

Reference Manual

Generated by Doxygen 1.7.3

Mon Dec 5 2011 00:02:23

Contents

1	Class Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	Action Class Reference	7
4.1.1	Detailed Description	8
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	closestPlayer	8
4.1.3.2	dribbleToGoal	9
4.1.3.3	findBall	9
4.1.3.4	FullBack_findBall	9
4.1.3.5	getTurn	9
4.1.3.6	goHome	10
4.1.3.7	gotoPoint	10
4.1.3.8	gotoPoint	10
4.1.3.9	gotoSidePoint	11
4.1.3.10	kickToPoint	11
4.1.3.11	kickToPoint	11
4.1.3.12	passBall	12
4.1.3.13	setMem	12
4.2	Brain Class Reference	12
4.2.1	Detailed Description	13
4.2.2	Constructor & Destructor Documentation	13
4.2.2.1	Brain	13
4.2.3	Member Function Documentation	13
4.2.3.1	getCurrentMode	13
4.2.3.2	getMarked_team	13
4.2.3.3	getMarked_unum	13
4.2.3.4	setDefensive	13
4.2.3.5	setMarked_team	14

4.2.3.6	setMarked_unum	14
4.2.3.7	setOffensive	14
4.3	Field Class Reference	14
4.3.1	Detailed Description	14
4.3.2	Constructor & Destructor Documentation	15
4.3.2.1	Field	15
4.4	Forward Class Reference	15
4.4.1	Detailed Description	15
4.5	FullBack Class Reference	15
4.5.1	Detailed Description	16
4.5.2	Constructor & Destructor Documentation	16
4.5.2.1	FullBack	16
4.5.2.2	FullBack	16
4.5.3	Member Function Documentation	17
4.5.3.1	closestPlayer	17
4.5.3.2	initFullBack	17
4.5.3.3	initFullBack	17
4.5.3.4	runDefense	18
4.6	FullBackBrain Class Reference	18
4.6.1	Constructor & Destructor Documentation	18
4.6.1.1	FullBackBrain	18
4.6.1.2	FullBackBrain	18
4.6.2	Member Function Documentation	19
4.6.2.1	getActions	19
4.6.2.2	getCurrentMode	19
4.6.2.3	getMarked_team	19
4.6.2.4	getMarked_unum	19
4.6.2.5	run	19
4.6.2.6	setActions	19
4.6.2.7	setDefensive	19
4.6.2.8	setMarked_team	20
4.6.2.9	setMarked_unum	20
4.6.2.10	setOffensive	20
4.7	Game Class Reference	20
4.7.1	Detailed Description	20
4.8	Goalie Class Reference	20
4.8.1	Detailed Description	21
4.8.2	Member Function Documentation	22
4.8.2.1	ballInGoalzone	22
4.8.2.2	catchable	22
4.8.2.3	catchball	22
4.8.2.4	closestPlayer	23
4.8.2.5	defendGoal	23
4.8.2.6	followBall	23
4.8.2.7	getBtwBallAndGoal	24
4.8.2.8	initGoalie	24
4.8.2.9	kickBallOutOfBounds	24
4.8.2.10	kickToPlayer	24
4.8.2.11	positionGoalie	25
4.9	GoalieBrain Class Reference	25

4.9.1	Constructor & Destructor Documentation	26
4.9.1.1	GoalieBrain	26
4.9.1.2	GoalieBrain	26
4.9.2	Member Function Documentation	26
4.9.2.1	getActions	26
4.9.2.2	getCurrentMode	26
4.9.2.3	getMarked_team	26
4.9.2.4	getMarked_unum	26
4.9.2.5	run	27
4.9.2.6	setActions	27
4.9.2.7	setDefensive	27
4.9.2.8	setMarked_team	27
4.9.2.9	setMarked_unum	27
4.9.2.10	setOffensive	27
4.10	MathHelp Class Reference	27
4.10.1	Member Function Documentation	28
4.10.1.1	edp	28
4.10.1.2	getDashPower	28
4.10.1.3	getKickPower	29
4.10.1.4	getKickPower	29
4.10.1.5	getNextBallPoint	29
4.10.1.6	getNextPlayerPoint	30
4.10.1.7	getPolar	30
4.10.1.8	getPolar	30
4.10.1.9	getPos	30
4.10.1.10	getPos	31
4.10.1.11	mag	31
4.10.1.12	norm	31
4.10.1.13	norm	32
4.10.1.14	vAdd	32
4.10.1.15	vDiv	32
4.10.1.16	vMul	32
4.10.1.17	vSub	33
4.11	Memory Class Reference	33
4.11.1	Constructor & Destructor Documentation	34
4.11.1.1	Memory	34
4.11.2	Member Function Documentation	35
4.11.2.1	getAbsPolar	35
4.11.2.2	getAmountOfSpeed	35
4.11.2.3	getBall	35
4.11.2.4	getBallPos	35
4.11.2.5	getClosestBoundary	36
4.11.2.6	getClosestFlag	36
4.11.2.7	getClosestLine	36
4.11.2.8	getClosestPenaltyFlag	36
4.11.2.9	getDirection	36
4.11.2.10	getDirectionOfSpeed	36
4.11.2.11	getEffort	37
4.11.2.12	getFlag	37
4.11.2.13	getFlagPos	37

4.11.2.14	getHeadDirection	37
4.11.2.15	getLine	37
4.11.2.16	getNullGoalAngle	38
4.11.2.17	getObj	38
4.11.2.18	getObjMemorySize	38
4.11.2.19	getOppGoal	38
4.11.2.20	getOppGoalPos	39
4.11.2.21	getOwnGoal	39
4.11.2.22	getOwnGoalPos	39
4.11.2.23	getPlayer	39
4.11.2.24	getPlayers	40
4.11.2.25	getPlayMode	40
4.11.2.26	getPosition	40
4.11.2.27	getRecovery	40
4.11.2.28	getStamina	40
4.11.2.29	isObjVisible	40
4.11.2.30	setCurrent	40
4.11.2.31	setField	41
4.11.2.32	setLocation	41
4.11.2.33	timeCheck	41
4.11.3	Member Data Documentation	42
4.11.3.1	ObjMem	42
4.11.3.2	oppGoal	42
4.11.3.3	oppSide	42
4.11.3.4	playMode	42
4.11.3.5	SenMem	42
4.11.3.6	side	42
4.11.3.7	uNum	42
4.12	Mode Class Reference	42
4.12.1	Detailed Description	43
4.12.2	Constructor & Destructor Documentation	43
4.12.2.1	Mode	43
4.12.3	Member Function Documentation	43
4.12.3.1	getModename	43
4.12.3.2	getTimeinmode	43
4.12.3.3	setModename	43
4.12.3.4	setTimeinmode	43
4.13	ObjBall Class Reference	44
4.13.1	Detailed Description	44
4.14	ObjFlag Class Reference	44
4.14.1	Constructor & Destructor Documentation	45
4.14.1.1	ObjFlag	45
4.14.2	Member Function Documentation	45
4.14.2.1	getFlagName	45
4.14.2.2	getFlagType	45
4.14.2.3	getX_pos	45
4.14.2.4	getY_pos	45
4.14.2.5	getYard	46
4.14.2.6	setFlagName	46
4.14.2.7	setFlagType	46

