目录

[1.引言 1](#_Toc414814601)

[1.1编制目的 1](#_Toc414814602)

[1.2词汇表 1](#_Toc414814603)

[1.3参考资料 1](#_Toc414814604)

[2.产品概述 1](#_Toc414814605)

[3.体系结构设计概述 1](#_Toc414814606)

[4.结构视角 2](#_Toc414814607)

[4.1业务逻辑层的分解 2](#_Toc414814608)

[4.1.1 playerbl模块 2](#_Toc414814609)

[4.1.2TeamBL模块 7](#_Toc414814610)

[4.2数据层的分解 10](#_Toc414814611)

[4.2.1 teamData模块 10](#_Toc414814612)

[4.2.2 PlayerData模块 13](#_Toc414814613)

[5.依赖视角 16](#_Toc414814614)

# 1.引言

## 1.1编制目的

本报告详细完成对进销存系统的详细设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| **词汇名称** | **词汇含义** | **备注** |
| businesslogic | 业务逻辑 |  |
|  |  |  |
|  |  |  |

## 1.3参考资料

NBA数据查询系统用例文档

NBA数据查询系统软件需求规格说明文档

NBA数据查询系统体系结构描述文档

# 2.产品概述

参考NBA数据查询系统用例文档和NBA数据查询系统软件需求规格说明文档对产品的概括描述。

# 3.体系结构设计概述

参考NBA数据查询系统体系结构描述文档中对体结构设计的概述。

# 4.结构视角

## 4.1业务逻辑层的分解

业务逻辑层的开发包图参见软件体系结构文档。

### 4.1.1 playerbl模块

（1）模块概述

Playerbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Playerbl模块承担的职责及接口参见软件体系结构描述文档。

（2）整体结构

我们将系统分为展示、业务逻辑、数据3层。我增加层与层之间的灵活性，展示与业务逻辑层间我们增加PlayerBLService接口；数据与业务逻辑层间我们增加PlayerDataService接口。为隔离业务逻辑职责和逻辑控制职责，我们增加PlayerController，这样业务逻辑处理委托给PlayerCalcultor。Sort负责排序，CalMethod封装计算方法，来隔离可能的变更。而诸多PO对象是作为球员数据的持久化对象被添加到设计模型中去。

