

Scars\_Healing

Generated by Doxygen 1.8.17



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 Namespace Documentation</b>	<b>7</b>
4.1 gtest_lite Namespace Reference	7
4.1.1 Detailed Description	8
4.1.2 Function Documentation	8
4.1.2.1 almostEQ()	8
4.1.2.2 eq()	8
4.1.2.3 eqstr()	8
4.1.2.4 EXPECT_() [1/2]	8
4.1.2.5 EXPECT_() [2/2]	9
4.1.2.6 EXPECTSTR()	9
4.1.2.7 ge()	10
4.1.2.8 gt()	10
4.1.2.9 le()	10
4.1.2.10 lt()	10
4.1.2.11 ne()	11
4.1.2.12 nestr()	11
<b>5 Class Documentation</b>	<b>13</b>
5.1 _Is_Types< F, T > Struct Template Reference	13
5.1.1 Detailed Description	14
5.1.2 Member Function Documentation	14
5.1.2.1 f() [1/2]	14
5.1.2.2 f() [2/2]	14
5.1.3 Member Data Documentation	14
5.1.3.1 convertible	14
5.2 AttributeCheck< C > Class Template Reference	15
5.2.1 Detailed Description	15
5.2.2 Constructor & Destructor Documentation	15
5.2.2.1 AttributeCheck()	15
5.2.3 Member Function Documentation	16
5.2.3.1 operator() [1/2]	16
5.2.3.2 operator() [2/2]	16
5.3 Enemy Class Reference	16
5.3.1 Detailed Description	18
5.3.2 Constructor & Destructor Documentation	19

5.3.2.1 Enemy() [1/2]	19
5.3.2.2 Enemy() [2/2]	19
5.3.2.3 ~Enemy()	19
5.3.3 Member Function Documentation	19
5.3.3.1 attackPlayer()	20
5.3.3.2 getInfoEntity()	20
5.4 Entity Class Reference	21
5.4.1 Detailed Description	23
5.4.2 Constructor & Destructor Documentation	24
5.4.2.1 Entity() [1/2]	24
5.4.2.2 Entity() [2/2]	24
5.4.2.3 ~Entity()	24
5.4.3 Member Function Documentation	24
5.4.3.1 getBaseDEF()	25
5.4.3.2 getBaseDMG()	25
5.4.3.3 getHP()	26
5.4.3.4 getInfoEntity()	26
5.4.3.5 getMaxHP()	27
5.4.3.6 getName()	27
5.4.3.7 setHP()	28
5.5 GenericArray< C > Class Template Reference	29
5.5.1 Detailed Description	30
5.5.2 Constructor & Destructor Documentation	30
5.5.2.1 GenericArray() [1/2]	30
5.5.2.2 GenericArray() [2/2]	30
5.5.2.3 ~GenericArray()	30
5.5.3 Member Function Documentation	30
5.5.3.1 add()	31
5.5.3.2 clear()	31
5.5.3.3 getPredNum()	31
5.5.3.4 getSize()	32
5.5.3.5 operator[]() [1/2]	32
5.5.3.6 operator[]() [2/2]	32
5.5.3.7 remove()	33
5.6 InfoPreset Class Reference	33
5.6.1 Detailed Description	35
5.6.2 Constructor & Destructor Documentation	35
5.6.2.1 InfoPreset()	35
5.6.2.2 ~InfoPreset()	35
5.6.3 Member Data Documentation	36
5.6.3.1 BaseDEF	36
5.6.3.2 BaseDMG	36

5.6.3.3 DisplayNum	36
5.6.3.4 Gold	36
5.6.3.5 HP	36
5.6.3.6 Items	37
5.6.3.7 Name	37
5.6.3.8 Num	37
5.6.3.9 Pre	37
5.6.3.10 Storage	37
5.7 Item Class Reference	38
5.7.1 Detailed Description	40
5.7.2 Constructor & Destructor Documentation	40
5.7.2.1 Item()	40
5.7.2.2 ~Item()	40
5.7.3 Member Function Documentation	40
5.7.3.1 getBonusDEF()	40
5.7.3.2 getBonusDEFM()	41
5.7.3.3 getBonusDMG()	41
5.7.3.4 getBonusDMGM()	42
5.7.3.5 getDesc()	42
5.7.3.6 getHeal()	42
5.7.3.7 getInfoltems()	43
5.7.3.8 getName()	43
5.7.3.9 getType()	43
5.7.3.10 useItem()	44
5.7.4 Member Data Documentation	44
5.7.4.1 BonusDEF	44
5.7.4.2 BonusDEFM	44
5.7.4.3 BonusDMG	44
5.7.4.4 BonusDMGM	45
5.7.4.5 Desc	45
5.7.4.6 Heal	45
5.7.4.7 Name	45
5.7.4.8 Type	45
5.8 Player Class Reference	46
5.8.1 Detailed Description	48
5.8.2 Constructor & Destructor Documentation	48
5.8.2.1 Player() [1/2]	48
5.8.2.2 Player() [2/2]	48
5.8.2.3 ~Player()	48
5.8.3 Member Function Documentation	49
5.8.3.1 attackEnemy() [1/2]	49
5.8.3.2 attackEnemy() [2/2]	49

5.8.3.3	<a href="#">getGold()</a>	50
5.8.3.4	<a href="#">getInfoEntity()</a>	51
5.8.3.5	<a href="#">getStorageType()</a>	51
5.8.3.6	<a href="#">Heal()</a>	52
5.9	<a href="#">Storage Class Reference</a>	52
5.9.1	<a href="#">Detailed Description</a>	55
5.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	55
5.9.2.1	<a href="#">Storage() [1/2]</a>	55
5.9.2.2	<a href="#">Storage() [2/2]</a>	55
5.9.2.3	<a href="#">~Storage()</a>	55
5.9.3	<a href="#">Member Function Documentation</a>	55
5.9.3.1	<a href="#">add()</a>	56
5.9.3.2	<a href="#">clear()</a>	56
5.9.3.3	<a href="#">getDescription()</a>	57
5.9.3.4	<a href="#">getInfo()</a>	57
5.9.3.5	<a href="#">getInfoItems() [1/2]</a>	57
5.9.3.6	<a href="#">getInfoItems() [2/2]</a>	58
5.9.3.7	<a href="#">getInfoItemsAll() [1/2]</a>	58
5.9.3.8	<a href="#">getInfoItemsAll() [2/2]</a>	59
5.9.3.9	<a href="#">getInfoItemsUse() [1/2]</a>	59
5.9.3.10	<a href="#">getInfoItemsUse() [2/2]</a>	60
5.9.3.11	<a href="#">getItems()</a>	60
5.9.3.12	<a href="#">getMaxSize()</a>	60
5.9.3.13	<a href="#">getName()</a>	60
5.9.3.14	<a href="#">getSize()</a>	61
5.9.3.15	<a href="#">isThereUpgrade()</a>	61
5.9.3.16	<a href="#">operator[]() [1/2]</a>	62
5.9.3.17	<a href="#">operator[]() [2/2]</a>	62
5.9.3.18	<a href="#">remove()</a>	62
5.9.3.19	<a href="#">setMaxSize()</a>	63
5.9.4	<a href="#">Member Data Documentation</a>	63
5.9.4.1	<a href="#">Desc</a>	63
5.9.4.2	<a href="#">Items</a>	63
5.9.4.3	<a href="#">MaxSize</a>	63
5.9.4.4	<a href="#">Name</a>	63
5.10	<a href="#">String Class Reference</a>	64
5.10.1	<a href="#">Detailed Description</a>	65
5.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	65
5.10.2.1	<a href="#">String() [1/3]</a>	65
5.10.2.2	<a href="#">String() [2/3]</a>	65
5.10.2.3	<a href="#">String() [3/3]</a>	65
5.10.2.4	<a href="#">~String()</a>	66

5.10.3 Member Function Documentation	66
5.10.3.1 getSize()	66
5.10.3.2 getStr()	66
5.10.3.3 operator+() [1/2]	67
5.10.3.4 operator+() [2/2]	67
5.10.3.5 operator+=()	67
5.10.3.6 operator=() [1/2]	67
5.10.3.7 operator=() [2/2]	68
5.11 gtest_lite::Test Struct Reference	68
5.11.1 Detailed Description	69
5.11.2 Constructor & Destructor Documentation	69
5.11.2.1 ~Test()	69
5.11.3 Member Function Documentation	69
5.11.3.1 begin()	69
5.11.3.2 end()	70
5.11.3.3 expect()	70
5.11.3.4 fail()	70
5.11.3.5 getTest()	70
5.11.4 Member Data Documentation	71
5.11.4.1 ablocks	71
5.11.4.2 failed	71
5.11.4.3 name	71
5.11.4.4 null	71
5.11.4.5 status	71
5.11.4.6 sum	71
5.11.4.7 tmp	71
<b>Index</b>	<b>73</b>





# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">gtest_lite</a>	Gtest_lite: a keretrendszer függvényinek és objektumainak névtére . . . . .	7
----------------------------	---	---



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

_Is_Types< F, T > . . . . .	13
AttributeCheck< C > . . . . .	15
Entity . . . . .	21
Enemy . . . . .	16
Player . . . . .	46
GenericArray< C > . . . . .	29
GenericArray< Item > . . . . .	29
InfoPreset . . . . .	33
Item . . . . .	38
Storage . . . . .	52
String . . . . .	64
gtest_lite::Test . . . . .	68



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">_Is_Types&lt; F, T &gt;</a>	Segédsablon típuskonverzió futás közbeni ellenőrzésére . . . . .	13
<a href="#">AttributeCheck&lt; C &gt;</a>	Predicatum, which determines if an item in an array has a specific attribute . . . . .	15
<a href="#">Enemy</a>	<a href="#">Enemy</a> can attack a <a href="#">Player</a> without the <a href="#">Player</a> damaging it, every <a href="#">Enemy</a> will have a Level, and will have a chance to drop an <a href="#">Item</a> if it dies according to its Level . . . . .	16
<a href="#">Entity</a>	An alive <a href="#">Entity</a> can be a <a href="#">Player</a> or an <a href="#">Enemy</a> (for now). It has HP, a Name and basic attributes, like BaseDMG and BaseDEF . . . . .	21
<a href="#">GenericArray&lt; C &gt;</a>	This <a href="#">GenericArray</a> can store any type of things dynamically. Maybe I will do a static version as well. Or just implement it with a bool. Time will tell.. . . .	29
<a href="#">InfoPreset</a>	A preset tool for displaying dynamic information about anything . . . . .	33
<a href="#">Item</a>	An <a href="#">Item</a> can be stored in a storage and also has a type. Depending on the type, an <a href="#">Item</a> can heal the player or damage an enemy, etc . . . . .	38
<a href="#">Player</a>	<a href="#">Player</a> has Gold and a <a href="#">Storage</a> in which it can hold some <a href="#">Item</a> . The <a href="#">Player</a> will be able to spend Gold in the Shop . . . . .	46
<a href="#">Storage</a>	A <a href="#">Storage</a> is able to store Items, but to a limited extent (MaxSize). Also has a Name and a Description . . . . .	52
<a href="#">String</a>	Seperate <a href="#">String</a> class, because STL are not allowed.. . . .	64
<a href="#">gtest_lite::Test</a>	. . . . .	68



## Chapter 4

# Namespace Documentation

### 4.1 gtest\_lite Namespace Reference

[gtest\\_lite](#): a keretrendszer függvényinek és objektumainak névtére

#### Classes

- struct [Test](#)

