|  |  |
| --- | --- |
|  | Dungeon of Dooom |
|  | **CM50109 Coursework 2**  **Document Two** |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **[**team memberS**]**  **MATTSI JANSKY**  **ARYA NALINKUMAR**  **SELIN KUTLAMIS**  **ANASTASIOS GEMTOS**  **QIAN ZHOU**  **XIAOXIAO FAN** |

1. Contents

[Contents 1](#_Toc469612962)

[TEST PLAN 3](#_Toc469612963)

[Test-Driven Development 3](#_Toc469612964)

[Tests 3](#_Toc469612965)

[Project Diaries 0](#_Toc469612966)

[Mattsi Jansky: 0](#_Toc469612967)

[Anastasios Gemtos: 2](#_Toc469612968)

[Selin Kutlamis: 4](#_Toc469612969)

[Qian Zhou : 5](#_Toc469612970)

[Xiao Fan: 6](#_Toc469612971)

[Arya Nalinkumar: 9](#_Toc469612972)

[Maintenance Guide 0](#_Toc469612973)

[Java Overview 0](#_Toc469612974)

[Overview 0](#_Toc469612975)

[com.dod.db.DatabaseConnection 0](#_Toc469612976)

[com.dod.db.repositories.DatabaseRepository<T> 1](#_Toc469612977)

[com.dod.db.repositories.IPlayerRepository 3](#_Toc469612978)

[com.dod.db.repositories.IScoreRepository 4](#_Toc469612979)

[com.dod.db.repositories.ScoreRepository 6](#_Toc469612980)

[com.dod.db.repositories.PlayerRepository 9](#_Toc469612981)

[com.dod.game.IMatchList 11](#_Toc469612982)

[com.dod.game.MatchList 13](#_Toc469612983)

[com.dod.models.TileType 15](#_Toc469612984)

[com.dod.models.Tile 17](#_Toc469612985)

[com.dod.models.Score 18](#_Toc469612986)

[com.dod.models.Point 20](#_Toc469612987)

[com.dod.models.Player 21](#_Toc469612988)

[com.dod.models.MatchState 22](#_Toc469612989)

[com.dod.models.Match 24](#_Toc469612990)

[com.dod.models.Map 27](#_Toc469612991)

[com.dod.models.Character 30](#_Toc469612992)

[com.dod.service.controller.ScoreController 32](#_Toc469612993)

[com.dod.service.controller.PlayerController 33](#_Toc469612994)

[com.dod.service.controller.MatchController 34](#_Toc469612995)

[com.dod.service.controller.GameController 37](#_Toc469612996)

[com.dod.service.filters.corsFilter 38](#_Toc469612997)

[com.dod.service.model.TileModel 39](#_Toc469612998)

[com.dod.service.model.MatchStatus 40](#_Toc469612999)

[com.dod.service.model.MatchResultModel 41](#_Toc469613000)

[com.dod.service.model.LoginModel 42](#_Toc469613001)

[com.dod.service.model.GameStateModel 43](#_Toc469613002)

[com.dod.service.model.CharacterModel 46](#_Toc469613003)

[com.dod.service.service.VisibilityService 47](#_Toc469613004)

[com.dod.service.service.StateService 48](#_Toc469613005)

[com.dod.service.service.ParseService 49](#_Toc469613006)

[com.dod.service.service.MovementService 50](#_Toc469613007)

[com.dod.service.service.MatchService 51](#_Toc469613008)

[com.dod.service.service.IVisibilityService 54](#_Toc469613009)

[com.dod.service.service.IStateService 55](#_Toc469613010)

[com.dod.service.service.IParseService 55](#_Toc469613011)

[com.dod.service.service.IOService 56](#_Toc469613012)

[com.dod.service.service.IMovementService 57](#_Toc469613013)

[com.dod.service.service.IMatchService 58](#_Toc469613014)

[com.dod.service.service.IIOService 60](#_Toc469613015)

[com.dod.service.service.IAuthenticationService 61](#_Toc469613016)

[com.dod.service.service.AuthenticationService 62](#_Toc469613017)

[com.dod.service.Main 63](#_Toc469613018)

[Creating uses cases with server and client side 67](#_Toc469613019)

[References 77](#_Toc469613020)

1. TEST PLAN

Testing is a crucial part of designing a software system. Testing enables us to make a clear concise design decisions early in development and by providing tests that match these designs ensures that if these design decisions are changed, appropriate attention and fair warning will be given to their impact. Kaner (2006) suggests that tests should not be absolute and final but should start off simply and evolve over time with the system. In line with this we aim to start with few, basic tests and add new tests as we add new features via Test Driven Development.

In this sense, we are closer to using Exploratory Testing rather than Automated Testing- that is, the responsibility for running tests belongs with the developer and not an automated system. We must vigilantly run tests ourselves.

We intend to test the system using Unit and Integration Tests, both Black-Box and White-Box, including boundary cases. Unit testing will likely necessitate that we use stubs and build our system in a component-oriented or modular way (ISTQB Exam Certification, n.d.).

* 1. Test-Driven Development

We will follow test-driven development, writing interfaces or stubs of our components first and tests for those unimplemented components.

* 1. Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test name | Component being tested | Input | Expected output | Purpose |
| ShouldConnectToDatabase | DatabaseConnection | n/a | An open database connection | Can generate a database connection |
| ShouldCloseDatabase | DatabaseConnection | n/a |  | Can close a generated connection |

1. Project Diaries
   1. Mattsi Jansky:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday** |
| **31s Oct** | Attended first group meeting |  | Attended second group meeting |  |  | Reviewed lecture slides/notes and spec in prep for Monday meet | Reviewed lecture slides/notes and spec in prep for Monday meet |
| **7th Nov** | Attended group meeting  Gave presentation on findings  Worked with group on requirements analysis Introduced team to Trello & setup backlog/sprint boards  Kept meeting notes  Proposed choosing leader | Worked with Tassos to produce several usecases | Attended group meeting  Worked with group on CRC cards  Added some tasks to Trello |  |  |  |  |
| **14th Nov** | Attended group meeting  Worked with group on CRCs and task estimations | Produced timescale document, to be discussed with team Weds. | Attended team meeting  Introduced team to timescale document, made final changes to it. |  | Worked with Tassos on database creation, basic project layout and interfacing with database. |  |  |
| **21st Nov** | Attended team meeting  Worked on UML |  | Attended team meeting |  |  | Worked on Test Plan and researched Java API frameworks | Worked on Test Plan and researched Java API frameworks |
| **28th Nov** | Attended team meeting  Arranged next Sprint Created all remaining Trello cards and filled in various technical details of cards  Had meeting with Julian  Tested Java API framework | Setup Jersey API framework  Refactored project structure  Worked with Tassos to setup skeleton of project ie blank endpoints, services and tests | Worked with Qian to: implement IOservice, add JSON framework to project, start JSON service (implement tests) |  | Worked with Qian to implement JSON service |  |  |
| **5th Dec** | Worked with Selin and Arya to review their work on the login/registration card  Picked up the login/registration card and started work on it  Set up Sprint 5 in Trello  Attended team meeting  Attended team meeting with customer | Implemented AuthorisationService  Fixed web service setup issues  Implemented PlayerController  Added validation to web service  Started client-server interaction | Continued client-server interaction, implemented login/registration from client side  Attended team meeting  Implemented state memory architecture (MatchList)  Began implementing MatchService/ MatchController | Continued implementing MatchService/ MatchController  Improved state memory architecture  Implemented JSON support in service  Implemented JSON support in Java test client to test endpoints (and later for bot?) | Finished implementing MatchService / MatchController  Implemented StateService (except for visibility)  Implemented “Game/Status” endpoint  Implemented basic client lobbying | Implemented match details, joining a match, starting a new match in client  Implemented starting a match in client  Implemented basic rendering in client based on Tassos’ render code, added players to display | Implemented game loop & communication  Implemented automatic refresh for lobby list & match status  Improved graphics  Added functionality to leave a game, in server and client  Added StateService use of VisibilityService  Implemented basic use of visible tiles in client  Began to implement bot project Began to merge client design with prototype |
| **12th Dec** | Finished merging functional prototype with client design  Improved the client and fixed bugs  Added ability to load different levels  Fixed a lot of bugs  Added endgame state  Implemented graphical visibility in the client  Implemented scoreboard controller/action in server  Implemented scoreboard in client | Fixed a lot of bugs  Implemented simultaneous start between players |  |  |  |  |  |

* 1. Anastasios Gemtos:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday** |
| **31s Oct** | Team Meeting  (sprint 0) |  | Team Meeting |  |  |  |  |
| **7th Nov** | Team Meeting (sprint 1) | I worked on use cases with Mattsi about player actions in the dungeon. | Team Meeting (use cases, CRC) |  |  |  |  |
| **14th Nov** | Team Meeting (CRC, Task estimation)  (sprint 2) |  | Team Meeting  (Created UML classes and discussed about system architecture) |  | Worked with Mattsi on setting up the database. We implemented some functionality based on models from UML classes |  |  |
| **21st Nov** | Team Meeting  (Completed the UML classes both in client and server)  (sprint 3) |  | Team Meeting  (Reviewed all the documentation so far) |  |  |  | Updated the requirement analysis and the use cases on our documents. |
| **28th Nov** | Team Meeting  (worked on a document about document style and code style conventions, created draft overview of our work so far, meeting with customer)  (sprint 4) | Worked with Mattsi on the skeleton of web server’s framework. | Team Meeting  (Worked with Xiao on database) | Completed the database functionality. | Worked on rendering the dungeon’s map on web browser. | Completed the rendering of dungeon’s map on web browser. |  |
| **5th Dec** | Team Meeting  (Meeting with Julian to discuss our progress)  (sprint 5) | Worked on Character model. | Worked with Qian on Visibility on the server-side. | Completed the Visibility service to meet unit tests requirements. |  |  | Worked with Arya on Movement service. |
| **12th Dec** | Added coin collection functionality on movement service. Worked on movement on client side. Updated documentation. Team Meeting (with Julian to show him our playable demo).  (sprint 6) |  |  |  |  |  |  |

* 1. Selin Kutlamis:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday** |
| **31s Oct** | First group meeting, understand the document of project |  | Second group meeting,  Understand the document of project |  |  |  | Prep for Monday meet |
| **7th Nov** | Team meeting  Worked with group on requirements analysis Meet Trello  Brainstorming, Discussion  Add meeting notes  Choose leader |  | Attended group meeting  Worked with group on use cases |  |  |  |  |
| **14th Nov** | Group meeting  Worked with group on CRCs and task estimations |  | Write timescales Pair Groups are defined |  |  | Preparation for Uml diagrams | Investigate layout design |
| **21st Nov** | Uml diagrams creation |  | Menu/Login/Score Layout  Worked with Pair Programmer Arya | Customer Requirements /Reader’s Guide documentation | Customer Requirements /Reader’s Guide documentation | Creation of mock up | Creation of uml diagrams and continued to document checking. |
| **28th Nov** | Weekly meeting and discussion |  | Sprint meeting |  | Investigation on login and registration with MVC  Work with pair programmer Arya | Investigation on login and registration restful api | Investigation on login and registration restful api |
| **5th Dec** | Working with pair programmers Mattsi,Arya on login and registration  Meeting with customer |  | Sprint meeting discussion  Produce documentation layout |  | Work on documentation | Work on documentation | Work on documentation  Score Interaction investigation |
| **12th Dec** | Team meeting  Writing Document 1 and document2  Meeting with customer | Worked with the Pair Programmer Tasos on score/timer interaction | Team Meeting  Writing Document 1 and document2 |  |  |  |  |

* 1. Qian Zhou :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Mondy** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday** |
| **31s Oct** | Team Meeting |  | Team Meeting |  |  |  |  |
| **7th Nov** | Team Meeting (sprint0) – worked with group on the requirements analysis. | Worked with Xiao on the use cases about the user interaction before playing the game. | Team Meeting – worked with group on the use cases and CRC cards. |  |  |  |  |
| **14th Nov** | Team Meeting – worked with group on the CRC cards and task estimations. |  | Team Meeting - worked with group on the UML classes and system architecture. Discuss about the timescale and divided the tasks for pair programming. |  | Worked on the design of the map interface. | Worked on the design of the map interface. |  |
| **21st Nov** | Team Meeting - worked with group on the UML classes both in client and server. |  | Team Meeting – Worked on the documentation. |  | Create the Json files for different levels of the dungeon. | Create the Json files for the maps. Write the documentation about the Json files. |  |
| **28th Nov** | Team Meeting - separate the tasks in next sprint, had a meeting with Julian.  Worked with Xiao on determining the final design of our maps and finished the Json file. | Create and add two classes - Map and Tile to the project. | Worked with Mattis to implement the IOservice, add Json Framework to project, start Json service (including tests). |  | Worked with Mattis to implement Json service (parse map Json, Model map in code). |  |  |
| **5th Dec** | Team Meeting – separate the tasks in Sprint 5. |  | Worked with Tasos on the visibility in server. |  |  |  |  |
| **12th Dec** | Team Meeting – list the unfinished work both for codes and documentation, and arrange the tasks in the final week. | Worked on the interface part of the documentation. | Worked on the documentation. |  |  |  |  |

* 1. Xiao Fan:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday** |
| **31s Oct** | Team Meeting |  | Team Meeting |  |  |  |  |
| **7th Nov** | Team Meeting (sprint 1) | I worked on a use case with Qian about log in and menu design in the dungeon. | Team Meeting (use cases, CRC)  (I took the Meeting Minutes) |  |  |  |  |
| **14th Nov** | Team Meeting (the rest of CRC)  (sprint 2) |  | Team Meeting (Task estimation) |  |  |  |  |
| **21st Nov** | Team Meeting (UML, Diagram)  (sprint 3) |  | Team Meeting (Client UML, Diagram, documentation tasks distribution) |  |  | Learn JSON | Dungeon interface |
| **28th Nov** | Team Meeting  Create JSON file about the interface of Dungeon(Qian)  (sprint 4) |  | Creating database  Build play model,  Connect SQL with JAVA(Tasos), Building web service framework |  |  |  |  |
| **5th Dec** | Researching web game named Forestry maze |  |  |  |  |  |  |
| **2th Dec** | Team meeting (found the unfinished work and divided to every team member, have discuss with Julian) | Finding the reference about UML Diagram, Writing documentation about UML Diagram | Writing documentation about UML Diagram |  |  |  |  |

* 1. Arya Nalinkumar:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday** |
| **31s Oct** | First group meeting |  | Second meeting, gone through the documentation of project |  |  |  |  |
| **7th Nov** |  | Worked on use cases for winning condition | Group meeting  Gone through use cases & CRC cards |  |  |  |  |
| **14th Nov** | Group meeting (Sprint 2)  Continued work on CRC cards |  | Group meeting  Discussed on System architecture |  |  |  |  |
| **21st Nov** | Group meeting  (Sprint 3) |  | Worked with Selin on template: Login, Menu & Score table section | Completed template in Score table section | Worked on documenting the system architecture |  | Completed System architecture document |
| **28th Nov** | Group meeting  (Sprint 4)  Checked the documents with team members  Customer meeting |  | Group meeting |  | Discussed with Selin on login registration functionality. | Worked on login functionality- Done the Sql query to database in login & registration section. | Continued work on login & Register on client side. |
| **5th Dec** | (Sprint 5)  Checked the login & register functionality with Selin & Mattsi.  Discussed the modification to template.  Customer Meeting | Worked on new template changes.  Completed  Login/Registration,  Lobby & Score table  Template section | Group meeting  Shown the new template and started work on How to play section & result Screen | Completed new templates work |  |  | Worked with Tassos on Player Movement  Service. |
| **12th Dec** | (Sprint 6)  Group meeting  Discussed the task need to complete  Customer meeting | Completed the content of user guide.  Updated documentation  Worked on new tutorial section.  (Change in game screen) | Team meeting  Checked the documents with team members.  Added new screenshots on the user guide & modification on How to play section in template. |  |  |  |  |

1. Maintenance Guide

* 1. Overview

This maintenance guide will attempt to assist future engineers in understanding, fixing and improving the “Dungeon of Dooom” source code. cover two distinct components to the project source code- our Java code and our Javascript code.

* 1. Java Overview

We have three Java projects: The domain library, the web service, and the bot. The domain contains models and database functionality common across the projects. The service is a web API using the Jersey framework that hosts a number of “matches” in-memory each of which has a “map” and some “players”. The service has endpoints allowing players to do such things as move around the map and when the players interact with some tiles special game events occur.

The bot is a (very simple) agent that uses Jersey’s client to connect to the web service and make automated requests, in order to make a bot character move around the screen- using the exact same API that the client for humans uses.

The package structure we use follows this project separation:

* com.dod- the root Dungeon of Doom (“dod”) package.
* com.dod.db- database classes
* com.dod.game- domain classes that relate game logic, that aren’t beans or models.
* com.dod.models- models, mostly simple beans
* com.dod.service- the web service
* com.dod.service.controller- controllers for the web service
* com.dod.service.filters- Jersey API filters
* com.dod.service.model- JAXB annotated models for returning JSON data from the service.
* com.dod.service.service- services that perform game logic functions. These generic services ensure separation of functionality from controllers and allow us to re-use that functionaliy between controllers.
* com.dod.bot- the root of the bot source code
* com.dod.bot.communicators- the “Communicator” classes that the bot uses to contact the web service.

In the following sections, we detail each individual Java class using individual documentation pages generated through Javadoc.

* 1. com.dod.db.DatabaseConnection

public class DatabaseConnection

extends java.lang.Object

Stores a connection to the database using the singleton pattern

|  |  |
| --- | --- |
| Constructors | |
| [DatabaseConnection](file:///C:\Users\Fortnox\AppData\com\dod\db\DatabaseConnection.html#DatabaseConnection--)() | |
| All Methods | |
| Modifier and Type | Method and Description |
| static void | [Close](file:///C:\Users\Fortnox\AppData\com\dod\db\DatabaseConnection.html#Close--)()  Closes the connection |
| static java.sql.Connection | [getConnection](file:///C:\Users\Fortnox\AppData\com\dod\db\DatabaseConnection.html#getConnection--)()  A static connection to ensure that all sessions use the same MySql connection Could be done more intelligently with connection pooling |

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

*Constructor Detail*

DatabaseConnection

public DatabaseConnection()

*Method Detail*

Close

public static void Close()

Closes the connection

getConnection

* + - public static java.sql.Connection getConnection()

throws java.sql.SQLException

A static connection to ensure that all sessions use the same MySql connection Could be done more intelligently with connection pooling

Returns:

Connection instance

Throws:

java.sql.SQLException - when the database connection cannot be established

* 1. com.dod.db.repositories.DatabaseRepository<T>
* Direct Known Subclasses:

[PlayerRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\PlayerRepository.html), [ScoreRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\ScoreRepository.html)

public class DatabaseRepository<T>

extends java.lang.Object

A base class of the Repository pattern

Introduces the generic getStatement() method to reuse that code across the different repositories

*Field Summary*

|  |  |
| --- | --- |
| Fields | |
| Modifier and Type | Field and Description |
| protected java.sql.PreparedStatement | [ps](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#ps) |

*Constructor Summary*

|  |
| --- |
| Constructors |
| [DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#DatabaseRepository--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| boolean | [delete](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#delete-T-)([T](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html) object)  Make a DELETE query to delete the object in question from the database |
| [T](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html) | [get](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#get-T-)([T](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html) object)  Make a SELECT query to fetch the unique object in question from the database |
| protected java.sql.PreparedStatement | [getStatement](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#getStatement-java.lang.String-)(java.lang.String text)  Prepares a statement from a string using the database connection |
| boolean | [insert](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#insert-T-)([T](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html) object)  Make an INSERT query to insert the object in question into the database |

*Field Detail*

* + - ps

protected java.sql.PreparedStatement ps

*Constructor Detail*

* + - DatabaseRepository

public DatabaseRepository()

*Method Detail*

* + - delete
    - public boolean delete([T](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html) object)

throws java.sql.SQLException

Make a DELETE query to delete the object in question from the database

Parameters:

object - the object in question with the unique field (but not necessarily others) filled out

Returns:

true if successful, false otherwise

Throws:

java.sql.SQLException - when the statement fails

* + - get
    - public [T](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html) get([T](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html) object)

throws java.sql.SQLException

Make a SELECT query to fetch the unique object in question from the database

Parameters:

object - an instance of the object in question with the unique field (but not necessarily others) filled out

Returns:

An instance of the object

Throws:

java.sql.SQLException - if the statement fails or connection cannot be established

* + - getStatement
    - protected java.sql.PreparedStatement getStatement(java.lang.String text)

throws java.sql.SQLException

Prepares a statement from a string using the database connection

Parameters:

text - the text of the statement

Returns:

a PreparedStatement instance

Throws:

java.sql.SQLException - when the statement fails

* + - insert
    - public boolean insert([T](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html) object)

throws java.sql.SQLException

Make an INSERT query to insert the object in question into the database

Parameters:

object - the object in question

Returns:

true if successful, false otherwise

Throws:

java.sql.SQLException

* 1. com.dod.db.repositories.IPlayerRepository
* All Known Implementing Classes:

[PlayerRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\PlayerRepository.html)

public interface IPlayerRepository

Follows the Repository pattern.

Intended for selecting/inserting/deleting "Player" entries from the database.

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| boolean | [delete](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html#delete-com.dod.models.Player-)([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)  Make an INSERT query to insert the Player in question into the database |
| [Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) | [get](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html#get-com.dod.models.Player-)([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)  Make a SELECT query to fetch the unique Player in question from the database |
| boolean | [insert](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html#insert-com.dod.models.Player-)([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)  Make a DELETE query to delete the Player in question from the database |

*Method Detail*

* + - delete
    - boolean delete([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)

throws java.sql.SQLException

Make an INSERT query to insert the Player in question into the database

Parameters:

object - the Player in question

Returns:

true if successful, false otherwise

Throws:

java.sql.SQLException - when the statement fails

* + - get
    - [Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) get([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)

throws java.sql.SQLException

Make a SELECT query to fetch the unique Player in question from the database

Parameters:

object - an instance of the Player in question with the unique field (but not necessarily others) filled out

Returns:

Player object fetched from the database

Throws:

java.sql.SQLException - if the statement fails or connection cannot be established

* + - insert
    - boolean insert([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)

throws java.sql.SQLException

Make a DELETE query to delete the Player in question from the database

Parameters:

object - the Player in question with the unique field (but not necessarily others) filled out

Returns:

true if successful, false otherwise

Throws:

java.sql.SQLException - when the statement fails

* 1. com.dod.db.repositories.IScoreRepository
* All Known Implementing Classes:

[ScoreRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\ScoreRepository.html)

public interface IScoreRepository

Follows the Repository pattern.

Intended for selecting/inserting/deleting "Score" entries from the database.

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| boolean | [delete](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html#delete-com.dod.models.Score-)([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) object)  Make a DELETE query to delete the Score in question from the database |
| [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) | [get](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html#get-com.dod.models.Score-)([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) object)  Make a SELECT query to fetch the unique Score in question from the database |
| [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)[] | [getHighestScores](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html#getHighestScores--)()  Get the 10 highest scores from database |
| [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)[] | [getPlayerScores](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html#getPlayerScores-com.dod.models.Player-)([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)  Get the 10 highest scores of the player |
| boolean | [insert](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html#insert-com.dod.models.Score-)([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) object)  Make an INSERT query to insert the Score in question into the database |

*Method Detail*

* + - delete
    - boolean delete([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) object)

throws java.sql.SQLException

Make a DELETE query to delete the Score in question from the database

Parameters:

object - the Score in question with the unique field (but not necessarily others) filled out

Returns:

true if successful, false otherwise

Throws:

java.sql.SQLException

* + - get
    - [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) get([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) object)

throws java.sql.SQLException

Make a SELECT query to fetch the unique Score in question from the database

Parameters:

object - an instance of the Score in question with the unique field (but not necessarily others) filled out

Returns:

Score fetched from the database

Throws:

java.sql.SQLException - if the statement fails or connection cannot be established

* + - getHighestScores
    - [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)[] getHighestScores()

throws java.sql.SQLException

Get the 10 highest scores from database

Returns:

Score[] array of 10 Score objects

Throws:

java.sql.SQLException - when the statement fails

* + - getPlayerScores
    - [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)[] getPlayerScores([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)

throws java.sql.SQLException

Get the 10 highest scores of the player

Parameters:

object - Player object

Returns:

Score[] array of 10 Score objects

Throws:

java.sql.SQLException - when the statement fails

* + - insert
    - boolean insert([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) object)

throws java.sql.SQLException

Make an INSERT query to insert the Score in question into the database

Parameters:

object - the Score in question

Returns:

true if successful, false otherwise

Throws:

java.sql.SQLException

* 1. com.dod.db.repositories.ScoreRepository
* All Implemented Interfaces:

[IScoreRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html)

public class ScoreRepository

extends [DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)<[Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)>

implements [IScoreRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html)

Implements IPlayerRepository.

Follows the Repository pattern.

Intended for selecting/inserting/deleting "Score" entries from the database.

*Field Summary*

* + - Fields inherited from class com.dod.db.repositories.[DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)

[ps](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#ps)

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [ScoreRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\ScoreRepository.html#ScoreRepository--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| boolean | [delete](file:///C:\Users\Fortnox\com\dod\db\repositories\ScoreRepository.html#delete-com.dod.models.Score-)([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) object)  Delete a score row from database !! We should not use that. |
| [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) | [get](file:///C:\Users\Fortnox\com\dod\db\repositories\ScoreRepository.html#get-com.dod.models.Score-)([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) object)  returns a Score based on id from the database |
| [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)[] | [getHighestScores](file:///C:\Users\Fortnox\com\dod\db\repositories\ScoreRepository.html#getHighestScores--)()  Get the 10 highest scores from database |
| [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)[] | [getPlayerScores](file:///C:\Users\Fortnox\com\dod\db\repositories\ScoreRepository.html#getPlayerScores-com.dod.models.Player-)([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)  Get the 10 highest scores of the player |
| boolean | [insert](file:///C:\Users\Fortnox\com\dod\db\repositories\ScoreRepository.html#insert-com.dod.models.Score-)([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) scoreObject)  Inserts a score value to score table of database based on player's username. |

* + - Methods inherited from class com.dod.db.repositories.[DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)

[getStatement](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#getStatement-java.lang.String-)

*Constructor Detail*

* + - ScoreRepository

public ScoreRepository()

*Method Detail*

* + - delete
    - public boolean delete([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) object)

throws java.sql.SQLException

Delete a score row from database !! We should not use that.

Specified by:

[delete](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html#delete-com.dod.models.Score-) in interface [IScoreRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html)

Overrides:

[delete](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#delete-T-) in class [DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)<[Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)>

Parameters:

object - score object to delete

Returns:

true if the deletion was successful else false

Throws:

java.sql.SQLException - when the statement fails

* + - get
    - public [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) get([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) object)

throws java.sql.SQLException

returns a Score based on id from the database

Specified by:

[get](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html#get-com.dod.models.Score-) in interface [IScoreRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html)

Overrides:

[get](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#get-T-) in class [DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)<[Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)>

Parameters:

Score - to be fetched must have unique identifier populated

Returns:

Score object

Throws:

java.sql.SQLException - when the statement fails

* + - getHighestScores
    - public [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)[] getHighestScores()

throws java.sql.SQLException

Get the 10 highest scores from database

Specified by:

[getHighestScores](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html#getHighestScores--) in interface [IScoreRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html)

Returns:

Score[] array of 10 Score objects

Throws:

java.sql.SQLException - when the statement fails

* + - getPlayerScores
    - public [Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)[] getPlayerScores([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)

throws java.sql.SQLException

Get the 10 highest scores of the player

Specified by:

[getPlayerScores](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html#getPlayerScores-com.dod.models.Player-) in interface [IScoreRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html)

Parameters:

object - Player object

Returns:

Score[] array of 10 Score objects

Throws:

java.sql.SQLException - when the statement fails

* + - insert
    - public boolean insert([Score](file:///C:\Users\Fortnox\com\dod\models\Score.html) scoreObject)

throws java.sql.SQLException

Inserts a score value to score table of database based on player's username.

Specified by:

[insert](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html#insert-com.dod.models.Score-) in interface [IScoreRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IScoreRepository.html)

Overrides:

[insert](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#insert-T-) in class [DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)<[Score](file:///C:\Users\Fortnox\com\dod\models\Score.html)>

Parameters:

scoreObject - current score that we need to score

Returns:

true if insertion was successful else false

Throws:

java.sql.SQLException - when the statement fails

* 1. com.dod.db.repositories.PlayerRepository
* All Implemented Interfaces:

[IPlayerRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html)

public class PlayerRepository

extends [DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)<[Player](file:///C:\Users\Fortnox\com\dod\models\Player.html)>

implements [IPlayerRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html)

Implements IPlayerRepository.

Follows the Repository pattern.

Intended for selecting/inserting/deleting "Player" entries from the database.

*Field Summary*

* + - Fields inherited from class com.dod.db.repositories.[DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)

[ps](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#ps)

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [PlayerRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\PlayerRepository.html#PlayerRepository--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| boolean | [delete](file:///C:\Users\Fortnox\com\dod\db\repositories\PlayerRepository.html#delete-com.dod.models.Player-)([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)  Make an INSERT query to insert the Player in question into the database |
| [Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) | [get](file:///C:\Users\Fortnox\com\dod\db\repositories\PlayerRepository.html#get-com.dod.models.Player-)([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)  Make a SELECT query to fetch the unique Player in question from the database |
| boolean | [insert](file:///C:\Users\Fortnox\com\dod\db\repositories\PlayerRepository.html#insert-com.dod.models.Player-)([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)  Make a DELETE query to delete the Player in question from the database |

Methods *inherited* from class com.dod.db.repositories.[DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)

[getStatement](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#getStatement-java.lang.String-)

*Constructor Detail*

* + - PlayerRepository

public PlayerRepository()

*Method Detail*

* + - delete
    - public boolean delete([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)

throws java.sql.SQLException

Make an INSERT query to insert the Player in question into the database

Specified by:

[delete](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html#delete-com.dod.models.Player-) in interface [IPlayerRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html)

Overrides:

[delete](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#delete-T-) in class [DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)<[Player](file:///C:\Users\Fortnox\com\dod\models\Player.html)>

Parameters:

object - the Player in question

Returns:

true if successful, false otherwise

Throws:

java.sql.SQLException - when the statement fails

* + - get
    - public [Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) get([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)

throws java.sql.SQLException

Make a SELECT query to fetch the unique Player in question from the database

Specified by:

[get](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html#get-com.dod.models.Player-) in interface [IPlayerRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html)

Overrides:

[get](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#get-T-) in class [DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)<[Player](file:///C:\Users\Fortnox\com\dod\models\Player.html)>

Parameters:

object - an instance of the Player in question with the unique field (but not necessarily others) filled out

Returns:

Player object fetched from the database

Throws:

java.sql.SQLException - if the statement fails or connection cannot be established

* + - insert
    - public boolean insert([Player](file:///C:\Users\Fortnox\com\dod\models\Player.html) object)

throws java.sql.SQLException

Make a DELETE query to delete the Player in question from the database

Specified by:

[insert](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html#insert-com.dod.models.Player-) in interface [IPlayerRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\IPlayerRepository.html)

Overrides:

[insert](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html#insert-T-) in class [DatabaseRepository](file:///C:\Users\Fortnox\com\dod\db\repositories\DatabaseRepository.html)<[Player](file:///C:\Users\Fortnox\com\dod\models\Player.html)>

Parameters:

object - the Player in question with the unique field (but not necessarily others) filled out

Returns:

true if successful, false otherwise

Throws:

java.sql.SQLException - when the statement fails

* 1. com.dod.game.IMatchList
* All Known Implementing Classes:

[MatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\MatchList.html)

public interface IMatchList

Stores ongoing matches in memory and provides functions to access these matches.

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| void | [addMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#addMatch-com.dod.models.Match-)([Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) match)  Returns a singleton instance of MatchList, creating it if it hasn't been initialised yet. |
| java.util.List<[Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html)> | [getLobbyingMatches](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#getLobbyingMatches--)()  Gets all matches that are in the Lobbying state |
| [Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) | [getMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#getMatch-java.util.UUID-)(java.util.UUID id)  Gets a Match by a particular ID. |
| [Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) | [getMatchForPlayer](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#getMatchForPlayer-java.lang.String-)(java.lang.String username)  Gets a match by player name. |
| boolean | [playerHasMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#playerHasMatch-java.lang.String-)(java.lang.String username)  Returns true if the player has a match in the list |
| void | [removeMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#removeMatch-java.util.UUID-)(java.util.UUID id)  Removes the match fitting the specified ID from the list |

*Method Detail*

* + - addMatch

void addMatch([Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) match)

Returns a singleton instance of MatchList, creating it if it hasn't been initialised yet.

* + - getLobbyingMatches

java.util.List<[Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html)> getLobbyingMatches()

Gets all matches that are in the Lobbying state

Returns:

List of Match objects

* + - getMatch

[Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) getMatch(java.util.UUID id)

Gets a Match by a particular ID. Returns null if the match is missing.

Parameters:

id - the UUID that corresponds to the match to be fetched

Returns:

Match

* + - getMatchForPlayer

[Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) getMatchForPlayer(java.lang.String username)

Gets a match by player name. Each player should only have one match. Returns null if player has no match.

Parameters:

username - the username of the player

Returns:

Match

* + - playerHasMatch

boolean playerHasMatch(java.lang.String username)

Returns true if the player has a match in the list

Parameters:

username - the player's username

Returns:

true if the player has a match in the list otherwise false

* + - removeMatch

void removeMatch(java.util.UUID id)

Removes the match fitting the specified ID from the list

Parameters:

id - the UUID that corresponds to the particular Match to be removed

* 1. com.dod.game.MatchList
* All Implemented Interfaces:

[IMatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html)

public class MatchList

extends java.lang.Object

implements [IMatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html)

Implementation of IMatchList

Stores ongoing matches in memory and provides functions to access these matches.

Uses a singleton so that we can fetch the same object between requests

(And because this is much easier to test than making all methods static)

*Constructor Summary*

|  |
| --- |
| Constructors |
| [MatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\MatchList.html#MatchList--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| void | [addMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\MatchList.html#addMatch-com.dod.models.Match-)([Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) match)  Adds a match to the list |
| java.util.List<[Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html)> | [getLobbyingMatches](file:///C:\Users\Fortnox\AppData\com\dod\game\MatchList.html#getLobbyingMatches--)()  Gets all matches that are in the Lobbying state |
| [Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) | [getMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\MatchList.html#getMatch-java.util.UUID-)(java.util.UUID id)  Gets a Match by a particular ID. |
| [Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) | [getMatchForPlayer](file:///C:\Users\Fortnox\AppData\com\dod\game\MatchList.html#getMatchForPlayer-java.lang.String-)(java.lang.String username)  Gets a match by player name. |
| static [IMatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html) | [instance](file:///C:\Users\Fortnox\AppData\com\dod\game\MatchList.html#instance--)()  Returns a singleton instance of MatchList, creating it if it hasn't been initialised yet. |
| boolean | [playerHasMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\MatchList.html#playerHasMatch-java.lang.String-)(java.lang.String username)  Returns true if the player has a match in the list |
| void | [removeMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\MatchList.html#removeMatch-java.util.UUID-)(java.util.UUID id)  Removes the match fitting the specified ID from the list |

*Constructor Detail*

* + - MatchList

public MatchList()

*Method Detail*

* + - addMatch

public void addMatch([Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) match)

Adds a match to the list

Specified by:

[addMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#addMatch-com.dod.models.Match-) in interface [IMatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html)

Parameters:

match - the match to add

* + - getLobbyingMatches

public java.util.List<[Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html)> getLobbyingMatches()

Gets all matches that are in the Lobbying state

Specified by:

[getLobbyingMatches](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#getLobbyingMatches--) in interface [IMatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html)

Returns:

List of Match objects

* + - getMatch

public [Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) getMatch(java.util.UUID id)

Gets a Match by a particular ID. Returns null if the match is missing.

Specified by:

[getMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#getMatch-java.util.UUID-) in interface [IMatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html)

Parameters:

id - the UUID that corresponds to the match to be fetched

Returns:

Match

* + - getMatchForPlayer

public [Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html) getMatchForPlayer(java.lang.String username)

Gets a match by player name. Each player should only have one match. Returns null if player has no match.

Specified by:

[getMatchForPlayer](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#getMatchForPlayer-java.lang.String-) in interface [IMatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html)

Parameters:

username - the username of the player

Returns:

Match

* + - instance

public static [IMatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html) instance()

Returns a singleton instance of MatchList, creating it if it hasn't been initialised yet.

Returns:

MatchList

* + - playerHasMatch

public boolean playerHasMatch(java.lang.String username)

Returns true if the player has a match in the list

Specified by:

[playerHasMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#playerHasMatch-java.lang.String-) in interface [IMatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html)

Parameters:

username - the player's username

Returns:

true if the player has a match in the list otherwise false

* + - removeMatch

public void removeMatch(java.util.UUID id)

Removes the match fitting the specified ID from the list

Specified by:

[removeMatch](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html#removeMatch-java.util.UUID-) in interface [IMatchList](file:///C:\Users\Fortnox\AppData\com\dod\game\IMatchList.html)

Parameters:

id - the UUID that corresponds to the particular Match to be removed

* 1. com.dod.models.TileType
* All Implemented Interfaces:

java.io.Serializable, java.lang.Comparable<[TileType](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html)>

public enum TileType

extends java.lang.Enum<[TileType](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html)>

The type of a tile, i.e is this tile a wall, floor or something else.

*Enum Constant Summary*

|  |
| --- |
| Enum Constants |
| Enum Constant and Description |
| [Coin](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html#Coin) |
| [Empty](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html#Empty) |
| [Exit](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html#Exit) |
| [Wall](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html#Wall) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| int | [getValue](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html#getValue--)() |
| static [TileType](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html) | [valueOf](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html#valueOf-java.lang.String-)(java.lang.String name)  Returns the enum constant of this type with the specified name. |
| static [TileType](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html)[] | [values](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html#values--)()  Returns an array containing the constants of this enum type, in the order they are declared. |

*Enum Constant Detail*

* + - Coin

public static final [TileType](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html) Coin

* + - Empty

public static final [TileType](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html) Empty

* + - Exit

public static final [TileType](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html) Exit

* + - Wall

public static final [TileType](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html) Wall

*Method Detail*

* + - getValue

public int getValue()

* + - valueOf

public static [TileType](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html) valueOf(java.lang.String name)

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

name - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

java.lang.IllegalArgumentException - if this enum type has no constant with the specified name

java.lang.NullPointerException - if the argument is null

* + - values

public static [TileType](file:///C:\Users\Fortnox\AppData\com\dod\models\TileType.html)[] values()

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

for (TileType c : TileType.values())

  System.out.println(c);

Returns:

an array containing the constants of this enum type, in the order they are declared

* 1. com.dod.models.Tile

public class Tile

extends java.lang.Object

A Tile represents single tile on the grid that is the Map

A Tile has a Type that indicates whether it is eg a wall, floor, coin or exit tile.

A Tile may or may not be visible

*Field Summary*

|  |  |
| --- | --- |
| Fields | |
| Modifier and Type | Field and Description |
| protected int | [type](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html#type) |
| protected boolean | [visibility](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html#visibility) |

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Tile](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html#Tile-int-)(int type) |
| [Tile](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html#Tile-int-boolean-)(int type, boolean visibility) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| int | [getType](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html#getType--)() |
| boolean | [isVisible](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html#isVisible--)() |
| void | [setType](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html#setType-int-)(int type) |
| void | [setVisibility](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html#setVisibility-boolean-)(boolean visibility) |
| java.lang.String | [toString](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html#toString--)() |

*Field Detail*

* + - type

protected int type

* + - visibility

protected boolean visibility

*Constructor Detail*

* + - Tile

public Tile(int type)

* + - Tile
    - public Tile(int type,

boolean visibility)

*Method Detail*

* + - getType

public int getType()

* + - isVisible

public boolean isVisible()

* + - setType

public void setType(int type)

* + - setVisibility

public void setVisibility(boolean visibility)

* + - toString

public java.lang.String toString()

Overrides:

toString in class java.lang.Object

* 1. com.dod.models.Score

public class Score

extends java.lang.Object

A Score stores the points a Player achieved when they completed a Match.

A Score as an ID in order to store the Score as a unique databaes record

A Score also has a value and the username of the player that the score is related to.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Score](file:///C:\Users\Fortnox\AppData\com\dod\models\Score.html#Score-int-java.lang.String-int-)(int id, java.lang.String username, int value) |
| [Score](file:///C:\Users\Fortnox\AppData\com\dod\models\Score.html#Score-java.lang.String-int-)(java.lang.String username, int value) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| int | [getId](file:///C:\Users\Fortnox\AppData\com\dod\models\Score.html#getId--)() |
| java.lang.String | [getUsername](file:///C:\Users\Fortnox\AppData\com\dod\models\Score.html#getUsername--)() |
| int | [getValue](file:///C:\Users\Fortnox\AppData\com\dod\models\Score.html#getValue--)() |
| void | [setId](file:///C:\Users\Fortnox\AppData\com\dod\models\Score.html#setId-int-)(int id) |
| void | [setUsername](file:///C:\Users\Fortnox\AppData\com\dod\models\Score.html#setUsername-java.lang.String-)(java.lang.String username) |
| void | [setValue](file:///C:\Users\Fortnox\AppData\com\dod\models\Score.html#setValue-int-)(int value) |

*Constructor Detail*

* + - Score
    - public Score(int id,
    - java.lang.String username,

int value)

* + - Score
    - public Score(java.lang.String username,

int value)

*Method Detail*

* + - getId

public int getId()

* + - getUsername

public java.lang.String getUsername()

* + - getValue

public int getValue()

* + - setId

public void setId(int id)

* + - setUsername

public void setUsername(java.lang.String username)

* + - setValue

public void setValue(int value)

* 1. com.dod.models.Point

public class Point

extends java.lang.Object

Bean class for storing a point (or vertex) in the map.

*Field Summary*

|  |  |
| --- | --- |
| Fields | |
| Modifier and Type | Field and Description |
| int | [x](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html#x) |
| int | [y](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html#y) |

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html#Point--)() |
| [Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html#Point-int-int-)(int x, int y) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| boolean | [equals](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html#equals-java.lang.Object-)(java.lang.Object obj) |

*Field Detail*

* + - x

public int x

* + - y

public int y

*Constructor Detail*

* + - Point

public Point()

* + - Point
    - public Point(int x,

int y)

*Method Detail*

* + - equals

public boolean equals(java.lang.Object obj)

Overrides:

equals in class java.lang.Object

* 1. com.dod.models.Player

public class Player

extends java.lang.Object

A Player represents the user that is in control of the game client

A Player can sign in with a username or password

A Player has a level and a password salt

A Player's password is always hashed

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Player](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html#Player-java.lang.String-)(java.lang.String name) |
| [Player](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html#Player-java.lang.String-java.lang.String-byte:A-)(java.lang.String name, java.lang.String hashedPassword, byte[] salt) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| java.lang.String | [getHashedPassword](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html#getHashedPassword--)() |
| int | [getLevel](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html#getLevel--)() |
| byte[] | [getSalt](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html#getSalt--)() |
| java.lang.String | [getUsername](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html#getUsername--)() |
| void | [setHashedPassword](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html#setHashedPassword-java.lang.String-)(java.lang.String hashedPassword) |
| void | [setLevel](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html#setLevel-int-)(int level) |
| void | [setSalt](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html#setSalt-byte:A-)(byte[] salt) |
| void | [setUsername](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html#setUsername-java.lang.String-)(java.lang.String value) |

* + - Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

*Constructor Detail*

* + - Player

public Player(java.lang.String name)

* + - Player
    - public Player(java.lang.String name,
    - java.lang.String hashedPassword,

byte[] salt)

*Method Detail*

* + - getHashedPassword

public java.lang.String getHashedPassword()

* + - getLevel

public int getLevel()

* + - getSalt

public byte[] getSalt()

* + - getUsername

public java.lang.String getUsername()

* + - setHashedPassword

public void setHashedPassword(java.lang.String hashedPassword)

* + - setLevel

public void setLevel(int level)

* + - setSalt

public void setSalt(byte[] salt)

* + - setUsername

public void setUsername(java.lang.String value)

* 1. com.dod.models.MatchState
* All Implemented Interfaces:

java.io.Serializable, java.lang.Comparable<[MatchState](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html)>

public enum MatchState

extends java.lang.Enum<[MatchState](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html)>

The state of a Match.

*Enum Constant Summary*

|  |
| --- |
| Enum Constants |
| Enum Constant and Description |
| [Ingame](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html#Ingame) |
| [Lobbying](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html#Lobbying) |
| [Over](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html#Over) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| static [MatchState](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html) | [valueOf](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html#valueOf-java.lang.String-)(java.lang.String name)  Returns the enum constant of this type with the specified name. |
| static [MatchState](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html)[] | [values](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html#values--)()  Returns an array containing the constants of this enum type, in the order they are declared. |

*Enum Constant Detail*

* + - Ingame

public static final [MatchState](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html) Ingame

* + - Lobbying

public static final [MatchState](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html) Lobbying

* + - Over

public static final [MatchState](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html) Over

*Method Detail*

* + - valueOf

public static [MatchState](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html) valueOf(java.lang.String name)

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

name - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

java.lang.IllegalArgumentException - if this enum type has no constant with the specified name

java.lang.NullPointerException - if the argument is null

*values*

public static [MatchState](file:///C:\Users\Fortnox\AppData\com\dod\models\MatchState.html)[] values()

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

for (MatchState c : MatchState.values())

  System.out.println(c);

Returns:

an array containing the constants of this enum type, in the order they are declared

* 1. com.dod.models.Match

public class Match

extends java.lang.Object

A Match represents a particular collection of Players that are playing on a particular Map stored in memory

A Match has a Map

A Match has a unique ID

A Match

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Match](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#Match-com.dod.models.Map-)(com.dod.models.Map map) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| void | [addCharacter](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#addCharacter-com.dod.models.Player-com.dod.models.Point-)(com.dod.models.Player player, com.dod.models.Point position)  Adds a Player to this Match with a new Character |
| com.dod.models.Character | [getCharacter](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#getCharacter-java.lang.String-)(java.lang.String username) |
| java.util.List<com.dod.models.Character> | [getCharactersOnTile](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#getCharactersOnTile-com.dod.models.Point-)(com.dod.models.Point point)  Returns all Characters on a particular Tile |
| com.dod.models.Character | [getCharacterWithHighestCoins](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#getCharacterWithHighestCoins--)()  Gets the Caracter with the highest score |
| java.util.UUID | [getId](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#getId--)() |
| com.dod.models.Map | [getMap](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#getMap--)() |
| java.lang.String[] | [getPlayerNames](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#getPlayerNames--)()  Gets a list of names of each Player currently in this Match |
| int | [getScore](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#getScore--)() |
| com.dod.models.MatchState | [getState](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#getState--)() |
| long | [getTimer](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#getTimer--)() |
| boolean | [hasCharacter](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#hasCharacter-java.lang.String-)(java.lang.String userName)  Returns where or not a character is in this Match |
| void | [removeCharacter](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#removeCharacter-com.dod.models.Player-)(com.dod.models.Player player) |
| void | [setScore](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#setScore-int-)(int score) |
| void | [setState](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#setState-com.dod.models.MatchState-)(com.dod.models.MatchState state) |
| void | [setTimer](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#setTimer-long-)(long timer) |
| void | [startGame](file:///C:\Users\Fortnox\AppData\com\dod\models\Match.html#startGame--)() |

*Constructor Detail*

* + - Match

public Match(com.dod.models.Map map)

*Method Detail*

* + - addCharacter
    - public void addCharacter(com.dod.models.Player player,

com.dod.models.Point position)

Adds a Player to this Match with a new Character

Parameters:

player - Player the Player who will join this Match as a Character

position - Point the position the new Character will occupy

* + - getCharacter

public com.dod.models.Character getCharacter(java.lang.String username)

* + - getCharactersOnTile

public java.util.List<com.dod.models.Character> getCharactersOnTile(com.dod.models.Point point)

Returns all Characters on a particular Tile

Parameters:

point - Point the location of the Tile to check

Returns:

List\ a list of Characters that are presently standing on that tile

* + - getCharacterWithHighestCoins

public com.dod.models.Character getCharacterWithHighestCoins()

Gets the Caracter with the highest score

Returns:

Character with the highest score

* + - getId

public java.util.UUID getId()

* + - getMap

public com.dod.models.Map getMap()

* + - getPlayerNames

public java.lang.String[] getPlayerNames()

Gets a list of names of each Player currently in this Match

Returns:

String[] array of players names

* + - getScore

public int getScore()

* + - getState

public com.dod.models.MatchState getState()

* + - getTimer

public long getTimer()

* + - hasCharacter

public boolean hasCharacter(java.lang.String userName)

Returns where or not a character is in this Match

Parameters:

userName - String the name of the Player to check

Returns:

boolean true if the Player is in this Match otherwise false

* + - removeCharacter

public void removeCharacter(com.dod.models.Player player)

* + - setScore

public void setScore(int score)

* + - setState

public void setState(com.dod.models.MatchState state)

* + - setTimer

public void setTimer(long timer)

* + - startGame

public void startGame()

* 1. com.dod.models.Map

public class Map

extends java.lang.Object

A Map stores a 2-dimensional grid of Tiles.

A Map has a name, width, height and numbe rof coins total and required to win.

*Field Summary*

|  |  |
| --- | --- |
| Fields | |
| Modifier and Type | Field and Description |
| protected int | [height](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#height) |
| protected java.lang.String | [name](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#name) |
| protected int | [numberOfCoinsNeededToWin](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#numberOfCoinsNeededToWin) |
| protected [Tile](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html)[][] | [tiles](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#tiles) |
| protected int | [totalNumberOfCoins](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#totalNumberOfCoins) |
| protected int | [width](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#width) |

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Map](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#Map-int-int-)(int width, int height) |
| [Map](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#Map-java.lang.String-int-int-int-int-com.dod.models.Point-)(java.lang.String name, int totalNumberOfCoins, int numberOfCoinsNeededToWin, int width, int height, [Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) mapSize) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| int | [getCoinNo](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#getCoinNo--)()  The total number of coins that should be created in the map. |
| int | [getCoinWin](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#getCoinWin--)()  The total number of coins needed to win on this map |
| int | [getHeight](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#getHeight--)() |
| java.lang.String | [getName](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#getName--)() |
| [Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) | [getRandomFreeTilePoint](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#getRandomFreeTilePoint--)()  Gets a random position of a tile that is not a wall, coin or exit. |
| [Tile](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html) | [getTile](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#getTile-com.dod.models.Point-)([Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) point) |
| int | [getWidth](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#getWidth--)() |
| void | [setCoinNo](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#setCoinNo-int-)(int coin\_no)  The total number of coins that should be created in the map. |
| void | [setCoinWin](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#setCoinWin-int-)(int coin\_win)  The total number of coins needed to win on this map |
| void | [setName](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#setName-java.lang.String-)(java.lang.String name) |
| void | [setTile](file:///C:\Users\Fortnox\AppData\com\dod\models\Map.html#setTile-com.dod.models.Point-com.dod.models.Tile-)([Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) position, [Tile](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html) tile) |

*Field Detail*

* + - height

protected int height

* + - name

protected java.lang.String name

* + - numberOfCoinsNeededToWin

protected int numberOfCoinsNeededToWin

* + - tiles

protected [Tile](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html)[][] tiles

* + - totalNumberOfCoins

protected int totalNumberOfCoins

* + - width

protected int width

*Constructor Detail*

* + - Map
    - public Map(int width,

int height)

* + - Map
    - public Map(java.lang.String name,
    - int totalNumberOfCoins,
    - int numberOfCoinsNeededToWin,
    - int width,
    - int height,

[Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) mapSize)

*Method Detail*

* + - getCoinNo

public int getCoinNo()

The total number of coins that should be created in the map.

Returns:

int

* + - getCoinWin

public int getCoinWin()

The total number of coins needed to win on this map

Returns:

int

* + - getHeight

public int getHeight()

* + - getName

public java.lang.String getName()

* + - getRandomFreeTilePoint

public [Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) getRandomFreeTilePoint()

Gets a random position of a tile that is not a wall, coin or exit.

Returns:

Point

* + - getTile

public [Tile](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html) getTile([Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) point)

* + - getWidth

public int getWidth()

* + - setCoinNo

public void setCoinNo(int coin\_no)

The total number of coins that should be created in the map.

Parameters:

coin\_no - int

* + - setCoinWin

public void setCoinWin(int coin\_win)

The total number of coins needed to win on this map

Parameters:

coin\_win - int

* + - setName

public void setName(java.lang.String name)

* + - setTile
    - public void setTile([Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) position,

[Tile](file:///C:\Users\Fortnox\AppData\com\dod\models\Tile.html) tile)

* 1. com.dod.models.Character

public class Character

extends java.lang.Object

A Character is a fictional entity that moves around the game world.

A Character belongs to a Player.

A Character has a position and can interact with coins and the exit.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Character](file:///C:\Users\Fortnox\AppData\com\dod\models\Character.html#Character-com.dod.models.Point-com.dod.models.Player-)([Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) position, [Player](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html) player) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| void | [addCollectedCoinsPos](file:///C:\Users\Fortnox\AppData\com\dod\models\Character.html#addCollectedCoinsPos-com.dod.models.Point-)([Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) newPoint)  Keeps track of which coins on the map this Character has collected. |
| int | [getCollectedCoins](file:///C:\Users\Fortnox\AppData\com\dod\models\Character.html#getCollectedCoins--)() |
| java.util.List<[Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html)> | [getCollectedCoinsPos](file:///C:\Users\Fortnox\AppData\com\dod\models\Character.html#getCollectedCoinsPos--)()  Keeps track of which coins on the map this Character has collected. |
| [Player](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html) | [getPlayer](file:///C:\Users\Fortnox\AppData\com\dod\models\Character.html#getPlayer--)()  The Player that this Character belongs to |
| [Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) | [getPosition](file:///C:\Users\Fortnox\AppData\com\dod\models\Character.html#getPosition--)()  The player's position in the game world |
| void | [setCollectedCoins](file:///C:\Users\Fortnox\AppData\com\dod\models\Character.html#setCollectedCoins-int-)(int collectedCoins) |
| void | [setPlayer](file:///C:\Users\Fortnox\AppData\com\dod\models\Character.html#setPlayer-com.dod.models.Player-)([Player](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html) player)  The Player that this Character belongs to |
| void | [setPosition](file:///C:\Users\Fortnox\AppData\com\dod\models\Character.html#setPosition-com.dod.models.Point-)([Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) position)  The player's position in the game world |

*Constructor Detail*

* + - Character
    - public Character([Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) position,

[Player](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html) player)

*Method Detail*

* + - addCollectedCoinsPos

public void addCollectedCoinsPos([Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) newPoint)

Keeps track of which coins on the map this Character has collected. This enables us to leave the coin on the Map once it has been picked up, thereby allowing other players to pick it up, and yet not send the same coin to the same player's client again.

Parameters:

newPoint - the Point to add to the collection

* + - getCollectedCoins

public int getCollectedCoins()

* + - getCollectedCoinsPos

public java.util.List<[Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html)> getCollectedCoinsPos()

Keeps track of which coins on the map this Character has collected. This enables us to leave the coin on the Map once it has been picked up, thereby allowing other players to pick it up, and yet not send the same coin to the same player's client again.

Returns:

a list of Point objects that represent the points on the map where the Character has collected a coin

* + - getPlayer

public [Player](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html) getPlayer()

The Player that this Character belongs to

Returns:

Player

* + - getPosition

public [Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) getPosition()

The player's position in the game world

Returns:

Point

* + - setCollectedCoins

public void setCollectedCoins(int collectedCoins)

* + - setPlayer

public void setPlayer([Player](file:///C:\Users\Fortnox\AppData\com\dod\models\Player.html) player)

The Player that this Character belongs to

Parameters:

player - Player

* + - setPosition

public void setPosition([Point](file:///C:\Users\Fortnox\AppData\com\dod\models\Point.html) position)

The player's position in the game world

Parameters:

position - Point

* 1. com.dod.service.controller.ScoreController

@Path(value="score")

public class ScoreController

extends java.lang.Object

Fetches and returns the top scores

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [ScoreController](file:///C:\Users\Fortnox\com\dod\service\controller\ScoreController.html#ScoreController--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| javax.ws.rs.core.Response | [top](file:///C:\Users\Fortnox\com\dod\service\controller\ScoreController.html#top--)()  Fetches the top 10 scores across all players. |

*Constructor Detail*

* + - ScoreController

public ScoreController()

*Method Detail*

* + - top
    - @GET
    - @Produces(value="application/json")
    - @Path(value="top")

public javax.ws.rs.core.Response top()

Fetches the top 10 scores across all players.

Returns:

Response 200 OK with a JSON encoded ScoreboardModel or 500 if an error occurred

* 1. com.dod.service.controller.PlayerController

@Path(value="player")

public class PlayerController

extends java.lang.Object

Manages registering and logging in a player

Creates the session that other controllers can use to fetch user details

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [PlayerController](file:///C:\Users\Fortnox\com\dod\service\controller\PlayerController.html#PlayerController--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| javax.ws.rs.core.Response | [login](file:///C:\Users\Fortnox\com\dod\service\controller\PlayerController.html#login-java.lang.String-java.lang.String-)(java.lang.String username, java.lang.String password)  Authorises a user and starts a session with them |
| javax.ws.rs.core.Response | [register](file:///C:\Users\Fortnox\com\dod\service\controller\PlayerController.html#register-java.lang.String-java.lang.String-)(java.lang.String username, java.lang.String password)  Registers a user for the service. |

*Constructor Detail*

* + - PlayerController

public PlayerController()

*Method Detail*

* + - login
    - @POST
    - @Produces(value="text/plain")
    - @Path(value="login")
    - public javax.ws.rs.core.Response login(@NotNull @Length(min=1,max=255) @FormParam(value="username")
    - java.lang.String username,
    - @NotNull @Length(min=1,max=255) @FormParam(value="password")

java.lang.String password)

Authorises a user and starts a session with them

Parameters:

username - must be unique, not empty and less than 256 characters

password - must not be empty and less than 256 characters

Returns:

Response with blank body, 200 if successful otherwise 400 or 500

* + - register
    - @POST
    - @Produces(value="text/plain")
    - @Path(value="register")
    - public javax.ws.rs.core.Response register(@NotNull @Length(min=1,max=255) @FormParam(value="username")
    - java.lang.String username,
    - @NotNull @Length(min=1,max=255) @FormParam(value="password")

java.lang.String password)

Registers a user for the service. Username must be unique.

Parameters:

username - must be unique, not empty and less than 256 characters

password - must not be empty and less than 256 characters

Returns:

Response with blank body, 200 if successful otherwise 400 or 500

* 1. com.dod.service.controller.MatchController

@Path(value="match")

public class MatchController

extends java.lang.Object

A controller to manage Matches- joining, listing, starting a new one etc.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [MatchController](file:///C:\Users\Fortnox\com\dod\service\controller\MatchController.html#MatchController--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| javax.ws.rs.core.Response | [join](file:///C:\Users\Fortnox\com\dod\service\controller\MatchController.html#join-java.util.UUID-)(java.util.UUID matchId)  Joins the Player in an ongoing Match |
| javax.ws.rs.core.Response | [leave](file:///C:\Users\Fortnox\com\dod\service\controller\MatchController.html#leave--)()  Removes the Player from their current Match |
| javax.ws.rs.core.Response | [list](file:///C:\Users\Fortnox\com\dod\service\controller\MatchController.html#list--)()  Lists all currently lobbying matches in a JSON array |
| javax.ws.rs.core.Response | [newMatch](file:///C:\Users\Fortnox\com\dod\service\controller\MatchController.html#newMatch-int-)(int level)  Starts a new Match in a particular level and responds with that Match's status |
| javax.ws.rs.core.Response | [result](file:///C:\Users\Fortnox\com\dod\service\controller\MatchController.html#result--)()  Fetches the result of a Match from memory |
| javax.ws.rs.core.Response | [start](file:///C:\Users\Fortnox\com\dod\service\controller\MatchController.html#start--)()  Changes a Match's status to Ingame (marking the start of the Match for all players) |
| javax.ws.rs.core.Response | [status](file:///C:\Users\Fortnox\com\dod\service\controller\MatchController.html#status--)()  Responds with the status of the player's current Match. |

*Constructor Detail*

* + - MatchController

public MatchController()

*Method Detail*

* + - join
    - @POST
    - @Produces(value="application/json")
    - @Path(value="join")
    - public javax.ws.rs.core.Response join(@NotNull @FormParam(value="matchId")

java.util.UUID matchId)

Joins the Player in an ongoing Match

Parameters:

matchId - the UUID ID of the Match, must not be null

Returns:

Response 200 OK with the latest MatchStatus encoded in JSON

* + - leave
    - @POST
    - @Produces(value="text/plain")
    - @Path(value="leave")

public javax.ws.rs.core.Response leave()

Removes the Player from their current Match

Returns:

Response 200 OK with a blank body

* + - list
    - @GET
    - @Produces(value="application/json")
    - @Path(value="list")

public javax.ws.rs.core.Response list()

Lists all currently lobbying matches in a JSON array

Returns:

Response 200 OK JSON array with encoded MatchStatus for each lobbying Match

* + - newMatch
    - @POST
    - @Produces(value="application/json")
    - @Path(value="new")
    - public javax.ws.rs.core.Response newMatch(@NotNull @FormParam(value="level")

int level)

Starts a new Match in a particular level and responds with that Match's status

Parameters:

level - int the level to load for this Match, must not be null

Returns:

Response 200 OK with MatchStatus encoded in JSON or null if a Match cannot be crated

* + - result
    - @GET
    - @Produces(value="application/json")
    - @Path(value="result")

public javax.ws.rs.core.Response result()

Fetches the result of a Match from memory

Returns:

Resepons 200 OK with JSON encoded MatchResultModel

* + - start
    - @POST
    - @Produces(value="text/plain")
    - @Path(value="start")

public javax.ws.rs.core.Response start()

Changes a Match's status to Ingame (marking the start of the Match for all players)

Returns:

MatchStatus encoded in JSON

* + - status
    - @GET
    - @Produces(value="application/json")
    - @Path(value="status")

public javax.ws.rs.core.Response status()

Responds with the status of the player's current Match. If Player has no current Match returns a 500 error.

Returns:

Response 200 OK with MatchStatus encoded in JSON

* 1. com.dod.service.controller.GameController

@Path(value="game")

public class GameController

extends java.lang.Object

A controller to manage in-game game-related functionality ie getting the current state of the world or moving.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [GameController](file:///C:\Users\Fortnox\com\dod\service\controller\GameController.html#GameController--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| javax.ws.rs.core.Response | [move](file:///C:\Users\Fortnox\com\dod\service\controller\GameController.html#move-java.lang.String-)(java.lang.String direction)  An endpoint to request the Player's Character move once in a particular direction. |
| javax.ws.rs.core.Response | [status](file:///C:\Users\Fortnox\com\dod\service\controller\GameController.html#status--)()  Responds with the current gamestate from the Player's Character's perspective, i.e. |

*Constructor Detail*

* + - GameController

public GameController()

*Method Detail*

* + - move
    - @POST
    - @Produces(value="application/json")
    - @Path(value="move")
    - public javax.ws.rs.core.Response move(@NotNull @FormParam(value="key")

java.lang.String direction)

An endpoint to request the Player's Character move once in a particular direction. Responds with game status after move. If Player has no current ongoing Match returns 500 error.

Parameters:

direction - a char from {W,S,A,D} pertaining to a particular direction in the WASD layout, must not be null

Returns:

Response 200 OK with GameStateModel as a JSON object

* + - status
    - @GET
    - @Produces(value="application/json")
    - @Path(value="status")

public javax.ws.rs.core.Response status()

Responds with the current gamestate from the Player's Character's perspective, i.e. only returning visible tiles If Player has no current ongoing Match returns 500 error.

Returns:

Response 200 OK with GameStateModel as a JSON object

* 1. com.dod.service.filters.corsFilter

@Provider

public class corsFilter

extends java.lang.Object

implements javax.ws.rs.container.ContainerResponseFilte

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [corsFilter](file:///C:\Users\Fortnox\com\dod\service\filters\corsFilter.html#corsFilter--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| void | [filter](file:///C:\Users\Fortnox\com\dod\service\filters\corsFilter.html#filter-javax.ws.rs.container.ContainerRequestContext-javax.ws.rs.container.ContainerResponseContext-)(javax.ws.rs.container.ContainerRequestContext request, javax.ws.rs.container.ContainerResponseContext response)  Adds CORS headers to the Response before sending it |

*Constructor Detail*

* + - corsFilter

public corsFilter()

*Method Detail*

* + - filter
    - public void filter(javax.ws.rs.container.ContainerRequestContext request,

javax.ws.rs.container.ContainerResponseContext response)

Adds CORS headers to the Response before sending it

Specified by:

filter in interface javax.ws.rs.container.ContainerResponseFilter

Parameters:

request - ContainerRequestContext

response - ContainerResponseContext

* 1. com.dod.service.model.TileModel

public class TileModel

extends java.lang.Object

A simpler Tile model just for JSON encoding

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [TileModel](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html#TileModel--)() |
| [TileModel](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html#TileModel-int-com.dod.models.Point-)(int type, com.dod.models.Point position) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| com.dod.models.Point | [getPosition](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html#getPosition--)() |
| int | [getType](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html#getType--)() |
| void | [setPosition](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html#setPosition-com.dod.models.Point-)(com.dod.models.Point position) |
| void | [setType](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html#setType-int-)(int type) |

*Constructor Detail*

* + - TileModel

public TileModel()

* + - TileModel
    - public TileModel(int type,

com.dod.models.Point position)

*Method Detail*

* + - getPosition

public com.dod.models.Point getPosition()

* + - getType

public int getType()

* + - setPosition

public void setPosition(com.dod.models.Point position)

* + - setType

public void setType(int type)

* 1. com.dod.service.model.MatchStatus

public class MatchStatus

extends java.lang.Object

Models the current state of a lobbying match.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html#MatchStatus--)() |
| [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html#MatchStatus-com.dod.models.Match-)(com.dod.models.Match match) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| java.util.UUID | [getId](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html#getId--)() |
| java.lang.String[] | [getPlayerNames](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html#getPlayerNames--)() |
| java.lang.String | [getState](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html#getState--)() |
| void | [setId](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html#setId-java.util.UUID-)(java.util.UUID id) |
| void | [setPlayerNames](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html#setPlayerNames-java.lang.String:A-)(java.lang.String[] playerNames) |
| void | [setState](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html#setState-java.lang.String-)(java.lang.String state) |

*Constructor Detail*

* + - MatchStatus

public MatchStatus()

* + - MatchStatus

public MatchStatus(com.dod.models.Match match)

*Method Detail*

* + - getId

public java.util.UUID getId()

* + - getPlayerNames

public java.lang.String[] getPlayerNames()

* + - getState

public java.lang.String getState()

* + - setId

public void setId(java.util.UUID id)

* + - setPlayerNames

public void setPlayerNames(java.lang.String[] playerNames)

* + - setState

public void setState(java.lang.String state)

* 1. com.dod.service.model.MatchResultModel

public class MatchResultModel

extends java.lang.Object

Models the information the client needs to display the end-game screen when the game ends.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [MatchResultModel](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html#MatchResultModel--)() |
| [MatchResultModel](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html#MatchResultModel-java.lang.String-int-int-)(java.lang.String winner, int winnerCoins, int score) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| int | [getScore](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html#getScore--)() |
| java.lang.String | [getWinner](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html#getWinner--)() |
| int | [getWinnerCoins](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html#getWinnerCoins--)() |
| void | [setScore](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html#setScore-int-)(int score) |
| void | [setWinner](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html#setWinner-java.lang.String-)(java.lang.String winner) |
| void | [setWinnerCoins](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html#setWinnerCoins-int-)(int winnerCoins) |

*Constructor Detail*

* + - MatchResultModel

public MatchResultModel()

* + - MatchResultModel
    - public MatchResultModel(java.lang.String winner,
    - int winnerCoins,

int score)

*Method Detail*

* + - getScore

public int getScore()

* + - getWinner

public java.lang.String getWinner()

* + - getWinnerCoins

public int getWinnerCoins()

* + - setScore

public void setScore(int score)

* + - setWinner

public void setWinner(java.lang.String winner)

* + - setWinnerCoins

public void setWinnerCoins(int winnerCoins)

* 1. com.dod.service.model.LoginModel

public class LoginModel

extends java.lang.Object

Simple model/bean used to pass information to/from the AuthorisationService

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [LoginModel](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html#LoginModel-java.lang.String-java.lang.String-)(java.lang.String userName, java.lang.String password) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| com.dod.models.Player | [asPlayer](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html#asPlayer--)()  Convenience method to return the LoginModel's username in the Player model |
| java.lang.String | [getPassword](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html#getPassword--)() |
| java.lang.String | [getUserName](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html#getUserName--)() |
| void | [setPassword](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html#setPassword-java.lang.String-)(java.lang.String password) |
| void | [setUserName](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html#setUserName-java.lang.String-)(java.lang.String userName) |

* + - Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

*Constructor Detail*

* + - LoginModel
    - public LoginModel(java.lang.String userName,

java.lang.String password)

*Method Detail*

* + - asPlayer

public com.dod.models.Player asPlayer()

Convenience method to return the LoginModel's username in the Player model

Returns:

Player

* + - getPassword

public java.lang.String getPassword()

* + - getUserName

public java.lang.String getUserName()

* + - setPassword

public void setPassword(java.lang.String password)

* + - setUserName

public void setUserName(java.lang.String userName)

* 1. com.dod.service.model.GameStateModel

public class GameStateModel

extends java.lang.Object

Represents the current GameState. Intended to be communicated to the client via JSON encoding.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [GameStateModel](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#GameStateModel--)() |
| [GameStateModel](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#GameStateModel-com.dod.service.model.TileModel:A-com.dod.service.model.CharacterModel:A-com.dod.service.model.CharacterModel-boolean-int-)([TileModel](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html)[] tiles, [CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html)[] characters, [CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html) playerCharacter, boolean hasEnded, int minNumOfCoins) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| [CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html)[] | [getCharacters](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#getCharacters--)() |
| int | [getMinNumOfCoins](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#getMinNumOfCoins--)()  The minimum number of coins needed to win the Match |
| [CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html) | [getPlayerCharacter](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#getPlayerCharacter--)()  The Character belonging to the Player that made the request |
| [TileModel](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html)[] | [getTiles](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#getTiles--)() |
| boolean | [isHasEnded](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#isHasEnded--)()  Whether the match is ongoing- triggers the client's endgame if true |
| void | [setCharacters](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#setCharacters-com.dod.service.model.CharacterModel:A-)([CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html)[] characters) |
| void | [setHasEnded](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#setHasEnded-boolean-)(boolean hasEnded)  Whether the match is ongoing- triggers the client's endgame if true |
| void | [setMinNumOfCoins](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#setMinNumOfCoins-int-)(int minNumOfCoins)  \* The minimum number of coins needed to win the Match |
| void | [setPlayerCharacter](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#setPlayerCharacter-com.dod.service.model.CharacterModel-)([CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html) playerCharacter)  The Character belonging to the Player that made the request |
| void | [setTiles](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html#setTiles-com.dod.service.model.TileModel:A-)([TileModel](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html)[] tiles) |

*Constructor Detail*

* + - GameStateModel

public GameStateModel()

* + - GameStateModel
    - public GameStateModel([TileModel](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html)[] tiles,
    - [CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html)[] characters,
    - [CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html) playerCharacter,
    - boolean hasEnded,

int minNumOfCoins)

*Method Detail*

* + - getCharacters

public [CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html)[] getCharacters()

* + - getMinNumOfCoins

public int getMinNumOfCoins()

The minimum number of coins needed to win the Match

Returns:

int

* + - getPlayerCharacter

public [CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html) getPlayerCharacter()

The Character belonging to the Player that made the request

Returns:

Character

* + - getTiles

public [TileModel](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html)[] getTiles()

* + - isHasEnded

public boolean isHasEnded()

Whether the match is ongoing- triggers the client's endgame if true

Returns:

boolean

* + - setCharacters

public void setCharacters([CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html)[] characters)

* + - setHasEnded

public void setHasEnded(boolean hasEnded)

Whether the match is ongoing- triggers the client's endgame if true

Parameters:

hasEnded - boolean

* + - setMinNumOfCoins

public void setMinNumOfCoins(int minNumOfCoins)

\* The minimum number of coins needed to win the Match

Parameters:

minNumOfCoins - int

* + - setPlayerCharacter

public void setPlayerCharacter([CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html) playerCharacter)

The Character belonging to the Player that made the request

Parameters:

playerCharacter - Character

* + - setTiles

public void setTiles([TileModel](file:///C:\Users\Fortnox\com\dod\service\model\TileModel.html)[] tiles)

* 1. com.dod.service.model.CharacterModel

public class CharacterModel

extends java.lang.Object

A simpler model of Character for JSON encoding

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html#CharacterModel--)() |
| [CharacterModel](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html#CharacterModel-java.lang.String-int-com.dod.models.Point-)(java.lang.String playerName, int noCoins, com.dod.models.Point position) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| int | [getNoCoins](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html#getNoCoins--)() |
| java.lang.String | [getPlayerName](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html#getPlayerName--)() |
| com.dod.models.Point | [getPosition](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html#getPosition--)() |
| void | [setNoCoins](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html#setNoCoins-int-)(int noCoins) |
| void | [setPlayerName](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html#setPlayerName-java.lang.String-)(java.lang.String playerName) |
| void | [setPosition](file:///C:\Users\Fortnox\com\dod\service\model\CharacterModel.html#setPosition-com.dod.models.Point-)(com.dod.models.Point position) |

*Constructor Detail*

* + - CharacterModel

public CharacterModel()

* + - CharacterModel
    - public CharacterModel(java.lang.String playerName,
    - int noCoins,

com.dod.models.Point position)

*Method Detail*

* + - getNoCoins

public int getNoCoins()

* + - getPlayerName

public java.lang.String getPlayerName()

* + - getPosition

public com.dod.models.Point getPosition()

* + - setNoCoins

public void setNoCoins(int noCoins)

* + - setPlayerName

public void setPlayerName(java.lang.String playerName)

* + - setPosition

public void setPosition(com.dod.models.Point position)

* 1. com.dod.service.service.VisibilityService

public class VisibilityService

extends java.lang.Object

implements [IVisibilityService](file:///C:\Users\Fortnox\com\dod\service\service\IVisibilityService.html)

Calculates the visible tiles from the perspective of a particular Character

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [VisibilityService](file:///C:\Users\Fortnox\com\dod\service\service\VisibilityService.html#VisibilityService--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| com.dod.models.Map | [createVisibleMap](file:///C:\Users\Fortnox\com\dod\service\service\VisibilityService.html#createVisibleMap-com.dod.models.Map-com.dod.models.Character-)(com.dod.models.Map dungeonMap, com.dod.models.Character pchar)  Generates a copy of a Map with the correct isVisible flags set for the perspective of a particular Character |

*Constructor Detail*

* + - VisibilityService

public VisibilityService()

*Method Detail*

* + - createVisibleMap
    - public com.dod.models.Map createVisibleMap(com.dod.models.Map dungeonMap,

com.dod.models.Character pchar)

Generates a copy of a Map with the correct isVisible flags set for the perspective of a particular Character

Specified by:

[createVisibleMap](file:///C:\Users\Fortnox\com\dod\service\service\IVisibilityService.html#createVisibleMap-com.dod.models.Map-com.dod.models.Character-) in interface [IVisibilityService](file:///C:\Users\Fortnox\com\dod\service\service\IVisibilityService.html)

Parameters:

deungeonMap - the Map pchar resides in

pchar - the Character the perspective of which we're generating visibility with

Returns:

a copy of dungeonMap with correct isVisible flags set for the perspective of pchar

* 1. com.dod.service.service.StateService

public class StateService

extends java.lang.Object

implements [IStateService](file:///C:\Users\Fortnox\com\dod\service\service\IStateService.html)

Generates a representation of the current game state form the perspective of a particular character

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [StateService](file:///C:\Users\Fortnox\com\dod\service\service\StateService.html#StateService-com.dod.service.service.IVisibilityService-com.dod.game.IMatchList-)([IVisibilityService](file:///C:\Users\Fortnox\com\dod\service\service\IVisibilityService.html) visibilityService, com.dod.game.IMatchList matchList) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| [GameStateModel](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html) | [GetState](file:///C:\Users\Fortnox\com\dod\service\service\StateService.html#GetState-com.dod.models.Player-)(com.dod.models.Player player)  Generates and returns a representation of the current game state form the perspective of a particular character |

* + - Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

*Constructor Detail*

* + - StateService
    - public StateService([IVisibilityService](file:///C:\Users\Fortnox\com\dod\service\service\IVisibilityService.html) visibilityService,

com.dod.game.IMatchList matchList)

*Method Detail*

* + - GetState

public [GameStateModel](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html) GetState(com.dod.models.Player player)

Generates and returns a representation of the current game state form the perspective of a particular character

Specified by:

[GetState](file:///C:\Users\Fortnox\com\dod\service\service\IStateService.html#GetState-com.dod.models.Player-) in interface [IStateService](file:///C:\Users\Fortnox\com\dod\service\service\IStateService.html)

Parameters:

player - Player the Player a GameStateModel will be generated for

Returns:

GameStateModel a model of the current game state

* 1. com.dod.service.service.ParseService

public class ParseService

extends java.lang.Object

implements [IParseService](file:///C:\Users\Fortnox\com\dod\service\service\IParseService.html)

Implementation of IParseService.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [ParseService](file:///C:\Users\Fortnox\com\dod\service\service\ParseService.html#ParseService--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| com.dod.models.Map | [parseMap](file:///C:\Users\Fortnox\com\dod\service\service\ParseService.html#parseMap-org.json.simple.JSONObject-)(org.json.simple.JSONObject input)  Parses a Map object from it's JSON encoding |

*Constructor Detail*

* + - ParseService

public ParseService()

*Method Detail*

* + - parseMap
    - public com.dod.models.Map parseMap(org.json.simple.JSONObject input)

throws java.lang.NullPointerException

Parses a Map object from it's JSON encoding

Specified by:

[parseMap](file:///C:\Users\Fortnox\com\dod\service\service\IParseService.html#parseMap-org.json.simple.JSONObject-) in interface [IParseService](file:///C:\Users\Fortnox\com\dod\service\service\IParseService.html)

Parameters:

input - JSONObject a JSON encoding of the Map

Returns:

Map an initialised Map parsed from JSON

Throws:

java.lang.NullPointerException - may be thrown by SimpleJson while parsing

* 1. com.dod.service.service.MovementService

public class MovementService

extends java.lang.Object

implements [IMovementService](file:///C:\Users\Fortnox\com\dod\service\service\IMovementService.html)

Implementation of IMovementService

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [MovementService](file:///C:\Users\Fortnox\com\dod\service\service\MovementService.html#MovementService--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| com.dod.models.Point | [Move](file:///C:\Users\Fortnox\com\dod\service\service\MovementService.html#Move-java.lang.String-com.dod.models.Player-)(java.lang.String direction, com.dod.models.Player player)  Moves the Player in a particular direction. |

* + - Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

*Constructor Detail*

* + - MovementService

public MovementService()

*Method Detail*

* + - Move
    - public com.dod.models.Point Move(java.lang.String direction,
    - com.dod.models.Player player)

throws java.sql.SQLException

Moves the Player in a particular direction. Will increment player's gold if interacting with gold coins, can trigger end of the Match when player interacts with Exit.

Specified by:

[Move](file:///C:\Users\Fortnox\com\dod\service\service\IMovementService.html#Move-java.lang.String-com.dod.models.Player-) in interface [IMovementService](file:///C:\Users\Fortnox\com\dod\service\service\IMovementService.html)

Parameters:

direction - String a char from {W,S,A,D} pertaining to a particular direction in the WASD layout

player - Player whom's Character will be moved

Returns:

Point that the Player has moved to

Throws:

java.sql.SQLException - if the database cannot be reached or statement fails while inserting new score

* 1. com.dod.service.service.MatchService
* All Implemented Interfaces:

[IMatchService](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html)

public class MatchService

extends java.lang.Object

implements [IMatchService](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html)

Manages joining/starting/ending matches.

Makes heavy use of MatchList to store matches in memory.

Uses PlayerRepository to fetch Player data.

Uses IOService and ParseService to load levels when starting a new Match.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [MatchService](file:///C:\Users\Fortnox\com\dod\service\service\MatchService.html#MatchService-com.dod.service.service.IIOService-com.dod.service.service.IParseService-com.dod.db.repositories.IPlayerRepository-com.dod.game.IMatchList-)([IIOService](file:///C:\Users\Fortnox\com\dod\service\service\IIOService.html) ioService, [IParseService](file:///C:\Users\Fortnox\com\dod\service\service\IParseService.html) parseService, com.dod.db.repositories.IPlayerRepository playerRepository, com.dod.game.IMatchList matchList) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html) | [createMatch](file:///C:\Users\Fortnox\com\dod\service\service\MatchService.html#createMatch-java.lang.String-int-)(java.lang.String userName, int level)  Creates a new Match |
| void | [endMatch](file:///C:\Users\Fortnox\com\dod\service\service\MatchService.html#endMatch-com.dod.models.Player-)(com.dod.models.Player player)  Changes a Match's state to Over |
| [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html)[] | [getLobbyingMatches](file:///C:\Users\Fortnox\com\dod\service\service\MatchService.html#getLobbyingMatches--)()  Get all Matches currently in the Lobbying state |
| [MatchResultModel](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html) | [getMatchResult](file:///C:\Users\Fortnox\com\dod\service\service\MatchService.html#getMatchResult-com.dod.models.Player-)(com.dod.models.Player player)  Gets the MatchResultModel for a finished Match todo why not remove the Player from the Match at this point rather than send another request? |
| [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html) | [getStatus](file:///C:\Users\Fortnox\com\dod\service\service\MatchService.html#getStatus-com.dod.models.Player-)(com.dod.models.Player player)  Returns the MatchStatus for a particular Player's Match |
| void | [joinMatch](file:///C:\Users\Fortnox\com\dod\service\service\MatchService.html#joinMatch-com.dod.models.Player-java.util.UUID-)(com.dod.models.Player player, java.util.UUID matchId)  Adds the Player to a particular Match |
| void | [leaveMatch](file:///C:\Users\Fortnox\com\dod\service\service\MatchService.html#leaveMatch-com.dod.models.Player-)(com.dod.models.Player player)  Removes a Player from their current ongoing Match |
| void | [startMatch](file:///C:\Users\Fortnox\com\dod\service\service\MatchService.html#startMatch-com.dod.models.Player-)(com.dod.models.Player player)  Changes a Match's state to InGame |

*Constructor Detail*

* + - MatchService
    - public MatchService([IIOService](file:///C:\Users\Fortnox\com\dod\service\service\IIOService.html) ioService,
    - [IParseService](file:///C:\Users\Fortnox\com\dod\service\service\IParseService.html) parseService,
    - com.dod.db.repositories.IPlayerRepository playerRepository,

com.dod.game.IMatchList matchList)

*Method Detail*

* + - createMatch
    - public [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html) createMatch(java.lang.String userName,

int level)

Creates a new Match

Specified by:

[createMatch](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#createMatch-java.lang.String-int-) in interface [IMatchService](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html)

Parameters:

userName - String username of the Player who is starting the Match

level - int the number of the level to load for this Match

Returns:

MatchStatus of the newly created Match

* + - endMatch

public void endMatch(com.dod.models.Player player)

Changes a Match's state to Over

Specified by:

[endMatch](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#endMatch-com.dod.models.Player-) in interface [IMatchService](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html)

Parameters:

player - Player whose ongoing Match will be modified

* + - getLobbyingMatches

public [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html)[] getLobbyingMatches()

Get all Matches currently in the Lobbying state

Specified by:

[getLobbyingMatches](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#getLobbyingMatches--) in interface [IMatchService](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html)

Returns:

MatchStatus[] array of all Matches in the Lobbying state

* + - getMatchResult

public [MatchResultModel](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html) getMatchResult(com.dod.models.Player player)

Gets the MatchResultModel for a finished Match todo why not remove the Player from the Match at this point rather than send another request?

Specified by:

[getMatchResult](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#getMatchResult-com.dod.models.Player-) in interface [IMatchService](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html)

Parameters:

player - Player the Player that has a finished Match

Returns:

MatchResultModel pertaining to the player's Match

* + - getStatus

public [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html) getStatus(com.dod.models.Player player)

Returns the MatchStatus for a particular Player's Match

Specified by:

[getStatus](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#getStatus-com.dod.models.Player-) in interface [IMatchService](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html)

Parameters:

player - Player whose ongoing Match will be fetched

Returns:

* + - joinMatch
    - public void joinMatch(com.dod.models.Player player,
    - java.util.UUID matchId)

throws java.sql.SQLException

Adds the Player to a particular Match

Specified by:

[joinMatch](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#joinMatch-com.dod.models.Player-java.util.UUID-) in interface [IMatchService](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html)

Parameters:

player - Player whom will be added

matchID - UUID of the Match that player will be addd to

Throws:

java.sql.SQLException - thrown if Player doesn't exist or a SQL connectivity issue occurs

* + - leaveMatch

public void leaveMatch(com.dod.models.Player player)

Removes a Player from their current ongoing Match

Specified by:

[leaveMatch](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#leaveMatch-com.dod.models.Player-) in interface [IMatchService](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html)

Parameters:

player - Player the Player whom will be removed from their ongoing Match

* + - startMatch

public void startMatch(com.dod.models.Player player)

Changes a Match's state to InGame

Specified by:

[startMatch](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#startMatch-com.dod.models.Player-) in interface [IMatchService](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html)

Parameters:

player - Player whose ongoing Match will be modified

* 1. com.dod.service.service.IVisibilityService
* All Known Implementing Classes:

[VisibilityService](file:///C:\Users\Fortnox\com\dod\service\service\VisibilityService.html)

public interface IVisibilityService

Calculates the visible tiles from the perspective of a particular Character

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| com.dod.models.Map | [createVisibleMap](file:///C:\Users\Fortnox\com\dod\service\service\IVisibilityService.html#createVisibleMap-com.dod.models.Map-com.dod.models.Character-)(com.dod.models.Map deungeonMap, com.dod.models.Character pchar)  Generates a copy of a Map with the correct isVisible flags set for the perspective of a particular Character |

*Method Detail*

* + - createVisibleMap
    - com.dod.models.Map createVisibleMap(com.dod.models.Map deungeonMap,

com.dod.models.Character pchar)

Generates a copy of a Map with the correct isVisible flags set for the perspective of a particular Character

Parameters:

deungeonMap - the Map pchar resides in

pchar - the Character the perspective of which we're generating visibility with

Returns:

a copy of dungeonMap with correct isVisible flags set for the perspective of pchar

* 1. com.dod.service.service.IStateService
* All Known Implementing Classes:

[StateService](file:///C:\Users\Fortnox\com\dod\service\service\StateService.html)

public interface IStateService

Generates a representation of the current game state form the perspective of a particular character

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| [GameStateModel](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html) | [GetState](file:///C:\Users\Fortnox\com\dod\service\service\IStateService.html#GetState-com.dod.models.Player-)(com.dod.models.Player player)  Generates and returns a representation of the current game state form the perspective of a particular character |

*Method Detail*

* + - GetState

[GameStateModel](file:///C:\Users\Fortnox\com\dod\service\model\GameStateModel.html) GetState(com.dod.models.Player player)

Generates and returns a representation of the current game state form the perspective of a particular character

Parameters:

player - Player the Player a GameStateModel will be generated for

Returns:

GameStateModel a model of the current game state

* 1. com.dod.service.service.IParseService
* All Known Implementing Classes:

[ParseService](file:///C:\Users\Fortnox\com\dod\service\service\ParseService.html)

public interface IParseService

Parses JSON objects- namely the Map

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| com.dod.models.Map | [parseMap](file:///C:\Users\Fortnox\com\dod\service\service\IParseService.html#parseMap-org.json.simple.JSONObject-)(org.json.simple.JSONObject input)  Parses a Map object from it's JSON encoding |

*Method Detail*

* + - parseMap
    - com.dod.models.Map parseMap(org.json.simple.JSONObject input)

throws java.lang.NullPointerException

Parses a Map object from it's JSON encoding

Parameters:

input - JSONObject a JSON encoding of the Map

Returns:

Map an initialised Map parsed from JSON

Throws:

java.lang.NullPointerException - may be thrown by SimpleJson while parsing

* 1. com.dod.service.service.IOService
* All Implemented Interfaces:

[IIOService](file:///C:\Users\Fortnox\com\dod\service\service\IIOService.html)

public class IOService

extends java.lang.Object

implements [IIOService](file:///C:\Users\Fortnox\com\dod\service\service\IIOService.html)

Handles IO within the Service

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [IOService](file:///C:\Users\Fortnox\com\dod\service\service\IOService.html#IOService--)() |
| [IOService](file:///C:\Users\Fortnox\com\dod\service\service\IOService.html#IOService-java.lang.String-)(java.lang.String pathToAssets) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| org.json.simple.JSONObject | [getJsonObject](file:///C:\Users\Fortnox\com\dod\service\service\IOService.html#getJsonObject-java.lang.String-)(java.lang.String path)  Fetches an asset as parsed JSON |
| java.lang.String | [getString](file:///C:\Users\Fortnox\com\dod\service\service\IOService.html#getString-java.lang.String-)(java.lang.String path)  Fetches an asset as a String |

*Constructor Detail*

* + - IOService

public IOService()

* + - IOService

public IOService(java.lang.String pathToAssets)

*Method Detail*

* + - getJsonObject
    - public org.json.simple.JSONObject getJsonObject(java.lang.String path)
    - throws java.io.IOException,

org.json.simple.parser.ParseException

Fetches an asset as parsed JSON

Specified by:

[getJsonObject](file:///C:\Users\Fortnox\com\dod\service\service\IIOService.html#getJsonObject-java.lang.String-) in interface [IIOService](file:///C:\Users\Fortnox\com\dod\service\service\IIOService.html)

Parameters:

path - String the path to the asset we are to fetch

Returns:

JSONObject the parsed content of the asset

Throws:

java.io.IOException - if the file is missing

org.json.simple.parser.ParseException - if the file isn't encoded in valid JSON

* + - getString
    - public java.lang.String getString(java.lang.String path)

throws java.io.IOException

Fetches an asset as a String

Specified by:

[getString](file:///C:\Users\Fortnox\com\dod\service\service\IIOService.html#getString-java.lang.String-) in interface [IIOService](file:///C:\Users\Fortnox\com\dod\service\service\IIOService.html)

Parameters:

path - String the path to the asset we are to fetch

Returns:

String the contents of the asset

Throws:

java.io.IOException - if the file is missing

* 1. com.dod.service.service.IMovementService
* All Known Implementing Classes:

[MovementService](file:///C:\Users\Fortnox\com\dod\service\service\MovementService.html)

public interface IMovementService

Interface for MovementService. Handles game logic to move a character from one point to another.

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| com.dod.models.Point | [Move](file:///C:\Users\Fortnox\com\dod\service\service\IMovementService.html#Move-java.lang.String-com.dod.models.Player-)(java.lang.String direction, com.dod.models.Player player)  Moves the Player in a particular direction. |

*Method Detail*

* + - Move
    - com.dod.models.Point Move(java.lang.String direction,
    - com.dod.models.Player player)

throws java.sql.SQLException

Moves the Player in a particular direction. Will increment player's gold if interacting with gold coins, can trigger end of the Match when player interacts with Exit.

Parameters:

direction - String a char from {W,S,A,D} pertaining to a particular direction in the WASD layout

player - Player whom's Character will be moved

Returns:

Point that the Player has moved to

Throws:

java.sql.SQLException - if the database cannot be reached or statement fails while inserting new score

* 1. com.dod.service.service.IMatchService
* All Known Implementing Classes:

[MatchService](file:///C:\Users\Fortnox\com\dod\service\service\MatchService.html)

public interface IMatchService

Manages joining/starting/ending matches.

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html) | [createMatch](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#createMatch-java.lang.String-int-)(java.lang.String userName, int level)  Creates a new Match |
| void | [endMatch](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#endMatch-com.dod.models.Player-)(com.dod.models.Player player)  Changes a Match's state to Over |
| [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html)[] | [getLobbyingMatches](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#getLobbyingMatches--)()  Get all Matches currently in the Lobbying state |
| [MatchResultModel](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html) | [getMatchResult](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#getMatchResult-com.dod.models.Player-)(com.dod.models.Player player)  Gets the MatchResultModel for a finished Match todo why not remove the Player from the Match at this point rather than send another request? |
| [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html) | [getStatus](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#getStatus-com.dod.models.Player-)(com.dod.models.Player player)  Returns the MatchStatus for a particular Player's Match |
| void | [joinMatch](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#joinMatch-com.dod.models.Player-java.util.UUID-)(com.dod.models.Player player, java.util.UUID matchID)  Adds the Player to a particular Match |
| void | [leaveMatch](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#leaveMatch-com.dod.models.Player-)(com.dod.models.Player player)  Removes a Player from their current ongoing Match |
| void | [startMatch](file:///C:\Users\Fortnox\com\dod\service\service\IMatchService.html#startMatch-com.dod.models.Player-)(com.dod.models.Player player)  Changes a Match's state to InGame |

*Method Detail*

* + - createMatch
    - [MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html) createMatch(java.lang.String userName,

int level)

Creates a new Match

Parameters:

userName - String username of the Player who is starting the Match

level - int the number of the level to load for this Match

Returns:

MatchStatus of the newly created Match

* + - endMatch

void endMatch(com.dod.models.Player player)

Changes a Match's state to Over

Parameters:

player - Player whose ongoing Match will be modified

* + - getLobbyingMatches

[MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html)[] getLobbyingMatches()

Get all Matches currently in the Lobbying state

Returns:

MatchStatus[] array of all Matches in the Lobbying state

* + - getMatchResult

[MatchResultModel](file:///C:\Users\Fortnox\com\dod\service\model\MatchResultModel.html) getMatchResult(com.dod.models.Player player)

Gets the MatchResultModel for a finished Match todo why not remove the Player from the Match at this point rather than send another request?

Parameters:

player - Player the Player that has a finished Match

Returns:

MatchResultModel pertaining to the player's Match

* + - getStatus

[MatchStatus](file:///C:\Users\Fortnox\com\dod\service\model\MatchStatus.html) getStatus(com.dod.models.Player player)

Returns the MatchStatus for a particular Player's Match

Parameters:

player - Player whose ongoing Match will be fetched

Returns:

* + - joinMatch
    - void joinMatch(com.dod.models.Player player,
    - java.util.UUID matchID)

throws java.sql.SQLException

Adds the Player to a particular Match

Parameters:

player - Player whom will be added

matchID - UUID of the Match that player will be addd to

Throws:

java.sql.SQLException - thrown if Player doesn't exist or a SQL connectivity issue occurs

* + - leaveMatch

void leaveMatch(com.dod.models.Player player)

Removes a Player from their current ongoing Match

Parameters:

player - Player the Player whom will be removed from their ongoing Match

* + - startMatch

void startMatch(com.dod.models.Player player)

Changes a Match's state to InGame

Parameters:

player - Player whose ongoing Match will be modified

* 1. com.dod.service.service.IIOService
* All Known Implementing Classes:

[IOService](file:///C:\Users\Fortnox\com\dod\service\service\IOService.html)

public interface IIOService

Handles IO within the Service

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| org.json.simple.JSONObject | [getJsonObject](file:///C:\Users\Fortnox\com\dod\service\service\IIOService.html#getJsonObject-java.lang.String-)(java.lang.String path)  Fetches an asset as parsed JSON |
| java.lang.String | [getString](file:///C:\Users\Fortnox\com\dod\service\service\IIOService.html#getString-java.lang.String-)(java.lang.String path)  Fetches an asset as a String |

*Method Detail*

* + - getJsonObject
    - org.json.simple.JSONObject getJsonObject(java.lang.String path)
    - throws java.io.IOException,

org.json.simple.parser.ParseException

Fetches an asset as parsed JSON

Parameters:

path - String the path to the asset we are to fetch

Returns:

JSONObject the parsed content of the asset

Throws:

java.io.IOException - if the file is missing

org.json.simple.parser.ParseException - if the file isn't encoded in valid JSON

* + - getString
    - java.lang.String getString(java.lang.String path)

throws java.io.IOException

Fetches an asset as a String

Parameters:

path - String the path to the asset we are to fetch

Returns:

String the contents of the asset

Throws:

java.io.IOException - if the file is missing

* 1. com.dod.service.service.IAuthenticationService
* All Known Implementing Classes:

[AuthenticationService](file:///C:\Users\Fortnox\com\dod\service\service\AuthenticationService.html)

public interface IAuthenticationService

Handles authenticating a user against their user/pass combo

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| boolean | [Login](file:///C:\Users\Fortnox\com\dod\service\service\IAuthenticationService.html#Login-com.dod.service.model.LoginModel-)([LoginModel](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html) model)  Registers a new user |
| boolean | [Register](file:///C:\Users\Fortnox\com\dod\service\service\IAuthenticationService.html#Register-com.dod.service.model.LoginModel-)([LoginModel](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html) model)  Registers a new user |

*Method Detail*

* + - Login

boolean Login([LoginModel](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html) model)

Registers a new user

Parameters:

model - LoginModel containing the user/pass to be authorised

Returns:

boolean true if the user is authorised, otherwise false

* + - Register

boolean Register([LoginModel](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html) model)

Registers a new user

Parameters:

model - LoginModel containing the user/pass to be registered

Returns:

boolean true if successful otherwise false

* 1. com.dod.service.service.AuthenticationService
* All Implemented Interfaces:

[IAuthenticationService](file:///C:\Users\Fortnox\com\dod\service\service\IAuthenticationService.html)

public class AuthenticationService

extends java.lang.Object

implements [IAuthenticationService](file:///C:\Users\Fortnox\com\dod\service\service\IAuthenticationService.html)

Handles authenticating a user against their user/pass combo

Uses a salt, generated using a secure RNG

Uses PlayerRepository to fetch Player database details

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [AuthenticationService](file:///C:\Users\Fortnox\com\dod\service\service\AuthenticationService.html#AuthenticationService-com.dod.db.repositories.IPlayerRepository-)(com.dod.db.repositories.IPlayerRepository repository) |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| boolean | [Login](file:///C:\Users\Fortnox\com\dod\service\service\AuthenticationService.html#Login-com.dod.service.model.LoginModel-)([LoginModel](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html) model)  Registers a new user |
| boolean | [Register](file:///C:\Users\Fortnox\com\dod\service\service\AuthenticationService.html#Register-com.dod.service.model.LoginModel-)([LoginModel](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html) model)  Registers a new user |

*Constructor Detail*

* + - AuthenticationService

public AuthenticationService(com.dod.db.repositories.IPlayerRepository repository)

*Method Detail*

* + - Login

public boolean Login([LoginModel](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html) model)

Registers a new user

Specified by:

[Login](file:///C:\Users\Fortnox\com\dod\service\service\IAuthenticationService.html#Login-com.dod.service.model.LoginModel-) in interface [IAuthenticationService](file:///C:\Users\Fortnox\com\dod\service\service\IAuthenticationService.html)

Parameters:

model - LoginModel containing the user/pass to be authorised

Returns:

boolean true if the user is authorised, otherwise false

* + - Register

public boolean Register([LoginModel](file:///C:\Users\Fortnox\com\dod\service\model\LoginModel.html) model)

Registers a new user

Specified by:

[Register](file:///C:\Users\Fortnox\com\dod\service\service\IAuthenticationService.html#Register-com.dod.service.model.LoginModel-) in interface [IAuthenticationService](file:///C:\Users\Fortnox\com\dod\service\service\IAuthenticationService.html)

Parameters:

model - LoginModel containing the user/pass to be registered

Returns:

boolean true if successful otherwise false

* 1. com.dod.service.Main

public class Main

extends java.lang.Object

Main class.

*Field Summary*

|  |  |
| --- | --- |
| Fields | |
| Modifier and Type | Field and Description |
| static java.lang.String | [BASE\_URI](file:///C:\Users\Fortnox\AppData\com\dod\service\Main.html#BASE_URI) |

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Main](file:///C:\Users\Fortnox\AppData\com\dod\service\Main.html#Main--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| static void | [main](file:///C:\Users\Fortnox\AppData\com\dod\service\Main.html#main-java.lang.String:A-)(java.lang.String[] args)  Main method. |
| static org.glassfish.grizzly.http.server.HttpServer | [startServer](file:///C:\Users\Fortnox\AppData\com\dod\service\Main.html#startServer--)()  Starts Grizzly HTTP server exposing JAX-RS resources defined in this application. |

*Field Detail*

* + - BASE\_URI

public static final java.lang.String BASE\_URI

See Also:

[Constant Field Values](file:///C:\Users\Fortnox\AppData\constant-values.html#com.dod.service.Main.BASE_URI)

*Constructor Detail*

* + - Main

public Main()

*Method Detail*

* + - main
    - public static void main(java.lang.String[] args)

throws java.io.IOException

Main method.

Parameters:

args -

Throws:

java.io.IOException

* + - startServer

public static org.glassfish.grizzly.http.server.HttpServer startServer()

Starts Grizzly HTTP server exposing JAX-RS resources defined in this application.

Returns:

Grizzly HTTP server.

* 1. com.dod.bot.communicators.CommunicatorBase
* Direct Known Subclasses:

[MatchCommunicator](file:///C:\Users\Fortnox\com\dod\bot\communicators\MatchCommunicator.html), [MoveCommunicator](file:///C:\Users\Fortnox\com\dod\bot\communicators\MoveCommunicator.html), [stateCommunicator](file:///C:\Users\Fortnox\com\dod\bot\communicators\stateCommunicator.html)

public class CommunicatorBase

extends java.lang.Object

A base class that handles generic communication to/from the server.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [CommunicatorBase](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#CommunicatorBase--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| protected javax.ws.rs.core.Response | [get](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#get-javax.ws.rs.client.Invocation.Builder-)(javax.ws.rs.client.Invocation.Builder request)  Invokes the specified Request as a GET request |
| protected javax.ws.rs.core.Response | [get](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#get-java.lang.String-)(java.lang.String path)  Sends a GET request to a particular path on the web service |
| protected static javax.ws.rs.client.WebTarget | [getTarget](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#getTarget--)() |
| protected javax.ws.rs.core.Response | [post](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#post-javax.ws.rs.client.Invocation.Builder-javax.ws.rs.core.MultivaluedMap-)(javax.ws.rs.client.Invocation.Builder request, javax.ws.rs.core.MultivaluedMap<java.lang.String,java.lang.String> params)  Invokes the request as a POST request with the specified parameters as form parameters. |
| protected javax.ws.rs.core.Response | [post](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#post-java.lang.String-javax.ws.rs.core.MultivaluedMap-)(java.lang.String path, javax.ws.rs.core.MultivaluedMap<java.lang.String,java.lang.String> params)  Posts a web request to the specified path with the specified parameters as form parameters. |
| protected javax.ws.rs.client.Invocation.Builder | [request](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#request-java.lang.String-)(java.lang.String path)  Generates a request to the specified path |

* + - Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

* + *Constructor Detail*
    - CommunicatorBase

public CommunicatorBase()

* + *Method Detail*
    - get

protected javax.ws.rs.core.Response get(javax.ws.rs.client.Invocation.Builder request)

Invokes the specified Request as a GET request

Parameters:

request - Invocation.Builder a Builder that generates an Invocation of a particular web resource.

Returns:

* + - get

protected javax.ws.rs.core.Response get(java.lang.String path)

Sends a GET request to a particular path on the web service

Parameters:

path - String the path to send the GET request to

Returns:

Response the response from the server

* + - getTarget

protected static javax.ws.rs.client.WebTarget getTarget()

* + - post
    - protected javax.ws.rs.core.Response post(javax.ws.rs.client.Invocation.Builder request,

javax.ws.rs.core.MultivaluedMap<java.lang.String,java.lang.String> params)

Invokes the request as a POST request with the specified parameters as form parameters.

Parameters:

request - Invocation.Builder a Builder that generates an Invocation of a particular web resource.

params - MultiValuedHashMap the parameters to send with the POST request

Returns:

Response the response from the service

* + - post
    - protected javax.ws.rs.core.Response post(java.lang.String path,

javax.ws.rs.core.MultivaluedMap<java.lang.String,java.lang.String> params)

Posts a web request to the specified path with the specified parameters as form parameters.

Parameters:

path - String the path to send the POST request to

params - MultiValuedHashMap the parameters to send with the POST request

Returns:

Response the response from the service

* + - request

protected javax.ws.rs.client.Invocation.Builder request(java.lang.String path)

Generates a request to the specified path

Parameters:

path - String the path to request

Returns:

Invocation.Builder a Builder that generates an Invocation of the specified web resource.

* 1. com.dod.bot.communicators.stateCommunicator

public class stateCommunicator

extends [CommunicatorBase](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html)

Communicates status requests to the server

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [stateCommunicator](file:///C:\Users\Fortnox\com\dod\bot\communicators\stateCommunicator.html#stateCommunicator--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| com.dod.service.model.GameStateModel | [getState](file:///C:\Users\Fortnox\com\dod\bot\communicators\stateCommunicator.html#getState--)()  Gets the current state from the web service. |

* + - Methods inherited from class com.dod.bot.communicators.[CommunicatorBase](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html)

[get](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#get-javax.ws.rs.client.Invocation.Builder-), [get](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#get-java.lang.String-), [getTarget](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#getTarget--), [post](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#post-javax.ws.rs.client.Invocation.Builder-javax.ws.rs.core.MultivaluedMap-), [post](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#post-java.lang.String-javax.ws.rs.core.MultivaluedMap-), [request](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#request-java.lang.String-)

*Constructor Detail*

* + - stateCommunicator

public stateCommunicator()

*Method Detail*

* + - getState

public com.dod.service.model.GameStateModel getState()

Gets the current state from the web service.

Returns:

GameStateModel a model representing the game's current state.

* 1. com.dod.bot.communicators.MoveCommunicator

public class MoveCommunicator

extends [CommunicatorBase](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html)

Communicates move requests to the server

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [MoveCommunicator](file:///C:\Users\Fortnox\com\dod\bot\communicators\MoveCommunicator.html#MoveCommunicator--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| void | [moveDirection](file:///C:\Users\Fortnox\com\dod\bot\communicators\MoveCommunicator.html#moveDirection-java.lang.String-)(java.lang.String direction)  Sends a request to the web service to move in a particular direction |

* + - Methods inherited from class com.dod.bot.communicators.[CommunicatorBase](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html)

[get](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#get-javax.ws.rs.client.Invocation.Builder-), [get](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#get-java.lang.String-), [getTarget](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#getTarget--), [post](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#post-javax.ws.rs.client.Invocation.Builder-javax.ws.rs.core.MultivaluedMap-), [post](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#post-java.lang.String-javax.ws.rs.core.MultivaluedMap-), [request](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#request-java.lang.String-)

*Constructor Detail*

* + - MoveCommunicator

public MoveCommunicator()

*Method Detail*

* + - moveDirection

public void moveDirection(java.lang.String direction)

Sends a request to the web service to move in a particular direction

Parameters:

direction - String the direction to move in, a char from the set {W,A,S,D} corresponding to WASD directions.

* 1. com.dod.bot.communicators.MatchCommunicator

public class MatchCommunicator

extends [CommunicatorBase](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html)

Handles match requests to the server

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [MatchCommunicator](file:///C:\Users\Fortnox\com\dod\bot\communicators\MatchCommunicator.html#MatchCommunicator--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| void | [joinMatch](file:///C:\Users\Fortnox\com\dod\bot\communicators\MatchCommunicator.html#joinMatch-java.util.UUID-)(java.util.UUID matchId)  Sends a request to the web service to join the specified Match. |

* + - Methods inherited from class com.dod.bot.communicators.[CommunicatorBase](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html)

[get](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#get-javax.ws.rs.client.Invocation.Builder-), [get](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#get-java.lang.String-), [getTarget](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#getTarget--), [post](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#post-javax.ws.rs.client.Invocation.Builder-javax.ws.rs.core.MultivaluedMap-), [post](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#post-java.lang.String-javax.ws.rs.core.MultivaluedMap-), [request](file:///C:\Users\Fortnox\com\dod\bot\communicators\CommunicatorBase.html#request-java.lang.String-)

* + - Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

*Constructor Detail*

* + - MatchCommunicator

public MatchCommunicator()

*Method Detail*

* + - joinMatch

public void joinMatch(java.util.UUID matchId)

Sends a request to the web service to join the specified Match.

Parameters:

matchId - UUID the ID of the Match to join

* 1. com.dod.bot.Map

public class Map

extends java.lang.Object

Map for the bot modeled on the responses from the server.

Should work in theory but not tested as we ran out of time.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Map](file:///C:\Users\Fortnox\AppData\com\dod\bot\Map.html#Map--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| void | [addTile](file:///C:\Users\Fortnox\AppData\com\dod\bot\Map.html#addTile-com.dod.service.model.TileModel:A-)(com.dod.service.model.TileModel[] tiles)  Add a tile to the map. |

*Constructor Detail*

* + - Map

public Map()

*Method Detail*

* + - addTile

public void addTile(com.dod.service.model.TileModel[] tiles)

Add a tile to the map. Expands the map to the correct size of necessary.

Parameters:

tiles - TileModel[] a collection of Tiles to add to the Map.

* 1. com.dod.bot.Main

public class Main

extends java.lang.Object

Gets command parameters and intitialises bot

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Main](file:///C:\Users\Fortnox\AppData\com\dod\bot\Main.html#Main--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| static void | [main](file:///C:\Users\Fortnox\AppData\com\dod\bot\Main.html#main-java.lang.String:A-)(java.lang.String[] args)  Parses the input and starts the Bot |

*Constructor Detail*

* + - Main

public Main()

*Method Detail*

* + - main

public static void main(java.lang.String[] args)

Parses the input and starts the Bot

Parameters:

args - expects 1 argument of ID for match to join

* 1. com.dod.bot.Bot

public class Bot

extends java.lang.Object

The main bot object. Makes basic decisions and uses the Communicators to enact these decisions.

Has no real intelligence at the moment. In the future we could make it much more intelligent using the Map class

to store beliefs about the world and use path-finding to hunt out gold to get the most score.

*Constructor Summary*

|  |
| --- |
| Constructors |
| Constructor and Description |
| [Bot](file:///C:\Users\Fortnox\AppData\com\dod\bot\Bot.html#Bot--)() |

*Method Summary*

|  |  |
| --- | --- |
| All Methods | |
| Modifier and Type | Method and Description |
| void | [play](file:///C:\Users\Fortnox\AppData\com\dod\bot\Bot.html#play-java.util.UUID-)(java.util.UUID matchId)  Joins a match and then randomly picks a direction to move in every 5th of a second. |

*Constructor Detail*

* + - Bot

public Bot()

*Method Detail*

* + - play

public void play(java.util.UUID matchId)

Joins a match and then randomly picks a direction to move in every 5th of a second. Stops when the Match is over.

Parameters:

matchId - UUID The ID of the match to join

* 1. Javascript

Proprietary Javascript is generally written in a single file because of the additional overhead of having multiple HTTP request to fetch various Javascript files, and also because having many separate Javascript files can introduce race conditions as different files load and execute in unpredictable times. Our proprietary Javascript file is *main.js*.

This makes it difficult to arrange Javascript in a readable manner but we’ve accounted for this by using Javascript namespacing (Croll, 2010). We created the root namespace “game” and from there have the following namespaces:

* game.menu- functionality surrounding menu buttons and switching between menu pages
* game.auth- functionality surrounding sending authorisation requests to and from the web service
* game.constants- a central location for storing constant values
* game.func- generic functionalities used across various namespaces
* game.match- functionality regarding joining, leaving, listing etc matches.
* game.match.var- variables pertaining to matches, for keeping track of a match status or list of matches.
* game.var- variables pertaining to the game, particularly involving graphics and timesteps.
* game.var.colours- the colours of various tiles in the first game display we wrote. Deprecated since we started using bitmap graphics.

Each function is, as much as possible, named in the most literal way to describe exactly what it does- such as “setAllTilesNotVisible” or “displayMatchMenu”.

The structure of our Javascript is to firstly declare each individual function and then when the document loads assigns functions to buttons, prepares the login screen and initialises a key press event listener for game interactivity.

What follows is a documentation of some but not all of the Javascript functions:

* game.func.get
  + Generic method to make a GET request. Uses xhr fields to ensure cookies are sent across domain.
  + Param “url”: {string} to send the request to
  + Param “data”: {string}to send with the request
  + Param “success”: function to execute on success
  + Param “error”: function to execute on failure
* game.func.post
* Generic method to make a POST request. Uses xhr fields to ensure cookies are sent across domain.
  + Param “url”: {string}to send the request to
  + Param “data”: {string}to send with the request
  + Param “success”: function to execute on success
  + Param “error”: function to execute on failure
* game.func.getApiPath
  + Constructs an url of an endpoint given the endpoint's controller and action names.
  + Param “controller”: {string} the controller to contact
  + Param “action”: {string} the action to contact
  + Returns: (string) the constructed path
* game.menu.displayScoreboard
  + Renders the scoreboard in the #score-table table.
  + Param “scoreboard”: the JSON object returned from a query to the score/top endpoint
* game.match.new
  + Starts a new Match by sending a request to the web service
* game.match.start
  + Starts the player's current Match by sending a request to the web service
* game.match.initGameScreen
  + Initialises the game screen. Resets game variables and creates a new HTML5 canvas. Begins the game loop.
* game.render
  + Renders the current game state to the canvas.
* game.updateStatus
  + Updates the current game status based on the result from a status request.
  + Will end the gam eif the status response indicates that the game is over.
* game.var.addTile
  + Adds a tile if it doesn't already exist in memory, or updates the tile if it does.
  + Will expand the size of game.var.tiles if it isn't large enough.
* game.updateGame
  + Makes a game status request if game.var.timestep has passed since the last request.
  + Loops while game.var.isRunning
* game.match.updateMatchList
  + Makes a match list request if game.match.var.timeStep has passed since the last request
* game.match.updateStatus
  + Makes a Match Status request if game.match.var.timeStep has passed since the last request.
  + Loops while game.match.var.isWaitingToStart
* game.match.displayMatchMenu
  + Generates the Match Status details on the Match screen.
  + Param “data”: Match Status as a JSON object
* game.menu.showEndGameScreen
  + Builds the end-game screen and switches to it
  + Param “result”: a MatchResultModel object

1. Meeting Minutes

|  |  |
| --- | --- |
| Accent icon | Meeting Minutes |

Call to order

A meeting of Team B was held at EB0.7 on 09-NOV-2016.

Attendees

Attendees included Arya, Mattsi, Anastasios, Xiao, Qian, Selin.

Members not in attendance

-

Approval of minutes

* Sprint 0
* Selin was elected as team leader unanimously.
* Format was set for Project Diary.
* Requirement analysis.
* Trello Setup.
* Github Setup.
* Documentation of Use Cases

Reports

* Validated Use Cases
* Produced layout for login-registration screen and main menu.
* Create CRC cards for the first two use cases.

Unfinished business

* Create CRC from all Use cases

New business

* Create tasks from Use cases

Announcements

-

|  |  |  |
| --- | --- | --- |
| Mattsi Jansky |  | 09-NOV-2016 |
| Secretary |  | Date of approval |

|  |  |
| --- | --- |
| Accent icon | Meeting Minutes |

Call to order

A meeting of Team B was held at CB5.12 on 14-NOV-2016.

Attendees

Attendees included Mattsi, Anastasios, Xiao, Qian, Selin.

Members not in attendance

Members not in attendance included Arya.

Approval of minutes

* Layout for login-registration screen and main menu uploaded on Trello.
* CRC cards for the first two use cases uploaded on Trello.

Reports

* Completed CRC cards for all use cases.
* The project was split into tasks.
* Time estimation on each task was set.

Unfinished business

## **Creating uses cases with server and client side**

New business

* UML design.

Announcements

-

|  |  |  |
| --- | --- | --- |
| Anastasios Gemtos |  | 14-NOV-2016 |
| Secretary |  | Date of approval |

|  |  |
| --- | --- |
| Accent icon | Meeting Minutes |

Call to order

A meeting of Team B was held at EB0.7 on 16-NOV-2016.

Attendees

Attendees included Mattsi, Anastasios, Xiao, Qian, Selin, Arya.

Members not in attendance

-

Approval of minutes

* CRC cards were added to project’s documentation.
* Few changes on time estimation were made and we added tasks on Trello.

Reports

* System architecture was discussed (3-tier architecture).
* Created UML classes for Server-side.

Unfinished business

* UML classes for client-side

New business

* Layouts for game screen and maps
* Database set up

Announcements

-

|  |  |  |
| --- | --- | --- |
| Anastasios Gemtos |  | 16-NOV-2016 |
| Secretary |  | Date of approval |

|  |  |
| --- | --- |
| Accent icon | Meeting Minutes |

Call to order

A meeting of Team B was held at CB5.12 on 21-NOV-2016.

Attendees

Attendees included Mattsi, Anastasios, Xiao, Qian, Selin, Arya.

Members not in attendance

-

Approval of minutes

* UML tool was used to create UML diagrams for server.

Reports

* Created UML classes for Client-side and made a few changes on Server-side UMLs.

Unfinished business

-

New business

* Review all the documents created so far.
* Start coding.

Announcements

-

|  |  |  |
| --- | --- | --- |
| Anastasios Gemtos |  | 21-NOV-2016 |
| Secretary |  | Date of approval |

|  |  |
| --- | --- |
| Accent icon | Meeting Minutes |

Call to order

A meeting of Team B was held at CB5.8 on 23-NOV-2016.

Attendees

Attendees included Mattsi, Anastasios, Xiao, Qian, Selin, Arya.

Members not in attendance

Approval of minutes

* UML diagrams were added into our project’s documentation.

Reports

* Reviewed all the documents about requirement analysis, Use Cases, CRC cards, system architecture and design, UML diagrams and test planning.

Unfinished business

* Need to integrate all documents into a single one.

New business

* Start coding.

Announcements

* The code development of the project begins next Monday

|  |  |  |
| --- | --- | --- |
| Anastasios Gemtos |  | 23-NOV-2016 |
| Secretary |  | Date of approval |

|  |  |
| --- | --- |
| Accent icon | Meeting Minutes |

Call to order

A meeting of Team B was held at CB5.12 on 28-NOV-2016.

Attendees

Attendees included Mattsi, Anastasios, Xiao, Qian, Selin, Arya.

Members not in attendance

-

Approval of minutes

* Changes were made to all documents to be clearer and less ambiguous.

Reports

* Created a single document that explains our system’s architecture and design based on requirement analysis.
* Created tasks on Trello (Sprint 4).
* Created a document for style conventions both for documentation and coding.
* Meeting with Julian Padget to demonstrate our progress so far.

Unfinished business

* Need to split the Use Cases into user stories and design use cases.
* Need to restructure our document.

New business

-

Announcements

-

|  |  |  |
| --- | --- | --- |
| Anastasios Gemtos |  | 28-NOV-2016 |
| Secretary |  | Date of approval |

|  |  |
| --- | --- |
| Accent icon | Meeting Minutes |

Call to order

A meeting of Team B was held at EB0.7 on 30-NOV-2016.

Attendees

Attendees included Mattsi, Anastasios, Xiao, Qian, Selin, Arya.

Members not in attendance

Approval of minutes

-

Reports

* We worked in pairs to complete database functionality, login/registration functionality and map parsing by the server.

Unfinished business

* Need to split the Use Cases into user stories and design use cases.
* Need to restructure our document.

New business

-

Announcements

-

|  |  |  |
| --- | --- | --- |
| Anastasios Gemtos |  | 30-NOV-2016 |
| Secretary |  | Date of approval |

|  |  |
| --- | --- |
| Accent icon | Meeting Minutes |

Call to order

A meeting of Team B was held at CB5.12 on 5-DEC-2016

Attendees

Attendees included Mattsi, Anastasios, Xiao, Qian, Selin, Arya.

Members not in attendance

Xiao

Approval of minutes

-

Reports

* We discussed about remaining functionality and tasks are assigned. Meeting with the customer is happened.

Unfinished business

* Score Implementation, bot functionality
* Need to restructure our document.
* Exit and leave condition
* User guide and tutorial

New business

-

Announcements

-

|  |  |  |
| --- | --- | --- |
| Selin Kutlamis |  | 05-DEC-2016 |
| Secretary |  | Date of approval |

|  |  |
| --- | --- |
| Accent icon | Meeting Minutes |

Call to order

A meeting of Team B was held at EB0.7 on 7-DEC-2016.

Attendees

Attendees included Mattsi, Anastasios, Xiao, Qian, Selin, Arya.

Members not in attendance

Xiao

Approval of minutes

-

Reports

* We worked in pairs to complete movement functionality, visibility functionality and documentation.

Unfinished business

* Score Implementation, bot functionality
* Need to restructure our document.
* Exit and leave condition
* User guide and tutorial

New business

-

Announcements

-

|  |  |  |
| --- | --- | --- |
| Selin KUTLAMIS |  | 07-DEC-2016 |
| Secretary |  | Date of approval |

|  |  |
| --- | --- |
| Accent icon | Meeting Minutes |

Call to order

A meeting of Team B was held at CB.5.12 on 12-DEC-2016.

Attendees

Attendees included Mattsi, Anastasios, Xiao, Qian, Selin, Arya.

Members not in attendance

-

Approval of minutes

-

Reports

* Discussion for documentation and improvement in the bug fixes.

Unfinished business

* Score interaction and exit condition

New business

-

Announcements

-

|  |  |  |
| --- | --- | --- |
| Selin KUTLAMIS |  | 12-DEC-2016 |
| Secretary |  | Date of approval |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | |  |  | |
| Accent icon | Meeting Minutes | | |

Call to order

A meeting of Team B was held at EB0.7 on 14-DEC-2016.

Attendees

Attendees included Mattsi, Anastasios, Xiao, Qian, Selin, Arya.

Members not in attendance

Approval of minutes

-

Reports

* Continued with documentation.

Unfinished business

-

New business

-

Announcements

-

|  |  |  |
| --- | --- | --- |
| Selin KUTLAMIS |  | 14-DEC-2016 |
| Secretary |  | Date of approval |

1. References

ISTQB Exam Certification. (n.d.). *What is Component testing?* Retrieved 11 26, 2016, from ISTQB Exam Certification: http://istqbexamcertification.com/what-is-component-testing/

Kaner, C. (2006, Novemer 17). *Exploratory Testing from Florida Institute of Technology, Quality Assurance Institute Worldwide Annual Software Testing Conference, Orlando, FL.* Retrieved from Kaner.com: http://www.kaner.com/pdfs/ETatQAI.pdf