QuizSystem要求与设计思路

`resources`

- 用户信息 `resources/users.csv`
  - 格式 `id, name, password`
- 测验的问题 `resources/questionBank`
  - 格式 `.xml`
  - 已提供 `ReadQuestions.java` 展示如何读取问题

已有库中的功能，来自javadoc

- `package xjtlu.cpt111.assignment.quiz.lang`
  - `AppConstants`
  - `AppConstants.Model`
  - `XmlEntity`
  - `XmlInputStream`
- `package xjtlu.cpt111.assignment.quiz.model` 提供DOM类
  - `Difficulty` 包含4个问题难度的枚举类
  - `Option` 问题的选项
    - 使用字符串构造， `isCorrectAnswer` 可选，指示选项是否正确
    - 有 `isCorrectAnswer()` 方法用于判定字符串代表的选项是否正确
    - 重写了 `toString()` ，说明可能有打印选项的需求
    - `getAnswer()` answer是什么?
  - `Question`
    - 构造：所属的topic，难度(optional，来自枚举类)，问题陈述，选项
    - 全是get和set方法
    - 有 `toString()`
- `package xjtlu.cpt111.assignment.quiz.util` 提供工具
  - `IOUtilities` 提供从XML文件读取问题的实现
    - `readQuestions()` 获得 `Question` 数组
      - 3种参数形式， `String`, `File`, `Path`
      - 示例 `ReadQuestions.java` 中使用字符串参数形式

交互要求概括：

- 菜单

- 用户注册和登入（未登入状态下的唯一选项）
- 答题
  - 开始答题：选择主题和quiz中问题数目
  - 逐个展示乱序的题目（展示的题目需要先通过验证）
  - 答题完毕后展示分数，分数写入用户的 score file，返回到菜单
- dashboard
  - 查看每个主题的 quiz results
  - 查看最近3次的答题情况
- Leaderboard
  - 展示每个主题的用户排行榜，按照分数排序

模块设计，MVC模式:

```
├── Main.java               // 主程序入口
├── controller/             // 控制类，用于控制交互，调用service和view
│   ├── MenuController.java    // 菜单控制类
│   ├── QuizController.java    // 答题控制类
│   ├── DashboardController.java // Dashboard控制类
│   ├── LeaderboardController.java // Leaderboard控制类
├── model/                  //
│   ├── User.java              // 用户实体类
│   ├── Question.java          // 题目实体类(包装类，因为包里已经提供
Question)
│   ├── Score.java             // 分数实体类
│   ├── QuizResult.java        // 答题结果实体类
├── service/
│   ├── UserService.java       // 用户服务类：注册、登录、更改用户名和密码..
│   ├── QuizService.java       // 答题服务类：题目加载，题目乱序，题目有效性
验证，分数计算
│   ├── DashboardService.java  // Dashboard服务类
│   ├── LeaderboardService.java // Leaderboard服务类
├── util/
│   ├── FileUtil.java          // 文件操作工具类：读题(IOUtilities)，读写
用户信息，读写分数信息，读写答题情况
│   ├── ValidationUtil.java    // 验证工具类:验证题目有效性，验证用户是否存在
├── view/
│   ├── MenuView.java          // 菜单视图类
│   ├── QuizView.java          // 答题视图类
```

```
│   ├── DashboardView.java     // Dashboard视图类
│   ├── LeaderboardView.java   // Leaderboard视图类
```

`Main.java`：

- 单一入口点，先初始化整个程序（创建所有model和view）
- 然后使用一个 `setActiveContoller()` 确保只有一个控制器活跃
  `view` 显示命令行界面，根据用户选择调用 `controller`
- `controller` 与 `view` 一一对应
  `controller` 调用 `service`，等待 `service` 结束后决定如何更新 `view`
  `service` 返回一些结果给 `controller`，可能操作 `model`