#### Functions

- `template<typename T1 , typename T2 >`  
`std::ostream & EXPECT\_ (T1 exp, T2 act, bool(*pred)(T1, T2), const char *file, int line, const char *expr,`  
`const char *lhs="elvart", const char *rhs="aktual")`  
*általános sablon a várt értékhez.*
- `template<typename T1 , typename T2 >`  
`std::ostream & EXPECT\_ (T1 *exp, T2 *act, bool(*pred)(T1 *, T2 *), const char *file, int line, const char`  
`*expr, const char *lhs="elvart", const char *rhs="aktual")`  
*pointerre specializált sablon a várt értékhez.*
- `std::ostream & EXPECTSTR (const char *exp, const char *act, bool(*pred)(const char *, const char *), const`  
`char *file, int line, const char *expr, const char *lhs="elvart", const char *rhs="aktual")`
- `template<typename T1 , typename T2 >`  
`bool eq (T1 a, T2 b)`
- `bool eqstr (const char *a, const char *b)`
- `template<typename T1 , typename T2 >`  
`bool ne (T1 a, T2 b)`
- `bool nestr (const char *a, const char *b)`
- `template<typename T1 , typename T2 >`  
`bool le (T1 a, T2 b)`
- `template<typename T1 , typename T2 >`  
`bool lt (T1 a, T2 b)`
- `template<typename T1 , typename T2 >`  
`bool ge (T1 a, T2 b)`
- `template<typename T1 , typename T2 >`  
`bool gt (T1 a, T2 b)`
- `template<typename T >`  
`bool almostEQ (T a, T b)`

### 4.1.1 Detailed Description

`gtest_lite`: a keretrendszer függvényinek és objektumainak névtére

### 4.1.2 Function Documentation

#### 4.1.2.1 `almostEQ()`

```
template<typename T >
bool gtest_lite::almostEQ (
    T a,
    T b )
```

Segédsablon valós számok összehasonlításához Nem bombabiztos, de nekünk most jó lesz Elméleti hátér:  
<http://www.cygnus-software.com/papers/comparingfloats/comparingfloats.htm>

#### 4.1.2.2 `eq()`

```
template<typename T1 , typename T2 >
bool gtest_lite::eq (
    T1 a,
    T2 b )
```

segéd sablonok a relációkhoz. azért nem STL (algorithm), mert csak a függvény lehet, hogy menjen a deduckció

#### 4.1.2.3 `eqstr()`

```
bool gtest_lite::eqstr (
    const char * a,
    const char * b ) [inline]
```

#### 4.1.2.4 `EXPECT_()` [1/2]

```
template<typename T1 , typename T2 >
std::ostream& gtest_lite::EXPECT_ (
    T1 * exp,
    T2 * act,
    bool(*) (T1 *, T2 *) pred,
    const char * file,
    int line,
    const char * expr,
    const char * lhs = "elvart",
    const char * rhs = "aktual" )
```

pointerre specializált sablon a várt értékhez.



Here is the call graph for this function:



#### 4.1.2.5 EXPECT\_() [2/2]

```
template<typename T1 , typename T2 >
std::ostream& gtest_lite::EXPECT_ (
    T1 exp,
    T2 act,
    bool(*) (T1, T2) pred,
    const char * file,
    int line,
    const char * expr,
    const char * lhs = "elvart",
    const char * rhs = "aktual" )
```

általános sablon a várt értékhez.

Here is the call graph for this function:



#### 4.1.2.6 EXPECTSTR()

```
std::ostream& gtest_lite::EXPECTSTR (
    const char * exp,
    const char * act,
    bool(*) (const char *, const char *) pred,
    const char * file,
    int line,
    const char * expr,
```

```
const char * lhs = "elvart",
const char * rhs = "aktual" ) [inline]
```

stringek összehasonlításához. azért nem spec. mert a sima EQ-ra másként kell működnie. Here is the call graph for this function:



#### 4.1.2.7 ge()

```
template<typename T1 , typename T2 >
bool gtest_lite::ge (
    T1 a,
    T2 b )
```

#### 4.1.2.8 gt()

```
template<typename T1 , typename T2 >
bool gtest_lite::gt (
    T1 a,
    T2 b )
```

#### 4.1.2.9 le()

```
template<typename T1 , typename T2 >
bool gtest_lite::le (
    T1 a,
    T2 b )
```

#### 4.1.2.10 lt()

```
template<typename T1 , typename T2 >
bool gtest_lite::lt (
    T1 a,
    T2 b )
```

#### 4.1.2.11 ne()

```
template<typename T1 , typename T2 >  
bool gtest_lite::ne (  
    T1 a,  
    T2 b )
```

#### 4.1.2.12 nestr()

```
bool gtest_lite::nestr (  
    const char * a,  
    const char * b ) [inline]
```



## Chapter 5

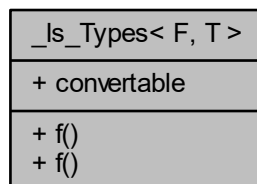
# Class Documentation

### 5.1 `_Is_Types< F, T >` Struct Template Reference

Segédsablon típuskonverzió futás közbeni ellenőrzésére.

```
#include <gtest_lite.h>
```

Collaboration diagram for `_Is_Types< F, T >`:



#### Static Public Member Functions

- `template<typename D >`  
`static char(& f (D))[1]`
- `template<typename D >`  
`static char(& f (...))[2]`

#### Static Public Attributes

- `static const bool convertible = sizeof(f<T>(F())) == 1`

### 5.1.1 Detailed Description

```
template<typename F, typename T>
struct _Is_Types< F, T >
```

Segédsablon típuskonverzió futás közbeni ellenőrzésére.

### 5.1.2 Member Function Documentation

#### 5.1.2.1 f() [1/2]

```
template<typename F , typename T >
template<typename D >
static char(& _Is_Types< F, T >::f (
    ... )) [2] [static]
```

#### 5.1.2.2 f() [2/2]

```
template<typename F , typename T >
template<typename D >
static char(& _Is_Types< F, T >::f (
    D )) [1] [static]
```

### 5.1.3 Member Data Documentation

#### 5.1.3.1 convertible

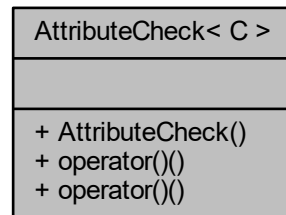
```
template<typename F , typename T >
const bool _Is_Types< F, T >::convertable = sizeof(f<T>(F())) == 1 [static]
```

## 5.2 AttributeCheck< C > Class Template Reference

Predicatum, which determines if an item in an array has a specific attribute.

```
#include <Item.h>
```

Collaboration diagram for AttributeCheck< C >:



### Public Member Functions

- [AttributeCheck](#) (C Attribute)  
*Default Constructor.*
- bool [operator\(\)](#) (const C &c)  
*Decides if the attribute is equal.*
- bool [operator\(\)](#) (const [Item](#) &c)  
*Checks if the type of an [Item](#) is something.*

### 5.2.1 Detailed Description

```
template<class C>
class AttributeCheck< C >
```

Predicatum, which determines if an item in an array has a specific attribute.

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 AttributeCheck()

```
template<class C >
AttributeCheck< C >::AttributeCheck (
    C Attribute ) [inline]
```

Default Constructor.

## 5.2.3 Member Function Documentation

### 5.2.3.1 operator() [1/2]

```
template<class C >
bool AttributeCheck< C >::operator() (
    const C & c ) [inline]
```

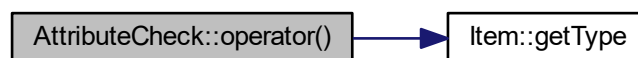
Decides if the attribute is equal.

### 5.2.3.2 operator() [2/2]

```
template<class C >
bool AttributeCheck< C >::operator() (
    const Item & c ) [inline]
```

Checks if the type of an [Item](#) is something.

Here is the call graph for this function:



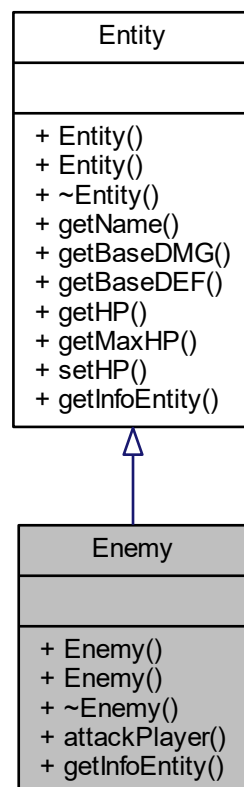
## 5.3 Enemy Class Reference

[Enemy](#) can attack a [Player](#) without the [Player](#) damaging it, every [Enemy](#) will have a Level, and will have a chance to drop an [Item](#) if it dies according to its Level.

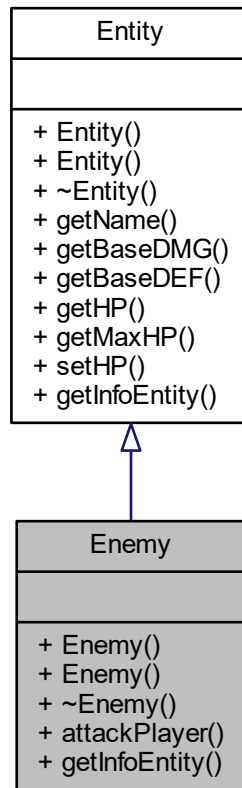
```
#include <Enemy.h>
```



Inheritance diagram for Enemy:



Collaboration diagram for Enemy:



## Public Member Functions

- [Enemy](#) (const char \*Name, double HP=100, double MaxHP=100, double BaseDMG=5, double BaseDEF=5)  
*Constructor.*
- [Enemy](#) (double HP=100, double MaxHP=100, double BaseDMG=5, double BaseDEF=5)  
*Default Constructor.*
- virtual [~Enemy](#) ()  
*Destructor.*
- void [attackPlayer](#) ([Player](#) &PLYR, [Item](#) WS=[Item](#)())  
*Attack [Player](#) without taking Damage from it.*
- std::ostream & [getInfoEntity](#) ([InfoPreset](#) Preset=[InfoPreset](#)()), std::ostream &os=std::cout)  
*Displays the info of an [Enemy](#), dynamically controlable with an [InfoPreset](#).*

### 5.3.1 Detailed Description

[Enemy](#) can attack a [Player](#) without the [Player](#) damaging it, every [Enemy](#) will have a Level, and will have a chance to drop an [Item](#) if it dies according to its Level.

## 5.3.2 Constructor & Destructor Documentation

### 5.3.2.1 Enemy() [1/2]

```
Enemy::Enemy (
    const char * Name,
    double HP = 100,
    double MaxHP = 100,
    double BaseDMG = 5,
    double BaseDEF = 5 )
```

Constructor.

### 5.3.2.2 Enemy() [2/2]

```
Enemy::Enemy (
    double HP = 100,
    double MaxHP = 100,
    double BaseDMG = 5,
    double BaseDEF = 5 )
```

Default Constructor.

### 5.3.2.3 ~Enemy()

```
Enemy::~~Enemy ( ) [virtual]
```

Destructor.

