

# Reference Manual

Generated by Doxygen 1.7.3

Thu Nov 10 2011 22:49:42



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	Action Class Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Constructor & Destructor Documentation . . . . .	8
4.1.2.1	Action . . . . .	8
4.1.2.2	Action . . . . .	8
4.1.3	Member Function Documentation . . . . .	8
4.1.3.1	dribbleToGoal . . . . .	8
4.1.3.2	findBall . . . . .	8
4.1.3.3	goHome . . . . .	9
4.1.3.4	gotoPoint . . . . .	9
4.1.3.5	gotoPoint . . . . .	9
4.1.3.6	kickToPoint . . . . .	10
4.1.3.7	kickToPoint . . . . .	10
4.1.3.8	setMem . . . . .	10
4.2	Brain Class Reference . . . . .	11
4.2.1	Detailed Description . . . . .	11
4.2.2	Constructor & Destructor Documentation . . . . .	11
4.2.2.1	Brain . . . . .	11
4.2.2.2	Brain . . . . .	12
4.2.3	Member Function Documentation . . . . .	12
4.2.3.1	getActions . . . . .	12
4.2.3.2	getCurrentMode . . . . .	12
4.2.3.3	getMarked_team . . . . .	12
4.2.3.4	getMarked_unum . . . . .	12
4.2.3.5	run . . . . .	12
4.2.3.6	setActions . . . . .	12
4.2.3.7	setDefensive . . . . .	13
4.2.3.8	setMarked_team . . . . .	13
4.2.3.9	setMarked_unum . . . . .	13

4.2.3.10	setOffensive	13
4.3	Field Class Reference	13
4.3.1	Detailed Description	13
4.3.2	Constructor & Destructor Documentation	14
4.3.2.1	Field	14
4.4	Forward Class Reference	14
4.4.1	Detailed Description	14
4.5	FullBack Class Reference	15
4.5.1	Detailed Description	15
4.6	Game Class Reference	15
4.6.1	Detailed Description	15
4.7	Goalie Class Reference	15
4.7.1	Detailed Description	16
4.7.2	Member Function Documentation	16
4.7.2.1	ballInGoalzone	16
4.7.2.2	catchable	17
4.7.2.3	catchball	17
4.7.2.4	closestPlayer	17
4.7.2.5	defendGoal	18
4.7.2.6	followBall	18
4.7.2.7	getBtwBallAndGoal	18
4.7.2.8	initGoalie	19
4.7.2.9	kickBallOutOfBounds	19
4.7.2.10	kickToPlayer	19
4.7.2.11	run	19
4.8	MathHelp Class Reference	20
4.8.1	Detailed Description	20
4.8.2	Member Function Documentation	21
4.8.2.1	edp	21
4.8.2.2	getDashPower	21
4.8.2.3	getKickPower	21
4.8.2.4	getKickPower	22
4.8.2.5	getNextBallPoint	22
4.8.2.6	getNextOpponentPoint	22
4.8.2.7	getPolar	22
4.8.2.8	getPolar	23
4.8.2.9	getPos	23
4.8.2.10	getPos	23
4.8.2.11	mag	24
4.8.2.12	norm	24
4.8.2.13	norm	24
4.8.2.14	vAdd	24
4.8.2.15	vDiv	25
4.8.2.16	vMul	25
4.8.2.17	vSub	25
4.9	Memory Class Reference	25
4.9.1	Constructor & Destructor Documentation	27
4.9.1.1	Memory	27
4.9.2	Member Function Documentation	27
4.9.2.1	getAmountOfSpeed	27

4.9.2.2	getBall	27
4.9.2.3	getClosestBoundary	27
4.9.2.4	getClosestFlag	27
4.9.2.5	getClosestLine	28
4.9.2.6	getClosestPenaltyFlag	28
4.9.2.7	getDirection	28
4.9.2.8	getDirectionOfSpeed	28
4.9.2.9	getEffort	28
4.9.2.10	getFlag	28
4.9.2.11	getFlagPos	29
4.9.2.12	getHeadDirection	29
4.9.2.13	getLine	29
4.9.2.14	getNullGoalAngle	29
4.9.2.15	getObj	30
4.9.2.16	getObjMemorySize	30
4.9.2.17	getOppGoal	30
4.9.2.18	getOppGoalPos	31
4.9.2.19	getOwnGoal	31
4.9.2.20	getOwnGoalPos	31
4.9.2.21	getPlayer	31
4.9.2.22	getPlayers	31
4.9.2.23	getPlayMode	32
4.9.2.24	getPosition	32
4.9.2.25	getRecovery	32
4.9.2.26	getStamina	32
4.9.2.27	isObjVisible	32
4.9.2.28	setField	32
4.9.2.29	setLocation	33
4.9.2.30	timeCheck	33
4.9.3	Member Data Documentation	33
4.9.3.1	ObjMem	33
4.9.3.2	oppGoal	33
4.9.3.3	oppSide	33
4.9.3.4	playMode	34
4.9.3.5	SenMem	34
4.9.3.6	side	34
4.9.3.7	uNum	34
4.10	Mode Class Reference	34
4.10.1	Detailed Description	34
4.10.2	Constructor & Destructor Documentation	34
4.10.2.1	Mode	34
4.10.3	Member Function Documentation	35
4.10.3.1	getModename	35
4.10.3.2	getTimeinmode	35
4.10.3.3	setModename	35
4.10.3.4	setTimeinmode	35
4.11	ObjBall Class Reference	35
4.11.1	Detailed Description	35
4.12	ObjFlag Class Reference	36
4.12.1	Constructor & Destructor Documentation	36

4.12.1.1	ObjFlag	36
4.12.2	Member Function Documentation	36
4.12.2.1	getFlagName	36
4.12.2.2	getFlagType	37
4.12.2.3	getX_pos	37
4.12.2.4	getY_pos	37
4.12.2.5	getYard	37
4.12.2.6	setFlagName	37
4.12.2.7	setFlagType	37
4.12.2.8	setX_pos	37
4.12.2.9	setY_pos	38
4.12.2.10	setYard	38
4.13	ObjGoal Class Reference	38
4.13.1	Detailed Description	38
4.14	ObjInfo Class Reference	38
4.14.1	Detailed Description	39
4.14.2	Constructor & Destructor Documentation	39
4.14.2.1	ObjInfo	39
4.14.2.2	ObjInfo	39
4.14.3	Member Function Documentation	39
4.14.3.1	getDirChng	39
4.14.3.2	getDirection	40
4.14.3.3	getDistance	40
4.14.3.4	getDistChng	40
4.14.3.5	getObjName	40
4.14.3.6	getSide	40
4.14.3.7	setDirChng	40
4.14.3.8	setDirection	40
4.14.3.9	setDistance	40
4.14.3.10	setDistChng	41
4.14.3.11	setObjName	41
4.14.3.12	setSide	41
4.15	ObjLine Class Reference	41
4.15.1	Detailed Description	41
4.16	ObjMemory Class Reference	41
4.16.1	Detailed Description	42
4.16.2	Constructor & Destructor Documentation	42
4.16.2.1	ObjMemory	42
4.16.2.2	ObjMemory	42
4.16.3	Member Function Documentation	42
4.16.3.1	addInfo	42
4.16.3.2	getObj	43
4.16.3.3	getObj	43
4.16.3.4	getSize	43
4.16.3.5	getTime	44
4.16.3.6	setTime	44
4.17	ObjPlayer Class Reference	44
4.17.1	Detailed Description	45
4.17.2	Member Function Documentation	45
4.17.2.1	getBodyDir	45

4.17.2.2	<a href="#">getHeadDir</a>	45
4.17.2.3	<a href="#">getTeam</a>	45
4.17.2.4	<a href="#">getuNum</a>	45
4.17.2.5	<a href="#">isGoalie</a>	45
4.17.2.6	<a href="#">setBodyDir</a>	46
4.17.2.7	<a href="#">setGoalie</a>	46
4.17.2.8	<a href="#">setHeadDir</a>	46
4.17.2.9	<a href="#">setTeam</a>	46
4.17.2.10	<a href="#">setuNum</a>	46
4.18	<a href="#">Parser Class Reference</a>	46
4.18.1	<a href="#">Detailed Description</a>	46
4.18.2	<a href="#">Constructor &amp; Destructor Documentation</a>	47
4.18.2.1	<a href="#">Parser</a>	47
4.18.3	<a href="#">Member Function Documentation</a>	47
4.18.3.1	<a href="#">initParse</a>	47
4.18.3.2	<a href="#">Parse</a>	47
4.18.4	<a href="#">Member Data Documentation</a>	48
4.18.4.1	<a href="#">input</a>	48
4.19	<a href="#">Player Class Reference</a>	48
4.19.1	<a href="#">Detailed Description</a>	49
4.19.2	<a href="#">Constructor &amp; Destructor Documentation</a>	49
4.19.2.1	<a href="#">Player</a>	49
4.19.3	<a href="#">Member Function Documentation</a>	50
4.19.3.1	<a href="#">closestOpponent</a>	50
4.19.3.2	<a href="#">dash</a>	50
4.19.3.3	<a href="#">getDirection</a>	50
4.19.3.4	<a href="#">getMem</a>	51
4.19.3.5	<a href="#">getObjInfo</a>	51
4.19.3.6	<a href="#">getParser</a>	51
4.19.3.7	<a href="#">getPosition</a>	51
4.19.3.8	<a href="#">getRoboClient</a>	51
4.19.3.9	<a href="#">getTime</a>	51
4.19.3.10	<a href="#">initPlayer</a>	51
4.19.3.11	<a href="#">kick</a>	52
4.19.3.12	<a href="#">move</a>	52
4.19.3.13	<a href="#">receiveInput</a>	53
4.19.3.14	<a href="#">run</a>	53
4.19.3.15	<a href="#">say</a>	53
4.19.3.16	<a href="#">setBrain</a>	53
4.19.3.17	<a href="#">setMem</a>	54
4.19.3.18	<a href="#">setObjInfo</a>	54
4.19.3.19	<a href="#">setParser</a>	54
4.19.3.20	<a href="#">setRoboclient</a>	54
4.19.3.21	<a href="#">setTime</a>	54
4.19.3.22	<a href="#">turn</a>	54
4.20	<a href="#">Polar Class Reference</a>	55
4.20.1	<a href="#">Detailed Description</a>	55
4.20.2	<a href="#">Constructor &amp; Destructor Documentation</a>	55
4.20.2.1	<a href="#">Polar</a>	55
4.20.2.2	<a href="#">Polar</a>	56

4.21	Pos Class Reference	56
4.21.1	Detailed Description	56
4.21.2	Constructor & Destructor Documentation	56
4.21.2.1	Pos	56
4.21.2.2	Pos	57
4.21.2.3	Pos	57
4.22	RoboClient Class Reference	57
4.22.1	Detailed Description	58
4.22.2	Constructor & Destructor Documentation	58
4.22.2.1	RoboClient	58
4.22.2.2	RoboClient	58
4.22.3	Member Function Documentation	58
4.22.3.1	catchball	58
4.22.3.2	dash	59
4.22.3.3	getPort	59
4.22.3.4	getTeam	59
4.22.3.5	init	59
4.22.3.6	initGoalie	60
4.22.3.7	kick	60
4.22.3.8	move	60
4.22.3.9	receive	61
4.22.3.10	say	61
4.22.3.11	send	62
4.22.3.12	setTeam	62
4.22.3.13	turn	62
4.23	SenseMemory Class Reference	63
4.23.1	Detailed Description	63
4.23.2	Constructor & Destructor Documentation	63
4.23.2.1	SenseMemory	63
4.23.2.2	SenseMemory	63
4.23.3	Member Function Documentation	64
4.23.3.1	getTime	64
4.23.3.2	setTime	64
4.23.3.3	setTime	64
<b>5</b>	<b>File Documentation</b>	<b>65</b>
5.1	Action.java File Reference	65
5.1.1	Detailed Description	65
5.2	Brain.java File Reference	65
5.2.1	Detailed Description	65
5.3	Field.java File Reference	66
5.3.1	Detailed Description	66
5.4	Forward.java File Reference	66
5.4.1	Detailed Description	66
5.5	FullBack.java File Reference	67
5.5.1	Detailed Description	67
5.6	Game.java File Reference	67
5.6.1	Detailed Description	67
5.7	Goalie.java File Reference	67
5.7.1	Detailed Description	67



