

**软件测试课程设计报告**

学 号： 20171000322

班级序号： 111172

学生姓名： 江悦

指导教师： 张剑波

**中国地质大学信息工程学院软件工程系**

**2020年 3 月**

# 1、动态测试技术

## 1. 软件说明

**程序名称**：《国航国内/国际航班行李托运计算程序》

**程序介绍**：本程序可以根据你携带的行李数量，行李的尺寸和重量，机票以及VIP种类计算托运的费用。

**程序说明**：

程序组成：使用B/S架构设计，前端使用html/css/js(jquery)设计，服务器使用golang。

程序使用：（本地测试）首先在浏览器打开localhost:5500/index.html, 把code文件夹下的server放入golang的环境配置目录下，请先打开浏览器的主页面，再运行服务器（因为服务器兼听5500端口，浏览器这时候可能无法访问，所以要先打开页面，在启动服务器）（部署到服务器后，才可以远程连接）

程序设计：

* 服务器分两个包，一个是bean包，一个是main包

* Bean包下设计
  + baerJsonBean结构体用于http.ResponseWriter的返回

|  |
| --- |
| // BaseJSONBean ：定义一个结构体，储存get的数据  type BaseJSONBean struct {  Code int `json:"code"`  Data interface{} `json:"data"`  Message string `json:"message"`  } |

* + ticket结构体用于接受国航的行李信息

|  |
| --- |
| // 国内航班行李票价计算结构体  type ticket struct {  ft FlightType  pt PassengersType  ct ClassType  hc HasCard  num Number  tfc int64  fp int64  lc int64  x []int64  y []int64  z []int64  weight []int64  } |

* + outticket结构体用于接受国际的行李信息

|  |
| --- |
| // 国际航班行李票价计算结构体  type outticket struct {  ft FlightType  pt PassengersType  ct ClassType  ota OriginTargetArea  num Number  tfc int64  fp int64  lc int64  x []int64  y []int64  z []int64  weight []int64  } |

* + 定义航班类型

|  |
| --- |
| // FlightType 定义航班类型  type FlightType int64  const (  // 不能使用下划线命名法，使用驼峰命名法  demosticType FlightType = 0  internationalType FlightType = 1  ) |

* + 定义乘客类型

|  |
| --- |
| // PassengersType 定义乘客类型  type PassengersType int64  const (  // 不能使用下划线命名法，使用驼峰命名法  adultChild PassengersType = 1  baby PassengersType = 2  ) |

* + 定义机舱类型

|  |
| --- |
| // ClassType 舱位  type ClassType int64  const (  // 不能使用下划线命名法，使用驼峰命名法  firstClass ClassType = 3  secendClass ClassType = 2  thirdClass ClassType = 1  ) |

* + 定义是否有会员卡类型

|  |
| --- |
| // HasCard :是否有会员卡  type HasCard int64  const (  // 不能使用下划线命名法，使用驼峰命名法  noCard HasCard = 1  platinumGoldSliverCard HasCard = 2  goldSliverCard HasCard = 3  starGoldCard HasCard = 4  ) |

* + 定义国际航线区域类型

|  |
| --- |
| // OriginTargetArea ：定义国际飞行区域的结构体  type OriginTargetArea int64  const (  origin1 OriginTargetArea = 1  origin2 OriginTargetArea = 2  origin3 OriginTargetArea = 3  origin4 OriginTargetArea = 4  origin5 OriginTargetArea = 5  ) |

* + 定义飞行次数类型

|  |
| --- |
| // Number : 乘坐次数  type Number string  const (  justGo Number = "go"  goBack Number = "goAndBack"  many Number = "many"  ) |

* + 定义国内和国外行李价格计算函数

|  |
| --- |
| func (t \*ticket) Calclate1() (ticketPrice int64) {}  func (t \*ticket) Calclate2() (ticketPrice int64) {} |

* Main包
  + 设计有启动函数

|  |
| --- |
| func main() {} |

* + 国内航班和国际航班路由函数

|  |
| --- |
| func routing1(w http.ResponseWriter, req \*http.Request) {}  func routing2(w http.ResponseWriter, req \*http.Request) {} |

