

数据结构课程设计报告

二级学院 计算机科学与工程学院

专 业 软件工程

学生姓名 周翔辉

学 号 11603080122

指导教师 王森 、朱常鹏

时 间 2018.1

目录

[1.课程设计的目的与要求 1](#_Toc503546376)

[1.1目的与意义 1](#_Toc503546377)

[1.2基本要求 1](#_Toc503546378)

[2.设计题目1:数据压缩与解压缩 2](#_Toc503546379)

[2.1 题目说明 2](#_Toc503546380)

[2.2 题目设计 3](#_Toc503546381)

[2.3 系统测试 5](#_Toc503546382)

[2.4存在问题与解决方案 8](#_Toc503546383)

[3. 设计题目2：24点扑克牌游戏 8](#_Toc503546384)

[3.1 题目说明 8](#_Toc503546385)

[3.2 题目设计 9](#_Toc503546386)

[3.3 系统测试 12](#_Toc503546387)

[3.4存在问题与解决方案 17](#_Toc503546388)

[4. 设计题目3：16枚硬币的反面的问题 17](#_Toc503546389)

[4.1题目说明 17](#_Toc503546390)

[4.2 题目设计 18](#_Toc503546391)

[4.3 系统测试 22](#_Toc503546392)

[4.4存在问题与解决方案 23](#_Toc503546393)

[5. 设计题目5：线性表、树、图的操作和演示 24](#_Toc503546394)

[5.1题目说明 24](#_Toc503546395)

[5.2 题目设计 26](#_Toc503546396)

[5.3 系统测试 30](#_Toc503546397)

[5.4存在问题与解决方案 34](#_Toc503546398)

[6. 设计题目5：农夫过河问题的求解 34](#_Toc503546399)

[6.1题目说明 34](#_Toc503546400)

[6.2 题目设计 35](#_Toc503546401)

[6.3 系统测试 37](#_Toc503546402)

[6.4存在问题与解决方案 37](#_Toc503546403)

[7. 设计题目6：迷宫问题 38](#_Toc503546404)

[7.1题目说明 38](#_Toc503546405)

[7.2 题目设计 40](#_Toc503546406)

[7.3 系统测试 42](#_Toc503546407)

[7.4存在问题与解决方案 42](#_Toc503546408)

[8.课程设计总结 43](#_Toc503546409)

[9.参考文献 43](#_Toc503546410)

数据结构课程设计报告

# 1.课程设计的目的与要求

## 1.1目的与意义

目的：进一步理解和掌握课堂上所学各种抽象数据类型的逻辑结构、存储结构和操作；针对选定的题目，学会分析数据的逻辑结构和物理结构，结合Java语言完成设计方案；掌握软件设计的基本内容和设计方法，培养规范化软件设计的能力；培养综合运用基础理论知识和专业知识去解决实际数据结构应用设计问题的能力；学习使用各种参考资料，培养查阅手册、图标和文献资料的能力。

意义：数据结构是一门实践性较强的软件基础课程，为了学好这门课程，必须在掌握理论基础的同时，加强编程实现。数据结构课程设计就是在学习数据结构理论知识的基础上，通过课程设计这一实践环节，强化对数据结构知识点和课程体系的理解；训练综合运用所学知识处理实际数据结构问题的能力，达到理论与实际应用相结合；使分析能力、程序设计、编程实现与调试水平有一个明显的提高，为学习后续课程打好基础；培养软件工程师的基本素质。

1.2基本要求

1. 每人应至少独立完成3个题目并撰写课程设计报告，具体题目由任课老师组织分配，题目一旦选定，未经教师同意，不得私自更换。否则总评成绩为缺成绩。每人完成指定的题目，每个题目有多个不同层次的要求，实现某一层次取得对应比例的分值。所有软件工程创新实验室的同学，均要求独立完成全部6个题目。  
 2. 代码遵守命名和缩进规则关键代码需要注释说明。  
 3. 按时提交课程设计资料、说明文档，主要包括功能分析、解决方案、主要步骤，如：采用的逻辑结构、存储结构、实现的操作、采用的算法、包括类的主要变量和主要方法（用UML类图说明，标注类继承、组合和依赖关系），以及输入输出方式等。  
 4. 学生如能根据实际情况，分析题目要求中的不合理因素，设计出更合理、更有效或创新性的解决方案并实现，每个题目可获得不超过20%的加分。

# 2.设计题目1:数据压缩与解压缩

### 2.1 题目说明

利用哈夫曼编码进行信息通讯可以大大提高信道利用率，缩短信息传输时间，降低传输成本。但是，这要求在发送端通过一个编码系统对待传数据预先编码；在接收端将传来的数据进行译码(复原)。对于双工信道 (即可以双向传输信息的信道)，每端都需要一个完整的编/译码系统。试对于任意的一段文本（可能是直接输入的，也可能是保存在本地文件中或者网络上的)，写一个哈夫曼码的编译码系统。

[基本要求]  
一 个完整的系统应具有以下功能：

**(l)I:初始化 (Initialization)**。从终端读入字符集大小 n，及 n 个字符和 m 个权值，建立哈夫曼树，并将它存于文件 hfmtree 中。

**(2)C:编码 (Coding)**。利用已建好的哈夫曼树(如不在内存，则从文件 hfmtree 中读入)，对文件 tobetrans 中的正文进行编码，然后将结果存入文件 codefile 中。

**(3)D:解码(Decoding)**。利用已建好的哈夫曼树将文件 codefile 中的代码进行译码，结果存入文件 textfile 中。

**(4)P:打印代码文件 (Print)**。将文件 codefile 以紧凑格式显示在终端上，每行 50 个代码。同时将此字符形式的编码文件写入文件 codeprint 中。

**(5)T:打印哈夫曼树 (Tree printing)**。将已在内存中的哈夫曼树以直观的方式 (树或凹入  
表形式)显示在终端上，同时将此字符形式的哈夫曼树写入文件 treeprint 中。

[实现提示]  
 可以根据题目要求把程序划成 5 个模块，设计成菜单方式，每次执行一个模块后返回菜单。除了初始化(I)过程外，在每次执行时都经过一次读取磁盘文件数据。这是因为如果在程序执行后一直没有进行初始化(I)过程，为了能使后面的操作顺利进行，可以通过读取旧的数据来进行工作。比如：如果程序的工作需要的字符集和权值数据是固定的，只要在安装程序时进行一次初始(I)化操作就可以了。再次运行程序时，不管进行哪项操作都可以把需要的数据读入到内存。

### 2.2 题目设计

哈夫曼编码的输入是用户的输入，输出是编码后的哈夫曼编码、或者是解压后的字符串。首先，我将哈夫曼编码与解码抽象成一个Huffman类，其中包括用户的属性（即部分为用户的输入）、方法（即实现哈夫曼编码和解码和工具方法、构建一棵哈夫曼树用于编码和界面）。并且Huffman类也实现了Serializable接口，用于保存序列化后的哈夫曼对象。

