哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2022 年春季学期
计算学部《软件构造》课程

Lab 1 实验报告

姓名	王炳轩
学号	120L022115
班号	2003007
电子邮件	1487819688@qq.com
手机号码	13552161805

目录

1 实验目标概述

本次实验通过求解三个问题，训练基本 Java 编程技能，能够利用 Java OO 开发基本的功能模块，能够阅读理解已有代码框架并根据功能需求补全代码，能够为所开发的代码编写基本的测试程序并完成测试，初步保证所开发代码的正确性。

另一方面，利用 Git 作为代码配置管理的工具，学会 Git 的基本使用方法。

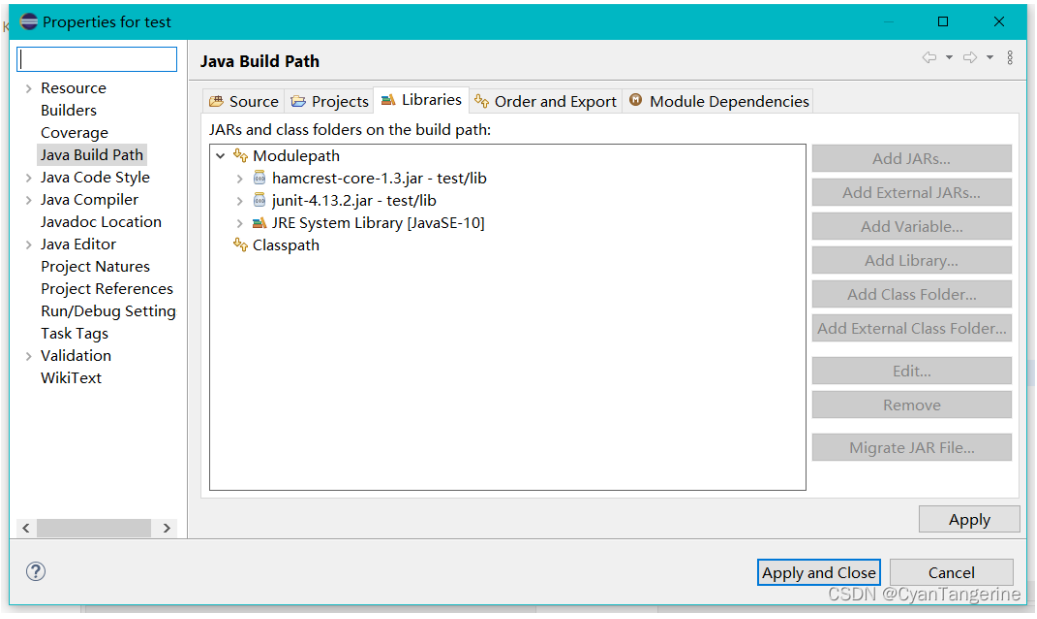
- 基本的 Java OO 编程
- 基于 Eclipse IDE 进行 Java 编程
- 基于 JUnit 的测试
- 基于 Git 的代码配置管理

2 实验环境配置

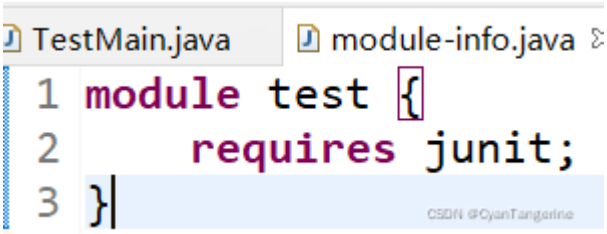
安装 JDK、Eclipse。下载 Junit 并配置。

1、下载 junit 的 jar 包：Download and Install · junit-team/junit4 Wiki · GitHub

2、在要使用 Junit 的 project 名上，右击菜单，点击 properties、java build path、libraries、Modulepath，点击 Add JARs，添加刚刚下载的 2 个 jar。



