Zappy AI

Generated by Doxygen 1.9.6

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 Action.Action Class Reference

Inheritance diagram for Action.Action:

Collaboration diagram for Action.Action:

Static Public Attributes

- str **FORWARD** = "Forward"
- str **RIGHT** = "Right"
- str **LEFT** = "Left"
- str **LOOK** = "Look"
- str **INVENTORY** = "Inventory"
- str **BROADCAST** = "Broadcast"
- str **CONNECT_NBR** = "Connect_nbr"
- str **FORK** = "Fork"
- str **EJECT** = "Eject"
- str **TAKE** = "Take"
- str **SET** = "Set"
- str **INCANTATION** = "Incantation"
- str **NONE** = "None"

3.1.1 Detailed Description

```
Action class
A class to list the actions the player can do
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Enum/Action.py

3.2 AI.AI Class Reference

Public Member Functions

- def __init__ (self, host, port, teamName)
- def serverCommunicationInThread (self)
- def run (self)

Public Attributes

- **api**
- **player**
- **teamName**
- **threads**

3.2.1 Detailed Description

```
AI class
A class to handle the AI of the Zappy project

Attributes :
    api : API
        the API to communicate with the server
    player : Player
        the player
    teamName : str
        the name of the team

----------

Methods :
    __init__(host : str, port : int, teamName : str)
        Constructor of the AI class
    run()
        Run the AI
```

3.2.2 Constructor & Destructor Documentation

3.2.2.1 __init__()

```
def AI.AI.__init__ (
              self,
              host,
              port,
              teamName )


Constructor of the AI class
Assign the API, the player and the team name

Parameters :
    host : str
        the host of the server
    port : int
        the port of the server
    teamName : str
        the name of the team
```

3.2.3 Member Function Documentation

3.2.3.1 run()

```
def AI.AI.run (
              self )
```

```
Run the AI (the main loop)
Connect to the server, initialize the connection and start the main loop

The main loop is an infinite loop that will select an action for the player
and send it to the server and after that, it will receive the response from the server
and handle it
```

3.2.3.2 serverCommunicationInThread()

```
def AI.AI.serverCommunicationInThread (
              self )
```

```
Handle the communication with the server in a thread
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/AI.py

3.3 API.API Class Reference

Public Member Functions

- def __init__ (self, str host, int port)
- def connect (self)
- def sendData (self, str data, int timeout=None)
- def receiveData (self, float timeout=None)
- def initConnection (self, str teamName, str fileName="")
- def close (self)

3.3.1 Detailed Description

```
API class
A class to communicate with the server

Attributes :
    host : str
        the host of the server
    port : int
        the port of the server
    inputs : list
        the list of inputs
    outputs : list
        the list of outputs
    sock : socket
        the socket to communicate with the server

----------

Methods :
    sendData(data : str, timeout : int = None)
        send data to the server
    receiveData(timeout : int = None)
        receive data from the server
    connect(team_name : str)
        connect to the server
    close()
        close the connection
```

3.3.2 Constructor & Destructor Documentation

3.3.2.1 __init__()

```
def API.API.__init__ (
              self,
             str host,
             int port )


Constructor of the API class

Assign the host and the port of the server
Create the socket to communicate with the server
Connect to the server and add the socket to the inputs and outputs lists

Parameters :
    host : str
        the host of the server
    port : int
        the port of the server
```

3.3.3 Member Function Documentation

3.3.3.1 close()

```
def API.API.close (
              self )
```

Close the connection with the server

3.3.3.2 connect()

```
def API.API.connect (
              self )
```

Connect to the server
Add the socket to the inputs and outputs lists

3.3.3.3 initConnection()

```
def API.API.initConnection (
              self,
             str teamName,
             str  fileName = "" )
```

Function to do the first exchange with the server

Send the team name to the server
Receive the client number and the map size from the server
Print the client number and the map size

Parameters :
    team_name : str
        the name of the team
    fileName : str
        the file name of logs

Returns :
    client_num : int
        the client number
    x : int
        the x size of the map
    y : int
        the y size of the map

3.3.3.4 receiveData()

```
def API.API.receiveData (
              self,
             float  timeout = None )
```

```
Receive data from the server

Parameters :
    timeout : float
        the timeout to wait for the server to send data
        (default is None which means no timeout)
```