###### 2.2.1 存储结构设计

使用数组存储编码的的哈夫曼编码

1. 用String保存输入的要编码的字符串；
2. 用String数组保存生成的哈夫曼编码；
3. 用int数组保存生成的哈夫曼编码的权重；
4. 由于要从文件中读取Huffman树，所以将Huffman类实现Serializable接口；
5. 因为哈夫曼编码是键值的组合，所以用Map来保存键和值，以便解码。

###### 2.2.2 关键算法设计

1 编码

通过循环将所有字符的哈夫曼编码通过二进制写入文件。

2 解码

1. 首先用一个StringBuffer来保存解码后的字符串；
2. 通过构造哈夫曼树时创建的Map键值对来获得一个解码器；
3. 循环整个编码后的哈夫曼编码，解码。

3 生成哈夫曼树

哈夫曼树是一个单独的内部类，其每个节点包括字符、字符的权重、字符出现的次数、字符的哈夫曼编码以及左右邻居。在编码时生成这个哈夫曼树（通过字符数组和哈夫曼编码数组）。

###### 2.2.3 系统静态模型

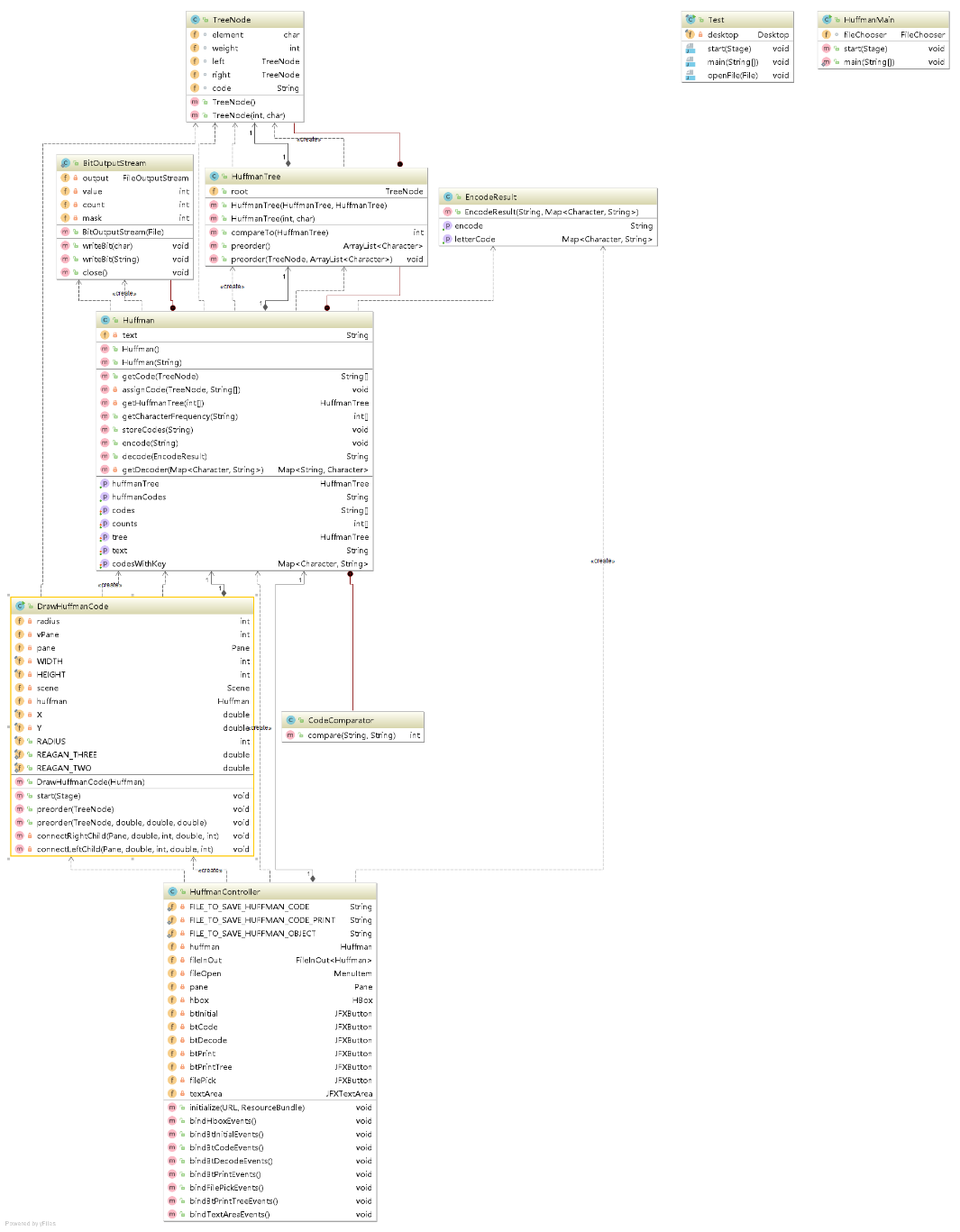


图1-1 题目1 Huffman UML系统静态模型

### 2.3 系统测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 1 | | 版本号 | | 1 |
| 测试环境 | Windows10 JDK1.8 IDEA2017.2 | | | | |
| 用例名称 | 编码 | | | | |
| 前提条件 | 输入ascii码以内的字符 | | | | |
| 测试步骤 | 读取输入字符，建立哈夫曼树，展示编码 | | | | |
| 输入数据 | The process | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 | 无 | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.20 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.20 |
| 再测试人 | 周翔辉 | 再测试日期 | | | 2017.12.20 |
| 问题修改摘要 | 无 | | | | |
| 修改人 |  | | 修改日期 | |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 2 | | 版本号 | | 1 |
| 测试环境 | Windows10 JDK1.8 IDEA2017.2 | | | | |
| 用例名称 | 解码 | | | | |
| 前提条件 | 编码之后 | | | | |
| 测试步骤 | 读取输入字符，建立哈夫曼树，展示编码 | | | | |
| 输入数据 | 01111110001010101110100101100110110 | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 | 无 | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.20 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.20 |
| 再测试人 | 周翔辉 | 再测试日期 | | | 2017.12.20 |
| 问题修改摘要 | 无 | | | | |
| 修改人 |  | | 修改日期 | |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 3 | | 版本号 | | 1 |
| 测试环境 | Windows10 JDK1.8 IDEA2017.2 | | | | |
| 用例名称 | 打印哈夫曼树 | | | | |
| 前提条件 | 编码之后 | | | | |
| 测试步骤 | 读取输入字符，建立哈夫曼树，展示编码 | | | | |
| 输入数据 | Hello world | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 | 无 | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.20 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.20 |
| 再测试人 | 周翔辉 | 再测试日期 | | | 2017.12.20 |
| 问题修改摘要 | 无 | | | | |
| 修改人 |  | | 修改日期 | |  |

### 2.4存在问题与解决方案

1. 存在问题

2. 解决方案

# 3. 设计题目2：24点扑克牌游戏

### 3.1 题目说明

一副牌中抽去大小王剩下 52 张（如果初练也可只用 1～10 这 40 张牌)，任意抽取 4 张牌（称牌组)，用加、减、乘、除（可加括号)把牌面上的数算成 24。每张牌必须用一次且只能用一次，如抽出的牌是 3、 8、 8、 9，那么算式为（ 9-8)×8×3 或 3×8+（ 9-8)或（ 9- 8÷8)×3等。本题主要考查栈、集合、数组、递归、穷举等知识。

