Lab15

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java I/O
2. 使用 I/O 知识实现存档和读档功能
3. 使用 I/O 知识读取外部地图

2 Java I/O

Java I/O 的内容非常丰富，鉴于 PJ 的需要，本次lab介绍一下 Java Binary I/O 的基本用法。

Java Binary I/O 中有两个重要的类 **ObjectOutputStream** 和 **ObjectInputStream**

下面是一些基本用法

```
// write to file
FileOutputStream fos = new FileOutputStream("t.tmp");
ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeInt(12345);
oos.writeObject("Today");
oos.writeObject(new Date());

oos.close();


// read from file in the same order
FileInputStream fis = new FileInputStream("t.tmp");
ObjectInputStream ois = new ObjectInputStream(fis);

int i = ois.readInt();
String today = (String) ois.readObject();
Date date = (Date) ois.readObject();

ois.close();
```

其中两个重要的方法就是 writeObject 和 saveObject 。这两个方法并不是对所有的对象都能够正常工作。为了正常解析对象类要实现相应的接口。

> Classes control how they are serialized by implementing either the **java.io.Serializable** or **java.io.Externalizable** interfaces.

Java 中内置的类一般都可以直接作为参数，自定义的类一般大家选择实现 **Serializable** 接口。

```
// implements Serializable to make the class Map support serialization
class Map implements Serializable
{
    String name;
    int[][] tiles;
}

// demo usage
FileOutputStream fos = new FileOutputStream("t.tmp");
ObjectOutputStream oos = new ObjectOutputStream(fos);

Map m = new Map();

// set Map properties ...
m.name = "SS";

oos.writeObject(m);
oos.close();
```

从以上例子可以看出我们的 Map 类虽然实现了 **Serializable** 接口，但却不需要做任何额外的工作。这是因为我们类中的属性(String and int[][])都是可序列化的。基本上平时使用也是这样，只要将自定义的类实现**Serializable** 接口就可以了

3 实现存档和读档

我们在是之前Lab上的demo上继续开发。

demo地址：https://gitee.com/fduss/mazegame_fx_demo

首先扩展 Sever 类支持save and load。

```java
package sample.server;

import sample.core.Direction;
import sample.core.Icon;
import sample.core.Map;

import java.io.*;

public class Server {

    private Icon[][] map;
    private int[] pos = {1, 1}; // y,x


    // to support save and load game
    public void saveGame() {
        try {
            checkDirectory("save");
            ObjectOutputStream oos = new ObjectOutputStream(
                                                new FileOutputStream("./save/1.save"));
            oos.writeObject(map);
            oos.writeObject(pos);
            oos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void checkDirectory(String name) {
        File file = new File(name);
        if (!file.exists() || file.isFile())
            while (!file.mkdir()) {
                System.out.println("can not create directory: " + name);
            }
    }

    public void loadGame() {
        try {
            ObjectInputStream ois = new ObjectInputStream(
                                                    new FileInputStream("./save/1.save"));
            try {
                // in the save order as writing
                map = (Icon[][]) ois.readObject();
                pos = (int[]) ois.readObject();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    // other methods
```

```
}
```

然后在GUI里添加这部分功能的交互按钮。这里我们使用 MenuBar

```
package sample.client;

/// some imports

/**
 * @author Duocai Wu
 * @Date 2017/11/13
 * @Time 16:58
 */
class MyMenuBar extends MenuBar {

    MyMenuBar(Server server, Main app) {
        super();
        // add file menu to support save and load game
        Menu menuFile = new Menu("File");
        MenuItem loadItem = new MenuItem("Load");
        loadItem.setOnAction(event -> {
            server.loadGame();
            app.updateGameView();
        });
        MenuItem saveItem = new MenuItem("Save");
        saveItem.setOnAction(event -> {
            server.saveGame();
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setContentText("saved.");
            alert.showAndWait();
        });
        menuFile.getItems().addAll(loadItem, saveItem);

        // other menu items

        // add all menu to menu bar
        this.getMenus().addAll(menuFile);
    }
}
```