3、在 module_info 中添加 require



4、在需要使用 junit 的类中 import

```
1 package test;
2 import org.junit.Test;
3
```


在这里给出你的 GitHub Lab1 仓库的 URL 地址。
https://github.com/ComputerScienceHIT/HIT-Lab1-120L022115

3 实验过程

3.1 Magic Squares

一个由正整数构成的矩阵，无论行或列以及两条对角线的和都是同一个数。

3.1.1 isLegalMagicSquare()

1、读入文件

```
try {
        Scanner sn = new Scanner(new
FileInputStream(fileName));
        String str = "";
        while (sn.hasNextLine()) {
            str += sn.nextLine() + "\n";
        }
    } catch (FileNotFoundException e) {
        System.out.println(fileName+": 错误：未找到文件：" +
fileName);
    }
```

2、转换为int[][]

```
        String lines[];
        try {
            lines = str.split("\n");
        } catch (PatternSyntaxException e) {
            sn.close();
            System.out.println(fileName+": 错误：文件格式错误!
行应用换行符分隔！");
            //e.printStackTrace();
            return false;
        }
        String thisline[];
        int width;
```

```
            int height = lines.length;
            int num[][] = new int[height][height];
            int i = 0, j = 0;
            for (i = 0; i < height; i++) {
                try {
                    thisline = lines[i].split("\t");
                } catch (PatternSyntaxException e) {
                    sn.close();
                    System.out.println(fileName+": 错误: 文件格式错
误! 数字应用tab符分隔! ");
                    //e.printStackTrace();
                    return false;
                }

                width = thisline.length;
                if (width != height) {
                    System.out.println(fileName+":
isNotMagicSquare: 行列数不相等: "+height+"行" + width+"列");
                    sn.close();
                    return false;
                }

                for (j = 0; j < width; j++) {
                    try {
                        num[i][j] = Integer.valueOf(thisline[j]);
                        if (num[i][j] <= 0) {
                            System.out.println(fileName+":
isNotMagicSquare: 第" + i + "行第" + j + "列输入为非正整数: " +
num[i][j]);
                            sn.close();
                            return false;
                        }
                    } catch (NumberFormatException e) {
                        System.out.println(fileName+": 错误: 文本到整
数格式转换错误: " + thisline[j]);
                        sn.close();
                        //e.printStackTrace();
                        return false;
                    }
                }
            }
```

3、计算行、列、对角线的和并判断

```
            // row sum
            int constnum = -1;
            int sum = 0;
            for (i = 0; i < height; i++) {
```

