Nagy Házi feladat

Kincses Benedek

DN5HSD

# Table of Contents

Table of contents

# Hierarchical Index

## Class Hierarchy

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

ActionListener

uilogic.GridButtonHandler

uilogic.MultipleButtonHandler

uilogic.InteractButtonHandler

uilogic.PlayFrameMenuBarHandler

uilogic.PlayerDeathFrameHandler

uilogic.UtilityButtonHandler

uilogic.CharacterFrameHandler

uilogic.CombatLogger

DataCreator

game.utility.Dice

game.global.DiceRoller

Dimension

ui.data.GridDimension

EnemyBehaviourController

game.behaviour.entities.enemy.controller.BerserkerEnemyController

game.behaviour.abstracts.EnemyBehaviourController

game.enums.EnemyBehaviourControllerType

Entity

game.behaviour.entities.InventoryEntity

game.behaviour.entities.enemy.EnemyEntity

game.behaviour.entities.player.PlayerEntity

game.enums.EntityCondition

game.enums.EntityType

Equipment

game.behaviour.entities.items.equipment.Armor

game.enums.EquipmentType

game.utility.event.Event

game.utility.event.EventArgument< T >

Exception

exception.dice.DefaultDiceNotSetException

exception.dice.InvalidDiceSideCountException

exception.entity.AlreadyAttackedThisTurnException

exception.entity.ItemNotInInventoryException

exception.entity.NoWeaponEquippedException

exception.general.ArgumentNullException

exception.general.ConfigNotLoadedException

exception.general.ElementAlreadyInCollectionException

exception.general.ElementNotFoundException

exception.general.InvalidArgumentException

exception.item.InvalidIDException

exception.save.CurrentSaveUnmodifiableException

exception.ui.ComponentAlreadyAtPositionException

exception.ui.PlayfieldNotEmptyException

exception.ui.UIHandlerAlreadyStartedException

uilogic.FileChooserType

game.global.FileHandler

file.FileIOUtil

game.GameActionController

game.global.GameHandler

game.utility.GenericDelegate

uilogic.GridEntityComponentHandler

ui.elements.GridPanel

Identifiable

file.elements.IconData

game.behaviour.entities.enemy.Enemy

Identity

file.elements.MapLayoutData

game.behaviour.entities.enemy.EnemyType

game.behaviour.entities.items.Item

game.behaviour.entities.items.Consumable

game.behaviour.entities.player.Player

game.utility.IDisplayable

IDisplayable

game.behaviour.entities.items.Item

game.utility.event.IEventListener

IEventListener

game.behaviour.Inventory

IGridPositionable

ui.elements.DummyComponent

ui.elements.GridButton

ui.elements.GridEntityComponent

uilogic.IGridPositionable

game.behaviour.entities.IInteractiveEntity

game.behaviour.entities.enemy.Enemy

game.behaviour.entities.player.Player

uilogic.InteractiveGridHandler

uilogic.PlayFieldHandler

game.behaviour.InventoryMarker< T >

Item

game.behaviour.abstracts.Equipment

game.behaviour.abstracts.Weapon

game.enums.ItemType

JButton

ui.elements.CustomButton

ui.elements.GridButton

JFrame

ui.elements.CharacterFrame

ui.elements.PlayFrame

ui.elements.PlayerDeathFrame

ui.elements.TravelFrame

JLabel

ui.elements.ImageComponent

ui.elements.GridEntityComponent

JPanel

ui.elements.DummyComponent

ui.elements.EquipmentItemPanel

ui.elements.EquipmentPanel

ui.elements.InteractButtonPanel

ui.elements.InteractiveGridPanel

ui.elements.InventoryPanel

ui.elements.LabelPanel

ui.elements.ScrollableTextPanel

ui.elements.UtilityButtonPanel

Main

game.enums.ModifierType

game.global.SaveHandler

Serializable

file.elements.EnemyMapData

file.elements.EnemyTypeSave

file.elements.GameConfigSave

file.elements.ItemMapData

file.elements.ItemSave

file.elements.PlayerProgressSave

game.behaviour.abstracts.Entity

game.utility.Identifiable

game.utility.Identity

game.utility.ModifiedEnemyData

uilogic.GridPosition

game.global.storage.Storage< T >

game.global.storage.Storage< Enemy >

game.global.storage.ActiveEnemyStorage

game.global.storage.Storage< EnemyType >

game.global.storage.EnemyTypeStorage

game.global.storage.Storage< IconData >

game.global.storage.IconDataStorage

game.global.storage.Storage< Item >

game.global.storage.ItemStorage

game.global.storage.Storage< ModifiedEnemyData >

game.global.storage.ModifiedEnemyStorage

game.global.storage.Storage< String >

game.global.storage.MapStorage

uilogic.TravelFrameHandler

game.global.UIHandler

Weapon

game.behaviour.entities.items.equipment.weapons.AutoPistol

game.behaviour.entities.items.equipment.weapons.Shotgun

game.enums.WeaponType

# Class Index

## Class List

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

[**game.global.storage.ActiveEnemyStorage**](#AAAAAAAANX)

[**exception.entity.AlreadyAttackedThisTurnException**](#AAAAAAAAAH)

[**exception.general.ArgumentNullException**](#AAAAAAAAAK)

[**game.behaviour.entities.items.equipment.Armor**](#AAAAAAAAJQ)

[**game.behaviour.entities.items.equipment.weapons.AutoPistol**](#AAAAAAAAKA)

[**game.behaviour.entities.enemy.controller.BerserkerEnemyController**](#AAAAAAAAIW)

[**ui.elements.CharacterFrame**](#AAAAAAAAQR)

[**uilogic.CharacterFrameHandler**](#AAAAAAAAUX)

[**uilogic.CombatLogger**](#AAAAAAAAVF)

[**exception.ui.ComponentAlreadyAtPositionException**](#AAAAAAAAAR)

[**exception.general.ConfigNotLoadedException**](#AAAAAAAAAL)

[**game.behaviour.entities.items.Consumable**](#AAAAAAAAIZ)

[**exception.save.CurrentSaveUnmodifiableException**](#AAAAAAAAAQ)

[**ui.elements.CustomButton**](#AAAAAAAARA)

[**DataCreator**](#AAAAAAAAAA)

[**exception.dice.DefaultDiceNotSetException**](#AAAAAAAAAF)

[**game.utility.Dice**](#AAAAAAAAOT)

[**game.global.DiceRoller**](#AAAAAAAAME)

[**ui.elements.DummyComponent**](#AAAAAAAARE)

[**exception.general.ElementAlreadyInCollectionException**](#AAAAAAAAAM)

[**exception.general.ElementNotFoundException**](#AAAAAAAAAN)

[**game.behaviour.entities.enemy.Enemy**](#AAAAAAAAII)

[**game.behaviour.abstracts.EnemyBehaviourController**](#AAAAAAAADY)

[**game.enums.EnemyBehaviourControllerType**](#AAAAAAAALA)

[**game.behaviour.entities.enemy.EnemyEntity**](#AAAAAAAAIP)

[**file.elements.EnemyMapData**](#AAAAAAAAAZ)

[**game.behaviour.entities.enemy.EnemyType**](#AAAAAAAAIK)

[**file.elements.EnemyTypeSave**](#AAAAAAAABF)

[**game.global.storage.EnemyTypeStorage**](#AAAAAAAAOC)

[**game.behaviour.abstracts.Entity**](#AAAAAAAAEK)

[**game.enums.EntityCondition**](#AAAAAAAALD)

[**game.enums.EntityType**](#AAAAAAAALH)

[**game.behaviour.abstracts.Equipment**](#AAAAAAAAFK)

[**ui.elements.EquipmentItemPanel**](#AAAAAAAAQY)

[**ui.elements.EquipmentPanel**](#AAAAAAAAQU)

[**game.enums.EquipmentType**](#AAAAAAAALK)

[**game.utility.event.Event**](#AAAAAAAAPZ)

[**game.utility.event.EventArgument< T >**](#AAAAAAAAQB)

[**uilogic.FileChooserType**](#AAAAAAAAVL)

[**game.global.FileHandler**](#AAAAAAAAMK)

[**file.FileIOUtil**](#AAAAAAAAAU)

[**game.GameActionController**](#AAAAAAAACZ)

[**file.elements.GameConfigSave**](#AAAAAAAABN)

[**game.global.GameHandler**](#AAAAAAAAMQ)

[**game.utility.GenericDelegate**](#AAAAAAAAOY)

[**ui.elements.GridButton**](#AAAAAAAART)

[**uilogic.GridButtonHandler**](#AAAAAAAAUZ)

[**ui.data.GridDimension**](#AAAAAAAAQN)

[**ui.elements.GridEntityComponent**](#AAAAAAAARZ)

[**uilogic.GridEntityComponentHandler**](#AAAAAAAAVS)

[**ui.elements.GridPanel**](#AAAAAAAASK)

[**uilogic.GridPosition**](#AAAAAAAAVZ)

[**file.elements.IconData**](#AAAAAAAABU)

[**game.global.storage.IconDataStorage**](#AAAAAAAAOE)

[**game.utility.Identifiable**](#AAAAAAAAPA)

[**game.utility.Identity**](#AAAAAAAAPE)

[**game.utility.IDisplayable**](#AAAAAAAAPL)

[**game.utility.event.IEventListener**](#AAAAAAAAQH)

[**uilogic.IGridPositionable**](#AAAAAAAAWL)

[**game.behaviour.entities.IInteractiveEntity**](#AAAAAAAAGR)

[**ui.elements.ImageComponent**](#AAAAAAAASF)

[**uilogic.InteractButtonHandler**](#AAAAAAAAWO)

[**ui.elements.InteractButtonPanel**](#AAAAAAAAST)

[**uilogic.InteractiveGridHandler**](#AAAAAAAAWU)

[**ui.elements.InteractiveGridPanel**](#AAAAAAAASW)

[**exception.general.InvalidArgumentException**](#AAAAAAAAAO)

[**exception.dice.InvalidDiceSideCountException**](#AAAAAAAAAG)

[**exception.item.InvalidIDException**](#AAAAAAAAAP)

[**game.behaviour.Inventory**](#AAAAAAAADG)

[**game.behaviour.entities.InventoryEntity**](#AAAAAAAAIC)

[**game.behaviour.InventoryMarker< T >**](#AAAAAAAADT)

[**ui.elements.InventoryPanel**](#AAAAAAAAQW)

[**game.behaviour.entities.items.Item**](#AAAAAAAAJK)

[**file.elements.ItemMapData**](#AAAAAAAABY)

[**exception.entity.ItemNotInInventoryException**](#AAAAAAAAAI)

[**file.elements.ItemSave**](#AAAAAAAACB)

[**game.global.storage.ItemStorage**](#AAAAAAAAOG)

[**game.enums.ItemType**](#AAAAAAAALN)

[**ui.elements.LabelPanel**](#AAAAAAAATR)

[**Main**](#AAAAAAAAAD)

[**file.elements.MapLayoutData**](#AAAAAAAACE)

[**game.global.storage.MapStorage**](#AAAAAAAAOI)

[**game.utility.ModifiedEnemyData**](#AAAAAAAAPP)

[**game.global.storage.ModifiedEnemyStorage**](#AAAAAAAAOK)

[**game.enums.ModifierType**](#AAAAAAAALR)

[**uilogic.MultipleButtonHandler**](#AAAAAAAAWQ)

[**exception.entity.NoWeaponEquippedException**](#AAAAAAAAAJ)

[**game.behaviour.entities.player.Player**](#AAAAAAAAKH)

[**ui.elements.PlayerDeathFrame**](#AAAAAAAATV)

[**uilogic.PlayerDeathFrameHandler**](#AAAAAAAAXO)

[**game.behaviour.entities.player.PlayerEntity**](#AAAAAAAAKJ)

[**file.elements.PlayerProgressSave**](#AAAAAAAACP)

[**uilogic.PlayFieldHandler**](#AAAAAAAAXP)

[**exception.ui.PlayfieldNotEmptyException**](#AAAAAAAAAS)

[**ui.elements.PlayFrame**](#AAAAAAAATX)

[**uilogic.PlayFrameMenuBarHandler**](#AAAAAAAAXV)

[**game.global.SaveHandler**](#AAAAAAAAMS)

[**ui.elements.ScrollableTextPanel**](#AAAAAAAAUN)

[**game.behaviour.entities.items.equipment.weapons.Shotgun**](#AAAAAAAAKC)

[**game.global.storage.Storage< T >**](#AAAAAAAANZ)

[**ui.elements.TravelFrame**](#AAAAAAAAUR)

[**uilogic.TravelFrameHandler**](#AAAAAAAAXW)

[**game.global.UIHandler**](#AAAAAAAANK)

[**exception.ui.UIHandlerAlreadyStartedException**](#AAAAAAAAAT)

[**uilogic.UtilityButtonHandler**](#AAAAAAAAXZ)

[**ui.elements.UtilityButtonPanel**](#AAAAAAAAUU)

[**game.behaviour.abstracts.Weapon**](#AAAAAAAAEO)

[**game.enums.WeaponType**](#AAAAAAAALW)

# Class Documentation

## game.global.storage.ActiveEnemyStorage Class Reference

### Static Public Member Functions

static [ActiveEnemyStorage](#AAAAAAAANX) **getInstance** ()

### Additional Inherited Members

#### Public Member Functions inherited from [game.global.storage.Storage< Enemy >](#AAAAAAAANZ)

void **add** (String id, T item) throws ArgumentNullException

void **remove** (String id) throws ArgumentNullException

T **get** (String id) throws ArgumentNullException

boolean **contains** (String id) throws ArgumentNullException

void **clear** ()

Set< Entry< String, T > > **entrySet** ()

ArrayList< T > [getAllItems](#AAAAAAAAOA) ()

ArrayList< String > [getAllKeys](#AAAAAAAAOB) ()

#### Protected Attributes inherited from [game.global.storage.Storage< Enemy >](#AAAAAAAANZ)

HashMap< String, T > **storage**

### Detailed Description

This class represents an ActiveEnemyStorage in the game. It stores all the active enemies on the current map. The key is the ID of the enemy. It extends the Storage class with Enemy type.

The class contains the following fields:

instance: The singleton instance of the ActiveEnemyStorage class.

The class provides a private constructor and a method to get the singleton instance.

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

game/global/storage/ActiveEnemyStorage.java

## exception.entity.AlreadyAttackedThisTurnException Class Reference

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

exception/entity/AlreadyAttackedThisTurnException.java

## exception.general.ArgumentNullException Class Reference

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

exception/general/ArgumentNullException.java

## game.behaviour.entities.items.equipment.Armor Class Reference

### Public Member Functions

**Armor** (int armorClass, int movementBonus)

int **getArmorClass** ()

int **getMovementBonus** ()

void **setArmorClass** (int armorClass)

void **setMovementBonus** (int movementBonus)

String **getDisplayInfo** ()

String **getStatistics** (int bearerLevel)

### Protected Attributes

int **armorClass**

int **movementBonus**

### Detailed Description

This class represents an Armor in the game. It extends the Equipment class and includes additional properties such as armor class and movement bonus.

The class contains the following fields:

armorClass: The armor class of the armor.

movementBonus: The movement bonus provided by the armor.

The class provides getter and setter methods for these fields. It also overrides the getDisplayInfo and getStatistics methods from the Equipment class.

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

game/behaviour/entities/items/equipment/Armor.java

## game.behaviour.entities.items.equipment.weapons.AutoPistol Class Reference

### Public Member Functions

[AutoPistol](#AAAAAAAAKB) (String id, String weaponName) throws ArgumentNullException

### Detailed Description

This class represents an AutoPistol in the game. It extends the Weapon class.

The class contains a constructor that initializes the id and name of the weapon and sets the weapon type to AUTOPISTOL and the number of attacks in a round to 2.

### Constructor & Destructor Documentation

#### game.behaviour.entities.items.equipment.weapons.AutoPistol.AutoPistol (String *id*, String *weaponName*) throws ArgumentNullException

Constructor for the AutoPistol class. Initializes the id and name of the weapon and sets the weapon type to AUTOPISTOL and the number of attacks in a round to 2.

##### Parameters

|  |  |
| --- | --- |
| *id* | The id of the weapon. |
| *weaponName* | The name of the weapon. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the id or weaponName is null. |

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

game/behaviour/entities/items/equipment/weapons/AutoPistol.java

## game.behaviour.entities.enemy.controller.BerserkerEnemyController Class Reference

### Public Member Functions

**BerserkerEnemyController** ([EnemyEntity](#AAAAAAAAIP) enemy) throws ArgumentNullException

void [runEnemy](#AAAAAAAAIY) ([IInteractiveEntity](#AAAAAAAAGR) target, double distance) throws Exception

### Detailed Description

This class represents a BerserkerEnemyController in the game. It extends the EnemyBehaviourController class and overrides the runEnemy method.

The class contains the following fields:

controllerType: The type of the controller, set to BERSERK in the constructor.

The class provides a constructor that initializes the enemyEntity and sets up the events.

### Member Function Documentation

#### void game.behaviour.entities.enemy.controller.BerserkerEnemyController.runEnemy ([IInteractiveEntity](#AAAAAAAAGR) *target*, double *distance*) throws Exception

Executes the berserker enemy's behaviour towards a target. The berserker enemy attempts to attack the target and if successful, deals damage to the target.

##### Parameters

|  |  |
| --- | --- |
| *target* | The target of the enemy's behaviour. |
| *distance* | The distance to the target. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the execution of the behaviour. |

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

game/behaviour/entities/enemy/controller/BerserkerEnemyController.java

## ui.elements.CharacterFrame Class Reference

### Public Member Functions

[CharacterFrame](#AAAAAAAAQS) (int inventoryCount, ActionListener buttonHandler) throws ArgumentNullException

void **refresh** ()

[EquipmentPanel](#AAAAAAAAQU) **getEquipmentPanel** ()

[InventoryPanel](#AAAAAAAAQW) **getInventoryPanel** ()

[EquipmentItemPanel](#AAAAAAAAQY) **getItemPanel** ()

### Detailed Description

This class represents a character frame in the game. It displays the character's equipment and inventory. It extends the JFrame class and contains an equipment panel, an inventory panel, and an equipment item panel.

The class contains the following fields:

equipmentPanel: The equipment panel of the character.

inventoryPanel: The inventory panel of the character.

itemPanel: The equipment item panel of the character.

### Constructor & Destructor Documentation

#### ui.elements.CharacterFrame.CharacterFrame (int *inventoryCount*, ActionListener *buttonHandler*) throws ArgumentNullException

Constructor for the CharacterFrame class. Initializes the frame with the specified inventory count and button handler.

##### Parameters

|  |  |
| --- | --- |
| *inventoryCount* | The count of the inventory. |
| *buttonHandler* | The handler of the button. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the button handler is null. |

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

ui/elements/CharacterFrame.java

## uilogic.CharacterFrameHandler Class Reference

### Public Member Functions

[CharacterFrameHandler](#AAAAAAAAUY) ()

[GridButtonHandler](#AAAAAAAAUZ) [getGridButtonHandler](#AAAAAAAAVA) ()

void [setCharacterFrame](#AAAAAAAAVB) (CharacterFrame frame) throws ArgumentNullException

void [selectItem](#AAAAAAAAVC) (Object o)

void [equipItem](#AAAAAAAAVD) ()

void [start](#AAAAAAAAVE) () throws ConfigNotLoadedException

### Detailed Description

This class handles the character frame in the UI. It contains a CharacterFrame, a GridButtonHandler for handling grid button actions, and an Item for the selected item.

The class contains the following fields:

frame: The CharacterFrame that this class handles.

gridButtonHandler: The GridButtonHandler for handling grid button actions.

item: The Item for the selected item.

### Constructor & Destructor Documentation

#### uilogic.CharacterFrameHandler.CharacterFrameHandler ()

Constructor for the CharacterFrameHandler class. Initializes the GridButtonHandler with a GenericDelegate that runs the selectItem method when a grid button is clicked.

### Member Function Documentation

#### void uilogic.CharacterFrameHandler.equipItem ()

Equips an item. If the selected item is not null and is an equipment, equips the selected item to the player, and shows the equipped item in the equipment panel.

#### [GridButtonHandler](#AAAAAAAAUZ) uilogic.CharacterFrameHandler.getGridButtonHandler ()

Gets the GridButtonHandler.

##### Returns

The GridButtonHandler.

#### void uilogic.CharacterFrameHandler.selectItem (Object *o*)

Selects an item. Sets the selected item to the item associated with the clicked grid button, and shows the selected item in the item panel.

##### Parameters

|  |  |
| --- | --- |
| *o* | The object associated with the clicked grid button. |

#### void uilogic.CharacterFrameHandler.setCharacterFrame (CharacterFrame *frame*) throws ArgumentNullException

Sets the CharacterFrame.

##### Parameters

|  |  |
| --- | --- |
| *frame* | The CharacterFrame to set. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the frame is null. |

#### void uilogic.CharacterFrameHandler.start () throws ConfigNotLoadedException

Starts the character frame handler. Gets the player, checks if the player is null, gets the inventory count, initializes the character frame with the inventory count and the grid button handler, fills the inventory, sets the equipment panel, sets the item display panel, and sets the visibility of the character frame to true.

##### Exceptions

|  |  |
| --- | --- |
| *ConfigNotLoadedException* | if the player is null. |

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

uilogic/CharacterFrameHandler.java

## uilogic.CombatLogger Class Reference

### Public Member Functions

[CombatLogger](#AAAAAAAAVG) (PlayFrame frame)

void [addSystemLog](#AAAAAAAAVH) (String message) throws ArgumentNullException

void [addEntityLog](#AAAAAAAAVI) (String entityName, String message) throws ArgumentNullException

void [addPlainText](#AAAAAAAAVJ) (String message) throws ArgumentNullException

void [addMapDescription](#AAAAAAAAVK) (String mapName, String description) throws ArgumentNullException

### Detailed Description

This class handles the logging of combat in the UI. It contains a PlayFrame for displaying the combat log.

The class contains the following field:

frame: The PlayFrame that this class uses to display the combat log.

### Constructor & Destructor Documentation

#### uilogic.CombatLogger.CombatLogger (PlayFrame *frame*)

Constructor for the CombatLogger class. Initializes the CombatLogger with a PlayFrame.

##### Parameters

|  |  |
| --- | --- |
| *frame* | The PlayFrame that this class uses to display the combat log. |

### Member Function Documentation

#### void uilogic.CombatLogger.addEntityLog (String *entityName*, String *message*) throws ArgumentNullException

Adds an entity log to the combat log. Prepends the entity name in brackets and a colon to the message and adds it to the combat log.

