《面向对象程序设计与训练》实验报告

信息 学院 计算机科学与技术 专业 19 级

实验时间 2020 年 12 月 7 日

姓名 白文强 学号 20191060064

实验名称 CSP 认证 2018 年 9 月测试真题

实验成绩

一、实验目的

1．理解 Java 语言是如何体现面向对象编程基本思想，
2．了解类的封装方法，以及如何创建类和对象，
3．了解成员变量和成员方法的特性。

了解类的继承性和多态性的作用。

二、实验仪器设备及软件

个人 PC， IDEA

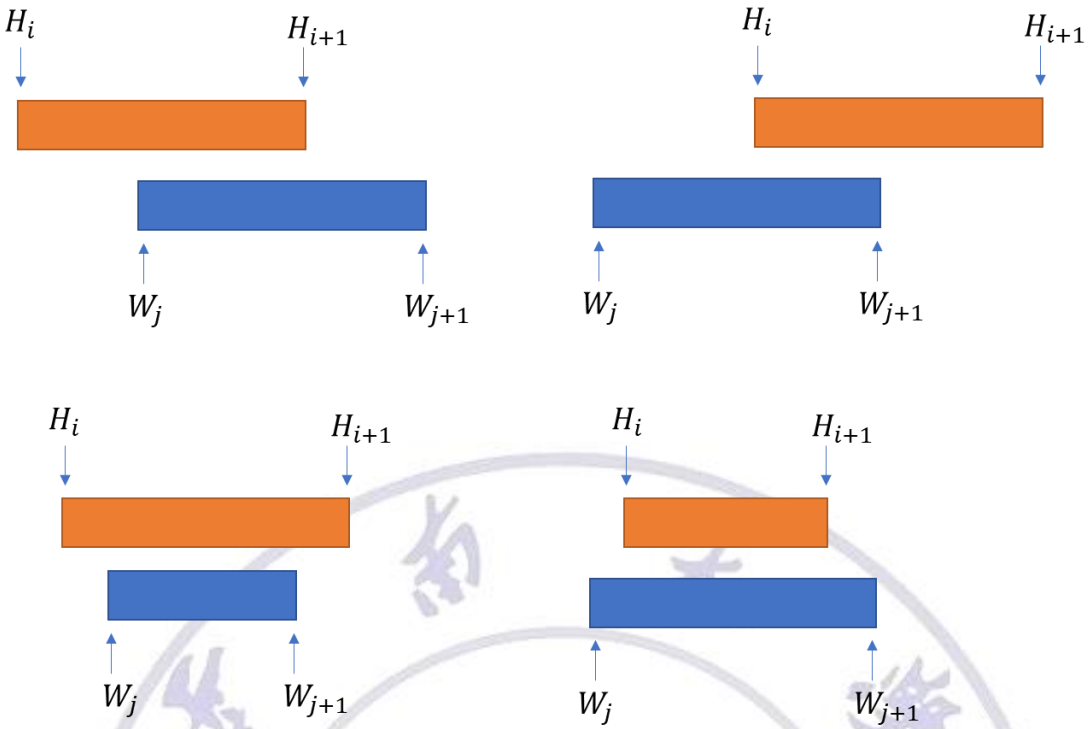
三、实验方案

卖菜：

此题比较简单，只需要注意计算第二天菜价时需要用一个新数组存放结果，而不能直接在一个数组中进行操作；此外还需要注意两个边界，计算第一家商户和最后一家商户第二天的价格时，要单独处理。

买菜：

此题是计算两个人同时空闲的时间，对于两个人的时间段，如果存在交叉，有以下几种情况：



我采用数组存放一个人的时间段，偶数下标存放开始，奇数下标存放结束，这样就完成了存储的要求；然后，固定 H 的一个时间段，对 W 的时间段进行遍历，出现时间段有交叉时，计算重合时间，直到出现W_j大于H_{i+1}时，就说明当前 H 的时间段下，与 W 重合的时间段计算完毕，退出内层循环，对 H 的时间段进行迭代。