3.3.3.5 sendData()

```
def API.API.sendData (
              self,
             str data,
             int  timeout = None )
```

```
Send data to the server

Parameters :
    data : str
        the data to send
    timeout : int
        the timeout to wait for the server to be ready to receive data
        (default is None which means no timeout)
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Network/API.py

3.4 APIException.APIException Class Reference

Inheritance diagram for APIException.APIException:

3.5 ArgsException.ArgsException Class Reference

Inheritance diagram for ArgsException.ArgsException:

Collaboration diagram for ArgsException.ArgsException:

Public Member Functions

- def __init__ (self, message)

3.5.1 Detailed Description

```
ArgsException class

A class to handle exceptions that can occur in the Args
The ArgsException class inherits from the IError class

Attributes :
    message : str
        the message of the exception
```

3.5.2 Constructor & Destructor Documentation

3.5.2.1 __init__()

```
def ArgsException.ArgsException.__init__ (
              self,
              message )

Constructor of the ArgsException class
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Errors/ArgsException.py

3.6 IError.IError Class Reference

Inheritance diagram for IError.IError:

Collaboration diagram for IError.IError:

Public Member Functions

- def __init__ (self, message)
- def __str__ (self)
- def __repr__ (self)

Public Attributes

- **message**

3.6.1 Detailed Description

```
IError class

A class to handle errors that can occur in the project

Attributes :
    message : str
        the message of the error

----------

Methods :
    __str__()
        return the message of the error
    __repr__()
        return the message of the error
```

3.6.2 Constructor & Destructor Documentation

3.6.2.1 __init__()

```
def IError.IError.__init__ (
              self,
              message )


Constructor of the IError class

Assign the message of the error

Parameters :
    message : str
        the message of the error
```

3.6.3 Member Function Documentation

3.6.3.1 __repr__()

```
def IError.IError.__repr__ (
              self )


Return the message of the error
```

3.6.3.2 __str__()

```
def IError.IError.__str__ (
              self )
```

Return the message of the error

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Errors/IError.py

3.7 Inventory.Inventory Class Reference

EPITECH PROJECT, 2024 Zappy File description: Inventory.

Public Member Functions

- def __init__ (self, food=10, linemate=0, deraumere=0, sibur=0, mendiane=0, phiras=0, thystame=0, player=0)
- def __str__ (self)
- def toStr (self)
- def __eq__ (self, inventory)
- def __add__ (self, inventory)
- def hasMoreStones (self, "Inventory" inventory)
- def updateInventory (self, str data)
- def updateCaseContent (self, list data)
- def addAnObject (self, str ressource)
- def removeAnObject (self, str ressource)
- def countStones (self)

Public Attributes

- **food**
- **linemate**
- **deraumere**
- **sibur**
- **mendiane**
- **phiras**
- **thystame**
- **player**

3.7.1 Detailed Description

EPITECH PROJECT, 2024 Zappy File description: Inventory.

```
Inventory class
A class to handle the inventory of the player

Attributes :
    food : int
        the number of food
    linemate : int
        the number of linemate
    deraumere : int
        the number of deraumere
    sibur : int
        the number of sibur
    mendiane : int
        the number of mendiane
    phiras : int
        the number of phiras
    thystame : int
        the number of thystame
    player : int
        the number of players

----------

Methods :
    __init__()
        Constructor of the Inventory class
    __str__()
        Print the inventory
    updateInventory(data)
        Update the inventory with the data from the inventory command
    updateCaseContent(data)
        Update the case content with the data from the vision command
    addAnObject(ressource)
        Add an object to the inventory
    removeAnObject(ressource)
        Remove an object from the inventory
```

3.7.2 Constructor & Destructor Documentation

3.7.2.1 __init__()

```
def Inventory.Inventory.__init__ (
              self,
              food = 10,
              linemate = 0,
              deraumere = 0,
              sibur = 0,
              mendiane = 0,
              phiras = 0,
              thystame = 0,
              player = 0 )
```

```
Constructor of the Inventory class
```

3.7.3 Member Function Documentation

3.7.3.1 __add__()

```
def Inventory.Inventory.__add__ (
              self,
              inventory )
```

```
Add two inventories

Parameters :
    inventory : Inventory
        the inventory to add

Returns :
    Inventory
        the self inventory with the inventory added
```

3.7.3.2 __eq__()

```
def Inventory.Inventory.__eq__ (
              self,
              inventory )
```

```
Compare two inventories

Parameters :
    inventory : Inventory
        the inventory to compare with

Returns :
    bool
        True if the inventories are the same, False otherwise
```

3.7.3.3 __str__()

```
def Inventory.Inventory.__str__ (
              self )
```

```
Print the inventory
```

3.7.3.4 addAnObject()

```
def Inventory.Inventory.addAnObject (
             self,
            str ressource )
```

Add an object to the inventory

```
Parameters :
    ressource : str
        the ressource to add
```

3.7.3.5 countStones()

```
def Inventory.Inventory.countStones (
             self )
```

Count the number of different stones in a case

```
Returns :
    int
        the number of different stones in a case
```

3.7.3.6 hasMoreStones()

```
def Inventory.Inventory.hasMoreStones (
             self,
            "Inventory" inventory )
```

Check if the self inventory has more stones than the inventory

```
Parameters :
    inventory : Inventory
        the inventory to compare with

Returns :
    bool
        True if the self inventory has more stones, False otherwise
```

3.7.3.7 removeAnObject()

```
def Inventory.Inventory.removeAnObject (
             self,
            str ressource )
```

Remove an object from the inventory

```
Parameters :
    ressource : str
        the ressource to remove
```

3.7.3.8 toStr()

```
def Inventory.Inventory.toStr (
              self )


Return the inventory as a string

Returns :
    str
        the inventory as a string
```

3.7.3.9 updateCaseContent()

```
def Inventory.Inventory.updateCaseContent (
              self,
             list data )


Update the case content with the data from the vision command

Parameters :
    data : list
        the data from the vision command
```

3.7.3.10 updateInventory()

```
def Inventory.Inventory.updateInventory (
              self,
             str data )


Update the inventory with the data from the inventory command

Parameters :
    data : str
        the data from the inventory command
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Player/Inventory.py

3.8 Item.Item Class Reference

Inheritance diagram for Item.Item:

Collaboration diagram for Item.Item:

Static Public Attributes

- str **FOOD** = "food"
- str **LINEMATE** = "linemate"
- str **DERAUMERE** = "deraumere"
- str **SIBUR** = "sibur"
- str **MENDIANE** = "mendiane"
- str **PHIRAS** = "phiras"
- str **THYSTAME** = "thystame"

3.8.1 Detailed Description

```
Item class
A class to list the items in the game
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Enum/Item.py

3.9 Mode.Mode Class Reference

Inheritance diagram for Mode.Mode:

Collaboration diagram for Mode.Mode:

Static Public Attributes

- int **FOOD** = 0
- int **STONES** = 1
- int **FORKING** = 2
- int **BROADCASTING** = 3
- int **HANDLINGRESPONSE** = 4
- int **WAITING** = 5
- int **ELEVATING** = 6
- int **REGROUP** = 7
- int **DROPPING** = 8
- int **NONE** = 9

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Enum/Mode.py

3.10 Player.Mode Class Reference

Inheritance diagram for Player.Mode:

Collaboration diagram for Player.Mode:

Static Public Attributes

- int **FOOD** = 0
- int **STONES** = 1
- int **FORKING** = 2
- int **BROADCASTING** = 3
- int **HANDLINGRESPONSE** = 4
- int **WAITING** = 5
- int **ELEVATING** = 6
- int **REGROUP** = 7
- int **DROPPING** = 8
- int **NONE** = 9

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Player/Player.py

3.11 Player.Player Class Reference

Public Member Functions

- def __init__ (self)
- def __str__ (self)
- def moveForward (self, callback=None)
- def turnRight (self, callback=None)
- def turnLeft (self, callback=None)
- def look (self, callback=None)
- def cmdInventory (self, callback=None)
- def broadcast (self, str message="Hello", callback=None)
- def connectNbr (self, callback=None)
- def fork (self, callback=None)
- def eject (self, callback=None)
- def take (self, str resource="food", callback=None)
- def set (self, str resource="food", callback=None)
- def incantation (self, callback=None)
- def none (self)
- def updateVision (self, str vision)
- def updateInventory (self, str inventory)
- def updateBroadcastReceived (self, str message)
- def updateEjectionReceived (self, str message)
- def updateLevel (self, int level)
- def handleElevation (self, str response)
- def hasSomethingHappened (self, str response)
- def handleResponse (self, str response)
- def connectMissingPlayers (self)
- def completeTeam (self)
- def updateModeSlave (self)
- def updateModeLeader (self)
- def updateMode (self)
- def lookingForFood (self)
- def lookingForStones (self)
- def askSlavesForInventory (self)

- def checkIfEnoughFood (self, str response)
- def handleResponseBroadcast (self)
- def slavesReponses (self)
- def waitingEveryone (self)
- def waitingDrop (self)
- def dropping (self)
- def regroupAction (self)
- def chooseAction (self)
- def goGetItem (self, index, List[Item] itemSeek)
- def foodInVision (self, list vision)
- def stonesInVision (self, list vision)