##### Parameters

|  |  |
| --- | --- |
| *entityName* | The name of the entity. |
| *message* | The message to add to the combat log. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the entity name or the message is null. |

#### void uilogic.CombatLogger.addMapDescription (String *mapName*, String *description*) throws ArgumentNullException

Adds a map description to the combat log.

##### Parameters

|  |  |
| --- | --- |
| *mapName* | The name of the map. |
| *description* | The description of the map. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the map name or the description is null. |

#### void uilogic.CombatLogger.addPlainText (String *message*) throws ArgumentNullException

Adds plain text to the combat log.

##### Parameters

|  |  |
| --- | --- |
| *message* | The text to add to the combat log. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the message is null. |

#### void uilogic.CombatLogger.addSystemLog (String *message*) throws ArgumentNullException

Adds a system log to the combat log. Prepends "[SYSTEM]: " to the message and adds it to the combat log.

##### Parameters

|  |  |
| --- | --- |
| *message* | The message to add to the combat log. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the message is null. |

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

uilogic/CombatLogger.java

## exception.ui.ComponentAlreadyAtPositionException Class Reference

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

exception/ui/ComponentAlreadyAtPositionException.java

## exception.general.ConfigNotLoadedException Class Reference

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

exception/general/ConfigNotLoadedException.java

## game.behaviour.entities.items.Consumable Class Reference

### Public Member Functions

[Consumable](#AAAAAAAAJA) (String id, ModifierType type, int amount) throws ArgumentNullException

int **getCharges** ()

ModifierType **getType** ()

boolean **isOn** ()

[Consumable](#AAAAAAAAIZ) **setCharges** (int amount) throws InvalidArgumentException

void **addCharges** (int amount) throws InvalidArgumentException

void [toggle](#AAAAAAAAJG) (boolean on)

double [use](#AAAAAAAAJH) () throws Exception

int [decreaseCharges](#AAAAAAAAJI) () throws Exception

void [addEventListener](#AAAAAAAAJJ) (IEventListener listener) throws ArgumentNullException

#### Public Member Functions inherited from [game.behaviour.entities.items.Item](#AAAAAAAAJK)

[Item](#AAAAAAAAJL) ()

ItemType **getItemType** ()

String **getDisplayInfo** ()

String **getStatistics** (int bearerLevel)

### Additional Inherited Members

#### Protected Attributes inherited from [game.behaviour.entities.items.Item](#AAAAAAAAJK)

ItemType **itemType**

### Detailed Description

This class represents a Consumable item in the game. It extends the Item class.

The class contains the following fields:

modifier: The modifier of the consumable.

charges: The number of charges of the consumable.

toggled: A flag indicating whether the consumable is toggled on.

type: The type of the consumable.

onOutOfChargesEvent: An event that is triggered when the consumable is out of charges.

The class provides a constructor that initializes these fields and methods to get the charges, type, and toggled status, and to set the charges.

### Constructor & Destructor Documentation

#### game.behaviour.entities.items.Consumable.Consumable (String *id*, ModifierType *type*, int *amount*) throws ArgumentNullException

Constructor for the Consumable class. Initializes the id, type, and charges of the consumable and sets the itemType to CONSUMABLE.

##### Parameters

|  |  |
| --- | --- |
| *id* | The id of the consumable. |
| *type* | The type of the consumable. |
| *amount* | The number of charges of the consumable. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the id is null. |

### Member Function Documentation

#### void game.behaviour.entities.items.Consumable.addEventListener (IEventListener *listener*) throws ArgumentNullException

Adds an event listener to the onOutOfChargesEvent.

##### Parameters

|  |  |
| --- | --- |
| *listener* | The event listener to add. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the listener is null. |

#### int game.behaviour.entities.items.Consumable.decreaseCharges () throws Exception

Decreases the charges of the consumable by 1. If the charges are less than 0, it sets the charges to 0. If the charges are 0, it triggers the onOutOfChargesEvent.

##### Returns

The number of charges of the consumable.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the decrease of the charges. |

#### void game.behaviour.entities.items.Consumable.toggle (boolean *on*)

Toggles the consumable on or off.

##### Parameters

|  |  |
| --- | --- |
| *on* | True to toggle the consumable on, false to toggle it off. |

#### double game.behaviour.entities.items.Consumable.use () throws Exception

Uses the consumable. If the consumable is toggled on, it decreases the charges and returns the modifier.

##### Returns

The modifier of the consumable if it is toggled on, 0 otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the use of the consumable. |

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

game/behaviour/entities/items/Consumable.java

## exception.save.CurrentSaveUnmodifiableException Class Reference

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

exception/save/CurrentSaveUnmodifiableException.java

## ui.elements.CustomButton Class Reference

### Public Member Functions

[CustomButton](#AAAAAAAARB) (int width, int height)

[CustomButton](#AAAAAAAARC) (int width, int height, String text) throws ArgumentNullException

[CustomButton](#AAAAAAAARA) [setButtonText](#AAAAAAAARD) (String text) throws ArgumentNullException

### Detailed Description

This class represents a custom button in the game. It extends the JButton class and contains methods to initialize the button and set its text.

The class contains the following fields:

width: The width of the button.

height: The height of the button.

text: The text of the button.

### Constructor & Destructor Documentation

#### ui.elements.CustomButton.CustomButton (int *width*, int *height*)

Constructor for the CustomButton class. Initializes the button with the specified width and height.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the button. |
| *height* | The height of the button. |

#### ui.elements.CustomButton.CustomButton (int *width*, int *height*, String *text*) throws ArgumentNullException

Constructor for the CustomButton class. Initializes the button with the specified width, height, and text.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the button. |
| *height* | The height of the button. |
| *text* | The text of the button. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the text is null. |

### Member Function Documentation

#### [CustomButton](#AAAAAAAARA) ui.elements.CustomButton.setButtonText (String *text*) throws ArgumentNullException

Sets the text of the button.

##### Parameters

|  |  |
| --- | --- |
| *text* | The new text to set. |

##### Returns

The button itself, for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the new text is null. |

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

ui/elements/CustomButton.java

## DataCreator Class Reference

### Static Public Member Functions

static void **createTestData** () throws Exception

### Static Public Attributes

static String **datafolder** = "G:\\uni\\sub\\3\\prog\\hf\\adventure-game-java\\project\\resources\\gamedata\\"

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

DataCreator.java

## exception.dice.DefaultDiceNotSetException Class Reference

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

exception/dice/DefaultDiceNotSetException.java

## game.utility.Dice Class Reference

### Public Member Functions

**Dice** (int sides) throws InvalidDiceSideCountException

void [setSides](#AAAAAAAAOV) (int sides) throws InvalidDiceSideCountException

int **getSideCount** ()

int [roll](#AAAAAAAAOX) ()

### Detailed Description

This class represents a dice with a variable number of sides. It contains methods to set the number of sides, get the number of sides, and roll the dice.

The class contains the following fields:

sideCount: The number of sides on the dice.

### Member Function Documentation

#### int game.utility.Dice.roll ()

Rolls the dice.

##### Returns

A random number between 1 and the number of sides on the dice.

#### void game.utility.Dice.setSides (int *sides*) throws InvalidDiceSideCountException

Sets the number of sides on the dice.

##### Parameters

|  |  |
| --- | --- |
| *sides* | The number of sides to set. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidDiceSideCountException* | if the number of sides is less than 1. |

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

game/utility/Dice.java

## game.global.DiceRoller Class Reference

### Public Member Functions

int **getDefault** ()

void [setDefault](#AAAAAAAAMG) (int sides) throws InvalidDiceSideCountException

int [rollDefault](#AAAAAAAAMH) (int bonus) throws DefaultDiceNotSetException

int [rollDice](#AAAAAAAAMI) (int sides, int rolls, int rollBonus) throws InvalidDiceSideCountException

### Static Public Member Functions

static [DiceRoller](#AAAAAAAAME) **getInstance** ()

### Detailed Description

This class represents a DiceRoller in the game. It is a singleton class that handles the rolling of dice.

The class contains the following fields:

diceRoller: The singleton instance of the DiceRoller class.

defaultDice: The default dice that is rolled on "rollDefault".

delegateOnRoll: The delegate that is called when a dice is rolled.

The class provides a private constructor and a method to get the singleton instance.

### Member Function Documentation

#### int game.global.DiceRoller.rollDefault (int *bonus*) throws DefaultDiceNotSetException

Rolls the default dice with a bonus.

##### Parameters

|  |  |
| --- | --- |
| *bonus* | The bonus to add to the dice roll. |

##### Returns

The result of the dice roll plus the bonus.

##### Exceptions

|  |  |
| --- | --- |
| *DefaultDiceNotSetException* | if the default dice is not set. |

#### int game.global.DiceRoller.rollDice (int *sides*, int *rolls*, int *rollBonus*) throws InvalidDiceSideCountException

Rolls a dice with a specified number of sides, a specified number of times, with a bonus.

##### Parameters

|  |  |
| --- | --- |
| *sides* | The number of sides on the dice. |
| *rolls* | The number of times to roll the dice. |
| *rollBonus* | The bonus to add to each dice roll. |

##### Returns

The total result of all the dice rolls plus the bonuses.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidDiceSideCountException* | if the number of sides is less than 1. |

#### void game.global.DiceRoller.setDefault (int *sides*) throws InvalidDiceSideCountException

Sets the default dice.

##### Parameters

|  |  |
| --- | --- |
| *sides* | The number of sides on the default dice. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidDiceSideCountException* | if the number of sides is less than 1. |

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

game/global/DiceRoller.java

## ui.elements.DummyComponent Class Reference

### Public Member Functions

[DummyComponent](#AAAAAAAARF) (int x, int y, GridPosition position) throws ArgumentNullException

GridPosition **getGridPosition** ()

GridPosition **setGridPosition** (GridPosition newPosition) throws ArgumentNullException

### Detailed Description

This class represents a dummy component in the game. It is used to fill empty grid positions. It extends the JPanel class and implements the IGridPositionable interface.

The class contains the following field:

gridPosition: The grid position of the dummy component.

### Constructor & Destructor Documentation

#### ui.elements.DummyComponent.DummyComponent (int *x*, int *y*, GridPosition *position*) throws ArgumentNullException

Constructor for the DummyComponent class. Initializes the dummy component with the specified x, y, and grid position.

##### Parameters

|  |  |
| --- | --- |
| *x* | The x of the dummy component. |
| *y* | The y of the dummy component. |
| *position* | The grid position of the dummy component. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the grid position is null. |

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

ui/elements/DummyComponent.java

## exception.general.ElementAlreadyInCollectionException Class Reference

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

exception/general/ElementAlreadyInCollectionException.java

## exception.general.ElementNotFoundException Class Reference

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

exception/general/ElementNotFoundException.java

## game.behaviour.entities.enemy.Enemy Class Reference

### Public Member Functions

[Enemy](#AAAAAAAAIJ) (String id, [EnemyType](#AAAAAAAAIK) enemyType) throws ArgumentNullException

[EnemyType](#AAAAAAAAIK) **getEnemyType** ()

int [getCurrentHealth](#AAAAAAAAIM) ()

double [getCurrentMovement](#AAAAAAAAIN) ()

String [getInstanceID](#AAAAAAAAHM) ()

String [getAssetID](#AAAAAAAAHK) ()

Entity [getEntity](#AAAAAAAAIO) ()

[IInteractiveEntity](#AAAAAAAAGR) [applyStats](#AAAAAAAAHG) ()

[Enemy](#AAAAAAAAII) [setCurrentHealth](#AAAAAAAAHZ) (int health) throws InvalidArgumentException

boolean [move](#AAAAAAAAHS) (double distance)

void [resetMovement](#AAAAAAAAHU) () throws Exception

boolean [takeDamage](#AAAAAAAAIA) (int damage) throws InvalidArgumentException

boolean [heal](#AAAAAAAAHO) (int amount) throws InvalidArgumentException

boolean [isDead](#AAAAAAAAHQ) ()

void [die](#AAAAAAAAHI) ()

void [resurrect](#AAAAAAAAHW) ()

### Detailed Description

This class represents an Enemy in the game. This is an instance of an enemy type. It extends the Identifiable class and implements the IInteractiveEntity interface.

The class contains the following fields:

enemyType: The type of the enemy.

currentHealth: The current health of the enemy.

condition: The condition of the enemy.

currentMovement: The current movement speed of the enemy.

The class provides a constructor that initializes these fields and methods to get the enemy type, current health, current movement, instance ID, and asset ID.

### Constructor & Destructor Documentation

#### game.behaviour.entities.enemy.Enemy.Enemy (String *id*, [EnemyType](#AAAAAAAAIK) *enemyType*) throws ArgumentNullException

Constructor for the Enemy class. Initializes the id, enemyType, and condition of the enemy.

##### Parameters

|  |  |
| --- | --- |
| *id* | The id of the enemy. |
| *enemyType* | The type of the enemy. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the enemyType is null. |

### Member Function Documentation

#### [IInteractiveEntity](#AAAAAAAAGR) game.behaviour.entities.enemy.Enemy.applyStats ()

Sets current stat values to the max.

##### Returns

This IInteractiveEntity, to allow for chaining.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGS).

#### void game.behaviour.entities.enemy.Enemy.die ()

Makes the entity die.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHE).

#### String game.behaviour.entities.enemy.Enemy.getAssetID ()

Gets the asset ID of the entity.

##### Returns

The asset ID of the entity.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGU).

#### int game.behaviour.entities.enemy.Enemy.getCurrentHealth ()

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGR).

#### double game.behaviour.entities.enemy.Enemy.getCurrentMovement ()

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGR).

#### Entity game.behaviour.entities.enemy.Enemy.getEntity ()

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGR).

#### String game.behaviour.entities.enemy.Enemy.getInstanceID ()

Gets the instance ID of the entity.

##### Returns

The instance ID of the entity.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGT).

#### boolean game.behaviour.entities.enemy.Enemy.heal (int *amount*) throws InvalidArgumentException

Heals the entity by a certain amount.

##### Parameters

|  |  |
| --- | --- |
| *amount* | The amount to heal the entity. |

##### Returns

True if the entity was able to be healed, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the amount is less than 0. |

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHC).

#### boolean game.behaviour.entities.enemy.Enemy.isDead ()

Checks if the entity is dead.

##### Returns

True if the entity is dead, false otherwise.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHD).

#### boolean game.behaviour.entities.enemy.Enemy.move (double *distance*)

Moves the entity a certain distance.

##### Parameters

|  |  |
| --- | --- |
| *distance* | The distance to move the entity. |

##### Returns

True if the entity was able to move the specified distance, false otherwise.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGZ).

#### void game.behaviour.entities.enemy.Enemy.resetMovement () throws Exception

Resets the movement of the entity.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the reset. |

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHA).

#### void game.behaviour.entities.enemy.Enemy.resurrect ()

Resurrects the entity.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHF).

#### [Enemy](#AAAAAAAAII) game.behaviour.entities.enemy.Enemy.setCurrentHealth (int *amount*) throws InvalidArgumentException

Sets the current health of the entity.

##### Parameters

|  |  |
| --- | --- |
| *amount* | The amount to set the current health to. |

##### Returns

This IInteractiveEntity, to allow for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the amount is less than 0. |

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGY).

#### boolean game.behaviour.entities.enemy.Enemy.takeDamage (int *damage*) throws InvalidArgumentException

Makes the entity take a certain amount of damage.

##### Parameters

|  |  |
| --- | --- |
| *damage* | The amount of damage for the entity to take. |

##### Returns

True if the entity was able to take the damage, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the damage is less than 0. |

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHB).

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

game/behaviour/entities/enemy/Enemy.java

## game.behaviour.abstracts.EnemyBehaviourController Class Reference

### Public Member Functions

EnemyBehaviourControllerType **getControllerType** ()

EnemyEntity **getEnemyEntity** ()

abstract void [runEnemy](#AAAAAAAAEB) (IInteractiveEntity target, double distance) throws Exception

void [addEventListeners](#AAAAAAAAEC) (IEventListener attemptedToHitListener, IEventListener damageEventListener)

### Static Public Member Functions

static [EnemyBehaviourController](#AAAAAAAADY) [getTypeInstance](#AAAAAAAAED) (EnemyBehaviourControllerType type, EnemyEntity entity) throws ArgumentNullException

### Protected Member Functions

**EnemyBehaviourController** (EnemyEntity enemy) throws ArgumentNullException

void [setEvents](#AAAAAAAAEF) ()

### Protected Attributes

EnemyEntity **enemyEntity**

EnemyBehaviourControllerType **controllerType**

Event **attemptedToHitEvent**

Event **damageEvent**

### Detailed Description

This abstract class represents an EnemyBehaviourController in the game. It includes properties such as enemyEntity, controllerType, attemptedToHitEvent, and damageEvent.

The class contains the following fields:

enemyEntity: The enemy entity this controller is controlling.

controllerType: The type of the controller.

attemptedToHitEvent: Event that is triggered when the enemy attempts to hit.

damageEvent: Event that is triggered when the enemy takes damage.

The class provides a constructor that initializes these fields and a setEvents method that sets up the events.

### Member Function Documentation

#### void game.behaviour.abstracts.EnemyBehaviourController.addEventListeners (IEventListener *attemptedToHitListener*, IEventListener *damageEventListener*)

Adds event listeners to the attemptedToHitEvent and damageEvent.

##### Parameters

|  |  |
| --- | --- |
| *attemptedToHitListener* | The listener for the attemptedToHitEvent. |
| *damageEventListener* | The listener for the damageEvent. |

#### static [EnemyBehaviourController](#AAAAAAAADY) game.behaviour.abstracts.EnemyBehaviourController.getTypeInstance (EnemyBehaviourControllerType *type*, EnemyEntity *entity*) throws ArgumentNullException[static]

Returns an instance of an EnemyBehaviourController of the specified type for the specified enemy entity.

##### Parameters

|  |  |
| --- | --- |
| *type* | The type of the EnemyBehaviourController. |
| *entity* | The enemy entity the controller will control. |

##### Returns

An instance of an EnemyBehaviourController of the specified type.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the entity is null. |

#### abstract void game.behaviour.abstracts.EnemyBehaviourController.runEnemy (IInteractiveEntity *target*, double *distance*) throws Exception[abstract]

Executes the enemy's behaviour towards a target.

##### Parameters

|  |  |
| --- | --- |
| *target* | The target of the enemy's behaviour. |
| *distance* | The distance to the target. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the execution of the behaviour. |

#### void game.behaviour.abstracts.EnemyBehaviourController.setEvents ()[protected]

Sets up the events for the controller. Initializes the attemptedToHitEvent and damageEvent and sets their removeOnRun property to false.

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

game/behaviour/abstracts/EnemyBehaviourController.java

## game.enums.EnemyBehaviourControllerType Enum Reference

### Public Attributes

**RANGER**

**BERSERK**

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

game/enums/EnemyBehaviourControllerType.java

## game.behaviour.entities.enemy.EnemyEntity Class Reference

### Public Member Functions

[EnemyEntity](#AAAAAAAAIQ) (int health, int movement, int level) throws InvalidArgumentException, ArgumentNullException

[EnemyEntity](#AAAAAAAAIP) [setRewardXP](#AAAAAAAAIR) (int xp) throws InvalidArgumentException

int **getRewardXP** ()

#### Public Member Functions inherited from [game.behaviour.entities.InventoryEntity](#AAAAAAAAIC)

[InventoryEntity](#AAAAAAAAID) (int health, int movement, int level, boolean removeConsumableFromInventoryWhenRanOut) throws InvalidArgumentException, ArgumentNullException

void [createInventory](#AAAAAAAAIE) (boolean removeWhenRanOut)

void [addToInventory](#AAAAAAAAIF) (Item item) throws ArgumentNullException

[Inventory](#AAAAAAAADG) **getInventory** ()

### Additional Inherited Members

#### Protected Attributes inherited from [game.behaviour.entities.InventoryEntity](#AAAAAAAAIC)

transient [Inventory](#AAAAAAAADG) **inventory**

### Detailed Description

This class represents an EnemyEntity in the game. It extends the InventoryEntity class and includes an additional property: rewardXP.

The class contains the following fields:

rewardXP: The experience points rewarded when the enemy is defeated.

The class provides a constructor that initializes these fields and methods to set and get the rewardXP.

### Constructor & Destructor Documentation

#### game.behaviour.entities.enemy.EnemyEntity.EnemyEntity (int *health*, int *movement*, int *level*) throws InvalidArgumentException, ArgumentNullException

Constructor for the EnemyEntity class. Initializes the health, movement, level, and rewardXP of the entity.

##### Parameters

|  |  |
| --- | --- |
| *health* | The health of the entity. |
| *movement* | The movement speed of the entity. |
| *level* | The level of the entity. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the health, movement, or level is invalid. |
| *ArgumentNullException* | if the inventory is null. |

### Member Function Documentation

#### [EnemyEntity](#AAAAAAAAIP) game.behaviour.entities.enemy.EnemyEntity.setRewardXP (int *xp*) throws InvalidArgumentException

Sets the experience points rewarded when the enemy is defeated.

##### Parameters

|  |  |
| --- | --- |
| *xp* | The experience points to set. |

##### Returns

This EnemyEntity, to allow for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the xp is less than 0. |

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

game/behaviour/entities/enemy/EnemyEntity.java

## file.elements.EnemyMapData Class Reference

### Public Member Functions

**EnemyMapData** (String assetID, String instanceID) throws ArgumentNullException

String **getAssetID** ()

String **getInstanceID** ()

GridPosition **getPosition** ()

[EnemyMapData](#AAAAAAAAAZ) **setPosition** (GridPosition position) throws ArgumentNullException

### Detailed Description

This class represents the data of an enemy on the map. It includes the asset ID, instance ID, and position of the enemy.

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

file/elements/EnemyMapData.java

## game.behaviour.entities.enemy.EnemyType Class Reference

### Public Member Functions

[EnemyType](#AAAAAAAAIT) (String id, EnemyBehaviourController enemyController) throws ArgumentNullException

EnemyBehaviourController **getController** ()

[EnemyEntity](#AAAAAAAAIP) **getEntity** ()

### Detailed Description

This class represents an EnemyType in the game. It extends the Identity class and includes an additional property: controller.

The class contains the following fields:

controller: The behaviour controller of the enemy type.

The class provides a constructor that initializes these fields and methods to get the controller and the enemy entity.

### Constructor & Destructor Documentation

#### game.behaviour.entities.enemy.EnemyType.EnemyType (String *id*, EnemyBehaviourController *enemyController*) throws ArgumentNullException

Constructor for the EnemyType class. Initializes the id and controller of the enemy type.