4.14.2.8	setX_pos	46
4.14.2.9	setY_pos	46
4.14.2.10	setYard	46
4.15	ObjGoal Class Reference	46
4.15.1	Detailed Description	47
4.16	ObjInfo Class Reference	47
4.16.1	Detailed Description	47
4.16.2	Constructor & Destructor Documentation	47
4.16.2.1	ObjInfo	47
4.16.2.2	ObjInfo	48
4.16.3	Member Function Documentation	48
4.16.3.1	getDirChng	48
4.16.3.2	getDirection	48
4.16.3.3	getDistance	48
4.16.3.4	getDistChng	48
4.16.3.5	getObjName	48
4.16.3.6	getSide	49
4.16.3.7	setDirChng	49
4.16.3.8	setDirection	49
4.16.3.9	setDistance	49
4.16.3.10	setDistChng	49
4.16.3.11	setObjName	49
4.16.3.12	setSide	49
4.17	ObjLine Class Reference	49
4.17.1	Detailed Description	50
4.18	ObjMemory Class Reference	50
4.18.1	Detailed Description	50
4.18.2	Constructor & Destructor Documentation	50
4.18.2.1	ObjMemory	50
4.18.2.2	ObjMemory	50
4.18.3	Member Function Documentation	51
4.18.3.1	addInfo	51
4.18.3.2	getObj	51
4.18.3.3	getObj	51
4.18.3.4	getSize	52
4.18.3.5	getTime	52
4.18.3.6	setTime	52
4.19	ObjPlayer Class Reference	52
4.19.1	Detailed Description	53
4.19.2	Member Function Documentation	53
4.19.2.1	getBodyDir	53
4.19.2.2	getHeadDir	53
4.19.2.3	getTeam	53
4.19.2.4	getuNum	54
4.19.2.5	isGoalie	54
4.19.2.6	setBodyDir	54
4.19.2.7	setGoalie	54
4.19.2.8	setHeadDir	54
4.19.2.9	setTeam	54
4.19.2.10	setuNum	54

4.20	Parser Class Reference	54
4.20.1	Detailed Description	55
4.20.2	Constructor & Destructor Documentation	55
4.20.2.1	Parser	55
4.20.3	Member Function Documentation	55
4.20.3.1	initParse	55
4.20.3.2	Parse	55
4.20.4	Member Data Documentation	56
4.20.4.1	input	56
4.21	Player Class Reference	56
4.21.1	Detailed Description	58
4.21.2	Constructor & Destructor Documentation	58
4.21.2.1	Player	58
4.21.3	Member Function Documentation	58
4.21.3.1	closestOpponent	58
4.21.3.2	dash	59
4.21.3.3	dash	59
4.21.3.4	getDirection	59
4.21.3.5	getHome	59
4.21.3.6	getMem	60
4.21.3.7	getMemTime	60
4.21.3.8	getObjInfo	60
4.21.3.9	getParser	60
4.21.3.10	getPosition	60
4.21.3.11	getRoboClient	60
4.21.3.12	getTime	60
4.21.3.13	initPlayer	61
4.21.3.14	initPlayer	61
4.21.3.15	kick	61
4.21.3.16	markOpponent	62
4.21.3.17	move	62
4.21.3.18	receiveInput	62
4.21.3.19	recievePass	62
4.21.3.20	runDefense	63
4.21.3.21	say	63
4.21.3.22	setHome	63
4.21.3.23	setMem	63
4.21.3.24	setObjInfo	64
4.21.3.25	setParser	64
4.21.3.26	setRoboclient	64
4.21.3.27	setTime	64
4.21.3.28	turn	64
4.21.3.29	turn_neck	65
4.22	Polar Class Reference	65
4.22.1	Detailed Description	65
4.22.2	Constructor & Destructor Documentation	65
4.22.2.1	Polar	65
4.22.2.2	Polar	66
4.23	Pos Class Reference	66
4.23.1	Detailed Description	66

4.23.2	Constructor & Destructor Documentation	66
4.23.2.1	Pos	66
4.23.2.2	Pos	67
4.23.2.3	Pos	67
4.24	RoboClient Class Reference	67
4.24.1	Detailed Description	68
4.24.2	Constructor & Destructor Documentation	68
4.24.2.1	RoboClient	68
4.24.2.2	RoboClient	68
4.24.3	Member Function Documentation	68
4.24.3.1	catchball	68
4.24.3.2	dash	69
4.24.3.3	dash	69
4.24.3.4	getTeam	70
4.24.3.5	init	70
4.24.3.6	initGoalie	70
4.24.3.7	kick	70
4.24.3.8	move	71
4.24.3.9	receive	71
4.24.3.10	say	72
4.24.3.11	send	72
4.24.3.12	setTeam	72
4.24.3.13	turn	73
4.25	SenseMemory Class Reference	73
4.25.1	Detailed Description	73
4.25.2	Constructor & Destructor Documentation	74
4.25.2.1	SenseMemory	74
4.25.2.2	SenseMemory	74
4.25.3	Member Function Documentation	74
4.25.3.1	getTime	74
4.25.3.2	setTime	74
4.25.3.3	setTime	74
5	File Documentation	77
5.1	Action.java File Reference	77
5.1.1	Detailed Description	77
5.2	Brain.java File Reference	77
5.2.1	Detailed Description	77
5.3	Field.java File Reference	78
5.3.1	Detailed Description	78
5.4	Forward.java File Reference	78
5.4.1	Detailed Description	78
5.5	FullBack.java File Reference	78
5.5.1	Detailed Description	79
5.6	Game.java File Reference	79
5.6.1	Detailed Description	79
5.7	Goalie.java File Reference	79
5.7.1	Detailed Description	79
5.8	MathHelp.java File Reference	80
5.8.1	Detailed Description	80

5.9	Memory.java File Reference	80
5.9.1	Detailed Description	80
5.10	Mode.java File Reference	80
5.10.1	Detailed Description	81
5.11	ObjInfo.java File Reference	81
5.11.1	Detailed Description	81
5.12	ObjMemory.java File Reference	81
5.12.1	Detailed Description	82
5.13	Parser.java File Reference	82
5.13.1	Detailed Description	82
5.14	Player.java File Reference	82
5.14.1	Detailed Description	82
5.15	Pos.java File Reference	83
5.15.1	Detailed Description	83
5.16	RoboClient.java File Reference	83
5.16.1	Detailed Description	83
5.17	SenseMemory.java File Reference	84
5.17.1	Detailed Description	84

Chapter 1

Class Index

1.1 Class Hierarchy

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

Action	7
Brain	12
Field	14
FullBackBrain	18
Game	20
GoalieBrain	25
MathHelp	27
Memory	33
Mode	42
ObjInfo	47
ObjBall	44
ObjFlag	44
ObjGoal	46
ObjLine	49
ObjPlayer	52
ObjMemory	50
Parser	54
Player	56
Forward	15
FullBack	15
Goalie	20
Polar	65
Pos	66
RoboClient	67
SenseMemory	73

Chapter 2

Class Index

2.1 Class List

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

Action	7
Brain	12
Field	14
Forward	15
FullBack	15
FullBackBrain	18
Game	20
Goalie	20
GoalieBrain	25
MathHelp	27
Memory	33
Mode	42
ObjBall	44
ObjFlag	44
ObjGoal	46
ObjInfo	47
ObjLine	49
ObjMemory	50
ObjPlayer	52
Parser	54
Player	56
Polar	65
Pos	66
RoboClient	67
SenseMemory	73

Chapter 3

File Index

3.1 File List

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

Action.java	77
Brain.java	77
Field.java	78
Forward.java	78
FullBack.java	78
Game.java	79
Goalie.java	79
MathHelp.java	80
Memory.java	80
Mode.java	80
ObjInfo.java	81
ObjMemory.java	81
Parser.java	82
Player.java	82
Pos.java	83
RoboClient.java	83
SenseMemory.java	84

Chapter 4

Class Documentation

4.1 Action Class Reference

Public Member Functions

- [Action](#) ()
- [Action](#) ([Memory](#) mem, [RoboClient](#) rc)
- void [setMem](#) ([Memory](#) mem)
- double [getTurn](#) ([Polar](#) go)
- void [gotoPoint](#) ([Polar](#) go)
- void [gotoSidePoint](#) ([Pos](#) p)
- void [gotoPoint](#) ([Pos](#) p)
- void [goHome](#) ()
- void [findBall](#) () throws [UnknownHostException](#), [InterruptedException](#)
- void [kickOff](#) () throws [UnknownHostException](#), [InterruptedException](#)
- [ObjPlayer](#) [closestPlayer](#) () throws [UnknownHostException](#), [InterruptedException](#)
- void [passBall](#) ([ObjBall](#) ball, [ObjPlayer](#) p)
- void [FullBack_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 Memory containing all the parsed information from the server
<i>rc</i>	The RoboClient 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 `ObjPlayer Action::closestPlayer ()` throws `UnknownHostException`, `InterruptedException` `[inline]`

Returns the closest player to the [FullBack](#) on the same team.

Postcondition

The closest player to the [FullBack](#) has been determined.

Returns

[ObjPlayer](#)

Exceptions

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

4.1.3.2 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

4.1.3.3 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>	

4.1.3.4 void Action::FullBack_findBall () throws UnknownHostException, InterruptedException [inline]

A method to find the ball on the field for FullBacks. If it's not in view, the [FullBack](#) turns until he finds it. If the ball is out of kickable range, he dashes to get to it. If the ball is within 15 distance, he intercepts the ball, and kicks it away.

Exceptions

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

4.1.3.5 double Action::getTurn (Polar *go*) [inline]

Gets the actual angle of a turn

Parameters

<i>go</i>	polar of angle to turn to
-----------	---------------------------

Returns

the double of the correct turn moment

4.1.3.6 void 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

4.1.3.7 void Action::gotoPoint (Pos p) [inline]

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

Parameters

<i>p</i>	The Cartesian Pos 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.

4.1.3.8 void Action::gotoPoint (Polar go) [inline]

This tells the player to turn and run to a point

Parameters

<i>go</i>	The Polar 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.

4.1.3.9 void Action::gotoSidePoint (Pos *p*) [inline]

A method to dash sideways or backwards

Parameters

<i>p</i>	the cartesian point to go to
----------	------------------------------

Postcondition

player dashes sideways or backwards while facing forward

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

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

Parameters

<i>ball</i>	
<i>p</i>	The Polar 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

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

A [Pos](#) wrapper for the kickToPoint

Parameters

<i>ball</i>	
<i>p</i>	the Pos of the coordinate to kick the ball to

4.1.3.12 void Action::passBall (ObjBall *ball*, ObjPlayer *p*) [inline]

Passes the ball to the nearest [Forward](#) (currently [Player](#)).

Parameters

<i>ball</i>	An ObjBall for the ball in play.
<i>fwd</i>	The player to pass the ball to.

Precondition

The [FullBack](#) has control of the ball.

Postcondition

The ball has been kicked to the forward.

4.1.3.13 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 Memory
------------	-------------------------------------

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*)
- [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**
- [MathHelp](#) **mh**

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

4.2.3 Member Function Documentation

4.2.3.1 **Mode Brain::getCurrentMode** () [inline]

Returns

the currentMode

4.2.3.2 **String Brain::getMarked_team** () [inline]

Returns

the marked_team

4.2.3.3 **String Brain::getMarked_unum** () [inline]

Returns

the marked_unum

4.2.3.4 **void Brain::setDefensive** () [inline]

Sets the player mode to defensive

4.2.3.5 void Brain::setMarked_team (String *marked_team*) [inline]

Parameters

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

4.2.3.6 void Brain::setMarked_unum (String *marked_unum*) [inline]

Parameters

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

4.2.3.7 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

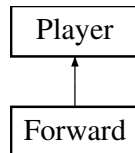
A new [Field](#) will be created with access to an array list of all the field's fixed points

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

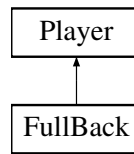
) The [Forward](#) class inherits from the [Player](#) class. The [Forward](#) is a specialized type of [Player](#) that focuses on offensive behaviors such as scoring and ball interception.

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

- [Forward.java](#)

4.5 FullBack Class Reference

Inheritance diagram for FullBack:



Public Member Functions

- **FullBack** ([RoboClient](#) rc, [Memory](#) m, [ObjInfo](#) i, [Parser](#) p, int time)
- **FullBack** (String team)
- void **initFullBack** (double x, double y) throws [SocketException](#), [UnknownHostException](#)
- void **initFullBack** (double x, double y, String pos) throws [SocketException](#), [UnknownHostException](#)
- [ObjPlayer](#) **closestPlayer** () throws [UnknownHostException](#), [InterruptedException](#)
- boolean **inFullBackZone** ()
- void **runDefense** () throws [UnknownHostException](#), [InterruptedException](#)
- void **run** ()

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.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 [FullBack::FullBack](#) ([RoboClient](#) rc, [Memory](#) m, [ObjInfo](#) i, [Parser](#) p, int time)
[inline]

Parameters

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

4.5.2.2 [FullBack::FullBack](#) ([String](#) team) [inline]

Parameters

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

4.5.3 Member Function Documentation

4.5.3.1 `ObjPlayer FullBack::closestPlayer ()` throws `UnknownHostException`, `InterruptedException` [inline]

Returns the closest player to the [FullBack](#) on the same team.

Postcondition

The closest player to the [FullBack](#) has been determined.

Returns

[ObjPlayer](#)

Exceptions

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

4.5.3.2 `void FullBack::initFullBack (double x, double y, String pos)` 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.

4.5.3.3 `void FullBack::initFullBack (double x, double y)` 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.

4.5.3.4 void FullBack::runDefense () throws UnknownHostException, InterruptedException [inline]

Follows opposing players on defense (Currently unused)

Reimplemented from [Player](#).

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

- [FullBack.java](#)

4.6 FullBackBrain Class Reference

Public Member Functions

- [FullBackBrain](#) ()
- [FullBackBrain](#) ([FullBack](#) f)
- [Action](#) [getActions](#) ()
- void [setActions](#) ([Action](#) actions)
- [FullBackBrain](#) ([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

- [FullBack](#) f
- [Memory](#) m

4.6.1 Constructor & Destructor Documentation

4.6.1.1 FullBackBrain::FullBackBrain () [inline]

Default constructor

4.6.1.2 FullBackBrain::FullBackBrain ([Mode](#) currentMode) [inline]

Constructor

Parameters

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

4.6.2 Member Function Documentation

4.6.2.1 Action FullBackBrain::getActions () [inline]

Returns

the actions

4.6.2.2 Mode FullBackBrain::getCurrentMode () [inline]

Returns

the currentMode

4.6.2.3 String FullBackBrain::getMarked_team () [inline]

Returns

the marked_team

4.6.2.4 String FullBackBrain::getMarked_unum () [inline]

Returns

the marked_unum

4.6.2.5 void FullBackBrain::run () [inline]

The [FullBackBrain](#) thread run method. It instructs the [FullBack](#) in soccer behaviors

Postcondition

[FullBack](#) will act accordingly during match.

4.6.2.6 void FullBackBrain::setActions (Action actions) [inline]

Parameters

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

4.6.2.7 void FullBackBrain::setDefensive () [inline]

Sets the player mode to defensive

4.6.2.8 void FullBackBrain::setMarked_team (String *marked_team*) [inline]

Parameters

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

4.6.2.9 void FullBackBrain::setMarked_unum (String *marked_unum*) [inline]

Parameters

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

4.6.2.10 void FullBackBrain::setOffensive () [inline]

Sets the player mode to be offensive

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

- FullBackBrain.java

4.7 Game Class Reference

Static Public Member Functions

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

4.7.1 Detailed Description

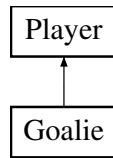
This serves as a main class to assemble the RoboCup team and set them into action for the match.

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

- [Game.java](#)

4.8 Goalie Class Reference

Inheritance diagram for Goalie:



Public Member Functions

- **Goalie** (String team)
- void **initGoalie** (double x, double y) 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 **positionGoalie** (ObjBall ball) throws 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

4.8.1 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.

4.8.2 Member Function Documentation

4.8.2.1 `boolean Goalie::ballInGoalzone (ObjBall ball)` `[inline]`

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

Parameters

<i>ball</i>	the ObjBall 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

4.8.2.2 `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.

4.8.2.3 `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.

4.8.2.4 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>	

4.8.2.5 void Goalie::defendGoal (ObjBall ball) throws UnknownHostException, InterruptedException [inline]

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

Parameters

ObjBall	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.

4.8.2.6 void Goalie::followBall () [inline]

Turns goalie toward the ball

Postcondition

The goalie will turn in the direction of the ball

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

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

Parameters

<i>ball</i>	An ObjBall .
-------------	------------------------------

Precondition

Ball is visible to the goalie.

Postcondition

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

4.8.2.8 void Goalie::initGoalie (double *x*, double *y*) 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.

4.8.2.9 void Goalie::kickBallOutOfBounds () [*inline*]

Causes the goalie to kick the ball out of bounds (Currently unused.)

Precondition

[Goalie](#) has control of the ball

Postcondition

Ball has been kicked out of bounds

4.8.2.10 void Goalie::kickToPlayer ([ObjPlayer](#) *player*) [*inline*]

Causes goalie to kick the ball to a specific player. (Currently unused.)

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 ObjPlayer representing the player to receive the ball.
---------------	---

4.8.2.11 void Goalie::positionGoalie ([ObjBall](#) *ball*) throws [InterruptedException](#)
 [inline]

Moves goalie to specific points within the goalbox dependent upon where the ball is on the field.

Parameters

<i>ball</i>	An ObjBall representing the ball in play.
-------------	---

Exceptions

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

Precondition

The ball is visible.

Postcondition

The goalie has moved to a strategic position to get between the ball and the goal.

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

- [Goalie.java](#)

4.9 GoalieBrain Class Reference

Public Member Functions

- [GoalieBrain](#) ()
- **[GoalieBrain](#)** ([Goalie](#) g)
- [Action](#) [getActions](#) ()
- void [setActions](#) ([Action](#) actions)
- [GoalieBrain](#) ([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

- [Goalie](#) **g**
- [Memory](#) **m**

4.9.1 Constructor & Destructor Documentation

4.9.1.1 **GoalieBrain::GoalieBrain** () [inline]

Default constructor

4.9.1.2 **GoalieBrain::GoalieBrain** (**Mode** *currentMode*) [inline]

Constructor

Parameters

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

4.9.2 Member Function Documentation

4.9.2.1 **Action** **GoalieBrain::getActions** () [inline]

Returns

the actions

4.9.2.2 **Mode** **GoalieBrain::getCurrentMode** () [inline]

Returns

the currentMode

4.9.2.3 **String** **GoalieBrain::getMarked_team** () [inline]

Returns

the marked_team

4.9.2.4 **String** **GoalieBrain::getMarked_unum** () [inline]

Returns

the marked_unum

4.9.2.5 void GoalieBrain::run () [inline]

The [Brain](#) thread run method. It causes the [Goalie](#) to exhibit soccer behaviors.

Postcondition

[Goalie](#) will perform [Goalie](#) functions during match.

4.9.2.6 void GoalieBrain::setActions (Action actions) [inline]**Parameters**

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

4.9.2.7 void GoalieBrain::setDefensive () [inline]

Sets the player mode to defensive

4.9.2.8 void GoalieBrain::setMarked_team (String marked_team) [inline]**Parameters**

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

4.9.2.9 void GoalieBrain::setMarked_unum (String marked_unum) [inline]**Parameters**

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

4.9.2.10 void GoalieBrain::setOffensive () [inline]

Sets the player mode to be offensive

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

- GoalieBrain.java

4.10 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](#) [getNextPlayerPoint](#) ([ObjPlayer](#) player)
- 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)

4.10.1 Member Function Documentation

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

The Effective Dash Power

Parameters

<i>effort</i>	From the stamina in the SenseMemory
<i>power</i>	The Power of the dash

Returns

the product of effort x power x dash_power_rate (0.006)

4.10.1.2 `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

4.10.1.3 `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

4.10.1.4 `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

4.10.1.5 `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

4.10.1.6 Polar MathHelp::getNextPlayerPoint ([ObjPlayer](#) *player*) [inline]

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

Parameters

<i>opponent</i>	An ObjPlayer object representing the opponent to track
-----------------	--

Returns

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

4.10.1.7 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

<i>p</i>	the Cartesian vector
----------	----------------------

Returns

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

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

Cartesian to polar converter

Parameters

<i>x</i>	the x coordinate of the Cartesian vector
<i>y</i>	the y coordinate of the Cartesian vector

Returns

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

4.10.1.9 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

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

[Polar](#) to Cartesian converter

Parameters

r	the length of the Polar 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

4.10.1.11 double MathHelp::mag (Pos p) [inline]

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

Parameters

p	the Pos of the vector
-----	---------------------------------------

Returns

A double containing the magnitude of the vector

4.10.1.12 Pos MathHelp::norm (Pos p) [inline]

A normalizer

Parameters

p	the vector to find the normal of
-----	----------------------------------

Returns

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

4.10.1.13 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

4.10.1.14 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

4.10.1.15 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

4.10.1.16 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

4.10.1.17 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](#)

4.11 Memory Class Reference

Public Member Functions

- [Memory](#) ()
- void [setField](#) (String [side](#))
- [ObjInfo](#) [getObj](#) (int *i*)
- int [getObjMemorySize](#) ()
- boolean [isObjVisible](#) (String *name*)
- [ObjBall](#) [getBall](#) ()
- [Pos](#) [getBallPos](#) ([ObjBall](#) *b*)
- [ObjFlag](#) [getFlag](#) (String *name*)
- [ObjGoal](#) [getOppGoal](#) ()
- [Pos](#) [getOppGoalPos](#) ()
- [ObjGoal](#) [getOwnGoal](#) ()
- [Pos](#) [getOwnGoalPos](#) ()
- [ObjPlayer](#) [getPlayer](#) ()
- [ObjLine](#) [getLine](#) ()
- boolean [timeCheck](#) (int *t*)
- ArrayList< [ObjPlayer](#) > [getPlayers](#) ()
- void [getPlayerArrays](#) ()
- [ObjLine](#) [getClosestLine](#) ()
- double [getDirection](#) ()
- [Polar](#) [getAbsPolar](#) ([Pos](#) *pt*)
- void [setLocation](#) (double *x*, double *y*)

- [ObjFlag](#) [getClosestFlag](#) ()
- [ObjFlag](#) [getClosestBoundary](#) ()
- [ObjFlag](#) [getClosestPenaltyFlag](#) ()
- [Pos](#) [getFlagPos](#) (String flagName)
- [Pos](#) [getPosition](#) ()
- void [setCurrent](#) ()
- 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**
- [Pos](#) **current** = new [Pos](#)()
- boolean **isHome** = true
- ArrayList< [ObjPlayer](#) > **teammates** = new ArrayList<[ObjPlayer](#)>()
- ArrayList< [ObjPlayer](#) > **opponents** = new ArrayList<[ObjPlayer](#)>()
- [ObjMemory](#) **ObjMem**
- [SenseMemory](#) **SenMem**
- String **playMode**
- String **oppSide**
- String **side**
- int **uNum**
- [Pos](#) **oppGoal**

4.11.1 Constructor & Destructor Documentation

4.11.1.1 [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.

4.11.2 Member Function Documentation

4.11.2.1 Polar Memory::getAbsPolar (Pos *pt*) [inline]

A method to convert a cartesian coordinate to polar with the global angle

Parameters

<i>pt</i>	the cartesian coordinate to convert
-----------	-------------------------------------

Returns

a polar with the global angle

4.11.2.2 double Memory::getAmountOfSpeed () [inline]

The getter for the magnitude of the Player's velocity

4.11.2.3 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

4.11.2.4 Pos Memory::getBallPos (ObjBall *b*) [inline]

Gets the cartesian coordiantes of the ball

Parameters

<i>b</i>	the ball
----------	----------

Returns

the cartesian coordiantes of the ball

4.11.2.5 **ObjFlag Memory::getClosestBoundary** () [inline]

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

Returns

closest boundary

4.11.2.6 **ObjFlag Memory::getClosestFlag** () [inline]

Finds the closest flag in your sight

Returns

[ObjFlag](#) containing closest flag

4.11.2.7 **ObjLine Memory::getClosestLine** () [inline]

This gets the closest line in your sight

Returns

line

4.11.2.8 **ObjFlag Memory::getClosestPenaltyFlag** () [inline]

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

Returns

closest penalty box flag

4.11.2.9 **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

4.11.2.10 **double Memory::getDirectionOfSpeed** () [inline]

The getter for the direction of the Player's velocity

4.11.2.11 `double Memory::getEffort () [inline]`

The getter for the Player's stamina effort

4.11.2.12 `ObjFlag Memory::getFlag (String name) [inline]`

The Flag Getter

If you're looking for a specific flag, this is you're 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

4.11.2.13 `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

4.11.2.14 `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)

4.11.2.15 `ObjLine Memory::getLine () [inline]`

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

Returns

[ObjLine](#)

4.11.2.16 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

4.11.2.17 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 ObjInfo 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

4.11.2.18 int Memory::getObjMemorySize () [inline]

The [ObjMemory](#) size

A getter to quickly retrieve the number of [ObjInfo](#) in [ObjMemory](#)

Returns

size of [ObjMemory](#)

4.11.2.19 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

4.11.2.20 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

4.11.2.21 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

4.11.2.22 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

4.11.2.23 ObjPlayer Memory::getPlayer () [inline]

The [Player](#) Getter

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

Returns

[ObjPlayer](#)

4.11.2.24 `ArrayList<ObjPlayer> Memory::getPlayers ()` [inline]

Gets an ArrayList with all of the Players in your sight

Returns

players

4.11.2.25 `String Memory::getPlayMode ()` [inline]

The getter for the game's current play mode

4.11.2.26 `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

4.11.2.27 `double Memory::getRecovery ()` [inline]

The getter for the Player's stamina recovery

4.11.2.28 `double Memory::getStamina ()` [inline]

The getter for the Player's stamina

4.11.2.29 `boolean Memory::isObjVisible (String name)` [inline]

Is this [ObjInfo](#) visible?

Parameters

<i>name</i>	the ObjName of the ObjInfo 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

4.11.2.30 `void Memory::setCurrent ()` [inline]

Sets the current position of the player

4.11.2.31 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

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

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

Parameters

<i>x</i>	
<i>y</i>	

4.11.2.33 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

<i>t</i>	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 <= the local time.

4.11.3 Member Data Documentation

4.11.3.1 `ObjMemory Memory::ObjMem`

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

4.11.3.2 `Pos Memory::oppGoal`

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

4.11.3.3 `String Memory::oppSide`

The string of the opponents side

4.11.3.4 `String Memory::playMode`

The play mode as told by the referee

4.11.3.5 `SenseMemory Memory::SenMem`

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

4.11.3.6 `String Memory::side`

The String of the player's side

4.11.3.7 `int Memory::uNum`

The player's uniform number

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

- [Memory.java](#)

4.12 Mode Class Reference

Public Member Functions

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

4.12.1 Detailed Description

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

4.12.2 Constructor & Destructor Documentation

4.12.2.1 `Mode::Mode (String modename, double timeinmode)` `[inline]`

Parameters

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

4.12.3 Member Function Documentation

4.12.3.1 `String Mode::getModename ()` `[inline]`

Returns

the modename

4.12.3.2 `double Mode::getTimeinmode ()` `[inline]`

Returns

the timeinmode

4.12.3.3 `void Mode::setModename (String modename)` `[inline]`

Parameters

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

4.12.3.4 `void Mode::setTimeinmode (double timeinmode)` `[inline]`

Parameters

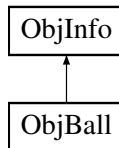
<i>timeinmode</i>	the timeinmode to set
-------------------	-----------------------

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

- [Mode.java](#)

4.13 ObjBall Class Reference

Inheritance diagram for ObjBall:



4.13.1 Detailed Description

container for the ball [ObjInfo](#),

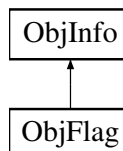
container for the flag [ObjInfo](#),

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

- [ObjInfo.java](#)

4.14 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)

4.14.1 Constructor & Destructor Documentation

4.14.1.1 ObjFlag::ObjFlag (String *name*) [inline]

Constructor of flag with flag name

4.14.2 Member Function Documentation

4.14.2.1 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)

4.14.2.2 String ObjFlag::getFlagType () [inline]

The Flag Type getter

Returns

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

4.14.2.3 String ObjFlag::getX_pos () [inline]

The X position getter

Returns

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

4.14.2.4 String ObjFlag::getY_pos () [inline]

The Y position getter

Returns

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

4.14.2.5 String ObjFlag::getYard () [inline]

The yard getter

Returns

the yard is a String of a number for boundaries

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

The Flag Name setter

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

The Flag Type setter

4.14.2.8 void ObjFlag::setX_pos (String *x_pos*) [inline]

The X position setter

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

The Y position setter

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

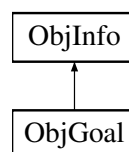
The yard setter

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

- [ObjInfo.java](#)

4.15 ObjGoal Class Reference

Inheritance diagram for ObjGoal:



4.15.1 Detailed Description

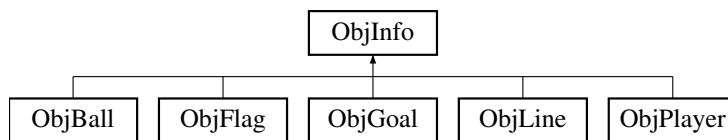
container for the goal [ObjInfo](#),

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

- [ObjInfo.java](#)

4.16 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)

4.16.1 Detailed Description

A container for items in the Player's vision

4.16.2 Constructor & Destructor Documentation

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

The Default constructor

4.16.2.2 `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 ObjInfo , either ball, player, goal, line, or flag
-------------	--

4.16.3 Member Function Documentation

4.16.3.1 `double ObjInfo::getDirChng ()` [inline]

The direction change getter

Returns

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

4.16.3.2 `double ObjInfo::getDirection ()` [inline]

The direction getter

Returns

the approximate direction of [ObjInfo](#)

4.16.3.3 `double ObjInfo::getDistance ()` [inline]

The distance getter

Returns

the approximate distance to the object

4.16.3.4 `double ObjInfo::getDistChng ()` [inline]

The distance change getter

Returns

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

4.16.3.5 `String ObjInfo::getObjName ()` [inline]

The ObjName getter

4.16.3.6 `String ObjInfo::getSide () [inline]`

The side getter

4.16.3.7 `void ObjInfo::setDirChng (double dirChng) [inline]`

The distance change setter

4.16.3.8 `void ObjInfo::setDirection (double direction) [inline]`

The direction setter

4.16.3.9 `void ObjInfo::setDistance (double distance) [inline]`

The distance setter

4.16.3.10 `void ObjInfo::setDistChng (double distChng) [inline]`

The distance change setter

4.16.3.11 `void ObjInfo::setObjName (String name) [inline]`

The ObjName setter

4.16.3.12 `void ObjInfo::setSide (String objSide) [inline]`

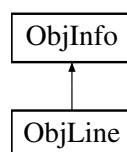
The side setter

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

- [ObjInfo.java](#)

4.17 ObjLine Class Reference

Inheritance diagram for ObjLine:



4.17.1 Detailed Description

container for line [ObjInfo](#)

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

- [ObjInfo.java](#)

4.18 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*

4.18.1 Detailed Description

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

4.18.2 Constructor & Destructor Documentation

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

Default constructor

This initializes the time to 0

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

[ObjMemory](#) constructor

Parameters

<i>ObjArray</i>	the ArrayList containing all the ObjInfos 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 available to add to the [Memory](#)

4.18.3 Member Function Documentation**4.18.3.1 void ObjMemory::addInfo (ObjInfo *newInfo*) [inline]**

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

Parameters

<i>newInfo</i>	the ObjInfo 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

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

An accessor of individual [ObjInfo](#)

Parameters

<i>index</i>	the index of the ObjInfo 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

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

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

Parameters

<i>name</i>	the ObjName of the ObjInfo 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.