* + 路由函数负责处理get请求的数据，并就爱那个数格式化
* 主程序设计

图表 1主程序设计

* 主程序功能设计

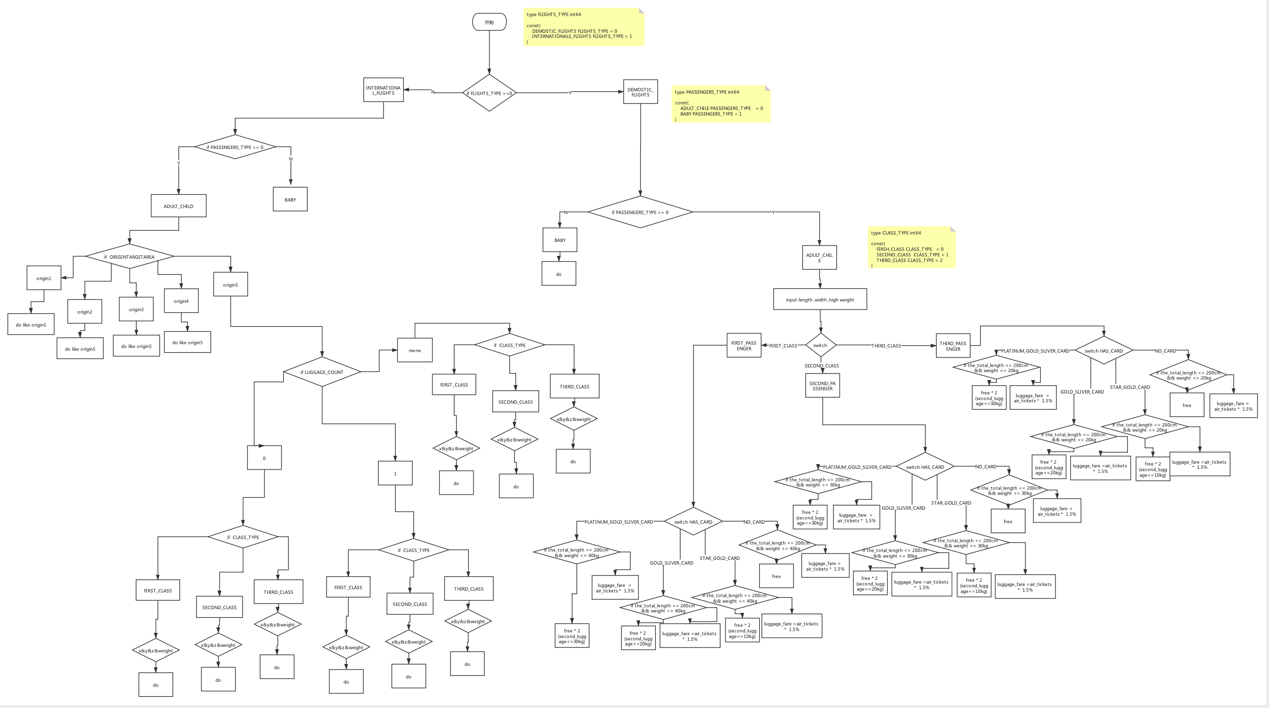
图表 2主程序功能设计

* 操作设计
  + 首先打开浏览器的主界面
  + 然后开启服务器兼听5500端口
  + 在浏览器上选择航班类型
  + 输入乘客信息
  + 输入行李信息，点击ok
  + 最后点击计算，即可计算行李价格



## 2. 测试内容

### 2.1流程图



### 2.2 思维导图

见 其他附件/航空包裹托运价格计算脑图.xmind

## 3. 测试用例设计说明

### 3.1 用例1

### 3.1.1 控制

国内航空无卡头等舱最大携带行李免费

|  |
| --- |
| ft:= 0  pt := 1  ct := 1  hc := 1  var num Number = justGo  var tfc int64= 1  var fp int64 = 1000  var lc int64 = 1  x := []int64{100}  y := []int64{60}  z := []int64{40}  weight := []int64{40} |