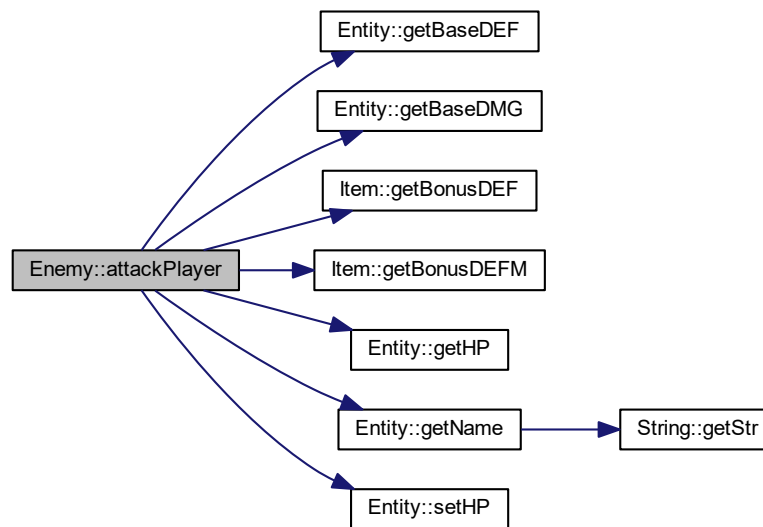
## 5.3.3 Member Function Documentation

### 5.3.3.1 attackPlayer()

```
void Enemy::attackPlayer (
    Player & PLYR,
    Item WS = Item() )
```

Attack [Player](#) without taking Damage from it.

Here is the call graph for this function:



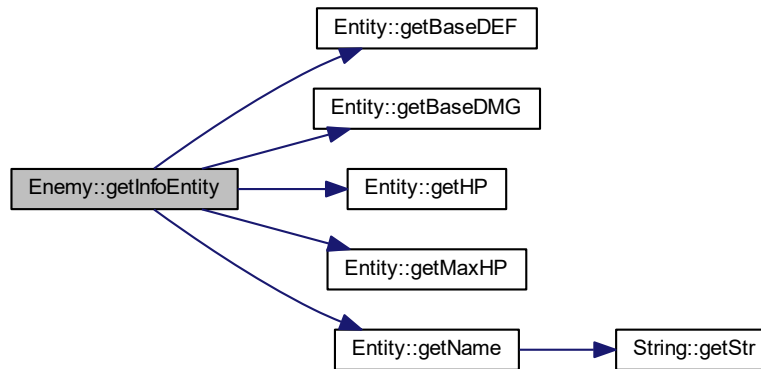
### 5.3.3.2 getInfoEntity()

```
std::ostream & Enemy::getInfoEntity (
    InfoPreset Preset = InfoPreset(),
    std::ostream & os = std::cout ) [virtual]
```

Displays the info of an [Enemy](#), dynamically controlable with an [InfoPreset](#).

Implements [Entity](#).

Here is the call graph for this function:

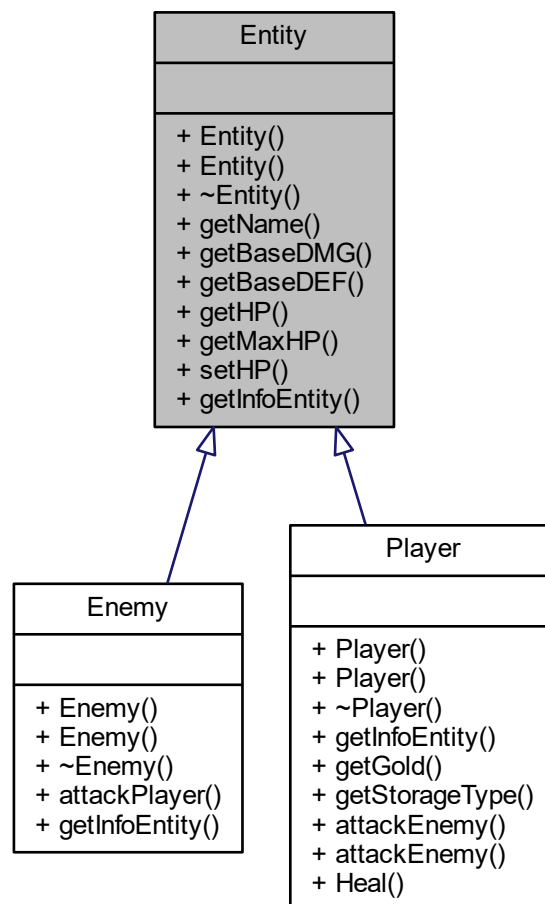


## 5.4 Entity Class Reference

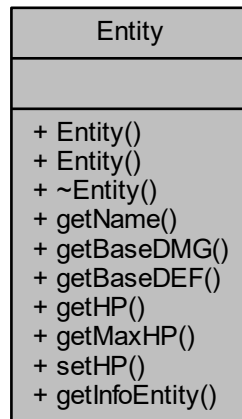
An alive [Entity](#) can be a [Player](#) or an [Enemy](#) (for now). It has HP, a Name and basic attributes, like BaseDMG and BaseDEF.

```
#include <Entity.h>
```

Inheritance diagram for Entity:



Collaboration diagram for Entity:



## Public Member Functions

- [Entity](#) (const char \*Name, double HP=100, double MaxHP=100, double BaseDMG=5, double BaseDEF=5)  
*Constructor.*
- [Entity](#) (double HP=100, double MaxHP=100, double BaseDMG=5, double BaseDEF=5)  
*Default Constructor.*
- virtual [~Entity](#) ()  
*Destructor.*
- const char \* [getName](#) ()  
*Gets [Entity](#)'s name.*
- double [getBaseDMG](#) ()  
*Gets [Entity](#)'s Base Damage.*
- double [getBaseDEF](#) ()  
*Gets [Entity](#)'s Base Defense.*
- double [getHP](#) ()  
*Gets [Entity](#)'s HP.*
- double [getMaxHP](#) ()  
*Gets [Entity](#)'s MaxHP.*
- void [setHP](#) (const double hp)  
*Sets [Entity](#)'s HP.*
- virtual std::ostream & [getInfoEntity](#) ([InfoPreset](#) Preset=[InfoPreset](#)(), std::ostream &os=std::cout)=0  
*Displays the info of an [Entity](#), dynamically controllable with an [InfoPreset](#).*

### 5.4.1 Detailed Description

An alive [Entity](#) can be a [Player](#) or an [Enemy](#) (for now). It has HP, a Name and basic attributes, like BaseDMG and BaseDEF.

## 5.4.2 Constructor & Destructor Documentation

### 5.4.2.1 Entity() [1/2]

```
Entity::Entity (
    const char * Name,
    double HP = 100,
    double MaxHP = 100,
    double BaseDMG = 5,
    double BaseDEF = 5 )
```

Constructor.

### 5.4.2.2 Entity() [2/2]

```
Entity::Entity (
    double HP = 100,
    double MaxHP = 100,
    double BaseDMG = 5,
    double BaseDEF = 5 )
```

Default Constructor.

### 5.4.2.3 ~Entity()

```
Entity::~~Entity ( ) [virtual]
```

Destructor.

## 5.4.3 Member Function Documentation

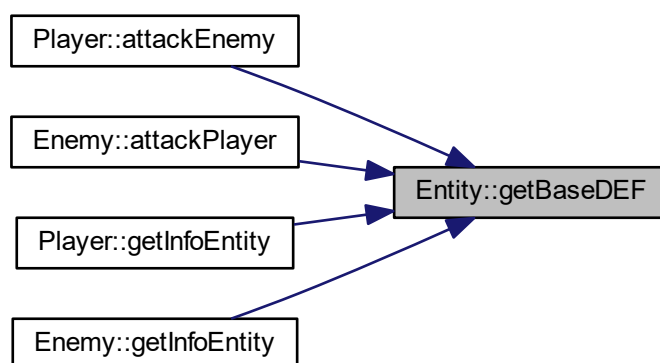


#### 5.4.3.1 getBaseDEF()

```
double Entity::getBaseDEF ( )
```

Gets [Entity](#)'s Base Defense.

Here is the caller graph for this function:

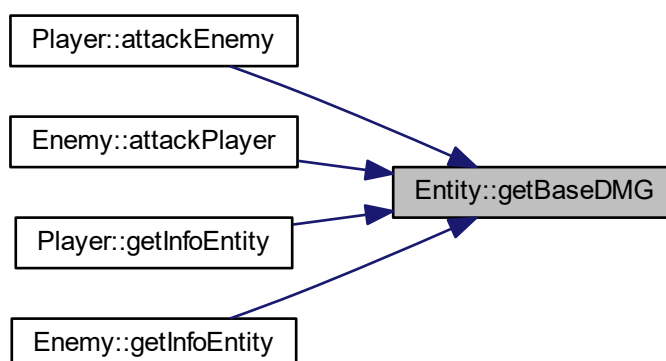


#### 5.4.3.2 getBaseDMG()

```
double Entity::getBaseDMG ( )
```

Gets [Entity](#)'s Base Damage.

Here is the caller graph for this function:

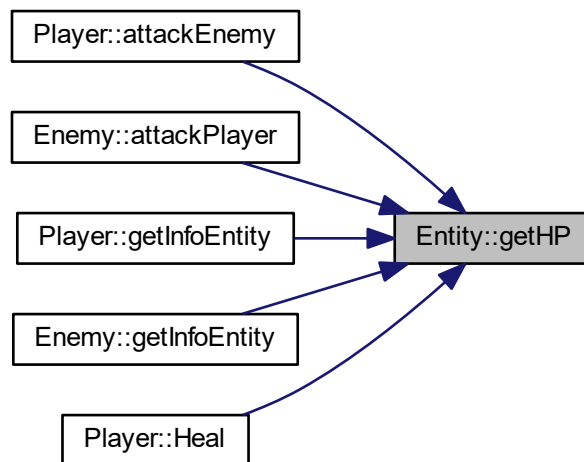


#### 5.4.3.3 getHP()

```
double Entity::getHP ( )
```

Gets [Entity](#)'s HP.

Here is the caller graph for this function:



#### 5.4.3.4 getInfoEntity()

```
virtual std::ostream& Entity::getInfoEntity (
    InfoPreset Preset = InfoPreset(),
    std::ostream & os = std::cout ) [pure virtual]
```

Displays the info of an [Entity](#), dynamically controllable with an [InfoPreset](#).

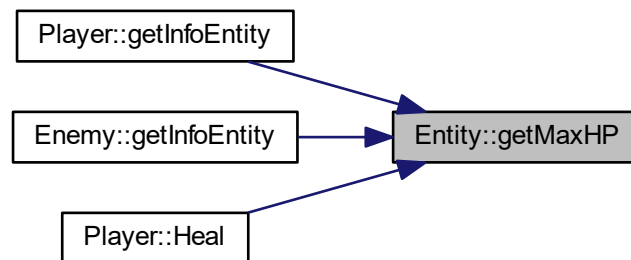
Implemented in [Enemy](#), and [Player](#).

#### 5.4.3.5 getMaxHP()

```
double Entity::getMaxHP ( )
```

Gets [Entity](#)'s MaxHP.

Here is the caller graph for this function:



#### 5.4.3.6 getName()

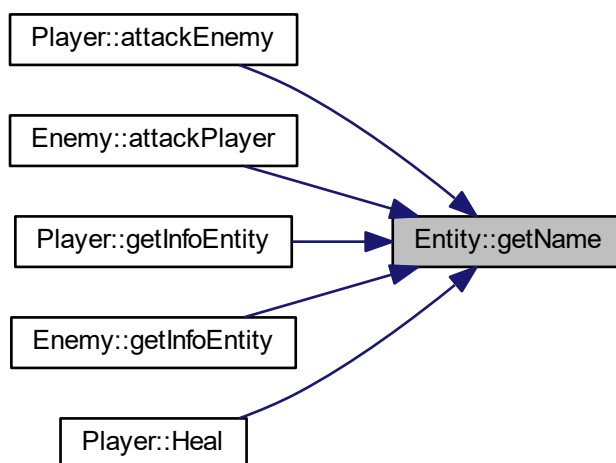
```
const char * Entity::getName ( )
```

Gets [Entity](#)'s name.