Palyerbl模块的设计如图1所示。

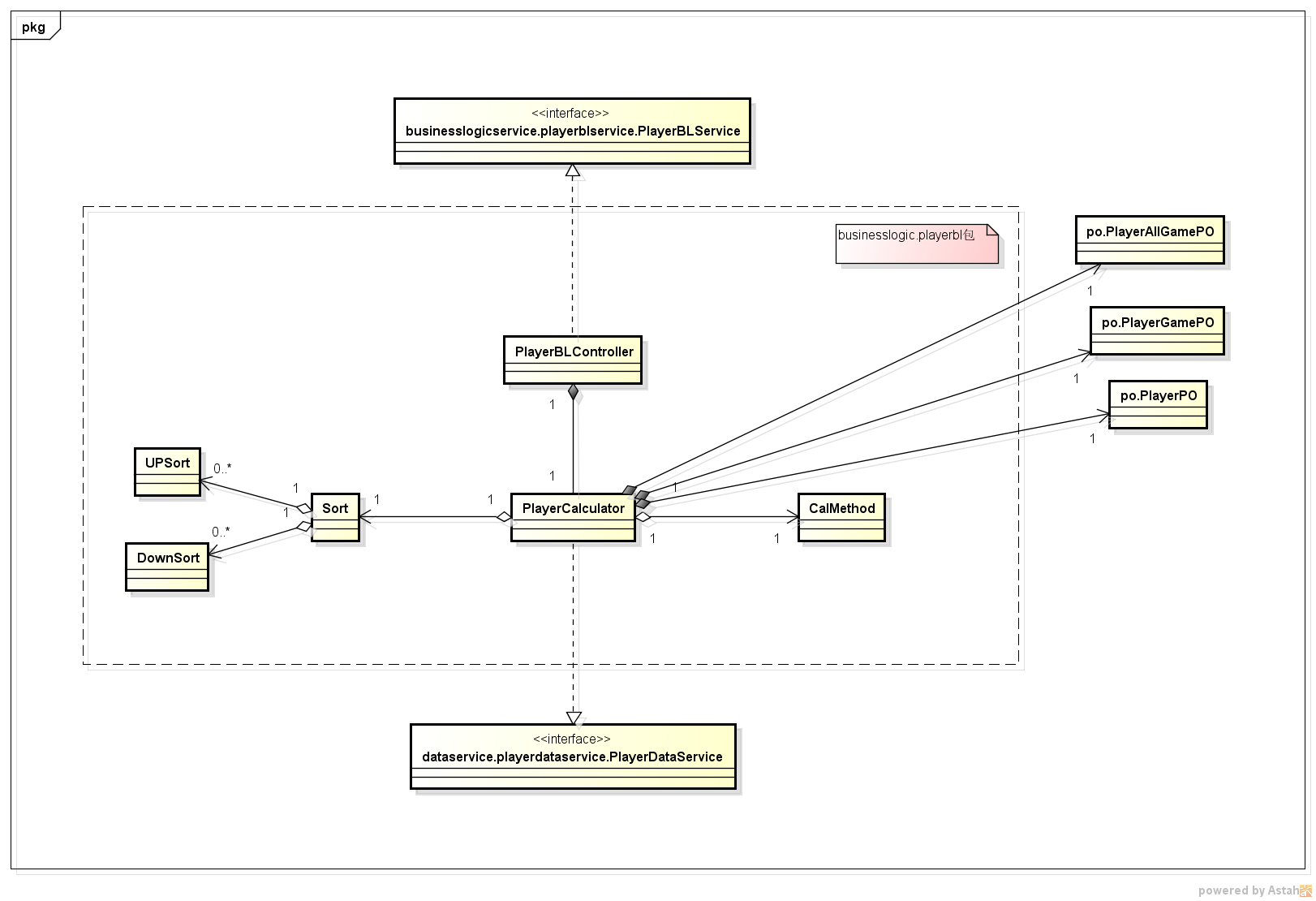


图1 playerbl模块各个类的设计

Palyerbl模块各个类的职责如表1所示。

表1 playerbl模块各个类的职责

|  |  |
| --- | --- |
| **模块** | **职责** |
| PlayerBLController | 负责实现球员界面所要的任务 |
| PlayerCalculator | 负责实现计算逻辑控制 |
| Sort | 负责排序 |
| CalMethod | 负责封装计算方法 |

（3）模块内部类的接口规范

PlayerBLController，PlayerCalculator，Sort，CalMethod的接口规范如表3、表4、表5、表6所示。

表3 PlayerBLController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PlayerBLController.getPlayerTotalInfo | 语法 | public PlayerVO getPlayerTotalInfo (String name) |
| 前置条件 | 已创建PlayerCalculator对象，且输入符合规则 |
| 后置条件 | 调用PlayerCalculator.getPlayerInfo |
| PlayerBLController.getPlayerAvgInOrder | 语法 | Public ArrayList<PlayerInfoVO> getPlayerAvgInOrder(Order order,PlayerStandard stan) |
| 前置条件 | 已创建PlayerCalculator对象，且输入符合规则 |
| 后置条件 | 调用PlayerCalculator.avgSort |
| PlayerBLController.getPlayerTotalInOrder | 语法 | public ArrayList<PlayerInfoVO> getPlayerTotalInOrder(Order order,PlayerStandard stan) |
| 前置条件 | 已创建PlayerCalculator对象，且输入符合规则 |
| 后置条件 | 调用PlayerCalculator.totalSort |
| PlayerBLController.getPlayerTop\_50 | 语法 | public ArrayList<PlayerInfoVO> getPlayerTop\_50(Order order,PlayerStandard stan,String pos,Zone zone) |
| 前置条件 | 已创建PlayerCalculator对象，且输入符合规则 |
| 后置条件 | 调用PlayerCalculator.getTotalTopList或PlayerCalculator.getAvgTopList |
| PlayerBLController.getHotPlayer | 语法 | **Public** ArrayList<PlayerHotInfo> getHotPlayer(Field field,int num); |
| 前置条件 | 已创建PlayerCalculator对象，且输入符合规则 |
| 后置条件 | 调用PlayerCalculator.getTotalTopList或PlayerCalculator.getAvgTopList |
| PlayerBLController.getKingPlayer | 语法 | public ArrayList<PlayerKingInfo> getSeasonKingPlayer(Field field,int num); |
| 前置条件 | 已创建PlayerCalculator对象，且输入符合规则 |
| 后置条件 | 调用PlayerCalculator.getTotalTopList或PlayerCalculator.getAvgTopList |
| |  |  |  | | --- | --- | --- | | PlayerBLController.getDailyKingPlayer | 语法 | public ArrayList<PlayerKingInfo> getDailyKingPlayer(Field field,int num); | | 前置条件 | 已创建PlayerCalculator对象，且输入符合规则 | | 后置条件 | 调用PlayerCalculator.getTotalTopList或PlayerCalculator.getAvgTopList | | | |
| PlayerCalculator.getPlayerInfo | | 获取单个球员信息 |
| PlayerCalculator.totalSort | | 按赛季总数据对球员排序 |
| PlayerCalculator.avgSort | | 按场均数据对球员排序 |
| PlayerCalculator.getTotalTopList | | 取赛季总数据前50球员 |
| PlayerCalculator.getAvgTopList | | 取场均数据前50球员 |

表4 PlayerCalculator的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PlayerCalculator.getPlayerInfo | 语法 | public PlayerVO getPlayerInfo(String name) |
| 前置条件 | 无 |
| 后置条件 | 调用PlayerCalculator.getPlayer |
| PlayerCalculator.totalSort | 语法 | public ArrayList<PlayerInfoVO> totalSort(Order order,PlayerStandard ps) |
| 前置条件 | 已创建Sort对象 |
| 后置条件 | 调用Sort.sort |
| PlayerCalculator.avgSort | 语法 | public ArrayList<PlayerInfoVO> avgSort(Order order,PlayerStandard ps) |
| 前置条件 | 已创建Sort对象 |
| 后置条件 | 调用Sort.sort |
| PlayerCalculator.getTotalTopList | 语法 | public ArrayList<PlayerInfoVO> getTotalTopList(Order order,PlayerStandard ps,String position,Zone zone) |
| 前置条件 | 已创建Sort对象 |
| 后置条件 | 调用Sort.sort |
| PlayerCalculator.getAvgTopList | 语法 | public ArrayList<PlayerInfoVO> getAvgTopList(Order order,PlayerStandard ps,String position,Zone zone) |
| 前置条件 | 已创建Sort对象 |
| 后置条件 | 调用Sort.sort |
| PlayerCalculator.getPlayer | 语法 | public PlayerPO getPlayer(String name) |
| 前置条件 | 已获取数据层接口 |
| 后置条件 | 获取数据层数据 |
| |  |  |  | | --- | --- | --- | | PlayerBLController.getHotPlayer | 语法 | **Public** ArrayList<PlayerHotInfo> getHotPlayer(Field field,int num); | | 前置条件 | 已创建PlayerCalculator对象，且输入符合规则 | | 后置条件 | 调用PlayerCalculator.getTotalTopList或PlayerCalculator.getAvgTopList | | PlayerBLController.getKingPlayer | 语法 | public ArrayList<PlayerKingInfo> getSeasonKingPlayer(Field field,int num); | | 前置条件 | 已创建PlayerCalculator对象，且输入符合规则 | | 后置条件 | 调用PlayerCalculator.getTotalTopList或PlayerCalculator.getAvgTopList | | |  |  |  | | --- | --- | --- | | PlayerBLController.getDailyKingPlayer | 语法 | public ArrayList<PlayerKingInfo> getDailyKingPlayer(Field field,int num); | | 前置条件 | 已创建PlayerCalculator对象，且输入符合规则 | | 后置条件 | 调用PlayerCalculator.getTotalTopList或PlayerCalculator.getAvgTopList | | | | | | |
| Sort.sort | | 进行排序 |

表5 Sort的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Sort.sort | 语法 | public void sort(ArrayList<PlayerInfoVO> list,Order order, PlayerStandard ps) |
| 前置条件 | 创建UPSort或DownSort对象 |
| 后置条件 | 调用UPSort.quickSort或DownSort.quickSort |
| 需要的服务（需接口） | | |
| UPSort.quickSort | | 进行升序排序 |
| DownSort.quickSort | | 进行降序排序 |

表6 CalMethod的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CalMethod.calT | 语法 | public double calT(int time,int allPlayerTime) |
| 前置条件 | 无 |
| 后置条件 | 返回计算中间变量T |
| CalMethod.calRate | 语法 | public double calRate(int sample,int sum) |
| 前置条件 | 无 |
| 后置条件 | 返回命中率 |
| CalMethod.calEfficiency | 语法 | public double calEfficiency(int rebTotalNum, int assistNum, int stealNum,int blockNum, int errorNum, int score, int hitShootNum, int shootNum, int freeHitNum,int freeNum) |
| 前置条件 | 无 |
| 后置条件 | 返回效率 |
| CalMethod.calGmScEfficiency | 语法 | public double calGmScEfficiency(int rebAttNum, int rebDefNum, int assistNum, int stealNum,int blockNum, int errorNum, int foulNum, int score, int hitShootNum, int shootNum, int freeHitNum,int freeNum) |
| 前置条件 | 无 |
| 后置条件 | 返回GmSc效率 |
| CalMethod.calRealHitRate | 语法 | public double calRealHitRate(int score, int shootNum, int freeNum) |
| 前置条件 | 无 |
| 后置条件 | 返回真实命中率 |
| CalMethod.calThrowRate | 语法 | public double calThrowRate(int hitShootNum, int shootNum, int threePointNum) |
| 前置条件 | 无 |
| 后置条件 | 返回投篮效率 |
| CalMethod.calRebRate | 语法 | public double calRebRate(int rebTotalNum, double T, int teamRebNum, int oppTeamRebNum) |
| 前置条件 | 无 |
| 后置条件 | 返回篮板率 |
| CalMethod.calAssistRate | 语法 | public double calAssistRate(int assistNum, int teamHitNum, int hitShootNum, double T) |
| 前置条件 | 无 |
| 后置条件 | 返回助攻率 |
| CalMethod.calStealRate | 语法 | public double calStealRate(int stealNum, int oppAttNum, double T) |
| 前置条件 | 无 |
| 后置条件 | 返回抢断率 |
| CalMethod.calBlockRate | 语法 | public double calBlockRate(int blockNum, int oppTwoNum, double T) |
| 前置条件 | 无 |
| 后置条件 | 返回盖帽率 |
| CalMethod.calErrorRate | 语法 | public double calErrorRate(int errorNum, int twoNum, int freeNum) |
| 前置条件 | 无 |
| 后置条件 | 返回失误率 |
| CalMethod.calUseRate | 语法 | public double calUseRate(int shootNum, int freeNum, int errorNum, double T, int teamThrowNum, int teamFreeNum, int teamErrorNum) |
| 前置条件 | 无 |
| 后置条件 | 返回使用率 |

（4）业务逻辑层的动态模型

图2表明了对场均数据排序的顺序图。

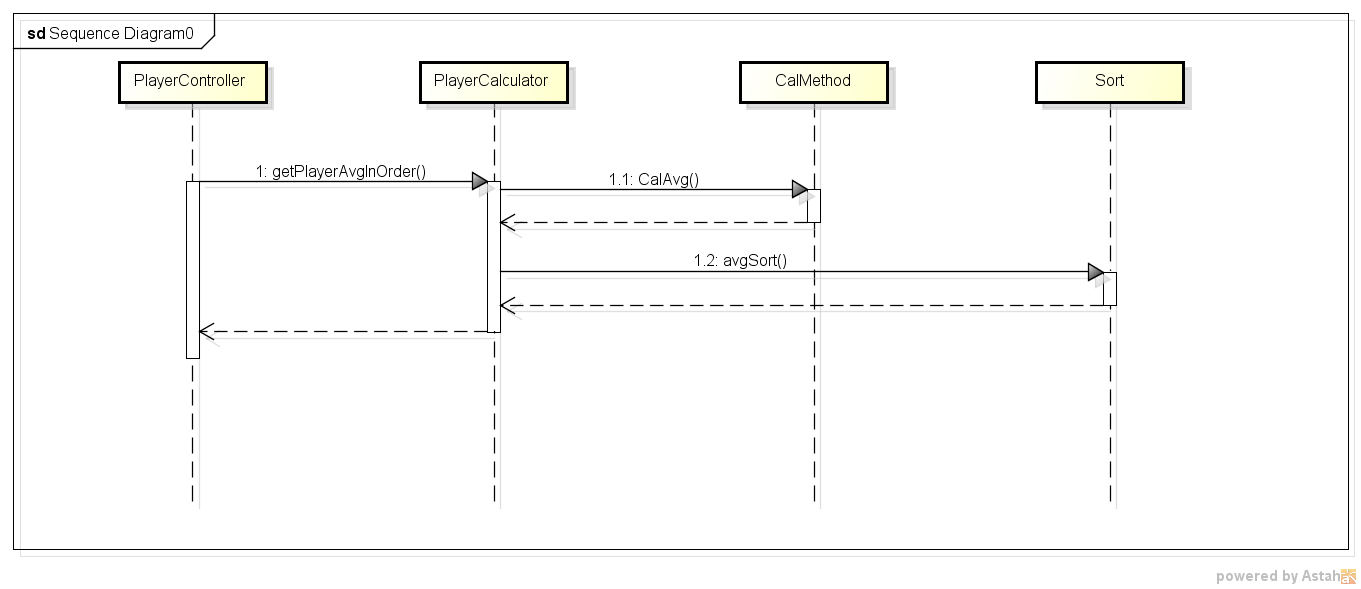


图2 对场均数据排序的顺序图

图3表明了Playercalculator的状态序列。

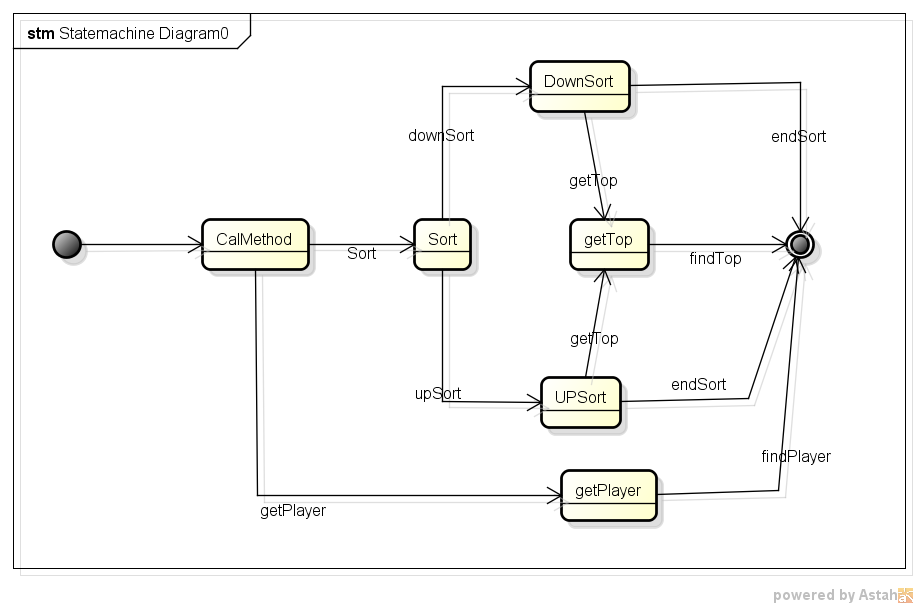


图3 PlayerCalculator对象状态图

（5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

### 4.1.2TeamBL模块

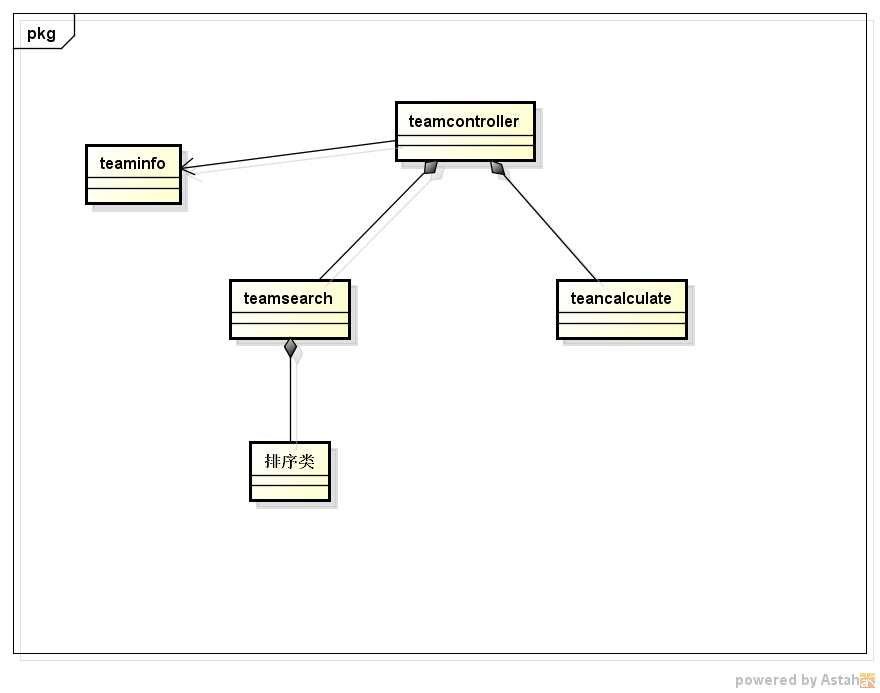
**（**1）模块概述

Checkbl模块承担的需求参见需求规格说明文档的功能需求及相应非功能需求

Checkbl模块的职责及接口参见软件体系结构文档

（2）整体结构

根据体系结构的设计，我们将系统分为展示层，逻辑层与数据层，每一层之间添加接口，在展示层与逻辑层间我们添加bussinesslogic.teamblservice.teamblservice接口，为了隔离业务逻辑职责与逻辑控制职责，我们增加了teamcontroller，这样checkcontroller会将球队数据处理的业务逻辑委托给teamcalculate与teamsearch对象处理。Teambl模块的设计如图所示



Teambl模块各个类职责如下

|  |  |
| --- | --- |
| 模块 | 职责 |
| Teamcontroller | 负责实现teamservice的接口方法，为展示层与playerbl提供服务 |
| Teamcalculate | 负责球队各项数据的计算，计算方法提供者 |
| Teamsearch | 负责按顺序将球队排序，要调用许多排序类的方法 |
| 排序类 | 为不同排序方法而写的不同排序类，为了减少if-else的循环数而写了许多类 |
| Teaminfo | 给playerbl的封装类 |