5.8	MathHelp.java File Reference . . . . .	68
5.8.1	Detailed Description . . . . .	68
5.9	Memory.java File Reference . . . . .	68
5.9.1	Detailed Description . . . . .	68
5.10	Mode.java File Reference . . . . .	69
5.10.1	Detailed Description . . . . .	69
5.11	ObjInfo.java File Reference . . . . .	69
5.11.1	Detailed Description . . . . .	69
5.12	ObjMemory.java File Reference . . . . .	70
5.12.1	Detailed Description . . . . .	70
5.13	Parser.java File Reference . . . . .	70
5.13.1	Detailed Description . . . . .	70
5.14	Player.java File Reference . . . . .	71
5.14.1	Detailed Description . . . . .	71
5.15	Pos.java File Reference . . . . .	71
5.15.1	Detailed Description . . . . .	71
5.16	RoboClient.java File Reference . . . . .	71
5.16.1	Detailed Description . . . . .	72
5.17	SenseMemory.java File Reference . . . . .	72
5.17.1	Detailed Description . . . . .	72



# Chapter 1

## Class Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Action . . . . .	7
Brain . . . . .	11
Field . . . . .	13
Game . . . . .	15
MathHelp . . . . .	20
Memory . . . . .	25
Mode . . . . .	34
ObjInfo . . . . .	38
ObjBall . . . . .	35
ObjFlag . . . . .	36
ObjGoal . . . . .	38
ObjLine . . . . .	41
ObjPlayer . . . . .	44
ObjMemory . . . . .	41
Parser . . . . .	46
Player . . . . .	48
Forward . . . . .	14
FullBack . . . . .	15
Goalie . . . . .	15
Polar . . . . .	55
Pos . . . . .	56
RoboClient . . . . .	57
SenseMemory . . . . .	63



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Action	7
Brain	11
Field	13
Forward	14
FullBack	15
Game	15
Goalie	15
MathHelp	20
Memory	25
Mode	34
ObjBall	35
ObjFlag	36
ObjGoal	38
ObjInfo	38
ObjLine	41
ObjMemory	41
ObjPlayer	44
Parser	46
Player	48
Polar	55
Pos	56
RoboClient	57
SenseMemory	63



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

Action.java	65
Brain.java	65
Field.java	66
Forward.java	66
FullBack.java	67
Game.java	67
Goalie.java	67
MathHelp.java	68
Memory.java	68
Mode.java	69
ObjInfo.java	69
ObjMemory.java	70
Parser.java	70
Player.java	71
Pos.java	71
RoboClient.java	71
SenseMemory.java	72





## Chapter 4

# Class Documentation

### 4.1 Action Class Reference

#### Public Member Functions

- [Action](#) ()
- [Action](#) ([Memory](#) mem, [RoboClient](#) rc)
- void [setMem](#) ([Memory](#) mem)
- void [gotoPoint](#) ([Polar](#) go)
- void [gotoPoint](#) ([Pos](#) p)
- boolean [goHome](#) ()
- void [findBall](#) () throws `UnknownHostException`, `InterruptedException`
- void [kickToPoint](#) ([ObjBall](#) ball, [Polar](#) p)
- void [kickToPoint](#) ([ObjBall](#) ball, [Pos](#) p)
- void [dribbleToGoal](#) ([ObjBall](#) ball)

#### Public Attributes

- [MathHelp](#) **m** = new [MathHelp](#)()
- [Memory](#) **mem**
- [RoboClient](#) **rc**
- [Polar](#) **OppGoal**
- boolean **atGoal**

#### 4.1.1 Detailed Description

This class holds basic actions for the player to perform, such as ball searching and intercepting, dashing to points, finding the ball and points and getting their coordinates.

## 4.1.2 Constructor & Destructor Documentation

### 4.1.2.1 Action::Action ( ) [inline]

Default constructor

### 4.1.2.2 Action::Action ( Memory *mem*, RoboClient *rc* ) [inline]

Constructor with parameters

#### Parameters

<i>mem</i>	The <a href="#">Memory</a> containing all the parsed information from the server
<i>rc</i>	The <a href="#">RoboClient</a> that is the player's connection to the server

#### Precondition

Both a full memory and initialized [RoboClient](#) must be passed in to avoid any errors

#### Postcondition

A new set of actions will be available for the player to call on

## 4.1.3 Member Function Documentation

### 4.1.3.1 void Action::dribbleToGoal ( ObjBall *ball* ) [inline]

This dribbles the ball in the direction of the goal until it's 18 feet outside of the goal, when it kicks the ball with maximum power into the goal.

#### Parameters

<i>ball</i>	
-------------	--

#### Precondition

The ball should not be null

#### Postcondition

This will result in a dribble and a shoot

### 4.1.3.2 void Action::findBall ( ) throws UnknownHostException, InterruptedException [inline]

A method to find the ball on the field. If it's not in view, the player turns until he finds it. If the ball is too far, he dashes to get to it. If the ball is within 20 distance, he intercepts the ball.

**Exceptions**

<i>UnknownHostException</i>	
<i>InterruptedException</i>	

**4.1.3.3 boolean Action::goHome ( ) [inline]**

Take the [Player](#) back to his home

**Precondition**

The player's home should be set at initialization

**Postcondition**

The player will be at his home point

**Returns**

true if the player is in the near vicinity of his home, false if he's not there yet

**4.1.3.4 void Action::gotoPoint ( Polar go ) [inline]**

This tells the player to turn and run to a point

**Parameters**

<i>go</i>	The <a href="#">Polar</a> coordinates of the final position, with the player's position as an origin
-----------	--

**Precondition**

The player must have a valid position on the field passed in

**Postcondition**

If the player is not facing the direction of the final position, s/he will first turn toward it. If the player is approximately facing the position, s/he will dash toward the direction of the position.

**4.1.3.5 void Action::gotoPoint ( Pos p ) [inline]**

A cartesian wrapper for the gotoPoint with [Polar](#) coordinate

**Parameters**

<i>p</i>	The Cartesian <a href="#">Pos</a> of position to go to
----------	--

**Precondition**

The player must have a valid position on the field passed in

**Postcondition**

First, the [Pos](#) will be converted to a [Polar](#) coordinate. If the player is not facing the direction of the final position, s/he will turn toward it. If the player is approximately facing the position, s/he will dash toward the direction of the position.

**4.1.3.6 void Action::kickToPoint ( [ObjBall](#) *ball*, [Polar](#) *p* ) [inline]**

Kicks ball to a certain [Polar](#) point

**Parameters**

<i>ball</i>	
<i>p</i>	The <a href="#">Polar</a> coordinate to kick the ball to

**Precondition**

The ball passed in should not be null and p should be within the field from the player

**Postcondition**

The ball will be kicked to the vicinity of the point

**4.1.3.7 void Action::kickToPoint ( [ObjBall](#) *ball*, [Pos](#) *p* ) [inline]**

A [Pos](#) wrapper for the kickToPoint

**Parameters**

<i>ball</i>	
<i>p</i>	the <a href="#">Pos</a> of the coordinate to kick the ball to

**4.1.3.8 void Action::setMem ( [Memory](#) *mem* ) [inline]**

This sets the [Memory](#) for the action to use. This is important as the [Memory](#) is constantly changing, and must be updated at every step.

**Parameters**

<i>mem</i>	The player's <a href="#">Memory</a>
------------	-------------------------------------

**Precondition**

The [Memory](#) should be the most up to date

**Postcondition**

The actions that require a [Memory](#) will be able to pull from it

The documentation for this class was generated from the following file:

- [Action.java](#)

## 4.2 Brain Class Reference

**Public Member Functions**

- [Brain](#) ()
- **Brain** ([Player](#) p)
- [Action](#) [getActions](#) ()
- void [setActions](#) ([Action](#) actions)
- [Brain](#) ([Mode](#) currentMode)
- [Mode](#) [getCurrentMode](#) ()
- void [setDefensive](#) ()
- void [setOffensive](#) ()
- String [getMarked\\_team](#) ()
- void [setMarked\\_team](#) (String marked\_team)
- String [getMarked\\_unum](#) ()
- void [setMarked\\_unum](#) (String marked\_unum)
- void [run](#) ()

**Public Attributes**

- [Player](#) p
- [Memory](#) m

### 4.2.1 Detailed Description

The brain serves as a place to store the [Player](#) modes, marked players for various functions, and a set of strategies for player actions.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 [Brain::Brain](#) ( ) [inline]

Default constructor

#### 4.2.2.2 Brain::Brain ( Mode *currentMode* ) [inline]

Constructor

##### Parameters

<i>current- Mode</i>	
--------------------------	--

#### 4.2.3 Member Function Documentation

##### 4.2.3.1 Action Brain::getActions ( ) [inline]

###### Returns

the actions

##### 4.2.3.2 Mode Brain::getCurrentMode ( ) [inline]

###### Returns

the currentMode

##### 4.2.3.3 String Brain::getMarked\_team ( ) [inline]

###### Returns

the marked\_team

##### 4.2.3.4 String Brain::getMarked\_unum ( ) [inline]

###### Returns

the marked\_unum

##### 4.2.3.5 void Brain::run ( ) [inline]

The [Brain](#) thread run method. It updates the [Memory](#) for the [Player](#)

###### Postcondition

[Memory](#) will continuously update

##### 4.2.3.6 void Brain::setActions ( Action *actions* ) [inline]

###### Parameters

<i>actions</i>	the actions to set
----------------	--------------------

#### 4.2.3.7 void Brain::setDefensive ( ) [inline]

Sets the player mode to defensive

#### 4.2.3.8 void Brain::setMarked\_team ( String *marked\_team* ) [inline]

##### Parameters

<i>marked_</i> - <i>team</i>	the marked_team to set
---------------------------------	------------------------

#### 4.2.3.9 void Brain::setMarked\_unum ( String *marked\_unum* ) [inline]

##### Parameters

<i>marked_</i> - <i>unum</i>	the marked_unum to set
---------------------------------	------------------------

#### 4.2.3.10 void Brain::setOffensive ( ) [inline]

Sets the player mode to be offensive

The documentation for this class was generated from the following file:

- [Brain.java](#)

## 4.3 Field Class Reference

### Public Member Functions

- [Field](#) (String side)

### Public Attributes

- ArrayList< [Pos](#) > **posList** = new ArrayList<[Pos](#)>()

#### 4.3.1 Detailed Description

This creates an ArrayList that holds all the coordinates for the fixed points on the field. As the orientation of the axes depends on the side of the field the starts on, there are two sets of coordinates, each with opposite signs.

**Author**

Grant Hays

**4.3.2 Constructor & Destructor Documentation****4.3.2.1 Field::Field ( String *side* ) [inline]**

[Field](#) constructor

**Parameters**

<i>side</i>	The side of the field the player's team starts on
-------------	---

**Precondition**

The side needs to be parsed from the server's (init) message and passed as the argument

**Postcondition**

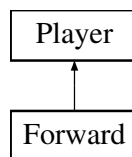
A new [Field](#) will be created with access to an array list of all the field's fixed points

The documentation for this class was generated from the following file:

- [Field.java](#)

**4.4 Forward Class Reference**

Inheritance diagram for Forward:

**4.4.1 Detailed Description**

The [Forward](#) class inherits from the [Player](#) class. The [Forward](#) is a specialized type of [Player](#) that focuses on offensive behaviors such as scoring and ball interception.

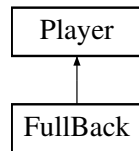
The documentation for this class was generated from the following file:

- [Forward.java](#)



## 4.5 FullBack Class Reference

Inheritance diagram for FullBack:



### 4.5.1 Detailed Description

The [FullBack](#) class inherits from the [Player](#) class. The [FullBack](#) is a specialized type of [Player](#) that focuses on defensive behaviors such as interfering with opponent scoring.

The documentation for this class was generated from the following file:

- [FullBack.java](#)

## 4.6 Game Class Reference

### Static Public Member Functions

- static void **main** (String args[ ]) throws Exception

### 4.6.1 Detailed Description

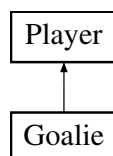
This serves as a main class to assemble the RoboCup team and set them into action for the match.