### 3.2 题目设计

本题我通过设计一个抽象出一个Point类来实现题目要求，首先是Point类的属性包括需要达成的分数（不仅仅局限与24点，其它任何点都可以）、随机产生数的个数及大小范围（即牌的张数和牌的大小）；Point类的方法包括所有的Get和Set方法，以及计算随机产生数的解法getAnswers()。另外抽象出一个计算类Caculator，用于计算表达式。

###### 3.2.1 存储结构设计

由于栈先进后出的特性，所有使用栈存储操作符和操作数；

使用数组来保存随机产生的数字；

静态static char数组保存操作符

由于答案的个数不确定，所以使用ArrayList来存储答案。

###### 3.2.2 关键算法设计

1）计算中缀表达式，使用栈:

在处理操作数和运算符之前，它们被压人栈，当要处理一个运算符时，从保存运算符的栈中弹出它，然后将它应用在来自操作数栈的前两个操作数上，再将结果压回操作数栈中，直到操作符栈为空。

2）穷举操作符和操作数：

在穷举时的难的在于数或操作符的个数不确定性，所有需要数学知识找出一个通项公式，具体的方法就是使用数学书的归纳法，先假设是n是这样，再确定n+1也成立。

3) 表达式转化

Point类最初得到的结果是\*\*\*XXXX结构的，所有这就需要表达式的转换了，方法是通过表达式运算的逆推得到表达式。

###### 3.2.3 系统静态模型

1) 版本1

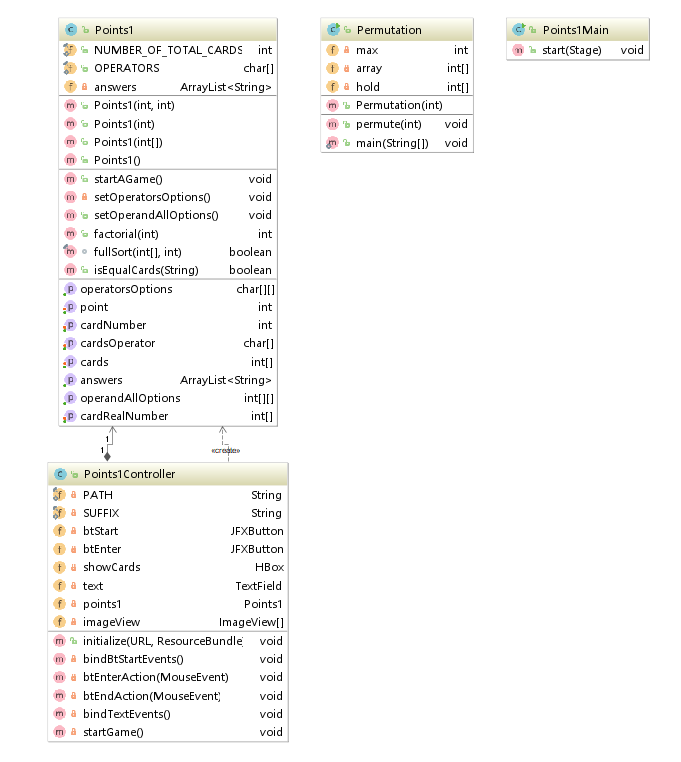


图2-1 题目2版本1系统静态模型

2) 版本2

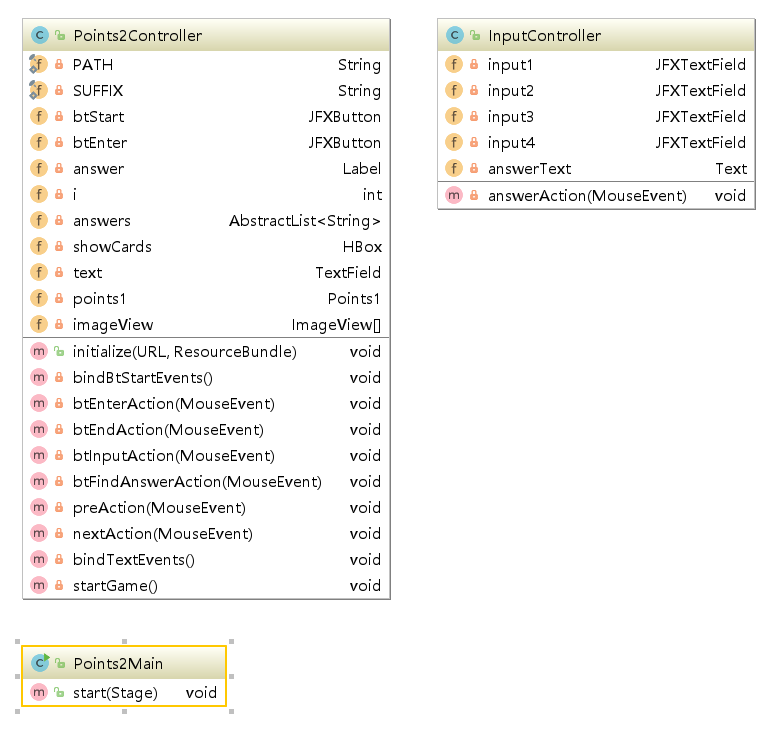


图2-2 题目2版本2系统静态模型

3) 版本3

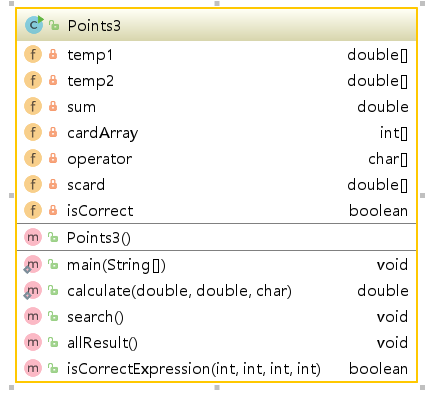


图2-3 题目2版本3系统静态模型

### 3.3 系统测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 1 | | 版本号 | | 1.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 24点扑克牌游戏 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 | 根据展示的纸牌输入合适的表达式 | | | | |
| 输入数据 | 表达式（9-6）+ 11 + 7 | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 2 | | 版本号 | | 1.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 24点扑克牌游戏 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 | 根据展示的纸牌输入合适的表达式 | | | | |
| 输入数据 | 表达式(7+1)\* (13-10) | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 3 | | 版本号 | | 2.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 24点扑克牌游戏 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 | 输入1-13内的四个数，寻找符合24点的 | | | | |
| 输入数据 | 2 3 4 4 | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 4 | | 版本号 | | 2.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 24点扑克牌游戏 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 |  | | | | |
| 输入数据 |  | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