### 3.1.2 输入

|  |
| --- |
| ticket:= NewTicket(FlightType(ft), PassengersType(pt),ClassType(ct), HasCard(hc), num, tfc, fp, lc, x, y, z, weight) |

### 3.1.3 输出

|  |
| --- |
| Running tool: /usr/local/go/bin/go test -timeout 30s -coverprofile=/var/folders/t7/strgy24d7f53xh2mc3g2ttqc0000gn/T/vscode-go6fM4h8/go-code-cover go\_code/project22/bean  ok go\_code/project22/bean 0.012s coverage: 1.9% of statements  Success: Tests passed. |

### 3.1.4 过程

|  |
| --- |
| 行李数为 1  第一件行李尺寸允许  国内航班  成人票  经济舱  没卡  重量没超  map[price:0] |

### 3.2 用例2

### 3.2.1 控制

|  |
| --- |
| ft:= 0  pt := 1  ct := 3  hc := 1  var num Number = justGo  var tfc int64= 1  var fp int64 = 1000  var lc int64 = 1  x := []int64{100}  y := []int64{60}  z := []int64{40}  weight := []int64{20} |

### 3.2.2 输入

|  |
| --- |
| ticket := NewTicket(FlightType(ft), PassengersType(pt),ClassType(ct), HasCard(hc), num, tfc, fp, lc, x, y, z, weight) |

### 3.2.3 输出

|  |
| --- |
| Running tool: /usr/local/go/bin/go test -timeout 30s go\_code/project22/bean -run ^(TestPrice2)$  ok go\_code/project22/bean 0.007s  Success: Tests passed.  //ft:0 pt:1 ct:3 hc:1 num:go tcf:1 fp:1000 lc:1 x:[100] y:[60] z:[40] weight:[40] |

### 3.2.4 过程

|  |
| --- |
| 行李数为 1  第一件行李尺寸允许  国内航班  成人票  头等舱  重量没超  map[price:0] |

### 3.3 用例3

### 3.3.1 控制

测试婴儿带行李

|  |
| --- |
| ft:= 0  pt := 2  ct := 1  hc := 1  var num Number = justGo  var tfc int64= 1  var fp int64 = 1000  var lc int64 = 1  x := []int64{100}  y := []int64{60}  z := []int64{40}  weight := []int64{40} |

### 3.3.2 输入

|  |
| --- |
| ticket:= NewTicket(FlightType(ft), PassengersType(pt),ClassType(ct), HasCard(hc), num, tfc, fp, lc, x, y, z, weight) |

### 3.2.3 输出

|  |
| --- |
| Running tool: /usr/local/go/bin/go test -timeout 30s go\_code/project22/bean -run ^(TestPrice3)$  ok go\_code/project22/bean 0.009s  Success: Tests passed.  //ft:0 pt:2 ct:1 hc:1 num:go tcf:1 fp:1000 lc:1 x:[100] y:[60] z:[40] weight:[40] |

### 3.3.4 过程

|  |
| --- |
| 行李数为 1  第一件行李尺寸允许  国内航班  婴儿票  一个行李  重量超了  map[price:600] |

### 3.4 用例4

### 3.4.1 控制

测试经济舱超长度

|  |
| --- |
| ft:= 0  pt := 1  ct := 1  hc := 1  var num Number = justGo  var tfc int64= 1  var fp int64 = 1000  var lc int64 = 1  x := []int64{120}  y := []int64{60}  z := []int64{40}  weight := []int64{40} |

### 3.4.2 输入

|  |
| --- |
| ticket:= NewTicket(FlightType(ft), PassengersType(pt),ClassType(ct), HasCard(hc), num, tfc, fp, lc, x, y, z, weight) |

### 3.4.3 输出

|  |
| --- |
| Running tool: /usr/local/go/bin/go test -timeout 30s go\_code/project22/bean -run ^(TestPrice6)$  ok go\_code/project22/bean 0.009s  Success: Tests passed.  //ft:0 pt:1 ct:1 hc:1 num:go tcf:1 fp:1000 lc:1 x:[120] y:[60] z:[40] weight:[40] |