The documentation for this class was generated from the following file:

- [Game.java](#)

## 4.7 Goalie Class Reference

Inheritance diagram for Goalie:



## Public Member Functions

- void [initGoalie](#) () throws SocketException, UnknownHostException
- void [catchball](#) (double d) throws UnknownHostException
- void [followBall](#) ()
- boolean [ballInGoalzone](#) (ObjBall ball)
- boolean [catchable](#) ()
- void [defendGoal](#) (ObjBall ball) throws UnknownHostException, InterruptedException
- void [getBtwBallAndGoal](#) (ObjBall ball)
- [ObjPlayer](#) [closestPlayer](#) () throws UnknownHostException, InterruptedException
- void [kickToPlayer](#) (ObjPlayer player)
- void [kickBallOutOfBounds](#) ()
- void [run](#) ()

## Public Attributes

- boolean **ballTurn** = false
- [MathHelp](#) **mh** = new [MathHelp](#)()

## Package Attributes

- boolean **ballCaught** = false

### 4.7.1 Detailed Description

The [Goalie](#) class inherits from the [Player](#) class. The [Goalie](#) is a specialized type of [Player](#) that may catch the ball under certain conditions and defends the goal from the opposing team.

### 4.7.2 Member Function Documentation

#### 4.7.2.1 boolean Goalie::ballInGoalzone ( ObjBall ball ) [inline]

A method to determine whether the ball is in the penalty box

#### Parameters

<i>ball</i>	the <a href="#">ObjBall</a> to follow
-------------	---------------------------------------

#### Precondition

this must be called with an [ObjBall](#)

**Postcondition**

true if ball is in penalty box, false if it's not

**Returns**

boolean

**4.7.2.2 boolean Goalie::catchable ( ) [inline]**

Returns true or false depending on whether the ball is within the catchable range of the goalie.

**Precondition**

The ball is visible to the goalie

**Postcondition**

The ball is determined to catchable or not.

**Returns**

boolean True if catchable, false if not.

**4.7.2.3 void Goalie::catchball ( double *d* ) throws UnknownHostException [inline]**

Causes the [Goalie](#) to catch the ball.

**Precondition**

Playmode is play-on, ball is within goalkeeper zone and in the catchable area.

**Postcondition**

The [Goalie](#) has caught the ball.

**4.7.2.4 ObjPlayer Goalie::closestPlayer ( ) throws UnknownHostException, InterruptedException [inline]**

Returns the closest player to the goalie on the same team.

**Postcondition**

The closest player to the goalie has been determined.

**Returns**

[ObjPlayer](#)

**Exceptions**

<i>InterruptedException</i>	
<i>UnknownHostException</i>	

**4.7.2.5 void Goalie::defendGoal ( [ObjBall](#) *ball* ) throws [UnknownHostException](#), [InterruptedException](#) [inline]**

Causes the goalie to act to intercept the ball as it approaches the goal.

#### Parameters

<a href="#">ObjBall</a>	representing the ball in play.
-------------------------	--------------------------------

#### Exceptions

<i>UnknownHostException</i>	
<i>InterruptedException</i>	

#### Precondition

The ball has entered the goal zone.

#### Postcondition

The ball has been caught by the goalie, or the goalie has missed the ball.

**4.7.2.6 void Goalie::followBall ( ) [inline]**

Turns goalie toward the ball

#### Postcondition

The goalie will turn in the direction of the ball

**4.7.2.7 void Goalie::getBtwBallAndGoal ( [ObjBall](#) *ball* ) [inline]**

Moves goalie between the ball and the goal (under construction)

#### Parameters

<i>ball</i>	An <a href="#">ObjBall</a> .
-------------	------------------------------

#### Precondition

Ball is visible to the goalie.

**Postcondition**

The goalie has moved to a point on the line between the ball and the goal.

**4.7.2.8** `void Goalie::initGoalie ( ) throws SocketException, UnknownHostException`  
[inline]

Initializes the [Player](#) with the RoboCup server as a goalie.

**Precondition**

A RoboCup server is available.

**Postcondition**

The [Player](#) has been initialized to the correct team as a goalie.

**4.7.2.9** `void Goalie::kickBallOutOfBounds ( )` [inline]

Causes the goalie to kick the ball out of bounds

**Precondition**

[Goalie](#) has control of the ball

**Postcondition**

Ball has been kicked out of bounds

**4.7.2.10** `void Goalie::kickToPlayer ( ObjPlayer player )` [inline]

Causes goalie to kick the ball to a specific player.

**Precondition**

A player is in sight of the goalie.

**Postcondition**

The goalie has kicked the ball to the player passed to the function.

**Parameters**

<i>player</i>	An <a href="#">ObjPlayer</a> representing the player to receive the ball.
---------------	---

**4.7.2.11** `void Goalie::run ( )` [inline]

The [Player](#) thread run method. It makes decisions for the player.

**Postcondition**

[Player](#) will act on decisions made.

Reimplemented from [Player](#).

The documentation for this class was generated from the following file:

- [Goalie.java](#)

## 4.8 MathHelp Class Reference

**Public Member Functions**

- [Pos](#) [getPos](#) (double r, double t)
- [Pos](#) [getPos](#) ([Polar](#) p)
- [Polar](#) [getPolar](#) (double x, double y)
- [Polar](#) [getPolar](#) ([Pos](#) p)
- [Pos](#) [vAdd](#) ([Pos](#) p1, [Pos](#) p2)
- [Pos](#) [vSub](#) ([Pos](#) p2, [Pos](#) p1)
- [Pos](#) [vMul](#) ([Pos](#) p, double n)
- [Pos](#) [vDiv](#) ([Pos](#) p, double n)
- double [mag](#) ([Pos](#) p)
- [Pos](#) [norm](#) ([Pos](#) p)
- [Pos](#) [norm](#) (double dist, [Pos](#) a)
- double [edp](#) (double effort, double stamina)
- double [getDashPower](#) ([Pos](#) p, double vel\_r, double vel\_t, double effort, double stamina)
- [Polar](#) [getNextBallPoint](#) ([ObjBall](#) ball)
- [Polar](#) [getNextOpponentPoint](#) ([ObjPlayer](#) opponent)
- double [getKickPower](#) ([Polar](#) p, double vel\_r, double vel\_t, double ball\_r, double ball\_t)
- double [getKickPower](#) ([Pos](#) p, double vel\_r, double vel\_t, double ball\_r, double ball\_t)

### 4.8.1 Detailed Description

This contains all the equations and calculators needed for various methods

**Author**

granthays

## 4.8.2 Member Function Documentation

### 4.8.2.1 `double MathHelp::edp ( double effort, double stamina ) [inline]`

The Effective Dash Power

#### Parameters

<i>effort</i>	From the stamina in the <a href="#">SenseMemory</a>
<i>power</i>	The Power of the dash

#### Returns

the product of effort x power x dash\_power\_rate (0.006)

### 4.8.2.2 `double MathHelp::getDashPower ( Pos p, double vel_r, double vel_t, double effort, double stamina ) [inline]`

A calculator for power needed to get to a position on the field. This is derived from the Movement Model equations in the Server Manual: section 4.4

#### Parameters

<i>p</i>	the position to go to
<i>vel_r</i>	the magnitude of the player's velocity
<i>vel_t</i>	the direction of the player's velocity

#### Returns

The power needed to accelerate the player to the desired location

### 4.8.2.3 `double MathHelp::getKickPower ( Polar p, double vel_r, double vel_t, double ball_r, double ball_t ) [inline]`

Calculates the power needed to kick the ball to a specified place on the field, using the equation from the manual

#### Parameters

<i>p</i>	A polar coordinate to kick the ball to
<i>vel_r</i>	The magnitude of the player's velocity
<i>vel_t</i>	the direction of the player's velocity
<i>ball_r</i>	the distance of the ball to the player
<i>ball_t</i>	the direction of the ball to the player

#### Returns

power of kick

#### 4.8.2.4 `double MathHelp::getKickPower ( Pos p, double vel_r, double vel_t, double ball_r, double ball_t ) [inline]`

A wrapper of the getKickPower with a [Pos](#) instead of [Polar](#)

##### Parameters

<i>p</i>	A polar coordinate to kick the ball to
<i>vel_r</i>	The magnitude of the player's velocity
<i>vel_t</i>	the direction of the player's velocity
<i>ball_r</i>	the distance of the ball to the player
<i>ball_t</i>	the direction of the ball to the player

##### Returns

power of kick

#### 4.8.2.5 `Polar MathHelp::getNextBallPoint ( ObjBall ball ) [inline]`

A method to find the ball's next point given it's velocity and position relative to player.

##### Parameters

<i>ball</i>	
-------------	--

##### Returns

A [Polar](#) coordinate with the theoretical position of the ball at time t+1

#### 4.8.2.6 `Polar MathHelp::getNextOpponentPoint ( ObjPlayer opponent ) [inline]`

A method to find an opponent's next point given his velocity and position relative to the player.

##### Parameters

<i>opponent</i>	An <a href="#">ObjPlayer</a> object representing the opponent to track
-----------------	--

##### Returns

A [Polar](#) coordinate with the predicted position of the opponent at time t+1

#### 4.8.2.7 `Polar MathHelp::getPolar ( Pos p ) [inline]`

Cartesian to polar wrapper

This is just a wrapper, so you can pass in a [Pos](#) instead of extracting it's x and y and passing them in.



**Parameters**

$p$	the Cartesian vector
-----	----------------------

**Returns**

A new [Polar](#) vector converted from the Cartesian vector

**4.8.2.8 Polar MathHelp::getPolar ( double x, double y ) [inline]**

Cartesian to polar converter

**Parameters**

$x$	the x coordinate of the Cartesian vector
$y$	the y coordinate of the Cartesian vector

**Returns**

A new [Polar](#) vector converted from the Cartesian vector

**4.8.2.9 Pos MathHelp::getPos ( Polar p ) [inline]**

[Polar](#) to Cartesian wrapper

This allows you to pass a whole polar in, instead of extracting it's r and t variables and passing them in

**Parameters**

$p$	The polar coordinates you want to convert
-----	---

**Returns**

A new [Pos](#) with the Cartesian version of your [Polar](#) vector

**4.8.2.10 Pos MathHelp::getPos ( double r, double t ) [inline]**

[Polar](#) to Cartesian converter

**Parameters**

$r$	the length of the <a href="#">Polar</a> arm
$t$	the angle, in degrees, of the arm from the x-axis

**Returns**

A new Cartesian [Pos](#) converted from the r and t of a [Polar](#) vector

**4.8.2.11 double MathHelp::mag ( Pos *p* ) [inline]**

Magnitude Calculates the Magnitude of a vector, same as r in a [Polar](#) vector

**Parameters**

<i>p</i>	the <a href="#">Pos</a> of the vector
----------	---------------------------------------

**Returns**

A double containing the magnitude of the vector

**4.8.2.12 Pos MathHelp::norm ( Pos *p* ) [inline]**

A normalizer

**Parameters**

<i>p</i>	the vector to find the normal of
----------	----------------------------------

**Returns**

a [Pos](#) of the unit vector of p

**4.8.2.13 Pos MathHelp::norm ( double *dist*, Pos *a* ) [inline]**

A normalizer

**Parameters**

<i>dist</i>	the magnitude of the vector
<i>a</i>	the vector to be normalized

**Returns**

a [Pos](#) of the unit vector of p

**4.8.2.14 Pos MathHelp::vAdd ( Pos *p1*, Pos *p2* ) [inline]**

Vector Addition

**Parameters**

<i>p1</i>	first position
<i>p2</i>	second position

**Returns**

New position with the sum of the two arguments

**4.8.2.15** `Pos MathHelp::vDiv ( Pos p, double n )` `[inline]`

Divide vector by scalar

**Parameters**

<i>p</i>	the vector
<i>n</i>	the scalar

**Returns**

A [Pos](#) vector divided by a scalar value

**4.8.2.16** `Pos MathHelp::vMul ( Pos p, double n )` `[inline]`

Multiply vector by scalar

**Parameters**

<i>p</i>	the vector
<i>n</i>	the scalar

**Returns**

A [Pos](#) vector multiplied by a scalar value

**4.8.2.17** `Pos MathHelp::vSub ( Pos p2, Pos p1 )` `[inline]`

Vector Subtraction

**Parameters**

<i>p2</i>	final position
<i>p1</i>	initial position

**Returns**

new [Pos](#) with the difference between *p2* and *p1*

The documentation for this class was generated from the following file:

- [MathHelp.java](#)

## 4.9 Memory Class Reference

**Public Member Functions**

- [Memory](#) ()

- void [setField](#) (String [side](#))
- [ObjInfo](#) [getObj](#) (int [i](#))
- int [getObjMemorySize](#) ()
- boolean [isObjVisible](#) (String [name](#))
- [ObjBall](#) [getBall](#) ()
- [ObjFlag](#) [getFlag](#) (String [name](#))
- [ObjGoal](#) [getOppGoal](#) ()
- [Pos](#) [getOppGoalPos](#) ()
- [ObjGoal](#) [getOwnGoal](#) ()
- [Pos](#) [getOwnGoalPos](#) ()
- [ObjPlayer](#) [getPlayer](#) ()
- [ObjLine](#) [getLine](#) ()
- boolean [timeCheck](#) (int [t](#))
- ArrayList< [ObjPlayer](#) > [getPlayers](#) ()
- [ObjLine](#) [getClosestLine](#) ()
- double [getDirection](#) ()
- void [setLocation](#) (double [x](#), double [y](#))
- [ObjFlag](#) [getClosestFlag](#) ()
- [ObjFlag](#) [getClosestBoundary](#) ()
- [ObjFlag](#) [getClosestPenaltyFlag](#) ()
- [Pos](#) [getFlagPos](#) (String [flagName](#))
- [Pos](#) [getPosition](#) ()
- double [getNullGoalAngle](#) ()
- double [getStamina](#) ()
- double [getRecovery](#) ()
- double [getEffort](#) ()
- double [getAmountOfSpeed](#) ()
- double [getDirectionOfSpeed](#) ()
- double [getHeadDirection](#) ()
- String [getPlayMode](#) ()

### Public Attributes

- [MathHelp](#) **m** = new [MathHelp](#)()
- [Field](#) **f**
- [Pos](#) **home**
- [ObjMemory](#) **ObjMem**
- [SenseMemory](#) **SenMem**
- String **playMode**
- String **oppSide**
- String **side**
- int **uNum**
- [Pos](#) **oppGoal**

## 4.9.1 Constructor & Destructor Documentation

### 4.9.1.1 `Memory::Memory ( )` [inline]

The default constructor for the [Memory](#).

This creates new, empty ArrayList for the [ObjMemory](#) and [SenseMemory](#), initiates the time at 0 for both, and creates an [ObjMemory](#) and [SenseMemory](#) with the new ArrayLists and time as parameters.

## 4.9.2 Member Function Documentation

### 4.9.2.1 `double Memory::getAmountOfSpeed ( )` [inline]

The getter for the magnitude of the Player's velocity

### 4.9.2.2 `ObjBall Memory::getBall ( )` [inline]

The Ball Getter

#### Precondition

Make sure you either check visibility first

#### Postcondition

If the ball is in the [Memory](#), it will be returned. Otherwise a Null [ObjBall](#) will be sent.

#### Returns

[ObjBall](#) containing the ball

### 4.9.2.3 `ObjFlag Memory::getClosestBoundary ( )` [inline]

Finds [ObjFlag](#) of the closest boundary flag in players sight.

#### Returns

closest boundary

### 4.9.2.4 `ObjFlag Memory::getClosestFlag ( )` [inline]

Finds the closest flag in your sight

#### Returns

[ObjFlag](#) containing closest flag

#### 4.9.2.5 **ObjLine Memory::getClosestLine** ( ) [inline]

This gets the closest line in your sight

##### **Returns**

line

#### 4.9.2.6 **ObjFlag Memory::getClosestPenaltyFlag** ( ) [inline]

Finds [ObjFlag](#) of the closest penalty box flag in players sight.

##### **Returns**

closest penalty box flag

#### 4.9.2.7 **double Memory::getDirection** ( ) [inline]

Calculates the direction your facing from the closest line in your vision. The direction returned from a line is the angle made by your line of sight and the point that it crosses the line. This will allow the facing direction to be calculated with some arithmetic.

##### **Returns**

the absolute direction you're facing

#### 4.9.2.8 **double Memory::getDirectionOfSpeed** ( ) [inline]

The getter for the direction of the Player's velocity

#### 4.9.2.9 **double Memory::getEffort** ( ) [inline]

The getter for the Player's stamina effort

#### 4.9.2.10 **ObjFlag Memory::getFlag** ( *String name* ) [inline]

The Flag Getter

If you're looking for a specific flag, this is your guy. You need to pass in the *FlagName* (i.e. flb30) into it, and out pops the [ObjFlag](#) with that *FlagName* attached to it.

##### **Precondition**

Make sure you either check visibility first

**Postcondition**

If the flag is in the [Memory](#), it will be returned. Otherwise a Null [ObjFlag](#) will be sent.

**Returns**

[ObjFlag](#) containing the flag with specified name

**4.9.2.11 Pos Memory::getFlagPos ( String *flagName* ) [inline]**

Returns the [Pos](#) of the coordinate of any flag on the field by name

**Parameters**

<i>flagName</i>	
-----------------	--

**Returns**

[Pos](#) with coordinate of flag

**4.9.2.12 double Memory::getHeadDirection ( ) [inline]**

The getter for the angle of the Player's head relative to the orientation of the Player's positive y-axis (up-field)

**4.9.2.13 ObjLine Memory::getLine ( ) [inline]**

The Line getter This will get the [ObjLine](#) of the first line you see.

**Returns**

[ObjLine](#)

**4.9.2.14 double Memory::getNullGoalAngle ( ) [inline]**

Calculates the angle of goal you're trying to score on when the goal is not in your sight. This allows the player to kick or dribble to the goal, even when it's information isn't available.

**Returns**

double containing the angle of the goal

#### 4.9.2.15 **ObjInfo** Memory::getObj ( int *i* ) [inline]

The **ObjInfo** getter

This fetches the **ObjInfo** at index *i* of the ArrayList ObjArray in **ObjMemory**, and returns it as an **ObjInfo**.

##### Parameters

<i>i</i>	the index number of the location of the desired <b>ObjInfo</b> in ObjArray
----------	--

##### Precondition

An index needs to be supplied when calling this

##### Postcondition

A basic **ObjInfo** will be given.

##### Returns

**ObjInfo** the **ObjInfo** at location *i* of the ObjArray

#### 4.9.2.16 **int** Memory::getObjMemorySize ( ) [inline]

The **ObjMemory** size

A getter to quickly retrieve the number of **ObjInfo** in **ObjMemory**

##### Returns

size of **ObjMemory**

#### 4.9.2.17 **ObjGoal** Memory::getOppGoal ( ) [inline]

The Goal Opponent Getter

This will get the Opponent's **ObjGoal** if it's in your field of vision.

##### Postcondition

If you're facing the opponenet's goal, an **ObjGoal** with it's information will be returned. Otherwise a null **ObjGoal** will be sent

##### Returns

**ObjGoal** containing the goal if it's in your vision, null if not



**4.9.2.18 Pos Memory::getOppGoalPos ( ) [inline]**

This returns the [Pos](#) with the coordinate to the goal you're trying to score on.

**Returns**

the [Pos](#) in the [Field](#) of your opponent's goal

**4.9.2.19 ObjGoal Memory::getOwnGoal ( ) [inline]**

The Goal Own Getter

This will get your own [ObjGoal](#) if it's in your field of vision.

**Postcondition**

If you're facing your goal, an [ObjGoal](#) with it's information will be returned. Otherwise a null [ObjGoal](#) will be sent

**Returns**

[ObjGoal](#) containing the goal if it's in your vision, null if not

**4.9.2.20 Pos Memory::getOwnGoalPos ( ) [inline]**

This returns the [Pos](#) with the coordinate to the goal you're trying to guard.

**Returns**

the [Pos](#) in the [Field](#) of your goal

**4.9.2.21 ObjPlayer Memory::getPlayer ( ) [inline]**

The [Player](#) Getter

This will get the [ObjPlayer](#) of the first player you see.

**Returns**

[ObjPlayer](#)

**4.9.2.22 ArrayList<ObjPlayer> Memory::getPlayers ( ) [inline]**

Gets an ArrayList with all of the Players in your sight

**Returns**

players

**4.9.2.23 String Memory::getPlayMode ( ) [inline]**

The getter for the game's current play mode

**4.9.2.24 Pos Memory::getPosition ( ) [inline]**

This finds the absolute position of a player using vector arithmetic and trigonometry and the closest flag to the player and the facing direction found from the closest line.

**Returns**

[Pos](#) containing the coordinate on the field of the player's absolute position

**4.9.2.25 double Memory::getRecovery ( ) [inline]**

The getter for the Player's stamina recovery

**4.9.2.26 double Memory::getStamina ( ) [inline]**

The getter for the Player's stamina

**4.9.2.27 boolean Memory::isObjVisible ( String *name* ) [inline]**

Is this [ObjInfo](#) visible?

**Parameters**

<i>name</i>	the ObjName of the <a href="#">ObjInfo</a> we're detecting visibility of
-------------	--

**Returns**

true if the ball is in the [ObjMemory](#), false if it is not or if the the [ObjMemory](#) is empty

**4.9.2.28 void Memory::setField ( String *side* ) [inline]**

This sets the orientation of the [Field](#) positions depending on side the player starts on.

**Parameters**

<i>side</i>	
-------------	--

**Precondition**

The side String should not be null

**Postcondition**

The [Field](#) orientation will be set

**4.9.2.29 void Memory::setLocation ( double x, double y ) [inline]**

Sets the [Pos](#) of the originating point.

**Parameters**

$x$	
$y$	

**4.9.2.30 boolean Memory::timeCheck ( int t ) [inline]**

This will test a players local time against the ObjMemory's time. This can be used to ensure that more than one action will not be attempted during a single simulation step.

**Parameters**

$t$	the Player's local time
-----	-------------------------

**Precondition**

A player's local time must be initialized and passed in

**Postcondition**

The player's local time needs to be set to the Memory's time after a true is returned.

**Returns**

true if the newly parsed Memory's time is greater than the players local time. False if the memory time is  $\leq$  the local time.

**4.9.3 Member Data Documentation****4.9.3.1 ObjMemory Memory::ObjMem**

The memory that stores all parsed [ObjInfo](#)

**4.9.3.2 Pos Memory::oppGoal**

The [Pos](#) of the coordinates of the opponents goal

**4.9.3.3 String Memory::oppSide**

The string of the opponents side

#### 4.9.3.4 String Memory::playMode

The play mode as told by the referee

#### 4.9.3.5 SenseMemory Memory::SenMem

The memory that stores all parsed SenseInfo

#### 4.9.3.6 String Memory::side

The String of the player's side

#### 4.9.3.7 int Memory::uNum

The player's uniform number

The documentation for this class was generated from the following file:

- [Memory.java](#)

## 4.10 Mode Class Reference

### Public Member Functions

- [Mode](#) (String modename, double timeinmode)
- String [getModename](#) ()
- void [setModename](#) (String modename)
- double [getTimeinmode](#) ()
- void [setTimeinmode](#) (double timeinmode)

#### 4.10.1 Detailed Description

The [Mode](#) class is a basic data structure to store the parameters for the player modes.

#### 4.10.2 Constructor & Destructor Documentation

##### 4.10.2.1 Mode::Mode ( String *modename*, double *timeinmode* ) `[inline]`

#### Parameters

<i>modename</i>	
<i>timeinmode</i>	

### 4.10.3 Member Function Documentation

#### 4.10.3.1 String Mode::getModename ( ) [inline]

##### Returns

the modename

#### 4.10.3.2 double Mode::getTimeinmode ( ) [inline]

##### Returns

the timeinmode

#### 4.10.3.3 void Mode::setModename ( String *modename* ) [inline]

##### Parameters

<i>modename</i>	the modename to set
-----------------	---------------------

#### 4.10.3.4 void Mode::setTimeinmode ( double *timeinmode* ) [inline]

##### Parameters

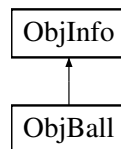
<i>timeinmode</i>	the timeinmode to set
-------------------	-----------------------

The documentation for this class was generated from the following file:

- [Mode.java](#)

## 4.11 ObjBall Class Reference

Inheritance diagram for ObjBall:



### 4.11.1 Detailed Description

container for the ball [ObjInfo](#),

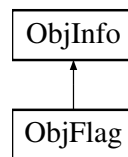
container for the flag [ObjInfo](#),

The documentation for this class was generated from the following file:

- [ObjInfo.java](#)

## 4.12 ObjFlag Class Reference

Inheritance diagram for ObjFlag:



### Public Member Functions

- [ObjFlag](#) (String name)
- String [getFlagType](#) ()
- void [setFlagType](#) (String flagType)
- String [getFlagName](#) ()
- void [setFlagName](#) (String name)
- String [getX\\_pos](#) ()
- void [setX\\_pos](#) (String x\_pos)
- String [getY\\_pos](#) ()
- void [setY\\_pos](#) (String y\_pos)
- String [getYard](#) ()
- void [setYard](#) (String yard)

### 4.12.1 Constructor & Destructor Documentation

#### 4.12.1.1 [ObjFlag::ObjFlag \( String name \)](#) [inline]

Constructor of flag with flag name

### 4.12.2 Member Function Documentation

#### 4.12.2.1 [String ObjFlag::getFlagName \( \)](#) [inline]

The Flag Name getter

#### Returns

The name of the flag, as given by the server but with no spaces (e.g. flt20 for boundary flag left, top, 20 yard line)

#### 4.12.2.2 String ObjFlag::getFlagType ( ) [inline]

The Flag Type getter

##### Returns

The type of flag depending on it's location: "b" - outer boundary "g" - goal post  
"p" - penalty box "c" - center of field "l" - border line

#### 4.12.2.3 String ObjFlag::getX\_pos ( ) [inline]

The X position getter

##### Returns

Either "l" for left, "r" for right, or "c" for center

#### 4.12.2.4 String ObjFlag::getY\_pos ( ) [inline]

The Y position getter

##### Returns

Either "t" for top, "b" for bottom, or "c" for center

#### 4.12.2.5 String ObjFlag::getYard ( ) [inline]

The yard getter

##### Returns

the yard is a String of a number for boundaries

#### 4.12.2.6 void ObjFlag::setFlagName ( String *name* ) [inline]

The Flag Name setter

#### 4.12.2.7 void ObjFlag::setFlagType ( String *flagType* ) [inline]

The Flag Type setter

#### 4.12.2.8 void ObjFlag::setX\_pos ( String *x\_pos* ) [inline]

The X position setter

**4.12.2.9** void `ObjFlag::setY_pos ( String y_pos )` `[inline]`

The Y position setter

**4.12.2.10** void `ObjFlag::setYard ( String yard )` `[inline]`

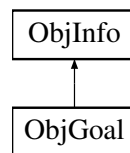
The yard setter

The documentation for this class was generated from the following file:

- [ObjInfo.java](#)

## 4.13 ObjGoal Class Reference

Inheritance diagram for ObjGoal:



### 4.13.1 Detailed Description

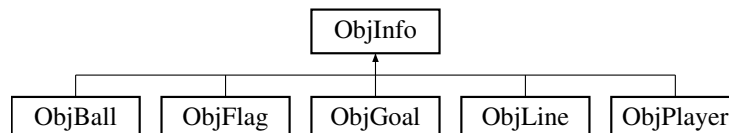
container for the goal [ObjInfo](#),

The documentation for this class was generated from the following file:

- [ObjInfo.java](#)

## 4.14 ObjInfo Class Reference

Inheritance diagram for ObjInfo:



### Public Member Functions

- [ObjInfo \(\)](#)



- [ObjInfo](#) (String name)
- String [getObjName](#) ()
- void [setObjName](#) (String name)
- String [getSide](#) ()
- void [setSide](#) (String objSide)
- double [getDistance](#) ()
- void [setDistance](#) (double distance)
- double [getDirection](#) ()
- void [setDirection](#) (double direction)
- double [getDistChng](#) ()
- void [setDistChng](#) (double distChng)
- double [getDirChng](#) ()
- void [setDirChng](#) (double dirChng)

#### 4.14.1 Detailed Description

A container for items in the Player's vision

#### 4.14.2 Constructor & Destructor Documentation

##### 4.14.2.1 [ObjInfo::ObjInfo](#) ( ) `[inline]`

The Default constructor

##### 4.14.2.2 [ObjInfo::ObjInfo](#) ( String *name* ) `[inline]`

The [ObjInfo](#) constructor

This initializes all the variables to 0.0 and sets the name

##### Parameters

<i>name</i>	The type of <a href="#">ObjInfo</a> , either ball, player, goal, line, or flag
-------------	--

#### 4.14.3 Member Function Documentation

##### 4.14.3.1 [double ObjInfo::getDirChng](#) ( ) `[inline]`

The direction change getter

##### Returns

the approximate direction change (direction of velocity) of [ObjInfo](#)

**4.14.3.2 double ObjInfo::getDirection ( ) [inline]**

The direction getter

**Returns**

the approximate direction of [ObjInfo](#)

**4.14.3.3 double ObjInfo::getDistance ( ) [inline]**

The distance getter

**Returns**

the approximate distance to the object

**4.14.3.4 double ObjInfo::getDistChng ( ) [inline]**

The distance change getter

**Returns**

the approximate distance change (magnitude of velocity) of [ObjInfo](#)

**4.14.3.5 String ObjInfo::getObjName ( ) [inline]**

The ObjName getter

**4.14.3.6 String ObjInfo::getSide ( ) [inline]**

The side getter

**4.14.3.7 void ObjInfo::setDirChng ( double *dirChng* ) [inline]**

The distance change setter

**4.14.3.8 void ObjInfo::setDirection ( double *direction* ) [inline]**

The direction setter

**4.14.3.9 void ObjInfo::setDistance ( double *distance* ) [inline]**

The distance setter

4.14.3.10 void ObjInfo::setDistChng ( double *distChng* ) [inline]

The distance change setter

4.14.3.11 void ObjInfo::setObjName ( String *name* ) [inline]

The ObjName setter

4.14.3.12 void ObjInfo::setSide ( String *objSide* ) [inline]

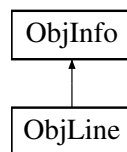
The side setter

The documentation for this class was generated from the following file:

- [ObjInfo.java](#)

## 4.15 ObjLine Class Reference

Inheritance diagram for ObjLine:



### 4.15.1 Detailed Description

container for line [ObjInfo](#)

The documentation for this class was generated from the following file:

- [ObjInfo.java](#)

## 4.16 ObjMemory Class Reference

### Public Member Functions

- [ObjMemory](#) ()
- [ObjMemory](#) (ArrayList< [ObjInfo](#) > ObjArray, int t)
- void [addInfo](#) ([ObjInfo](#) newInfo)
- int [getTime](#) ()
- void [setTime](#) (int t)
- int [getSize](#) ()

- [ObjInfo](#) `getObj` (int index)
- [ObjInfo](#) `getObj` (String name)

### Public Attributes

- `ArrayList< ObjInfo > ObjArray`

#### 4.16.1 Detailed Description

The [ObjMemory](#) saves all the [ObjInfo](#) (and it's children) objects from a parse into `ArrayList` along with the time parsed.

#### 4.16.2 Constructor & Destructor Documentation

##### 4.16.2.1 `ObjMemory::ObjMemory ( )` `[inline]`

Default constructor

This initializes the time to 0

##### 4.16.2.2 `ObjMemory::ObjMemory ( ArrayList< ObjInfo > ObjArray, int t )` `[inline]`

[ObjMemory](#) constructor

#### Parameters

<i>ObjArray</i>	the ArrayList containing all the <a href="#">ObjInfos</a> from the server's parsed (see) message
<i>t</i>	the time parsed from the server's (see) message

#### Precondition

This should only be called inside of the parser. It's merely a way to store [ObjInfos](#) from the (see) message into the greater [Memory](#) class

#### Postcondition

A new [ObjMemory](#) containing the list of visible [ObjInfos](#) and the most recent time will be availbe to add to the [Memory](#)

#### 4.16.3 Member Function Documentation

##### 4.16.3.1 `void ObjMemory::addInfo ( ObjInfo newInfo )` `[inline]`

A method to add new [ObjInfo](#) to the [ObjMemory](#)

#### Parameters

<i>newInfo</i>	the <a href="#">ObjInfo</a> to add to the ObjMemory's ArrayList
----------------	---

**Precondition**

A non-null [ObjInfo](#) will be passed into the method

**Postcondition**

The newInfo will be added to the ObjArray

**4.16.3.2 [ObjInfo](#) ObjMemory::getObj ( int *index* ) [inline]**

An accessor of individual [ObjInfo](#)

**Parameters**

<i>index</i>	the index of the <a href="#">ObjInfo</a> to retrieve
--------------	--

**Precondition**

The ObjArray should have at least one [ObjInfo](#) in it

**Postcondition**

The [ObjInfo](#) at the given index will be returned, this is a good way to traverse the ObjInfos visible to you

**4.16.3.3 [ObjInfo](#) ObjMemory::getObj ( String *name* ) [inline]**

A method to get an [ObjInfo](#) by name

**Parameters**

<i>name</i>	the ObjName of the <a href="#">ObjInfo</a> searched for (e.g. "ball")
-------------	---

**Precondition**

The [ObjInfo](#) should be checked for visibility first, otherwise you run the risk of getting an empty [ObjInfo](#)

**Postcondition**

The first [ObjInfo](#) with the name will be returned. Remember, this won't return all the ObjInfos of an ObjName, if there are multiple.

**4.16.3.4 int ObjMemory::getSize ( ) [inline]**

Returns the size of the ObjArray

#### 4.16.3.5 int ObjMemory::getTime ( ) [inline]

A method to access the time the message was parsed, provided by the server's (see) message

#### 4.16.3.6 void ObjMemory::setTime ( int t ) [inline]

The time setter

##### Parameters

<i>t</i>	the time integer from the server's latest (see) parse
----------	---

##### Postcondition

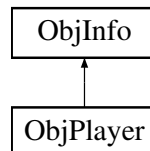
the time will be set and ready to access

The documentation for this class was generated from the following file:

- [ObjMemory.java](#)

## 4.17 ObjPlayer Class Reference

Inheritance diagram for ObjPlayer:



### Public Member Functions

- String [getTeam](#) ()
- void [setTeam](#) (String team)
- int [getuNum](#) ()
- void [setuNum](#) (int uNum)
- boolean [isGoalie](#) ()
- void [setGoalie](#) (boolean goalie)
- double [getHeadDir](#) ()
- void [setHeadDir](#) (double headDir)
- double [getBodyDir](#) ()
- void [setBodyDir](#) (double bodyDir)

### 4.17.1 Detailed Description

container for player [ObjInfo](#)

### 4.17.2 Member Function Documentation

#### 4.17.2.1 double ObjPlayer::getBodyDir ( ) [inline]

A getter for the player's body direction

##### Returns

a double of the angle, in degrees, of the direction of the player's body relative to your own. The angle is 0 if their bodies are both facing each other.

#### 4.17.2.2 double ObjPlayer::getHeadDir ( ) [inline]

A getter for the player's head direction

##### Returns

a double of the angle, in degrees, of the direction of the player's head relative to your own. The angle is 0 if they are both facing each other.

#### 4.17.2.3 String ObjPlayer::getTeam ( ) [inline]

The Team Name getter

##### Returns

the name of the team the player is on, if they're close enough to see the team

#### 4.17.2.4 int ObjPlayer::getuNum ( ) [inline]

The Uniform Number getter

##### Returns

the Uniform Number on the player's shirt, if they're close enough to see it

#### 4.17.2.5 boolean ObjPlayer::isGoalie ( ) [inline]

A check to see if the player is a goalie or field player

##### Returns

true if the player is the goalie, false if s/he is not

**4.17.2.6** void ObjPlayer::setBodyDir ( double *bodyDir* ) [inline]

The body direction setter

**4.17.2.7** void ObjPlayer::setGoalie ( boolean *goalie* ) [inline]

The goalie check setter

**4.17.2.8** void ObjPlayer::setHeadDir ( double *headDir* ) [inline]

The head direction setter

**4.17.2.9** void ObjPlayer::setTeam ( String *team* ) [inline]

The Team Name setter

**4.17.2.10** void ObjPlayer::setuNum ( int *uNum* ) [inline]

The Uniform Number getter

The documentation for this class was generated from the following file:

- [ObjInfo.java](#)

## 4.18 Parser Class Reference

### Public Member Functions

- [Parser](#) ()
- void [initParse](#) (String inputPacket, [Memory](#) mem)
- void [Parse](#) (String inputPacket, [Memory](#) InfoMem)

### Public Attributes

- String [input](#)

#### 4.18.1 Detailed Description

This class takes in the the messages sent by the parser and parses them into information that can be stored in [Memory](#) and used by Players.



## 4.18.2 Constructor & Destructor Documentation

### 4.18.2.1 Parser::Parser ( ) [inline]

Default constructor

## 4.18.3 Member Function Documentation

### 4.18.3.1 void Parser::initParse ( String *inputPacket*, Memory *mem* ) [inline]

This parses the (init) message, the first message sent by the server, directly after a new [Player](#) is initialized.

#### Parameters

<i>inputPacket</i>	The init message from the server
<i>mem</i>	the player's memory

#### Precondition

A memory must be created for the information to be stored in, and this must be called directly after an (init) is sent to the server.

#### Postcondition

Vital information about the [Player](#) will be saved, such as the side of the field the player starts on, the Player's uniform number and the play mode, which is "before\_kickoff."

### 4.18.3.2 void Parser::Parse ( String *inputPacket*, Memory *InfoMem* ) [inline]

The actual message Parsing method

#### Parameters

<i>inputPacket</i>	the incoming String message from the server
<i>InfoMem</i>	the <a href="#">Memory</a> to store all the information in

#### Precondition

A [Memory](#) must be created and passed in, along with the message from the server

#### Postcondition

The message will be parsed and stored either as SenseInfos from the (sense\_body) message, or ObjInfos from the (see) message, or the playMode from the referee (hear) message

#### 4.18.4 Member Data Documentation

##### 4.18.4.1 String Parser::input

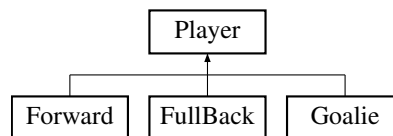
The String of the incoming message

The documentation for this class was generated from the following file:

- [Parser.java](#)

### 4.19 Player Class Reference

Inheritance diagram for Player:



#### Public Member Functions

- **Player** (String team)
- **Player** (RoboClient rc, Memory m, ObjInfo i, Parser p, Brain b, int time)
- void **setBrain** (Brain b)
- **Action** **getAction** ()
- void **setAction** (Action a)
- **RoboClient** **getRoboClient** ()
- void **setRoboclient** (RoboClient rc)
- **Memory** **getMem** ()
- void **setMem** (Memory m)
- **ObjInfo** **getObjInfo** ()
- void **setObjInfo** (ObjInfo i)
- **Parser** **getParser** ()
- void **setParser** (Parser p)
- int **getTime** ()
- void **setTime** (int time)
- double **getDirection** ()
- **Pos** **getPosition** ()
- void **initPlayer** () throws SocketException, UnknownHostException
- void **receiveInput** () throws InterruptedException
- void **move** (double x, double y) throws UnknownHostException, InterruptedException
- void **kick** (double power, double dir) throws UnknownHostException, InterruptedException

- void [dash](#) (double power) throws Exception
- void [turn](#) (double moment) throws UnknownHostException, InterruptedException
- void [say](#) (String message) throws UnknownHostException, InterruptedException
- void **markOpponent** (String team, String number)
- void **runDefense** () throws UnknownHostException, InterruptedException
- [ObjPlayer](#) **closestOpponent** () throws UnknownHostException, InterruptedException
- void [run](#) ()

### Public Attributes

- boolean **wait** = true

### Protected Attributes

- [RoboClient](#) **rc** = new [RoboClient](#)()

#### 4.19.1 Detailed Description

The [Player](#) class defines all objects and methods used for the [Player](#) within the RoboCup match. The [Player](#) establishes a connection to the server, initializes itself on the team, and performs all actions related to a RoboCup soccer player such as (but not limited to) kicking, dashing, dribbling, passing and scoring. The [Player](#) class has a [Memory](#) for storing the current RoboCup worldstate. It reacts to stimuli based on strategies provided by the [Brain](#) (TBD).

#### 4.19.2 Constructor & Destructor Documentation

4.19.2.1 **Player::Player ( [RoboClient](#) *rc*, [Memory](#) *m*, [ObjInfo](#) *i*, [Parser](#) *p*, [Brain](#) *b*, int *time* )** [inline]

##### Parameters

<i>rc</i>	
<i>m</i>	
<i>i</i>	
<i>p</i>	
<i>b</i>	
<i>time</i>	

### 4.19.3 Member Function Documentation

#### 4.19.3.1 **ObjPlayer** Player::closestOpponent ( ) throws UnknownHostException, InterruptedException [inline]

Returns the closest opponent to the player

##### Precondition

Players are in sight of the goalie.

##### Postcondition

The closest opponent to the player has been determined.

##### Returns

[ObjPlayer](#)

##### Exceptions

<i>InterruptedException</i>	
<i>UnknownHostException</i>	

#### 4.19.3.2 void Player::dash ( double power ) throws Exception [inline]

Causes [Player](#) to dash.

##### Parameters

<i>power</i>	The power with which to dash in the form of a decimal value.
--------------	--

##### Exceptions

<i>Exception</i>	
------------------	--

##### Precondition

Play mode is play\_on.

##### Postcondition

The player has dashed at the given power.

#### 4.19.3.3 double Player::getDirection ( ) [inline]

Returns the direction of the player

**4.19.3.4** `Memory Player::getMem ( ) [inline]`**Returns**

The [Memory](#) for this [Player](#).

**4.19.3.5** `ObjInfo Player::getObjInfo ( ) [inline]`**Returns**

The [ObjInfo](#) for this [Player](#).

**4.19.3.6** `Parser Player::getParser ( ) [inline]`**Returns**

The [Parser](#) for this [Player](#).

**4.19.3.7** `Pos Player::getPosition ( ) [inline]`

Returns the current absolute coordinates of the player.

**Returns**

[Pos](#)

**4.19.3.8** `RoboClient Player::getRoboClient ( ) [inline]`**Returns**

The [RoboClient](#) object for this [Player](#).

**4.19.3.9** `int Player::getTime ( ) [inline]`

Returns the current player time.

**Returns**

the time

**4.19.3.10** `void Player::initPlayer ( ) throws SocketException, UnknownHostException [inline]`

Initializes the [Player](#) with the RoboCup server.

**Precondition**

A RoboCup server is available.

**Postcondition**

The [Player](#) has been initialized to the correct team.

**4.19.3.11** `void Player::kick ( double power, double dir ) throws UnknownHostException, InterruptedException` `[inline]`

Causes [Player](#) to kick the ball.

**Parameters**

<i>dir</i>	The direction in which to kick the ball in the form of a decimal value. representing the angle in degrees in relation go the player.
<i>power</i>	The power with which to kick the ball in the form of a decimal value.

**Exceptions**

<i>InterruptedException</i>	
-----------------------------	--

**Precondition**

Playmode is play\_on, ball is in kickable range.

**Postcondition**

The ball has been kicked in the specified direction and power.

**4.19.3.12** `void Player::move ( double x, double y ) throws UnknownHostException, InterruptedException` `[inline]`

Teleports the [Player](#) to the specified coordinates.

**Parameters**

<i>x</i>	x-coordinate of the point to move the player to.
<i>y</i>	y-coordinate of the point to move the player to.

**Exceptions**

<i>InterruptedException</i>	
-----------------------------	--

**Precondition**

Playmode is before-kickoff, goal-scored, free-kick.

**Postcondition**

The [Player](#) has been moved to the correct position.

**4.19.3.13** void Player::receiveInput ( ) throws InterruptedException [inline]

Receives worldstate data from the RoboCup server.

**Precondition**

A RoboCup server is available.

**Postcondition**

The current worldstate has been stored in the [Memory](#).

**4.19.3.14** void Player::run ( ) [inline]

The [Player](#) thread run method. It makes decisions for the player.

**Postcondition**

[Player](#) will act on decisions made.

Reimplemented in [Goalie](#).

**4.19.3.15** void Player::say ( String *message* ) throws UnknownHostException, InterruptedException [inline]

Causes [Player](#) to say the given message. It has a limitation of 512 characters by default.

**Parameters**

<i>message</i>	The string to be spoken by the player.
----------------	--

**Exceptions**

<i>InterruptedException</i>	
-----------------------------	--

**Precondition**

None

**Postcondition**

The player has spoken the message.

**4.19.3.16** void Player::setBrain ( Brain *b* ) [inline]**Parameters**

<i>b</i>	the b to set
----------	--------------

**4.19.3.17 void Player::setMem ( Memory *m* )** [inline]**Parameters**

<i>m</i>	The <a href="#">Memory</a> to set.
----------	------------------------------------

**4.19.3.18 void Player::setObjInfo ( ObjInfo *i* )** [inline]**Parameters**

<i>i</i>	The <a href="#">ObjInfo</a> to set.
----------	-------------------------------------

**4.19.3.19 void Player::setParser ( Parser *p* )** [inline]

Sets the parser for the player.

**Parameters**

<i>p</i>	The <a href="#">Parser</a> to set.
----------	------------------------------------

**4.19.3.20 void Player::setRoboclient ( RoboClient *rc* )** [inline]**Parameters**

<i>rc</i>	The <a href="#">RoboClient</a> to set.
-----------	--

**4.19.3.21 void Player::setTime ( int *time* )** [inline]

Sets the current time for the player.

**Parameters**

<i>time</i>	the time to set
-------------	-----------------

**4.19.3.22 void Player::turn ( double *moment* )** throws [UnknownHostException](#), [InterruptedException](#) [inline]

Causes [Player](#) to turn according to a specified turn moment.

**Parameters**

<i>moment</i>	The turn angle in degrees.
---------------	----------------------------

**Exceptions**

<i>InterruptedException</i>	
-----------------------------	--



**Precondition**

Playmode is play\_on, ball is in kickable range.

**Postcondition**

The ball has been kicked in the specified direction and power.

The documentation for this class was generated from the following file:

- [Player.java](#)

## 4.20 Polar Class Reference

**Public Member Functions**

- [Polar](#) ()
- [Polar](#) (double r, double t)

**Public Attributes**

- double **r**
- double **t**

### 4.20.1 Detailed Description

A container for polar coordinates. It holds distance (r) and direction (t) of an object with respect to the player.

**Author**

Grant Hays

**Date**

10/14/11

**Version**

1

### 4.20.2 Constructor & Destructor Documentation

#### 4.20.2.1 [Polar::Polar \( \)](#) [[inline](#)]

Default constructor

**Postcondition**

initializes distance and angle to 0.0

#### 4.20.2.2 Polar::Polar ( double *r*, double *t* ) [inline]

Constructor with parameters

##### Parameters

<i>r</i>	The length of the distance to the object
<i>t</i>	The angle of the object from the players line of sight

The documentation for this class was generated from the following file:

- Polar.java

## 4.21 Pos Class Reference

### Public Member Functions

- [Pos](#) ()
- [Pos](#) (String name, double x, double y)
- [Pos](#) (double x, double y)

### Public Attributes

- String **name**
- double **x**
- double **y**

#### 4.21.1 Detailed Description

This class holds the information for Cartesian coordinate versions of positions of players and objects

#### 4.21.2 Constructor & Destructor Documentation

##### 4.21.2.1 Pos::Pos ( ) [inline]

Default constructor

##### Postcondition

initializes x and y to 0 and name to space, so as not to have a pointer error

**4.21.2.2 Pos::Pos ( String name, double x, double y ) [inline]**

Constructor with name

This is a constructor for coordinates that are given a name. It is mostly used for the positions of the flags in the [Field](#) class

**Parameters**

<i>name</i>	The name associated with the <a href="#">Pos</a> , for easier searching
<i>x</i>	x-coordinate
<i>y</i>	y-coordinate

**4.21.2.3 Pos::Pos ( double x, double y ) [inline]**

Constructor with no name

This is a constructor for positions that aren't given a name. Used for positions that change often.