由此，当对每一个 H 的时间段遍历都结束后，得出来的重合时间之和就是答案。

元素选择器：

Main 类中含有三个变量，tag 表示标签，id 表示 id，rank 表示元素的等级（rank 越大，等级越低，这是根据输入过程中元素前的点个个数计算的）。

对于单级查询，分为查询 id 和 tag 两种情况，只需要遍历每个元素，如果查到就记住元素的行数以及计数器更新加一。

对于多级查询，一级一级查找，例如查找 html div p，首先查找所有的 html，将其存放到集合里，接下来查询 div，依次取出集合中的元素，从该元素开始查询 div，当遇到等级大于等于取出的元素的等级时，退出循环，取出下一个元素继续向后查询。将查询到的结果放到另一个集合里。每个循环开始时，第一个集合会克隆第二个集合，第二个集合会清空用来记录新查找到的元素。

这样，直到找完最后一个元素，最后集合中的元素中存放的就是要查询的所有元素，至此就完成了查询过程。

四、实验步骤

卖菜：

```
1.  import java.util.Scanner;
2.
3.  public class Main {
4.      private static Scanner in = new Scanner(System.in);
5.
6.      public static void main(String[] args) {
7.          String[] str = in.nextLine().split(" ");    //读入第一行
8.          int n = Integer.parseInt(str[0]);
9.
10.         int[] price = new int[n];                           //原价格
11.         int[] newPrice = new int[n];                        //现价格
12.
13.         String[] str1 = in.nextLine().split(" ");   //第二行，每个商店的时菜
    价
14.         for(int i = 0; i < n; ++i) {
15.             price[i] = Integer.parseInt(str1[i]);
16.         }
17.         StringBuffer result = new StringBuffer("");
18.         //计算第二天价格
19.         for(int i = 0; i < n; ++i) {
20.             if (i == 0) {
21.                 newPrice[i] = (price[i] + price[i + 1]) / 2;
22.             } else if (i == n - 1) {
23.                 newPrice[i] = (price[i - 1] + price[i]) / 2;
24.             } else {
25.                 newPrice[i] = (price[i - 1] + price[i] + price[i + 1]) / 3;
26.             }
27.             result.append(newPrice[i]).append(" ");
28.         }
29.         String res = new String(result);
30.         //去除最后空格输出
31.         System.out.println(res.trim());
32.     }
33. }
```

买菜：

```java
import java.util.Scanner;

public class Main {
    private static Scanner in = new Scanner(System.in);

    public static void main(String[] args) {
        String lineFirst = in.nextLine();
        int n = Integer.parseInt(lineFirst);
        int[] HTime = new int[n * 2];
        int[] WTime = new int[n * 2];

        //数据读入
        String[] lineTwo;
        for(int i = 0; i < 2 * n; i += 2) {
            lineTwo = in.nextLine().split(" ");
            HTime[i] = Integer.parseInt(lineTwo[0]);
            HTime[i + 1] = Integer.parseInt(lineTwo[1]);
        }
        for(int i = 0; i < 2 * n; i += 2) {
            lineTwo = in.nextLine().split(" ");
            WTime[i] = Integer.parseInt(lineTwo[0]);
            WTime[i + 1] = Integer.parseInt(lineTwo[1]);
        }
        //计算可聊天的时间
        int sumTime = 0;
        for(int i = 0; i < 2 * n; i += 2) {
            for(int j = 0; j < 2 * n && WTime[j] <= HTime[i + 1]; j += 2) {
                if (HTime[i] <= WTime[j] && HTime[i + 1] >= WTime[j] && WTime[j + 1] >= HTime[i + 1])
                    sumTime += HTime[i + 1] - WTime[j];          //有交叉，W 靠后
                else if (HTime[i] >= WTime[j] && HTime[i] <= WTime[j + 1] && HTime[i + 1] >= WTime[j + 1])
                    sumTime += WTime[j + 1] - HTime[i];          //有交叉，H 靠后
                else if (HTime[i] <= WTime[j] && HTime[i + 1] >= WTime[j + 1])
                    sumTime = sumTime + (WTime[j + 1] - WTime[j]);      //W 包含 H
                else if (WTime[j] <= HTime[i] && WTime[j + 1] >= HTime[i + 1])
                    sumTime += HTime[i + 1] - HTime[i];          //H 包含 W
            }
        }
        System.out.println(sumTime);
    }
}
```