### 3.4.4 过程

|  |
| --- |
| 行李数为 1  第一件行李尺寸超了  经济舱  重量超了  map[price:600] |

### 3.5 用例5

### 3.5.1 控制

测试头等舱两个行李不超长度

|  |
| --- |
| ft:= 0  pt := 1  ct := 3  hc := 1  var num Number = justGo  var tfc int64= 1  var fp int64 = 1000  var lc int64 = 2  x := []int64{100,100}  y := []int64{60,60}  z := []int64{40,60}  weight := []int64{40,30} |

### 3.5.2 输入

|  |
| --- |
| ticket:= NewTicket(FlightType(ft), PassengersType(pt),ClassType(ct), HasCard(hc), num, tfc, fp, lc, x, y, z, weight) |

### 3.5.3 输出

|  |
| --- |
| Running tool: /usr/local/go/bin/go test -timeout 30s go\_code/project22/bean -run ^(TestPrice7)$  ok go\_code/project22/bean 0.008s  Success: Tests passed.  //ft:0 pt:1 ct:1 hc:1 num:go tcf:1 fp:1000 lc:1 x:[120] y:[60] z:[40] weight:[40] |

### 3.5.4 过程

|  |
| --- |
| 行李数为 2  第一件行李尺寸没超  国内航班  头等舱  两个行李  重量没超  map[price:0] |

### 3.6 用例6

### 3.6.1 控制

测试头等舱两个行李，第一件超长度

|  |
| --- |
| ft:= 0  pt := 1  ct := 3  hc := 1  var num Number = justGo  var tfc int64= 1  var fp int64 = 1000  var lc int64 = 2  x := []int64{120,100}  y := []int64{60,60}  z := []int64{40,60}  weight := []int64{40,30} |

### 3.6.2 输入

|  |
| --- |
| ticket:= NewTicket(FlightType(ft), PassengersType(pt),ClassType(ct), HasCard(hc), num, tfc, fp, lc, x, y, z, weight) |

### 3.6.3 输出

|  |
| --- |
| Running tool: /usr/local/go/bin/go test -timeout 30s go\_code/project22/bean -run ^(TestPrice8)$  ok go\_code/project22/bean 0.010s  Success: Tests passed.  //ft:0 pt:1 ct:1 hc:1 num:go tcf:1 fp:1000 lc:1 x:[120] y:[60] z:[40] weight:[40] |

### 3.6.4 过程

|  |
| --- |
| 行李数为 2  第一件行李尺寸超了  国内航班  头等舱  两个行李  重量没超  map[price:1050] |

### 3.7 用例7

### 3.7.1 控制

测试国内航班金金银卡

|  |
| --- |
| ft:= 0  pt := 1  ct := 2  hc := 3  var num Number = justGo  var tfc int64= 1  var fp int64 = 1000  var lc int64 = 1  x := []int64{100}  y := []int64{60}  z := []int64{40}  weight := []int64{30} |

### 3.7.2 输入

|  |
| --- |
| ticket:= NewTicket(FlightType(ft), PassengersType(pt),ClassType(ct), HasCard(hc), num, tfc, fp, lc, x, y, z, weight) |

### 3.7.3 输出

|  |
| --- |
| Running tool: /usr/local/go/bin/go test -timeout 30s go\_code/project22/bean -run ^(TestPrice9)$  ok go\_code/project22/bean 0.009s  Success: Tests passed. |

### 3.7.4 过程

|  |
| --- |
| 行李数为 1  第一件行李尺寸允许  国内航班  成人票  商务舱  金金银卡  重量没超  map[price:0] |

### 3.8 用例8

### 3.8.1 控制

测试下国际航班超长度

|  |
| --- |
| ft:= 1  pt := 1  ct := 3  ota := 1  var num Number = justGo  var tfc int64= 1  var fp int64 = 1000  var lc int64 = 1  x := []int64{100}  y := []int64{60}  z := []int64{40}  weight := []int64{23} |

### 3.8.2 输入

|  |
| --- |
| ticket := NewOutTicket(FlightType(ft), PassengersType(pt),ClassType(ct), OriginTargetArea(ota), num, tfc, fp, lc, x, y, z, weight) |