**Parameters**

<i>x</i>	x-coordinate
<i>y</i>	y-coordinate

The documentation for this class was generated from the following file:

- [Pos.java](#)

**4.22 RoboClient Class Reference****Public Member Functions**

- [RoboClient](#) (int port)
- [RoboClient](#) (String team)
- String [getTeam](#) ()
- int [getPort](#) ()
- void [setTeam](#) (String team)
- void [send](#) (String message) throws UnknownHostException
- String [receive](#) ()
- void [init](#) ([Parser](#) p, [Memory](#) m) throws UnknownHostException
- void [initGoalie](#) ([Parser](#) p, [Memory](#) m) throws UnknownHostException
- void [dash](#) (double power) throws Exception
- void [kick](#) (double power, double dir) throws UnknownHostException
- void [turn](#) (double moment) throws UnknownHostException
- void [move](#) (double x, double y) throws UnknownHostException
- void [catchball](#) (double d) throws UnknownHostException
- void [say](#) (String message) throws UnknownHostException

## Public Attributes

- DatagramSocket **dsock**

## Package Attributes

- String **reply**

### 4.22.1 Detailed Description

The [RoboClient](#) class operates as a client for the RoboCup session. It is mainly designed to be used by the [Player](#) class to handle all client-server communication. The connection protocol is UDP.

### 4.22.2 Constructor & Destructor Documentation

#### 4.22.2.1 RoboClient::RoboClient ( int *port* ) [inline]

##### Parameters

<i>port</i>	
-------------	--

#### 4.22.2.2 RoboClient::RoboClient ( String *team* ) [inline]

##### Parameters

<i>team</i>	
-------------	--

### 4.22.3 Member Function Documentation

#### 4.22.3.1 void RoboClient::catchball ( double *d* ) throws UnknownHostException [inline]

This function causes the active player to catch the ball. It can only be used by a [Goalie](#) type player.

##### Parameters

<i>d</i>	An integer value representing the direction from which to catch the ball.
----------	---

##### Precondition

Playmode is play\_on or goal\_kick, ball is in catchable area.

##### Postcondition

The player has caught the ball.

### Exceptions

<i>UnknownHostException</i>	
-----------------------------	--

#### 4.22.3.2 void RoboClient::dash ( double *power* ) throws Exception [inline]

This function sends the dash command to the server.

### Parameters

<i>power</i> ,:	a double representing the power of the dash.
-----------------	--

### Precondition

The RoboCup server is available, client has been initialized.

### Postcondition

The player has dashed according to the given power.

### Returns

None

#### 4.22.3.3 int RoboClient::getPort ( ) [inline]

### Returns

the port

#### 4.22.3.4 String RoboClient::getTeam ( ) [inline]

### Returns

the team

#### 4.22.3.5 void RoboClient::init ( Parser *p*, Memory *m* ) throws UnknownHostException [inline]

This function initializes the client with the RoboCup server.

### Precondition

The RoboCup server is hosting connections.

### Postcondition

The client has been initialized.

#### 4.22.3.6 void RoboClient::initGoalie ( Parser *p*, Memory *m* ) throws UnknownHostException [inline]

This function initializes the client as a goalie with the RoboCup server.

##### Parameters

<i>message,:</i>	none
------------------	------

##### Precondition

The RoboCup server is hosting connections.

##### Postcondition

The goalie has been initialized.

##### Returns

None

#### 4.22.3.7 void RoboClient::kick ( double *power*, double *dir* ) throws UnknownHostException [inline]

This function causes the active player to kick.

##### Parameters

<i>power,:</i>	a double representing the power of the kick.
<i>dir,:</i>	a double representing the direction of the kick.

##### Precondition

The RoboCup server is available, team has been initialized.

##### Postcondition

The player has kicked according to the given power and direction.

##### Returns

None

#### 4.22.3.8 void RoboClient::move ( double *x*, double *y* ) throws UnknownHostException [inline]

This function causes the active player to be teleported to a given set of coordinates within the soccer field.

##### Parameters

<i>x,:</i>	an integer value for the x-coordinate to move to.
<i>y,:</i>	an integer value for the y-coordinate to move to.

**Precondition**

The RoboCup server is available, team has been initialized, kickoff has not yet occurred.

**Postcondition**

The player has moved to the given coordinates.

**Returns**

None

**4.22.3.9 String RoboClient::receive ( ) [inline]**

This function receives a UDP packet from the RoboCup server, and converts it to a String.

**Precondition**

The RoboCup server is available.

**Postcondition**

The packet from the RoboCup server has been processed.

**Returns**

String

**4.22.3.10 void RoboClient::say ( String message ) throws UnknownHostException [inline]**

This function causes the active player to speak the given message.

**Parameters**

<i>message</i>	A string representing the message to be spoken by the player.
----------------	---

**Precondition**

None

**Postcondition**

The player has spoken the message.

**Exceptions**

<i>UnknownHostException</i>	
-----------------------------	--

#### 4.22.3.11 void RoboClient::send ( String *message* ) throws UnknownHostException [inline]

This function reads in a message string, and sends it to the RoboCup server. It primarily serves as a method to send commands to the server to control server and player actions.

##### Parameters

<i>message,:</i>	A String.
------------------	-----------

##### Precondition

message is a valid String value, the RoboCup server is available.

##### Postcondition

The message has been delivered to the RoboCup server.

##### Returns

None

#### 4.22.3.12 void RoboClient::setTeam ( String *team* ) [inline]

##### Parameters

<i>team</i>	the team to set
-------------	-----------------

#### 4.22.3.13 void RoboClient::turn ( double *moment* ) throws UnknownHostException [inline]

This function causes the active player to turn.

##### Parameters

<i>moment,:</i>	a double representing the turning angle in degrees.
-----------------	---

##### Precondition

The RoboCup server is available, team has been initialized.

##### Postcondition

The player has turned the given number of degrees from original orientation.

##### Returns

None

The documentation for this class was generated from the following file:

- [RoboClient.java](#)



## 4.23 SenseMemory Class Reference

### Public Member Functions

- [SenseMemory](#) ()
- [SenseMemory](#) (int time)
- int [getTime](#) ()
- void [setTime](#) (int t)
- void [setTime](#) (String[] seeOrSense)

### Public Attributes

- double **stamina**
- double **recovery**
- double **effort**
- double **amountOfSpeed**
- double **directionOfSpeed**
- double **headDirection**

#### 4.23.1 Detailed Description

This holds all the usable information parsed from the (sense\_body) message sent from the server. It holds information about a Player's stamina, speed, and head direction angle, as well as the time parsed.

#### 4.23.2 Constructor & Destructor Documentation

##### 4.23.2.1 `SenseMemory::SenseMemory ( )` `[inline]`

Default constructor

#### Postcondition

initializes time to 0

##### 4.23.2.2 `SenseMemory::SenseMemory ( int time )` `[inline]`

Constructor with time

#### Parameters

<i>time</i>	The time the information was parsed, as told by the server.
-------------	---

#### Postcondition

A new [SenseMemory](#) with updated time

### 4.23.3 Member Function Documentation

#### 4.23.3.1 `int SenseMemory::getTime ( )` `[inline]`

The time getter

##### Returns

the time that the [SenseMemory](#) was parsed

#### 4.23.3.2 `void SenseMemory::setTime ( String[] seeOrSense )` `[inline]`

Time setter from the unparsed message sent by server

##### Parameters

<i>seeOrSense</i>	A String array with the split first argument of a (see) message from the server
-------------------	---

#### 4.23.3.3 `void SenseMemory::setTime ( int t )` `[inline]`

The time setter

##### Parameters

<i>t</i>	the time hat the <a href="#">SenseMemory</a> was parsed
----------	---

The documentation for this class was generated from the following file:

- [SenseMemory.java](#)

## Chapter 5

# File Documentation

### 5.1 Action.java File Reference

#### Classes

- class [Action](#)

#### 5.1.1 Detailed Description

##### Author

Grant Hays

##### Date

11/9/11

##### Version

3.0

### 5.2 Brain.java File Reference

#### Classes

- class [Brain](#)

#### 5.2.1 Detailed Description

##### Author

Joel Tanzi

**Date**

17 October 2011

## 5.3 Field.java File Reference

**Classes**

- class [Field](#)

### 5.3.1 Detailed Description

A container for fixed points.

**Author**

Grant Hays

**Date**

10/13/11

**Version**

1

## 5.4 Forward.java File Reference

**Classes**

- class [Forward](#)

### 5.4.1 Detailed Description

Class file for [Forward](#) class

**Author**

Joel Tanzi

**Date**

5 November 2011

**Version**

1.0

## 5.5 FullBack.java File Reference

### Classes

- class [FullBack](#)

#### 5.5.1 Detailed Description

Class file for [FullBack](#) class

#### Author

Joel Tanzi

#### Date

5 November 2011

#### Version

1.0

## 5.6 Game.java File Reference

### Classes

- class [Game](#)

#### 5.6.1 Detailed Description

#### Author

Joel Tanzi\*

#### Date

18 September 2011

## 5.7 Goalie.java File Reference

### Classes

- class [Goalie](#)

#### 5.7.1 Detailed Description

Class file for [Goalie](#) class

**Author**

Joel Tanzi

**Date**

11 October 2011

**Version**

1.3

## 5.8 MathHelp.java File Reference

**Classes**

- class [MathHelp](#)

### 5.8.1 Detailed Description

This has functions of the math I need for calculations.

**Author**

granthays

**Date**

10/09/11

**Version**

1

## 5.9 Memory.java File Reference

**Classes**

- class [Memory](#)

### 5.9.1 Detailed Description

The [Memory](#) class stores instances of [ObjMemory](#) and [SenseMemory](#) and supplies methods to access their innards.

**Author**

granthays

**Date**

11/10/11

**Version**

3.0

## **5.10 Mode.java File Reference**

**Classes**

- class [Mode](#)

### **5.10.1 Detailed Description**

**Author**

Joel Tanzi\*

**Date**

18 October 2011

**Version**

1.0

## **5.11 ObjInfo.java File Reference**

**Classes**

- class [ObjInfo](#)
- class [ObjBall](#)
- class [ObjGoal](#)
- class [ObjFlag](#)
- class [ObjPlayer](#)
- class [ObjLine](#)

### **5.11.1 Detailed Description**

The [ObjInfo](#) container

**Author**

Grant Hays

**Date**

09/01/11

**Version**

1

## 5.12 ObjMemory.java File Reference

**Classes**

- class [ObjMemory](#)

### 5.12.1 Detailed Description

A container for ObjInfo's visible to the player after a parse

**Author**

Grant Hays

**Date**

09/03/11

**Version**

1

## 5.13 Parser.java File Reference

**Classes**

- class [Parser](#)

### 5.13.1 Detailed Description

The server message parser.