##### Parameters

|  |  |
| --- | --- |
| *id* | The id of the enemy type. This is the assetID of the enemy type. |
| *enemyController* | The behaviour controller of the enemy type. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the enemyController is null. |

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

game/behaviour/entities/enemy/EnemyType.java

## file.elements.EnemyTypeSave Class Reference

### Public Attributes

String **enemyTypeID**

EnemyEntity **entity**

EnemyBehaviourControllerType **controllerType**

String **iconFilePath**

String **enemyArmorID**

String **enemyWeaponID**

ArrayList< String > **inventory**

### Detailed Description

This class represents the saved data of an enemy type. It includes the enemy's type ID, entity, behavior controller type, icon file path, armor ID, weapon ID, and inventory.

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

file/elements/EnemyTypeSave.java

## game.global.storage.EnemyTypeStorage Class Reference

### Static Public Member Functions

static [EnemyTypeStorage](#AAAAAAAAOC) **getInstance** ()

### Additional Inherited Members

#### Public Member Functions inherited from [game.global.storage.Storage< EnemyType >](#AAAAAAAANZ)

void **add** (String id, T item) throws ArgumentNullException

void **remove** (String id) throws ArgumentNullException

T **get** (String id) throws ArgumentNullException

boolean **contains** (String id) throws ArgumentNullException

void **clear** ()

Set< Entry< String, T > > **entrySet** ()

ArrayList< T > [getAllItems](#AAAAAAAAOA) ()

ArrayList< String > [getAllKeys](#AAAAAAAAOB) ()

#### Protected Attributes inherited from [game.global.storage.Storage< EnemyType >](#AAAAAAAANZ)

HashMap< String, T > **storage**

### Detailed Description

This class represents an EnemyTypeStorage in the game. It stores all the enemy types in the game so that they can be reused. The key is the ID of the enemy type. It extends the Storage class with EnemyType.

The class contains the following fields:

instance: The singleton instance of the EnemyTypeStorage class.

The class provides a private constructor and a method to get the singleton instance.

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

game/global/storage/EnemyTypeStorage.java

## game.behaviour.abstracts.Entity Class Reference

### Public Member Functions

int **getHealth** ()

int **getLevel** ()

String **getName** ()

[Weapon](#AAAAAAAAEO) **getWeapon** ()

Armor **getArmor** ()

EntityType **getEntityType** ()

[Entity](#AAAAAAAAEK) **setName** (String name) throws ArgumentNullException

[Entity](#AAAAAAAAEK) **setHealth** (int health) throws InvalidArgumentException

[Entity](#AAAAAAAAEK) **setMovement** (int movement) throws InvalidArgumentException

[Entity](#AAAAAAAAEK) **setLevel** (int level) throws InvalidArgumentException

double [getMovement](#AAAAAAAAEW) ()

int [getArmorClass](#AAAAAAAAEX) ()

void [equip](#AAAAAAAAEY) ([Weapon](#AAAAAAAAEO) weapon) throws ArgumentNullException

void [equip](#AAAAAAAAEZ) (Armor armor) throws ArgumentNullException

boolean [attack](#AAAAAAAAFA) (int targetAC, double distance) throws Exception

int [damage](#AAAAAAAAFB) (double distance) throws Exception

### Protected Member Functions

**Entity** (int health, int movement, int level) throws InvalidArgumentException, ArgumentNullException

### Protected Attributes

String **name**

EntityType **entityType**

int **health**

double **movement**

int **level**

transient Armor **armor**

transient [Weapon](#AAAAAAAAEO) **weapon**

### Detailed Description

This abstract class represents an Entity in the game. It includes properties such as name, entityType, health, movement, level, armor, and weapon.

The class contains the following fields:

name: The name of the entity.

entityType: The type of the entity.

health: The health of the entity.

movement: The movement speed of the entity.

level: The level of the entity.

armor: The armor the entity is wearing.

weapon: The weapon the entity is wielding.

The class provides getter methods for these fields and a setter for the name.

### Member Function Documentation

#### boolean game.behaviour.abstracts.Entity.attack (int *targetAC*, double *distance*) throws Exception

Performs an attack roll with the equipped weapon against a target's armor class.

##### Parameters

|  |  |
| --- | --- |
| *targetAC* | The armor class of the target. |
| *distance* | The distance to the target. |

##### Returns

true if the attack hits, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if the entity has no weapon equipped. |

#### int game.behaviour.abstracts.Entity.damage (double *distance*) throws Exception

Calculates the damage dealt by the entity's weapon.

##### Parameters

|  |  |
| --- | --- |
| *distance* | The distance to the target. |

##### Returns

The calculated damage.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if the entity has no weapon equipped. |

#### void game.behaviour.abstracts.Entity.equip (Armor *armor*) throws ArgumentNullException

Equips the entity with armor.

##### Parameters

|  |  |
| --- | --- |
| *armor* | The armor to equip. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the armor is null. |

#### void game.behaviour.abstracts.Entity.equip ([Weapon](#AAAAAAAAEO) *weapon*) throws ArgumentNullException

Equips the entity with a weapon.

##### Parameters

|  |  |
| --- | --- |
| *weapon* | The weapon to equip. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the weapon is null. |

#### int game.behaviour.abstracts.Entity.getArmorClass ()

Calculates and returns the armor class of the entity. If the entity has no armor equipped, it returns the sum of the entity's level and 5. If the entity has armor equipped, it returns the sum of the armor's armor class and the entity's level.

##### Returns

The armor class of the entity.

#### double game.behaviour.abstracts.Entity.getMovement ()

Gets the movement speed of the entity.

##### Returns

The movement speed of the entity. Applies armor bonus if armor is equipped.

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

game/behaviour/abstracts/Entity.java

## game.enums.EntityCondition Enum Reference

### Public Attributes

**NORMAL**

**DEAD**

**INCAPACITATED**

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

game/enums/EntityCondition.java

## game.enums.EntityType Enum Reference

### Public Attributes

**ENEMY**

**PLAYER**

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

game/enums/EntityType.java

## game.behaviour.abstracts.Equipment Class Reference

### Public Member Functions

EquipmentType **getEquipmentType** ()

### Protected Attributes

EquipmentType **equipmentType**

### Detailed Description

This abstract class represents an Equipment in the game. It extends the Item class and includes additional properties such as equipment type.

The class contains the following fields:

equipmentType: The type of the equipment.

The class provides getter methods for these fields.

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

game/behaviour/abstracts/Equipment.java

## ui.elements.EquipmentItemPanel Class Reference

### Public Member Functions

[EquipmentItemPanel](#AAAAAAAARI) (int panelWidth, int panelHeight, int imgWidth, int imgHeight, String title) throws ArgumentNullException

[EquipmentItemPanel](#AAAAAAAAQY) [setUpContent](#AAAAAAAARJ) (IDisplayable item, String iconPath, int bearerLevel, boolean extended, ActionListener listener, boolean equipable) throws ArgumentNullException, IOException

### Public Attributes

int **panelWidth** = 280

int **panelHeight** = 280

int **imgWidth** = 100

int **imgHeight** = 100

### Detailed Description

This class represents an equipment item panel in the game. It is used to display an item in the character frame. It extends the JPanel class and contains methods to initialize the panel and set up its content.

The class contains the following fields:

panelWidth: The width of the panel.

panelHeight: The height of the panel.

imgWidth: The width of the image.

imgHeight: The height of the image.

### Constructor & Destructor Documentation

#### ui.elements.EquipmentItemPanel.EquipmentItemPanel (int *panelWidth*, int *panelHeight*, int *imgWidth*, int *imgHeight*, String *title*) throws ArgumentNullException

Constructor for the EquipmentItemPanel class. Initializes the panel with the specified panel width, panel height, image width, image height, and title.

##### Parameters

|  |  |
| --- | --- |
| *panelWidth* | The width of the panel. |
| *panelHeight* | The height of the panel. |
| *imgWidth* | The width of the image. |
| *imgHeight* | The height of the image. |
| *title* | The title of the panel. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the title is null. |

### Member Function Documentation

#### [EquipmentItemPanel](#AAAAAAAAQY) ui.elements.EquipmentItemPanel.setUpContent (IDisplayable *item*, String *iconPath*, int *bearerLevel*, boolean *extended*, ActionListener *listener*, boolean *equipable*) throws ArgumentNullException, IOException

Sets up the content of the panel. Initializes and adds the icon and name label with the specified item, icon path, bearer level, extended status, listener, and equipable status.

##### Parameters

|  |  |
| --- | --- |
| *item* | The item of the panel. |
| *iconPath* | The path of the icon. |
| *bearerLevel* | The level of the bearer. |
| *extended* | The extended status of the panel. If true, the description, text field, and button will be added. |
| *listener* | The listener of the panel. |
| *equipable* | The equipable status of the panel. |

##### Returns

The panel itself, for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the item, icon path, or listener is null. |
| *IOException* | if the setup of the icon or name label throws an IOException. |

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

ui/elements/EquipmentItemPanel.java

## ui.elements.EquipmentPanel Class Reference

### Public Member Functions

[EquipmentPanel](#AAAAAAAARO) (String topText, String botText) throws ArgumentNullException

[EquipmentItemPanel](#AAAAAAAAQY) **getTopPanel** ()

[EquipmentItemPanel](#AAAAAAAAQY) **getBotPanel** ()

### Static Public Attributes

static int **PANEL\_WIDTH** = 300

static int **PANEL\_HEIGHT** = 500

### Detailed Description

This class represents an equipment panel in the game. It stores two non extended equipment item panels. (top and bottom) It extends the JPanel class and contains methods to initialize the panel and set up its content.

The class contains the following fields:

PANEL\_WIDTH: The width of the panel.

PANEL\_HEIGHT: The height of the panel.

topPanel: The top panel of the equipment panel.

botPanel: The bottom panel of the equipment panel.

### Constructor & Destructor Documentation

#### ui.elements.EquipmentPanel.EquipmentPanel (String *topText*, String *botText*) throws ArgumentNullException

Constructor for the EquipmentPanel class. Initializes the panel with the specified top text and bottom text.

##### Parameters

|  |  |
| --- | --- |
| *topText* | The text of the top panel. |
| *botText* | The text of the bottom panel. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the top text or bottom text is null. |

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

ui/elements/EquipmentPanel.java

## game.enums.EquipmentType Enum Reference

### Public Attributes

**WEAPON**

**ARMOR**

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

game/enums/EquipmentType.java

## game.utility.event.Event Class Reference

### Public Member Functions

[Event](#AAAAAAAAQA) ([EventArgument](#AAAAAAAAQB) object)

[EventArgument](#AAAAAAAAQB) **getArgument** ()

[Event](#AAAAAAAAPZ) [setArgument](#AAAAAAAAQD) ([EventArgument](#AAAAAAAAQB) argument)

final boolean [isRemovingOnRun](#AAAAAAAAQE) ()

final void [setRemoveOnRun](#AAAAAAAAQF) (boolean removeOnRun)

final void [addEventListener](#AAAAAAAAQG) ([IEventListener](#AAAAAAAAQH) listener) throws ArgumentNullException

final void [removeEventListener](#AAAAAAAAQI) ([IEventListener](#AAAAAAAAQH) listener)

final void [triggerEvent](#AAAAAAAAQJ) () throws Exception

### Detailed Description

This class represents an event in the game. It contains an event argument and a list of event listeners, and can be set to remove on run.

The class contains the following fields:

eventArgument: The argument of the event.

eventListeners: The list of listeners of the event.

removeOnRun: Whether the event is removed on run.

### Constructor & Destructor Documentation

#### game.utility.event.Event.Event ([EventArgument](#AAAAAAAAQB) *object*)

Constructor for the Event class. Initializes the event with the specified argument, an empty list of listeners, and a remove on run status of false.

##### Parameters

|  |  |
| --- | --- |
| *object* | The argument of the event. |

### Member Function Documentation

#### final void game.utility.event.Event.addEventListener ([IEventListener](#AAAAAAAAQH) *listener*) throws ArgumentNullException

Adds a listener to the event. It will be triggered when the event is triggered.

##### Parameters

|  |  |
| --- | --- |
| *listener* | The listener to add. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the listener is null. |

#### final boolean game.utility.event.Event.isRemovingOnRun ()

Checks whether the event is removed on run.

##### Returns

Whether the event is removed on run.

#### final void game.utility.event.Event.removeEventListener ([IEventListener](#AAAAAAAAQH) *listener*)

Removes a listener from the event.

##### Parameters

|  |  |
| --- | --- |
| *listener* | The listener to remove. |

#### [Event](#AAAAAAAAPZ) game.utility.event.Event.setArgument ([EventArgument](#AAAAAAAAQB) *argument*)

Sets the argument of the event.

##### Parameters

|  |  |
| --- | --- |
| *argument* | The new argument to set. |

##### Returns

The event itself, for chaining.

#### final void game.utility.event.Event.setRemoveOnRun (boolean *removeOnRun*)

Sets whether the event is removed on run.

##### Parameters

|  |  |
| --- | --- |
| *removeOnRun* | The new remove on run status to set. |

#### final void game.utility.event.Event.triggerEvent () throws Exception

Triggers the event. Each listener of the event is run with the event argument and the event itself. If the event is set to remove on run, the listener is removed after it is run.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if any listener throws an exception. |

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

game/utility/event/Event.java

## game.utility.event.EventArgument< T > Class Template Reference

### Public Member Functions

T **getArgument** ()

[EventArgument](#AAAAAAAAQB)< T > [setArgument](#AAAAAAAAQL) (T argument) throws ArgumentNullException

### Detailed Description

This class represents an argument of an event in the game. It contains a generic argument and methods to get and set the argument.

The class contains the following field:

argument: The argument of the event.

### Member Function Documentation

#### [EventArgument](#AAAAAAAAQB)< T > [game.utility.event.EventArgument](#AAAAAAAAQB)< T >.setArgument (T *argument*) throws ArgumentNullException

Sets the argument of the event.

##### Parameters

|  |  |
| --- | --- |
| *argument* | The new argument to set. |

##### Returns

The event argument itself, for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the new argument is null. |

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

game/utility/event/EventArgument.java

## uilogic.FileChooserType Enum Reference

### Public Attributes

**CONFIG**

**PLAYERPROGRESS**

**NEWSAVE**

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

uilogic/FileChooserType.java

## game.global.FileHandler Class Reference

### Public Member Functions

void [loadConfigFile](#AAAAAAAAML) (String filePath) throws Exception

void [loadPlayerProgressSave](#AAAAAAAAMM) (String filePath) throws Exception

void [loadCurrentMap](#AAAAAAAAMN) (String id, List< Item > playerInventory) throws Exception

void [saveProgress](#AAAAAAAAMO) (String filePath, boolean appendFileExtension) throws ArgumentNullException

### Static Public Member Functions

static [FileHandler](#AAAAAAAAMK) [getInstance](#AAAAAAAAMP) ()

### Member Function Documentation

#### static [FileHandler](#AAAAAAAAMK) game.global.FileHandler.getInstance ()[static]

Singleton pattern method to get the instance of FileHandler. If the instance is null, a new instance is created.

##### Returns

the instance of FileHandler.

#### void game.global.FileHandler.loadConfigFile (String *filePath*) throws Exception

Loads the configuration file from the given file path. It also loads the maps and player progress save.

##### Parameters

|  |  |
| --- | --- |
| *filePath* | The path of the configuration file. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if the filePath is null or if the file does not exist. |

#### void game.global.FileHandler.loadCurrentMap (String *id*, List< Item > *playerInventory*) throws Exception

Loads the current map from the given id and player inventory. Modifies the file path to full path. Clears the ActiveEnemyStorage. Sets the current map layout in the GameHandler. Loads the enemies on the map.

##### Parameters

|  |  |
| --- | --- |
| *id* | The id of the map. |
| *playerInventory* | The player's inventory. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs while loading the map. |

#### void game.global.FileHandler.loadPlayerProgressSave (String *filePath*) throws Exception

Loads the player progress save from the given file path.

##### Parameters

|  |  |
| --- | --- |
| *filePath* | The path of the player progress save file. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs while loading the player progress save. |

#### void game.global.FileHandler.saveProgress (String *filePath*, boolean *appendFileExtension*) throws ArgumentNullException

Saves the current game progress to the given file path. It saves the current map, player progress, and modified enemies.

##### Parameters

|  |  |
| --- | --- |
| *filePath* | The path where the progress will be saved. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the filePath is null. |

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

game/global/FileHandler.java

## file.FileIOUtil Class Reference

### Public Member Functions

Object **readObjectFromFile** (String filePath) throws ArgumentNullException, FileNotFoundException, IOException, ClassNotFoundException

Object [readObjectFromFile](#AAAAAAAAAW) (File file) throws ArgumentNullException, FileNotFoundException, IOException, ClassNotFoundException

void [writeObjectToFile](#AAAAAAAAAX) (String filePath, Serializable object) throws ArgumentNullException, FileNotFoundException, IOException

String [readFile](#AAAAAAAAAY) (String filePath) throws ArgumentNullException, FileNotFoundException

### Member Function Documentation

#### String file.FileIOUtil.readFile (String *filePath*) throws ArgumentNullException, FileNotFoundException

Reads a file and returns its content as a String.

##### Parameters

|  |  |
| --- | --- |
| *filePath* | The path of the file to read. |

##### Returns

The content of the file as a String.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | If the filePath is null. |
| *FileNotFoundException* | If the file does not exist. |

#### Object file.FileIOUtil.readObjectFromFile (File *file*) throws ArgumentNullException, FileNotFoundException, IOException, ClassNotFoundException

Reads an object from a file.

##### Parameters

|  |  |
| --- | --- |
| *file* | The file to read the object from. |

##### Returns

The object read from the file.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | If the file is null. |
| *FileNotFoundException* | If the file does not exist. |
| *IOException* | If an I/O error occurs while reading. |
| *ClassNotFoundException* | If the class of a serialized object cannot be found. |

#### void file.FileIOUtil.writeObjectToFile (String *filePath*, Serializable *object*) throws ArgumentNullException, FileNotFoundException, IOException

Writes a Serializable object to a file.

##### Parameters

|  |  |
| --- | --- |
| *filePath* | The path of the file to write the object to. |
| *object* | The object to write to the file. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | If the filePath or the object is null. |
| *FileNotFoundException* | If the file does not exist. |
| *IOException* | If an I/O error occurs while writing. |

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

file/FileIOUtil.java

## game.GameActionController Class Reference

### Public Member Functions

void [resetPlayerAttacksLeft](#AAAAAAAADA) ()

void [runEnemyTurns](#AAAAAAAADB) ()

void [playerPickUpAction](#AAAAAAAADC) ()

void [playerMoveAction](#AAAAAAAADD) ()

void [playerAttackAction](#AAAAAAAADE) ()

void [playerEndTurnAction](#AAAAAAAADF) ()

### Detailed Description

This class controls the actions in the game. It contains methods to reset the player's attacks left, run the enemy turns, and handle the player's pick up action.

The class contains the following fields:

playerAttacksLeft: The number of attacks left for the player.

### Member Function Documentation

#### void game.GameActionController.playerAttackAction ()

Handles the player's action of attacking an enemy. If the player is in a condition that allows attacking and the selected tile contains an enemy, the player attacks the enemy. If the enemy dies from the attack, the enemy death is handled. If the enemy does not die, the enemy data is modified.

#### void game.GameActionController.playerEndTurnAction ()

Handles the player's action of ending their turn. If the player is in a condition that allows ending their turn, the player's attacks left and movement are reset, the player controls are toggled off, and the enemy turns are run.

#### void game.GameActionController.playerMoveAction ()

Handles the player's action of moving. If the player is in a condition that allows movement, the player is moved to the selected tile.

#### void game.GameActionController.playerPickUpAction ()

Handles the player's action of picking up an item. If the player is in a condition that allows the action, the action is performed.

#### void game.GameActionController.resetPlayerAttacksLeft ()

Resets the number of attacks left for the player. The number of attacks left is set to the number of attacks in a round of the player's weapon.

#### void game.GameActionController.runEnemyTurns ()

Runs the turns for all the active enemies in ActiveEnemyStorage. For each enemy, if the player is not dead, the enemy's controller runs the enemy's turn. After all the enemy turns, the player controls are toggled on.

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

game/GameActionController.java

## file.elements.GameConfigSave Class Reference

### Public Attributes

String **itemFolder**

String **enemyFolder**

String **mapdataFolder**

String **imageAssetFolder**

String **defaultMapID**

String **defaultPlayerSaveFile**

### Detailed Description

This class represents the saved game configuration. It includes the game settings, player settings, and other related data.

The class contains the following fields:

itemFolder: The relative path to the folder containing item configuration files.

enemyFolder: The relative path to the folder containing enemy configuration files.

mapdataFolder: The relative path to the folder containing map data files.

imageAssetFolder: The relative path to the folder containing image assets.

defaultMapID: The ID of the default map to be loaded.

defaultPlayerSaveFile: The relative path to the default player save file.

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

file/elements/GameConfigSave.java

## game.global.GameHandler Class Reference

### Public Member Functions

[GameActionController](#AAAAAAAACZ) **getActionController** ()

[SaveHandler](#AAAAAAAAMS) **getSaveHandler** ()

Player **getPlayer** ()

void **setSessionPlayer** (Player player) throws ArgumentNullException

void [start](#AAAAAAAAMW) () throws Exception

void [setCurrentMapLayout](#AAAAAAAAMX) (MapLayoutData data) throws Exception

void [handleChosenFile](#AAAAAAAAMY) (String filePath, FileChooserType type) throws ArgumentNullException, FileNotFoundException

void [handleEnemyDeath](#AAAAAAAAMZ) (Enemy enemy) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException

void [modifyEnemy](#AAAAAAAANA) (Enemy enemy, boolean dead) throws ArgumentNullException

void [handlePlayerDeath](#AAAAAAAANB) ()

void [handlePlayerLevelUp](#AAAAAAAANC) ()

void [quitGame](#AAAAAAAAND) (boolean instant)

boolean [checkPlayerConditionForAction](#AAAAAAAANE) ()

### Static Public Member Functions

static [GameHandler](#AAAAAAAAMQ) **getInstance** ()

### Detailed Description

This class represents a GameHandler in the game. It is responsible for handling the main game logic. It is a singleton class that handles the game actions and saving.

The class contains the following fields:

instance: The singleton instance of the GameHandler class.

actionController: The GameActionController that handles the game actions.

saveHandler: The SaveHandler that handles the game saving.

player: The Player of the game.

gameTitle: The title of the game.

gameCreator: The creator of the game.

The class provides a private constructor and a method to get the singleton instance.

### Member Function Documentation

#### boolean game.global.GameHandler.checkPlayerConditionForAction ()

Checks the player's condition for action. And wheter the player is prevented from executing an action. If the player is dead, it logs a message and returns true. Otherwise, it returns false.

##### Returns

Whether the player is dead.

#### void game.global.GameHandler.handleChosenFile (String *filePath*, FileChooserType *type*) throws ArgumentNullException, FileNotFoundException

Handles what happens to the chosen file based on the given type. Loads the game from the chosen file.