Here is the call graph for this function:



Here is the caller graph for this function:

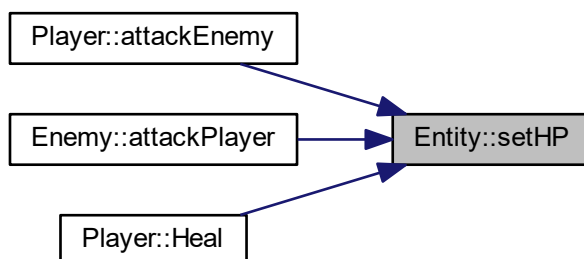


#### 5.4.3.7 setHP()

```
void Entity::setHP (
    const double hp )
```

Sets [Entity](#)'s HP.

Here is the caller graph for this function:

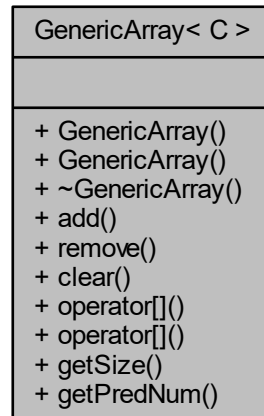


## 5.5 GenericArray< C > Class Template Reference

This [GenericArray](#) can store any type of things dynamically. Maybe I will do a static version as well. Or just implement it with a bool. Time will tell...

```
#include <GenericArray.h>
```

Collaboration diagram for GenericArray< C >:



### Public Member Functions

- [GenericArray](#) ()  
*Default Constructor.*
- [GenericArray](#) (const [GenericArray](#)< C > &GA)  
*Copy Constructor.*
- virtual [~GenericArray](#) ()  
*Destructor.*
- void [add](#) (const C &Element)  
*Adding an Element to the Array.*
- void [remove](#) (const size\_t &Index)  
*Removes an Element from the Array.*
- void [clear](#) ()  
*Clears (deletes and then reallocate) the Array.*
- C & [operator\[\]](#) (size\_t Index) const  
*Accessing Data as constant.*
- C & [operator\[\]](#) (size\_t Index)  
*Accessing Data.*
- const size\_t & [getSize](#) ()  
*Get Size of the Array.*
- template<class Pred >  
size\_t [getPredNum](#) (Pred pred)  
*Get Number of specific Items in the Array.*

### 5.5.1 Detailed Description

```
template<class C>
class GenericArray< C >
```

This [GenericArray](#) can store any type of things dynamically. Maybe I will do a static version as well. Or just implement it with a bool. Time will tell...

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 GenericArray() [1/2]

```
template<class C >
GenericArray< C >::GenericArray ( ) [inline]
```

Default Constructor.

#### 5.5.2.2 GenericArray() [2/2]

```
template<class C >
GenericArray< C >::GenericArray (
    const GenericArray< C > & GA ) [inline]
```

Copy Constructor.

#### 5.5.2.3 ~GenericArray()

```
template<class C >
virtual GenericArray< C >::~~GenericArray ( ) [inline], [virtual]
```

Destructor.

### 5.5.3 Member Function Documentation

#### 5.5.3.1 add()

```
template<class C >
void GenericArray< C >::add (
    const C & Element ) [inline]
```

Adding an Element to the Array.

Here is the caller graph for this function:



#### 5.5.3.2 clear()

```
template<class C >
void GenericArray< C >::clear ( ) [inline]
```

Clears (deletes and then reallocate) the Array.

Here is the caller graph for this function:



#### 5.5.3.3 getPredNum()

```
template<class C >
template<class Pred >
size_t GenericArray< C >::getPredNum (
    Pred pred ) [inline]
```

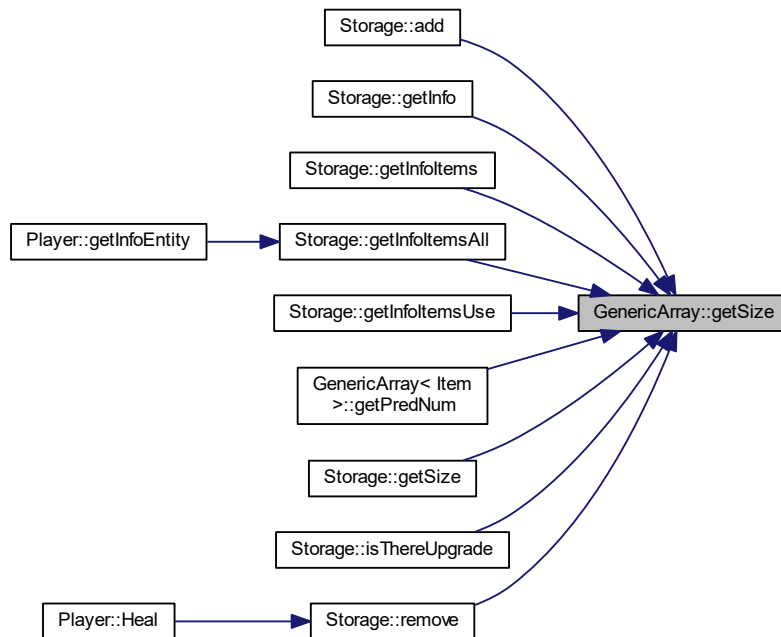
Get Number of specific Items in the Array.

#### 5.5.3.4 getSize()

```
template<class C >
const size_t& GenericArray< C >::getSize ( ) [inline]
```

Get Size of the Array.

Here is the caller graph for this function:



#### 5.5.3.5 operator[]() [1/2]

```
template<class C >
C& GenericArray< C >::operator[] (
    size_t Index ) [inline]
```

Accessing Data.

#### 5.5.3.6 operator[]() [2/2]

```
template<class C >
C& GenericArray< C >::operator[] (
    size_t Index ) const [inline]
```

Accessing Data as constant.

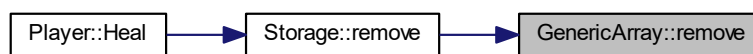


### 5.5.3.7 remove()

```
template<class C >
void GenericArray< C >::remove (
    const size_t & Index ) [inline]
```

Removes an Element from the Array.

Here is the caller graph for this function:

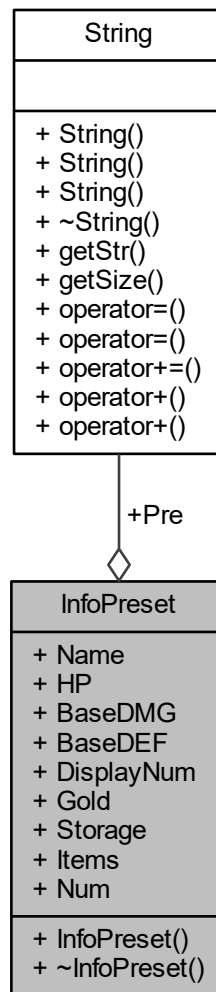


## 5.6 InfoPreset Class Reference

A preset tool for displaying dynamic information about anything.

```
#include <InfoPreset.h>
```

Collaboration diagram for InfoPreset:



## Public Member Functions

- `InfoPreset` (const char \*cPre="", bool `Name`=true, bool `HP`=true, bool `BaseDMG`=true, bool `BaseDEF`=true, bool `DisplayNum`=false, bool `Gold`=true, bool `Storage`=true, bool `Items`=true, const size\_t &dNum=0)

*Constructor.*

- virtual `~InfoPreset` ()

*Destructor.*

## Public Attributes

- bool `Name`

*Displays Name.*

- bool `HP`

- Displays HP of an [Entity](#) or Healing for a Potion.*

  - bool [BaseDMG](#)

*Displays BonusDMG and BonusDMGM of a Weapon/Shield or BaseDMG for an [Entity](#).*
- bool [BaseDEF](#)

*Displays BonusDEF and BonusDEFM of a Weapon/Shield or BaseDEF for an [Entity](#).*
- bool [DisplayNum](#)

*Displays a given number, like the index of an [Enemy](#) before its Name.*
- [String Pre](#)

*A [String](#), which is displayed before everything, it can give some context.*
- bool [Gold](#)

*Displays the [Player](#)'s Gold.*
- bool [Storage](#)

*Displays the [Player](#)'s [Storage](#).*
- bool [Items](#)

*Displays the [Player](#)'s [Storage](#)'s Items.*
- size\_t [Num](#)

*If DisplayNum is TRUE, then it displays this Number.*

### 5.6.1 Detailed Description

A preset tool for displaying dynamic information about anything.

### 5.6.2 Constructor & Destructor Documentation

#### 5.6.2.1 InfoPreset()

```
InfoPreset::InfoPreset (
    const char * cPre = "",
    bool Name = true,
    bool HP = true,
    bool BaseDMG = true,
    bool BaseDEF = true,
    bool DisplayNum = false,
    bool Gold = true,
    bool Storage = true,
    bool Items = true,
    const size_t & dNum = 0 ) [inline]
```

Constructor.

#### 5.6.2.2 ~InfoPreset()

```
virtual InfoPreset::~~InfoPreset ( ) [inline], [virtual]
```

Destructor.

## 5.6.3 Member Data Documentation

### 5.6.3.1 BaseDEF

```
bool InfoPreset::BaseDEF
```

Displays BonusDEF and BonusDEFM of a Weapon/Shield or BaseDEF for an [Entity](#).

### 5.6.3.2 BaseDMG

```
bool InfoPreset::BaseDMG
```

Displays BonusDMG and BonusDMGM of a Weapon/Shield or BaseDMG for an [Entity](#).

### 5.6.3.3 DisplayNum

```
bool InfoPreset::DisplayNum
```

Displays a given number, like the index of an [Enemy](#) before its Name.

### 5.6.3.4 Gold

```
bool InfoPreset::Gold
```

Displays the [Player](#)'s Gold.

### 5.6.3.5 HP

```
bool InfoPreset::HP
```

Displays HP of an [Entity](#) or Healing for a Potion.

#### 5.6.3.6 Items

```
bool InfoPreset::Items
```

Displays the [Player's Storage](#)'s Items.

#### 5.6.3.7 Name

```
bool InfoPreset::Name
```

Displays Name.

#### 5.6.3.8 Num

```
size_t InfoPreset::Num
```

If DisplayNum is TRUE, then it displays this Number.

#### 5.6.3.9 Pre

```
String InfoPreset::Pre
```

A [String](#), which is displayed before everything, it can give some context.

#### 5.6.3.10 Storage

```
bool InfoPreset::Storage
```

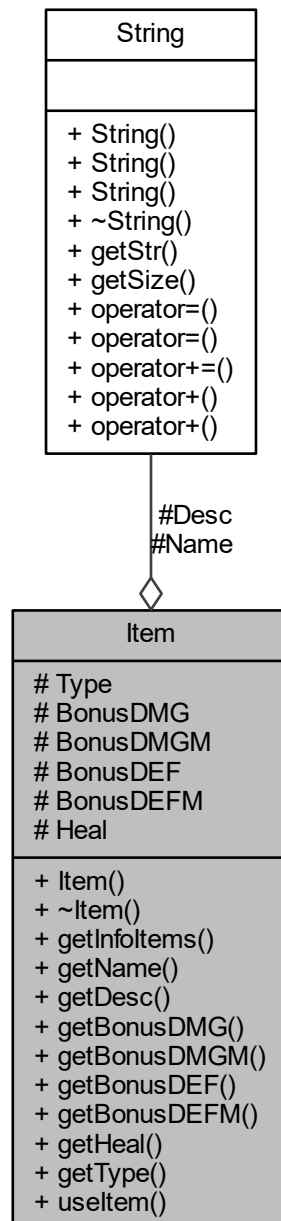
Displays the [Player's Storage](#).