（3）模块内接口规范

Teamcontroller接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Teamcontroller.getteaminfo | 语法 | public TeamInfo getTeamInfo(String team, String date) |
| 前置条件 | 已创建teamcontroller对象 |
| 后置条件 | 返回对应球队相应日期比赛信息 |
| Teamcontroller.getarea | 语法 | public String getArea(String teamName); |
| 前置条件 | 已创建teamcontroller对象 |
| 后置条件 | 返回球队分区与赛区 |
| Teamcontroller.getteaminfo | 语法 | public TeamVO getTeamInfo(String name); |
| 前置条件 | 已创建teamcontroller对象 |
| 后置条件 | 返回对应球队全部信息 |
| Teamcontroller.getteaminorder | 语法 | public ArrayList<String> getTeamInOrder(Order order,TeamStandard stan); |
| 前置条件 | 已创建teamcontroller对象 |
| 后置条件 | 调用teamsearch的对应方法 |
| Teamcontroller.getallteam | 语法 | public HashMap<String,Image> getAllTeam(); |
| 前置条件 | 已创建teamcontroller对象 |
| 后置条件 | 调用teamdataservice的对应方法 |
| 需要的服务（需接口） | | |
| Teamdataservice.getteamgamedata | | 得到球队比赛数据 |
| Teamdataservice.getteamdata | | 得到球队基本数据 |
| Teamdataservice.getteamimage | | 得到球队图片 |
| Teamcalculater | | 提供计算方法 |
| Teamsearch.sort | | 排序算法 |

Teamcalculate接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Teamcalculate.calallpoint | 语法 | Public int getallpoint（teamAllGamePO） |
| 前置条件 | 无 |
| 后置条件 | 返回总得分 |

其余接口类似

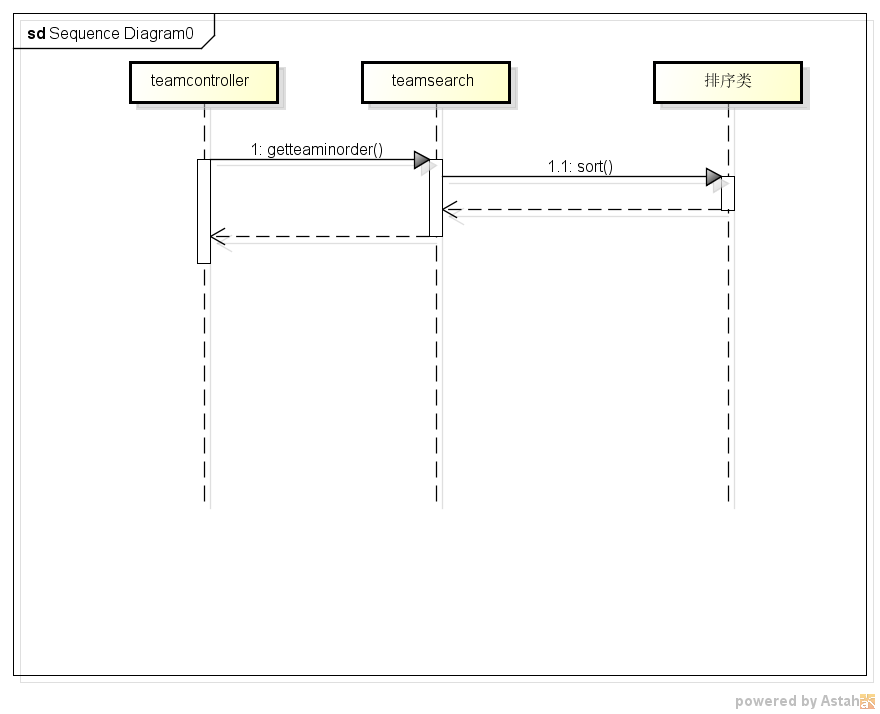
Teamsearch接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Teamsearch.sort | 语法 | public ArrayList<String> sort(ArrayList<TeamVO> v,Order order,TeamStandard stan) |
| 前置条件 | 无 |
| 后置条件 | 返回排序好的球队名称 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 排序类 | 具体排序方法 |

