )

{ //此处getSeat应该修改为getfPrice。

strTmp = hfiFlight.GetInfo();

sbInfo.append(strTmp+"\r\n");

}

} //end of for

return sbInfo.toString();

}

第四个错误的位置：

public boolean DelCompanyFlight(String strName)

{

boolean flag = false;

int nSize = m\_listFlight.size();

for(int i=0;i<nSize;i++)

{

HFlightInfo hfiFlight=(HFlightInfo)m\_listFlight.get(i);

if(hfiFlight.getStrCompany().equals(strName))

{

m\_listFlight.remove(i);

flag = true;

break; //此处的break应该去掉，这样做，使得后面的航班都能删除。

}

}//end of for

return flag;

}

实验：FEST-Swing结合JUnit自动化测试实验中(何海江)，

(1)类HSearchPrice的方法

private boolean ValidPrice()

其代码 boolean flag = true;

不要修改。

(2) 类HSearchByCompanyName的方法

private boolean ValidCompanyName()

其代码boolean flag = true;

不要修改。

(3) 类HInputFlight的方法

private boolean ValidPrice()

其代码 if( (true==flag) && (price>=100.0f) && (price<=1800.0f) )

flag = true;

应该修改为：if( (true==flag) && (price>=100.0f) && (price<=1800.0f) )

flag = true;

else

flag = false;

另外，如方法ValidFname()、ValidID()、ValidCompany()等也可能修改修改。

(4)方法

public void actionPerformed(ActionEvent e)

必须测试。