##### Parameters

|  |  |
| --- | --- |
| *file* | The chosen file. |

#### void game.global.GameHandler.handleEnemyDeath (Enemy *enemy*) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException

Handles the death of an enemy. Removes the enemy from the active enemies and updates the UI. Removes the enemy from the active enemies and updates the UI.

##### Parameters

|  |  |
| --- | --- |
| *enemy* | The enemy that died. |

#### void game.global.GameHandler.handlePlayerDeath ()

Handles the death of the player. Disables the player controls and displays the player death screen. Disables the player controls and displays the player death screen.

#### void game.global.GameHandler.handlePlayerLevelUp ()

Handles the level up of the player. Displays a message indicating the player has leveled up. Displays a message indicating the player has leveled up.

#### void game.global.GameHandler.modifyEnemy (Enemy *enemy*, boolean *dead*) throws ArgumentNullException

Modifies an enemy. Saves the enemy's data and position in the ModifiedEnemyStorage. Updates the enemy's data and position.

##### Parameters

|  |  |
| --- | --- |
| *enemy* | The enemy to modify. |
| *enemyPosition* | The new position of the enemy. |

#### void game.global.GameHandler.quitGame (boolean *instant*)

Quits the game. Prompts the player for confirmation before quitting. If instant is true, it quits the game immediately. Otherwise, it asks the player for confirmation before quitting.

##### Parameters

|  |  |
| --- | --- |
| *instant* | Whether to quit the game immediately. |

#### void game.global.GameHandler.setCurrentMapLayout (MapLayoutData *data*) throws Exception

Sets the current map layout.

##### Parameters

|  |  |
| --- | --- |
| *layout* | The layout to set. |

#### void game.global.GameHandler.start () throws Exception

Starts the game. Must be called first. When program starts. Initializes the game and displays the main menu.

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

game/global/GameHandler.java

## game.utility.GenericDelegate Interface Reference

### Public Member Functions

void [run](#AAAAAAAAOZ) (Object o)

### Detailed Description

This interface represents a generic delegate in the game. It is used to pass a callback with generic parameters. It contains a single method, run, that takes an object as a parameter.

### Member Function Documentation

#### void game.utility.GenericDelegate.run (Object *o*)

Runs the delegate with the specified object.

##### Parameters

|  |  |
| --- | --- |
| *o* | The object to run the delegate with. |

#### The documentation for this interface was generated from the following file:

game/utility/GenericDelegate.java

## ui.elements.GridButton Class Reference

### Public Member Functions

[GridButton](#AAAAAAAARU) (int width, int height, Color borderColor, GridPosition position, ActionListener handler)

[GridButton](#AAAAAAAARV) (int width, int height, int x, int y, Color borderColor, ActionListener handler) throws InvalidArgumentException

void [hightlightButton](#AAAAAAAARW) (boolean highlight)

GridPosition **getGridPosition** ()

GridPosition **setGridPosition** (GridPosition newPosition) throws ArgumentNullException

### Detailed Description

This class represents a grid button in the game. It is placed on the top of an InteractiveGrid panel. It extends the JButton class and implements the IGridPositionable interface.

The class contains the following field:

gridPosition: The grid position of the grid button.

### Constructor & Destructor Documentation

#### ui.elements.GridButton.GridButton (int *width*, int *height*, Color *borderColor*, GridPosition *position*, ActionListener *handler*)

Constructor for the GridButton class. Initializes the grid button with the specified width, height, border color, grid position, and handler.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the grid button. |
| *height* | The height of the grid button. |
| *borderColor* | The border color of the grid button. |
| *position* | The grid position of the grid button. |
| *handler* | The handler of the grid button. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the border color, grid position, or handler is null. |

#### ui.elements.GridButton.GridButton (int *width*, int *height*, int *x*, int *y*, Color *borderColor*, ActionListener *handler*) throws InvalidArgumentException

Constructor for the GridButton class. Initializes the grid button with the specified width, height, x, y, border color, and handler.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the grid button. |
| *height* | The height of the grid button. |
| *x* | The x of the grid button. |
| *y* | The y of the grid button. |
| *borderColor* | The border color of the grid button. |
| *handler* | The handler of the grid button. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the x or y is invalid. |
| *ArgumentNullException* | if the border color or handler is null. |

### Member Function Documentation

#### void ui.elements.GridButton.hightlightButton (boolean *highlight*)

Highlights the grid button. Sets the opacity, content area filled status, and background color of the grid button.

##### Parameters

|  |  |
| --- | --- |
| *highlight* | A boolean that determines whether to highlight the grid button. |

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

ui/elements/GridButton.java

## uilogic.GridButtonHandler Class Reference

### Public Member Functions

[GridButtonHandler](#AAAAAAAAVP) (GenericDelegate delegate, boolean highlightButton) throws ArgumentNullException

void [clearSelected](#AAAAAAAAVQ) ()

void [actionPerformed](#AAAAAAAAVR) (ActionEvent e)

### Detailed Description

This class handles the actions of grid buttons in the UI. It implements the ActionListener interface and contains a GenericDelegate for handling grid button actions, a GridButton for the last selected grid button, and a boolean for whether to highlight the selected grid button.

The class contains the following fields:

delegate: The GenericDelegate for handling grid button actions.

lastSelected: The GridButton for the last selected grid button.

highlightButton: The boolean for whether to highlight the selected grid button.

### Constructor & Destructor Documentation

#### uilogic.GridButtonHandler.GridButtonHandler (GenericDelegate *delegate*, boolean *highlightButton*) throws ArgumentNullException

Constructor for the GridButtonHandler class. Initializes the GridButtonHandler with a GenericDelegate for handling grid button actions and a boolean for whether to highlight the selected grid button.

##### Parameters

|  |  |
| --- | --- |
| *delegate* | The GenericDelegate for handling grid button actions. |
| *highlightButton* | The boolean for whether to highlight the selected grid button on press. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the delegate is null. |

### Member Function Documentation

#### void uilogic.GridButtonHandler.actionPerformed (ActionEvent *e*)

Handles the action of a grid button. If the highlightButton field is true, clears the selected grid button, gets the source of the action event, sets the last selected grid button to the source, and highlights the last selected grid button.

##### Parameters

|  |  |
| --- | --- |
| *e* | The action event. |

#### void uilogic.GridButtonHandler.clearSelected ()

Clears the selected grid button. If the last selected grid button is not null, unhighlights the last selected grid button, revalidates and repaints its parent, and sets the last selected grid button to null.

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

uilogic/GridButtonHandler.java

## ui.data.GridDimension Class Reference

### Public Member Functions

[GridDimension](#AAAAAAAAQO) (int x, int y)

int **getHorizontal** ()

int **getVertical** ()

### Detailed Description

This class represents a dimension in a grid. It extends the Dimension class and contains methods to get the horizontal and vertical dimensions.

### Constructor & Destructor Documentation

#### ui.data.GridDimension.GridDimension (int *x*, int *y*)

Constructor for the GridDimension class. Initializes the dimension with the specified horizontal and vertical dimensions.

##### Parameters

|  |  |
| --- | --- |
| *x* | The horizontal dimension. |
| *y* | The vertical dimension. |

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

ui/data/GridDimension.java

## ui.elements.GridEntityComponent Class Reference

### Public Member Functions

[GridEntityComponent](#AAAAAAAASA) (String id, int width, int height, GridPosition position) throws ArgumentNullException

String [getID](#AAAAAAAASB) ()

[GridEntityComponent](#AAAAAAAARZ) [setImage](#AAAAAAAASC) (String filePath) throws ArgumentNullException, IOException

GridPosition **getGridPosition** ()

GridPosition **setGridPosition** (GridPosition newPosition) throws ArgumentNullException

#### Public Member Functions inherited from [ui.elements.ImageComponent](#AAAAAAAASF)

[ImageComponent](#AAAAAAAASG) (int width, int height)

[ImageComponent](#AAAAAAAASH) (int width, int height, String filePath) throws ArgumentNullException, IOException

void [refresh](#AAAAAAAASI) ()

### Additional Inherited Members

#### Protected Attributes inherited from [ui.elements.ImageComponent](#AAAAAAAASF)

GridDimension **preferredSize**

### Detailed Description

This class represents a grid entity component in the game. It is used to display and store the position of an entity on the grid. It extends the ImageComponent class and implements the IGridPositionable interface.

The class contains the following fields:

id: The id of the grid entity component.

gridPosition: The grid position of the grid entity component.

### Constructor & Destructor Documentation

#### ui.elements.GridEntityComponent.GridEntityComponent (String *id*, int *width*, int *height*, GridPosition *position*) throws ArgumentNullException

Constructor for the GridEntityComponent class. Initializes the grid entity component with the specified id, width, height, and grid position.

##### Parameters

|  |  |
| --- | --- |
| *id* | The id of the grid entity component. |
| *width* | The width of the grid entity component. |
| *height* | The height of the grid entity component. |
| *position* | The grid position of the grid entity component. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the id or grid position is null. |

### Member Function Documentation

#### String ui.elements.GridEntityComponent.getID ()

Gets the id of the grid entity component.

##### Returns

The id of the grid entity component.

#### [GridEntityComponent](#AAAAAAAARZ) ui.elements.GridEntityComponent.setImage (String *filePath*) throws ArgumentNullException, IOException

Sets the image of the image component.

##### Parameters

|  |  |
| --- | --- |
| *filePath* | The file path of the image. |

##### Returns

The image component itself, for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the file path is null. |
| *IOException* | if the file does not exist or is a directory. |

Reimplemented from [ui.elements.ImageComponent](#AAAAAAAASJ).

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

ui/elements/GridEntityComponent.java

## uilogic.GridEntityComponentHandler Class Reference

### Public Member Functions

[GridEntityComponentHandler](#AAAAAAAAVT) ()

boolean [isEmpty](#AAAAAAAAVU) ()

void [clear](#AAAAAAAAVV) ()

GridEntityComponent [add](#AAAAAAAAVW) (GridEntityComponent entity) throws ArgumentNullException

GridEntityComponent [getByID](#AAAAAAAAVX) (String id) throws ArgumentNullException, ElementNotFoundException

GridEntityComponent [getByPosition](#AAAAAAAAVY) ([GridPosition](#AAAAAAAAVZ) position) throws ArgumentNullException, ElementNotFoundException

GridEntityComponent [remove](#AAAAAAAAWA) (String id) throws ArgumentNullException, ElementNotFoundException

GridEntityComponent [remove](#AAAAAAAAWB) (GridEntityComponent entity) throws ArgumentNullException

### Detailed Description

This class handles the grid entity components in the UI. It stores the grid entity components in a HashMap. The keys are the IDs of the grid entity components. It contains a HashMap of GridEntityComponents, with their IDs as keys.

The class contains the following field:

entities: The HashMap of GridEntityComponents.

### Constructor & Destructor Documentation

#### uilogic.GridEntityComponentHandler.GridEntityComponentHandler ()

Constructor for the GridEntityComponentHandler class. Initializes the HashMap of GridEntityComponents.

### Member Function Documentation

#### GridEntityComponent uilogic.GridEntityComponentHandler.add (GridEntityComponent *entity*) throws ArgumentNullException

Adds a GridEntityComponent to the HashMap of GridEntityComponents.

##### Parameters

|  |  |
| --- | --- |
| *entity* | The GridEntityComponent to add. |

##### Returns

The added GridEntityComponent.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the GridEntityComponent is null. |

#### void uilogic.GridEntityComponentHandler.clear ()

Clears the HashMap of GridEntityComponents.

#### GridEntityComponent uilogic.GridEntityComponentHandler.getByID (String *id*) throws ArgumentNullException, ElementNotFoundException

Gets a GridEntityComponent from the HashMap of GridEntityComponents by its ID.

##### Parameters

|  |  |
| --- | --- |
| *id* | The ID of the GridEntityComponent. |

##### Returns

The GridEntityComponent with the specified ID.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the ID is null. |
| *ElementNotFoundException* | if the GridEntityComponent with the specified ID is not found. |

#### GridEntityComponent uilogic.GridEntityComponentHandler.getByPosition ([GridPosition](#AAAAAAAAVZ) *position*) throws ArgumentNullException, ElementNotFoundException

Gets a GridEntityComponent from the HashMap of GridEntityComponents by its position.

##### Parameters

|  |  |
| --- | --- |
| *position* | The position of the GridEntityComponent. |

##### Returns

The GridEntityComponent with the specified position.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the position is null. |
| *ElementNotFoundException* | if the GridEntityComponent with the specified position is not found. |

#### boolean uilogic.GridEntityComponentHandler.isEmpty ()

Checks if the HashMap of GridEntityComponents is empty.

##### Returns

A boolean indicating whether the HashMap of GridEntityComponents is empty.

#### GridEntityComponent uilogic.GridEntityComponentHandler.remove (GridEntityComponent *entity*) throws ArgumentNullException

Removes a GridEntityComponent from the HashMap of GridEntityComponents.

##### Parameters

|  |  |
| --- | --- |
| *entity* | The GridEntityComponent to remove. |

##### Returns

The removed GridEntityComponent.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the GridEntityComponent is null. |

#### GridEntityComponent uilogic.GridEntityComponentHandler.remove (String *id*) throws ArgumentNullException, ElementNotFoundException

Removes a GridEntityComponent from the HashMap of GridEntityComponents by its ID.

##### Parameters

|  |  |
| --- | --- |
| *id* | The ID of the GridEntityComponent. |

##### Returns

The removed GridEntityComponent.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the ID is null. |
| *ElementNotFoundException* | if the GridEntityComponent with the specified ID is not found. |

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

uilogic/GridEntityComponentHandler.java

## ui.elements.GridPanel Class Reference

### Public Member Functions

[GridPanel](#AAAAAAAASL) (int width, int height, int preferredHorizontalComponentCount, int preferredVerticalComponentCount)

JPanel [getJPanel](#AAAAAAAASM) ()

void **refresh** ()

void [add](#AAAAAAAASO) (IGridPositionable component, GridPosition position, boolean force, boolean allowOutOfBounds) throws InvalidArgumentException, ComponentAlreadyAtPositionException, ArgumentNullException

IGridPositionable [getComponentAt](#AAAAAAAASP) (GridPosition position) throws ElementNotFoundException

void [remove](#AAAAAAAASQ) (IGridPositionable component) throws ArgumentNullException, InvalidArgumentException, ElementNotFoundException

void [removeAt](#AAAAAAAASR) (GridPosition position) throws ArgumentNullException, ElementNotFoundException

### Detailed Description

This class represents a grid panel in the game.GridPanel stores IGridPositionable objects in a GridBagLayout. Its purpose is to handle the UI components in the layout and manage them by position It stores IGridPositionable objects in a GridBagLayout. Its purpose is to handle the UI components in the layout and manage them by position.

The class contains the following fields:

components: The components of the grid panel.

preferredComponentDimension: The preferred component dimension of the grid panel.

panel: The panel of the grid panel.

### Constructor & Destructor Documentation

#### ui.elements.GridPanel.GridPanel (int *width*, int *height*, int *preferredHorizontalComponentCount*, int *preferredVerticalComponentCount*)

Constructor for the GridPanel class. Initializes the grid panel with the specified width, height, preferred horizontal component count, and preferred vertical component count.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the grid panel. |
| *height* | The height of the grid panel. |
| *preferredHorizontalComponentCount* | The preferred horizontal component count of the grid panel. |
| *preferredVerticalComponentCount* | The preferred vertical component count of the grid panel. |

### Member Function Documentation

#### void ui.elements.GridPanel.add (IGridPositionable *component*, GridPosition *position*, boolean *force*, boolean *allowOutOfBounds*) throws InvalidArgumentException, ComponentAlreadyAtPositionException, ArgumentNullException

Adds a component to the grid panel at the specified grid position. If force is true, it will replace any existing component at the position. If allowOutOfBounds is true, it will allow adding the component outside of the preferred dimension.

##### Parameters

|  |  |
| --- | --- |
| *component* | The component to be added. |
| *position* | The grid position where the component will be added. |
| *force* | A boolean that determines whether to replace any existing component at the position. |
| *allowOutOfBounds* | A boolean that determines whether to allow adding the component outside of the preferred dimension. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the component is not a Component child. |
| *ComponentAlreadyAtPositionException* | if a component already exists at the position and force is false. |
| *ArgumentNullException* | if the component or position is null. |
| *IndexOutOfBoundsException* | if the position is outside of the preferred dimension and allowOutOfBounds is false. |

#### IGridPositionable ui.elements.GridPanel.getComponentAt (GridPosition *position*) throws ElementNotFoundException

Gets the component at the specified grid position.

##### Parameters

|  |  |
| --- | --- |
| *position* | The grid position. |

##### Returns

The component at the specified grid position.

##### Exceptions

|  |  |
| --- | --- |
| *ElementNotFoundException* | if no component is found at the specified grid position. |

#### JPanel ui.elements.GridPanel.getJPanel ()

Gets the JPanel of the grid panel.

##### Returns

The JPanel of the grid panel.

#### void ui.elements.GridPanel.remove (IGridPositionable *component*) throws ArgumentNullException, InvalidArgumentException, ElementNotFoundException

Removes a component from the grid panel.

##### Parameters

|  |  |
| --- | --- |
| *component* | The component to be removed. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the component is null. |
| *InvalidArgumentException* | if the component is not a Component child. |
| *ElementNotFoundException* | if the component does not exist in the grid. |

#### void ui.elements.GridPanel.removeAt (GridPosition *position*) throws ArgumentNullException, ElementNotFoundException

Removes the component at the specified grid position.

##### Parameters

|  |  |
| --- | --- |
| *position* | The grid position. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the position is null. |
| *ElementNotFoundException* | if no component is found at the specified grid position. |

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

ui/elements/GridPanel.java

## uilogic.GridPosition Class Reference

### Public Member Functions

[GridPosition](#AAAAAAAAWC) () throws InvalidArgumentException

[GridPosition](#AAAAAAAAWD) (int x, int y) throws InvalidArgumentException

[GridPosition](#AAAAAAAAWE) (GridBagConstraints gbc) throws InvalidArgumentException

int [getX](#AAAAAAAAWF) ()

int [getY](#AAAAAAAAWG) ()

GridBagConstraints [getAsGridBagConstraints](#AAAAAAAAWH) ()

[GridPosition](#AAAAAAAAVZ) [setPosition](#AAAAAAAAWI) (int x, int y) throws InvalidArgumentException

boolean [equals](#AAAAAAAAWJ) ([GridPosition](#AAAAAAAAVZ) cmp)

### Static Public Member Functions

static double [calculateAbsoluteDistance](#AAAAAAAAWK) ([GridPosition](#AAAAAAAAVZ) src, [GridPosition](#AAAAAAAAVZ) dst) throws ArgumentNullException

### Detailed Description

This class represents a position on a grid. It contains x and y coordinates and implements Serializable for object serialization.

The class contains the following fields:

x: The x-coordinate of the grid position.

y: The y-coordinate of the grid position.

### Constructor & Destructor Documentation

#### uilogic.GridPosition.GridPosition () throws InvalidArgumentException

Default constructor for the GridPosition class. Initializes the grid position with x and y coordinates set to 0.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the x-coordinate or y-coordinate is less than 0. |

#### uilogic.GridPosition.GridPosition (int *x*, int *y*) throws InvalidArgumentException

Constructor for the GridPosition class. Initializes the grid position with specified x and y coordinates.

##### Parameters

|  |  |
| --- | --- |
| *x* | The x-coordinate of the grid position. |
| *y* | The y-coordinate of the grid position. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the x-coordinate or y-coordinate is less than 0. |

#### uilogic.GridPosition.GridPosition (GridBagConstraints *gbc*) throws InvalidArgumentException

Constructor for the GridPosition class. Initializes the grid position with the x and y coordinates of a GridBagConstraints object.

##### Parameters

|  |  |
| --- | --- |
| *gbc* | The GridBagConstraints object. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the x-coordinate or y-coordinate of the GridBagConstraints object is less than 0. |

### Member Function Documentation

#### static double uilogic.GridPosition.calculateAbsoluteDistance ([GridPosition](#AAAAAAAAVZ) *src*, [GridPosition](#AAAAAAAAVZ) *dst*) throws ArgumentNullException[static]

Calculates the absolute distance between two grid positions.

##### Parameters

|  |  |
| --- | --- |
| *src* | The source grid position. |
| *dst* | The destination grid position. |

##### Returns

The absolute distance between the source and destination grid positions.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the source or destination grid position is null. |

#### boolean uilogic.GridPosition.equals ([GridPosition](#AAAAAAAAVZ) *cmp*)

Checks if the grid position is equal to another grid position.

##### Parameters

|  |  |
| --- | --- |
| *cmp* | The grid position to compare with. |

##### Returns

A boolean indicating whether the grid position is equal to the other grid position.

#### GridBagConstraints uilogic.GridPosition.getAsGridBagConstraints ()

Gets the grid position as a GridBagConstraints object.

##### Returns

The GridBagConstraints object representing the grid position.

#### int uilogic.GridPosition.getX ()

Gets the x-coordinate of the grid position.

##### Returns

The x-coordinate of the grid position.

#### int uilogic.GridPosition.getY ()

Gets the y-coordinate of the grid position.

##### Returns

The y-coordinate of the grid position.

#### [GridPosition](#AAAAAAAAVZ) uilogic.GridPosition.setPosition (int *x*, int *y*) throws InvalidArgumentException

Sets the x-coordinate and y-coordinate of the grid position.

##### Parameters

|  |  |
| --- | --- |
| *x* | The x-coordinate to set. |
| *y* | The y-coordinate to set. |

##### Returns

The GridPosition object with the set coordinates.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the x-coordinate or y-coordinate is less than 0. |

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

uilogic/GridPosition.java

## file.elements.IconData Class Reference

### Public Member Functions

**IconData** (String normalPath, String absolutPath) throws ArgumentNullException

String **getNormalPath** ()

String **getAbsolutPath** ()

### Detailed Description

This class represents the icon data of a game element. It includes the icon's normal path and absolute path.

The class contains the following fields:

normalPath: The relative path to the icon.

absolutPath: The absolute path to the icon.

The class provides getter methods for these fields.

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

file/elements/IconData.java

## game.global.storage.IconDataStorage Class Reference

### Static Public Member Functions

static [IconDataStorage](#AAAAAAAAOE) **getInstance** ()

### Additional Inherited Members

#### Public Member Functions inherited from [game.global.storage.Storage< IconData >](#AAAAAAAANZ)

void **add** (String id, T item) throws ArgumentNullException

void **remove** (String id) throws ArgumentNullException

T **get** (String id) throws ArgumentNullException

boolean **contains** (String id) throws ArgumentNullException

void **clear** ()

Set< Entry< String, T > > **entrySet** ()

ArrayList< T > [getAllItems](#AAAAAAAAOA) ()

ArrayList< String > [getAllKeys](#AAAAAAAAOB) ()

#### Protected Attributes inherited from [game.global.storage.Storage< IconData >](#AAAAAAAANZ)

HashMap< String, T > **storage**

### Detailed Description

This class represents an IconDataStorage in the game. It stores all the icons for the game elements. The key is the ID of the element. It extends the Storage class with IconData type.