之后将我们的MenuBar加入到游戏界面中。修改一下我们的主界面类：Main。

```
package sample.client;

/// some imports

public class Main extends Application {
    private GridPane map;
    private Stage stage;


    private void initGameFrame() {
        VBox root = new VBox();
        // add our menubar
        MenuBar menuBar = new MyMenuBar(server, this);
        // add the origin map
        map = new GridPane();
        root.getChildren().addAll(menuBar, map);
        this.stage.setScene(new Scene(root));
    }


    // other methods
}
```

文档只展现了扩展点，下载demo查看完整代码。

4 读取外部地图

为了能够读取外部地图，首先定义外部地图规则。

如下：

```
// rowNumber colNumber
9 27
// 0: space, 1: wall, 2: treasure, 3: monster, 4: end, 5: hero
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 5 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
1 0 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1
1 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 2 0 1 0 1 0 0 0 1
1 0 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1
1 0 0 0 0 0 1 0 0 0 0 0 1 0 3 0 1 0 0 0 0 0 0 0 1 0 1
1 1 1 0 1 0 1 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1
1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 4 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

接着前面的demo开发，我们在 Server 里增加新的方法

```
public boolean loadExternalMap(File file) {
  try {
    Scanner in = new Scanner(file);
    int height = in.nextInt();
    int width = in.nextInt();
    Icon[][] map = new Icon[height][width];
    int[] pos = new int[]{1, 1};
    for (int i = 0; i < height; i++) {
      for (int j = 0; j < width; j++) {
        int num = in.nextInt();
        switch (num) {
          case 1:
            map[i][j] = WALL;
            break;
          case 0:
            map[i][j] = EMPTY;
            break;
          case 2:
            map[i][j] = TREASURE;
            break;
          case 3:
            map[i][j] = MONSTER;
            break;
          case 4:
            map[i][j] = END;
            break;
          case 5:
            map[i][j] = EMPTY;
            pos = new int[]{i, j};
            break;
        }
      }
    }
    this.map = map;
    this.pos = pos;
    return true;
  } catch (Exception e) {
    return false;
  }
}
```

至此，我们的Server已经非常臃肿了。按照前面的lab的设计，Server只应当起到在游戏逻辑和界面UI之前传递消息的作用。但是demo是不可能将游戏逻辑也展现出来的，所以在Server里做了简化的虚假的实现。同学们应当在demo里学到一些处理FX UI的方法，然后用到自己的设计中。

之后我们在Menu Bar里加上交互控件。

```
// add file menu
Menu menuFile = new Menu("File");
MenuItem loadItem = new MenuItem("Load");
MenuItem saveItem = new MenuItem("Save");
// load external map
MenuItem loadExternal = new MenuItem("Load External Map");
loadExternal.setOnAction(event -> {
  // use FileChooser to choose map file
  final FileChooser fileChooser = new FileChooser();
  fileChooser.setInitialDirectory(new File("./"));
  File file = fileChooser.showOpenDialog(app.getStage());
  if (file != null && file.isFile()) {
    // load map
    boolean ok = server.loadExternalMap(file);
    if (ok)
      app.createAndShowGameView(); // update ui
    else {
      Alert alert = new Alert(Alert.AlertType.ERROR);
      alert.setContentText("Not Valid Map File");
      alert.showAndWait();
    }
  }
});
menuFile.getItems().addAll(loadItem, saveItem, loadExternal);
```

5 作业要求

1. 能够展现地图UI
2. 玩家能够行走
3. 能够存档和读取存档，具体形式不限
4. 能够读取自定义地图文件，map文件夹下有四个测试样例。

6 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/lab15/**
2. 提交物：可运行制品，使用说明书，项目源码； 命名格式为**lab15_[学号]**， 例如：**lab15_17302010001.zip**
3. Deadline: 2018年1月7日23:59:59

7 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