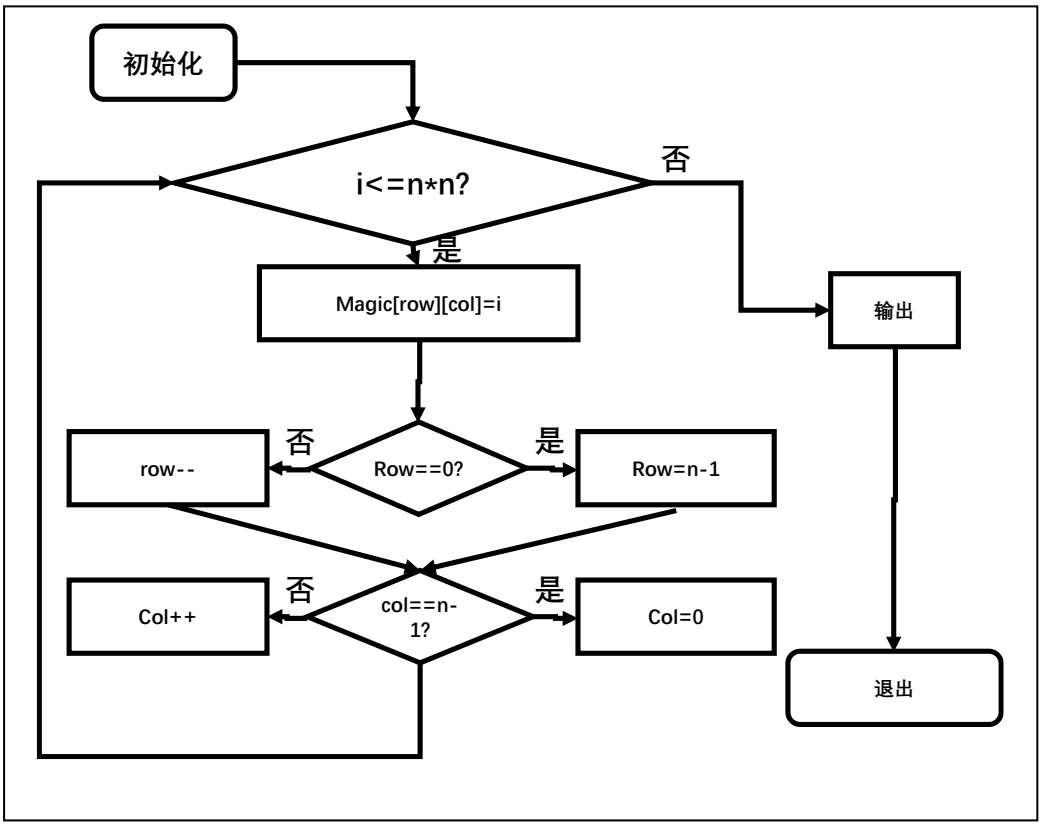
```
                sum = 0;
                for (j = 0; j < height; j++) {
                    sum += num[i][j];
                }
                if (constnum == -1)
                    constnum = sum;
                else if (constnum != sum) {
                    sn.close();
                    System.out.println(fileName+": 
isNotMagicSquare: 行和错误! 第" + i + "行和为"  + sum + ", 应为"
+ constnum);
                    //System.out.println("row" + ' ' + i + ' ' + j
+ ' ' + sum + ' ' + constnum);
                    return false;
                }
            }

            // col sum
            for (i = 0; i < height; i++) {
                sum = 0;
                for (j = 0; j < height; j++) {
                    sum += num[j][i];
                }
                if (constnum != sum) {
                    sn.close();
                    System.out.println(fileName+": 
isNotMagicSquare: 列和错误! 第" + i + "列和为"  + sum + ", 应为"
+ constnum);
                    return false;
                }
            }

            // dia sum
            sum = 0;
            for (i = 0; i < height; i++) {
                sum += num[i][i];
            }
            if (constnum != sum) {
                sn.close();
                System.out.println(fileName+": isNotMagicSquare:
主对角线和错误!  和: " + sum + ", 应该是: " + constnum);
                return false;
            }
            sum = 0;
            for (i = 0; i < height; i++) {
                sum += num[i][height - 1 - i];
            }
            if (constnum != sum) {
```

```
            sn.close();
            System.out.println(fileName+": isNotMagicSquare:
副对角线和错误！ 和: " + sum + ", 应该是: " + constnum);
            return false;
        }
        return true;
```

3.1.2 generateMagicSquare()

按步骤给出你的设计和实现思路/过程/结果。

1、分析函数并给出流程图与注释



这样生成可以使每一行、列、主副对角线的和都是相同的值。

```
    //生成一个n*n的MagicSquare并打印，如果成功则返回True，否则返回
假
    public static boolean generateMagicSquare(int n) {
        int magic[][] = new int[n][n];//新建n*n数组
        int row = 0, col = n / 2, i, j, square = n * n;//初始化
        //从第一行的中心开始
        for (i = 1; i <= square; i++) {//循环n个数
            magic[row][col] = i;
            if (i % n == 0) //沿着对角线生成递增的序列
                row++; //这个对角线填满了，换下一个对角线
            else {//沿着对角线，row--，col++，触碰到边界返回起始
```

```
                if (row == 0)
                    row = n - 1;
                else
                    row--;
                if (col == (n - 1))
                    col = 0;
                else
                    col++;
            }
        }
        for (i = 0; i < n; i++) {//逐行输出数组的值
            for (j = 0; j < n; j++)
                System.out.print(magic[i][j] + "\t");
            System.out.println();
        }
        return true;
        }
```