4.18.3.4 int ObjMemory::getSize () [inline]

Returns the size of the ObjArray

4.18.3.5 int ObjMemory::getTime () [inline]

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

4.18.3.6 void ObjMemory::setTime (int t) [inline]

The time setter

Parameters

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

Postcondition

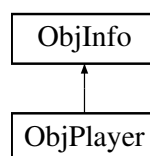
the time will be set and ready to access

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

- [ObjMemory.java](#)

4.19 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)

4.19.1 Detailed Description

container for player [ObjInfo](#)

4.19.2 Member Function Documentation

4.19.2.1 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.

4.19.2.2 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.

4.19.2.3 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

4.19.2.4 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

4.19.2.5 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

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

The body direction setter

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

The goalie check setter

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

The head direction setter

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

The Team Name setter

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

The Uniform Number getter

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

- [ObjInfo.java](#)

4.20 Parser Class Reference

Public Member Functions

- [Parser \(\)](#)

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

Public Attributes

- String [input](#)

4.20.1 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.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 `Parser::Parser ()` [inline]

Default constructor

4.20.3 Member Function Documentation

4.20.3.1 `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."

4.20.3.2 `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 Memory 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

4.20.4 Member Data Documentation**4.20.4.1 String Parser::input**

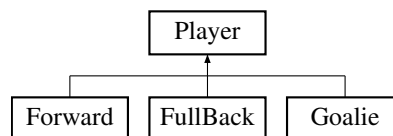
The String of the incoming message

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

- [Parser.java](#)

4.21 Player Class Reference

Inheritance diagram for Player:

**Public Member Functions**

- **Player** (String team)
- **Player** ([RoboClient](#) rc, [Memory](#) m, [ObjInfo](#) i, [Parser](#) p, int time)
- **Action** [getAction](#) ()
- void **setAction** ([Action](#) a)
- void **setHome** ([Pos](#) home)
- [Pos](#) [getHome](#) ()
- [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 [getMemTime](#) ()
- int [getTime](#) ()
- void [setTime](#) (int time)
- double [getDirection](#) ()
- [Pos](#) [getPosition](#) ()
- void [initPlayer](#) (double x, double y, String pos) throws [SocketException](#), [UnknownHostException](#)
- void [initPlayer](#) (double x, double y) 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 [dash](#) (double power, double direction) throws [Exception](#)
- void [turn](#) (double moment) throws [UnknownHostException](#), [InterruptedException](#)
- void [turn_neck](#) (double moment) throws [UnknownHostException](#), [InterruptedException](#)
- void [recievePass](#) ([ObjBall](#) ball, [ObjPlayer](#) p) 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

- [MathHelp](#) **mh** = new [MathHelp](#)()
- boolean **wait** = true
- String **position** = "left"

Protected Attributes

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

4.21.1 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).

4.21.2 Constructor & Destructor Documentation

4.21.2.1 `Player::Player (RoboClient rc, Memory m, ObjInfo i, Parser p, int time)`
`[inline]`

Parameters

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

4.21.3 Member Function Documentation

4.21.3.1 `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>	

4.21.3.2 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.

4.21.3.3 void Player::dash (double *power*, double *direction*) throws Exception [inline]

Causes [Player](#) to dash.

Parameters

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

Exceptions

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

Precondition

Play mode is play_on.

Postcondition

The player has dashed at the given power.

4.21.3.4 double Player::getDirection () [inline]

Returns the direction of the player

4.21.3.5 Pos Player::getHome () [inline]**Returns**

the home coordinates of the player

4.21.3.6 Memory Player::getMem () [inline]**Returns**

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

4.21.3.7 int Player::getMemTime () [inline]**Returns**

the time in the [ObjMemory](#)

4.21.3.8 ObjInfo Player::getObjInfo () [inline]**Returns**

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

4.21.3.9 Parser Player::getParser () [inline]**Returns**

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

4.21.3.10 Pos Player::getPosition () [inline]

Returns the current absolute coordinates of the player.

Returns

[Pos](#)

4.21.3.11 RoboClient Player::getRoboClient () [inline]**Returns**

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

4.21.3.12 int Player::getTime () [inline]

Returns the current player time.

Returns

the time

4.21.3.13 `void Player::initPlayer (double x, double y, String pos) 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.

4.21.3.14 `void Player::initPlayer (double x, double y) 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.

4.21.3.15 `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 to 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.

4.21.3.16 void Player::markOpponent (String *team*, String *number*) [inline]

Marks opposing players for defense

4.21.3.17 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.

4.21.3.18 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](#).

4.21.3.19 void Player::recievePass (ObjBall *ball*, ObjPlayer *p*) throws UnknownHostException, InterruptedException [inline]

Instructs the player to prepare to receive a pass from another teammate.

Parameters

<i>ball</i>	The ball in play.
<i>p</i>	The player to receive the ball from.

Precondition

Playmode is play_on, ball is being passed to player.

Postcondition

The player has possession of the ball.

4.21.3.20 `void Player::runDefense () throws UnknownHostException, InterruptedException`
[inline]

Follows opposing players on defense (Currently unused)

Reimplemented in [FullBack](#).

4.21.3.21 `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.

4.21.3.22 `void Player::setHome (Pos home)` [inline]

Sets the home coordinates of the player

4.21.3.23 `void Player::setMem (Memory m)` [inline]

Parameters

<i>m</i>	The Memory to set.
----------	------------------------------------

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

<i>i</i>	The ObjInfo to set.
----------	-------------------------------------

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

Sets the parser for the player.

Parameters

<i>p</i>	The Parser to set.
----------	------------------------------------

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

<i>rc</i>	The RoboClient to set.
-----------	--

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

Sets the current time for the player.

Parameters

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

4.21.3.28 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.

4.21.3.29 void `Player::turn_neck` (double *moment*) throws `UnknownHostException`,
`InterruptedException` [inline]

Turns the neck of the player

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

- [Player.java](#)

4.22 Polar Class Reference

Public Member Functions

- [Polar](#) ()
- [Polar](#) (double r, double t)
- void **print** (String a)
- void **print** ()

Public Attributes

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

4.22.1 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

4.22.2 Constructor & Destructor Documentation

4.22.2.1 `Polar::Polar` () [inline]

Default constructor

Postcondition

initializes distance and angle to 0.0

4.22.2.2 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

4.23 Pos Class Reference

Public Member Functions

- [Pos](#) ()
- [Pos](#) (String name, double x, double y)
- [Pos](#) (double x, double y)
- void **print** (String a)
- void **print** ()

Public Attributes

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

4.23.1 Detailed Description

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

4.23.2 Constructor & Destructor Documentation

4.23.2.1 Pos::Pos () [inline]

Default constructor

Postcondition

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

4.23.2.2 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 Pos , for easier searching
<i>x</i>	x-coordinate
<i>y</i>	y-coordinate

4.23.2.3 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](#)

4.24 RoboClient Class Reference**Public Member Functions**

- [RoboClient](#) (int port)
- [RoboClient](#) (String team)
- String [getTeam](#) ()
- void [setTeam](#) (String team)
- void [send](#) (String message) throws UnknownHostException
- String [receive](#) ()
- void [init](#) ([Parser](#) p, [Memory](#) m) throws UnknownHostException
- void [changePlayMode](#) (String playmode) throws UnknownHostException
- void [initGoalie](#) ([Parser](#) p, [Memory](#) m) throws UnknownHostException
- void [dash](#) (double power) throws Exception
- void [dash](#) (double power, double direction) throws Exception
- void [kick](#) (double power, double dir) throws UnknownHostException
- void [turn](#) (double moment) throws UnknownHostException
- void [turn_neck](#) (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**

4.24.1 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.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 RoboClient::RoboClient (int *port*) [inline]

Parameters

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

4.24.2.2 RoboClient::RoboClient (String *team*) [inline]

Parameters

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

4.24.3 Member Function Documentation

4.24.3.1 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>	
-----------------------------	--

4.24.3.2 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

4.24.3.3 void RoboClient::dash (double *power*, double *direction*) throws Exception [inline]

This function sends the dash command to the server.