（4）业务逻辑层动态模型

下图显示了排序时业务逻辑层各个相关对象的协作



（5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑有各自的控制器委托给不同的领域对象。

## 4.2数据层的分解

### 4.2.1 teamData模块

（1）模块概述

teamData模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

TeamData模块的职责及接口参见软件体系结构文档

（2）整体结构

teamDataService接口用于隔离逻辑层对数据层的访问

TeamData是该模块的核心类，承担了bl层所需的所有访问需求

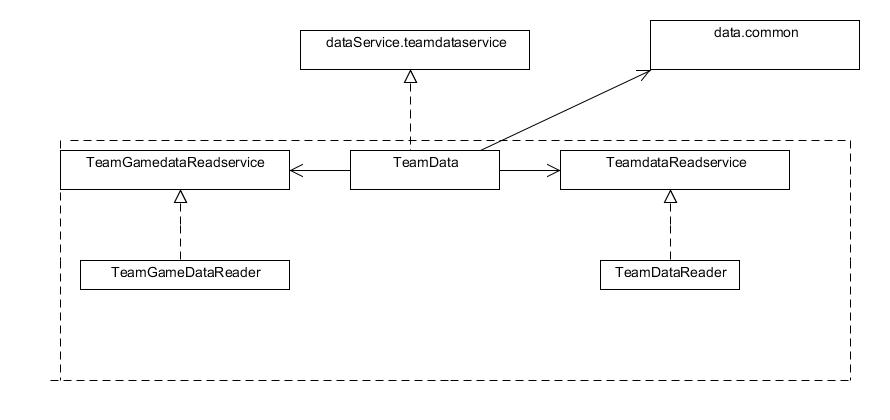
TeamdataReadService是对球队基本信息读取的接口

TeamDataReader实现了TeamdataReadService

TeamGamedataReadService是球队比赛数据读取的接口

TeamGameDataReader实现了TeamGameDataReadService

该模块用两个读文件接口隔离了对不同文件的读写，以应对未来可能出现的文件格式而引起的读文件实现的变更



（3）teamData模块各个类的职责

参见上一项目

（4）模块内部类接口规范

TeamData类的接口规范

表10 TeamData类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TeamData.getTeamData | 语法 | Public ArrayList<TeamPO> getTeamData(); |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有球队基本信息的Arraylist |
| TeamData.getTeamImage | 语法 | Public HashMap<String,Image> getTeamImage(); |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有球队名称及相对应图片的HashMap |
| TeamData.getTeamGameData | 语法 | Public ArrayList<TeamAllGamePO> getTeamGameData() |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有球队比赛数据的ArrayList |

需要的接口（需接口）

|  |  |
| --- | --- |
| 服务名 | 服务 |
| TeamDataReader.getTeamPO | 获取TeamPo的ArrayList |
| TeamDataReadr.getTeamImage | 获取球队的图片 |
| TeamGameDataReader.getTeamGameData | 获取球队的比赛数据 |

表11 TeamDataReader的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TeamDataReader.getTeamPO | 语法 | Public ArrayList<TeamPO> getTeamPO(); |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有球队基本信息的Arraylist |
| TeamData.getTeamImage | 语法 | Public HashMap<String，Image> getTeamImage(); |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有球队名称及相对应图片的HashMap |

需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Data.common.filter | 过滤文件行中的无效信息 |

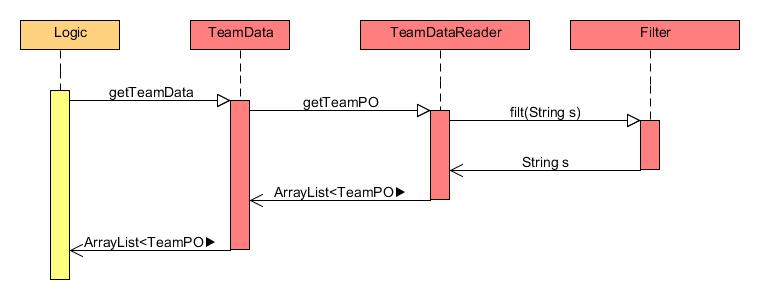
表12 TeamGameDataReader的接口规范

|  |  |  |
| --- | --- | --- |
| TeamGameDataReader.getTeamAllGamePO | 语法 | HashMap<String, TeamAllGamePO> getTeamAllGamePo() |
| 前置条件 | 无 |
| 后置条件 | 返回所有球队的比赛数据PO的Hashmap |

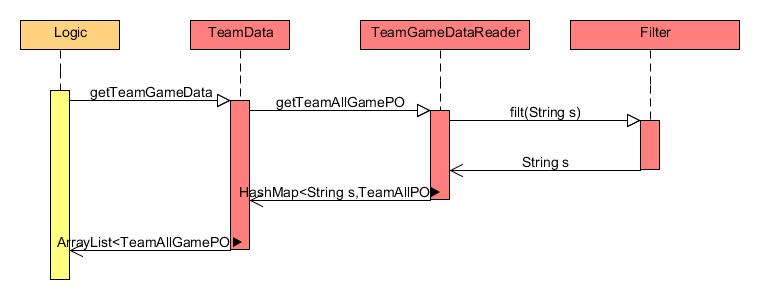
需要的服务（需接口）

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Data.common.filter | 过滤文件行中的无效信息 |