The class contains the following fields:

instance: The singleton instance of the IconDataStorage class.

The class provides a private constructor and a method to get the singleton instance.

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

game/global/storage/IconDataStorage.java

## game.utility.Identifiable Class Reference

### Public Member Functions

final [Identifiable](#AAAAAAAAPA) [setID](#AAAAAAAAPB) (String newID) throws ArgumentNullException

final String **getID** ()

### Protected Attributes

String **id**

### Detailed Description

This abstract class represents an identifiable object in the game. It contains methods to set and get the ID of the object.

The class contains the following fields:

id: The ID of the game object.

### Member Function Documentation

#### final [Identifiable](#AAAAAAAAPA) game.utility.Identifiable.setID (String *newID*) throws ArgumentNullException

Sets the ID of the object.

##### Parameters

|  |  |
| --- | --- |
| *newID* | The new ID to set. |

##### Returns

The object itself, for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the new ID is null. |

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

game/utility/Identifiable.java

## game.utility.Identity Class Reference

### Public Member Functions

String **getName** ()

String **getDescription** ()

[Identity](#AAAAAAAAPE) [setName](#AAAAAAAAPH) (String name) throws ArgumentNullException

[Identity](#AAAAAAAAPE) [setDescription](#AAAAAAAAPI) (String description) throws ArgumentNullException

#### Public Member Functions inherited from [game.utility.Identifiable](#AAAAAAAAPA)

final [Identifiable](#AAAAAAAAPA) [setID](#AAAAAAAAPB) (String newID) throws ArgumentNullException

final String **getID** ()

### Protected Attributes

String **name**

String **description**

#### Protected Attributes inherited from [game.utility.Identifiable](#AAAAAAAAPA)

String **id**

### Detailed Description

This abstract class represents an identifiable object in the game with a name and description. It extends the Identifiable class and contains methods to set and get the name and description of the object.

The class contains the following fields:

name: The name of the game object.

description: The description of the game object.

### Member Function Documentation

#### [Identity](#AAAAAAAAPE) game.utility.Identity.setDescription (String *description*) throws ArgumentNullException

Sets the description of the object.

##### Parameters

|  |  |
| --- | --- |
| *description* | The new description to set. |

##### Returns

The object itself, for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the new description is null. |

#### [Identity](#AAAAAAAAPE) game.utility.Identity.setName (String *name*) throws ArgumentNullException

Sets the name of the object.

##### Parameters

|  |  |
| --- | --- |
| *name* | The new name to set. |

##### Returns

The object itself, for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the new name is null. |

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

game/utility/Identity.java

## game.utility.IDisplayable Interface Reference

### Public Member Functions

String [getName](#AAAAAAAAPM) ()

String [getDisplayInfo](#AAAAAAAAPN) ()

String [getStatistics](#AAAAAAAAPO) (int bearerLevel)

### Detailed Description

This interface represents a displayable item in the game. It contains methods to get the name, display information, and statistics of the object.

### Member Function Documentation

#### String game.utility.IDisplayable.getDisplayInfo ()

Gets the display information of the object.

##### Returns

The display information of the object.

#### String game.utility.IDisplayable.getName ()

Gets the name of the object.

##### Returns

The name of the object.

#### String game.utility.IDisplayable.getStatistics (int *bearerLevel*)

Gets the statistics of the object.

##### Parameters

|  |  |
| --- | --- |
| *bearerLevel* | The level of the bearer of the item. |

##### Returns

The statistics of the object.

#### The documentation for this interface was generated from the following file:

game/utility/IDisplayable.java

## game.utility.event.IEventListener Interface Reference

### Public Member Functions

void [run](#AAAAAAAAQM) ([EventArgument](#AAAAAAAAQB) object, [Event](#AAAAAAAAPZ) triggeredEvent) throws Exception

### Detailed Description

This interface represents a listener of an event in the game. It contains a method to run the listener with an event argument and the triggered event.

### Member Function Documentation

#### void game.utility.event.IEventListener.run ([EventArgument](#AAAAAAAAQB) *object*, [Event](#AAAAAAAAPZ) *triggeredEvent*) throws Exception

Runs the listener with the specified event argument and triggered event.

##### Parameters

|  |  |
| --- | --- |
| *object* | The argument of the event. |
| *triggeredEvent* | The triggered event. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if the listener throws an exception. |

#### The documentation for this interface was generated from the following file:

game/utility/event/IEventListener.java

## uilogic.IGridPositionable Interface Reference

### Public Member Functions

[GridPosition](#AAAAAAAAVZ) [getGridPosition](#AAAAAAAAWM) ()

[GridPosition](#AAAAAAAAVZ) [setGridPosition](#AAAAAAAAWN) ([GridPosition](#AAAAAAAAVZ) newPosition) throws ArgumentNullException

### Detailed Description

This interface represents an object that can be positioned on a grid. It contains methods for getting and setting the grid position of the object.

### Member Function Documentation

#### [GridPosition](#AAAAAAAAVZ) uilogic.IGridPositionable.getGridPosition ()

Gets the grid position of the object.

##### Returns

The grid position of the object.

#### [GridPosition](#AAAAAAAAVZ) uilogic.IGridPositionable.setGridPosition ([GridPosition](#AAAAAAAAVZ) *newPosition*) throws ArgumentNullException

Sets the grid position of the object.

##### Parameters

|  |  |
| --- | --- |
| *newPosition* | The new grid position of the object. |

##### Returns

The new grid position of the object.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the new grid position is null. |

#### The documentation for this interface was generated from the following file:

uilogic/IGridPositionable.java

## game.behaviour.entities.IInteractiveEntity Interface Reference

### Public Member Functions

[IInteractiveEntity](#AAAAAAAAGR) [applyStats](#AAAAAAAAGS) ()

String [getInstanceID](#AAAAAAAAGT) ()

String [getAssetID](#AAAAAAAAGU) ()

int **getCurrentHealth** ()

double **getCurrentMovement** ()

Entity **getEntity** ()

[IInteractiveEntity](#AAAAAAAAGR) [setCurrentHealth](#AAAAAAAAGY) (int amount) throws InvalidArgumentException

boolean [move](#AAAAAAAAGZ) (double distance)

void [resetMovement](#AAAAAAAAHA) () throws Exception

boolean [takeDamage](#AAAAAAAAHB) (int damage) throws InvalidArgumentException

boolean [heal](#AAAAAAAAHC) (int amount) throws InvalidArgumentException

boolean [isDead](#AAAAAAAAHD) ()

void [die](#AAAAAAAAHE) ()

void [resurrect](#AAAAAAAAHF) ()

### Detailed Description

This interface represents an interactive entity in the game. It provides methods to get and set the current health, get the current movement, move the entity, take damage, heal, and check if the entity is dead.

### Member Function Documentation

#### [IInteractiveEntity](#AAAAAAAAGR) game.behaviour.entities.IInteractiveEntity.applyStats ()

Sets current stat values to the max.

##### Returns

This IInteractiveEntity, to allow for chaining.

Implemented in [game.behaviour.entities.enemy.Enemy](#AAAAAAAAHG), and [game.behaviour.entities.player.Player](#AAAAAAAAHH).

#### void game.behaviour.entities.IInteractiveEntity.die ()

Makes the entity die.

Implemented in [game.behaviour.entities.enemy.Enemy](#AAAAAAAAHI), and [game.behaviour.entities.player.Player](#AAAAAAAAHJ).

#### String game.behaviour.entities.IInteractiveEntity.getAssetID ()

Gets the asset ID of the entity.

##### Returns

The asset ID of the entity.

Implemented in [game.behaviour.entities.enemy.Enemy](#AAAAAAAAHK), and [game.behaviour.entities.player.Player](#AAAAAAAAHL).

#### String game.behaviour.entities.IInteractiveEntity.getInstanceID ()

Gets the instance ID of the entity.

##### Returns

The instance ID of the entity.

Implemented in [game.behaviour.entities.enemy.Enemy](#AAAAAAAAHM), and [game.behaviour.entities.player.Player](#AAAAAAAAHN).

#### boolean game.behaviour.entities.IInteractiveEntity.heal (int *amount*) throws InvalidArgumentException

Heals the entity by a certain amount.

##### Parameters

|  |  |
| --- | --- |
| *amount* | The amount to heal the entity. |

##### Returns

True if the entity was able to be healed, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the amount is less than 0. |

Implemented in [game.behaviour.entities.enemy.Enemy](#AAAAAAAAHO), and [game.behaviour.entities.player.Player](#AAAAAAAAHP).

#### boolean game.behaviour.entities.IInteractiveEntity.isDead ()

Checks if the entity is dead.

##### Returns

True if the entity is dead, false otherwise.

Implemented in [game.behaviour.entities.enemy.Enemy](#AAAAAAAAHQ), and [game.behaviour.entities.player.Player](#AAAAAAAAHR).

#### boolean game.behaviour.entities.IInteractiveEntity.move (double *distance*)

Moves the entity a certain distance.

##### Parameters

|  |  |
| --- | --- |
| *distance* | The distance to move the entity. |

##### Returns

True if the entity was able to move the specified distance, false otherwise.

Implemented in [game.behaviour.entities.enemy.Enemy](#AAAAAAAAHS), and [game.behaviour.entities.player.Player](#AAAAAAAAHT).

#### void game.behaviour.entities.IInteractiveEntity.resetMovement () throws Exception

Resets the movement of the entity.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the reset. |

Implemented in [game.behaviour.entities.enemy.Enemy](#AAAAAAAAHU), and [game.behaviour.entities.player.Player](#AAAAAAAAHV).

#### void game.behaviour.entities.IInteractiveEntity.resurrect ()

Resurrects the entity.

Implemented in [game.behaviour.entities.enemy.Enemy](#AAAAAAAAHW), and [game.behaviour.entities.player.Player](#AAAAAAAAHX).

#### [IInteractiveEntity](#AAAAAAAAGR) game.behaviour.entities.IInteractiveEntity.setCurrentHealth (int *amount*) throws InvalidArgumentException

Sets the current health of the entity.

##### Parameters

|  |  |
| --- | --- |
| *amount* | The amount to set the current health to. |

##### Returns

This IInteractiveEntity, to allow for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the amount is less than 0. |

Implemented in [game.behaviour.entities.player.Player](#AAAAAAAAHY), and [game.behaviour.entities.enemy.Enemy](#AAAAAAAAHZ).

#### boolean game.behaviour.entities.IInteractiveEntity.takeDamage (int *damage*) throws InvalidArgumentException

Makes the entity take a certain amount of damage.

##### Parameters

|  |  |
| --- | --- |
| *damage* | The amount of damage for the entity to take. |

##### Returns

True if the entity was able to take the damage, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the damage is less than 0. |

Implemented in [game.behaviour.entities.enemy.Enemy](#AAAAAAAAIA), and [game.behaviour.entities.player.Player](#AAAAAAAAIB).

#### The documentation for this interface was generated from the following file:

game/behaviour/entities/IInteractiveEntity.java

## ui.elements.ImageComponent Class Reference

### Public Member Functions

[ImageComponent](#AAAAAAAASG) (int width, int height)

[ImageComponent](#AAAAAAAASH) (int width, int height, String filePath) throws ArgumentNullException, IOException

[ImageComponent](#AAAAAAAASF) [setImage](#AAAAAAAASJ) (String filePath) throws ArgumentNullException, IOException

void [refresh](#AAAAAAAASI) ()

### Protected Attributes

GridDimension **preferredSize**

### Detailed Description

This class represents an image component in the UI. It displays an image with the specified width and height. It extends the JLabel class and is used to display an image.

The class contains the following field:

preferredSize: The preferred size of the image component.

### Constructor & Destructor Documentation

#### ui.elements.ImageComponent.ImageComponent (int *width*, int *height*)

Constructor for the ImageComponent class. Initializes the image component with the specified width and height.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the image component. |
| *height* | The height of the image component. |

#### ui.elements.ImageComponent.ImageComponent (int *width*, int *height*, String *filePath*) throws ArgumentNullException, IOException

Constructor for the ImageComponent class. Initializes the image component with the specified width, height, and file path.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the image component. |
| *height* | The height of the image component. |
| *filePath* | The file path of the image. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the file path is null. |
| *IOException* | if the file does not exist or is a directory. |

### Member Function Documentation

#### void ui.elements.ImageComponent.refresh ()

Refreshes the image component. Revalidates and repaints the image component.

#### [ImageComponent](#AAAAAAAASF) ui.elements.ImageComponent.setImage (String *filePath*) throws ArgumentNullException, IOException

Sets the image of the image component.

##### Parameters

|  |  |
| --- | --- |
| *filePath* | The file path of the image. |

##### Returns

The image component itself, for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the file path is null. |
| *IOException* | if the file does not exist or is a directory. |

Reimplemented in [ui.elements.GridEntityComponent](#AAAAAAAASC).

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

ui/elements/ImageComponent.java

## uilogic.InteractButtonHandler Class Reference

### Protected Member Functions

void [initActions](#AAAAAAAAWP) ()

#### Protected Member Functions inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

[MultipleButtonHandler](#AAAAAAAAWR) ()

### Additional Inherited Members

#### Public Member Functions inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

void [actionPerformed](#AAAAAAAAWS) (ActionEvent e)

#### Protected Attributes inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

HashMap< String, GenericDelegate > **actions**

### Detailed Description

This class handles the actions of interaction buttons in the UI. Used for the InteractButtonPanel. It extends the MultipleButtonHandler class and overrides the initActions method to initialize the actions of the interaction buttons.

The class contains the following methods:

initActions: Initializes the actions of the interaction buttons.

### Member Function Documentation

#### void uilogic.InteractButtonHandler.initActions ()[protected]

Initializes the actions of the interaction buttons. The actions include moving the player, ending the player's turn, attacking with the player, and picking up with the player. Each action is associated with a key in the actions HashMap of the MultipleButtonHandler class.

Reimplemented from [uilogic.MultipleButtonHandler](#AAAAAAAAWT).

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

uilogic/InteractButtonHandler.java

## ui.elements.InteractButtonPanel Class Reference

### Public Member Functions

[InteractButtonPanel](#AAAAAAAASU) (int width, int height, ActionListener listener)

void [toggleButtons](#AAAAAAAASV) (boolean enabled)

### Detailed Description

This class represents an interact button panel in the game. Buttons are used to move, attack, pick up, and end turn. It extends the JPanel class and is used to display a set of interaction buttons.

The class contains the following field:

buttons: The buttons of the interact button panel.

### Constructor & Destructor Documentation

#### ui.elements.InteractButtonPanel.InteractButtonPanel (int *width*, int *height*, ActionListener *listener*)

Constructor for the InteractButtonPanel class. Initializes the interact button panel with the specified width, height, and action listener.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the interact button panel. |
| *height* | The height of the interact button panel. |
| *listener* | The action listener of the interact button panel. |

### Member Function Documentation

#### void ui.elements.InteractButtonPanel.toggleButtons (boolean *enabled*)

Toggles the enabled state of the buttons in the interact button panel. If enabled is true, it enables the buttons; otherwise, it disables them. After toggling, it revalidates and repaints the panel.

##### Parameters

|  |  |
| --- | --- |
| *enabled* | A boolean that determines whether to enable the buttons. |

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

ui/elements/InteractButtonPanel.java

## uilogic.InteractiveGridHandler Class Reference

### Public Member Functions

[InteractiveGridHandler](#AAAAAAAAWV) (InteractiveGridPanel panel, boolean highlightButton)

InteractiveGridPanel **getPlayField** ()

[GridButtonHandler](#AAAAAAAAUZ) **getGridButtonHandler** ()

[GridPosition](#AAAAAAAAVZ) **getSelectedTile** ()

void [setPanel](#AAAAAAAAWZ) (InteractiveGridPanel panel) throws ArgumentNullException

void [selectTile](#AAAAAAAAXA) (Object o)

void [placeEntity](#AAAAAAAAXB) (String id, [GridPosition](#AAAAAAAAVZ) position, String imagePath) throws ArgumentNullException, InvalidArgumentException, ComponentAlreadyAtPositionException, IOException

void [removeEntity](#AAAAAAAAXC) (String id) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException

void [replaceEntity](#AAAAAAAAXD) (String id, [GridPosition](#AAAAAAAAVZ) newPosition) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException, ComponentAlreadyAtPositionException

[GridPosition](#AAAAAAAAVZ) [getEntityPositionByID](#AAAAAAAAXE) (String id) throws ArgumentNullException, ElementNotFoundException

String [getEntityIDByPosition](#AAAAAAAAXF) ([GridPosition](#AAAAAAAAVZ) position) throws ElementNotFoundException, ArgumentNullException

### Protected Attributes

InteractiveGridPanel **panel**

[GridButtonHandler](#AAAAAAAAUZ) **gridButtonHandler**

[GridPosition](#AAAAAAAAVZ) **selectedTile**

### Detailed Description

This class handles the interactive grid in the UI. It is used for placing, removing and handling entities on the grid. It contains an InteractiveGridPanel, a GridButtonHandler, and a GridPosition for the selected tile.

The class contains the following fields:

panel: The InteractiveGridPanel of the interactive grid.

gridButtonHandler: The GridButtonHandler for handling grid button actions.

selectedTile: The GridPosition of the selected tile.

### Constructor & Destructor Documentation

#### uilogic.InteractiveGridHandler.InteractiveGridHandler (InteractiveGridPanel *panel*, boolean *highlightButton*)

Constructor for the InteractiveGridHandler class. Initializes the InteractiveGridHandler with an InteractiveGridPanel and a boolean for whether to highlight the selected tile.

##### Parameters

|  |  |
| --- | --- |
| *panel* | The InteractiveGridPanel of the interactive grid. |
| *highlightButton* | The boolean for whether to highlight the selected tile. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the InteractiveGridPanel is null. |

### Member Function Documentation

#### String uilogic.InteractiveGridHandler.getEntityIDByPosition ([GridPosition](#AAAAAAAAVZ) *position*) throws ElementNotFoundException, ArgumentNullException

Gets the ID of an entity by its grid position.

##### Parameters

|  |  |
| --- | --- |
| *position* | The grid position of the entity. |

##### Returns

The ID of the entity.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the grid position is null. |
| *ElementNotFoundException* | if the entity at the specified grid position is not found. |

#### [GridPosition](#AAAAAAAAVZ) uilogic.InteractiveGridHandler.getEntityPositionByID (String *id*) throws ArgumentNullException, ElementNotFoundException

Gets the grid position of an entity by its ID.

##### Parameters

|  |  |
| --- | --- |
| *id* | The ID of the entity. |

##### Returns

The grid position of the entity.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the ID is null. |
| *ElementNotFoundException* | if the entity with the specified ID is not found. |

#### void uilogic.InteractiveGridHandler.placeEntity (String *id*, [GridPosition](#AAAAAAAAVZ) *position*, String *imagePath*) throws ArgumentNullException, InvalidArgumentException, ComponentAlreadyAtPositionException, IOException

Places an entity on the interactive grid.

##### Parameters

|  |  |
| --- | --- |
| *id* | The ID of the entity. |
| *position* | The grid position of the entity. |
| *imagePath* | The image path of the entity. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the ID, grid position, or image path is null. |
| *InvalidArgumentException* | if the ID is empty, or the grid position or image path is invalid. |
| *ComponentAlreadyAtPositionException* | if there is already an entity at the grid position. |
| *IOException* | if there is an error reading the image. |

#### void uilogic.InteractiveGridHandler.removeEntity (String *id*) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException

Removes an entity from the interactive grid.

##### Parameters

|  |  |
| --- | --- |
| *id* | The ID of the entity. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the ID is null. |
| *ElementNotFoundException* | if the entity with the specified ID is not found. |
| *InvalidArgumentException* | if the ID is empty. |

#### void uilogic.InteractiveGridHandler.replaceEntity (String *id*, [GridPosition](#AAAAAAAAVZ) *newPosition*) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException, ComponentAlreadyAtPositionException

Replaces an entity on the interactive grid with a new grid position.

##### Parameters

|  |  |
| --- | --- |
| *id* | The ID of the entity. |
| *newPosition* | The new grid position of the entity. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the ID or new grid position is null. |
| *ElementNotFoundException* | if the entity with the specified ID is not found. |
| *InvalidArgumentException* | if the ID is empty, or the new grid position is invalid. |
| *ComponentAlreadyAtPositionException* | if there is already an entity at the new grid position. |

#### void uilogic.InteractiveGridHandler.selectTile (Object *o*)

Selects a tile on the interactive grid.

##### Parameters

|  |  |
| --- | --- |
| *o* | The object representing the grid position of the tile to select. |

Reimplemented in [uilogic.PlayFieldHandler](#AAAAAAAAXJ).

#### void uilogic.InteractiveGridHandler.setPanel (InteractiveGridPanel *panel*) throws ArgumentNullException

Sets the InteractiveGridPanel of the interactive grid.

##### Parameters

|  |  |
| --- | --- |
| *panel* | The new InteractiveGridPanel of the interactive grid. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the new InteractiveGridPanel is null. |

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

uilogic/InteractiveGridHandler.java

## ui.elements.InteractiveGridPanel Class Reference

### Public Member Functions

[InteractiveGridPanel](#AAAAAAAASX) (int width, int height) throws Exception

GridDimension **getPreferredSize** ()

GridDimension **getComponentSize** ()

[InteractiveGridPanel](#AAAAAAAASW) [setMapLayout](#AAAAAAAATA) (MapLayoutData data, ActionListener buttonHandler, boolean force) throws Exception

[InteractiveGridPanel](#AAAAAAAASW) [setMapLayout](#AAAAAAAATB) (MapLayoutData data, ActionListener buttonHandler, boolean force, GridDimension componentSize) throws Exception

[GridEntityComponent](#AAAAAAAARZ) [addEntity](#AAAAAAAATC) ([GridEntityComponent](#AAAAAAAARZ) entity) throws ArgumentNullException, InvalidArgumentException, ComponentAlreadyAtPositionException

[GridEntityComponent](#AAAAAAAARZ) [getEntity](#AAAAAAAATD) (String id) throws ArgumentNullException, ElementNotFoundException

[GridEntityComponent](#AAAAAAAARZ) [removeEntity](#AAAAAAAATE) (String id, boolean removeFromList) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException

[GridEntityComponent](#AAAAAAAARZ) [removeEntity](#AAAAAAAATF) ([GridEntityComponent](#AAAAAAAARZ) entity, boolean removeFromList) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException

[GridEntityComponent](#AAAAAAAARZ) [replaceEntity](#AAAAAAAATG) (String id, GridPosition newPosition) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException, ComponentAlreadyAtPositionException

[GridEntityComponent](#AAAAAAAARZ) [replaceEntity](#AAAAAAAATH) ([GridEntityComponent](#AAAAAAAARZ) entity, GridPosition newPosition) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException, ComponentAlreadyAtPositionException

String [getEntityByPosition](#AAAAAAAATI) (GridPosition position) throws ArgumentNullException, ElementNotFoundException

### Detailed Description

This class represents an interactive grid panel in the game. It has 3 layers: background, entity, and button. Background is the lowest layer displaying the background, entity is the middle layer displaying entities, and button is the top layer with GridButtons. It extends the JPanel class and is used to display the game area.

The class contains the following fields:

layeredPane: The inner layout of the interactive grid panel.

background: The background of the interactive grid panel.

entityPanel: The entity panel of the interactive grid panel.

buttonPanel: The button panel of the interactive grid panel.

entityHandler: The entity handler of the interactive grid panel.

preferredSize: The preferred size of the interactive grid panel.

componentSize: The component size of the interactive grid panel.

### Constructor & Destructor Documentation

#### ui.elements.InteractiveGridPanel.InteractiveGridPanel (int *width*, int *height*) throws Exception

Constructor for the InteractiveGridPanel class. Initializes the interactive grid panel with the specified width and height.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the interactive grid panel. |
| *height* | The height of the interactive grid panel. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if the initialization of the interactive grid panel throws an Exception. |

### Member Function Documentation

#### [GridEntityComponent](#AAAAAAAARZ) ui.elements.InteractiveGridPanel.addEntity ([GridEntityComponent](#AAAAAAAARZ) *entity*) throws ArgumentNullException, InvalidArgumentException, ComponentAlreadyAtPositionException

Adds an entity to the interactive grid panel.

##### Parameters

|  |  |
| --- | --- |
| *entity* | The entity to add. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the entity is null. |
| *ComponentAlreadyAtPositionException* | if there is already a component at the position of the entity. |

#### [GridEntityComponent](#AAAAAAAARZ) ui.elements.InteractiveGridPanel.getEntity (String *id*) throws ArgumentNullException, ElementNotFoundException

Gets an entity from the interactive grid panel by its ID.

##### Parameters

|  |  |
| --- | --- |
| *id* | The ID of the entity. |

##### Returns

The entity with the specified ID.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the ID is null. |
| *ElementNotFoundException* | if there is no entity with the specified ID. |

#### String ui.elements.InteractiveGridPanel.getEntityByPosition (GridPosition *position*) throws ArgumentNullException, ElementNotFoundException

Gets the ID of an entity from the interactive grid panel by its position.

##### Parameters

|  |  |
| --- | --- |
| *position* | The position of the entity. |

##### Returns

The ID of the entity at the specified position.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the position is null. |
| *ElementNotFoundException* | if there is no entity at the specified position. |

#### [GridEntityComponent](#AAAAAAAARZ) ui.elements.InteractiveGridPanel.removeEntity ([GridEntityComponent](#AAAAAAAARZ) *entity*, boolean *removeFromList*) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException

Removes an entity from the interactive grid panel.

##### Parameters

|  |  |
| --- | --- |
| *entity* | The entity to remove. |
| *removeFromList* | A boolean that determines whether to remove the entity from the entity handler. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the entity is null. |

#### [GridEntityComponent](#AAAAAAAARZ) ui.elements.InteractiveGridPanel.removeEntity (String *id*, boolean *removeFromList*) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException

Removes an entity from the interactive grid panel.

##### Parameters

|  |  |
| --- | --- |
| *id* | The ID of the entity to remove. |
| *removeFromList* | A boolean that determines whether to remove the entity from the entity handler. |

##### Returns

The removed entity.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the entity is null. |
| *ElementNotFoundException* | if there is no entity with the specified ID. |
| *InvalidArgumentException* | if the entity is invalid. |

#### [GridEntityComponent](#AAAAAAAARZ) ui.elements.InteractiveGridPanel.replaceEntity ([GridEntityComponent](#AAAAAAAARZ) *entity*, GridPosition *newPosition*) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException, ComponentAlreadyAtPositionException

Replaces an entity in the interactive grid panel with a new position.

##### Parameters

|  |  |
| --- | --- |
| *entity* | The entity to replace. |
| *newPosition* | The new position of the entity. |

##### Returns

The replaced entity.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the entity or the new position is null. |
| *ElementNotFoundException* | if there is no entity with the specified ID. |
| *InvalidArgumentException* | if the new position is invalid. |
| *ComponentAlreadyAtPositionException* | if there is already a component at the new position. |

#### [GridEntityComponent](#AAAAAAAARZ) ui.elements.InteractiveGridPanel.replaceEntity (String *id*, GridPosition *newPosition*) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException, ComponentAlreadyAtPositionException

Replaces an entity in the interactive grid panel with a new position.

##### Parameters

|  |  |
| --- | --- |
| *id* | The ID of the entity to replace. |
| *newPosition* | The new position of the entity. |

##### Returns

The replaced entity.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the ID or the new position is null. |
| *ElementNotFoundException* | if there is no entity with the specified ID. |
| *InvalidArgumentException* | if the new position is invalid. |
| *ComponentAlreadyAtPositionException* | if there is already a component at the new position. |

#### [InteractiveGridPanel](#AAAAAAAASW) ui.elements.InteractiveGridPanel.setMapLayout (MapLayoutData *data*, ActionListener *buttonHandler*, boolean *force*) throws Exception

Sets the map layout of the interactive grid panel with auto component size. If the entity handler is not empty and force is false, it throws a PlayfieldNotEmptyException; otherwise, it calculates the component size and calls the overloaded setMapLayout method.

##### Parameters

|  |  |
| --- | --- |
| *data* | The new map layout data. |
| *buttonHandler* | The new button handler. |
| *force* | A boolean that determines whether to force the setting of the map layout. |

##### Returns

The interactive grid panel itself, for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if the setting of the map layout throws an Exception. |

#### [InteractiveGridPanel](#AAAAAAAASW) ui.elements.InteractiveGridPanel.setMapLayout (MapLayoutData *data*, ActionListener *buttonHandler*, boolean *force*, GridDimension *componentSize*) throws Exception

Sets the map layout of the interactive grid panel. If the entity handler is not empty and force is false, it throws a PlayfieldNotEmptyException; otherwise, it sets the component size, clears the entity handler, and initializes the entity panel and button panel.

##### Parameters

|  |  |
| --- | --- |
| *data* | The new map layout data. |
| *buttonHandler* | The new button handler. |
| *force* | A boolean that determines whether to force the setting of the map layout. |
| *componentSize* | The new component size. |

##### Returns

The interactive grid panel itself, for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if the setting of the map layout throws an Exception. |

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

ui/elements/InteractiveGridPanel.java

## exception.general.InvalidArgumentException Class Reference

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

exception/general/InvalidArgumentException.java

## exception.dice.InvalidDiceSideCountException Class Reference

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

exception/dice/InvalidDiceSideCountException.java

## exception.item.InvalidIDException Class Reference

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

exception/item/InvalidIDException.java

## game.behaviour.Inventory Class Reference

### Public Member Functions

[Inventory](#AAAAAAAADH) (boolean removeWhenRanOut)

int [size](#AAAAAAAADI) ()

void **setRemoveOnRanOut** (boolean [remove](#AAAAAAAADJ))

void [add](#AAAAAAAADL) (Item item) throws ArgumentNullException

Item [remove](#AAAAAAAADJ) (String id) throws ArgumentNullException

boolean [contains](#AAAAAAAADM) (String id) throws ArgumentNullException

double [calculateModifiers](#AAAAAAAADN) (ModifierType type) throws Exception

List< Consumable > **getConsumables** ()

List< Equipment > **getEquipments** ()

List< Item > **getSimpleItems** ()

List< Item > [getAllItems](#AAAAAAAADR) ()

void [run](#AAAAAAAADS) (EventArgument object, Event triggeredEvent) throws Exception

### Detailed Description

This class represents an Inventory in the game. It implements the IEventListener interface.

The class contains the following fields:

removeOnRanOut: A flag indicating whether to remove an item when it runs out.

equipments: A collection of equipment items.

consumables: A collection of consumable items.

simpleItems: A collection of simple items.

The class provides a constructor that initializes these fields and a method to get the size of the inventory.

### Constructor & Destructor Documentation

#### game.behaviour.Inventory.Inventory (boolean *removeWhenRanOut*)

Constructor for the Inventory class. Initializes the removeOnRanOut flag and the collections of items.

##### Parameters

|  |  |
| --- | --- |
| *removeWhenRanOut* | The value to set the removeOnRanOut flag to. |

### Member Function Documentation

#### void game.behaviour.Inventory.add (Item *item*) throws ArgumentNullException

Adds an item to the inventory.

##### Parameters

|  |  |
| --- | --- |
| *item* | The item to add. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the item is null. |

#### double game.behaviour.Inventory.calculateModifiers (ModifierType *type*) throws Exception

Calculates the modifiers of the inventory.

##### Parameters

|  |  |
| --- | --- |
| *bearerLevel* | The level of the bearer of the inventory. |

##### Returns

The modifiers of the inventory.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the calculation. |

#### boolean game.behaviour.Inventory.contains (String *id*) throws ArgumentNullException

Checks if the inventory contains a specific item.

##### Parameters

|  |  |
| --- | --- |
| *item* | The item to check for. |

##### Returns

True if the inventory contains the item, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the item is null. |

#### List< Item > game.behaviour.Inventory.getAllItems ()

Gets all items in the inventory.

##### Returns

A list of all items in the inventory.

#### Item game.behaviour.Inventory.remove (String *id*) throws ArgumentNullException

Removes an item from the inventory.

##### Parameters

|  |  |
| --- | --- |
| *item* | The item to remove. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the item is null. |

#### void game.behaviour.Inventory.run (EventArgument *object*, Event *triggeredEvent*) throws Exception

Runs an event when a consumable is out of charge.It removes the consumable from the inventory if the removeOnRanOut flag is set.

##### Parameters

|  |  |
| --- | --- |
| *object* | The argument for the event. |
| *triggeredEvent* | The event to run. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the event. |

#### int game.behaviour.Inventory.size ()

Gets the size of the inventory.

##### Returns

The size of the inventory. Size is the sum of the number of equipments, consumables, and simple items.

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

game/behaviour/Inventory.java

## game.behaviour.entities.InventoryEntity Class Reference

### Public Member Functions

[InventoryEntity](#AAAAAAAAID) (int health, int movement, int level, boolean removeConsumableFromInventoryWhenRanOut) throws InvalidArgumentException, ArgumentNullException

void [createInventory](#AAAAAAAAIE) (boolean removeWhenRanOut)

void [addToInventory](#AAAAAAAAIF) (Item item) throws ArgumentNullException

[Inventory](#AAAAAAAADG) **getInventory** ()

### Protected Attributes

transient [Inventory](#AAAAAAAADG) **inventory**

### Detailed Description

This class represents an InventoryEntity in the game. It extends the Entity class and includes an additional property: inventory.

The class contains the following fields:

inventory: The inventory of the entity.

The class provides a constructor that initializes these fields and methods to add to the inventory and get the inventory.

### Constructor & Destructor Documentation

#### game.behaviour.entities.InventoryEntity.InventoryEntity (int *health*, int *movement*, int *level*, boolean *removeConsumableFromInventoryWhenRanOut*) throws InvalidArgumentException, ArgumentNullException

Constructor for the InventoryEntity class. Initializes the health, movement, level, and inventory of the entity.

##### Parameters

|  |  |
| --- | --- |
| *health* | The health of the entity. |
| *movement* | The movement speed of the entity. |
| *level* | The level of the entity. |
| *removeConsumableFromInventoryWhenRanOut* | Whether to remove consumables from the inventory when they run out. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the health, movement, or level is invalid. |
| *ArgumentNullException* | if the inventory is null. |

### Member Function Documentation

#### void game.behaviour.entities.InventoryEntity.addToInventory (Item *item*) throws ArgumentNullException

Adds an item to the entity's inventory.

##### Parameters

|  |  |
| --- | --- |
| *item* | The item to add. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the item is null. |

#### void game.behaviour.entities.InventoryEntity.createInventory (boolean *removeWhenRanOut*)

Creates a new inventory for the entity.

##### Parameters

|  |  |
| --- | --- |
| *removeWhenRanOut* | Whether to remove items from the inventory when they run out. |

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

game/behaviour/entities/InventoryEntity.java

## game.behaviour.InventoryMarker< T > Class Template Reference

### Public Member Functions

[InventoryMarker](#AAAAAAAADU) (T t) throws ArgumentNullException, InvalidArgumentException

void **mark** (boolean mark)

boolean **isMarked** ()

T **getItem** ()

### Detailed Description

This class represents an InventoryMarker in the game. It is used to mark items in the inventory for removal. It is a generic class that can hold an item of type T.

The class contains the following fields:

item: The item of type T.

markedForRemoval: A flag indicating whether the item is marked for removal.

The class provides a constructor that initializes these fields and methods to get the item, mark it for removal, and check if it is marked.

### Constructor & Destructor Documentation

#### [game.behaviour.InventoryMarker](#AAAAAAAADT)< T >.[InventoryMarker](#AAAAAAAADT) (T *t*) throws ArgumentNullException, InvalidArgumentException

Constructor for the InventoryMarker class. Initializes the item and sets the markedForRemoval flag to false.

##### Parameters

|  |  |
| --- | --- |
| *t* | The item to initialize. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the item is null. |
| *InvalidArgumentException* | if the item is not of type Consumable or Equipment. |

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

game/behaviour/InventoryMarker.java

## ui.elements.InventoryPanel Class Reference

### Public Member Functions

[InventoryPanel](#AAAAAAAATJ) (int rows, int columns, ActionListener buttonHandler) throws Exception

[InteractiveGridPanel](#AAAAAAAASW) **getGrid** ()

int **getRowCount** ()

int **getColumnCount** ()

### Static Public Attributes

static int **COMPONENT\_WIDTH** = 48

static int **COMPONENT\_HEIGHT** = 48

static int **PANEL\_WIDTH** = 300

static int **PANEL\_HEIGHT** = 500

### Detailed Description

This class represents an inventory panel in the game. It uses the InteractiveGridPanel class to display the inventory. It extends the JPanel class and is used to display the inventory area.

The class contains the following fields:

COMPONENT\_WIDTH: The width of each component in the inventory panel.

COMPONENT\_HEIGHT: The height of each component in the inventory panel.

PANEL\_WIDTH: The width of the inventory panel.

PANEL\_HEIGHT: The height of the inventory panel.

rowCount: The number of rows in the inventory panel.

columnCount: The number of columns in the inventory panel.

grid: The interactive grid panel of the inventory panel.

### Constructor & Destructor Documentation

#### ui.elements.InventoryPanel.InventoryPanel (int *rows*, int *columns*, ActionListener *buttonHandler*) throws Exception

Constructor for the InventoryPanel class. Initializes the inventory panel with the specified number of rows and columns and the specified button handler.

##### Parameters

|  |  |
| --- | --- |
| *rows* | The number of rows in the inventory panel. |
| *columns* | The number of columns in the inventory panel. |
| *buttonHandler* | The button handler of the inventory panel. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if the initialization of the inventory panel throws an Exception. |

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

ui/elements/InventoryPanel.java

## game.behaviour.entities.items.Item Class Reference

### Public Member Functions

[Item](#AAAAAAAAJL) ()

ItemType **getItemType** ()

String **getDisplayInfo** ()

String **getStatistics** (int bearerLevel)

### Protected Attributes

ItemType **itemType**

### Detailed Description

This class represents an Item in the game. It extends the Identity class and implements the IDisplayable interface.

The class contains the following field:

itemType: The type of the item.

The class provides a constructor that initializes the itemType of the item and methods to get the itemType, display info, and statistics.

### Constructor & Destructor Documentation

#### game.behaviour.entities.items.Item.Item ()

Constructor for the Item class. Initializes the itemType of the item to SIMPLE.

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

game/behaviour/entities/items/Item.java

## file.elements.ItemMapData Class Reference

### Public Attributes

String **itemID**

GridPosition **position**

### Detailed Description

This class represents the data of an item on the map. It includes the item's ID and its position on the grid.

The class contains the following fields:

itemID: The ID of the item.

position: The position of the item on the grid.

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

file/elements/ItemMapData.java

## exception.entity.ItemNotInInventoryException Class Reference

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

exception/entity/ItemNotInInventoryException.java

## file.elements.ItemSave Class Reference

### Public Attributes

Item **item**

String **iconFilePath**

### Detailed Description

This class represents the saved data of an item. It includes the item's object and the path to its icon file.

The class contains the following fields:

item: The item object.

iconFilePath: The path to the item's icon file.

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

file/elements/ItemSave.java

## game.global.storage.ItemStorage Class Reference

### Static Public Member Functions

static [ItemStorage](#AAAAAAAAOG) **getInstance** ()

### Additional Inherited Members

#### Public Member Functions inherited from [game.global.storage.Storage< Item >](#AAAAAAAANZ)

void **add** (String id, T item) throws ArgumentNullException

void **remove** (String id) throws ArgumentNullException

T **get** (String id) throws ArgumentNullException

boolean **contains** (String id) throws ArgumentNullException

void **clear** ()

Set< Entry< String, T > > **entrySet** ()

ArrayList< T > [getAllItems](#AAAAAAAAOA) ()

ArrayList< String > [getAllKeys](#AAAAAAAAOB) ()

#### Protected Attributes inherited from [game.global.storage.Storage< Item >](#AAAAAAAANZ)

HashMap< String, T > **storage**

### Detailed Description

This class represents an ItemStorage in the game. It stores all the items in the game so that they can be reused. The key is the ID of the item. It extends the Storage class with Item type.

The class contains the following fields:

instance: The singleton instance of the ItemStorage class.

The class provides a private constructor and a method to get the singleton instance.

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

game/global/storage/ItemStorage.java

## game.enums.ItemType Enum Reference

### Public Attributes

**CONSUMABLE**

**EQUIPMENT**

**SIMPLE**

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

game/enums/ItemType.java

## ui.elements.LabelPanel Class Reference

### Public Member Functions

**LabelPanel** (boolean border)

[LabelPanel](#AAAAAAAATR) [setLabelText](#AAAAAAAATT) (String text)

[LabelPanel](#AAAAAAAATR) [setLabelText](#AAAAAAAATU) (String text, Font font)

### Detailed Description

This class represents a label panel in the UI. Its a panel with a centered label. It extends the JPanel class and is used to display a label with optional border.

The class contains the following field:

text: The JLabel that is displayed in the panel.

The class provides methods to set the text and font of the label, and to format the text.

### Member Function Documentation

#### [LabelPanel](#AAAAAAAATR) ui.elements.LabelPanel.setLabelText (String *text*)

Sets the text of the label in the panel.

##### Parameters

|  |  |
| --- | --- |
| *text* | The new text of the label. |

##### Returns

The label panel itself, for chaining.

#### [LabelPanel](#AAAAAAAATR) ui.elements.LabelPanel.setLabelText (String *text*, Font *font*)

Sets the text and font of the label in the panel.

##### Parameters

|  |  |
| --- | --- |
| *text* | The new text of the label. |
| *font* | The new font of the label. |

##### Returns

The label panel itself, for chaining.

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

ui/elements/LabelPanel.java

## Main Class Reference

### Static Public Member Functions

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

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

Main.java

## file.elements.MapLayoutData Class Reference

### Public Member Functions

[MapLayoutData](#AAAAAAAACF) (String id, int horizontal, int vertical, String file, GridPosition playerPosition) throws ArgumentNullException

int **getHorizontal** ()

int **getVertical** ()

String **getBackgroundFilePath** ()

ArrayList< [EnemyMapData](#AAAAAAAAAZ) > **getEnemies** ()

ArrayList< [ItemMapData](#AAAAAAAABY) > **getItems** ()

GridPosition **getPlayerPosition** ()

void **setBackgroundFilePath** (String path) throws ArgumentNullException

void [addEnemy](#AAAAAAAACN) ([EnemyMapData](#AAAAAAAAAZ) enemy) throws ArgumentNullException

void [addItem](#AAAAAAAACO) ([ItemMapData](#AAAAAAAABY) item) throws ArgumentNullException

### Detailed Description

This class represents the layout data for a map in the game. It extends the Identity class.

The class contains the following fields:

horizontal: The number of tiles horizontally.

vertical: The number of tiles vertically.

backgroundFilePath: The file path to the background image for the map.

enemies: A list of enemy data for the map.

items: A list of item data for the map.

playerStartPosition: The starting position of the player on the map.

The class provides a constructor that initializes these fields and methods to get and set these fields.

### Constructor & Destructor Documentation

#### file.elements.MapLayoutData.MapLayoutData (String *id*, int *horizontal*, int *vertical*, String *file*, GridPosition *playerPosition*) throws ArgumentNullException

Constructor for the MapLayoutData class. Initializes the id, horizontal size, vertical size, background file path, and player start position.

##### Parameters

|  |  |
| --- | --- |
| *id* | The id of the map layout data. |
| *horizontal* | The number of tiles horizontally. |
| *vertical* | The number of tiles vertically. |
| *file* | The file path to the background image for the map. |
| *playerPosition* | The starting position of the player on the map. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the id, file, or playerPosition is null. |

### Member Function Documentation

#### void file.elements.MapLayoutData.addEnemy ([EnemyMapData](#AAAAAAAAAZ) *enemy*) throws ArgumentNullException

Adds an enemy to the map.

##### Parameters

|  |  |
| --- | --- |
| *enemy* | The enemy data to add. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the enemy data is null. |

#### void file.elements.MapLayoutData.addItem ([ItemMapData](#AAAAAAAABY) *item*) throws ArgumentNullException

Adds an item to the map.

##### Parameters

|  |  |
| --- | --- |
| *item* | The item data to add. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the item data is null. |

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

file/elements/MapLayoutData.java

## game.global.storage.MapStorage Class Reference

### Static Public Member Functions

static [MapStorage](#AAAAAAAAOI) **getInstance** ()

### Additional Inherited Members

#### Public Member Functions inherited from [game.global.storage.Storage< String >](#AAAAAAAANZ)

void **add** (String id, T item) throws ArgumentNullException

void **remove** (String id) throws ArgumentNullException

T **get** (String id) throws ArgumentNullException

boolean **contains** (String id) throws ArgumentNullException

void **clear** ()

Set< Entry< String, T > > **entrySet** ()

ArrayList< T > [getAllItems](#AAAAAAAAOA) ()

ArrayList< String > [getAllKeys](#AAAAAAAAOB) ()

#### Protected Attributes inherited from [game.global.storage.Storage< String >](#AAAAAAAANZ)

HashMap< String, T > **storage**

### Detailed Description

This class represents a MapStorage in the game. It stores map name-ID pairs. The key is the name of the map. Only used to load new maps based on user input. It extends the Storage class with String type.

The class contains the following fields:

instance: The singleton instance of the MapStorage class.

The class provides a private constructor and a method to get the singleton instance.

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

game/global/storage/MapStorage.java

## game.utility.ModifiedEnemyData Class Reference

### Public Member Functions

**ModifiedEnemyData** (String id, GridPosition position, int health, boolean looted, boolean dead) throws ArgumentNullException, InvalidArgumentException

GridPosition **getPosition** ()

int **getHealth** ()

boolean **isLooted** ()

boolean **isDead** ()

void **setPosition** (GridPosition position) throws ArgumentNullException

void **setHealth** (int health) throws InvalidArgumentException

void **setLooted** (boolean looted)

void **setDead** (boolean dead)

#### Public Member Functions inherited from [game.utility.Identifiable](#AAAAAAAAPA)

final [Identifiable](#AAAAAAAAPA) [setID](#AAAAAAAAPB) (String newID) throws ArgumentNullException

final String **getID** ()

### Additional Inherited Members

#### Protected Attributes inherited from [game.utility.Identifiable](#AAAAAAAAPA)

String **id**

### Detailed Description

This class represents the modified data of an enemy in the game. Used to store and save the modified enemies. It extends the Identifiable class and contains methods to set and get the position, health, looted status, and death status of the enemy.

The class contains the following fields:

position: The position of the enemy on the grid.

currentHealth: The current health of the enemy.

looted: Whether the enemy has been looted.

dead: Whether the enemy is dead.

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

game/utility/ModifiedEnemyData.java

## game.global.storage.ModifiedEnemyStorage Class Reference

### Static Public Member Functions

static [ModifiedEnemyStorage](#AAAAAAAAOK) **getInstance** ()

### Additional Inherited Members

#### Public Member Functions inherited from [game.global.storage.Storage< ModifiedEnemyData >](#AAAAAAAANZ)

void **add** (String id, T item) throws ArgumentNullException

void **remove** (String id) throws ArgumentNullException

T **get** (String id) throws ArgumentNullException

boolean **contains** (String id) throws ArgumentNullException

void **clear** ()

Set< Entry< String, T > > **entrySet** ()

ArrayList< T > [getAllItems](#AAAAAAAAOA) ()

ArrayList< String > [getAllKeys](#AAAAAAAAOB) ()

#### Protected Attributes inherited from [game.global.storage.Storage< ModifiedEnemyData >](#AAAAAAAANZ)

HashMap< String, T > **storage**

### Detailed Description

This class represents a ModifiedEnemyStorage in the game. It stores the currently acitve enemies that have been modified by the player. The key is the ID of the enemy. It extends the Storage class with ModifiedEnemyData type.

The class contains the following fields:

instance: The singleton instance of the ModifiedEnemyStorage class.

The class provides a private constructor and a method to get the singleton instance.

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

game/global/storage/ModifiedEnemyStorage.java

## game.enums.ModifierType Enum Reference

### Public Attributes

**ATTACK**

**DAMAGE**

**ARMOR\_CLASS**

**MOVEMENT**

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

game/enums/ModifierType.java

## uilogic.MultipleButtonHandler Class Reference

### Public Member Functions

void [actionPerformed](#AAAAAAAAWS) (ActionEvent e)

### Protected Member Functions

[MultipleButtonHandler](#AAAAAAAAWR) ()

abstract void [initActions](#AAAAAAAAWT) ()

### Protected Attributes

HashMap< String, GenericDelegate > **actions**

### Detailed Description

This abstract class handles multiple buttons in the UI. It implements ActionListener and contains a HashMap of actions for the buttons. Each subclass must implement the initActions method to initialize the actions of the buttons.

The class contains the following fields:

actions: The HashMap of actions for the buttons.

### Constructor & Destructor Documentation

#### uilogic.MultipleButtonHandler.MultipleButtonHandler ()[protected]

Constructor for the MultipleButtonHandler class. Initializes the MultipleButtonHandler with a HashMap of actions and calls the initActions method to initialize the actions of the buttons.

### Member Function Documentation

#### void uilogic.MultipleButtonHandler.actionPerformed (ActionEvent *e*)

Handles the action performed by a button. Gets the action associated with the button from the actions HashMap and runs the action.

##### Parameters

|  |  |
| --- | --- |
| *e* | The ActionEvent object representing the action performed by the button. |

#### abstract void uilogic.MultipleButtonHandler.initActions ()[abstract], [protected]

Initializes the actions of the buttons. This method must be implemented by each subclass of the MultipleButtonHandler class.

Reimplemented in [uilogic.InteractButtonHandler](#AAAAAAAAWP), [uilogic.PlayerDeathFrameHandler](#AAAAAAAAXL), [uilogic.PlayFrameMenuBarHandler](#AAAAAAAAXM), and [uilogic.UtilityButtonHandler](#AAAAAAAAXN).

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

uilogic/MultipleButtonHandler.java

## exception.entity.NoWeaponEquippedException Class Reference

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

exception/entity/NoWeaponEquippedException.java

## game.behaviour.entities.player.Player Class Reference

### Public Member Functions

[Player](#AAAAAAAAKI) (String id, [PlayerEntity](#AAAAAAAAKJ) entity) throws ArgumentNullException

int **getXP** ()

int **getRequiredXP** ()

int [getCurrentHealth](#AAAAAAAAKM) ()

double [getCurrentMovement](#AAAAAAAAKN) ()

void [addEventListeners](#AAAAAAAAKO) (IEventListener playerDied, IEventListener playerLeveledUp) throws ArgumentNullException

String [getInstanceID](#AAAAAAAAHN) ()

String [getAssetID](#AAAAAAAAHL) ()

[PlayerEntity](#AAAAAAAAKJ) [getEntity](#AAAAAAAAKP) ()

[IInteractiveEntity](#AAAAAAAAGR) [applyStats](#AAAAAAAAHH) ()

void **setRequiredXP** (int amount) throws InvalidArgumentException

int [addXP](#AAAAAAAAKR) (int newXP)

void [levelUp](#AAAAAAAAKS) ()

void [addToInventory](#AAAAAAAAKT) (Item item) throws ArgumentNullException

boolean [attack](#AAAAAAAAKU) (int targetAC, double distance) throws Exception

int [damage](#AAAAAAAAKV) (double distance) throws Exception

int [getArmorClass](#AAAAAAAAKW) () throws Exception

void **equip** (Weapon weapon) throws ItemNotInInventoryException, ArgumentNullException

void **equip** (Armor armor) throws ItemNotInInventoryException, ArgumentNullException

void [resetMovement](#AAAAAAAAHV) () throws Exception

boolean [move](#AAAAAAAAHT) (double distance)

boolean [takeDamage](#AAAAAAAAIB) (int [damage](#AAAAAAAAKV)) throws InvalidArgumentException

boolean [heal](#AAAAAAAAHP) (int amount) throws InvalidArgumentException

void [die](#AAAAAAAAHJ) ()

void [resurrect](#AAAAAAAAHX) ()

[IInteractiveEntity](#AAAAAAAAGR) [setCurrentHealth](#AAAAAAAAHY) (int amount) throws InvalidArgumentException

boolean [isDead](#AAAAAAAAHR) ()

### Detailed Description

This class represents a Player in the game. It extends the Identity class and implements the IInteractiveEntity interface.

The class contains the following fields:

entity: The player entity.

xp: The experience points of the player.

requiredXP: The experience points required for the player to level up.

currentHealth: The current health of the player.

condition: The condition of the player.

currentMovement: The current movement speed of the player.

onPlayerDied: Event that is triggered when the player dies.

onPlayerLeveledUp: Event that is triggered when the player levels up.

The class provides a constructor that initializes the id and entity of the player.

### Constructor & Destructor Documentation

#### game.behaviour.entities.player.Player.Player (String *id*, [PlayerEntity](#AAAAAAAAKJ) *entity*) throws ArgumentNullException

Constructor for the Player class. Initializes the id and entity of the player.

##### Parameters

|  |  |
| --- | --- |
| *id* | The id of the player. |
| *entity* | The player entity. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the id or entity is null. |

### Member Function Documentation

#### void game.behaviour.entities.player.Player.addEventListeners (IEventListener *playerDied*, IEventListener *playerLeveledUp*) throws ArgumentNullException

Constructor for the Player class. Initializes the id and entity of the player.

##### Parameters

|  |  |
| --- | --- |
| *id* | The id of the player. |
| *entity* | The player entity. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the id or entity is null. |

#### void game.behaviour.entities.player.Player.addToInventory (Item *item*) throws ArgumentNullException

Adds an item to the player's inventory.

##### Parameters

|  |  |
| --- | --- |
| *item* | The item to add. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the item is null. |

#### int game.behaviour.entities.player.Player.addXP (int *newXP*)

Adds experience points to the player. If the player has enough experience points to level up, it levels up the player.

##### Parameters

|  |  |
| --- | --- |
| *xp* | The experience points to add. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the xp is less than 0. |

#### [IInteractiveEntity](#AAAAAAAAGR) game.behaviour.entities.player.Player.applyStats ()

Sets current stat values to the max.

##### Returns

This IInteractiveEntity, to allow for chaining.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGS).

#### boolean game.behaviour.entities.player.Player.attack (int *targetAC*, double *distance*) throws Exception

Makes the player attack a target. If the player has no weapon equipped, it throws a NoWeaponEquippedException. Calculates the final target armor class by subtracting the player's attack modifier from the target's armor class.

##### Parameters

|  |  |
| --- | --- |
| *targetAC* | The armor class of the target. |
| *distance* | The distance to the target. |

##### Returns

True if the attack is successful, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the attack. |

#### int game.behaviour.entities.player.Player.damage (double *distance*) throws Exception

Calculates the damage the player deals to a target. Adds the player's damage modifier to the damage dealt.

##### Parameters

|  |  |
| --- | --- |
| *distance* | The distance to the target. |

##### Returns

The damage dealt to the target.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the damage calculation. |

#### void game.behaviour.entities.player.Player.die ()

Makes the entity die.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHE).

#### int game.behaviour.entities.player.Player.getArmorClass () throws Exception

Calculates the armor class of the player. The armor class is calculated by adding the player's base armor class and any armor class modifiers from the player's inventory.

##### Returns

The armor class of the player.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the calculation. |

#### String game.behaviour.entities.player.Player.getAssetID ()

Gets the asset ID of the entity.

##### Returns

The asset ID of the entity.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGU).

#### int game.behaviour.entities.player.Player.getCurrentHealth ()

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGR).

#### double game.behaviour.entities.player.Player.getCurrentMovement ()

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGR).

#### [PlayerEntity](#AAAAAAAAKJ) game.behaviour.entities.player.Player.getEntity ()

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGR).

#### String game.behaviour.entities.player.Player.getInstanceID ()

Gets the instance ID of the entity.

##### Returns

The instance ID of the entity.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGT).

#### boolean game.behaviour.entities.player.Player.heal (int *amount*) throws InvalidArgumentException

Heals the entity by a certain amount.

##### Parameters

|  |  |
| --- | --- |
| *amount* | The amount to heal the entity. |

##### Returns

True if the entity was able to be healed, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the amount is less than 0. |

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHC).

#### boolean game.behaviour.entities.player.Player.isDead ()

Checks if the entity is dead.

##### Returns

True if the entity is dead, false otherwise.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHD).

#### void game.behaviour.entities.player.Player.levelUp ()

Levels up the player.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during leveling up. |

#### boolean game.behaviour.entities.player.Player.move (double *distance*)

Moves the entity a certain distance.

##### Parameters

|  |  |
| --- | --- |
| *distance* | The distance to move the entity. |

##### Returns

True if the entity was able to move the specified distance, false otherwise.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGZ).

#### void game.behaviour.entities.player.Player.resetMovement () throws Exception

Resets the movement of the entity.

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if an error occurs during the reset. |

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHA).

#### void game.behaviour.entities.player.Player.resurrect ()

Resurrects the entity.

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHF).

#### [IInteractiveEntity](#AAAAAAAAGR) game.behaviour.entities.player.Player.setCurrentHealth (int *amount*) throws InvalidArgumentException

Sets the current health of the entity.

##### Parameters

|  |  |
| --- | --- |
| *amount* | The amount to set the current health to. |

##### Returns

This IInteractiveEntity, to allow for chaining.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the amount is less than 0. |

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAGY).

#### boolean game.behaviour.entities.player.Player.takeDamage (int *damage*) throws InvalidArgumentException

Makes the entity take a certain amount of damage.

##### Parameters

|  |  |
| --- | --- |
| *damage* | The amount of damage for the entity to take. |

##### Returns

True if the entity was able to take the damage, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the damage is less than 0. |

Implements [game.behaviour.entities.IInteractiveEntity](#AAAAAAAAHB).

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

game/behaviour/entities/player/Player.java

## ui.elements.PlayerDeathFrame Class Reference

### Public Member Functions

[PlayerDeathFrame](#AAAAAAAATW) (Component relativeLocation, ActionListener listener) throws ArgumentNullException

### Detailed Description

This class represents a frame that is displayed when the player dies. It allows the player to load a save or quit the game. It extends the JFrame class.

The class contains the following methods:

PlayerDeathFrame: The constructor of the class.

initFrame: Initializes the frame.

setupPanel: Sets up the panel in the frame.

### Constructor & Destructor Documentation

#### ui.elements.PlayerDeathFrame.PlayerDeathFrame (Component *relativeLocation*, ActionListener *listener*) throws ArgumentNullException

Constructor for the PlayerDeathFrame class. Initializes the frame with the specified relative location and action listener.

##### Parameters

|  |  |
| --- | --- |
| *relativeLocation* | The component in relation to which the frame's location is determined. |
| *listener* | The action listener of the frame. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the listener is null. |

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

ui/elements/PlayerDeathFrame.java

## uilogic.PlayerDeathFrameHandler Class Reference

### Protected Member Functions

void [initActions](#AAAAAAAAXL) ()

#### Protected Member Functions inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

[MultipleButtonHandler](#AAAAAAAAWR) ()

### Additional Inherited Members

#### Public Member Functions inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

void [actionPerformed](#AAAAAAAAWS) (ActionEvent e)

#### Protected Attributes inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

HashMap< String, GenericDelegate > **actions**

### Detailed Description

This class handles the actions of the player death frame in the UI. It extends the MultipleButtonHandler class and overrides the initActions method to initialize the actions of the player death frame.

The class contains the following methods:

initActions: Initializes the actions of the player death frame.

### Member Function Documentation

#### void uilogic.PlayerDeathFrameHandler.initActions ()[protected]

Initializes the actions of the player death frame. The actions include loading a save and quitting the game. Each action is associated with a key in the actions HashMap of the MultipleButtonHandler class.

Reimplemented from [uilogic.MultipleButtonHandler](#AAAAAAAAWT).

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

uilogic/PlayerDeathFrameHandler.java

## game.behaviour.entities.player.PlayerEntity Class Reference

### Public Member Functions

[PlayerEntity](#AAAAAAAAKZ) (int health, int movement, int level) throws InvalidArgumentException, ArgumentNullException

#### Public Member Functions inherited from [game.behaviour.entities.InventoryEntity](#AAAAAAAAIC)

[InventoryEntity](#AAAAAAAAID) (int health, int movement, int level, boolean removeConsumableFromInventoryWhenRanOut) throws InvalidArgumentException, ArgumentNullException

void [createInventory](#AAAAAAAAIE) (boolean removeWhenRanOut)

void [addToInventory](#AAAAAAAAIF) (Item item) throws ArgumentNullException

[Inventory](#AAAAAAAADG) **getInventory** ()

### Additional Inherited Members

#### Protected Attributes inherited from [game.behaviour.entities.InventoryEntity](#AAAAAAAAIC)

transient [Inventory](#AAAAAAAADG) **inventory**

### Detailed Description

This class represents a PlayerEntity in the game. It extends the InventoryEntity class and includes an additional property: entityType.

The class contains the following fields:

entityType: The type of the entity, which is PLAYER for this class.

The class provides a constructor that initializes the health, movement, level, and entityType of the player entity.

### Constructor & Destructor Documentation

#### game.behaviour.entities.player.PlayerEntity.PlayerEntity (int *health*, int *movement*, int *level*) throws InvalidArgumentException, ArgumentNullException

Constructor for the PlayerEntity class. Initializes the health, movement, level, and entityType of the player entity.

##### Parameters

|  |  |
| --- | --- |
| *health* | The health of the player entity. |
| *movement* | The movement speed of the player entity. |
| *level* | The level of the player entity. |

##### Exceptions

|  |  |
| --- | --- |
| *InvalidArgumentException* | if the health, movement, or level is invalid. |
| *ArgumentNullException* | if the inventory is null. |

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

game/behaviour/entities/player/PlayerEntity.java

## file.elements.PlayerProgressSave Class Reference

### Public Attributes

boolean **modifiable**

String **currentMapID**

String **currentIconFile**

Player **player**

GridPosition **playerPosition**

String **playerArmorID**

String **playerWeaponID**

ArrayList< String > **inventory**

ArrayList< ModifiedEnemyData > **modifiedEnemies**

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

file/elements/PlayerProgressSave.java

## uilogic.PlayFieldHandler Class Reference

### Public Member Functions

[PlayFieldHandler](#AAAAAAAAXQ) (InteractiveGridPanel playField)

MapLayoutData [getCurrentMapLayoutData](#AAAAAAAAXR) ()

double [getSelectedTileDistance](#AAAAAAAAXS) ()

void [selectTile](#AAAAAAAAXJ) (Object o)

void [setCurrentMapLayout](#AAAAAAAAXT) (MapLayoutData data) throws Exception

double [getDistanceBetweenEntities](#AAAAAAAAXU) (String srcID, String dstID) throws ArgumentNullException, ElementNotFoundException

#### Public Member Functions inherited from [uilogic.InteractiveGridHandler](#AAAAAAAAWU)

[InteractiveGridHandler](#AAAAAAAAWV) (InteractiveGridPanel panel, boolean highlightButton)

InteractiveGridPanel **getPlayField** ()

[GridButtonHandler](#AAAAAAAAUZ) **getGridButtonHandler** ()

[GridPosition](#AAAAAAAAVZ) **getSelectedTile** ()

void [setPanel](#AAAAAAAAWZ) (InteractiveGridPanel panel) throws ArgumentNullException

void [placeEntity](#AAAAAAAAXB) (String id, [GridPosition](#AAAAAAAAVZ) position, String imagePath) throws ArgumentNullException, InvalidArgumentException, ComponentAlreadyAtPositionException, IOException

void [removeEntity](#AAAAAAAAXC) (String id) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException

void [replaceEntity](#AAAAAAAAXD) (String id, [GridPosition](#AAAAAAAAVZ) newPosition) throws ArgumentNullException, ElementNotFoundException, InvalidArgumentException, ComponentAlreadyAtPositionException

[GridPosition](#AAAAAAAAVZ) [getEntityPositionByID](#AAAAAAAAXE) (String id) throws ArgumentNullException, ElementNotFoundException

String [getEntityIDByPosition](#AAAAAAAAXF) ([GridPosition](#AAAAAAAAVZ) position) throws ElementNotFoundException, ArgumentNullException

### Additional Inherited Members

#### Protected Attributes inherited from [uilogic.InteractiveGridHandler](#AAAAAAAAWU)

InteractiveGridPanel **panel**

[GridButtonHandler](#AAAAAAAAUZ) **gridButtonHandler**

[GridPosition](#AAAAAAAAVZ) **selectedTile**

### Detailed Description

This class handles the play field in the UI. It extends the InteractiveGridHandler class and contains a MapLayoutData for the current map layout and a double for the distance of the selected tile.

The class contains the following fields:

selectedTileDistance: The distance of the selected tile.

currentMapLayoutData: The MapLayoutData for the current map layout.

### Constructor & Destructor Documentation

#### uilogic.PlayFieldHandler.PlayFieldHandler (InteractiveGridPanel *playField*)

Constructor for the PlayFieldHandler class. Initializes the PlayFieldHandler with an InteractiveGridPanel and sets the highlightButton to true.

##### Parameters

|  |  |
| --- | --- |
| *playField* | The InteractiveGridPanel of the play field. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the InteractiveGridPanel is null. |

### Member Function Documentation

#### MapLayoutData uilogic.PlayFieldHandler.getCurrentMapLayoutData ()

Gets the MapLayoutData for the current map layout.

##### Returns

The MapLayoutData for the current map layout.

#### double uilogic.PlayFieldHandler.getDistanceBetweenEntities (String *srcID*, String *dstID*) throws ArgumentNullException, ElementNotFoundException

Gets the distance between two entities by their IDs.

##### Parameters

|  |  |
| --- | --- |
| *srcID* | The ID of the source entity. |
| *dstID* | The ID of the destination entity. |

##### Returns

The distance between the source entity and the destination entity.

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the source ID or destination ID is null. |
| *ElementNotFoundException* | if the source entity or destination entity with the specified ID is not found. |

#### double uilogic.PlayFieldHandler.getSelectedTileDistance ()

Gets the distance of the selected tile.

##### Returns

The distance of the selected tile.

#### void uilogic.PlayFieldHandler.selectTile (Object *o*)

Selects a tile on the play field and calculates the distance of the selected tile from the player.

##### Parameters

|  |  |
| --- | --- |
| *o* | The object representing the grid position of the tile to select. |

Reimplemented from [uilogic.InteractiveGridHandler](#AAAAAAAAXA).

#### void uilogic.PlayFieldHandler.setCurrentMapLayout (MapLayoutData *data*) throws Exception

Sets the MapLayoutData for the current map layout and updates the map layout of the play field.

##### Parameters

|  |  |
| --- | --- |
| *data* | The new MapLayoutData for the current map layout. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if there is an error setting the map layout. |

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

uilogic/PlayFieldHandler.java

## exception.ui.PlayfieldNotEmptyException Class Reference

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

exception/ui/PlayfieldNotEmptyException.java

## ui.elements.PlayFrame Class Reference

### Public Member Functions

[PlayFrame](#AAAAAAAATY) (ActionListener menuBarListener, ActionListener utilityButtonListener, ActionListener interactButtonListener, WindowAdapter closeOperation) throws Exception

[InteractiveGridPanel](#AAAAAAAASW) **getPlayField** ()

void [modifyMapLayout](#AAAAAAAAUA) (MapLayoutData data, GridButtonHandler buttonHandler, boolean force) throws Exception

void [addToCombatLog](#AAAAAAAAUB) (String text)

void [clearCombatLog](#AAAAAAAAUC) ()

void [refresh](#AAAAAAAAUD) ()

void [togglePlayerControlls](#AAAAAAAAUE) (boolean enabled)

### Static Public Attributes

static int **PLAYFIELD\_WIDTH** = 1200

static int **PLAYFIELD\_HEIGHT** = 675

static int **UTILITYPANEL\_WIDTH** = 300

static int **UTILITYPANEL\_HEIGHT** = 844

static int **INTERACTPANEL\_WIDTH** = 1200

static int **INTERACTPANEL\_HEIGHT** = 169

static int **COMBATLOGPANEL\_WIDTH** = 300

static int **COMBATLOGPANEL\_HEIGHT** = 844

### Detailed Description

This class represents the main play frame of the game. It is the main interface of the game. It extends the JFrame class and contains various panels for different parts of the game interface.

The class contains the following fields:

panel: The main panel of the frame.

playfieldPanel: The panel for the playfield.

combatLogPanel: The panel for the combat log.

interactButtonPanel: The panel for the interaction buttons.

utilityButtonPanel: The panel for the utility buttons.

PLAYFIELD\_WIDTH, PLAYFIELD\_HEIGHT: The dimensions of the playfield panel.

UTILITYPANEL\_WIDTH, UTILITYPANEL\_HEIGHT: The dimensions of the utility panel.

INTERACTPANEL\_WIDTH, INTERACTPANEL\_HEIGHT: The dimensions of the interaction panel.

COMBATLOGPANEL\_WIDTH, COMBATLOGPANEL\_HEIGHT: The dimensions of the combat log panel.

### Constructor & Destructor Documentation

#### ui.elements.PlayFrame.PlayFrame (ActionListener *menuBarListener*, ActionListener *utilityButtonListener*, ActionListener *interactButtonListener*, WindowAdapter *closeOperation*) throws Exception

Constructor for the PlayFrame class. Initializes the play frame with the specified action listeners and window adapter.

##### Parameters

|  |  |
| --- | --- |
| *menuBarListener* | The action listener for the menu bar. |
| *utilityButtonListener* | The action listener for the utility buttons. |
| *interactButtonListener* | The action listener for the interaction buttons. |
| *closeOperation* | The window adapter for the close operation. |

##### Exceptions

|  |  |
| --- | --- |
| *Exception* | if the initialization of the play frame throws an Exception. |

### Member Function Documentation

#### void ui.elements.PlayFrame.addToCombatLog (String *text*)

Adds a text to the combat log panel.

##### Parameters

|  |  |
| --- | --- |
| *text* | The text to add. |

#### void ui.elements.PlayFrame.clearCombatLog ()

Clears the combat log panel.

#### void ui.elements.PlayFrame.modifyMapLayout (MapLayoutData *data*, GridButtonHandler *buttonHandler*, boolean *force*) throws Exception

Modifies the map layout of the playfield panel.

##### Parameters

|  |  |
| --- | --- |
| *data* | The new map layout data. |
| *buttonHandler* | The new grid button handler. |
| *force* | A boolean that determines whether to force the modification. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the data or the button handler is null. |
| *Exception* | if the modification of the map layout throws an Exception. |

#### void ui.elements.PlayFrame.refresh ()

Refreshes the frame. Revalidates and repaints the frame.

#### void ui.elements.PlayFrame.togglePlayerControlls (boolean *enabled*)

Toggles the player controls. Enables or disables the utility buttons and interaction buttons, and refreshes the frame.

##### Parameters

|  |  |
| --- | --- |
| *enabled* | A boolean that determines whether to enable or disable the player controls. |

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

ui/elements/PlayFrame.java

## uilogic.PlayFrameMenuBarHandler Class Reference

### Protected Member Functions

final void [initActions](#AAAAAAAAXM) ()

#### Protected Member Functions inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

[MultipleButtonHandler](#AAAAAAAAWR) ()

### Additional Inherited Members

#### Public Member Functions inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

void [actionPerformed](#AAAAAAAAWS) (ActionEvent e)

#### Protected Attributes inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

HashMap< String, GenericDelegate > **actions**

### Detailed Description

This class handles the actions of the menu bar in the play frame. It extends the MultipleButtonHandler class and overrides the initActions method to initialize the actions of the menu bar.

The class contains the following methods:

initActions: Initializes the actions of the menu bar.

### Member Function Documentation

#### final void uilogic.PlayFrameMenuBarHandler.initActions ()[protected]

Initializes the actions of the menu bar. The actions include quick saving the game, creating a new save game, loading a config file, and loading a game. Each action is associated with a key in the actions HashMap of the MultipleButtonHandler class.

Reimplemented from [uilogic.MultipleButtonHandler](#AAAAAAAAWT).

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

uilogic/PlayFrameMenuBarHandler.java

## game.global.SaveHandler Class Reference

### Public Member Functions

void **setCurrentSavePath** (String path) throws ArgumentNullException

void [setModifiable](#AAAAAAAANH) (boolean modifiable)

void [quickSave](#AAAAAAAANI) () throws CurrentSaveUnmodifiableException

void [save](#AAAAAAAANJ) (String filePath, boolean appendFileExtension) throws ArgumentNullException

### Detailed Description

This class handles the saving of the game. It contains methods to set the current save path, set whether the save is modifiable, quick save, and save.

The class contains the following fields:

currentSavePath: The path of the current save.

modifiable: Whether the current save can be overwritten.

### Member Function Documentation

#### void game.global.SaveHandler.quickSave () throws CurrentSaveUnmodifiableException

Quick saves the game. If the current save is not modifiable, it throws a CurrentSaveUnmodifiableException.

##### Exceptions

|  |  |
| --- | --- |
| *CurrentSaveUnmodifiableException* | if the current save is not modifiable. |

#### void game.global.SaveHandler.save (String *filePath*, boolean *appendFileExtension*) throws ArgumentNullException

Saves the game.

##### Parameters

|  |  |
| --- | --- |
| *filePath* | The path of the file to save to. |
| *appendFileExtension* | Whether to append the file extension to the file path. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the file path is null. |

#### void game.global.SaveHandler.setModifiable (boolean *modifiable*)

Sets whether the current save is modifiable.

##### Parameters

|  |  |
| --- | --- |
| *modifiable* | Whether the current save is modifiable. |

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

game/global/SaveHandler.java

## ui.elements.ScrollableTextPanel Class Reference

### Public Member Functions

[ScrollableTextPanel](#AAAAAAAAUO) (int width, int height)

void [addToText](#AAAAAAAAUP) (String text)

void [clearText](#AAAAAAAAUQ) ()

### Detailed Description

This class represents a scrollable text panel in the UI. It is a panel with a scrollable text area. It extends the JPanel class and contains a JTextArea for displaying text.

The class contains the following field:

textArea: The JTextArea that is displayed in the panel.

### Constructor & Destructor Documentation

#### ui.elements.ScrollableTextPanel.ScrollableTextPanel (int *width*, int *height*)

Constructor for the ScrollableTextPanel class. Initializes the panel and sets up the text area with the specified width and height.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the panel. |
| *height* | The height of the panel. |

### Member Function Documentation

#### void ui.elements.ScrollableTextPanel.addToText (String *text*)

Adds a text to the text area. Appends the text followed by a newline character to the text area, and sets the caret position to the end of the text area.

##### Parameters

|  |  |
| --- | --- |
| *text* | The text to add. |

#### void ui.elements.ScrollableTextPanel.clearText ()

Clears the text area. Sets the text of the text area to an empty string.

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

ui/elements/ScrollableTextPanel.java

## game.behaviour.entities.items.equipment.weapons.Shotgun Class Reference

### Public Member Functions

[Shotgun](#AAAAAAAAKD) (String id, String weaponName, double hitRange) throws ArgumentNullException, InvalidArgumentException

double **getHitRange** ()

[Shotgun](#AAAAAAAAKC) **setHitRange** (double newHitRange) throws InvalidArgumentException

boolean [attack](#AAAAAAAAKG) (int targetAC, double distance) throws DefaultDiceNotSetException

### Detailed Description

This class represents a Shotgun in the game. It extends the Weapon class.

The class contains the following fields:

hitRange: The hit range of the shotgun in which the shotgun will always hit the target.

The class provides a constructor that initializes the id, name, and hit range of the shotgun and methods to get and set the hit range and to attack a target.

### Constructor & Destructor Documentation

#### game.behaviour.entities.items.equipment.weapons.Shotgun.Shotgun (String *id*, String *weaponName*, double *hitRange*) throws ArgumentNullException, InvalidArgumentException

Constructor for the Shotgun class. Initializes the id, name, and hit range of the shotgun and sets the weapon type to SHOTGUN.

##### Parameters

|  |  |
| --- | --- |
| *id* | The id of the shotgun. |
| *weaponName* | The name of the shotgun. |
| *hitRange* | The hit range of the shotgun. |

##### Exceptions

|  |  |
| --- | --- |
| *ArgumentNullException* | if the id or weaponName is null. |
| *InvalidArgumentException* | if the hitRange is less than 0 or greater than 1. |

### Member Function Documentation

#### boolean game.behaviour.entities.items.equipment.weapons.Shotgun.attack (int *targetAC*, double *distance*) throws DefaultDiceNotSetException

Attacks a target. If the distance is less than or equal to the range times the hit range, the attack is guaranteed to hit.

##### Parameters

|  |  |
| --- | --- |
| *targetAC* | The armor class of the target. |
| *distance* | The distance to the target. |

##### Returns

True if the attack hits, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *DefaultDiceNotSetException* | if the default dice is not set. |

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

game/behaviour/entities/items/equipment/weapons/Shotgun.java

## game.global.storage.Storage< T > Class Template Reference

### Public Member Functions

void **add** (String id, T item) throws ArgumentNullException

void **remove** (String id) throws ArgumentNullException

T **get** (String id) throws ArgumentNullException

boolean **contains** (String id) throws ArgumentNullException

void **clear** ()

Set< Entry< String, T > > **entrySet** ()

ArrayList< T > [getAllItems](#AAAAAAAAOA) ()

ArrayList< String > [getAllKeys](#AAAAAAAAOB) ()

### Protected Attributes

HashMap< String, T > **storage**

### Detailed Description

This class represents a Storage in the game. It is a generic class that can store items of type T.

The class contains the following fields:

storage: A HashMap that stores the items.

The class provides a constructor that initializes the storage and methods to add, remove, and get items.

### Member Function Documentation

#### ArrayList< T > [game.global.storage.Storage](#AAAAAAAANZ)< T >.getAllItems ()

Gets all items from the storage.

##### Returns

An ArrayList containing all items in the storage.

#### ArrayList< String > [game.global.storage.Storage](#AAAAAAAANZ)< T >.getAllKeys ()

Gets all keys from the storage.

##### Returns

An ArrayList containing all keys in the storage.

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

game/global/storage/Storage.java

## ui.elements.TravelFrame Class Reference

### Public Member Functions

[TravelFrame](#AAAAAAAAUS) (String[] mapNames, GenericDelegate onMapSelect)

void [selectMap](#AAAAAAAAUT) ()

### Detailed Description

This class represents a travel frame in the UI. It allows the player to select a map to travel to in a combobox. It extends the JFrame class and contains a JComboBox for selecting a map and a GenericDelegate for handling map selection.

The class contains the following fields:

comboBox: The JComboBox for selecting a map.

onMapSelect: The GenericDelegate for handling map selection.

### Constructor & Destructor Documentation

#### ui.elements.TravelFrame.TravelFrame (String[] *mapNames*, GenericDelegate *onMapSelect*)

Constructor for the TravelFrame class. Initializes the frame, sets up the content, and sets the GenericDelegate for handling map selection.

##### Parameters

|  |  |
| --- | --- |
| *mapNames* | The names of the maps. |
| *onMapSelect* | The GenericDelegate for handling map selection. |

### Member Function Documentation

#### void ui.elements.TravelFrame.selectMap ()

Selects a map. Gets the selected item from the JComboBox, and if the GenericDelegate for handling map selection is not null, runs it with the selected item.

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

ui/elements/TravelFrame.java

## uilogic.TravelFrameHandler Class Reference

### Public Member Functions

void [start](#AAAAAAAAXX) ()

void [selectMap](#AAAAAAAAXY) (String mapName)

### Detailed Description

This class handles the travel frame in the UI. It contains methods to start the travel frame and select a map.

The class contains the following methods:

start: Starts the travel frame.

selectMap: Selects a map.

### Member Function Documentation

#### void uilogic.TravelFrameHandler.selectMap (String *mapName*)

Selects a map. Gets the ID of the map from the MapStorage, loads the current map with the ID and the items in the player's inventory, and catches any exceptions.

##### Parameters

|  |  |
| --- | --- |
| *mapName* | The name of the map. |

#### void uilogic.TravelFrameHandler.start ()

Starts the travel frame. Gets all the keys from the MapStorage, creates a new TravelFrame with the keys and a delegate to select a map, and sets the TravelFrame to visible.

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

uilogic/TravelFrameHandler.java

## game.global.UIHandler Class Reference

### Public Member Functions

PlayFieldHandler **getPlayFieldHandler** ()

CombatLogger **getCombatLogger** ()

void [start](#AAAAAAAANN) () throws Exception

void [refreshUI](#AAAAAAAANO) ()

void [openFileDialog](#AAAAAAAANP) (FileChooserType type)

void [showMessage](#AAAAAAAANQ) (String message, int messageType)

void [togglePlayerControlls](#AAAAAAAANR) (boolean on)

void [displayDiceRollResult](#AAAAAAAANS) (Integer roll)

void [displayPlayerDeath](#AAAAAAAANT) ()

void **displayCharacterFrame** ()

void [displayTravelFrame](#AAAAAAAANV) ()

### Static Public Member Functions

static [UIHandler](#AAAAAAAANK) **getInstance** ()

### Detailed Description

This class handles the user interface of the game. It contains handlers for the play field, combat logger, and various buttons.

The class contains the following fields:

instance: The singleton instance of the UIHandler class.

playFieldHandler: The handler for the play field.

logger: The combat logger.

playFrameMenuBarHandler: The handler for the play frame menu bar.

utilityButtonHandler: The handler for the utility buttons.

interactButtonHandler: The handler for the interact buttons.

playerDeathFrameHandler: The handler for the player death frame.

characterFrameHandler: The handler for the character frame.

travelHandler: The handler for travel.

### Member Function Documentation

#### void game.global.UIHandler.displayDiceRollResult (Integer *roll*)

Displays the dice roll result. Acts as a callback for the dice roller. Logs the result of a dice roll.

##### Parameters

|  |  |
| --- | --- |
| *roll* | The result of the dice roll. |

#### void game.global.UIHandler.displayPlayerDeath ()

Displays the player death screen. Prompts the player to restart or quit the game. Shows a screen indicating the player has died.

#### void game.global.UIHandler.displayTravelFrame ()

Displays the travel frame. Allows the player to travel to a different map. Shows a frame allowing the player to travel.

#### void game.global.UIHandler.openFileDialog (FileChooserType *type*)

Opens a file dialog for text files. Allows the user to choose a file to load.

#### void game.global.UIHandler.refreshUI ()

Refreshes the UI. Updates the UI to reflect the current game state.

#### void game.global.UIHandler.showMessage (String *message*, int *messageType*)

Shows a message. Displays a message to the user.

##### Parameters

|  |  |
| --- | --- |
| *message* | The message to display. |
| *messageType* | The type of the message. Use the JOptionPane constants. |

#### void game.global.UIHandler.start () throws Exception

Starts the UIHandler. Sets up the main UI frame. Initializes the UI and displays the main menu.

#### void game.global.UIHandler.togglePlayerControlls (boolean *on*)

Toggles the player controls. Enables or disables the player controls.

##### Parameters

|  |  |
| --- | --- |
| *on* | Whether to enable the player controls. |

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

game/global/UIHandler.java

## exception.ui.UIHandlerAlreadyStartedException Class Reference

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

exception/ui/UIHandlerAlreadyStartedException.java

## uilogic.UtilityButtonHandler Class Reference

### Protected Member Functions

void [initActions](#AAAAAAAAXN) ()

#### Protected Member Functions inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

[MultipleButtonHandler](#AAAAAAAAWR) ()

### Additional Inherited Members

#### Public Member Functions inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

void [actionPerformed](#AAAAAAAAWS) (ActionEvent e)

#### Protected Attributes inherited from [uilogic.MultipleButtonHandler](#AAAAAAAAWQ)

HashMap< String, GenericDelegate > **actions**

### Detailed Description

This class handles the actions of the utility buttons in the UI. It extends the MultipleButtonHandler class and overrides the initActions method to initialize the actions of the utility buttons.

The class contains the following methods:

initActions: Initializes the actions of the utility buttons.

### Member Function Documentation

#### void uilogic.UtilityButtonHandler.initActions ()[protected]

Initializes the actions of the utility buttons. The actions include displaying the character frame and the travel frame. Each action is associated with a key in the actions HashMap of the MultipleButtonHandler class.

Reimplemented from [uilogic.MultipleButtonHandler](#AAAAAAAAWT).

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

uilogic/UtilityButtonHandler.java

## ui.elements.UtilityButtonPanel Class Reference

### Public Member Functions

[UtilityButtonPanel](#AAAAAAAAUV) (int width, int height, ActionListener listener)

void [toggleButtons](#AAAAAAAAUW) (boolean enabled)

### Detailed Description

This class represents a utility button panel in the UI. It allows the player to travel and manage their equipment. It extends the JPanel class and contains a list of JButtons for utility functions.

The class contains the following field:

buttons: The list of JButtons in the panel.

### Constructor & Destructor Documentation

#### ui.elements.UtilityButtonPanel.UtilityButtonPanel (int *width*, int *height*, ActionListener *listener*)

Constructor for the UtilityButtonPanel class. Initializes the panel and the buttons with the specified width, height, and action listener.

##### Parameters

|  |  |
| --- | --- |
| *width* | The width of the panel. |
| *height* | The height of the panel. |
| *listener* | The action listener of the buttons. |

### Member Function Documentation

#### void ui.elements.UtilityButtonPanel.toggleButtons (boolean *enabled*)

Toggles the buttons. Enables or disables the buttons.

##### Parameters

|  |  |
| --- | --- |
| *enabled* | A boolean that determines whether to enable or disable the buttons. |

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

ui/elements/UtilityButtonPanel.java

## game.behaviour.abstracts.Weapon Class Reference

### Public Member Functions

[Weapon](#AAAAAAAAEO) **setAttackModifier** (int modifier)

[Weapon](#AAAAAAAAEO) **setDamageDice** (int sides) throws InvalidDiceSideCountException

[Weapon](#AAAAAAAAEO) **setDiceCount** (int count) throws InvalidDiceSideCountException

[Weapon](#AAAAAAAAEO) **setDamageModifier** (int modifier)

[Weapon](#AAAAAAAAEO) **setRange** (int newRange) throws InvalidArgumentException

[Weapon](#AAAAAAAAEO) **setAttacksInRound** (int amount) throws InvalidArgumentException

[Weapon](#AAAAAAAAEO) **setName** (String newName) throws ArgumentNullException

[Weapon](#AAAAAAAAEO) **setDescription** (String newDescription) throws ArgumentNullException

int **getDamageDice** ()

int **getAttackModifier** ()

int **getDamageDiceCount** ()

int **getDamageModifier** ()

int **getAttacksInRound** ()

double **getRange** ()

String **getName** ()

String **getDescription** ()

WeaponType **getWeaponType** ()

String **getDisplayInfo** ()

String **getStatistics** (int bearerLevel)

boolean [checkRange](#AAAAAAAAGG) (double distance)

boolean [attack](#AAAAAAAAGH) (int targetAC, double distance) throws DefaultDiceNotSetException

int [damage](#AAAAAAAAGI) (double distance) throws InvalidDiceSideCountException

#### Public Member Functions inherited from [game.behaviour.abstracts.Equipment](#AAAAAAAAFK)

EquipmentType **getEquipmentType** ()

### Protected Member Functions

**Weapon** (String id, String weaponName) throws ArgumentNullException

### Protected Attributes

int **attackModifier**

int **damageDice**

int **diceCount**

int **damageModifier**

double **range**

WeaponType **weaponType**

int **attacksInRound**

#### Protected Attributes inherited from [game.behaviour.abstracts.Equipment](#AAAAAAAAFK)

EquipmentType **equipmentType**

### Detailed Description

This abstract class represents a Weapon in the game. It extends the Equipment class and includes additional properties such as attack modifier, damage dice, dice count, damage modifier, range, weapon type, and attacks in round.

The class contains the following fields:

attackModifier: The bonus value that is added to the default attack roll.

damageDice: Represents the sideCount of the damage dice it rolls with.

diceCount: The number of dice that is rolled for damage.

damageModifier: The bonus value that is added to the rolled damage.

range: The maximum distance the weapon can deal damage in.

weaponType: The type of the weapon, used for type casting.

attacksInRound: The number of attacks in a round.

The class provides getter and setter methods for these fields.

### Member Function Documentation

#### boolean game.behaviour.abstracts.Weapon.attack (int *targetAC*, double *distance*) throws DefaultDiceNotSetException

Performs an attack roll with default dice and checks if it succeeds against the target's armor class. Also checks if the target is within the weapon's range.

##### Parameters

|  |  |
| --- | --- |
| *targetAC* | The armor class of the target. |
| *distance* | The distance to the target. |

##### Returns

true if the attack hits and the target is within range, false otherwise.

##### Exceptions

|  |  |
| --- | --- |
| *DefaultDiceNotSetException* | if the default dice is not set. |

#### boolean game.behaviour.abstracts.Weapon.checkRange (double *distance*)

Checks if the weapon can attack to the specified distance.

##### Parameters

|  |  |
| --- | --- |
| *distance* | The distance to check. |

##### Returns

true if the weapon can attack to the distance, false otherwise.

#### int game.behaviour.abstracts.Weapon.damage (double *distance*) throws InvalidDiceSideCountException

Calculates the damage dealt by the weapon. Rolls the weapon's damage dice and adds the damage modifier.

##### Parameters

|  |  |
| --- | --- |
| *distance* | The distance to the target. |

##### Returns

The calculated damage.

##### Exceptions

|  |  |
| --- | --- |
| *InvalidDiceSideCountException* | if the side count of the damage dice is invalid. |

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

game/behaviour/abstracts/Weapon.java

## game.enums.WeaponType Enum Reference

### Public Attributes

**RIFLE**

**SHOTGUN**

**SNIPER**

**PISTOL**

**AUTOPISTOL**

**AUTORIFLE**

**MELEE**

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

game/enums/WeaponType.java

# Index

INDEX