2、对该函数做扩展:

(1) 将产生的 magic square 写入文件\src\P1\txt\6.txt 中;

添加如下代码:

```
        try {
            FileWriter fw = new
FileWriter(".\\src\\P1\\txt\\6.txt");
            for (i = 0; i < n; i++) {//逐行输出数组的值
                for (j = 0; j < n; j++){
                    fw.write(magic[i][j] + "\t");
                }
                fw.write("\n");
            }
            fw.close();
        } catch (IOException e) {
            System.out.println("文件写出失败！");
            return false;
        }
```

(2) 当输入的 n 不合法时(n 为偶数、n 为负数等), 不要该函数抛出异常并非法退出, 而是提示错误并"优雅的"退出——函数输出 false 结束。

添加如下代码:

```
if(n%2==0||n<0) {
    System.out.println("n必须是正奇数！");
    return false;
}
```

(3) 利用你前面已经写好的 isLegalMagicSquare()函数, 在 main()函数判断该函数新生成的文本文件 6.txt 是否符合 magic square 的定义。

添加如下代码:

```
    Boolean testRes =
```

```
MagicSquare.isLegalMagicSquare(".\\src\\P1\\txt\\6.txt");
        System.out.println(testRes);
```

3.2 Turtle Graphics

3.2.1 Problem 1: Clone and import

如何从 GitHub 获取该任务的代码、在本地创建 git 仓库、使用 git 管理本地开发。

git init 或 git clone URL

3.2.2 Problem 3: Turtle graphics and drawSquare

```
turtle.forward(sideLength);
turtle.turn(90);
turtle.forward(sideLength);
turtle.turn(90);
turtle.forward(sideLength);
turtle.turn(90);
turtle.forward(sideLength);
turtle.turn(90);
```

3.2.3 Problem 5: Drawing polygons

```
if(sides<=2) throw new RuntimeException("边数必须大于 2");
        return (sides-2)*180.0/sides;

if(angle>=180||angle<=0) throw new RuntimeException("角度必须在(0,180)之间");
        return (int) Math.round(360/(180-angle));

double angle = TurtleSoup.calculateRegularPolygonAngle(sides);
 for(int i=0;i<sides;i++) {
        turtle.forward(sideLength);
        turtle.turn(180-angle);
 }
```

3.2.4 Problem 6: Calculating Bearings

```
double dx = -(currentX-targetX);
double dy = -currentY+targetY;
double at = Math.abs(Math.atan2(Math.abs(dx), Math.abs(dy)));
at = ((Math.round(at/Math.PI*180*1000)/1000.0));
double k = dx*(double)dy;
double da = at-currentBearing;
if(k>0) {
```

```
            if(dx<0&&dy<0) return doublemod(180+da,360);
            return doublemod(da,360);
        }else if(k<0) {
            da = (at+currentBearing);
            if(dx<0&&dy>0) return doublemod(180-da,360);
            return doublemod(360-da,360);
        }else {
            if(dx==0 && dy !=0) {
                double db = 180-currentBearing;
                return dy>0?doublemod(db+180,360):(doublemod(db,360));
            }else if(dy==0 && dx!=0) {
                double db = dx>0?90-currentBearing:270-currentBearing;
                return doublemod(db,360);
            }else {
                return 0;
            }
        }
    private static double doublemod(double d,int mod) {
        if(d>=0)
            return d-((int)d/mod)*mod;
        else
            return d-((int)d/mod)*mod+360;
    }
    if(xCoords.size()<2) return null;
            if(xCoords.size()!=yCoords.size()) return null;
            int lx = xCoords.get(0), ly = yCoords.get(0);
            double deg = 0;
            List<Double> list = new ArrayList<Double>();
            for(int i=1;i<xCoords.size();i++) {
                int x = xCoords.get(i), y= yCoords.get(i);
                double turn = calculateBearingToPoint(deg,lx,ly,x,y);
                list.add(turn);
                lx=x;ly=y;deg=doublemod(turn+deg,360);
            }
            return list;
```