### 3.4存在问题与解决方案

1. 存在问题

2. 解决方案

# 4. 设计题目3：16枚硬币的反面的问题

### 4.1题目说明

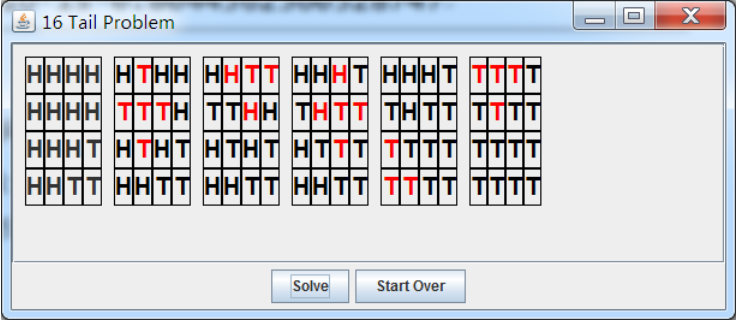
教材中的 9 枚硬币问题使用的是 3\*3 的矩阵，假设在一个 4\*4 的矩阵中放置了 16 枚硬币。该问题可进一步变化为如 2\*3， 2\*4， 3\*4 等等任意结构的情形。本题主要考查对图的结构和图的广度优先遍历操作的掌握。

**版本 1**（ 30%)

参考 9 枚硬币反面问题的模型，建立 16 枚硬币反面问题的模型，以及其他结构的模型。

**版本 2**（ 40%)

参考 9 枚硬币反面问题的解决方法，解决 16 枚硬币的反面问题。进一步解决其它结构的反面问题。



**版本 3**（ 30%)

修改硬币翻转规则，如规则改为对角线上的邻居被翻转，或者任意自定义的翻转规则。重新解决上述问题。

### 4.2 题目设计

通过对题目要求的分析，我抽象出了一个TailGame类，其中包括属性硬币的行数和列数，方法包括随机产生一个节点、得到到最终节点的最短路径、设置翻转的规则。其中在初始化TailGame对象时，可以设置硬币的行和列数，这就保证的程序的的灵活性，还可以设置规则（通过传人一个规则ArrayList）。

###### 4.2.1 存储结构设计

1）用图来保存硬币的所有情况；

2）用数组来保存图中每个节点硬币的翻转情况;

3）使用ArrayList<Rule>来保存设置的规则;

4）设计了一个规则类Rule来保存翻转的规则.

###### 4.2.2 关键算法设计

1）设置规则

（1）通过ArrayList来保存规则

（2）获取规则类Rules类保存相对于要翻转中心position位置的行row列column差；

（3）通过setRules()方法设置规则。

2）翻转节点

（1）读取规则rules;

（1）循环规则rules线性表，通过每个规则对象进行翻转。

###### 4.2.3 系统静态模型

1) 版本1

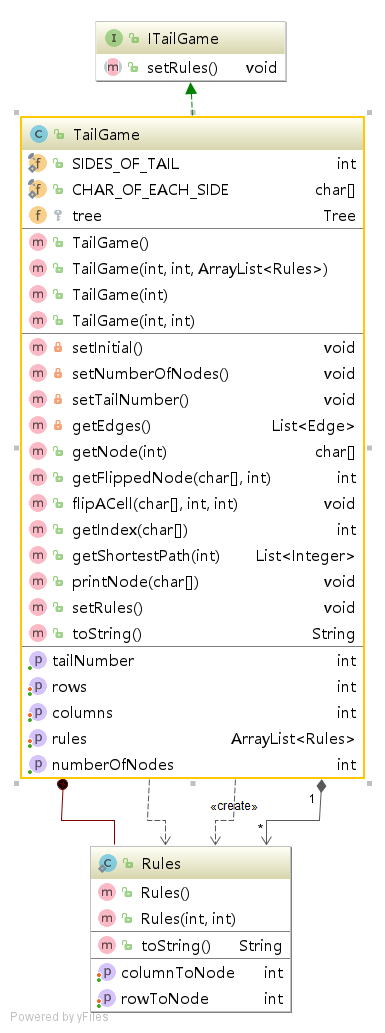


图3-1 题目3 版本1 uml图

2) 版本2

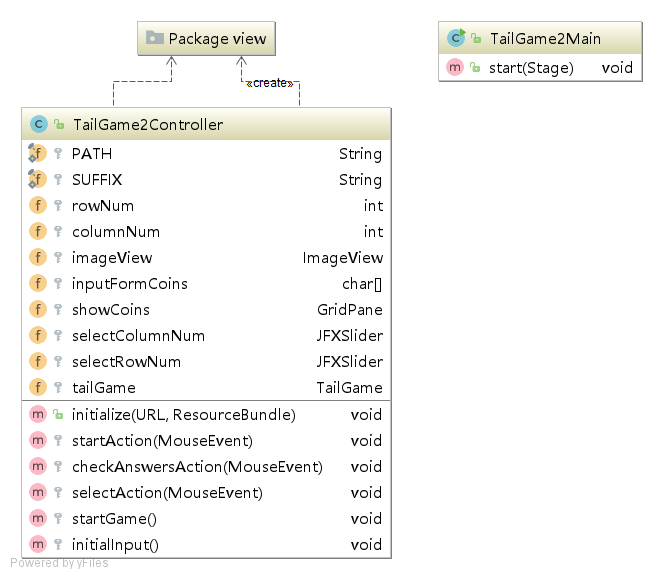


图3-2 题目三 版本2 uml图

3) 版本3

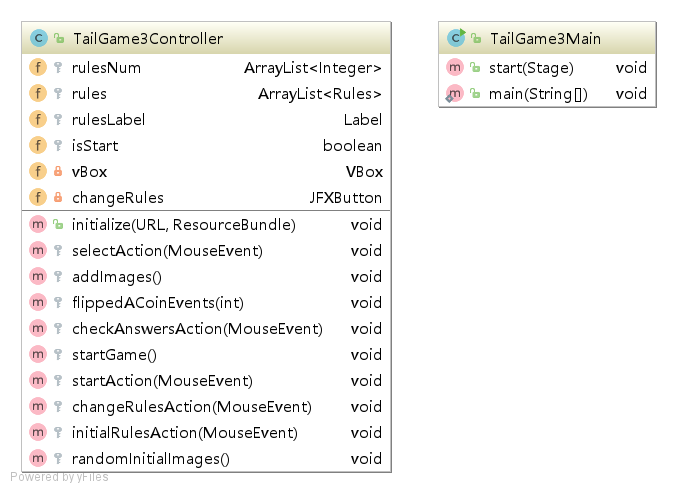


图3-3 题目三 版本3 uml图

### 4.3 系统测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 1 | | 版本号 | | 2.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 16枚硬币的反面问题 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 | 点击可以改变初始状态，查看答案按钮开始求解过程 | | | | |
| 输入数据 |  | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 2 | | 版本号 | | 3.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 16枚硬币的反面问题 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 | 点击可以改变初始状态，查看答案按钮开始求解过程 | | | | |
| 输入数据 | 规则 | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

### 4.4存在问题与解决方案

1. 存在问题

2. 解决方案

# 5. 设计题目5：线性表、树、图的操作和演示

### 5.1题目说明

线性表、树、图的操作和演示

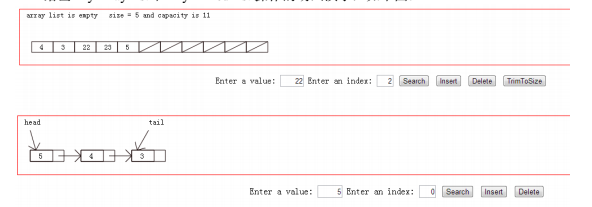
**版本 1** （40%)