### 3.8.3 输出

|  |
| --- |
| Running tool: /usr/local/go/bin/go test -timeout 30s go\_code/project22/bean -run ^(TestPrice10)$  ok go\_code/project22/bean 0.007s  Success: Tests passed. |

### 3.8.4 过程

|  |
| --- |
| 成人/儿童  区域一  有一个行李  头等舱  长度超了  map[price:980] |

### 3.9 用例9

### 3.9.1 控制

测试下国际航班不超

|  |
| --- |
| ft:= 1  pt := 1  ct := 3  ota := 1  var num Number = justGo  var tfc int64= 1  var fp int64 = 1000  var lc int64 = 1  x := []int64{50}  y := []int64{40}  z := []int64{30}  weight := []int64{23} |

### 3.9.2 输入

|  |
| --- |
| ticket := NewOutTicket(FlightType(ft), PassengersType(pt),ClassType(ct), OriginTargetArea(ota), num, tfc, fp, lc, x, y, z, weight) |

### 3.9.3 输出

|  |
| --- |
| Running tool: /usr/local/go/bin/go test -timeout 30s go\_code/project22/bean -run ^(TestPrice11)$  ok go\_code/project22/bean (cached)  Success: Tests passed. |

### 3.9.4 过程

|  |
| --- |
| 成人/儿童  区域一  有一个行李  头等舱  长度没超  map[price:0] |

### 3.10 用例10

更多测试用例看code/服务器/server/bean/ticket\_test.go

## 4. 测试用例表

**表1-1 等价类划分表**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 输入等价类 | 有效等价类 | 无效等价类 |
| 1 | 航班区域 | 1为012345（0代表国内，12345代表国际五个区域） | 10．不是纯数字  11、除012345之外的数字 |
| 2 | 乘客类型 | 2只能为0|1 | 12．不是纯数字  13．除01之外的数字 |
| 3 | 机舱类型 | 3只能为1|2|3 | 14．不是纯数字  15．除123之外的数字 |
| 4 | Vip卡 | 4只能为0|1|2|3 | 16．不是纯数字  17．除0123之外的数字 |
| 5 | 行李数 | 5行李数>=0 | 18．行李数<0 |
| 6 | 长 | 6长为数字且>=0 | 19．长非数字  20．长<0 |
| 7 | 宽 | 7宽为数字且>=0 | 21．宽非数字  22. 宽<0 |
| 8 | 高 | 8高为数字且>=0 | 23．高非数字  24. 高<0 |
| 9 | 重量 | 9重量为数字且>=0 | 25．重量为数字  26重量>0 |

**表1-2 测试有效等价类表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | **用例描述** | 测试数据 | 期望结果 | 覆盖等价类 |
| 1 | **国内普通** | [0,1,1,1,1,1,100,60,40,20] | 输入有效 | 123456789 |
| 2 | **国际普通** | [5,1,1,1,1,1,100,60,40,20] | 输入有效 | 12356789 |

**表1-2 测试无效等价类设计表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | **用例操作描述** | 测试数据 | 期望结果 | 覆盖等价类 |
| 1 | 不是纯数字 | [a,1,1,1 ,1,1,100,60,40,20] | 输入无效 | 10 |
| 2 | 除012345之外 | [6,1,1,1 ,1,1,100,60,40,20] | 输入无效 | 11 |
| 3 | 不是纯数字 | [0,a,1,1 ,1,1,100,60,40,20] | 输入无效 | 12 |
| 4 | 除01之外 | 0,2,1,1 ,1,1,100,60,40,20] | 输入无效 | 13 |
| 5 | 不是纯数字 | [0,1,a,1 ,1,1,100,60,40,20] | 输入无效 | 14 |
| 6 | 除123之外 | [0,1,4,1 ,1,1,100,60,40,20] | 输入无效 | 15 |
| 7 | 不是纯数字 | [0,1,1,a ,1,1,100,60,40,20] | 输入无效 | 16 |
| 8 | 除0123之外 | 0,1,1,5 ,1,1,100,60,40,20] | 输入无效 | 17 |
| 9 | 18．行李数<0 | [0,1,1,1 ,-1,1,100,60,40,20] | 输入无效 | 18 |
| 10 | 长非数字 | [0,1,1,1 ,-1,1,a,60,40,20] | 输入无效 | 19 |
| 11 | 长<0 | [0,1,1,1 ,1,1,-1,60,40,20] | 输入无效 | 20 |
| 12 | 宽非数字 | [0,1,1,1 ,-1,1,100,a,40,20] | 输入无效 | 21 |
| 13 | 宽<0 | [0,1,1,1 ,1,1,100,-1,40,20] | 输入无效 | 22 |
| 14 | 高非数字 | [0,1,1,1 ,1,1,100,60,a,20] | 输入无效 | 23 |
| 15 | 高<0 | [0,1,1,1 ,-1,1,,60,-1,20] | 输入无效 | 24 |
| 16 | 重量非数字 | [0,1,1,1 ,1,1,100,60,40,a] | 输入无效 | 25 |
| 17 | 重量<0 | [0,1,1,1 ,1,1,100,60,40,-1] | 输入无效 | 26 |