Parameters

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

Precondition

The RoboCup server is available, client has been initialized.

Postcondition

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

Returns

None

4.24.3.4 String RoboClient::getTeam () [inline]**Returns**

the team

4.24.3.5 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.

4.24.3.6 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

4.24.3.7 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

4.24.3.8 `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

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

4.24.3.9 `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

4.24.3.10 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>	
-----------------------------	--

4.24.3.11 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

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

Parameters

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

4.24.3.13 `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](#)

4.25 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**

4.25.1 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.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 `SenseMemory::SenseMemory ()` `[inline]`

Default constructor

Postcondition

initializes time to 0

4.25.2.2 `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

4.25.3 Member Function Documentation

4.25.3.1 `int SenseMemory::getTime ()` `[inline]`

The time getter

Returns

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

4.25.3.2 `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
-------------------	---

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

The time setter

Parameters

<i>t</i>	the time hat the SenseMemory 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

Actions for the player to use

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 *

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](#)
 - closestPlayer, [8](#)
 - dribbleToGoal, [8](#)
 - findBall, [9](#)
 - FullBack_findBall, [9](#)
 - getTurn, [9](#)
 - goHome, [10](#)
 - gotoPoint, [10](#)
 - gotoSidePoint, [11](#)
 - kickToPoint, [11](#)
 - passBall, [11](#)
 - setMem, [12](#)
- Action.java, [77](#)
- addInfo
 - ObjMemory, [51](#)
- ballInGoalzone
 - Goalie, [22](#)
- Brain, [12](#)
 - Brain, [13](#)
 - getCurrentMode, [13](#)
 - getMarked_team, [13](#)
 - getMarked_unum, [13](#)
 - setDefensive, [13](#)
 - setMarked_team, [13](#)
 - setMarked_unum, [14](#)
 - setOffensive, [14](#)
- Brain.java, [77](#)
- catchable
 - Goalie, [22](#)
- catchball
 - Goalie, [22](#)
 - RoboClient, [68](#)
- closestOpponent
 - Player, [58](#)
- closestPlayer
 - Action, [8](#)
 - FullBack, [17](#)
 - Goalie, [22](#)
- dash
 - Player, [58, 59](#)
 - RoboClient, [69](#)
- defendGoal
 - Goalie, [23](#)
- dribbleToGoal
 - Action, [8](#)
- edp
 - MathHelp, [28](#)
- Field, [14](#)
 - Field, [15](#)
- Field.java, [78](#)
- findBall
 - Action, [9](#)
- followBall
 - Goalie, [23](#)
- Forward, [15](#)
- Forward.java, [78](#)
- FullBack, [15](#)
 - closestPlayer, [17](#)
 - FullBack, [16](#)
 - initFullBack, [17](#)
 - runDefense, [17](#)
- FullBack.java, [78](#)
- FullBack_findBall
 - Action, [9](#)
- FullBackBrain, [18](#)
 - FullBackBrain, [18](#)
 - getActions, [19](#)
 - getCurrentMode, [19](#)
 - getMarked_team, [19](#)
 - getMarked_unum, [19](#)
 - run, [19](#)
 - setActions, [19](#)
 - setDefensive, [19](#)
 - setMarked_team, [19](#)
 - setMarked_unum, [20](#)
 - setOffensive, [20](#)
- Game, [20](#)

- Game.java, 79
- getAbsPolar
 - Memory, 35
- getActions
 - FullBackBrain, 19
 - GoalieBrain, 26
- getAmountOfSpeed
 - Memory, 35
- getBall
 - Memory, 35
- getBallPos
 - Memory, 35
- getBodyDir
 - ObjPlayer, 53
- getBtwBallAndGoal
 - Goalie, 23
- getClosestBoundary
 - Memory, 35
- getClosestFlag
 - Memory, 36
- getClosestLine
 - Memory, 36
- getClosestPenaltyFlag
 - Memory, 36
- getCurrentMode
 - Brain, 13
 - FullBackBrain, 19
 - GoalieBrain, 26
- getDashPower
 - MathHelp, 28
- getDirChng
 - ObjInfo, 48
- getDirection
 - Memory, 36
 - ObjInfo, 48
 - Player, 59
- getDirectionOfSpeed
 - Memory, 36
- getDistance
 - ObjInfo, 48
- getDistChng
 - ObjInfo, 48
- getEffort
 - Memory, 36
- getFlag
 - Memory, 37
- getFlagName
 - ObjFlag, 45
- getFlagPos
 - Memory, 37
- getFlagType
 - ObjFlag, 45
- getHeadDir
 - ObjPlayer, 53
- getHeadDirection
 - Memory, 37
- getHome
 - Player, 59
- getKickPower
 - MathHelp, 29
- getLine
 - Memory, 37
- getMarked_team
 - Brain, 13
 - FullBackBrain, 19
 - GoalieBrain, 26
- getMarked_unum
 - Brain, 13
 - FullBackBrain, 19
 - GoalieBrain, 26
- getMem
 - Player, 59
- getMemTime
 - Player, 60
- getModename
 - Mode, 43
- getNextBallPoint
 - MathHelp, 29
- getNextPlayerPoint
 - MathHelp, 30
- getNullGoalAngle
 - Memory, 37
- getObj
 - Memory, 38
 - ObjMemory, 51
- getObjInfo
 - Player, 60
- getObjMemorySize
 - Memory, 38
- getObjName
 - ObjInfo, 48
- getOppGoal
 - Memory, 38
- getOppGoalPos
 - Memory, 39
- getOwnGoal
 - Memory, 39
- getOwnGoalPos
 - Memory, 39
- getParser

- Player, 60
- getPlayer
 - Memory, 39
- getPlayers
 - Memory, 39
- getPlayMode
 - Memory, 40
- getPolar
 - MathHelp, 30
- getPos
 - MathHelp, 30, 31
- getPosition
 - Memory, 40
 - Player, 60
- getRecovery
 - Memory, 40
- getRoboClient
 - Player, 60
- getSide
 - ObjInfo, 48
- getSize
 - ObjMemory, 52
- getStamina
 - Memory, 40
- getTeam
 - ObjPlayer, 53
 - RoboClient, 69
- getTime
 - ObjMemory, 52
 - Player, 60
 - SenseMemory, 74
- getTimeinmode
 - Mode, 43
- getTurn
 - Action, 9
- getuNum
 - ObjPlayer, 53
- getX_pos
 - ObjFlag, 45
- getY_pos
 - ObjFlag, 45
- getYard
 - ObjFlag, 45
- Goalie, 20
 - ballInGoalzone, 22
 - catchable, 22
 - catchball, 22
 - closestPlayer, 22
 - defendGoal, 23
 - followBall, 23
 - getBtwBallAndGoal, 23
 - initGoalie, 24
 - kickBallOutOfBounds, 24
 - kickToPlayer, 24
 - positionGoalie, 25
- Goalie.java, 79
- GoalieBrain, 25
 - getActions, 26
 - getCurrentMode, 26
 - getMarked_team, 26
 - getMarked_unum, 26
 - GoalieBrain, 26
 - run, 26
 - setActions, 27
 - setDefensive, 27
 - setMarked_team, 27
 - setMarked_unum, 27
 - setOffensive, 27
- goHome
 - Action, 10
- gotoPoint
 - Action, 10
- gotoSidePoint
 - Action, 11
- init
 - RoboClient, 70
- initFullBack
 - FullBack, 17
- initGoalie
 - Goalie, 24
 - RoboClient, 70
- initParse
 - Parser, 55
- initPlayer
 - Player, 60, 61
- input
 - Parser, 56
- isGoalie
 - ObjPlayer, 54
- isObjVisible
 - Memory, 40
- kick
 - Player, 61
 - RoboClient, 70
- kickBallOutOfBounds
 - Goalie, 24
- kickToPlayer
 - Goalie, 24