**Author**

Grant Hays

**Date**

10/1/11

**Version**

2



## 5.14 Player.java File Reference

### Classes

- class [Player](#)

#### 5.14.1 Detailed Description

Class file for [Player](#) class

#### Author

Joel Tanzi

#### Date

11 October 2011

#### Version

1.0

## 5.15 Pos.java File Reference

### Classes

- class [Pos](#)

#### 5.15.1 Detailed Description

The Position vector for Cartesian Coordinates

#### Author

Grant Hays

#### Date

10/11/11

#### Version

1

## 5.16 RoboClient.java File Reference

### Classes

- class [RoboClient](#)

### 5.16.1 Detailed Description

Class file for [RoboClient](#) class

**Author**

Joel Tanzi

**Date**

September 20, 2011

**Version**

1.2

## 5.17 SenseMemory.java File Reference

**Classes**

- class [SenseMemory](#)

### 5.17.1 Detailed Description

Container for parsed (sense\_body) information

**Author**

Grant Hays

**Date**

09/10/11

**Version**

1

# Index

Action, [7](#)  
    Action, [8](#)  
    dribbleToGoal, [8](#)  
    findBall, [8](#)  
    goHome, [9](#)  
    gotoPoint, [9](#)  
    kickToPoint, [10](#)  
    setMem, [10](#)  
Action.java, [65](#)  
addInfo  
    ObjMemory, [42](#)  
  
ballInGoalzone  
    Goalie, [16](#)  
Brain, [11](#)  
    Brain, [11](#)  
    getActions, [12](#)  
    getCurrentMode, [12](#)  
    getMarked\_team, [12](#)  
    getMarked\_unum, [12](#)  
    run, [12](#)  
    setActions, [12](#)  
    setDefensive, [13](#)  
    setMarked\_team, [13](#)  
    setMarked\_unum, [13](#)  
    setOffensive, [13](#)  
Brain.java, [65](#)  
  
catchable  
    Goalie, [17](#)  
catchball  
    Goalie, [17](#)  
    RoboClient, [58](#)  
closestOpponent  
    Player, [50](#)  
closestPlayer  
    Goalie, [17](#)  
  
dash  
    Player, [50](#)  
    RoboClient, [59](#)  
  
defendGoal  
    Goalie, [18](#)  
dribbleToGoal  
    Action, [8](#)  
  
edp  
    MathHelp, [21](#)  
  
Field, [13](#)  
    Field, [14](#)  
Field.java, [66](#)  
findBall  
    Action, [8](#)  
followBall  
    Goalie, [18](#)  
Forward, [14](#)  
Forward.java, [66](#)  
FullBack, [15](#)  
FullBack.java, [67](#)  
  
Game, [15](#)  
Game.java, [67](#)  
getActions  
    Brain, [12](#)  
getAmountOfSpeed  
    Memory, [27](#)  
getBall  
    Memory, [27](#)  
getBodyDir  
    ObjPlayer, [45](#)  
getBtwBallAndGoal  
    Goalie, [18](#)  
getClosestBoundary  
    Memory, [27](#)  
getClosestFlag  
    Memory, [27](#)  
getClosestLine  
    Memory, [27](#)  
getClosestPenaltyFlag  
    Memory, [28](#)  
getCurrentMode

- Brain, 12
- getDashPower
  - MathHelp, 21
- getDirChng
  - ObjInfo, 39
- getDirection
  - Memory, 28
  - ObjInfo, 39
  - Player, 50
- getDirectionOfSpeed
  - Memory, 28
- getDistance
  - ObjInfo, 40
- getDistChng
  - ObjInfo, 40
- getEffort
  - Memory, 28
- getFlag
  - Memory, 28
- getFlagName
  - ObjFlag, 36
- getFlagPos
  - Memory, 29
- getFlagType
  - ObjFlag, 36
- getHeadDir
  - ObjPlayer, 45
- getHeadDirection
  - Memory, 29
- getKickPower
  - MathHelp, 21
- getLine
  - Memory, 29
- getMarked\_team
  - Brain, 12
- getMarked\_unum
  - Brain, 12
- getMem
  - Player, 50
- getModename
  - Mode, 35
- getNextBallPoint
  - MathHelp, 22
- getNextOpponentPoint
  - MathHelp, 22
- getNullGoalAngle
  - Memory, 29
- getObj
  - Memory, 29
  - ObjMemory, 43
- getObjInfo
  - Player, 51
- getObjMemorySize
  - Memory, 30
- getObjName
  - ObjInfo, 40
- getOppGoal
  - Memory, 30
- getOppGoalPos
  - Memory, 30
- getOwnGoal
  - Memory, 31
- getOwnGoalPos
  - Memory, 31
- getParser
  - Player, 51
- getPlayer
  - Memory, 31
- getPlayers
  - Memory, 31
- getPlayMode
  - Memory, 31
- getPolar
  - MathHelp, 22, 23
- getPort
  - RoboClient, 59
- getPos
  - MathHelp, 23
- getPosition
  - Memory, 32
  - Player, 51
- getRecovery
  - Memory, 32
- getRoboClient
  - Player, 51
- getSide
  - ObjInfo, 40
- getSize
  - ObjMemory, 43
- getStamina
  - Memory, 32
- getTeam
  - ObjPlayer, 45
  - RoboClient, 59
- getTime
  - ObjMemory, 43
  - Player, 51
  - SenseMemory, 64
- getTimeinmode
  - Mode, 35

- getuNum
  - ObjPlayer, 45
- getX\_pos
  - ObjFlag, 37
- getY\_pos
  - ObjFlag, 37
- getYard
  - ObjFlag, 37
- Goalie, 15
  - ballInGoalzone, 16
  - catchable, 17
  - catchball, 17
  - closestPlayer, 17
  - defendGoal, 18
  - followBall, 18
  - getBtwBallAndGoal, 18
  - initGoalie, 19
  - kickBallOutOfBounds, 19
  - kickToPlayer, 19
  - run, 19
- Goalie.java, 67
- goHome
  - Action, 9
- gotoPoint
  - Action, 9
- init
  - RoboClient, 59
- initGoalie
  - Goalie, 19
  - RoboClient, 59
- initParse
  - Parser, 47
- initPlayer
  - Player, 51
- input
  - Parser, 48
- isGoalie
  - ObjPlayer, 45
- isObjVisible
  - Memory, 32
- kick
  - Player, 52
  - RoboClient, 60
- kickBallOutOfBounds
  - Goalie, 19
- kickToPlayer
  - Goalie, 19
- kickToPoint
  - Action, 10
- mag
  - MathHelp, 23
- MathHelp, 20
  - edp, 21
  - getDashPower, 21
  - getKickPower, 21
  - getNextBallPoint, 22
  - getNextOpponentPoint, 22
  - getPolar, 22, 23
  - getPos, 23
  - mag, 23
  - norm, 24
  - vAdd, 24
  - vDiv, 24
  - vMul, 25
  - vSub, 25
- MathHelp.java, 68
- Memory, 25
  - getAmountOfSpeed, 27
  - getBall, 27
  - getClosestBoundary, 27
  - getClosestFlag, 27
  - getClosestLine, 27
  - getClosestPenaltyFlag, 28
  - getDirection, 28
  - getDirectionOfSpeed, 28
  - getEffort, 28
  - getFlag, 28
  - getFlagPos, 29
  - getHeadDirection, 29
  - getLine, 29
  - getNullGoalAngle, 29
  - getObj, 29
  - getObjMemorySize, 30
  - getOppGoal, 30
  - getOppGoalPos, 30
  - getOwnGoal, 31
  - getOwnGoalPos, 31
  - getPlayer, 31
  - getPlayers, 31
  - getPlayMode, 31
  - getPosition, 32
  - getRecovery, 32
  - getStamina, 32
  - isObjVisible, 32
  - Memory, 27
  - ObjMem, 33
  - oppGoal, 33

- oppSide, 33
- playMode, 33
- SenMem, 34
- setField, 32
- setLocation, 33
- side, 34
- timeCheck, 33
- uNum, 34
- Memory.java, 68
- Mode, 34
  - getModename, 35
  - getTimeinmode, 35
  - Mode, 34
  - setModename, 35
  - setTimeinmode, 35
- Mode.java, 69
- move
  - Player, 52
  - RoboClient, 60
- norm
  - MathHelp, 24
- ObjBall, 35
- ObjFlag, 36
  - getFlagName, 36
  - getFlagType, 36
  - getX\_pos, 37
  - getY\_pos, 37
  - getYard, 37
  - ObjFlag, 36
  - setFlagName, 37
  - setFlagType, 37
  - setX\_pos, 37
  - setY\_pos, 37
  - setYard, 38
- ObjGoal, 38
- ObjInfo, 38
  - getDirChng, 39
  - getDirection, 39
  - getDistance, 40
  - getDistChng, 40
  - getObjName, 40
  - getSide, 40
  - ObjInfo, 39
  - setDirChng, 40
  - setDirection, 40
  - setDistance, 40
  - setDistChng, 40
  - setObjName, 41
  - setSide, 41
- ObjInfo.java, 69
- ObjLine, 41
- ObjMem
  - Memory, 33
- ObjMemory, 41
  - addInfo, 42
  - getObj, 43
  - getSize, 43
  - getTime, 43
  - ObjMemory, 42
  - setTime, 44
- ObjMemory.java, 70
- ObjPlayer, 44
  - getBodyDir, 45
  - getHeadDir, 45
  - getTeam, 45
  - getuNum, 45
  - isGoalie, 45
  - setBodyDir, 45
  - setGoalie, 46
  - setHeadDir, 46
  - setTeam, 46
  - setuNum, 46
- oppGoal
  - Memory, 33
- oppSide
  - Memory, 33
- Parse
  - Parser, 47
- Parser, 46
  - initParse, 47
  - input, 48
  - Parse, 47
  - Parser, 47
- Parser.java, 70
- Player, 48
  - closestOpponent, 50
  - dash, 50
  - getDirection, 50
  - getMem, 50
  - getObjInfo, 51
  - getParser, 51
  - getPosition, 51
  - getRoboClient, 51
  - getTime, 51
  - initPlayer, 51
  - kick, 52
  - move, 52

- Player, 49
- receiveInput, 53
- run, 53
- say, 53
- setBrain, 53
- setMem, 53
- setObjInfo, 54
- setParser, 54
- setRoboclient, 54
- setTime, 54
- turn, 54
- Player.java, 71
- playMode
  - Memory, 33
- Polar, 55
  - Polar, 55
- Pos, 56
  - Pos, 56, 57
- Pos.java, 71
- receive
  - RoboClient, 61
- receiveInput
  - Player, 53
- RoboClient, 57
  - catchball, 58
  - dash, 59
  - getPort, 59
  - getTeam, 59
  - init, 59
  - initGoalie, 59
  - kick, 60
  - move, 60
  - receive, 61
  - RoboClient, 58
  - say, 61
  - send, 61
  - setTeam, 62
  - turn, 62
- RoboClient.java, 71
- run
  - Brain, 12
  - Goalie, 19
  - Player, 53
- say
  - Player, 53
  - RoboClient, 61
- send
  - RoboClient, 61
- SenMem
  - Memory, 34
- SenseMemory, 63
  - getTime, 64
  - SenseMemory, 63
  - setTime, 64
- SenseMemory.java, 72
- setActions
  - Brain, 12
- setBodyDir
  - ObjPlayer, 45
- setBrain
  - Player, 53
- setDefensive
  - Brain, 13
- setDirChng
  - ObjInfo, 40
- setDirection
  - ObjInfo, 40
- setDistance
  - ObjInfo, 40
- setDistChng
  - ObjInfo, 40
- setField
  - Memory, 32
- setFlagName
  - ObjFlag, 37
- setFlagType
  - ObjFlag, 37
- setGoalie
  - ObjPlayer, 46
- setHeadDir
  - ObjPlayer, 46
- setLocation
  - Memory, 33
- setMarked\_team
  - Brain, 13
- setMarked\_unum
  - Brain, 13
- setMem
  - Action, 10
  - Player, 53
- setModename
  - Mode, 35
- setObjInfo
  - Player, 54
- setObjName
  - ObjInfo, 41
- setOffensive
  - Brain, 13

- setParser
  - Player, [54](#)
- setRoboclient
  - Player, [54](#)
- setSide
  - ObjInfo, [41](#)
- setTeam
  - ObjPlayer, [46](#)
  - RoboClient, [62](#)
- setTime
  - ObjMemory, [44](#)
  - Player, [54](#)
  - SenseMemory, [64](#)
- setTimeinmode
  - Mode, [35](#)
- setuNum
  - ObjPlayer, [46](#)
- setX\_pos
  - ObjFlag, [37](#)
- setY\_pos
  - ObjFlag, [37](#)
- setYard
  - ObjFlag, [38](#)
- side
  - Memory, [34](#)
- timeCheck
  - Memory, [33](#)
- turn
  - Player, [54](#)
  - RoboClient, [62](#)
- uNum
  - Memory, [34](#)
- vAdd
  - MathHelp, [24](#)
- vDiv
  - MathHelp, [24](#)
- vMul
  - MathHelp, [25](#)
- vSub
  - MathHelp, [25](#)