## 5.7 Item Class Reference

An [Item](#) can be stored in a storage and also has a type. Depending on the type, an [Item](#) can heal the player or damage an enemy, etc.

```
#include <Item.h>
```

Collaboration diagram for Item:



## Public Member Functions

- [Item](#) (const ItemType [Type](#)=ItemType::WeaponShield, const char \*[Name](#)="DisItem", const char \*[Desc](#)="This is an Item.", const double &[BonusDMG](#)=0, const double &[BonusDMGM](#)=1, const double &[BonusDEF](#)=0, const double &[BonusDEFM](#)=1, const double &[Heal](#)=0)  
*Default Constructor.*
- virtual [~Item](#) ()  
*Destructor.*
- virtual std::ostream & [getInfoItems](#) ([InfoPreset](#) Preset=[InfoPreset](#)(), std::ostream &os=std::cout)  
*Writes out the [Item](#)'s info to an output.*
- const char \* [getName](#) ()  
*Gets Name.*
- const char \* [getDesc](#) ()  
*Gets Description.*
- double [getBonusDMG](#) ()  
*Gets Additive Bonus Damage of [Item](#).*
- double [getBonusDMGM](#) ()  
*Gets Multiplied Bonus Damage of [Item](#).*
- double [getBonusDEF](#) ()  
*Gets Additive Bonus Defense of [Item](#).*
- double [getBonusDEFM](#) ()  
*Gets Multiplied Bonus Defense of [Item](#).*
- double [getHeal](#) ()  
*Gets Healing of [Item](#).*
- ItemType [getType](#) () const  
*Gets Type of [Item](#).*
- UseCases [useItem](#) ()  
*Use an [Item](#).*

## Protected Attributes

- ItemType [Type](#)  
*Type of the [Item](#).*
- String [Name](#)  
*Name of the [Item](#).*
- String [Desc](#)  
*Description of the [Item](#).*
- double [BonusDMG](#)  
*Additive Bonus Damage of [Item](#).*
- double [BonusDMGM](#)  
*Multiplied Bonus Damage of [Item](#).*
- double [BonusDEF](#)  
*Additive Bonus Defense of [Item](#).*
- double [BonusDEFM](#)  
*Multiplied Bonus Defense of [Item](#).*
- double [Heal](#)  
*Healing of [Item](#).*

### 5.7.1 Detailed Description

An [Item](#) can be stored in a storage and also has a type. Depending on the type, an [Item](#) can heal the player or damage an enemy, etc.

### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 Item()

```
Item::Item (
    const ItemType Type = ItemType::WeaponShield,
    const char * Name = "DisItem",
    const char * Desc = "This is an Item.",
    const double & BonusDMG = 0,
    const double & BonusDMGM = 1,
    const double & BonusDEF = 0,
    const double & BonusDEFM = 1,
    const double & Heal = 0 )
```

Default Constructor.

#### 5.7.2.2 ~Item()

```
Item::~Item ( ) [virtual]
```

Destructor.

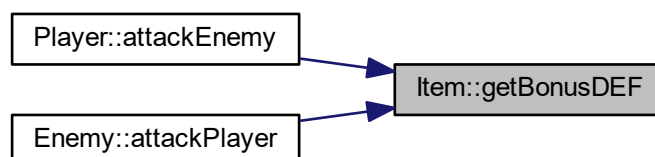
### 5.7.3 Member Function Documentation

#### 5.7.3.1 getBonusDEF()

```
double Item::getBonusDEF ( )
```

Gets Additive Bonus Defense of [Item](#).

Here is the caller graph for this function:



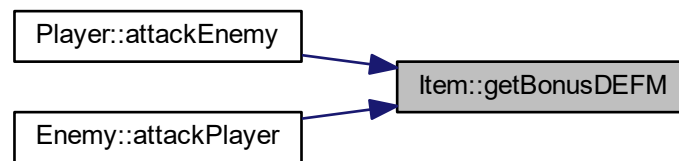


### 5.7.3.2 getBonusDEFM()

```
double Item::getBonusDEFM ( )
```

Gets Multiplied Bonus Defense of [Item](#).

Here is the caller graph for this function:



### 5.7.3.3 getBonusDMG()

```
double Item::getBonusDMG ( )
```

Gets Additive Bonus Damage of [Item](#).

Here is the caller graph for this function:



#### 5.7.3.4 getBonusDMGM()

```
double Item::getBonusDMGM ( )
```

Gets Multiplied Bonus Damage of [Item](#).

Here is the caller graph for this function:



#### 5.7.3.5 getDesc()

```
const char * Item::getDesc ( )
```

Gets Description.

Here is the call graph for this function:



#### 5.7.3.6 getHeal()

```
double Item::getHeal ( )
```

Gets Healing of [Item](#).

### 5.7.3.7 getInfoItems()

```
std::ostream & Item::getInfoItems (
    InfoPreset Preset = InfoPreset(),
    std::ostream & os = std::cout ) [virtual]
```

Writes out the [Item](#)'s info to an output.

### 5.7.3.8 getName()

```
const char * Item::getName ( )
```

Gets Name.

Here is the call graph for this function:



Here is the caller graph for this function:

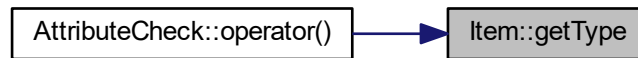


### 5.7.3.9 getType()

```
ItemType Item::getType ( ) const
```

Gets Type of [Item](#).

Here is the caller graph for this function:



#### 5.7.3.10 useItem()

UseCases Item::useItem ( )

Use an [Item](#).

Use a Weapon/Shield

Use a Healing Potion

### 5.7.4 Member Data Documentation

#### 5.7.4.1 BonusDEF

```
double Item::BonusDEF [protected]
```

Additive Bonus Defense of [Item](#).

#### 5.7.4.2 BonusDEFM

```
double Item::BonusDEFM [protected]
```

Multiplied Bonus Defense of [Item](#).

#### 5.7.4.3 BonusDMG

```
double Item::BonusDMG [protected]
```

Additive Bonus Damage of [Item](#).

#### 5.7.4.4 BonusDMGM

```
double Item::BonusDMGM [protected]
```

Multiplied Bonus Damage of [Item](#).

#### 5.7.4.5 Desc

```
String Item::Desc [protected]
```

Description of the [Item](#).

#### 5.7.4.6 Heal

```
double Item::Heal [protected]
```

Healing of [Item](#).

#### 5.7.4.7 Name

```
String Item::Name [protected]
```

Name of the [Item](#).

#### 5.7.4.8 Type

```
ItemType Item::Type [protected]
```

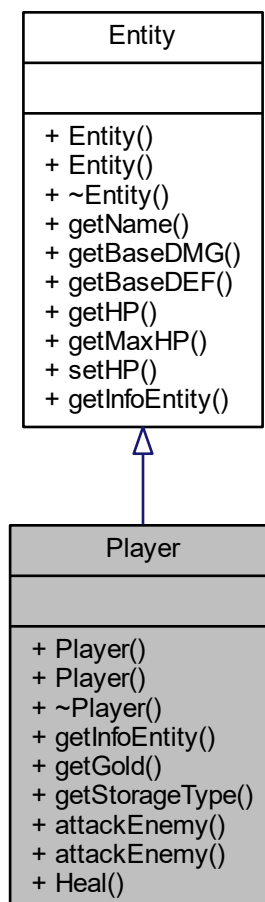
Type of the [Item](#).

## 5.8 Player Class Reference

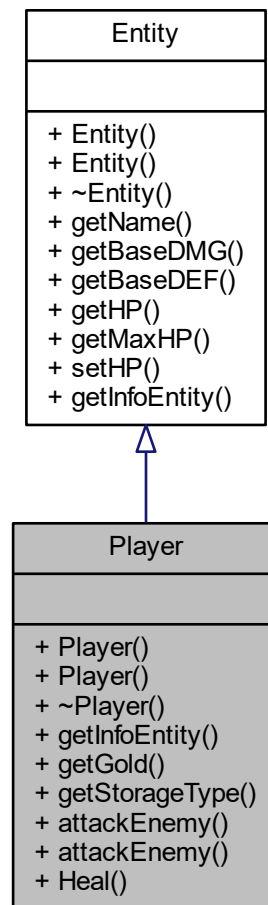
[Player](#) has Gold and a [Storage](#) in which it can hold some [Item](#). The [Player](#) will be able to spend Gold in the Shop.

```
#include <Player.h>
```

Inheritance diagram for Player:



Collaboration diagram for Player:



## Public Member Functions

- **Player** (const char \*Name, double HP=100, double MaxHP=100, double BaseDMG=5, double BaseDEF=5, size\_t Gold=0, **Storage** StorageType=**Storage**())  
*Constructor.*
- **Player** (double HP=100, double MaxHP=100, double BaseDMG=5, double BaseDEF=5, size\_t Gold=0)  
*Default Constructor.*
- virtual **~Player** ()  
*Destructor.*
- std::ostream & **getInfoEntity** (**InfoPreset** Preset=**InfoPreset**(), std::ostream &os=std::cout)  
*Writes out the **Player**'s info to an output.*
- size\_t **getGold** ()  
*Gets **Player**'s amount of Gold.*
- **Storage** & **getStorageType** ()  
*Gets **Player**'s type of **Storage**.*
- void **attackEnemy** (**Enemy** &e)

- *Attack an [Enemy](#).*  
• void [attackEnemy](#) ([Enemy](#) &e, [Item](#) &WS)  
*Attack an [Enemy](#) with a Weapon/Shield.*
- void [Heal](#) (const size\_t Index)  
*Healing with an [Item](#).*

### 5.8.1 Detailed Description

[Player](#) has Gold and a [Storage](#) in which it can hold some [Item](#). The [Player](#) will be able to spend Gold in the Shop.

### 5.8.2 Constructor & Destructor Documentation

#### 5.8.2.1 [Player\(\)](#) [1/2]

```
Player::Player (  
    const char * Name,  
    double HP = 100,  
    double MaxHP = 100,  
    double BaseDMG = 5,  
    double BaseDEF = 5,  
    size_t Gold = 0,  
    Storage StorageType = Storage() )
```

Constructor.

#### 5.8.2.2 [Player\(\)](#) [2/2]

```
Player::Player (  
    double HP = 100,  
    double MaxHP = 100,  
    double BaseDMG = 5,  
    double BaseDEF = 5,  
    size_t Gold = 0 )
```

Default Constructor.

#### 5.8.2.3 [~Player\(\)](#)

```
Player::~~Player ( ) [virtual]
```

Destructor.



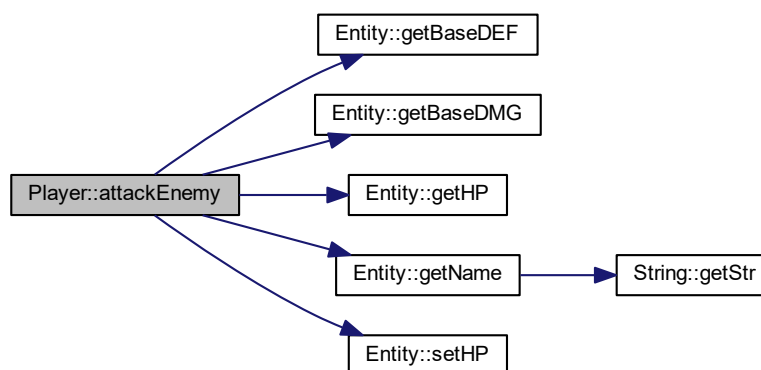
### 5.8.3 Member Function Documentation

#### 5.8.3.1 attackEnemy() [1/2]

```
void Player::attackEnemy (  
    Enemy & e )
```

Attack an [Enemy](#).