- kickToPoint
 - Action, [11](#)
- mag
 - MathHelp, [31](#)
- markOpponent
 - Player, [61](#)
- MathHelp, [27](#)
 - edp, [28](#)
 - getDashPower, [28](#)
 - getKickPower, [29](#)
 - getNextBallPoint, [29](#)
 - getNextPlayerPoint, [30](#)
 - getPolar, [30](#)
 - getPos, [30](#), [31](#)
 - mag, [31](#)
 - norm, [31](#)
 - vAdd, [32](#)
 - vDiv, [32](#)
 - vMul, [32](#)
 - vSub, [33](#)
- MathHelp.java, [80](#)
- Memory, [33](#)
 - getAbsPolar, [35](#)
 - getAmountOfSpeed, [35](#)
 - getBall, [35](#)
 - getBallPos, [35](#)
 - getClosestBoundary, [35](#)
 - getClosestFlag, [36](#)
 - getClosestLine, [36](#)
 - getClosestPenaltyFlag, [36](#)
 - getDirection, [36](#)
 - getDirectionOfSpeed, [36](#)
 - getEffort, [36](#)
 - getFlag, [37](#)
 - getFlagPos, [37](#)
 - getHeadDirection, [37](#)
 - getLine, [37](#)
 - getNullGoalAngle, [37](#)
 - getObj, [38](#)
 - getObjMemorySize, [38](#)
 - getOppGoal, [38](#)
 - getOppGoalPos, [39](#)
 - getOwnGoal, [39](#)
 - getOwnGoalPos, [39](#)
 - getPlayer, [39](#)
 - getPlayers, [39](#)
 - getPlayMode, [40](#)
 - getPosition, [40](#)
 - getRecovery, [40](#)
 - getStamina, [40](#)
 - isObjVisible, [40](#)
 - Memory, [34](#)
 - ObjMem, [42](#)
 - oppGoal, [42](#)
 - oppSide, [42](#)
 - playMode, [42](#)
 - SenMem, [42](#)
 - setCurrent, [40](#)
 - setField, [40](#)
 - setLocation, [41](#)
 - side, [42](#)
 - timeCheck, [41](#)
 - uNum, [42](#)
- Memory.java, [80](#)
- Mode, [42](#)
 - getModename, [43](#)
 - getTimeinmode, [43](#)
- Mode, [43](#)
 - setModename, [43](#)
 - setTimeinmode, [43](#)
- Mode.java, [80](#)
- move
 - Player, [62](#)
 - RoboClient, [71](#)
- norm
 - MathHelp, [31](#)
- ObjBall, [44](#)
- ObjFlag, [44](#)
 - getFlagName, [45](#)
 - getFlagType, [45](#)
 - getX_pos, [45](#)
 - getY_pos, [45](#)
 - getYard, [45](#)
 - ObjFlag, [45](#)
 - setFlagName, [46](#)
 - setFlagType, [46](#)
 - setX_pos, [46](#)
 - setY_pos, [46](#)
 - setYard, [46](#)
- ObjGoal, [46](#)
- ObjInfo, [47](#)
 - getDirChng, [48](#)
 - getDirection, [48](#)
 - getDistance, [48](#)
 - getDistChng, [48](#)
 - getObjName, [48](#)
 - getSide, [48](#)

- ObjInfo, 47
 - setDirChng, 49
 - setDirection, 49
 - setDistance, 49
 - setDistChng, 49
 - setObjName, 49
 - setSide, 49
- ObjInfo.java, 81
- ObjLine, 49
- ObjMem
 - Memory, 42
- ObjMemory, 50
 - addInfo, 51
 - getObj, 51
 - getSize, 52
 - getTime, 52
 - ObjMemory, 50
 - setTime, 52
- ObjMemory.java, 81
- ObjPlayer, 52
 - getBodyDir, 53
 - getHeadDir, 53
 - getTeam, 53
 - getuNum, 53
 - isGoalie, 54
 - setBodyDir, 54
 - setGoalie, 54
 - setHeadDir, 54
 - setTeam, 54
 - setuNum, 54
- oppGoal
 - Memory, 42
- oppSide
 - Memory, 42
- Parse
 - Parser, 55
- Parser, 54
 - initParse, 55
 - input, 56
 - Parse, 55
 - Parser, 55
- Parser.java, 82
- passBall
 - Action, 11
- Player, 56
 - closestOpponent, 58
 - dash, 58, 59
 - getDirection, 59
 - getHome, 59
 - getMem, 59
 - getMemTime, 60
 - getObjInfo, 60
 - getParser, 60
 - getPosition, 60
 - getRoboClient, 60
 - getTime, 60
 - initPlayer, 60, 61
 - kick, 61
 - markOpponent, 61
 - move, 62
 - Player, 58
 - receiveInput, 62
 - recievePass, 62
 - runDefense, 63
 - say, 63
 - setHome, 63
 - setMem, 63
 - setObjInfo, 63
 - setParser, 64
 - setRoboclient, 64
 - setTime, 64
 - turn, 64
 - turn_neck, 64
- Player.java, 82
- playMode
 - Memory, 42
- Polar, 65
 - Polar, 65
- Pos, 66
 - Pos, 66, 67
- Pos.java, 83
- positionGoalie
 - Goalie, 25
- receive
 - RoboClient, 71
- receiveInput
 - Player, 62
- recievePass
 - Player, 62
- RoboClient, 67
 - catchball, 68
 - dash, 69
 - getTeam, 69
 - init, 70
 - initGoalie, 70
 - kick, 70
 - move, 71
 - receive, 71

- RoboClient, 68
- say, 71
- send, 72
- setTeam, 72
- turn, 72
- RoboClient.java, 83
- run
 - FullBackBrain, 19
 - GoalieBrain, 26
- runDefense
 - FullBack, 17
 - Player, 63
- say
 - Player, 63
 - RoboClient, 71
- send
 - RoboClient, 72
- SenMem
 - Memory, 42
- SenseMemory, 73
 - getTime, 74
 - SenseMemory, 74
 - setTime, 74
- SenseMemory.java, 84
- setActions
 - FullBackBrain, 19
 - GoalieBrain, 27
- setBodyDir
 - ObjPlayer, 54
- setCurrent
 - Memory, 40
- setDefensive
 - Brain, 13
 - FullBackBrain, 19
 - GoalieBrain, 27
- setDirChng
 - ObjInfo, 49
- setDirection
 - ObjInfo, 49
- setDistance
 - ObjInfo, 49
- setDistChng
 - ObjInfo, 49
- setField
 - Memory, 40
- setFlagName
 - ObjFlag, 46
- setFlagType
 - ObjFlag, 46
- setGoalie
 - ObjPlayer, 54
- setHeadDir
 - ObjPlayer, 54
- setHome
 - Player, 63
- setLocation
 - Memory, 41
- setMarked_team
 - Brain, 13
 - FullBackBrain, 19
 - GoalieBrain, 27
- setMarked_unum
 - Brain, 14
 - FullBackBrain, 20
 - GoalieBrain, 27
- setMem
 - Action, 12
 - Player, 63
- setModename
 - Mode, 43
- setObjInfo
 - Player, 63
- setObjName
 - ObjInfo, 49
- setOffensive
 - Brain, 14
 - FullBackBrain, 20
 - GoalieBrain, 27
- setParser
 - Player, 64
- setRoboclient
 - Player, 64
- setSide
 - ObjInfo, 49
- setTeam
 - ObjPlayer, 54
 - RoboClient, 72
- setTime
 - ObjMemory, 52
 - Player, 64
 - SenseMemory, 74
- setTimeinmode
 - Mode, 43
- setuNum
 - ObjPlayer, 54
- setX_pos
 - ObjFlag, 46
- setY_pos
 - ObjFlag, 46

setYard
 ObjFlag, [46](#)
side
 Memory, [42](#)

timeCheck
 Memory, [41](#)
turn
 Player, [64](#)
 RoboClient, [72](#)
turn_neck
 Player, [64](#)

uNum
 Memory, [42](#)

vAdd
 MathHelp, [32](#)
vDiv
 MathHelp, [32](#)
vMul
 MathHelp, [32](#)
vSub
 MathHelp, [33](#)