3.2.5 Problem 7: Convex Hulls

使用 GIFT-WARPPING 算法：

```
Set<Point> set = new HashSet<Point>();
            //System.out.println(points.toString());
            //System.out.println(points.toArray().toString());
            Point[] list = points.toArray(new Point[0]);

            if(list.length==0) return set;
```

```
Point p = list[0];
int len = list.length;
double x=p.x(),y=p.y();
int lp = 0;
for(int i=1;i<len;i++) {
	p = list[i];
	if(p.x()<x) {
		lp=i;
		x=p.x();
	}
}
set.add(list[lp]);
p = list[lp];


int minpi=lp;
Point np,minp=p,startp=p;
double maxdistance=0,distance=0;
double deg=0,turn=0,minturn=360;

//list[lp] = null;
do {
	double lx = p.x();
	double ly = p.y();
	minturn=360;
	maxdistance=0;
	for(int j=0;j<len;j++) {
		np = list[j];
		//if(np==null) continue;
		x=np.x();y=np.y();
		if(np.equals(p)) continue;
		turn = calculateBearingToPoint(deg,lx,ly,x,y);
		if(turn<=minturn) {
			double dx=x-lx,dy=y-ly;
			distance = Math.sqrt(dx*dx+dy*dy);
			if(turn==minturn) {
				if(distance>maxdistance) {
					maxdistance = distance;
					minpi = j;
					minp = np;
					minturn = turn;
				}
			}else {
				minpi = j;
```

```
                        minp = np;
                        minturn = turn;
                        maxdistance = distance;
                    }

                }

            }
            if(minp.equals(startp)) break;
            set.add(minp);
            deg=doublemod(minturn+deg,360);
            lx=x;ly=y;
            //list[minpi] = null;
            p = minp;
        }while(!p.equals(startp));
        return set;
```

3.2.6 Problem 8: Personal art

用以下代码写出学号 120L022115：

```
turtle.color(PenColor.CYAN);
        int h = 40,hh=20,s=10;
        turtle.turn(180);
        //1
        turtle.forward(h);

        turtle.turn(270);
        turtle.forward(s);
        //2
        turtle.forward(hh);
        turtle.turn(180);
        turtle.forward(hh);
        turtle.turn(90);
        turtle.forward(hh);
        turtle.turn(90);
        turtle.forward(hh);
        turtle.turn(270);
        turtle.forward(hh);
        turtle.turn(270);
        turtle.forward(hh);
        turtle.turn(180);
        turtle.forward(hh+s);
        //0
        turtle.forward(hh);
        turtle.turn(90);
```

```
turtle.forward(h);
turtle.turn(90);
turtle.forward(hh);
turtle.turn(90);
turtle.forward(h);
turtle.turn(90);
turtle.forward(hh);

turtle.forward(s);
//L
turtle.turn(90);
turtle.forward(h);
turtle.turn(270);
turtle.forward(hh+s);
//0
turtle.forward(hh);
turtle.turn(270);
turtle.forward(h);
turtle.turn(270);
turtle.forward(hh);
turtle.turn(270);
turtle.forward(h);
turtle.turn(270);
turtle.forward(hh+s);
//2
turtle.forward(hh);
turtle.turn(180);
turtle.forward(hh);
turtle.turn(90);
turtle.forward(hh);
turtle.turn(90);
turtle.forward(hh);
turtle.turn(270);
turtle.forward(hh);
turtle.turn(270);
turtle.forward(hh);
turtle.turn(180);
turtle.forward(hh+s);
//2-1
turtle.forward(hh);
turtle.turn(90);
turtle.forward(hh);
turtle.turn(90);
turtle.forward(hh);
```

```
turtle.turn(270);
turtle.forward(hh);
turtle.turn(270);
turtle.forward(hh+s);
//1
turtle.turn(270);
turtle.forward(h);
//1
turtle.turn(90);
turtle.forward(s);
turtle.turn(90);
turtle.forward(h);
//5
turtle.turn(270);
turtle.forward(hh+s);
turtle.turn(270);
turtle.forward(hh);
turtle.turn(270);
turtle.forward(hh);
turtle.turn(90);
turtle.forward(hh);
turtle.turn(90);
turtle.forward(hh);
```