实现接口 MyList、抽象类 MyAbstractList、基于顺序存储线性表 MyArrayList、基于链

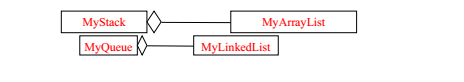
式存储的线性表 MyLinkedList。实现方式如下：



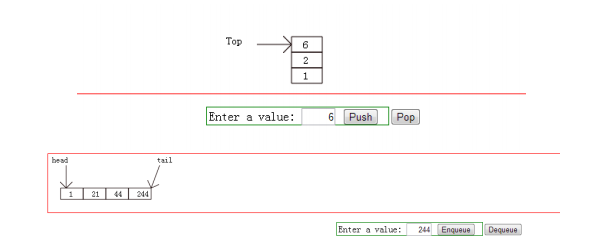
给出 MyArrayList 和 MyLinkedList 操作的动画演示，如下图：



实现栈 MyStack 和队列 MyQueue。实现方式如下：



给出 MyStack 和 MyQueue 的动画操作，如下图：

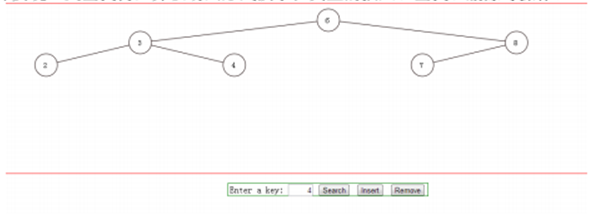


MyArrayList MyStack

MyLinkedList MyQueue

**版本 2** （30%)

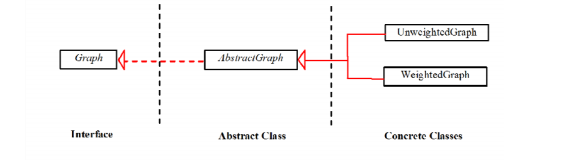
设计实现二叉查找树，并以动画形式演示在其上的插入、查找、删除等操作。



**版本 3** （30%)

设计实现接口 Graph、抽象类 AbstractGraph、具体类 UnweightedGraph 和 WeightedGraph

类。实现方式如下：



并用动画方式显示无权图的深度优先遍历、广度优先遍历；

使用动画方式显示带权图的最小生成树、最短路径：

### 5.2 题目设计

先设计每个题目的展示，然后将所有的界面全部放入一个总的界面内。

###### 5.2.1 存储结构设计

线性表存储输入的数据；

使用每个题目要求的存储类型存储数据。

###### 5.2.2 关键算法设计

1. 版本一

* 线性表的增加删除
* 链表的增加删除
* 栈的增加删除
* 队列的增加删除
* 计算各个图形的坐标

2) 版本二

树的删除和添加节点

计算每个节点的坐标

1. 版本3

得到最短路径：

得到一棵最小生成树

###### 5.2.3 系统静态模型

1）版本一

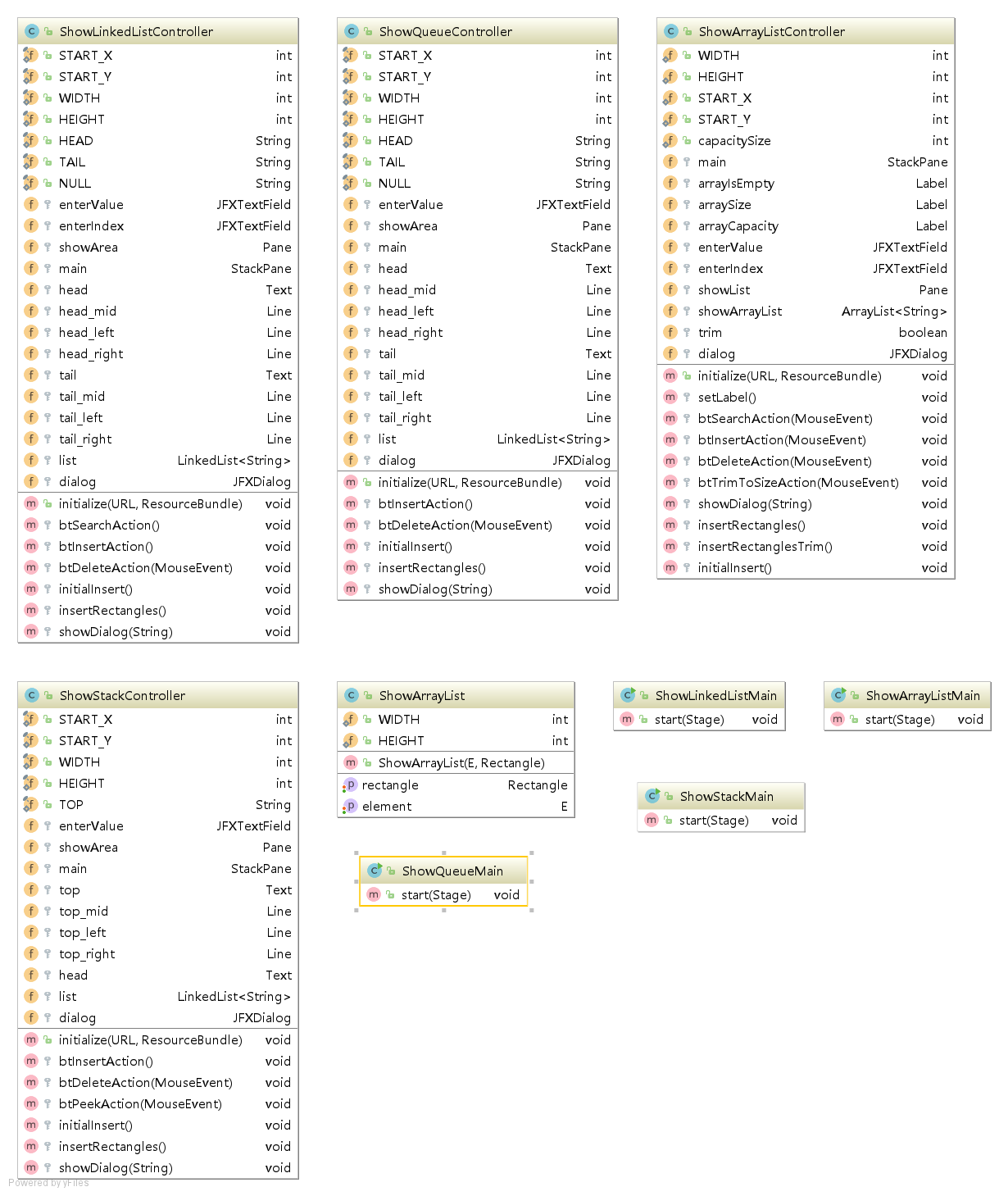


