



仲恺农业工程学院自编教材

数据结构实验指导书

V1.2

计算科学实验室编

（信息与计算科学专业用）

仲恺农业工程学院
二〇一二年

目 录

前言	1
实验 1 Java 运行环境的安装、配置与运行	3
一. 实验目的	3
二. 实验要求	3
三. 实验内容	3
实验 2 Array 的 Workshop 操作和拓展	11
一. 实验目的	11
二. 实验要求	11
三. 实验内容	11
实验 3 简单排序 (Simple Sorting)	13
一. 实验目的	13
二. 实验要求	13
三. 实验内容	13
实验 4 栈与队列 (Stacks & Queues)	15
一. 实验目的	15
二. 实验要求	15
三. 实验内容	15
实验 5 链表 (Linked Lists)	18
一. 实验目的	18
二. 实验要求	18
三. 实验内容	18
实验 6 递归 (Recursion)	20
一. 实验目的	20
二. 实验要求	20
三. 实验内容	20
实验 7 高级排序 (Advanced Sorting)	22
一. 实验目的	22
二. 实验要求	22
三. 实验内容	22
实验 8 二叉树 (Binary Tree)	24
一. 实验目的	24
二. 实验要求	24
三. 实验内容	24
实验 9 图 (Graph)	28
一. 实验目的	28
二. 实验要求	28
三. 实验内容	28
附录 1 Java 学习资源	29
附录 2 Java 编程命名规范	30
附录 3 Java 编程的注意事项	33

前言

一、数据结构实验目的

1. 培养学生数据抽象、对现实问题数据进行建模并实现为数据结构的能力；
2. 训练学生设计算法、分析算法的能力；
3. 培养学生运用多个类来分担程序职责，编写“大程序”的能力；
4. 训练学生调试程序，测试算法的能力；
3. 锻炼学生查询相关文献、网上资源，撰写科技实验论文的能力。

二、实验步骤及要求

1. 问题分析 （在实验课前完成）

充分分析和理解问题本身，弄清要求做什么，用什么数据结构和算法。

2. 程序设计 （在实验课前完成 80%）

（1）根据实验任务中的需求，抽象出数据所需要的存储结构以及对数据要进行的操作，设计数据结构，画出 class diagram。

（2）算法设计部分，要画出流程图或者活动图，分析算法的时间复杂度，并编程。

（3）准备调试程序的数据及测试方案，画出数据规模为自变量（按 10 倍增加），运算时间为因变量的图表。并做分析。

3. 上机调试

（1）对程序进行编译，纠正程序中可能出现的语法错误。

（2）调试前，先运行一遍程序看看究竟将会发生什么。

（3）如果情况很糟，根据事先设计的测试方案并结合现场情况进行错误跟踪，包括单步调试、设置观察窗输出中间变量值等手段。

4. 注册选课

数字大学城 <http://gd.nclass.org/sc8/>，需要[注册](#)、选课后才可提交实验作业，选课需要填入选课码：

111 班 QKVT-0306

112 班 AEUE-3560

113 班 BDWJ-8547

114 班 PWRA-0370

5. 整理提交实验报告

要求在数字大学城中提交电子版（实验报告和源代码）、名字命名示例：

200711314105-张兆敏-实验二-实验报告.doc

200711314105-张兆敏-实验二-源代码.rar （压缩包）

每个人的文件存储在名为：“学号-姓名-实验 X”的文件夹中。请在规定期限内（一般为布置后一周内）提交，逾期将无法提交，成绩记为 0 分！

三、实验报告

实验报告格式参见模版。其中应包含以下主要部分：

1. 实验采用的算法名称；
2. 问题描述：包括目标、任务、条件约束描述等；

3. 设计：数据结构设计和核心算法设计。主要功能模块的输入，处理（算法框架）和输出；
4. 测试范例：测试结果的分析讨论，测试过程中遇到的主要问题及所采用的解决措施；
5. 心得：包括实验或程序的改进设想，经验和体会；
6. 程序 Listing：源程序，其中包括变量说明及详细的注释。

实验1 Java 运行环境的安装、配置与运行

一. 实验目的

1. 掌握 Java 实验环境搭建的方法;
2. 掌握在 BlueJ 环境下编写、编译与运行 Java 程序的方法;
3. 了解 Java 语言的概貌。

二. 实验要求

1. 要求独立完成，并以实际上机操作计成绩;
2. 所有上机测试必须在实验课内完成，如果抽查发现未完成，此次实验成绩计 0 分;
3. 编写实验报告（按照统一的报告模板编写，**必须写出详细的实验步骤及必要的屏幕截图**）。

三. 实验内容

【Task 1】安装 JDK 软件包和 BlueJ。

安装文件在上课前准备好，可在 Sun 和 BlueJ 官方网站下载：

JDK: <http://java.sun.com/javase/downloads/index.jsp>

BlueJ: <http://www.bluej.org/download/download.html>。

【Task 2】在 BlueJ 下编辑\编译\执行代码。

BlueJ 的一个显著的优点是：你不仅仅能够执行一个完整的应用程序，而且能直接和任何类的对象交互并执行其中的公共（public）方法。在 BlueJ 中，一个执行过程通常通过创建一个对象，然后调用其中的方法来完成。这种模式使得你可以不需要编写完整的测试代码，就单独测试每个类。

我们要创建一个 **Staff** 对象，右键单击 **Staff** 图标(会弹出图 1-2 中显示的菜单)。这个菜单显示了两个构造函数可以用来构造 **Staff** 类。一个有参数，另外一个没有。首先，选择没有参数的构造函数。弹出的对话框应该如图 1-3 所示。这个对话框要求你输入一个被创建对象的名字。同时提供了一个缺省的名字 (**staff1**)，点击 **OK**。一个 **Staff** 对象就会被创建。

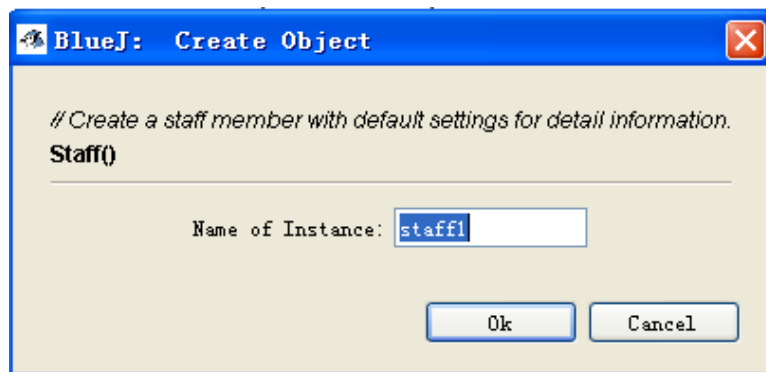


图 1-3 创建无参数的 **Staff** 对象

一旦一个对象被创建，它会被放在窗口底部的对象槽里（如图 1-4 中下方红色的方块 **Staff1**）。

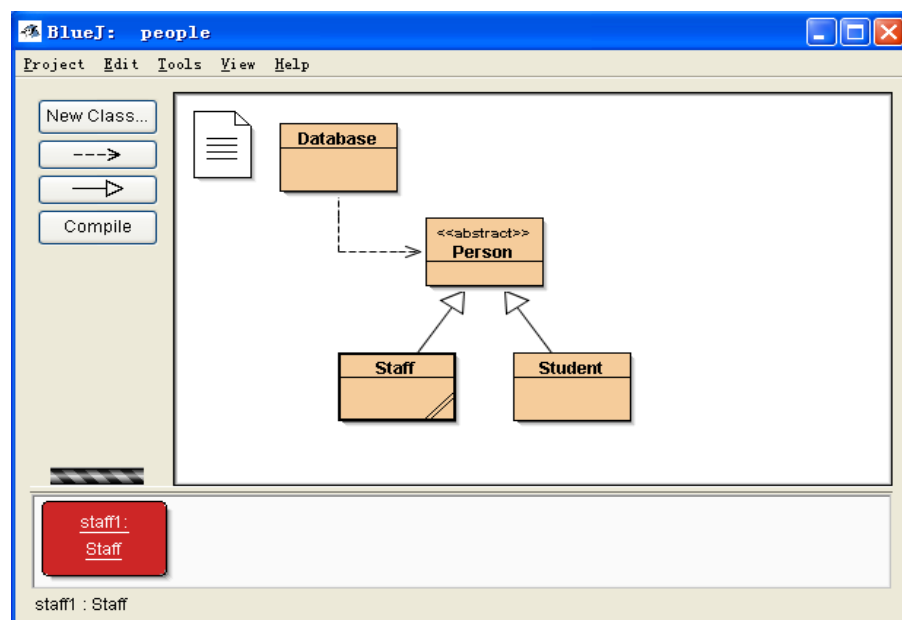


图 1-4 一个 **Staff** 对象在对象槽中

(3) 执行方法

现在你已经创建了一个对象，可以执行它的公共（public）方法。用右键单击 **Staff1** 对象图标，就会弹出一个包含对象操作的菜单（图 1-5 左下方弹出菜单）。这个菜单显示了该对象所有可执行的方法和两个 BlueJ 环境提供的操作（查看对象和删除对象）。尝试执行其方法和操作，观察结果。

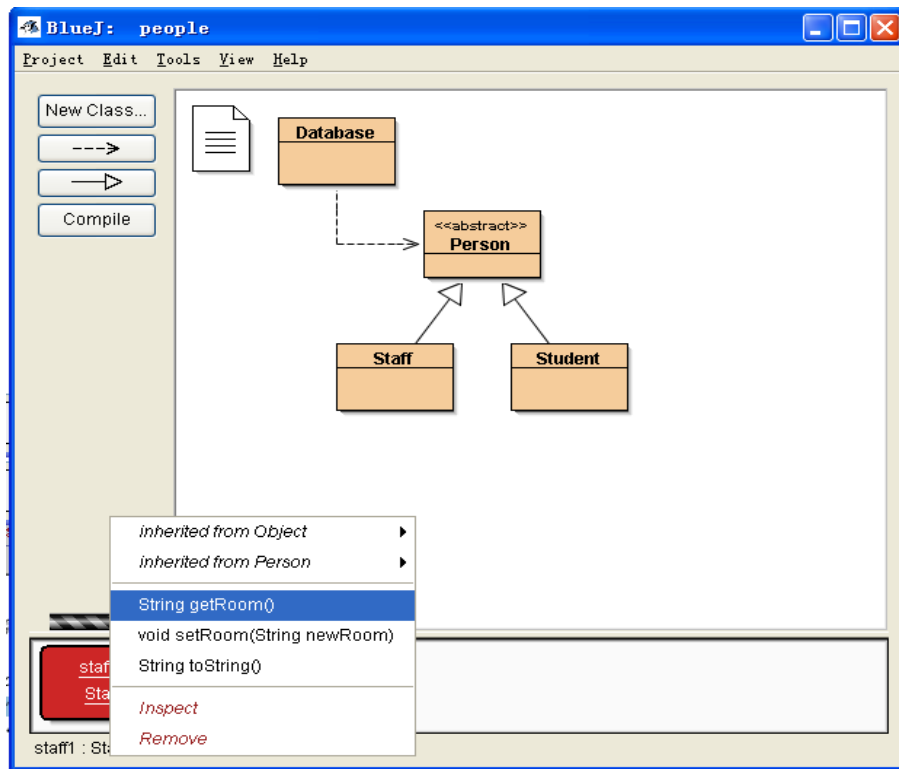


图 1-5 Staff 对象的方法

(4) 编辑一个类

双击类图标可以打开代码编辑器。例如双击 **Staff** 类，找到 **getRoom** 方法的实现。该函数会返回该 **Staff** 对象的房间号。我们在这个函数返回值的前面加上一个“room”前缀（这样这个方法就会返回，“room G.4.24”而不是“G.4.24”）。即：

将 `return room;`

改为

`return “room ” + room;`

(5) 编译

改变代码之后马上检查工程主窗口。你会发现 **Staff** 的图标出现了条纹。条纹的出现意味着类文件需要重新编译。

在编辑器顶端的工具条包含一些经常使用的功能按钮，其中一个 **Compile** (编译)。可以使用这个按钮直接编译当前打开的类文件。现在点击 **Compile** 按钮。如果你没有犯任何错误，那么在编辑器最下方的消息区会出现一条消息 (Class compiled – no syntax errors) 提示这个类通过了编译检查。

如果程序中有语法错误，错误行会高亮显示，并且在消息区（代码编辑窗口下方）会有相应的错误提示。（假设你第一次进行编译是没有出错，现在试着造成一个语法错误：比如漏写分号，然后再次编译，看看会出现什么效果？）编译错误

解析详见 <http://mindprod.com/jgloss/compileerrormessages.html#PARENEXPECTED>
成功编译完后，关闭编辑器。

工程窗口的工具条上同样也有编译按钮。这个编译操作会编译整个工程（实际上它判断哪些类需要重新编译然后按照正确的顺序重新编译这些类）。你可以试着修改2个或多个类（这样会有2个或多个类的图标出现条纹），然后点击**Compile**按钮，如果在被编译的几个类中出现错误，编辑器就会被打开，错误位置和错误信息会显示出来。

【Task 3】创建一个新工程

(1) 创建工程目录

要创建一个新的工程，从菜单中选择 **Project—New Project……**。然后自动打开一个文件选择对话框，对话框要求你为新工程确定一个名字和位置（注意不能与现有的重名！）。输入名字并选择 **Create** 之后，将按照你提供的名字创建一个目录，并且主窗口显示当前这个新建的空的工程。

(2) 创建类

可以通过在工程工具条上点击 **New Class** 按钮创建你的类。你要为这个类提供一个名字——这个名字必须是一个合法的 Java 标识符。

可以一下四种不同类中选择：**abstract**、**interface**、**Applet** 或者“**standard**”。这种选择将决定你的类将以何种初始的代码框架创建。你也可以通过编辑源代码修改类的类型（例如，把“**abstract**”关键字加入你的代码中）。

在创建一个类之后，它在图中以一个图标表示，不同的颜色标识不同类型的类。例如，蓝色表示一般的类，浅蓝色表示抽象类，绿色表示接口。当你打开一个新类的编辑窗口时，会发现一个默认类框架已经搭好了。默认框架代码是没有语法错误的，可以通过编译（但是它没有什么功能）。试着创建一些类并且编译它们。

(3) 创建依赖关系

类框图以箭头显示框中各个类之间的依赖关系。继承关系（“**extends**”或者“**implements**”）被显示为实线箭头；“**uses**”关系被显示为虚线箭头。

你既可以通过图形方式（直接在框图中）也可以通过在源代码中以文本方式添加依赖关系。如果你以图形方式添加了一个箭头，源文件也同时自动的更新了；如果你在源代码中添加了关系，框图也会自动更新。

想以图形方式添加一个箭头，点击想要的箭头按钮（实线箭头是“**extends**”或者“**implements**”关系，虚线是“**uses**”关系）。

添加一个继承箭头将在源文件中加入“**extends**”或者“**implements**”定义（依赖于目标是一个类还是一个接口）。

添加一个“**uses**”箭头不会直接改变源代码（除非目标是在另一个包中的类。那样将会产生一个“**import**”语句，但是在我们的例子中是看不见的）。如果拥有一个“**uses**”箭头指向另一个类而在源代码中实际上没有用到这个类，稍后将会产生一个警告，告知程序员声明了对另一个类的“**uses**”关系但是这个类却没有用到。

用文本方式添加箭头很容易：只需要像平常一样敲入代码即可。当代码被保存的时候，框图也更新了（记住：关闭文本编辑器将自动保存文本）。

(4) 删除元素

想从框图中删除一个类，选中这个类并且从编辑菜单中选择“Remove”。你同样可以从这个类的右键弹出菜单中选择“Remove”。想删除一个箭头，从菜单中选择“Remove”并且选中你想删除的箭头。

【Task 4】 调试程序

调试器的有以下三个主要功能：

- 设置断点
- 单步执行
- 查看变量

打开工程 debugdemo，（在 BlueJ 安装目录下的 examples 目录下）。这个工程包含了一些专门为演示调试器功能的类。

(1) 设置断点

设置断点允许你在源代码的某一行打断程序的执行。当程序的执行被打断后，你可以查看你的工程的状态。它通常可以帮助你理解你的源代码到底做了什么。

在编辑器文本区的左边是断点区（图 1-6）。在断点区单击来设置断点。一个小的停止图标会出现来标明断点。

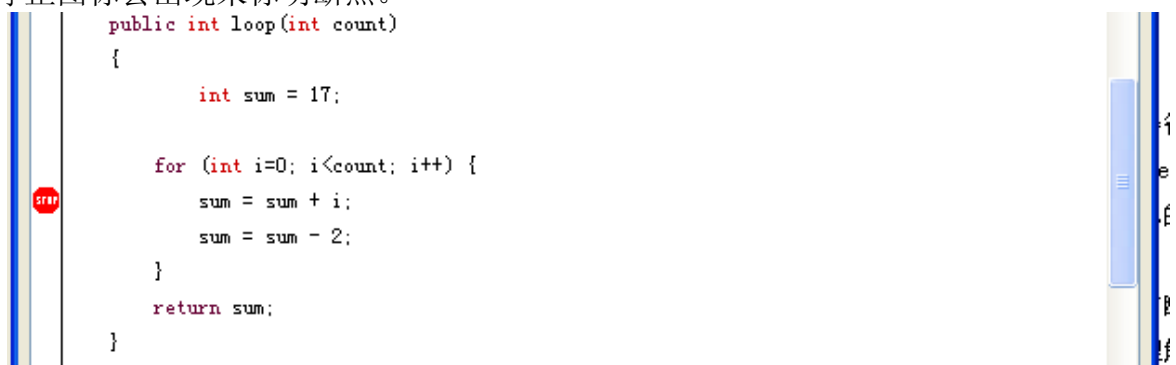


图 1-6 在 loop 方法中设置断点

现在打开类 Demo，找到方法 loop，在 for 循环内部设置一个断点，停止图标（红色的 STOP 标志）会出现在你的编辑器窗口里。

当程序执行到断点所在的行时，执行过程就会被中断。

创建一个 Demo 对象，并且用参数（10）调用 loop 方法。只要执行到断点所在行，编辑器窗口就会自动跳出，显示当前行的代码。同时调试器窗口也会出现，它应该看起来如图 1-7 所示。

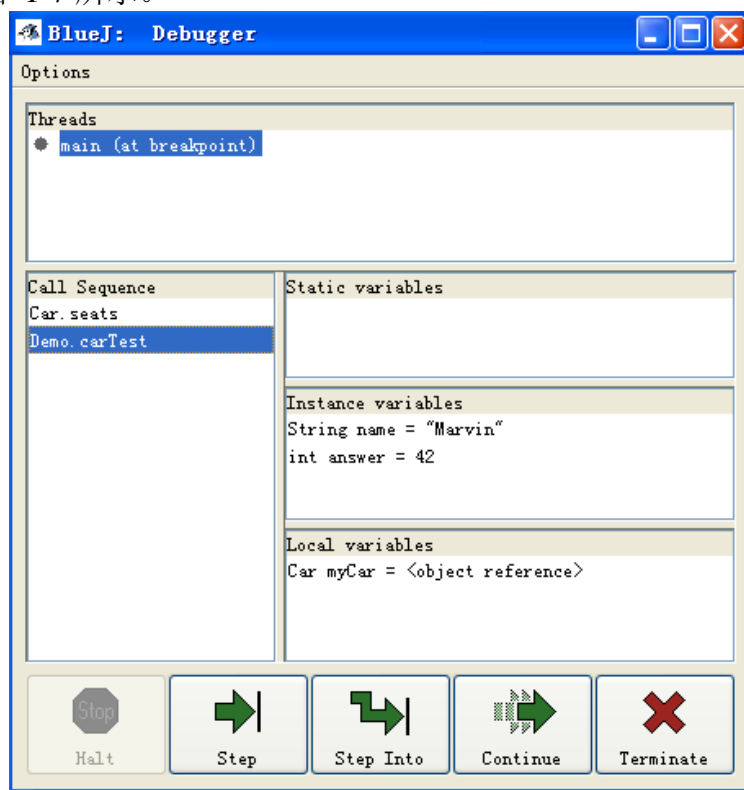


图 1-7 调试器窗口中查看变量的值

高亮显示的行是下一步要执行的行。

(2) 单步执行

程序的中断后，我们能单步执行去查看这个程序是如何向下执行的。可以通过不断的点击调试窗口里的 **Step** 按钮。你应该可以看到编辑器里的行号不断的变化（高亮显示的行随着将要被执行的行而移动）。每次你点击 **Step**，一行代码会被执行，并且又一次停止执行。同时请注意在调试窗口里显示的变量值也在不停的改变（比如 `sum`）。所以你可以一步一步的执行来观察发生了哪些变化。一旦你不需要单步执行，你可以再一次点击断点图标来删除它，然后点击调试器里的 **Continue** 图标来重新启动执行过程，按正常的执行顺序执行。

让我们用另一个方法来试一下。在 **Demo** 类 `carTest()` 方法里如下一行设置一个断点：

```
places = myCar.seats();
```

调用这个方法。当该行被执行到时，正要执行 **Car** 类的 `seats()` 方法。单击 **Step** 会执行整个行，而不会进入 `seats()` 方法。

这次让我们试试 **Step Into**。如果你使用 **Step Into**，你就会进入到一个方法的内部，单步执行该方法（就跟 **Step** 一样）。注意调试器显示的变化。

(3) 查看变量

当调试代码时，掌握对象的当前状态是很重要的（局部变量和成员实例变量）。当前对象的成员变量和当前方法的局部变量都会显示在调试窗口中。你可以选择调试器窗口中的 **Call Sequence** 窗格的方法来查看其他当前活动的对象或方法的变量。试一下，比如，在 `carTest()` 方法里再一次设置断点。在调试器窗口的左端，你能看见调用栈，当前的显示是

`Car.seats`

`Demo.carTest`

这表示 `Car.seats` 被 `Demo.carTest` 调用了，你可以选择这个列表中的 `Demo.carTest` 来查看源代码和当前各个变量的值。

实验2 Array 的 Workshop 操作和拓展

一. 实验目的

1. 理解并掌握 unordered array 和 ordered array 中插入、删除和查找操作算法实现的原理;
2. 学习运用 Array 对现实世界数据进行建模的设计方法。
3. 进一步熟悉 BlueJ 开发环境、了解 Java 的编程规范。

二. 实验要求

1. 能熟练操作 Workshop 中的 Applet, 完整回忆算法执行的过程, 能准确预测按键后进一步的执行结果;
2. 仔细阅读书上给出的程序 Listing-2. 3 和 Listing-2. 4, 能对其功能进行扩展。
3. 提交实验报告, 报告按指导书中的要求撰写;
4. 能完成 Task1 可得 C, 完成 Task1、Task2 可得 B, 完成 Task1、Task2、Task3 可得 A。

三. 实验内容

【Task1】操作 Workshop 中的 Applet, 验证算法。

1. 使用 Array 专题 Applet 进行插入、查找和删除数据项的操作。确保自己在点击前能预知执行的结果。分别在允许重复值和不允许重复值的情况下都做一遍。
2. 请事先想好 Ordered 专题中二分查找 Applet 每步选择的范围, 然后实验以验证自己的猜想, 如果没有猜对, 分析原因。
3. 在一个含有偶数个数据项的数组中是没有 middle Item 的。在这种情况下, 二分查找算法会先检查哪个 middle Item 呢? 使用 Ordered 专题 Applet 来验证你的想法。

【Task2】对 highArray 类进行扩展

1. 向 highArray.java 程序(Listing-2.3)的 HighArray 类添加一个名为 getMax() 的方法, 它返回数组中最大关键字的值, 当数组为空时返回-1。向 main()中添加一些代码来使用这个方法。可以假设所有关键字都是正数。
2. 修改 1 中的方法, 使之不仅返回最大的关键字, 而且还将该关键字从数组中删除。将这个方法命名为 removeMax()。

【Task3】对 orderedArray.java(Listing-2.4)进行扩展

1. 修改 ordArray 类, 使 insert(), delete()与 find()方法一样, 都使用二分查找来定位。

2. 向 `OrdArray` 类加入一个 `merge()`方法，使之可以将两个有序的源数组合并成一个有序的目标数组。

实现提示：

在 `main()`中添加代码，向两个源数组中插入随机数（见附录参考代码），调用 `merge()`方法，并将结果目的数组显示出来。两个源数组的数据项个数可能不同。在算法中需要先比较源数组中的关键字，从中选出最小的一个数据项复制到目的数组。同时还要考虑如何解决当一个源数组的数据项已经取完而另一个还剩一些数据项的情况。

附录：产生随机数数据的参考代码：

```
public long[] getRandArray(int maxSize) {
    long upperBound = 1000;           // 随机数的上限为 1000，可以任取其他数
    long[] result = new long[maxSize]; // 返回数组，内部装满随机产生的整数
    for(int j=0; j<maxSize; j++) // fill array with
    { // random numbers
        long n = (long)( java.lang.Math.random()*(upperBound-1) );
        result[j] = n;
    }
    return result;
}
```

实验3 简单排序 (Simple Sorting)

一. 实验目的

1. 理解冒泡排序、选择排序和插入排序的算法思想;
2. 掌握运用大 O 记号衡量算法时间复杂度的方法;
3. 了解其他简单排序算法-奇偶排序算法。

二. 实验要求

1. 提交实验报告, 报告按照模版编写, 内容包括:
 - (1) 自己编写的代码;
 - (2) 必要的流程图;
 - (3) 实测数据和结果分析;
 - (4) 程序运行结果截图。
2. 所有源码的输入按照规范进行 (变量、函数名称用小写字母开头, 类名以大写字母开头, 命名尽量名符其实, 推荐用英文。
3. 完成 Task1 可得 C, 完成 Task1、Task2 可得 B, 完成 Task1、Task2、Task3 可得 A。

三. 实验内容

【Task1】比较三种排序算法的速度

修改 bubbleSort.java(Listing3. 1)中的 main()方法, 新建一个大数组并给数组赋值。可以用下面的代码产生随机数:

```
for(int j=0;j<maxSize;j++)// fill array with random numbers
{
    long n=(long)(java.lang.Math. random()*(maxSize-1));
    arr. insert(n);
}
```

试插入 10000 个数据。在排序前和排序后分别显示这些数据。滚屏显示将花费很长的时间。把 display()方法注释掉就可以看出排序本身花了多少时间。在不同的机器上时间是不同的。选择一个适当的数组容量 (例如 10000, 20000, ..., 10^6), 排序并计时。然后用同样的数组容量对 selectSort.java(Listing 3.1) 和 insertsort.java(Listing 3.3)计时。观察这三种排序算法的速度的不同。在实验报告中写出实测数据(用图表方式), 并分析效率差别的原因。

【Task2】 逆序并分析效率

在 bubbleSort.java(Listing3.1)的 main 方法中设计代码, 使得输入的数据呈逆序 (99999, 99998, 99997, ..., 1, 0)。用和实验 1 中同样数量的数据, 观察排序运行的快慢, 用 selectSort.java 和 insertSort.java 重复这个实验。在实验报告中写出实测

数据，并分析效率差别的原因。

【Task3】 实现奇偶排序算法。

算法思路：

在数组中重复两趟扫描。第一趟扫描选择所有的数据项对， $a[j]$ 和 $a[j+1]$, j 是奇数($j=1, 3, 5, \dots$)。如果它们的关键字的值次序颠倒，就交换它们。第二趟扫描对所有的偶数数据项进行同样的操作($j=0, 2, 4, 6, \dots$)。重复进行这样两趟的排序直到数组全部有序。用 `oddEvenSort()`方法替换 `bubbleSort.java` 程序(Listing3.1)中的 `bubbleSort()`方法。尝试在不同数据量的排序中运行，统计出两趟扫描的进行的次数。

注：奇偶排序实际上在多处理器环境中很有用，处理器可以分别同时处理每一个奇数对，然后又同时处理偶数对。因为奇数对是彼此独立的，每一对都可以用不同的处理器比较和交换。这样可以快速地排序。

实验4 栈与队列 (Stacks & Queues)

一. 实验目的

1. 理解 Stack、Queue 和 PriorityQueue 的逻辑操作，及其以数组为基础的实现方式；
2. 理解数组为基础的 Queue 实现空间充分利用的算法思想，对其功能进行扩展；
3. 了解 Queue 的简单应用。

二. 实验要求

1. 仔细阅读实验指导书，并于实验课前将大部分代码录入。
2. 提交实验报告，报告按照模版编写，内容包括：
 - (1) 基本操作步骤；
 - (2) 问题的回答；
 - (3) 算法的基本思路、自己编写的代码和运行结果截图。
3. 完成 Task1 可得 C，完成 Task1、Task2 可得 B，完成 Task1、Task2、Task3 可得 A。

三. 实验内容

【Task1】操作 workshop

1. 操作 Queue 的 Applet，从初始设置开始，交替地移除和插入数据项。(这样，你可以重复地用删掉的关键字值，而不必键入新数据项的值了。)注意观察这些数据项是如何缓慢地上升到队列的顶端，然后出现在队列底端又接着往上移动的。
2. 操作 PriorityQ 的 Applet，计算优先级队列满和空的时候 Front 箭头和 Rear 箭头的位置。为什么优先级队列不能像普通队列那样回绕呢？
3. 联想现实生活中栈和队列的例子各 2 个，写在实验报告中。

【Task2】编写一个新的方法 display ()

在 Listing 4.4 中的 Queue 类编写一个新的方法 display ()，显示队列的内容。

注意:不是简单地显示出数组的内容。要求按队列的状态，从队头的数据到队尾的数据，按次序显示出来。注意:不要输出因为在数组末端回绕而折成两半的样子，也不要使用出队的操作来获取队列元素。

【Task3】模拟超市收款队列

队列通常用于模拟人、汽车、飞机、业务等等的流动情况。

应用 queue.java 程序(Listing 4.4)的 Queue 类，编写一个程序模拟超市的收款队列。可以用 Task2 中的 display()方法，显示出顾客的几条队列。

可以通过敲击一个特定的键（例如'e'）插入一个新的顾客（可以用一个随机整数来代替，如下图中的 33, 35...。并为顾客选择排在哪一个队列上(可选择任一排队策略，例如：按人最少，或按结账时间最短)。