* **国内行李托运费计算因果图绘制**

原因：1.航班类型

2.乘客类型

3.vip卡

4.舱位类型

中间状态：

6.长宽高限制

7.重量限制

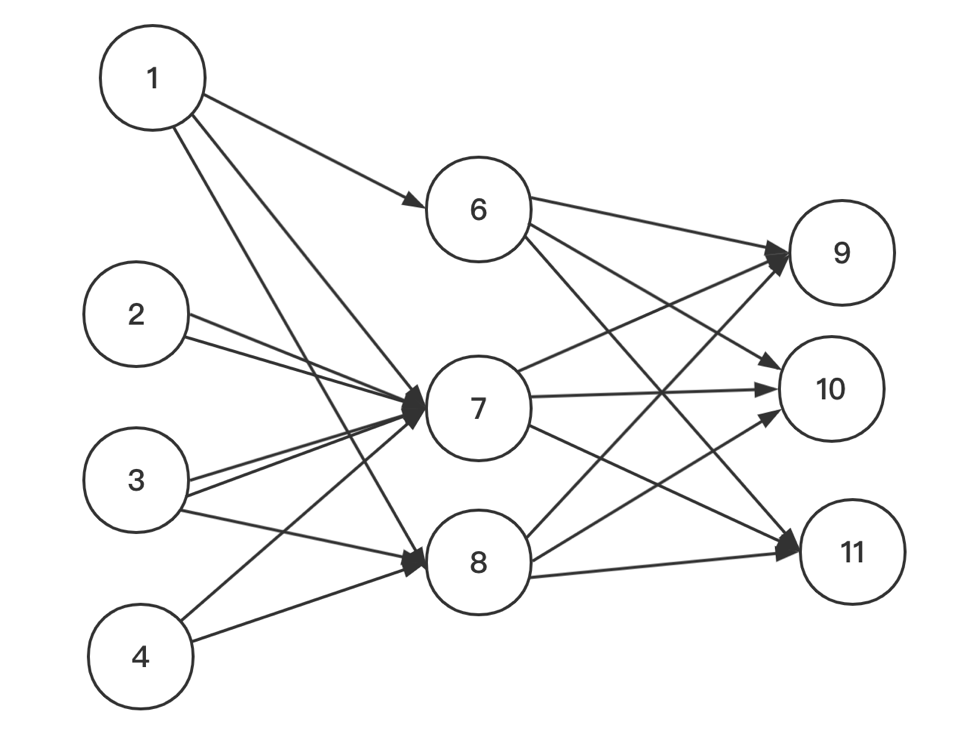
8.行李个数限制

结果：

9.免费

10.付一部分费用

11.全部付费

图示效果如下：

* **国际行李托运费计算因果图绘制**

原因：1.航班类型

2.乘客类型

3.区域限制

4.舱位类型

中间状态：

5.长宽高限制

6.重量限制

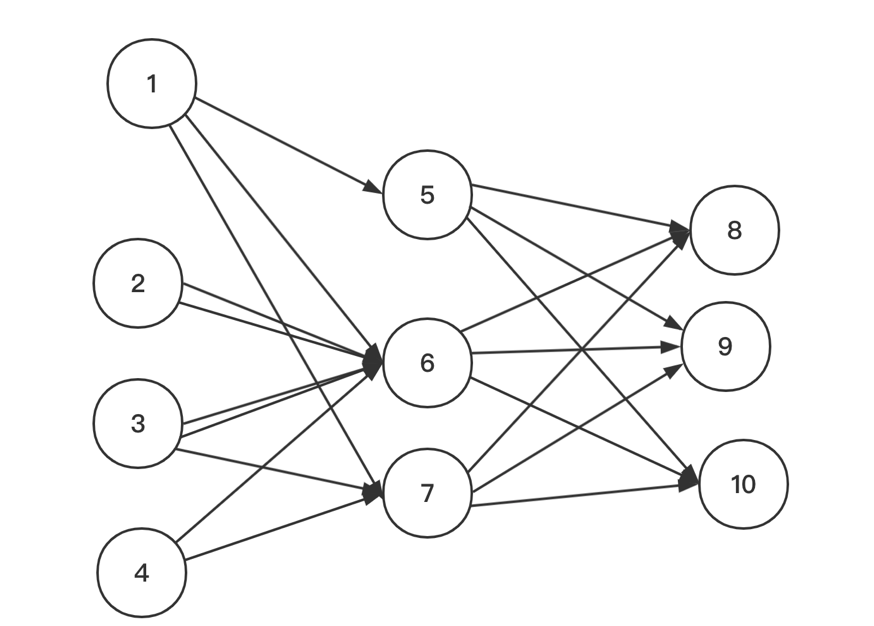
7.行李个数限制

结果：

8.免费

9.付一部分费用

10.全部付费

图示效果如下：

# 2、单元测试框架

## 1. 单元测试内容

### 1.1 模块接口测试

### 1.2 数据结构测试

### 1.3边界条件测试

### 1.4 独立路径测试

### 1.5 出错处理测试

## 2. 测试对象

每个被测试单元（即测试对象）主要按照以下步骤来写。

### 2.1 测试对象一

**2.1.1 软件单元描述**

简单描述被测试单元或与之相关单元的产品项目名称、所属子系统、单元要完成的功能、需求和设计要求等；

**2.1.2 单元结构**

画出本单元的组织结构，包括本单元包括的属性、方法、输入/输出等；

**2.1.3 单元控制**

根据本单元的控制结构或操作时序，画出其大概过程；

**2.1.4 测试过程**

**表2-1 测试过程表**

|  |  |  |
| --- | --- | --- |
| **测试步骤** | **动作描述** | **预计输出结果** |
| 例如：删除原有计划 | 使用菜单中的编辑功能，选中“计划删除”，删除原有的正常巡检计划  1）程序有确认删除的提示 | 记录程序提示并执行查询 |
|  |  |  |
|  |  |  |

### 2.2 测试对象二

**2.2.1 软件单元描述**

简单描述被测试单元或与之相关单元的产品项目名称、所属子系统、单元要完成的功能、需求和设计要求等；

**2.2.2 单元结构**

画出本单元的组织结构，包括本单元包括的属性、方法、输入/输出等；

**2.2.3 单元控制**

根据本单元的控制结构或操作时序，画出其大概过程；

**2.2.4 测试过程**

**表2-1 测试过程表**

|  |  |  |
| --- | --- | --- |
| **测试步骤** | **动作描述** | **预计输出结果** |
| 例如：删除原有计划 | 使用菜单中的编辑功能，选中“计划删除”，删除原有的正常巡检计划  1）程序有确认删除的提示 | 记录程序提示并执行查询 |
|  |  |  |
|  |  |  |

………

## 3. 测试结果

测试结果包括代码审查结果、测试用例的设计以及测试后的效果图。

**表2-2 测代码审查结果表**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bug ID** | **审查人员** | **审查日期** | **问题描述** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**附表三**

**测试用例执行结果统计表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试项** | **测试用例号** | **测试特性** | **用例描述** | **测试结论** | **对应bug ID** |
|  |  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |