初步设计说明文档

——2014/5/26

目录

一、文件与文件夹结构定义.....2
二、类详细说明.....2
I. Main.java.....2
II. Word.java.....4
III. Controller.java.....5
IV. Lexicon.java.....8
V. WordHandler.java.....8
VI. ConfGenerator.java.....9
VII. LexiconHandler.java.....10
三、配置文件格式说明.....12

一、文件与文件夹结构定义

Package：
- view：
  - Main.java
  - view.css
- controller：
  - Word.java
  - Controller.java
  - Lexicon.java
- model：
  - WordHandler.java
  - ConfGenerator.java
  - LexiconHandler.java
- test：
  - 测试用例

二、类详细说明

I. Main.java

前台界面类，采用 javafx。具体界面的跳转可见需求文档的活动图。

1. 在 main 函数中要有：
   //初始化生成 user.conf 文件
   调用 controller.java 中 makeConf(String user)，因为本次只有一个用户，默认直接传入“user”。
   //给 controller.java 里的 word 添加 2 个观察者，一个是 WordHandler，一个是本文件中处理背单词界面的类。

2. 选择词库。选择后调用 controller.java 的 handleLexiconChosen(String name)，传入所选的词库名，如 A 词库，传的是“a”。

3. 在各个页面合理的位置要有“返回”按钮，点击可返回起始选择词库的页面。比如，在查看统计信息后，和背单词结束后 etc。

4. 选择查看统计信息
   之后有 2 种选择：

a. 查看全部词库：调用 controller.java 的 handleTotalInfo()；
b. 查看当前词库：调用 controller.java 的 handleCurrentInfo()。

这两个函数均会返回大小为 3 的 Arraylist<Integer>，第一个为单词总数，第二个为已背词数，第三个为正确词数。

根据这些计算所需统计信息并显示。

显示信息：

a. 查看全部词库：单词总数，已背总数，正确数，错误数，正确率；
b. 查看当前词库：词库名，词库单词总数，已背词数，正确数，错误数，正确率。

表现形式：表格、饼图、柱状图（具体说明见需求文档）。

5. 选择起始单词

有 3 种选择：

a. （默认）词库的第一个单词
b. 上次背到的单词
c. 用户输入

a 和 b 通过调用 controller.java 的 getBeginWord()，该函数返回 String[2]，第一个为词库第一个单词，第二个为上次背到的单词。界面上显示相应单词，点击选择 a 或 b 后会把对应单词填到输入框里。

输入框中起始默认填入词库第一个单词。

c 为用户输入：

用户每输入一个字母，就触发查找匹配单词事件，调用 controller.java 的 handleMatchWord(String str)，str 为当前输入框中的字符串，该函数返回一个 ArrayList<String>列表（单词英文），长度为 0-10。返回的这些匹配单词显示最多 10 个，当点击某个匹配的单词时，将该单词英文填入输入框。

点击“确认”后提交框内单词，调用 controller.java 的 submitBeginWord(String word)，word 为最后框内单词。函数返回的 int 值若为 0，则表示输入的单词不合法，界面提示错误信息（此时后台函数会把起始单词设为默认第一个），然后进入设定单词数量页面。

6. 设定单词数量。在界面输入数字（必须为正数>0，即至少为 1），点击提交，调用 controller.java 的 handleSetCount(int countSet)，countSet 为输入的单词数量。函数返回的 int 值若为 0，则表示输入数量不合法，界面提示错误信息（此时后台函数会把单词数量设为词库中剩余单词数量），然后进入背单词界面。

7. 背单词。当前的单词信息，通过 update 监听 controller.java 中的 word 变化，根据变化进行显示。处理当前界面的这个类要 implements Observer，并实现函数：

```
update(Observable word, Object args1){
    /*  其中 args1 应为新单词中文，*
    *  所以判断第一个是 Word 的实例，*
    *  且第二个参数是 String 之后，才进行处理  */
    If(word instanceof Word && args1 instanceof String){
        显示 args1 即新单词中文在界面上
    }
```

}

8. 输入单词英文，点击“下一个”。调用 controller.java 的 handleNextOne(String wordtxt)，返回一个列表 ArrayList<Integer>，列表第一个元素表示状态：
   0 为输入错误，此时要提醒错误信息，然后继续下个单词；
   1 表示输入正确，直接继续下个单词；
   2 表示已经背完设定数量，本次背单词结束，这种情况下，ArrayList 共有 3 个元素，第二个为本次背诵设定的单词数量，第三个为本次背诵的正确词数。根据这些计算本次背诵统计信息，结束背诵并显示统计信息，形式为表格，具体为：所选库名（我觉得可能可以在这个类里设个库名的变量，最开始选择库名的时候把这个变量也初始化一下，从 controller 那边传过来有点烦捏⊙__⊙），所选单词数，正确词数，错误词数，正确率。

II. Word.java

单词类，被观察者，要 extends Observable。

变量：

String user：当前用户
Integer wordEntry：当前处理单词的入口（byte 数目）
String english：单词英文
String chinese：单词中文
Integer state：单词的状态，0 为未背过，1 为正确，2 为错误

函数：
实现 constructor 和各属性的 getter 和 setter，其中 setter 里应调用 setChanged()和 notifyObservers()来通知观察者 Word 的改变。
其中主要用到的函数有两个：

1. 设置单词状态的 setter
   用于 controller 处理用户点击下一个事件处理中的调用。将单词背的对错的状态作为参数传入 notifyObservers(state)，观察者 WordHandler 监听到变化进行处理。
   ```
   public void setState(Integer state){
       this.state = state;
       this.setChanged();
       this. notifyObservers(state);
   }
   ```
2. 设置新单词的函数
   用于 controller 中显示下一个单词 showNextWord(String user, Integer wordEntry, String english, String chinese, Integer state)调用，setWord 之后，Main.java 前台的观察者，监听到变化会更改单词信息，显示新单词。
   ```
   public void setWord(String user, Integer wordEntry, String English, String Chinese, Integer state){
   ```

```
        this. user = user;
        this. wordEntry = wordEntry;
        ……
        this.state = state;
        this.setChanged();
        //传入新单词中文，方便传到 UI 显示
        this. notifyObservers(chinese);
    }
```

III. Controller.java

控制器，负责从前台 view 与后台 model 之间的交互。

变量：

```
Word word = new Word();       //当前的单词
LexiconHandler lh = new LexiconHandler();    //负责处理词库相关的操作
int count = 0;    //计数器，用于判断何时背单词结束
int countTotal = 0;    //本次背单词设置的单词总数
int countCorrect = 0;       //本次背单词正确的词数
```

函数：

1. 处理“下一个”点击事件，在 Main.java 中调用，传输用户输入的单词，返回 ArrayList<Integer>。ArrayList<Integer>的第 1 个元素为单词背诵状态：0 为背错；1 为背对；2 为背诵数目已到，结束，这种情况下 ArrayList<Integer>还有 2 个元素，第 2 个为 countTotal，第 3 个为 countCorrect。

```
public ArrayList<Integer> handleNextOne(String wordtxt){
        ArrayList<Integer> list = new ArrayList<Integer>();
        Integer state;
        比较 word.getEnglish()和 wordtxt，判断得出对错状态，赋给 state；
        If(state == 1){
            countCorrect ++;
        }
        //改变当前单词的状态，WordHandler 观察到变化会修改 conf 文件，
        //然后读取下一个单词信息
        word.setState(state);
        count --;
        if(count = = 0){
            state = 2;
        }
        /*  处理 list 存的东西，*
        *  当 state 为 0 或 1 时，只需要存 state 一个元素；*
        *  当 state 为 2 时，需要存 state，countTotal 和 countCorrect。*/
```

```
        ……
        return list;
    }
```

2. 用于 WordHandler 读完新单词信息后调用，更新当前 word 为新单词，调用 setWord(String user, Integer wordEntry, String English, String Chinese, Integer state)，Main.java 中的观察者观察到单词变化，会显示新单词。

```
public void showNextWord(String user, Integer wordEntry, String english, String chinese, Integer state){
        //判断是否还要继续背单词
        if(count != 0){
            word.setWord(user, wordEntry, English, Chinese, state);
        }
}
```

3. Main.java 程序初始化的时候生成 user.conf 文件。

```
public void makeConf(String user){
        ConfGenerator.generateConf(user);
}
```

4. 处理选择词库事件，传入词库名，如 A 词库传入“a”。

```
public void handleLexiconChosen(String name){
        lh.initLexicon(name);
        //此时 lh 中的 lexicon 和 wordlist 都完成初始化
}
```

5. 处理查看全部词库统计信息，返回统计信息。

```
public ArrayList<Integer> handleTotalInfo(){
        ArrayList<Integer> info;
        info = lh.getTotalInfo();
        return info;
}
```

6. 处理查看当前词库统计信息，返回统计信息。

```
public ArrayList<Integer> handleCurrentInfo(){
    ArrayList<Integer> info;
    info = lh.getCurrentInfo();
    return info;
}
```

7. 获得起始单词，返回长度为 2 的 String[] 数组，第 1 个为当前词库第一个单词，第 2 个为上次背到的单词。

```
public String[] getBeginWord(){
    String first = lh.getFirstWord();
```

```
        String lastTime = lh.getLastTimeWord();
        //返回这 2 个字符串组成的数组
        ……
    }
```

8. 处理用户输入自动匹配单词，返回匹配的单词列表，最多 10 个。

```
    public ArrayList<String> handleMatchWord(String str){
        ArrayList<String> match;
        match = lh.getMatchWords(str);
        return match;
    }
```

9. 处理提交起始单词事件，返回 int 值，表示输入单词是否存在词库里，单词合法为 1，不合法为 0。不合法时，将起始单词设置为词库第一个单词。

```
    public int submitBeginWord(String wordstr){
        //先判断单词是否合法
        int flag = lh.checkValidWord(wordstr);
        if(flag == 0){
          wordstr = lh.getFirstWord();  //设置为第一个单词
        }
        //找到起始单词，并将新单词信息赋给 word
        WordHandler.findBeginWord(wordstr, lh.getLaxiconName);
        return flag;
    }
```

10. 处理设定单词数量事件，返回 int 值，表示数量是否合法，合法为 1，不合法为 0。

```
    public int handleSetCount(int countSet){
        //获得当前的单词，即选定的起始单词
        String wordstr = word.getEnglish();
        Int flag;
        int maxCount = lh.getRestCount(wordstr);
        if(countSet > maxCount){
                flag = 0;
                countTotal = maxCount;
        } else{
                flag = 1;
                countTotal = countSet;
        }
        count = countTotal;
        return flag;
    }
```

IV. Lexicon.java

词库类，用于存储当前词库的信息。

变量：

String lexiconName：词库名
int totalCount：词库单词总数
int recitedCount：已背词数
int correctCount：正确词数
int lexiconEntry：词库起始单词入口（即 byte 数）
int lastTime：上次背到的单词入口（即 byte 数）

函数：
实现 constructor 和各属性的 getter 和 setter。

V. WordHandler.java

观察者，观察 Word，处理单词变化等对单词的操作，该类要 implements Observer。

函数：

1. 实现 update 函数，用于观察“下一个”操作引起的单词状态变化，修改 user.conf 中的单词信息，并读取下一个单词信息，调用 controller 的 showNextWord 进行显示。

```
public void update(Observable word, Object arg1){
  //传入的 args1 为单词状态（Integer）时，才执行函数
  if(word instanceof Word && arg1 instanceof Integer){
      ①修改刚刚背过的单词状态
      /*  判断传入的 arg1 值是否为 0 或 1，*
      *  若是，则 arg1 为用户输入单词正确（1）或错误（0）的状态；  *
      *  否则，其他状况不进行处理  */
      int newState;
      //将 arg1 的值赋给 newState
      ……
      /*  由 Integer we = word.getWordEntry()得到单词入口，*
      *  修改 user.conf 中当前词库统计信息中上次背到的单词入口为 we。  *
      *  由 we 读到当前处理单词那行，读取该单词信息，  *
      *  取出 user.conf 文件中原来的单词状态，记为 oldState */

      修改 user.conf：
      If(oldState == 0){    //原来没背过
          当前处理单词状态修改为 newState；
```

当前词库统计信息里的已背词数+1；
If(newState == 1){ //这次背对了
当前词库统计信息里的正确词数+1；
}
} else if(oldState == 2){ //原来背过但背错了
if(newState == 1){ //在这次背对了
当前处理单词状态修改为 newState；
当前词库统计信息里的正确词数+1；
}
}
//其他情况 oldState == 1，原来就背对了，
//不管这次如何对 user.conf 都不需要修改。

②读取下一个单词信息
由刚刚背过的单词入口 we，readline 得到下一行，即下一个单词信息，解析文本整理成一个新的 Word 信息，包括 user、wordEntry 等。
调用 controller.java 中的 showNextWord(String user, Integer wordEntry, String english, String chinese, Integer state)显示下一个单词信息。
}
}

2. 设定起始单词时使用。根据传入的单词英文，找出 user.conf 文件中相应单词信息，并调用 controller 的 showNextWord 显示第一个单词信息。
public void findBeginWord(String wordstr, String lexiconname){
由词库名 lexiconname，在 user.conf 查找词库信息，获得词库入口。
从词库入口开始循环比较，找出英文为 wordstr 的单词信息，解析文本整理成一个新的 Word 信息，包括 user、wordEntry 等。
调用 controller.java 中的 showNextWord(String user, Integer wordEntry, String english, String chinese, Integer state)显示这个起始单词信息。
}

VI. ConfGenerator.java

主要用于初始化生成 user.conf 文件。

函数：
public void generateConf(String user){
新建一个 user.conf（以传入的 user 命名）；
读取 dictionary.txt 文件，根据配置文件格式填写 user.conf：
前 26 行为各词库统计信息，
后面为各单词信息。
}

说明：
生成 conf 文件所需要调用的其他函数，如统计 dictionary 里某词库单词总数、计算某词库起始单词入口、计算每个单词的单词入口等等，在实现 generateConf()过程中自行定义。

注意：
初始生成 user.conf 文件时，
词库统计信息中：
①上次背到的单词入口：设置为各词库起始单次入口；
②已背词数：设置为 0；
③正确词数：设置为 0；
单词信息中：
单词状态均设置为 0。

VII. LexiconHandler.java

处理与词库相关操作的类。

变量：
Lexicon lexicon = new Lexicon();　　//所操作的当前词库
ArrayList<String> wordlist = new ArrayList<String> ();　//存放词库中所有单词的英文，
　　//按照词库中的顺序排列，在自动匹配 etc 中使用

函数：
1. 初始化词库信息，传入词库名（A 词库传入“a”）。
   public void initLexicon(String name){
   　　读取 user.conf 文件，A 到 Z 词库信息在前面 26 行，
   　　可以通过 ASCII 码什么的计算该词库信息在第几行，
   　　然后读出那行的信息。
   　　①解析得到 lexicon 中各变量值，调用 Lexicon 的 setter 进行设置；
   　　②从中得到起始单词入口和单词总数，循环读出当前词库中的所有单词英文，存在 wordlist 里。
   }

2. 查看全部词库信息，返回的列表长度为 3，第一个为单词总数，第二个为已背词数，第三个为正确词数。
   public ArrayList<Integer> getTotalInfo(){
   　　读取 user.conf 文件中前 26 行各词库信息，相加得到全部词库信息，并返回。
   }

3. 查看当前词库信息，返回的列表长度为 3，第一个为当前词库单词总数，第二个

为已背词数，第三个为正确词数。

public ArrayList<Integer> getCurrentInfo(){

返回由 lexicon 中 totalCount，recitedCount 和 correctCount 组成的 ArrayList<Integer>列表。

}

4. 获得词库的第一个单词

public String getFirstWord(){

//由 lexicon 的 lexiconEntry 去读取词库起始单词信息，返回该单词英文。

}

5. 获得当前词库上次背到的单词

public String getLastTimeWord(){

//由 lexicon 的 lastTime 去读取词库上次背到的单词信息，返回该单词英文。

}

6. 根据字符串匹配单词，返回匹配的单词英文列表，最多为 10 个。

public ArrayList<String> getMatchWords(String str){

//用循环，将 str 与 wordlist 中的单词英文的前缀进行匹配，找出匹配的单词英文列表。当找够 10 个单词时，即可退出循环。返回找到的匹配单词列表。

}

7. 检查输入单词是否在当前词库中，返回 int 值，若为 0 表示不在词库中；若为 1 表示在词库中。

public int checkValidWord(String word){

int flag = 0;

//用循环比较 word 与 wordlist 中的单词英文，若找到相同单词，则将 flag 设为 1，退出循环。

return flag;

}

8. 获得从传入的单词开始，当前词库剩余的单词数量。

public int getRestCount(String wordstr){

int cnt;

①用循环比较 wordstr 与 wordlist（按词库顺序排列）中的单词，找到输入单词是该词库的第几个，记下标为 index，退出循环。

②由 index 和 lexicon.getTotalCount()（当前词库单词总数），计算剩余单词数量 cnt。

return cnt;

}

三、配置文件格式说明

用户配置文件 user.conf，其中 user 为当前用户名。
文件前 26 行存储词库信息，从第 27 行开始存储单词信息。
文件格式如下（行数这列不存储）：

行数	存储字段					
	词库名	词库入口	上次背到的单词入口	单词总数	已背词数	正确词数
1	a	abandon 的单词入口				
2	b					
…						
26	z					

	单词英文	单词中文	单词状态	本单词入口
说明			0：未背；1：正确；2：错误	文件起始处到本单词这行的 byte 数
27	abandon	v. 抛弃，放弃	0	
28	abandonment	…		
…	…			