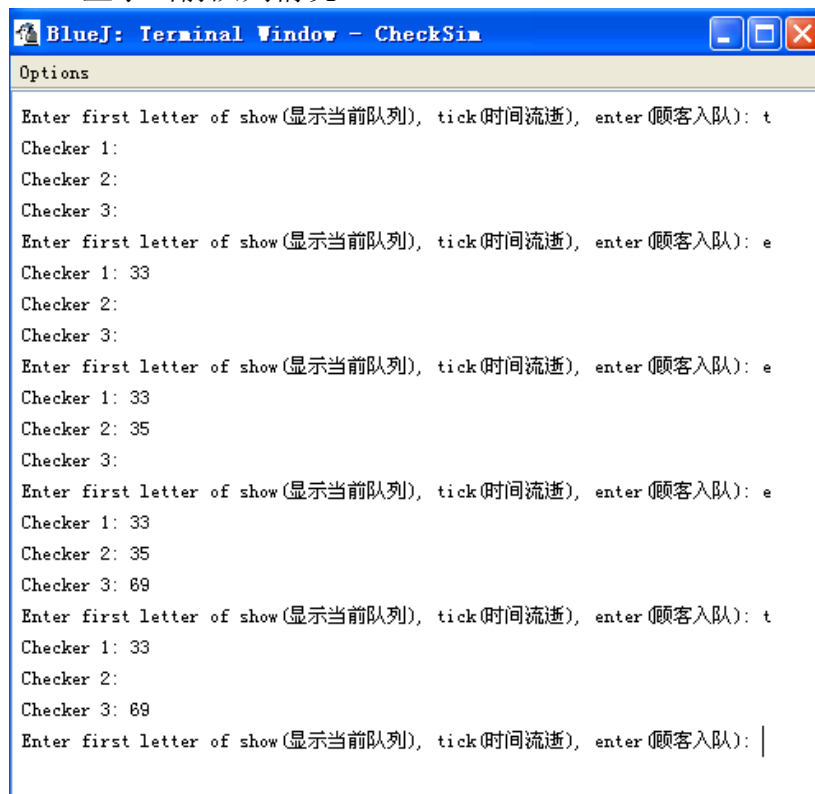
收银员为每个顾客服务的时间是随机的（也可以按顾客编号的个位数，例如顾客“33”，轮到他以后，所需的结账时间为 $33\%10=3$ 分钟）。一旦结完账，就从队列中删除该顾客。为了简单起见，通过敲击某个特定键（例如't'）模拟时间的流逝。例如：每点击一下键表示时间过去了 1 分钟。

运行界面如下图所示，分别输入：

“t”---时间流逝

“e”---顾客入队

“s”---显示当前队列情况



```
BlueJ: Terminal Window - CheckSim
Options
Enter first letter of show (显示当前队列), tick (时间流逝), enter (顾客入队): t
Checker 1:
Checker 2:
Checker 3:
Enter first letter of show (显示当前队列), tick (时间流逝), enter (顾客入队): e
Checker 1: 33
Checker 2:
Checker 3:
Enter first letter of show (显示当前队列), tick (时间流逝), enter (顾客入队): e
Checker 1: 33
Checker 2: 35
Checker 3:
Enter first letter of show (显示当前队列), tick (时间流逝), enter (顾客入队): e
Checker 1: 33
Checker 2: 35
Checker 3: 69
Enter first letter of show (显示当前队列), tick (时间流逝), enter (顾客入队): t
Checker 1: 33
Checker 2:
Checker 3: 69
Enter first letter of show (显示当前队列), tick (时间流逝), enter (顾客入队): |
```

其中的 Checker 1-3 所在的行，是模拟收银台对应的顾客队列。33、35、69 模拟等待结账的顾客编号。数字消失表示顾客已经结账离开。

提示 1：考虑引进多几个类来分担职责。不要把所有的“*How it Does*”的重任都交给 main 函数完成！

提示 2:几个有用的函数

```
import java.io. *; // for I/O
class SomeApp
{
    public static String getString() throws IOException // 接受键盘输入一个字符串
    {
        InputStreamReader isr = new InputStreamReader(System. in);
        BufferedReader br = new BufferedReader(isr);
        String s = br.readLine();
    }
}
```

```

        return s;
    }
    // -----
    public static char getChar() throws IOException // 接受键盘输入单字符
    {
        String s = getString();
        if(s.length() == 0)
            return '#';
        else
            return s.charAt(0);
    }
    // -----
    public static int getInt() throws IOException // 接受键盘输入数字
    {
        String s = getString();
        return Integer.parseInt(s);
    }

    public static void main(String[] args) throws IOException
    {
        while(true)
        {
            System.out.println("Enter your choice (A,B,C,D)");
            char choice = getChar();
            if(choice == 'A')
            {
                System.out.println("Unbelievable! You make the right choice! You choice is "
+ choice);
            }
            else
                System.out.println("You are dead. . . ");
        }
    } // end main()
}

```

实验5 链表 (Linked Lists)

一. 实验目的

1. 理解链表的基本操作，并掌握其实现算法；
2. 理解基于链表的栈的实现；
3. 了解循环链表及其应用。

二. 实验要求

1. 仔细阅读实验指导书，并于实验课前将大部分代码录入。
2. 提交实验报告，报告按照模版编写，内容包括：
 - (1) 基本操作步骤；
 - (2) 问题的回答
 - (3) 类图、算法的基本思路、自己编写的代码和运行结果截图。
3. 完成 Task1、Task2 可得 C, 完成 Task1、Task2、Task3 可得 B, 完成 Task1~Task4 可得 A。

三. 实验内容

【Task1】Workshop 和 Listing 程序更改

1. 用 LinkList 专题 Applet 执行插入、查找和删除操作，在有序链表和无序链表上都尝试一下。对于这些操作，有序链表有什么优势吗?请在在实验报告中说明。

2. 修改 linkList.java 程序(Listing 5.1)的 main()方法，使它能够继续插入 Link，直到内存耗尽。每隔 1000 个数据项，让它显示一次插入的 Link 数目。通过这种方法，可以粗略知道你机器可以容纳多少个 Link。(当然，Link 数目的多少也依赖于内存中其他程序和许多其他因素。)请在在实验报告中说明实测结果。

【Task2】基于有序链表的优先级队列

实现一个基于有序链表的优先级队列。队列的删除操作应该删除具有最小关键字的 Link。

【Task3】循环单链表类 CircleLinkList

编写一个循环单链表类 CircleLinkList，它没有表头也没有表尾。编写相应的测试代码，项目类图可参考图 5-1。

循环链表是一种链表，它的最后一个 Link 指向第一个 Link。设计循环链表有许多方法。有时，用一个指向链表“开始”的指针。然而这样做使链表不像一个真正的环，而更像一个传统的链表，只不过这个链表的表头和表尾系在了一起。访问这个链表的惟一方式是一个引用 current，它能指向任意一个 Link。这个引用在需要的时候可以沿链表移动。(参考 Task4，那种情况下用循环链表很合适。)

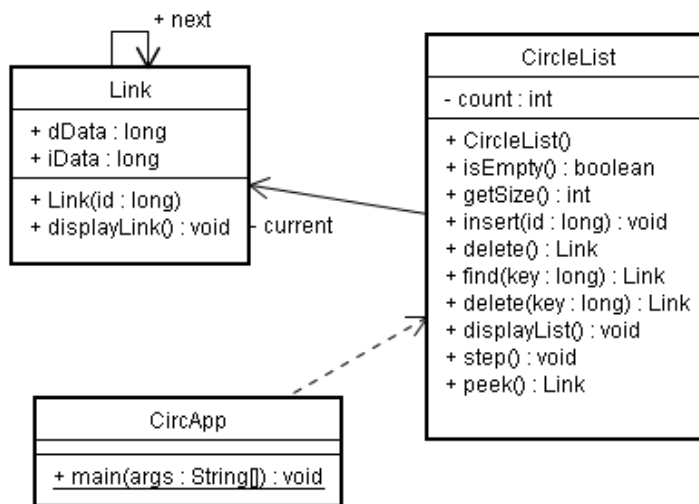


图 5-1 循环链表的 Class diagram

设计要求:

- (1) 循环链表类应提供插入、查找和删除的方法;
- (2) 可以显示链表(尽管需要在循环链表的某处切断环, 以把它们打印到屏幕上);
- (3) 编写一个 step()方法, 可以把 current 移动到下一个 Link, 在 Task4 中, 这个方法能也会派上用场。

【Task4】Josephus 问题

这是古代一个著名的数学难题。围绕这个问题有很多故事。其中一个说 Josephus 是一群被罗马人抓获的犹太人中的一个, 为了不被奴役, 他们选择了自杀。他们排成一个圆圈, 从某个人开始, 沿着圆圈计数。每报第 n 个数的人就要离开圆圈去自杀。Josephus 不想死, 所以他制定了规则, 以使他成为最后一个离开圆圈的人。如果有(例如)20 个人的话, 他就是从开头数第 7 个人, 那么他让他们用什么数来进行报数呢? 这个问题会越来越复杂, 因为随着过程进行, 圆圈在缩小。

使用【Task3】的循环链表创建一个应用来模拟这个问题。输入是组成圆圈的人数, 要报的数和第一个开始报的人的位置(通常是 1)0 输出是被消去的人的列表。当一个人出了圆圈, 再继续从他左边那个人开始计数(假设沿顺时针旋转)。这有一个例子。有 7 个人, 从 1 到 7, 从第一个人开始报数, 报到 4 出圆圈, 最后被消去的人的顺序是 4, 1, 6, 5, 7, 3。最后剩下的人是 20。

实验6 递归 (Recursion)

一. 实验目的

1. 理解递归结构及其算法思想;
2. 掌握运用递归解决程序设计问题的方法。

二. 实验要求

1. 仔细阅读实验指导书, 并于实验课前将大部分代码录入。
2. 提交实验报告, 报告按照模版编写, 内容包括:
 - (1) 基本操作步骤;
 - (2) 问题的回答;
 - (3) 算法的基本思路、自己编写的代码和运行结果截图。
3. 完成 Task1 可得 C, 完成 Task1、Task2 可得 B, 完成 Task1、Task2、Task3 可得 A。

三. 实验内容

【Task1】修改并拓展教材 Listing 中的程序

- 1). 在 triangle.java 程序(Listing 6.1)中, 删除基值条件的代码:

```
if(n==1)
    return 1;
else
```

然后运行程序, 看看会发生什么?分析其发生的原因。

- 2). 重写 mergeSort.java 程序(Listing 6.5)中的 main()部分: 给一个数组添加 10000, 20000, 30000, 40000, 50000 个随机数。运行程序来给这些数排序, 并且和第 3 章“简单排序”中的排序算法的速度进行比较。在报告中给出实验结果(整理为表格或者 Excel 图表), 并进行分析(时间、空间复杂度两方面)。

【Task2】应用递归的算法来实现求一个数的乘方

编写递归的 power()方法以及一个 main()来测试它。

【Task3】用递归的方法来显示由一群人组队的所有可能方案

(在 n 个每次挑 k 个 (n 大于等于 k))。

编写递归的 showTeams()方法和一个 main()方法来提示用户输入人群的人数以及组队的人数, 以此来作为 showTeams()的参数, 然后显示所有的组合。

提示:

(1) 组合数的递推公式: $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$;

(2) 用字符 A~Z, a~z 代表每个组员, 每队表示为一个字符串。
注意组合数并不是组合方案, 如何用递归来求出所有组合方案?

实验7 高级排序 (Advanced Sorting)

一. 实验目的

1. 理解希尔排序和快速排序的算法思想;
2. 了解快速排序中枢纽 pivot 选择的策略。

二. 实验要求

1. 仔细阅读实验指导书, 并于实验课前将大部分代码录入。
2. 提交实验报告, 报告按照模版编写, 内容包括:
 - (1) 基本操作步骤;
 - (2) 问题的回答;
 - (3) 算法的基本思路、自己编写的代码和运行结果截图。
3. 完成 Task1 可得 C, 完成 Task1、Task2 可得 B, 完成 Task1、Task2、Task3 可得 A。

三. 实验内容

【Task1】操作 Workshop 和修改教材 Listing 中的程序

1. 观察在 Partition 专题 Applet 中运行 100 个逆序排列的竖条的情况。它的结果是基本有序的吗?
2. 修改 shellSort.java 程序(Listing 7.1), 使它在完成每一趟 n-增量排序后显示数组的完整内容。数组足够小, 可以只显示在一行中。分析这些中间步骤, 看看算法的执行是否和分析的一样。
3. 修改 shellSort.java 程序(Listing 7.1)和 quickSort3.java 程序(Listing 7.5), 对大一些的数组排序, 并且比较它们的速度。同时, 比较它们和第 3 章中排序算法的速度。

【Task2】修改 partition.java (Listing 7.2) 程序

修改程序, 使 partitionIt()方法总是用具有最大的下标值的数组(最右)数据项作为枢纽, 而不是用任意一个数据项。并确保程序对三个或少于三个数据项的数组也能执行。

【Task3】扩展 Task1, 实现在一个数组中快速查找中心值数据项的算法。

在数组共 N 个元素中, 找到第 N/2 大 (小) 的元素。(注意: 不一定是索引为 N/2 的元素!)

例: 数组为{80, 20, 90, 77, 65, 60, 5}, 共有 7 个元素, 则其中心值为 65。

要求: 不能用先排序再选择中心值的办法

算法提示:

情形 1: 假设在 **Partition** 数据时, 若发现枢纽最后停在数组中间的位置上。工作就完成了! 枢纽右边的所有数据项都大于(或等于)枢纽, 而所有左边的数据项都小于(或等于)枢纽, 所以如果枢纽停在数组正中间的位置上, 那么它就是中心值。

情形 2: 枢纽通常是不会恰好停在数组中间位置上的, 但是可以通过再划分包含了中间位置数据项的分组来找到它。可以使用类似快速排序的递归调用, 但是只用来划分子数组, 而不是对整个数组排序。当找到中心值数据项时这个过程就停止, 而不必等数组排序完成之后才停止。

实验8 二叉树 (Binary Tree)

一. 实验目的

1. 理解二叉树的定义及其遍历算法;
2. 掌握二叉树的存储表示方法。

二. 实验要求

1. 仔细阅读实验指导书, 并于实验课前将大部分代码录入。
2. 提交实验报告, 报告按照模版编写, 内容包括:
 - (1) 基本操作步骤;
 - (2) 问题的回答;
 - (3) 算法的基本思路、自己编写的代码和运行结果截图。
3. 完成 Task1 可得 C, 完成 Task1、Task2 可得 B, 完成 Task1、Task2、Task3 可得 A。

三. 实验内容

【Task1】操作 Workshop 和修改教材 Listing 中的程序

1. 用二叉树专题 Applet 建立 20 棵树, 试统计出现严重不平衡树[附]有多少棵?
2. 删除二叉搜索树中的节点时会有各种可能性, 请画出 UML 的活动图 (附 2 参考)。应该详细地表示前面讲的三种情况: 包括左右子节点后代的不同情况, 以及根删除的特殊情况。例如, 第一种情况(左和右子节点)时有两种可能性。活动图中每个路径最后的方框中应表示出在此种情况下如何删除节点。
3. 在各种可能的情况下 (分别列出), 用二叉树专题 Applet 删除节点。

【Task2】扩展 tree.java 程序(Listing 8.1)建立二叉树, 输出树节点。

参考 tree.java 程序(Listing 8.1), 扩展为能将用户输入的字符串建立二叉树(如 A、B、...等等)。每个字母在各自的节点中显示, 并输出到命令行。

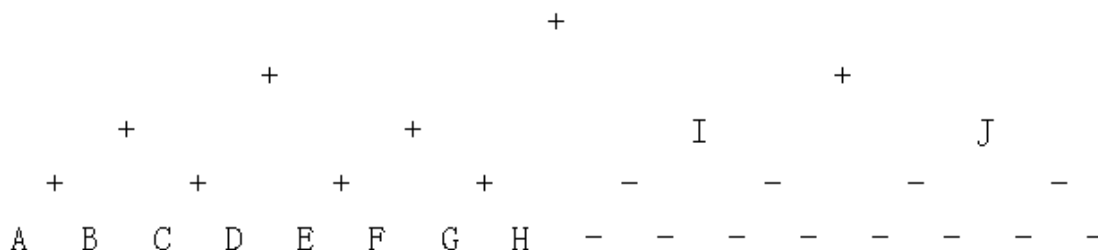
注意: 每个包含字母的节点是叶节点; 父节点用非字母标志 ‘+’ 表示, 每个父节点都要恰好有两个子节点。

不要担心树不平衡, 注意这不是建立二叉查找树, 没有快速的方法来查找节点。输出结果如下所示:

假设用户输入 “ABCDEFGG” 得到:



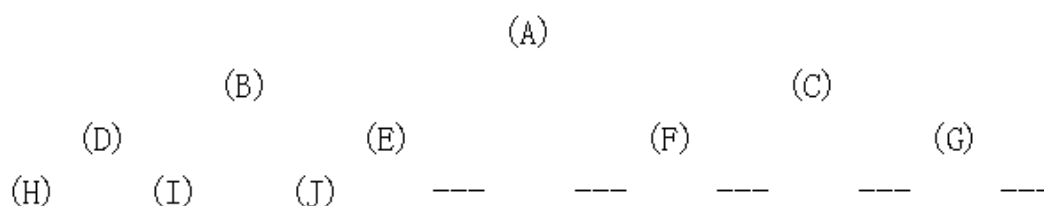
用户输入“ABCDEFGHJK”，得到：



提示：一种方法是建立一个树的数组。(一组没有连接的树称为森林。)用用户输入的每个字母作为一个节点。把每个节点作为一棵树，节点就是根。先把这些单节点的树放到数组中。接着建立一棵以‘+’为根的树，两个单节点树为它的子节点。依次把数组中的单节点树加到这棵大点的树中。不要担心它是不是非平衡树。实际操作中可以用新生成的树覆盖已被合并的树所占单元中。原来 Listing 程序中 find(), insert() 和 delete() 方法只用于 Binary Search Tree 树，可以删掉。保留 displayTree() 方法和遍历方法，在 main 方法中调用。

【Task3】进一步扩展 tree.java 程序(Listing 8.1)建立二叉树，输出树节点。

还是从 tree.java 程序开始，根据用户输入的字符创建树。这次，建立完全树——除了底层的最右边可能为空，其他层节点完全满的树。字母要从上到下以及从左到右有序，好像就是建立了一个字符的金字塔。例如，输入字符串 ABCDEFGHIJ 后，创建得到的树会排列成下面的样子：其中 A 为根，H、J 分别为 D 的左右叶子，J 为 E 的左叶子。



提示：建立这种树的一种方法是从上到下。从建立一个作为最后树的根 of 节点开始。如果节点编号和字母排列顺序一样，1 是根，则编号为 n 的节点的左子节点的编号为 2*n，右子节点的编号为 2*n+1。可以用递归方法，创建两个子节点并在每个子节点中调用它自己。节点创建的顺序不需要和它们插入到树中的顺序相同。像编程作业中前面的题那样，可以删掉 Tree 类中搜索树的一些方法。

附 1:

二叉树的平衡因子 K 的计算方法:

$$K = \begin{cases} 1 & \text{当结点个数为0} \\ \frac{\log_2(N+1) \cdot 2Leaf}{Level+1 \cdot N+1} & \text{当结点个数不为0} \end{cases}$$

公式中各变量意义如下:

N -----树结点总数

$Leaf$ -----叶子结点总数

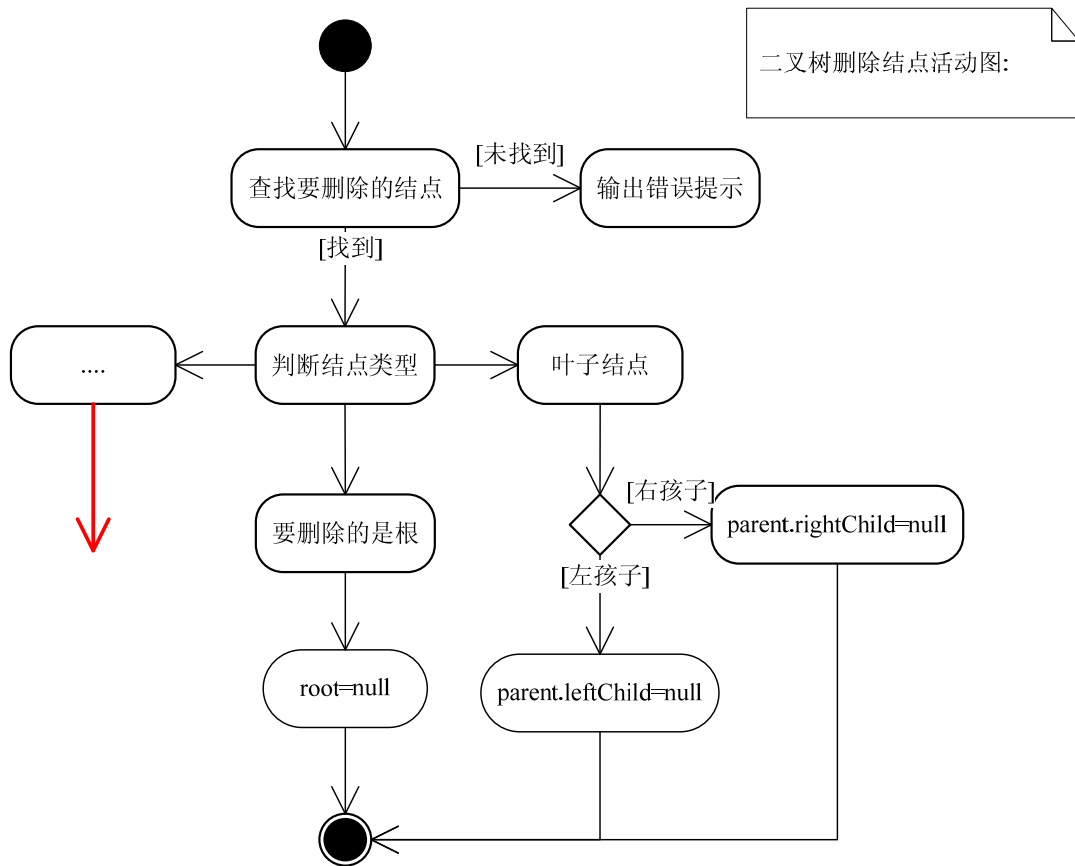
$Level$ -----树结点的最大层次 (根的 $level$ 为 0)

The closer to 0 the value is the more unbalanced 有以下三个等级:

$K=1$ -----perfectly balanced tree. (完全平衡)

$K < 0.5$ -----seriously unbalanced (严重不平衡)

附 2:



UML 活动图绘制参考资料:

<http://www.ibm.com/developerworks/cn/rational/tip-drawuml/>

绘图可用 Visio 绘制: 文件->新建->软件->UML 模型图->找到 UML 活动图, 直接拖放相关图元组装为活动图。

实验9 图 (Graph)

一. 实验目的

- 1.理解图的邻接矩阵存储方式;
- 2.掌握图的深度优先、广度优先、最小生成树算法;
- 3.了解图的邻接表存储方式及其遍历算法

二. 实验要求

1. 仔细阅读实验指导书, 并于实验课前将大部分代码录入。
2. 提交实验报告, 报告按照模版编写, 内容包括:
 - (1) 基本操作步骤;
 - (2) 问题的回答;
 - (3) 算法的基本思路、自己编写的代码和运行结果截图。
3. 完成 Task1 可得 C, 完成 Task1、Task2 可得 B, 完成 Task1、Task2、Task3 可得 A。

三. 实验内容

【Task1】操作 Workshop

1. 用 GraphN 专题 Applet, 画具有五个顶点、七条边的图, 然后先不按 View 按钮, 写出图的邻接矩阵。做完后, 按 View 按钮, 比较结果是否一致。
2. 创建一个有五个顶点的邻接矩阵, 随机插入 0 和 1。不必考虑对称。现在, 不使用 View 按钮, 用 GraphD 专题 Applet 创建对应的有向图。完成后, 点击 View 按钮比较, 看图与邻接矩阵是否一致。

【Task2】修改 bfs.java 程序(Listing 13.2), 通过 BFS 找到最小生成树

在 mst.java 程序(Listing 13.3)中, 这个工作是由 DFS 完成的。在 main()中, 创建有 9 个顶点和 12 条边的图, 然后用你写的程序 (基于 BFS) 找到最小生成树。

【Task3】扩展 dfs.java 程序(Listing 13.1), 用邻接表存储方式实现 DFS。

修改 dfs.java 程序(Listing 13. 1), 用邻接表而不是邻接矩阵执行 DFS。可以通过修改第 5 章 linkList2.java 程序(Listing 5. 2)的 Link 类和 LinkList 类得到链表。修改 LinkList 中的 find()方法, 搜索一个未访问的顶点, 而不是一个关键字值。

附录 1 Java 学习资源

1、Java 程序编译错误原因

http://java.syntaxerrors.info/index.php?title=All_Syntax_errors&oldid=305

2、Java 学习网站

<http://java.javaeye.com/>

3、开源软件下载

<http://sourceforge.net/>

附录 2 Java 编程命名规范

定义规范的目的是为了使项目的代码样式统一，使程序有良好的可读性。

包的命名 （全部小写，由域名定义）

Java 包的名字都是由小写单词组成。但是由于 Java 面向对象编程的特性，每一名 Java 程序员都可以编写属于自己的 Java 包，为了保障每个 Java 包命名的唯一性，在最新的 Java 编程规范中，要求程序员在自己定义的包的名称之前加上唯一的前缀。由于互联网上的域名称是不会重复的，所以程序员一般采用自己在互联网上的域名称作为自己程序包的唯一前缀。

例如：cn.zhku.cs

类的命名 （单词首字母大写）

根据约定，Java 类名通常以大写字母开头，如果类名称由多个单词组成，则每个单词的首字母均应为大写，例如 TestPage；如果类名称中包含单词缩写，则这个所写词的每个字母均应大写，如：XMLExample，还有一点命名技巧就是由于类是设计用来代表对象的，所以在命名类时应**尽量选择名词**。

例如：Graphics

方法的命名 （首字母小写，字母开头大写）

方法的名字的第一个单词应以小写字母作为开头，后面的单词则用大写字母开头。方法名一般以动词开头。

例如：drawImage

常量的命名 （全部大写，常加下划线）

常量的名字应该都使用大写字母，并且指出该常量完整含义。如果一个常量名称由多个单词组成，则应该用下划线来分割这些单词。

例如：MAX_VALUE

参数的命名

参数的命名规范和方法的命名规范相同，而且为了避免阅读程序时造成迷惑，

请在尽量保证参数名称为一个单词的情况下使参数的命名尽可能明确。

Javadoc 注释

Java 除了可以采用我们常见的注释方式之外，Java 语言规范还定义了一种特殊的注释，也就是我们所说的 Javadoc 注释，它是用来记录我们代码中的 API 的。Javadoc 注释是一种多行注释，以/**开头，而以*/结束，注释可以包含一些 HTML 标记符和专门的关键词。使用 Javadoc 注释的好处是编写的注释可以被自动转为在线文档，省去了单独编写程序文档的麻烦。

例如：

```
/**
 *   This   is   an   example   of
 *   Javadoc
 *   @author   darchon
 *   @version   0.1,   10/11/2002
 */
```

在每个程序的最开始部分，一般都用 Javadoc 注释对程序的总体描述以及版权信息，之后在主程序中 可以为每个类、接口、方法、字段添加 Javadoc 注释，每个注释的开头部分先用一句话概括该类、接口、方法、字段所完成的功能，这句话应单独占据一行以突出其概括作用，在这句话后面可以跟 随更加详细的描述段落。在描述性段落之后还可以跟随一些以 Javadoc 注释标签开头的特殊段落，例如上面例子中的 @author 和 @version，这 些段落将在生成文档中以特定方式显示。

变量和常量命名

变量的命名

在变量命名时要注意以下几点：

- 选择有意义的名字，注意每个单词首字母要大写。
- 在一段函数中不使用同一个变量表示前后意义不同的两个数值。
- i、j、k 等只作为小型循环的循环索引变量。
- 避免用 Flag 来命名状态变量。

- 用 `is` 来命名逻辑变量，如：`isFound`。通过这种给布尔变量肯定形式的命名方式，使得其它开发人员能够更为清楚的理解布尔变量所代表的意义。

- 如果需要的话，在变量最后附加计算限定词，如：`curSalesSum`。

- 命名不相包含，`curSales` 和 `curSalesSum`。

- `static final` 变量(常量)的名字应该都大写，并且指出完整含义。

- 如果需要对变量名进行缩写时，一定要注意整个代码中缩写规则的一致性。例如，如果在代码的某些区域中使用 `intCnt`，而在另一些区域中又使用 `intCount`，就会给代码增加不必要的复杂性。建议变量名中尽量不要出现缩写。

- 通过在结尾处放置一个量词，就可创建更加统一的变量，它们更容易理解，也更容易搜索。例如，请使用 `strCustomerFirst` 和 `strCustomerLast`，而不要使用 `strFirstCustomer` 和 `strLastCustomer`。常用的量词后缀有：`First`（一组变量中的第一个）、`Last`（一组变量中的最后一个）、`Next`（一组变量中的下一个变量）、`Prev`（一组变量中的上一个）、`Cur`（一组变量中的当前变量）。

附录 3 Java 编程的注意事项

- 为每个变量选择最佳的数据类型

这样即能减少对内存的需求量，加快代码的执行速度，又会降低出错的可能性。用于变量的数据类型可能会影响该变量进行计算所产生的结果。在这种情况下，编译器不会产生运行期错误，它只是迫使该值符合数据类型的要求。这类问题极难查找。

- 尽量缩小变量的作用域。

如果变量的作用域大于它应有的范围，变量可继续存在，并且在不再需要该变量后的很长时间内仍然占用资源。它们的主要问题是，任何类中的任何方法都能对它们进行修改，并且很难跟踪究竟是何处进行修改的。占用资源是作用域涉及的一个重要问题。对变量来说，尽量缩小作用域将会对应用程序的可靠性产生巨大的影响。

- 常量取代数字

在 JAVA 代码中，无论什么时候，均提倡应用常量取代数字、固定字符串。也就是说，程序中除 0, 1 以外，尽量不应该出现其他数字。常量可以集中在程序开始部分定义或者更宽的作用域内，名字应该都使用大写字母，并且指出该常量完整含义。如果一个常量名称由多个单词组成，则应该用下划线“_”来分割这些单词如：NUM_DAYS_IN_WEEK、MAX_VALUE。