3.2.7 Submitting

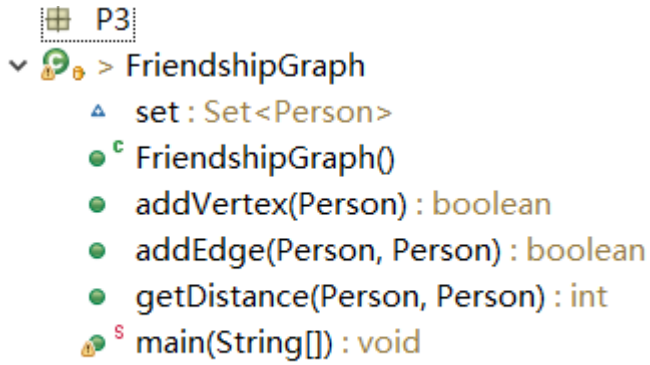
如何通过 Git 提交当前版本到 GitHub 上你的 Lab1 仓库。

法 1：
git commit -a “Target 2”
git push

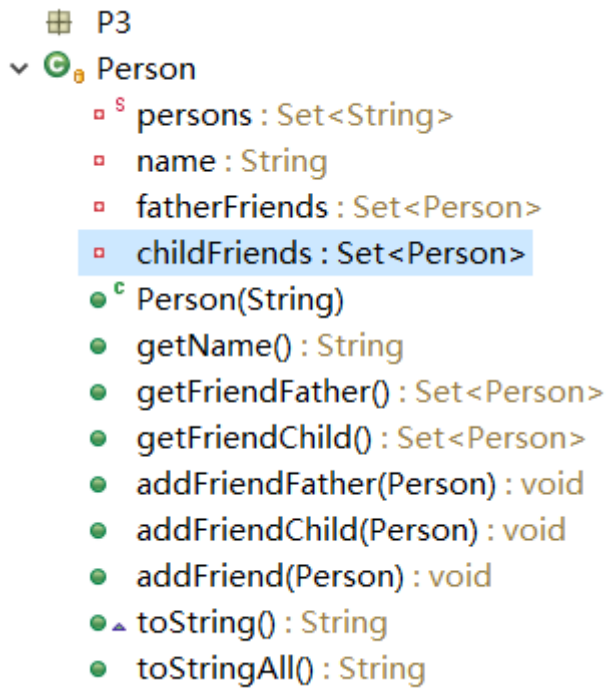
法 2：
安装 Github Desktop
登录
按照页面提示进行操作

3.3 Social Network

3.3.1 设计/实现 FriendshipGraph 类



3.3.2 设计/实现 Person 类



3.3.3 设计/实现客户端代码 main()

通过设计、编写代码，实现了与用户的命令行交互：

```
欢迎来到友情图!
使用以下命令来操作：
new name //创建一个新人物
add name //把一个人物加入图
dis name1 name2 //计算两个人物的距离
edg name1 name2 //添加name1到name2的友情线
info name1 //查看name1人的信息
exit //退出
>>new 1
1 已创建
>>new 1
已有名字为1的人，请更换名字后再试
>>new 2
2 已创建
>>add 1
1 已添加
>>add2
命令错误!
```

```
>>edg 2
命令错误!
>>dis 1 2
1->2没有路径!
>>edg 1 2
1->2 已添加
>>dis 1 2
1->2的距离是：1
>>dis 1 1
1->1的距离是：0
>>dis 2 2
2->2的距离是：0
>>info 1
{
        name=1
        childs=2
        fathers=
}
```