图4-1 版本1 集合框架的uml图

2）版本二

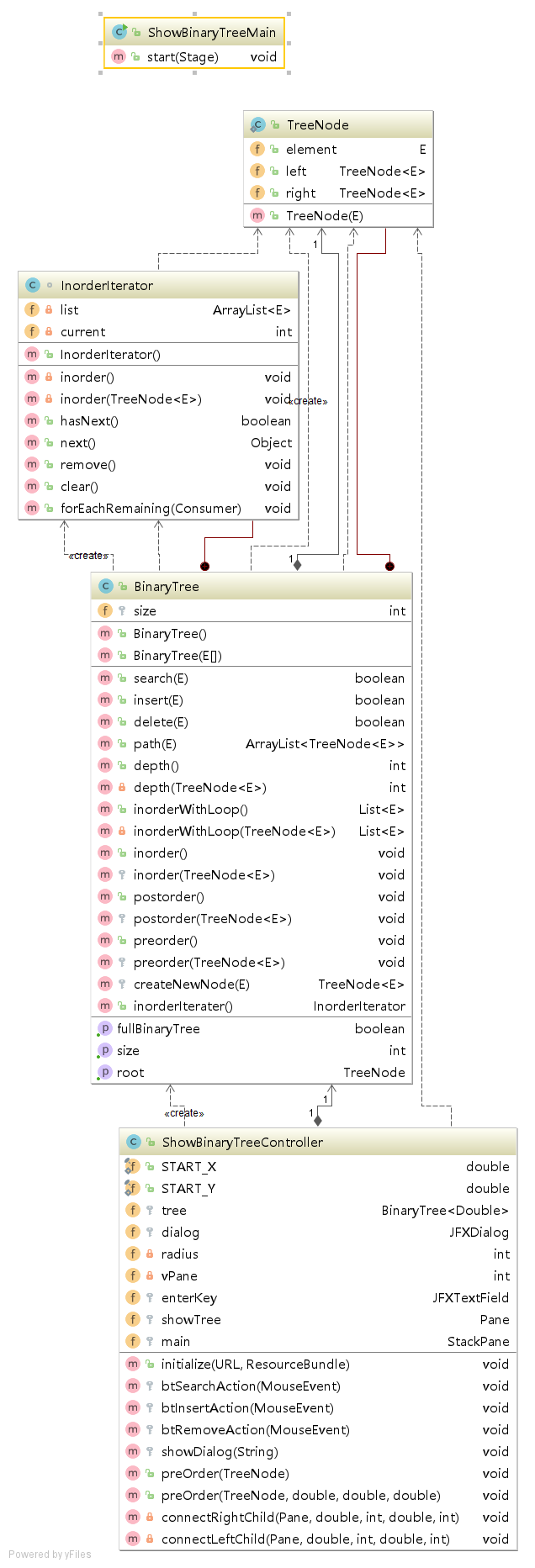


图4-2 版本2 二叉树的uml图

3）版本三

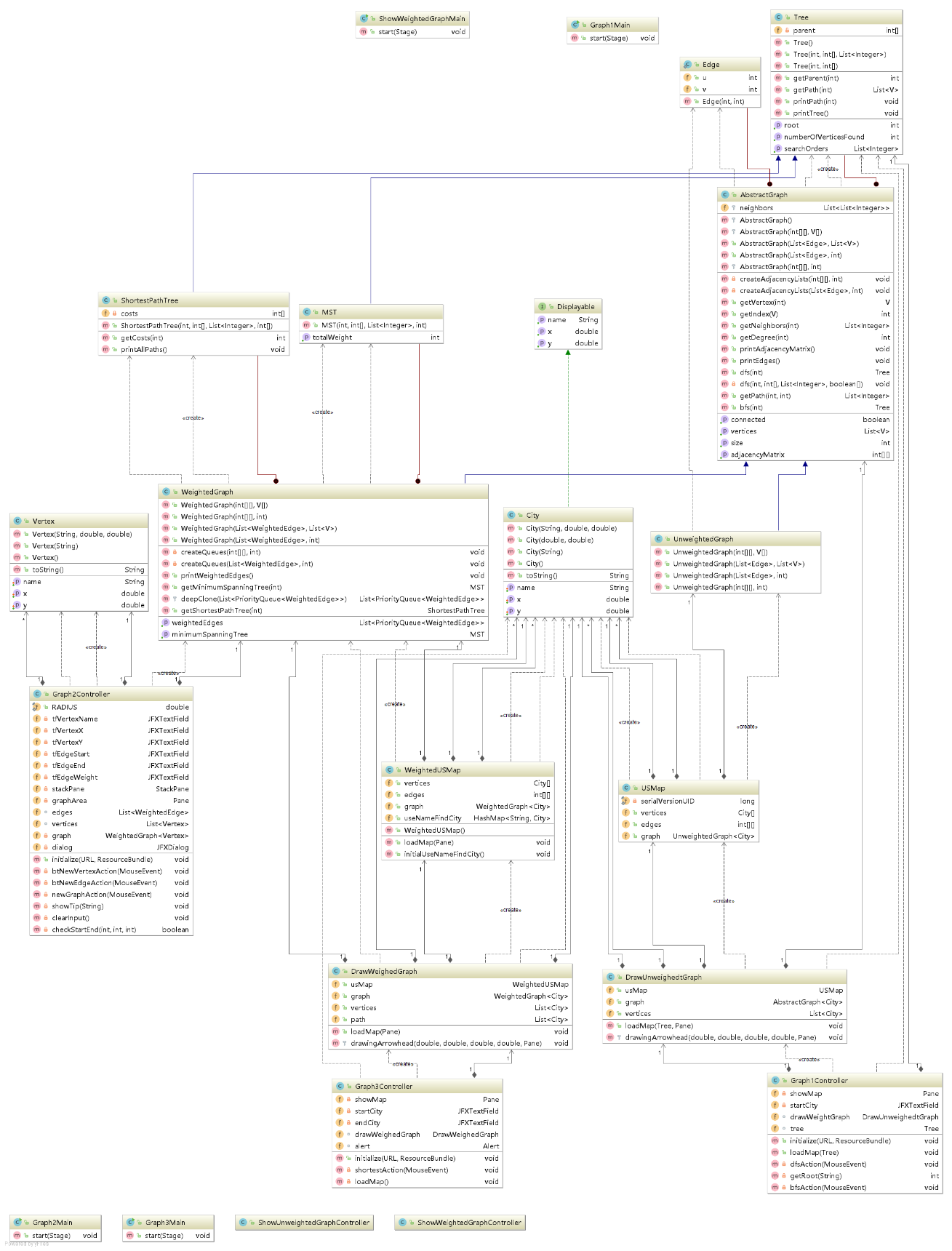


图4-3 版本3展示图的uml图

### 5.3 系统测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 1 | | 版本号 | | 1.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 线性表的演示 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 |  | | | | |
| 输入数据 | 1 2 3 | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 2 | | 版本号 | | 2.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 树的演示 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 |  | | | | |
| 输入数据 | 5 6 4 2 1 3 | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 3 | | 版本号 | | 3.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 图的演示 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 |  | | | | |
| 输入数据 | 起点城市 Dallas  4个点  起点城市：Los Angeles 终点城市：Boston | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

### 5.4存在问题与解决方案

1. 存在问题

2. 解决方案

# 6. 设计题目5：农夫过河问题的求解

### 6.1题目说明

一个农夫带着—只狼、一只羊和—棵白菜，身处河的南岸。他要把这些东西全部运到北岸。他面前只有一条小船，船只能容下他和—件物品，另外只有农夫才能撑船。如果农夫在场，则狼不能吃羊，羊不能吃白菜，否则狼会吃羊，羊会吃白菜，所以农夫不能留下羊和白菜自己离开，也不能留下狼和羊自己离开，而狼不吃白菜。请求出农夫将所有的东西运过河的方案。

### 6.2 题目设计

本题通过求出安全状态来判断过河顺序。即需要保证羊与狼不能单独在一起、白菜和羊不能单独在一起。我通过抽象成一个CrossRiver类来解决这个问题，首先得到每种东西的状态，并且保存在数组中，然后获取状态图，并得到所有的安全状态，最后寻找在保证安全状态下的通路。

###### 6.2.1 存储结构设计

数组存储在不同情况下农夫，狼，羊，白菜的位置；

用二维数组保存最后的解。

###### 6.2.2 关键算法设计

1. 使用十进制数和二进制数表示不同的状态；
2. 二进制的的操作表示状态的改变和判断；
3. 判断该状态是否安全：首先判断每种生物的状态、然后排除狼与羊、羊与狼的情况；
4. 获取每种生物的状态：首先传人一个状态的数字，然后将这个数字变成二进制，每位二进制的数字即表示该生物的状态，1表示南岸、0表示北岸。

###### 6.2.3 系统静态模型

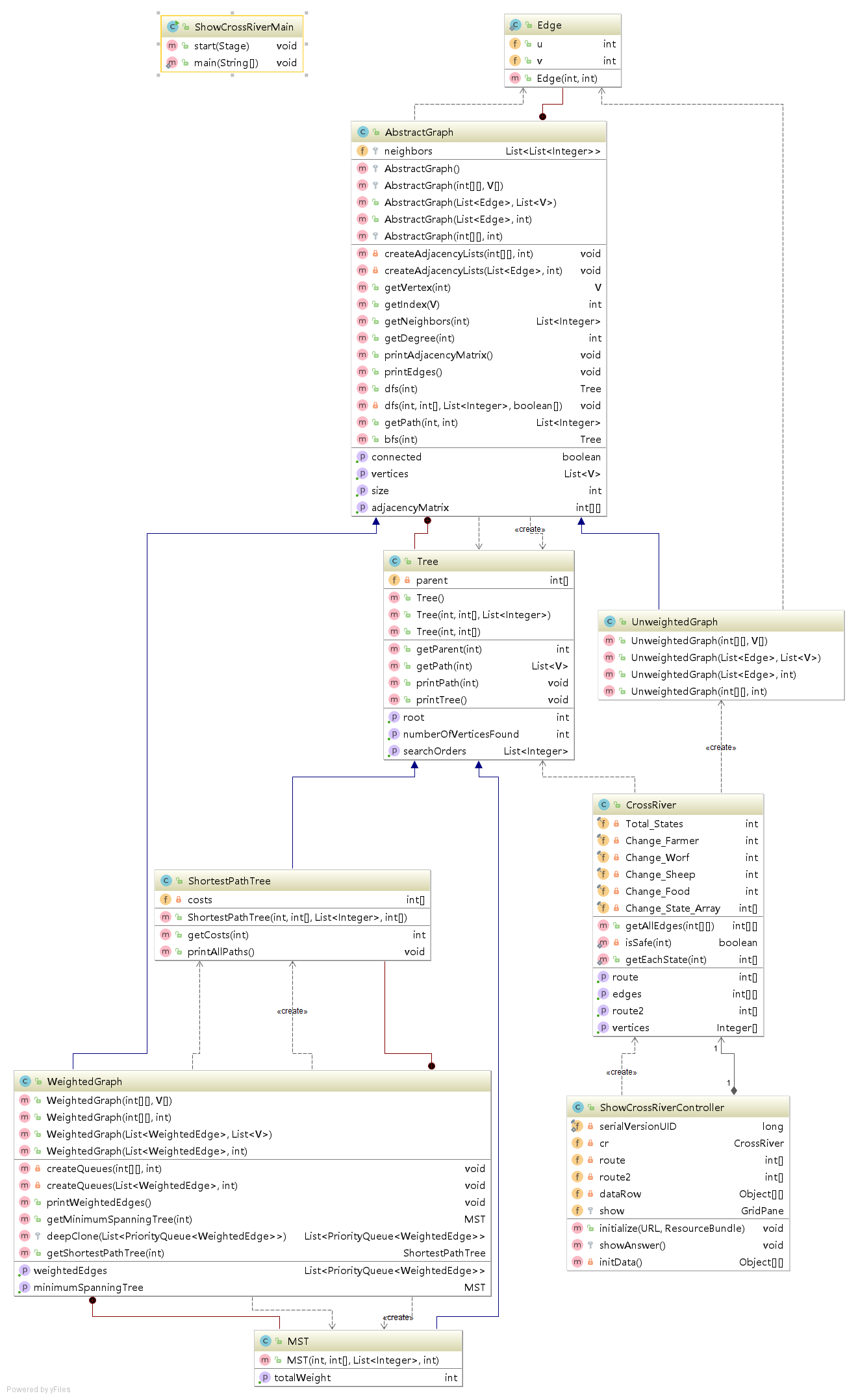


图 5-1 题目5农夫过河问题uml图

### 6.3 系统测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 1 | | 版本号 | | 1.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 农夫过河问题 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 |  | | | | |
| 输入数据 |  | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

### 6.4存在问题与解决方案

1. 存在问题

2. 解决方案

# 7. 设计题目6：迷宫问题

### 7.1题目说明

迷宫问题

[**问题描述**]

以一个 m\*n 的长方阵表示迷宫，0 和 1 分别表示迷宫中的通路和障碍。设计一个程序，

对任意设定的迷宫，求出一条从入口到出口的通路，或得出没有通路的结论。

[**基本要求**]

**（1）** 实现一个以链表作存储结构的栈类型，然后编写一个求解迷宫的非递归程序。求得

的通路以三元组（i,j,d)的形式输出，其中：（i,j)指示迷宫中的一个坐标，d 表示走到

下一坐标的方向。

**（2）** 编写递归形式的算法，求得迷宫中所有可能的通路；

**（3）** 以方阵形式输出迷宫及其通路。（选做)

[**测试数据**]

