Lab6

TA:

- 14302010005@fudan.edu.cn 姜卓立
- 14302010040@fudan.edu.cn 武多才
- 14302010042@fudan.edu.cn 何培剑

1 摘要

1. 学习并熟练掌握一维数组的使用
2. 学习 并熟练掌握多维数组的使用
3. 综合使用条件语句，循环语句，数组等知识画出迷宫界面，并控制人物行走

2 一维数组

Java 中， **数组**是多个相同**类型**元素的集合， 例如多个 **int** 的集合， 多个 **String** 的集合， 多个 **char** 的集合等。下面以代码形式介绍数一维组的**创建**，**访问**和**修改**。

```java
        /*
            Initialize Array.
            Let's create an array that store the position of the player.
         */
        // method 1
        int[] posOfPlayer1 = new int[]{0, 0};
        // method 2, a simple way of method 1
        int[] posOfPlayer2 = {0, 0};
        // method 3
        int[] posOfplayer3 = new int[2]; // declare
        // initialize (we usually use iteration to do this work)
        posOfplayer3[0] = 0;
        posOfplayer3[1] = 0;

        /*
            Ways to modify the values in an array: use index
            Caution: in java, the index should be from 0 to (array length - 1)
         */
        posOfPlayer1[0]++;
        posOfPlayer1[1] = 5;
        // posOfPlayer1[2] = 5 // out of bound error

        /*
            Ways to get the values in an array: use index too.
            Caution: in java, the index should from 0 to (array length - 1)
         */
        int posX = posOfPlayer1[0];
        int posY = posOfPlayer1[1];
         // Result: The player is at position (1, 5).
        System.out.printf("The player is at position (%d, %d).", posX, posY);
         // get the length of an array
         int len = posOfPlayer1.length // len = 2
```

注意：数组的长度是固定的，所以数组不支持**插入**和**删除**的操作。java 中有其它高级的集合类型来完成这些功能。

3 多维数组

以二维数组为例，从一维数组扩展到二维数组就像从一维坐标系扩展到二维坐标系一摸一样。一维数组的索引需要一个值指定位置，二维数组的索引就需要两个值指定位置。Java 中的实现一个二维数组就是一个数组的数组。下面以代码形式介绍数二维组的**创建**，**访问**和**修改**。

```
        /*
            Some preparation work.
         */
        char E = '\u00A0'; // empty
        char W = '▓'; // wall
        char H = '웃'; // hero, player

        /*
            Initialize array.
            Let's create an 2-d array to store the map of our maze game.
         */
        // similar to 1-d array
        // method 1
        char[][] level0 = new char[][]{
                {W, W, W, W},
                {W, E, E, W},
                {W, E, E, W},
                {W, W, W, W}
        };
        // method 2
        char[][] level1 = {
                {W, W, W, W, W},
                {W, E, E, W, W},
                {W, W, E, W, W},
                {W, E, E, E, W},
                {W, W, W, W, W}
        };
        // method 3
        char[][] level2 = new char[10][10];
        for (int i = 0; i < level2.length; i++) {
            for (int j = 0; j < level2[i].length; j++) {
                if (i == 0 || j == 0 || i == 9 || j == 9) // border
                    level2[i][j] = W;
                else
                    level2[i][j] = E;
            }
        }

         /*
            Ways to modify the values in an array: use index
            Caution: in java, the index should from 0 to (array length - 1)
         */
        level2[4][4] = W;
        level2[4][5] = W;
        // Caution: 2-d array is a 1-d array of a 1-d array
        level2[5] = new char[]{W, E, W, W, W, W, W, W, W, W};
        // level2[5] = E; // type error: level2[5] is a 1-d array, but E is a single value
        // level2[4][10] = W; // out of bound error

        /*
            Ways to get the values in an array: use index too.
            Caution: in java, the index should from 0 to (array length - 1)
         */
```

```
        char cell45 = level2[4][5];
        char[] row4 = level2[4];
        // char[] col5 = level2[][5]; Sadly, no such operation


        /*
            3-d array or more.
         */
        // 3-d array is just a 1-d array of a 2-d array.
        char[][][] allLevels = {level0, level1, level2};
        // So, 4-d, 5-d or even more dimensions is just .....


        // Demo application: to show your map.
        // 1. copy the map
        char[][] mapToShow = new char[level2.length][level2[0].length];
        for (int i = 0; i < mapToShow.length; i++) {
            for (int j = 0; j < mapToShow[i].length; j++) {
                mapToShow[i][j] = level2[i][j];
            } // equal to: System.arraycopy(level2[i], 0, mapToShow[i], 0, mapToShow[i].length);
        }
        // 2. set the hero to the mapToShow
        mapToShow[posOfplayer3[0]][posOfplayer3[1]] = H;
        // 3. print mapToShow
        // ....
```

4 作业：迷宫游戏

1. 设计一层合理的二维的迷宫，大小至少为15*20.
2. 选用字符表示墙， 空地，玩家，能够在控制台正常输出迷宫和玩家。
3. 能够输入w, a, s, d控制玩家合理行走。玩家不能穿墙，不能跨越迷宫边界。

5 提交

1. 提交地址：**ftp://10.132.141.33/classes/17/171 程序设计A(戴开宇)/WORK_UPLOAD/lab6/**
2. 提交物：写一个readme文档说明哪个文件是程序入口，然后将其和java代码一起打包， 命名格式为**lab6_[学号]**， 例如：**lab6_17302010001.zip**
3. Deadline: 2017年11月05日23:59:59

6 声明

任何形式的作业都欢迎同学们相互讨论，但抄袭是严格禁止的。一旦发现抄袭行为，抄袭者和被抄袭者都以0分处理