元素选择器：

```java
import java.util.*;

public class Main {
    private String tag; //标签
    private String id;  //id
    private int rank;   //等级，0 1 2 3 ，数字越高等级越低

    private static Scanner in;

    public static void main(String[] args) {
        in = new Scanner(System.in);
        List<Main> l = new LinkedList<>();          //存放每一个元素
        List<String> result = new ArrayList<>();    //存放输出结果

        String[] s = in.nextLine().split(" ");
        int n = Integer.parseInt(s[0]);    //标签个数
        int m = Integer.parseInt(s[1]);    //查询次数

        int i;
        String[] str2;

        //输入过程
        for (i = 0; i < n; ++i) {
            str2 = in.nextLine().split(" ");
            int preLength = str2[0].length();
            str2[0] = str2[0].replace(".", "");
            int afterLength = str2[0].length();

            Main temp = new Main();
            temp.tag = str2[0].toLowerCase();
            if (str2.length != 1) {
                temp.id = str2[1];
            }
            temp.rank = (preLength - afterLength) / 2;

            l.add(temp);
        }

        //查询过程
        for (i = 0; i < m; ++i) {
            str2 = in.nextLine().split(" ");
            StringBuffer outCount = new StringBuffer();
```

```
43.             StringBuilder out = new StringBuilder();
44.
45.             int count = 0;      //查询到的个数
46.             if (str2.length == 1) {     //只查询 tag 或者 id
47.                 Main main;
48.                 if (str2[0].charAt(0) == '#') {         //查询 id
49.                     for (int k = 0; k < n; k++) {
50.                         main = l.get(k);
51.                         if (str2[0].equals(main.id)) {
52.                             out.append(k + 1).append(" ");
53.                             count++;
54.                         }
55.                     }
56.                 } else {                                //查询标签
57.                     for (int k = 0; k < n; k++) {
58.                         main = l.get(k);
59.                         if (main.tag.equalsIgnoreCase(str2[0])) {
60.                             out.append(k + 1).append(" ");
61.                             count++;
62.                         }
63.                     }
64.                 }
65.             } else {            //分级查询
66.                 Set<Integer> buff = new HashSet<>();        //临时
67.                 Set<Integer> res = new HashSet<>();         //结果
68.                 for (int k = 0; k < n; k++) {
69.                     if (l.get(k).tag.equals(str2[0].toLowerCase()) || str2[0].equals(
    l.get(k).id)) {
70.                         res.add(k);                     //查询第一个元素
71.                     }
72.                 }
73.                 int start = 1;      //str2[]的索引，开始查询下标为 1 的元素
74.                 while (start < str2.length) {
75.                     buff.clear();       //清空集合
76.                     buff.addAll(res);   //拷贝上次得到的结果
77.                     res.clear();        //清空上次结果
78.                     Object[] buf = buff.toArray(); //转数组
79.                     for (Object o : buf) {  //依次取出匹配到的下一级的下标
80.                         int location = (int) o;
81.                         Main main;
82.                         //找下一级的元素下标，放入 res 集合中
83.                         for (int p = location + 1; p < n && l.get(p).rank > l.get(loc
    ation).rank; p++) {
84.                             main = l.get(p);
```

```
                            if (main.tag.equals(str2[start].toLowerCase()) || str2[st
    art].equals(main.id)) {
                                res.add(p);
                            }
                        }
                    }
                    start++;    //迭代查找下一级
                }
                count = res.size();     //结果集合元素的个数就是结果的个数
                Object[] out2 = res.toArray();
                for (int o = 0; o < count; o++) {
                    out.append((int) out2[o] + 1).append(" "); //构建找到的下标字符串
                }
            }
            outCount.append(count).append(" ").append(out); //构建结果字符串
            String newStr = new String(outCount);
            result.add(newStr.trim());      //去掉首尾空格
        }
        //输出部分
        for (String newResult : result) {
            System.out.println(newResult);
        }
    }
}
```

五、实验结果及分析

提交清单

提交编号	用户名	姓名	试题名称	提交时间	代码长度	编程语言	评测结果	得分	时间使用	空间使用
2271352	〈18863798338〉	〈白文强〉	卖菜	12-14 22:00	1.166KB	JAVA	正确	100	140ms	23.79MB

提交清单

提交编号	用户名	姓名	试题名称	提交时间	代码长度	编程语言	评测结果	得分	时间使用	空间使用
2224945	〈18863798338〉	〈白文强〉	买菜	12-07 21:58	1.603KB	JAVA	正确	100	171ms	35.71MB

提交清单

提交编号	用户名	姓名	试题名称	提交时间	代码长度	编程语言	评测结果	得分	时间使用	空间使用
2275619	〈18863798338〉	〈白文强〉	元素选择器	12-24 19:33	4.491KB	JAVA	错误	80	140ms	24.06MB

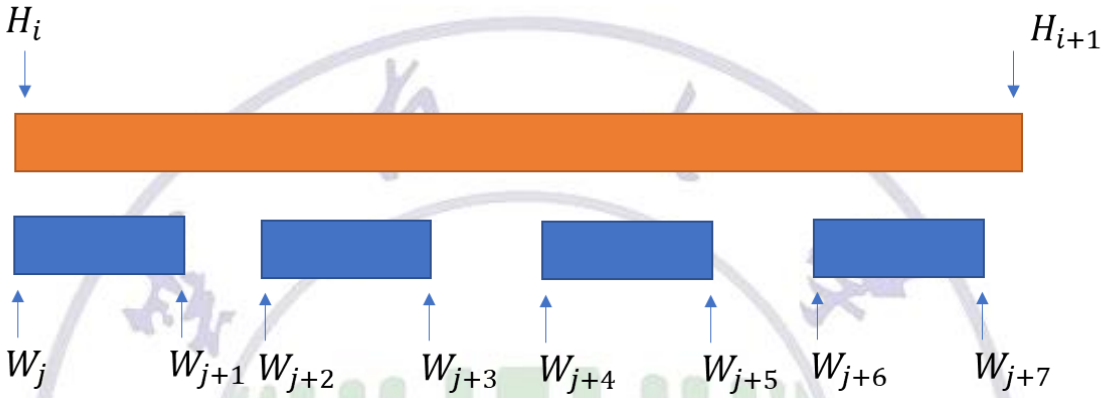
六、实验总结及体会

卖菜：

第一次，在一个数组中进行操作，第二天的价格把第一天的价格覆盖，计算后面的商户的第二天的价格时会出现错误。解决方法即采用两个数组进行操作，将计算出的结果放到新数组中。

买菜：

此题一开始采用单循环，即一段一段比较，但是会有极端情况的出现：



因此，最终采用双循环的做法，对内层循环做了优化，另外此题还可以采用双指针法完成，不需要每次都从W_0开始。

元素选择器：

实现单级查询时，比较简单，当只写完单级查询时，提交得到 50 分，但是实现多级查询时遇到较大困难，一开始的思路出现较大问题，找到第一个符合条件的元素就会返回，忽略了可能存在多个符合条件的情况。后面改变思路，将每次的结果都存下来，然后查询，但是会出现重复的情况，为了避免出现重复，采用了 Set 集合，集合的元素不可重复。由此解决了重复的问题。

通过此次实验，对 CSP 的基本情况有了更深的理解，对题型和要用到的一些知识有了初步掌握。

七、教师评语