3.3.4 设计/实现测试用例

分别测试 getDistance()、addVertex()、addEdge()。使用 assertEquals()。

```
TestGetDistance：  {
        assertEquals(1,graph.getDistance(rachel, ross));
        //should print 1
        assertEquals(2,graph.getDistance(rachel, ben));
        //should print 2
        assertEquals(0,graph.getDistance(rachel, rachel));
        //should print 0
        assertEquals(-1,graph.getDistance(rachel, kramer));
        //should print -1
    }
    TestGetDistance2：{
```

```
        assertEquals(1,graph.getDistance(rachel, ross));
        //should print 1
        assertEquals(2,graph.getDistance(rachel, ben));
        //should print 2
        assertEquals(0,graph.getDistance(rachel, rachel));
        //should print 0
        assertEquals(3,graph.getDistance(rachel, kramer));
        //should print -1
    }
    TestAddVertex：  {

        assertEquals(true,graph.addVertex(rachel));
        assertEquals(true,graph.addVertex(ross));
        assertEquals(true,graph.addVertex(ben));
        assertEquals(true,graph.addVertex(kramer));
    }
    TestAddEdge：  {

        assertEquals(true,graph.addEdge(rachel, ross));
        assertEquals(true,graph.addEdge(rachel, ben));
        assertEquals(true,graph.addEdge(rachel, kramer));
        assertEquals(true,graph.addEdge(ben, ross));
        assertEquals(true,graph.addEdge(kramer, ross));
    }
```

4 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

日期	时间段	任务	实际完成情况
2022-04-19	18:30-21:30	完成 3.1 Magic Squares	按计划完成
2022-04-26	18:00-23:00	完成 3.2 Turtle Graphics	按计划完成
2022-04-27	15:30-17:00 18:00-21:00	完成 3.3 Social Network	按计划完成

5 实验过程中遇到的困难与解决途径

遇到的困难	解决途径
空指针异常、数组越界、计算错误等	单步调试解决问题

6 实验过程中收获的经验、教训、感想

6.1 实验过程中收获的经验和教训（必答）

常规的编程练习。
学会了 Junit、Git。
重温了数据结构与算法和算法设计与分析。

6.2 针对以下方面的感受（必答）

(1) Java 编程语言是否对你的口味？与你熟悉的其他编程语言相比，Java 有何优势和不足？

对。面向对象，优美、严谨、健壮、不易出错。很多规范易懂的语法，但也限制了发挥，有时就成了条条框框。Java 的运行速度相对较慢因为 Java 是靠虚拟机运行，所以相对于其他语言(汇编,C,C++)编写的程序慢，因为它不是直接执行机器码。因为 Java 是跨平台的，所以不能和底层打交道。Java 使用虚拟机来实现，不能接近操作系统，也就不能和操作系统的底层打交道了。不够灵活因为 Java 删除了指针，所以不如 C/C++等语言灵活。

(2) 关于 Eclipse 或 IntelliJ IDEA，它们作为 IDE 的优势和不足；

优势：语法即时报错，代码自动补全，格式化，图形化界面。
不足：自动补全有时弹不出来。

(3) 关于 Git 和 GitHub，是否感受到了它在版本控制方面的价值；

是。

(4) 关于 CMU 和 MIT 的作业，你有何感受；

一步一步完成作业的同时，教会你很多技巧与知识。

(5) 关于本实验的工作量、难度、deadline；

工作量：稍多。
难度：中等，部分算法部分需要推敲。
deadline：合适。

(6) 关于初接触“软件构造”课程；

学会了一些软件，理清了一些思想。