（4）动态模型



获取球队基本信息的顺序图



获取球队比赛数据的顺序图

### 4.2.2 PlayerData模块

（1）模块概述

同TeamData模块

（2）整体结构

PlayerDataService接口用于隔离逻辑层对数据层的访问

PlayerData是该模块的核心类，承担了bl层所需的所有访问需求

PlayerdataReadService是对球队基本信息读取的接口

PlayerDataReader实现了TeamdataReadService

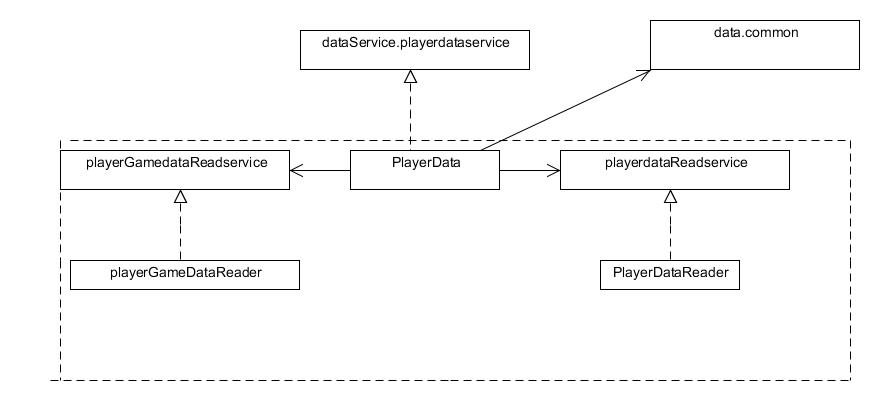
PlayerGamedataReadService是球队比赛数据读取的接口

PlayerGameDataReader实现了TeamGameDataReadService

该模块用两个读文件接口隔离了对不同文件的读写，以应对未来可能出现的文件格式而引起的读文件实现的变更

（3）PlayerData模块各个类的职责

参见上一项目



(4)模块内部类的接口规范

表13 PlayerData类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PlayerData.getPlayerData | 语法 | Public ArrayList<PlayerPO> getPlayerData(); |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有球员基本信息的Arraylist |
| TeamData.getPlayerGameData | 语法 | Public ArrayList<PlayerAllGamePO> getPlayerGameData() |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有球员比赛数据的ArrayList |

需要的接口（需接口）

|  |  |
| --- | --- |
| 服务名 | 服务 |
| PlayerDataReader.getPlayerPO | 获取TeamPo的ArrayList |
| PlayerGameDataReader.getTeamGameData | 获取球队的比赛数据 |

表14 PlayerDataReader的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PlayerDataReader.getPlayerPO | 语法 | Public ArrayList<PlayerPO> getPlayerPO(); |
| 前置条件 | 无 |
| 后置条件 | 返回包含所有球队基本信息的Arraylist |

需要的接口（需接口）

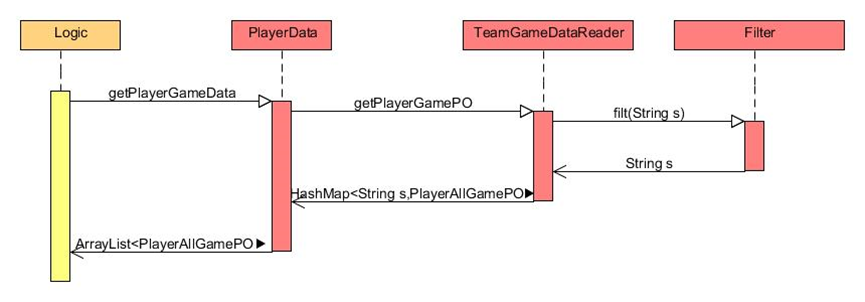
|  |  |
| --- | --- |
| 服务名 | 服务 |
| Data.common.filt | 过滤文件行中的无效信息 |

表15 PlayerGameDataReader的接口规范

|  |  |  |
| --- | --- | --- |
| PlayerGameDataReader.getPlayerGamePO | 语法 | HashMap<String, PlayerAllGamePO> getPlayerGamePo() |
| 前置条件 | 无 |
| 后置条件 | 返回所有球员的比赛数据PO的Hashmap |

需要的接口（需接口）

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Data.common.filt | 过滤文件行中的无效信息 |



# 5.依赖视角