Here is the call graph for this function:

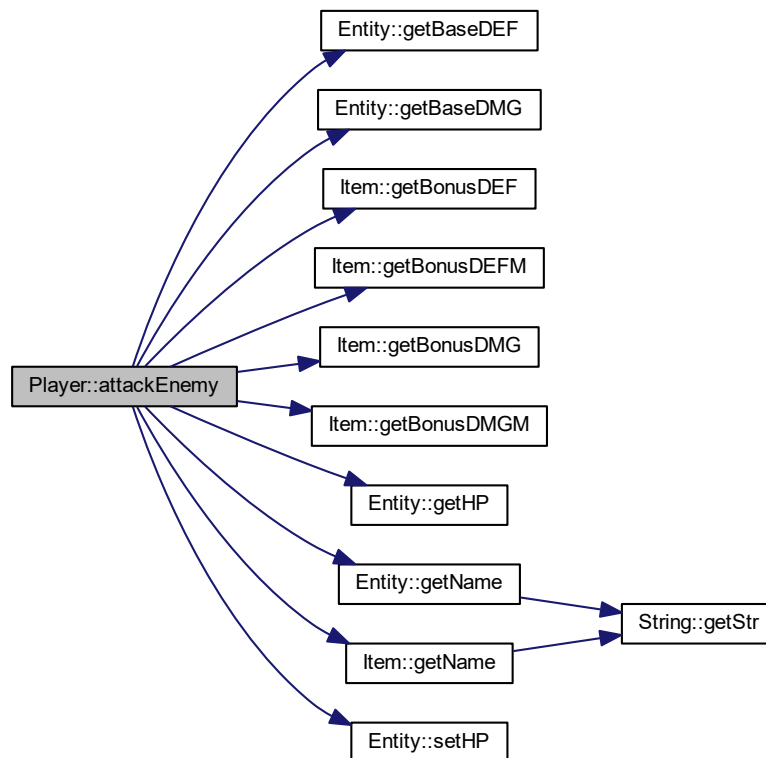


#### 5.8.3.2 attackEnemy() [2/2]

```
void Player::attackEnemy (  
    Enemy & e,  
    Item & WS )
```

Attack an [Enemy](#) with a Weapon/Shield.

Here is the call graph for this function:



### 5.8.3.3 getGold()

```
size_t Player::getGold ( )
```

Gets `Player`'s amount of Gold.

Here is the caller graph for this function:



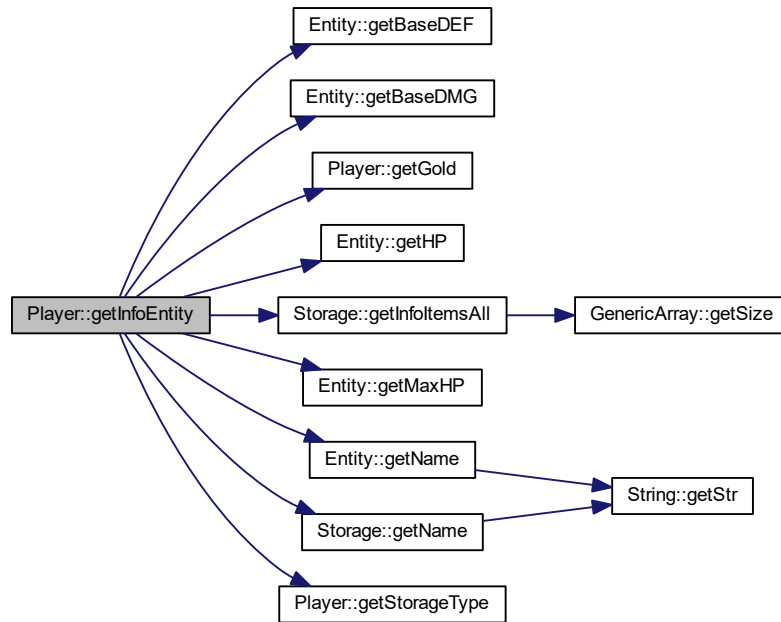
#### 5.8.3.4 getInfoEntity()

```
std::ostream & Player::getInfoEntity (
    InfoPreset Preset = InfoPreset(),
    std::ostream & os = std::cout ) [virtual]
```

Writes out the [Player](#)'s info to an output.

Implements [Entity](#).

Here is the call graph for this function:

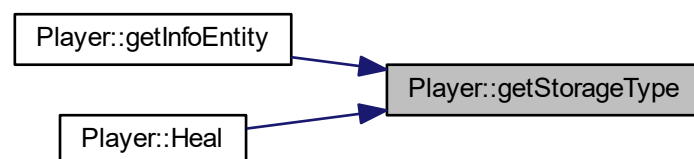


#### 5.8.3.5 getStorageType()

```
Storage & Player::getStorageType ( )
```

Gets [Player](#)'s type of [Storage](#).

Here is the caller graph for this function:

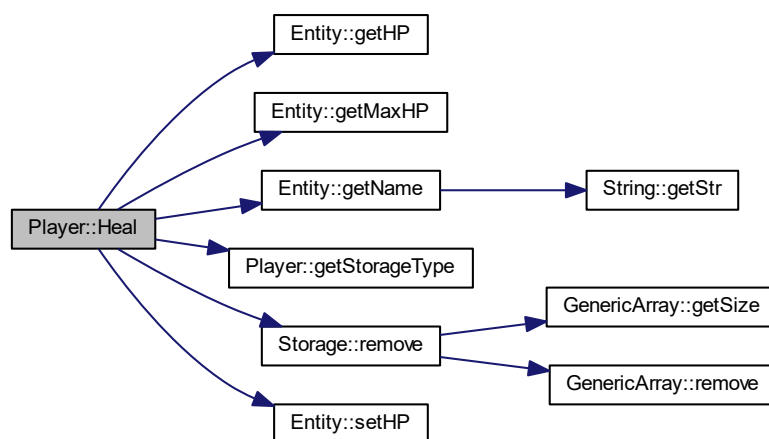


### 5.8.3.6 Heal()

```
void Player::Heal (
    const size_t Index )
```

Healing with an [Item](#).

Here is the call graph for this function:

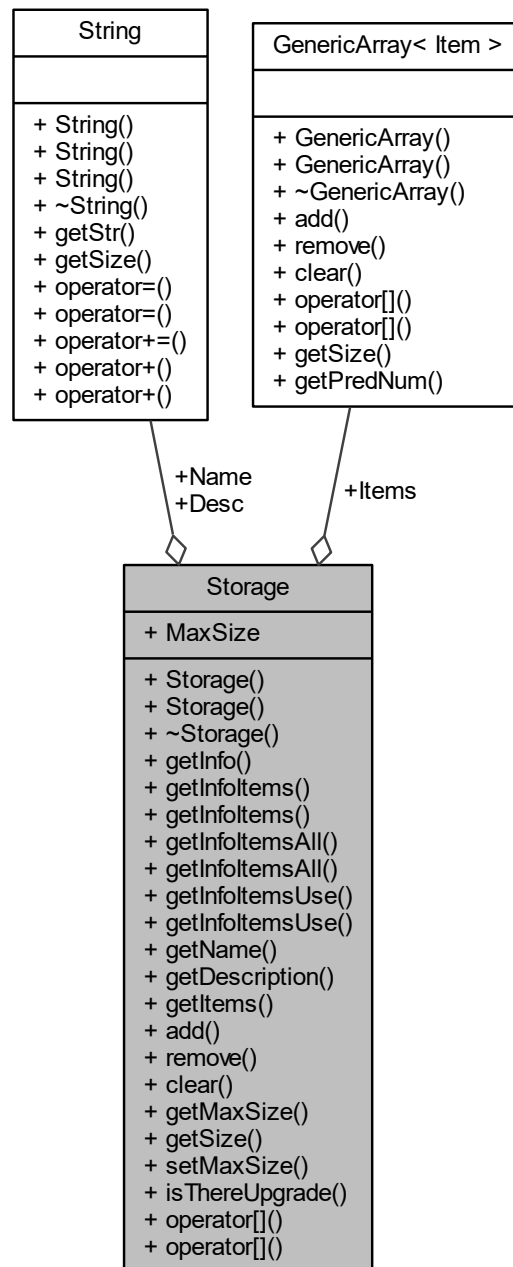


## 5.9 Storage Class Reference

A [Storage](#) is able to store Items, but to a limited extent (`MaxSize`). Also has a Name and a Description.

```
#include <Storage.h>
```

Collaboration diagram for Storage:



## Public Member Functions

- **Storage** (const char \***Name**="BakPak", const char \***Desc**="This is a BakPak.", size\_t **MaxSize**=5)  
*Default Constructor.*
- **Storage** (const **Storage** &**S**)  
*Copy Constructor.*
- virtual **~Storage** ()

*Destructor.*

- `std::ostream & getInfo (std::ostream &os=std::cout)`  
Writes out the *Storage*'s info to the console.
- `std::ostream & getInfoItems (InfoPreset Preset=InfoPreset(), std::ostream &os=std::cout)`  
Writes out the *Item*'s infos in the *Storage* to the console.
- `std::ostream & getInfoItems (const ItemType &Type, InfoPreset Preset=InfoPreset(), std::ostream &os=std::cout)`  
Writes out a specific *Item*'s infos in the *Storage* to the console.
- `std::ostream & getInfoItemsAll (std::ostream &os=std::cout)`  
Writes out the *Item*'s infos in the *Storage* to the console.
- `std::ostream & getInfoItemsAll (const ItemType &Type, std::ostream &os=std::cout)`  
Writes out a specific *Item*'s infos in the *Storage* to the console.
- `virtual std::ostream & getInfoItemsUse (std::ostream &os=std::cout)`  
Writes out the *Item*'s infos in the *Storage* to the console.
- `virtual std::ostream & getInfoItemsUse (const ItemType &Type, std::ostream &os=std::cout)`  
Writes out a specific *Item*'s infos in the *Storage* to the console.
- `const char * getName ()`  
Gets Name.
- `const char * getDescription ()`  
Gets Description.
- `GenericArray< Item > getItems () const`  
Gets the *Item*'s array, though cannot modify it.
- `void add (const Item &Element)`  
Adds an *Item* to the *Storage*.
- `void remove (const size_t &Index)`  
Removes an *Item* from the *Storage* according to its index.
- `void clear ()`  
Clears the *Storage*.
- `size_t getMaxSize ()`  
Gets size of *Storage*.
- `size_t getSize ()`  
Gets number of Items in the of *Storage*.
- `void setMaxSize (const size_t &SizeC)`  
Sets size of *Storage*.
- `bool isThereUpgrade ()`  
Returns if there are upgrades in the inventory.
- `Item & operator[] (size_t Index) const`  
Accessing an *Item* as constant.
- `Item & operator[] (size_t Index)`  
Accessing an *Item*.

## Public Attributes

- `String Name`  
Name of the *Storage*.
- `String Desc`  
Description of the *Storage*.
- `size_t MaxSize`  
Size of *Storage*, how many *Item* it can hold.
- `GenericArray< Item > Items`  
Items in the *Storage*.

## 5.9.1 Detailed Description

A [Storage](#) is able to store Items, but to a limited extent (MaxSize). Also has a Name and a Description.

## 5.9.2 Constructor & Destructor Documentation

### 5.9.2.1 Storage() [1/2]

```
Storage::Storage (
    const char * Name = "BakPak",
    const char * Desc = "This is a BakPak.",
    size_t MaxSize = 5 )
```

Default Constructor.

### 5.9.2.2 Storage() [2/2]

```
Storage::Storage (
    const Storage & S )
```

Copy Constructor.

### 5.9.2.3 ~Storage()

```
Storage::~Storage ( ) [virtual]
```

Destructor.

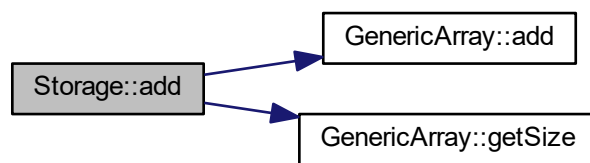
## 5.9.3 Member Function Documentation

### 5.9.3.1 add()

```
void Storage::add (
    const Item & Element )
```

Adds an [Item](#) to the [Storage](#).

Here is the call graph for this function:



### 5.9.3.2 clear()

```
void Storage::clear ( )
```

Clears the [Storage](#).

Here is the call graph for this function:



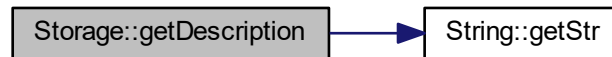


### 5.9.3.3 getDescription()

```
const char * Storage::getDescription ( )
```

Gets Description.

Here is the call graph for this function:



### 5.9.3.4 getInfo()

```
std::ostream & Storage::getInfo (
    std::ostream & os = std::cout )
```

Writes out the [Storage](#)'s info to the console.

Here is the call graph for this function:



### 5.9.3.5 getInfoItems() [1/2]

```
std::ostream & Storage::getInfoItems (
    const ItemType & Type,
    InfoPreset Preset = InfoPreset(),
    std::ostream & os = std::cout )
```

Writes out a specific [Item](#)'s infos in the [Storage](#) to the console.

Here is the call graph for this function:



#### 5.9.3.6 `getInfoItems()` [2/2]

```
std::ostream & Storage::getInfoItems (
    InfoPreset Preset = InfoPreset(),
    std::ostream & os = std::cout )
```

Writes out the `Item`'s infos in the `Storage` to the console.

Here is the call graph for this function:



#### 5.9.3.7 `getInfoItemsAll()` [1/2]

```
std::ostream & Storage::getInfoItemsAll (
    const ItemType & Type,
    std::ostream & os = std::cout )
```

Writes out a specific `Item`'s infos in the `Storage` to the console.

Here is the call graph for this function:



### 5.9.3.8 getInfoItemsAll() [2/2]

```
std::ostream & Storage::getInfoItemsAll (
    std::ostream & os = std::cout )
```

Writes out the [Item](#)'s infos in the [Storage](#) to the console.

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.9.3.9 getInfoItemsUse() [1/2]

```
std::ostream & Storage::getInfoItemsUse (
    const ItemType & Type,
    std::ostream & os = std::cout ) [virtual]
```

Writes out a specific [Item](#)'s infos in the [Storage](#) to the console.

Here is the call graph for this function:



#### 5.9.3.10 `getInfoItemsUse()` [2/2]

```
std::ostream & Storage::getInfoItemsUse (
    std::ostream & os = std::cout ) [virtual]
```

Writes out the [Item](#)'s infos in the [Storage](#) to the console.

Here is the call graph for this function:



#### 5.9.3.11 `getItems()`

```
GenericArray< Item > Storage::getItems ( ) const
```

Gets the [Item](#)'s array, though cannot modify it.

#### 5.9.3.12 `getMaxSize()`

```
size_t Storage::getMaxSize ( )
```

Gets size of [Storage](#).

#### 5.9.3.13 `getName()`

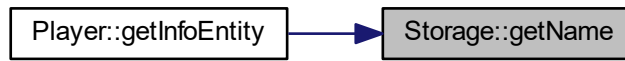
```
const char * Storage::getName ( )
```

Gets Name.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.9.3.14 `getSize()`

```
size_t Storage::getSize ( )
```

Gets number of Items in the of [Storage](#).

Here is the call graph for this function:



#### 5.9.3.15 `isThereUpgrade()`

```
bool Storage::isThereUpgrade ( )
```

Returns if there are upgrades in the inventory.

Here is the call graph for this function:



#### 5.9.3.16 operator[]() [1/2]

```
Item& Storage::operator[] (
    size_t Index ) [inline]
```

Accessing an [Item](#).

#### 5.9.3.17 operator[]() [2/2]

```
Item& Storage::operator[] (
    size_t Index ) const [inline]
```

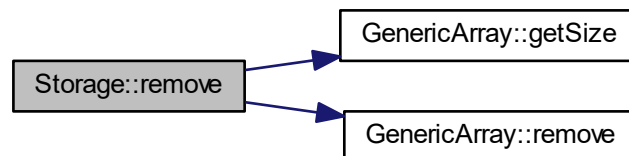
Accessing an [Item](#) as constant.

#### 5.9.3.18 remove()

```
void Storage::remove (
    const size_t & Index )
```

Removes an [Item](#) from the [Storage](#) according to its index.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.9.3.19 setMaxSize()

```
void Storage::setMaxSize (
    const size_t & SizeC )
```

Sets size of [Storage](#).

### 5.9.4 Member Data Documentation

#### 5.9.4.1 Desc

```
String Storage::Desc
```

Description of the [Storage](#).

#### 5.9.4.2 Items

```
GenericArray<Item> Storage::Items
```

Items in the [Storage](#).

#### 5.9.4.3 MaxSize

```
size_t Storage::MaxSize
```

Size of [Storage](#), how many [Item](#) it can hold.

#### 5.9.4.4 Name

```
String Storage::Name
```

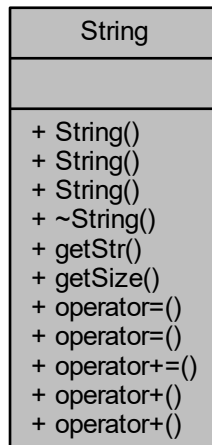
Name of the [Storage](#).

## 5.10 String Class Reference

Seperate [String](#) class, because STL are not allowed...

```
#include <String.h>
```

Collaboration diagram for String:



### Public Member Functions

- [String](#) ([String](#) const &str)  
*Copy Constructor.*
- [String](#) (const char \*str="")  
*Default Constructor from const char\*.*
- [String](#) (const char c)  
*Constructor from char.*
- virtual [~String](#) ()  
*Destructor.*
- const char \* [getStr](#) ()  
*Return the content of the [String](#).*
- const size\_t [getSize](#) ()  
*Return the length of the [String](#).*
- [String](#) & [operator=](#) (const [String](#) &rhs\_s)  
*Assign the [String](#)'s data to the [String](#).*
- [String](#) & [operator=](#) (const char \*rhs\_s)  
*Assign const char\*'s data to the [String](#).*
- [String](#) & [operator+=](#) (const [String](#) &rhs\_s)  
*Adding [String](#) to the original.*
- [String](#) [operator+](#) (const [String](#) &rhs\_s) const  
*Adding 2 Strings together (returns constant)*
- [String](#) [operator+](#) (char rhs\_c) const  
*Adding a char to the [String](#) (returns constant)*



### 5.10.1 Detailed Description

Seperate [String](#) class, because STL are not allowed...

### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 String() [1/3]

```
String::String (  
    String const & str )
```

Copy Constructor.

Here is the caller graph for this function:



#### 5.10.2.2 String() [2/3]

```
String::String (  
    const char * str = "" )
```

Default Constructor from const char\*.

#### 5.10.2.3 String() [3/3]

```
String::String (  
    const char c )
```

Constructor from char.

#### 5.10.2.4 ~String()

```
virtual String::~~String ( ) [inline], [virtual]
```

Destructor.

### 5.10.3 Member Function Documentation

#### 5.10.3.1 getSize()

```
const size_t String::getSize ( ) [inline]
```

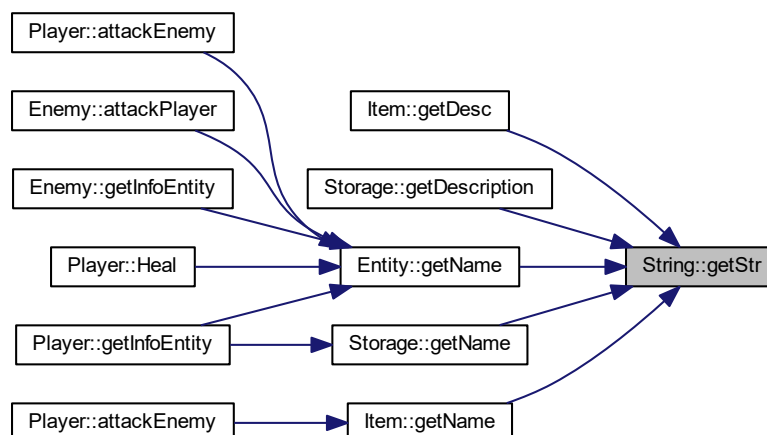
Return the length of the [String](#).

#### 5.10.3.2 getStr()

```
const char* String::getStr ( ) [inline]
```

Return the content of the [String](#).

Here is the caller graph for this function:



### 5.10.3.3 operator+() [1/2]

```
String String::operator+ (
    char rhs_c ) const [inline]
```

Adding a char to the [String](#) (returns constant)

Here is the call graph for this function:



### 5.10.3.4 operator+() [2/2]

```
String String::operator+ (
    const String & rhs_s ) const
```

Adding 2 Strings together (returns constant)

### 5.10.3.5 operator+=()

```
String& String::operator+= (
    const String & rhs_s ) [inline]
```

Adding [String](#) to the original.

### 5.10.3.6 operator=() [1/2]

```
String & String::operator= (
    const char * rhs_s )
```

Assign const char\*'s data to the [String](#).

### 5.10.3.7 operator=() [2/2]

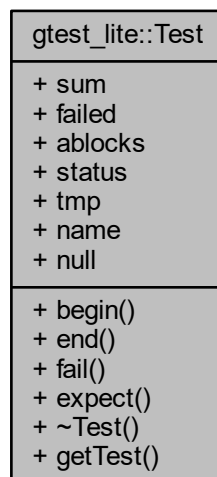
```
String & String::operator= (
    const String & rhs_s )
```

Assign the [String](#)'s data to the [String](#).

## 5.11 gtest\_lite::Test Struct Reference

```
#include <gtest_lite.h>
```

Collaboration diagram for gtest\_lite::Test:



### Public Member Functions

- void [begin](#) (const char \*n)  
*Teszt kezdete.*
- std::ostream & [end](#) (bool memchk=false)  
*Teszt vége.*
- bool [fail](#) ()
- std::ostream & [expect](#) (bool st, const char \*file, int line, const char \*expr, bool pr=false)  
*Eredményt adminisztráló tagfüggvény True a jó eset.*
- [~Test](#) ()  
*Destruktor.*

### Static Public Member Functions

- static [Test](#) & [getTest](#) ()

## Public Attributes

- int `sum`  
*tesztek számlálója*
- int `failed`  
*hibás tesztek*
- int `ablocks`  
*allokált blokkok száma*
- bool `status`  
*éppen futó teszt státusza.*
- bool `tmp`  
*temp a kivételkezeléshez;*
- std::string `name`  
*éppen futó teszt neve.*
- std::fstream `null`  
*nyelő, ha nem kell kiírni semmit*

### 5.11.1 Detailed Description

Tesztek állapotát tároló osztály. Egyetlen egy statikus példány keletkezik, aminek a destruktora a futás végén hívódik meg.

### 5.11.2 Constructor & Destructor Documentation

#### 5.11.2.1 ~Test()

```
gtest_lite::Test::~~Test ( ) [inline]
```

Destruktor.

### 5.11.3 Member Function Documentation

#### 5.11.3.1 begin()

```
void gtest_lite::Test::begin (
    const char * n ) [inline]
```

Teszt kezdete.

### 5.11.3.2 end()

```
std::ostream& gtest_lite::Test::end (
    bool memchk = false )    [inline]
```

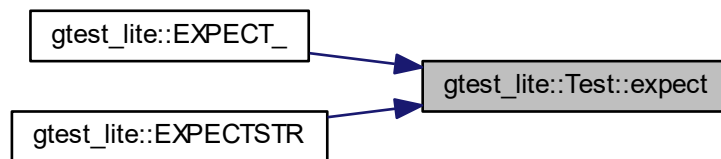
Teszt vége.

### 5.11.3.3 expect()

```
std::ostream& gtest_lite::Test::expect (
    bool st,
    const char * file,
    int line,
    const char * expr,
    bool pr = false )    [inline]
```

Eredményt adminisztráló tagfüggvény True a jó eset.

Here is the caller graph for this function:



### 5.11.3.4 fail()

```
bool gtest_lite::Test::fail ( )    [inline]
```

### 5.11.3.5 getTest()

```
static Test& gtest_lite::Test::getTest ( )    [inline], [static]
```

< egyedüli (singleton) példány

## 5.11.4 Member Data Documentation

### 5.11.4.1 ablocks

```
int gtest_lite::Test::ablocks
```

allokált blokkok száma

### 5.11.4.2 failed

```
int gtest_lite::Test::failed
```

hibás tesztek

### 5.11.4.3 name

```
std::string gtest_lite::Test::name
```

éppen futó teszt neve.

### 5.11.4.4 null

```
std::fstream gtest_lite::Test::null
```

nyelő, ha nem kell kiírni semmit

### 5.11.4.5 status

```
bool gtest_lite::Test::status
```

éppen futó teszt státusza.

### 5.11.4.6 sum

```
int gtest_lite::Test::sum
```

tesztek számlálója

### 5.11.4.7 tmp

```
bool gtest_lite::Test::tmp
```

temp a kivételkezeléshez;





# Index

- [\\_Is\\_Types< F, T >, 13](#)
  - [convertable, 14](#)
    - [f, 14](#)
- [~Enemy](#)
  - [Enemy, 19](#)
- [~Entity](#)
  - [Entity, 24](#)
- [~GenericArray](#)
  - [GenericArray< C >, 30](#)
- [~InfoPreset](#)
  - [InfoPreset, 35](#)
- [~Item](#)
  - [Item, 40](#)
- [~Player](#)
  - [Player, 48](#)
- [~Storage](#)
  - [Storage, 55](#)
- [~String](#)
  - [String, 65](#)
- [~Test](#)
  - [gtest\\_lite::Test, 69](#)
- [ablocks](#)
  - [gtest\\_lite::Test, 71](#)
- [add](#)
  - [GenericArray< C >, 30](#)
  - [Storage, 55](#)
- [almostEQ](#)
  - [gtest\\_lite, 8](#)
- [attackEnemy](#)
  - [Player, 49](#)
- [attackPlayer](#)
  - [Enemy, 19](#)
- [AttributeCheck](#)
  - [AttributeCheck< C >, 15](#)
- [AttributeCheck< C >, 15](#)
  - [AttributeCheck, 15](#)
  - [operator\(\), 16](#)
- [BaseDEF](#)
  - [InfoPreset, 36](#)
- [BaseDMG](#)
  - [InfoPreset, 36](#)
- [begin](#)
  - [gtest\\_lite::Test, 69](#)
- [BonusDEF](#)
  - [Item, 44](#)
- [BonusDEFM](#)
  - [Item, 44](#)
- [BonusDMG](#)
  - [Item, 44](#)
- [BonusDMGM](#)
  - [Item, 44](#)
- [clear](#)
  - [GenericArray< C >, 31](#)
  - [Storage, 56](#)
- [convertable](#)
  - [\\_Is\\_Types< F, T >, 14](#)
- [Desc](#)
  - [Item, 45](#)
  - [Storage, 63](#)
- [DisplayNum](#)
  - [InfoPreset, 36](#)
- [end](#)
  - [gtest\\_lite::Test, 69](#)
- [Enemy, 16](#)
  - [~Enemy, 19](#)
  - [attackPlayer, 19](#)
  - [Enemy, 19](#)
  - [getInfoEntity, 20](#)
- [Entity, 21](#)
  - [~Entity, 24](#)
  - [Entity, 24](#)
  - [getBaseDEF, 24](#)
  - [getBaseDMG, 25](#)
  - [getHP, 25](#)
  - [getInfoEntity, 26](#)
  - [getMaxHP, 26](#)
  - [getName, 27](#)
  - [setHP, 28](#)
- [eq](#)
  - [gtest\\_lite, 8](#)
- [eqstr](#)
  - [gtest\\_lite, 8](#)
- [expect](#)
  - [gtest\\_lite::Test, 70](#)
- [EXPECT\\_](#)
  - [gtest\\_lite, 8, 9](#)
- [EXPECTSTR](#)
  - [gtest\\_lite, 9](#)
- [f](#)
  - [\\_Is\\_Types< F, T >, 14](#)
- [fail](#)
  - [gtest\\_lite::Test, 70](#)
- [failed](#)
  - [gtest\\_lite::Test, 71](#)

- ge
  - gtest\_lite, 10
- GenericArray
  - GenericArray< C >, 30
- GenericArray< C >, 29
  - ~GenericArray, 30
  - add, 30
  - clear, 31
  - GenericArray, 30
  - getPredNum, 31
  - getSize, 31
  - operator[], 32
  - remove, 32
- getBaseDEF
  - Entity, 24
- getBaseDMG
  - Entity, 25
- getBonusDEF
  - Item, 40
- getBonusDEFM
  - Item, 40
- getBonusDMG
  - Item, 41
- getBonusDMGM
  - Item, 41
- getDesc
  - Item, 42
- getDescription
  - Storage, 56
- getGold
  - Player, 50
- getHeal
  - Item, 42
- getHP
  - Entity, 25
- getInfo
  - Storage, 57
- getInfoEntity
  - Enemy, 20
  - Entity, 26
  - Player, 50
- getInfoItems
  - Item, 42
  - Storage, 57, 58
- getInfoItemsAll
  - Storage, 58
- getInfoItemsUse
  - Storage, 59
- getItems
  - Storage, 60
- getMaxHP
  - Entity, 26
- getMaxSize
  - Storage, 60
- getName
  - Entity, 27
  - Item, 43
  - Storage, 60
- getPredNum
  - GenericArray< C >, 31
- getSize
  - GenericArray< C >, 31
  - Storage, 61
  - String, 66
- getStorageType
  - Player, 51
- getStr
  - String, 66
- getTest
  - gtest\_lite::Test, 70
- getType
  - Item, 43
- Gold
  - InfoPreset, 36
- gt
  - gtest\_lite, 10
- gtest\_lite, 7
  - almostEQ, 8
  - eq, 8
  - eqstr, 8
  - EXPECT\_, 8, 9
  - EXPECTSTR, 9
  - ge, 10
  - gt, 10
  - le, 10
  - lt, 10
  - ne, 10
  - nestr, 11
- gtest\_lite::Test, 68
  - ~Test, 69
  - ablocks, 71
  - begin, 69
  - end, 69
  - expect, 70
  - fail, 70
  - failed, 71
  - getTest, 70
  - name, 71
  - null, 71
  - status, 71
  - sum, 71
  - tmp, 71
- Heal
  - Item, 45
  - Player, 52
- HP
  - InfoPreset, 36
- InfoPreset, 33
  - ~InfoPreset, 35
  - BaseDEF, 36
  - BaseDMG, 36
  - DisplayNum, 36
  - Gold, 36
  - HP, 36
  - InfoPreset, 35

- Items, [36](#)
- Name, [37](#)
- Num, [37](#)
- Pre, [37](#)
- Storage, [37](#)
- isThereUpgrade
  - Storage, [61](#)
- Item, [38](#)
  - ~Item, [40](#)
  - BonusDEF, [44](#)
  - BonusDEFM, [44](#)
  - BonusDMG, [44](#)
  - BonusDMGM, [44](#)
  - Desc, [45](#)
  - getBonusDEF, [40](#)
  - getBonusDEFM, [40](#)
  - getBonusDMG, [41](#)
  - getBonusDMGM, [41](#)
  - getDesc, [42](#)
  - getHeal, [42](#)
  - getInfolItems, [42](#)
  - getName, [43](#)
  - getType, [43](#)
  - Heal, [45](#)
  - Item, [40](#)
  - Name, [45](#)
  - Type, [45](#)
  - useItem, [44](#)
- Items
  - InfoPreset, [36](#)
  - Storage, [63](#)
- le
  - gtest\_lite, [10](#)
- lt
  - gtest\_lite, [10](#)
- MaxSize
  - Storage, [63](#)
- Name
  - InfoPreset, [37](#)
  - Item, [45](#)
  - Storage, [63](#)
- name
  - gtest\_lite::Test, [71](#)
- ne
  - gtest\_lite, [10](#)
- nestr
  - gtest\_lite, [11](#)
- null
  - gtest\_lite::Test, [71](#)
- Num
  - InfoPreset, [37](#)
- operator()
  - AttributeCheck< C >, [16](#)
- operator+
  - String, [66](#), [67](#)
- operator+=
  - String, [67](#)
- operator=
  - String, [67](#)
- operator[]
  - GenericArray< C >, [32](#)
  - Storage, [61](#), [62](#)
- Player, [46](#)
  - ~Player, [48](#)
  - attackEnemy, [49](#)
  - getGold, [50](#)
  - getInfoEntity, [50](#)
  - getStorageType, [51](#)
  - Heal, [52](#)
  - Player, [48](#)
- Pre
  - InfoPreset, [37](#)
- remove
  - GenericArray< C >, [32](#)
  - Storage, [62](#)
- setHP
  - Entity, [28](#)
- setMaxSize
  - Storage, [62](#)
- status
  - gtest\_lite::Test, [71](#)
- Storage, [52](#)
  - ~Storage, [55](#)
  - add, [55](#)
  - clear, [56](#)
  - Desc, [63](#)
  - getDescription, [56](#)
  - getInfo, [57](#)
  - getInfolItems, [57](#), [58](#)
  - getInfolItemsAll, [58](#)
  - getInfolItemsUse, [59](#)
  - getItems, [60](#)
  - getMaxSize, [60](#)
  - getName, [60](#)
  - getSize, [61](#)
  - InfoPreset, [37](#)
  - isThereUpgrade, [61](#)
  - Items, [63](#)
  - MaxSize, [63](#)
  - Name, [63](#)
  - operator[], [61](#), [62](#)
  - remove, [62](#)
  - setMaxSize, [62](#)
  - Storage, [55](#)
- String, [64](#)
  - ~String, [65](#)
  - getSize, [66](#)
  - getStr, [66](#)
  - operator+, [66](#), [67](#)
  - operator+=, [67](#)
  - operator=, [67](#)

String, [65](#)  
sum  
  gtest\_lite::Test, [71](#)  
  
tmp  
  gtest\_lite::Test, [71](#)  
Type  
  Item, [45](#)  
  
useItem  
  Item, [44](#)