Public Attributes

- **inventory**
- **level**
- **actions**
- **currentAction**
- **commands**
- **currentCommand**
- **callbacks**
- **currentCallback**
- **vision**
- **broadcastReceived**
- **ejectionReceived**
- **isLeader**
- **unusedSlots**
- **currentlyElevating**
- **currentMode**
- **currentFood**
- **nbSlaves**
- **waitingResponse**
- **regroupDirection**
- **arrived**
- **isTimed**
- **nbSlavesHere**
- **callback**

3.11.1 Detailed Description

```
Player class
A class to handle the player

Attributes :
    inventory : Inventory
        the inventory of the player
    level : int
        the level of the player
    actions : list
        the actions of the player
    currentAction : Action
        the current action of the player
    commands : list
        the commands of the player
    currentCommand : str
        the current command of the player
```

```
    callbacks : list
        the callbacks of the player
    currentCallback : function
        the current callback of the player
    vision : list
        the vision of the player
    broadcastReceived : list
        the broadcast received by the player
    ejectionReceived : list
        the ejection received by the player
    isLeader : Role
        if the player is the leader/undefined/slave
    unusedSlots : int
        the unused slots
    currentlyElevating : bool
        if the player is currently elevating
    currentMode : Mode
        the current mode of the player
    currentFood : int
        the current food of the player
    nbSlaves : int
        the number of slaves that are alive
    waitingResponse : bool
        if the player is waiting for a response
    regroupDirection : int
        the direction of the regroup
    arrived : bool
        if the player arrived to the regroup
    isTimed : bool
        if the player is timed

----------

Methods :
    __init__()
        Constructor of the Player class
    __str__()
        Print the player
    moveForward(callback = None)
        Move the player forward
    turnRight(callback = None)
        Turn the player right
    turnLeft(callback = None)
        Turn the player left
    look(callback = None)
        Look around the player
    cmdInventory(callback = None)
        Get the inventory of the player
    broadcast(message : str = "Hello", callback = None)
        Broadcast a message
    connectNbr(callback = None)
        Connect to the number of players
    fork(callback = None)
        Fork the player
    eject(callback = None)
        Eject the player
    take(resource : str = "food", callback = None)
        Take a resource
    set(resource : str = "food", callback = None)
        Set a resource
    incantation(callback = None)
        Start the incantation
    none()
        Do nothing
    updateVision(vision : str)
        Update the vision of the player
    updateInventory(inventory : str)
        Update the inventory of the player
    updateBroadcastReceived(message : str)
        Update the broadcast received by the player
    updateEjectionReceived(message : str)
        Update the ejection received by the player
    updateLevel(level : int)
```

```
        Update the level of the player
    handleElevation(response : str)
        Handle the elevation
    hasSomethingHappened(response : str)
        Check if something happened
    handleResponse(response : str)
        Handle the response
    connectMissingPlayers()
        Connect the missing players
    completeTeam()
        Complete the team
    updateModeSlave()
        Update the mode of the player when he is a slave
    updateModeLeader()
        Update the mode of the player when he is a leader
    updateMode()
        Update the mode of the player
    lookingForFood()
        Look for food
    lookingForStones()
        Look for stones
    askSlavesForInventory()
        Ask the slaves for their inventory
    checkIfEnoughFood(response : str)
        Check if the slave has enough food
    handleResponseBroadcast()
        Handle the response of the broadcast
    slavesReponses()
        Handle the leader order as a slave
    waitingEveryone()
        Wait for everyone to finish the regroup
    waitingDrop()
        Wait for everyone to finish droping the stones
    dropping()
        Drop the stones
    regroupAction()
        Regroup the players
    chooseAction()
        Choose the action of the player
    goGetItem()
        Go get items at tile
    foodInVision()
        Check if there is food in the vision
    stonesInVision()
        Check if there are stones in the vision
```

3.11.2 Constructor & Destructor Documentation

3.11.2.1 __init__()

```
def Player.Player.__init__ (
              self )
```

Constructor of the Player class

3.11.3 Member Function Documentation

3.11.3.1 __str__()

```
def Player.Player.__str__ (
              self )
```

Print the player

3.11.3.2 askSlavesForInventory()

```
def Player.Player.askSlavesForInventory (
              self )
```

Ask the slaves for their inventory
The leader will ask the slaves for their inventory

3.11.3.3 broadcast()

```
def Player.Player.broadcast (
              self,
             str  message = "Hello",
              callback = None )
```

Set the current action to broadcast

```
Parameters :
    message : str
        the message to broadcast
    callback : function
        the callback to call after the action
        (default is None)
```

3.11.3.4 checkIfEnoughFood()

```
def Player.Player.checkIfEnoughFood (
              self,
             str response )
```

Check if the slave has enough food to survive the regroup

3.11.3.5 chooseAction()

```
def Player.Player.chooseAction (
              self )
```

Choose the action of the player
The action is chosen depending on the mode of the player
The mode is updated before choosing the action

3.11.3.6 cmdInventory()

```
def Player.Player.cmdInventory (
              self,
              callback = None )
```

Set the current action to inventory

Parameters :
    callback : function
        the callback to call after the action
        (default is None)

3.11.3.7 completeTeam()

```
def Player.Player.completeTeam (
              self )
```

Complete the team

3.11.3.8 connectMissingPlayers()

```
def Player.Player.connectMissingPlayers (
              self )
```

Connect the missing players

3.11.3.9 connectNbr()

```
def Player.Player.connectNbr (
              self,
              callback = None )

Set the current action to connect_nbr

Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

3.11.3.10 dropping()

```
def Player.Player.dropping (
              self )

Drop the stones
As a leader, you will wait for the slaves to drop the stones
As a slave, you will drop the stones until you have none left
```

3.11.3.11 eject()

```
def Player.Player.eject (
              self,
              callback = None )

Set the current action to eject

Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

3.11.3.12 foodInVision()

```
def Player.Player.foodInVision (
              self,
             list vision )

Allows us to know if there is food in view

Parameters:
    vision (list) :
        List of items in view

Returns:
    tuple : Contains bool if food found, and the tile's index
```

3.11.3.13 fork()

```
def Player.Player.fork (
              self,
              callback = None )

Set the current action to fork

Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

3.11.3.14 goGetItem()

```
def Player.Player.goGetItem (
              self,
              index,
             List[Item] itemSeek )

Allows us to go get items on the map at index X

Parameters :
    index : int
        index of the tile to go on
    itemSeek : List[Item]
        List of items to take on the tile
```

3.11.3.15 handleElevation()

```
def Player.Player.handleElevation (
              self,
             str response )

Handle the response of the elevation command

Parameters :
    response : str
        the response from the server
```

3.11.3.16 handleResponse()

```
def Player.Player.handleResponse (
              self,
             str response )

Handle the response from the server

Parameters :
    response : str
        the response from the server
```

3.11.3.17 handleResponseBroadcast()

```
def Player.Player.handleResponseBroadcast (
              self )
```

Handle the response of the broadcast

3.11.3.18 hasSomethingHappened()

```
def Player.Player.hasSomethingHappened (
              self,
             str response )
```

Check if something happened to the player
Look if the player is dead, if he received a message or if he was ejected

Parameters :
    response : str
        the response from the server

3.11.3.19 incantation()

```
def Player.Player.incantation (
              self,
              callback = None )
```

Set the current action to incantation

Parameters :
    callback : function
        the callback to call after the action
        (default is None)

3.11.3.20 look()

```
def Player.Player.look (
              self,
              callback = None )
```

Set the current action to look

Parameters :
    callback : function
        the callback to call after the action
        (default is None)

3.11.3.21 lookingForFood()

```
def Player.Player.lookingForFood (
              self )
```

```
Look for food
The player will look for the nearest food in his vision.
When he finds food, he will  go to the case
where there is food and take it.
```

3.11.3.22 lookingForStones()

```
def Player.Player.lookingForStones (
              self )
```

```
Look for stones
The player will look for the case with the most stones in his vision.
When he finds stones, he will  go to the case
where there are stones and take them.
```

3.11.3.23 moveForward()

```
def Player.Player.moveForward (
              self,
              callback = None )
```

```
Set the current action to forward

Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

3.11.3.24 none()

```
def Player.Player.none (
              self )
```

```
Set the current action to none
```

3.11.3.25 regroupAction()

```
def Player.Player.regroupAction (
              self )
```

```
Regroup the players
As a leader, you will wait for the slaves to regroup
As a slave, you will regroup with the leader
```

3.11.3.26 set()

```
def Player.Player.set (
              self,
             str  resource = "food",
              callback = None )
```

```
Set the current action to set

Parameters :
    resource : str
        the resource to set
    callback : function
        the callback to call after the action
        (default is None)
```

3.11.3.27 slavesReponses()

```
def Player.Player.slavesReponses (
              self )
```

```
Handle the leader order as a slave
```

3.11.3.28 stonesInVision()

```
def Player.Player.stonesInVision (
              self,
             list vision )
```

```
Allows us to know if there are stones in view

Parameters :
    vision : list:
        List of items in view

Returns :
    truple: bool for found stones, tile's index, list of stones enum
```

3.11.3.29 take()

```
def Player.Player.take (
              self,
             str  resource = "food",
              callback = None )
```

Set the current action to take

```
Parameters :
    resource : str
        the resource to take
    callback : function
        the callback to call after the action
        (default is None)
```

3.11.3.30 turnLeft()

```
def Player.Player.turnLeft (
              self,
              callback = None )
```

Set the current action tl moderation bot designed for mo left

```
Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

3.11.3.31 turnRight()

```
def Player.Player.turnRight (
              self,
              callback = None )
```

Set the current action to right

```
Parameters :
    callback : function
        the callback to call after the action
        (default is None)
```

3.11.3.32 updateBroadcastReceived()

```
def Player.Player.updateBroadcastReceived (
             self,
            str message )
```

Update the broadcast received by the player

```
Parameters :
    message : str
        the message from the server
```

3.11.3.33 updateEjectionReceived()

```
def Player.Player.updateEjectionReceived (
             self,
            str message )
```

Update the ejection received by the player

```
Parameters :
    message : str
        the message from the server
```

3.11.3.34 updateInventory()

```
def Player.Player.updateInventory (
             self,
            str inventory )
```

Update the inventory of the player with the data from the inventory command

```
Parameters :
    inventory : str
        the inventory from the server
```

3.11.3.35 updateLevel()

```
def Player.Player.updateLevel (
             self,
            int level )
```

Update the level of the player

```
Parameters :
    level : int
        the level of the player
```

3.11.3.36 updateMode()

```
def Player.Player.updateMode (
              self )
```

Update the mode of the player

3.11.3.37 updateModeLeader()

```
def Player.Player.updateModeLeader (
              self )
```

Update the mode of the player when he is a leader

3.11.3.38 updateModeSlave()

```
def Player.Player.updateModeSlave (
              self )
```

Update the mode of the player when he is a slave

3.11.3.39 updateVision()

```
def Player.Player.updateVision (
              self,
             str vision )
```

Update the vision of the player with the data from the look command

```
Parameters :
    vision : str
        the vision from the server
```

3.11.3.40 waitingDrop()

```
def Player.Player.waitingDrop (
              self )
```

Wait for everyone to finish droping the stones

3.11.3.41 waitingEveryone()

```
def Player.Player.waitingEveryone (
              self )

Wait for everyone to finish the regroup
```

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Player/Player.py

3.12 PlayerException.PlayerDeathException Class Reference

Inheritance diagram for PlayerException.PlayerDeathException:

Collaboration diagram for PlayerException.PlayerDeathException:

Public Member Functions

- def __init__ (self, message)
- def __init__ (self, message)

3.12.1 Detailed Description

```
PlayerDeathException class

A class to handle the death of the player
The PlayerDeathException class inherits from the PlayerException class

Attributes :
    message : str
        the message of the exception
```

3.12.2 Constructor & Destructor Documentation

3.12.2.1 __init__()

```
def PlayerException.PlayerDeathException.__init__ (
              self,
              message )

Constructor of the PlayerDeathException class
```

Reimplemented from PlayerException.PlayerException.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Player/PlayerException.py

3.13 PlayerException.PlayerException Class Reference

Inheritance diagram for PlayerException.PlayerException:

Collaboration diagram for PlayerException.PlayerException:

Public Member Functions

- def __init__ (self, message)

3.13.1 Detailed Description

```
PlayerException class

A class to handle exceptions that can occur in the Player
The PlayerException class inherits from the IError class

Attributes :
    message : str
        the message of the exception
```

3.13.2 Constructor & Destructor Documentation

3.13.2.1 __init__()

```
def PlayerException.PlayerException.__init__ (
              self,
              message )

Constructor of the PlayerException class
```

Reimplemented in PlayerException.PlayerDeathException.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Player/PlayerException.py

3.14 Role.Role Class Reference

Inheritance diagram for Role.Role:

Collaboration diagram for Role.Role:

Static Public Attributes

- int **UNDEFINED** = 0
- int **LEADER** = 1
- int **SLAVE** = 2

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/ai/src/Enum/Role.py

Index