迷宫的测试数据如下：左上角（1，1)为入口，右下角（9，8)为出口。

1 2 3 4 5 6 7 8

0 0 1 0 0 0 1 0

0 0 1 0 0 0 1 0

0 0 0 0 1 1 0 1

0 1 1 1 0 0 1 0

0 0 0 1 0 0 0 0

0 1 0 0 0 1 0 1

0 1 1 1 1 0 0 1

1 1 0 0 0 1 0 1

1. 1 0 0 0 0 0 0

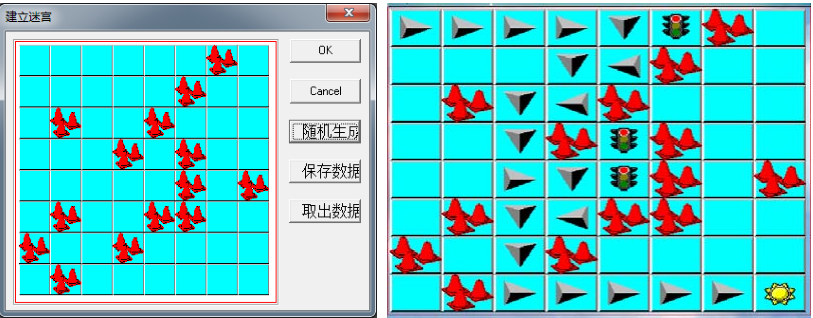
[**实现提示**]

计算机解迷宫通常用的是“穷举求解”方法，即从入口出发，顺着某一个方向进行探索，若能走通，则继续往前进；否则沿着原路退回，换一个方向继续探索，直至出口位置，求得一条通路。假如所有可能的通路都探索到而未能到达出口，则所设定的迷宫没有通路。

可以二维数组存储迷宫数据，通常设定入口点的下标为（1，1)，出口点的下标为（m,n)。

为处理方便起见，可在迷宫的四周加一圈障碍。对于迷宫中任一位置，均可约定有东、南、西、北四个方向可通。

图形界面实现可参见下图。



### 7.2 题目设计

本题主要是对图的遍历。将本小问抽象成Maze类，在新建Maze对象时，可以设置迷宫的长和宽、还可以设置墙出现的概率

###### 7.2.1 存储结构设计

二位数组存储迷宫数据；

用一个Position对象来保存迷宫每个节点的信息，包括其x、y坐标、方向、是否被访问；

最后用一个ArrayList来保存迷宫的通路。

###### 7.2.2 关键算法设计

1. 寻找通路

逐一判断当前位置的东西南北方向是否有通路，记录走过的路径

1. 寻找所有通路

寻找路径的过程中，将可以走的位置的信息存在对象中存入到栈中

###### 7.2.3 系统静态模型

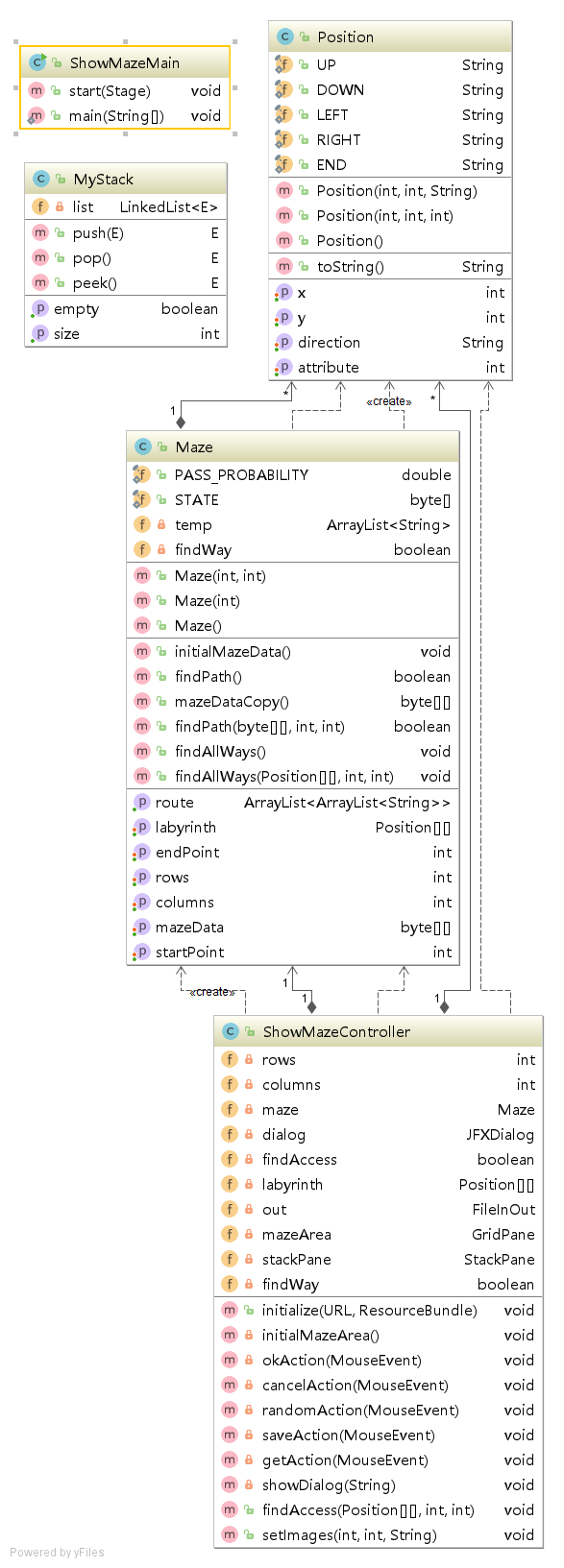


图6-1 迷宫问题uml图

### 7.3 系统测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 1 | | 版本号 | | 1.0 |
| 测试环境 | Windows10, JDK1.8, IDEA2017.02 | | | | |
| 用例名称 | 迷宫问题 | | | | |
| 前提条件 |  | | | | |
| 测试步骤 |  | | | | |
| 输入数据 |  | | | | |
| 预期输出 |  | | | | |
| 实际输出 |  | | | | |
| 问题描述 |  | | | | |
| 设计人 | 周翔辉 | 设计日期 | | 2017.12.25 | |
| 测试人 | 周翔辉 | 测试日期 | | | 2017.12.25 |
| 再测试人 |  | 再测试日期 | | |  |
| 问题修改摘要 |  | | | | |
| 修改人 |  | | 修改日期 | |  |

### 7.4存在问题与解决方案

1. 存在问题

2. 解决方案

# 8.课程设计总结

通过本次课程设计我学会了很多，不仅加强了我对Java知识的了解，也增加了我对编程的兴趣，更加懂的的编程不只是只靠自己脑子去想的，需要很多的学习与参考，学以致用才是我们现阶段的目标，而不是好高骛远。我想从这些方面来总结一下本次课程设计。

1 工作意识：做课程设计是是一个比较枯燥的过程，拥有良好的工作意识能够事半功倍，比如每天规定时间去敲代码，规定时间去复习知识或者学习。工作意识是慢慢培养的，并不是我们一开始就具有的，在这个过程中，也逐渐加强了我的工作意识。什么时候该玩，什么时候该学习，这些都了然于胸。

2 工作方法：不要一昧的去陷入某个陷阱中，不能解决的问题不要去死凿，要学会去发现问题；去问别人或者去上网搜索。在看待问题时应当具有大局观，应当先去构建整个框架，然后再去实现。

3解决问题：解决问题时应当独立思考，没有办法解决时可以查资料。所有的问题及其解决办法一定要有记录，包括解决的过程。

# 9.参考文献

1. Y.Daniel Liang.《Java语言程序设计进阶篇》.机械工业出版社.2011
2. Y.Daniel Liang.《Java语言程序设计(基础篇)》[M].北京：机械工业出版社，2016