TerminalArena

0.1

Erzeugt von Doxygen 1.8.20

1 Verzeichnis der Namensbereiche

1.1 Pakete

Hier folgen die Pakete mit einer Kurzbeschreibung (wenn verfügbar):

2 Hierarchie-Verzeichnis

2.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

3 Klassen-Verzeichnis

3.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

4 Dokumentation der Namensbereiche

4.1 TerminalArena-Namensbereichsreferenz

Klassen

- class Program

4.2 TerminalArena.Classes-Namensbereichsreferenz

Klassen

- class Action

  Eine ausführbare Aktion.
- class Arena

  Repräsentiert die Arena für einen Kampf.
- class Coordinate

  Klasse, um eine Koordinate in der Konsole darzustellen.
- class Fight

  Diese Klasse repräsentiert einen Kampf.
- class Fighter

  Die Implementierung des IEntity-Interfaces. Hier wird ein Kämpfer erstellt.
- class HealthPotion

  Abstrakte Klasse für Heiltränke.
- class Helper

  Hilfsklasse. Primär für die Namensgenerierung der Kämpfer.
- class Inventory

  Repräsentiert das Inventar eines Kämpfers.
- class LargeHealthPotion

  Ein mittlerer Heiltrank, der 50 LP wiederherstellen kann. Abgeleitet von der Heiltrank-Klasse.
- class MediumHealthPotion

  Ein mittlerer Heiltrank, der 20 LP wiederherstellen kann. Abgeleitet von der Heiltrank-Klasse.
- class SmallHealthPotion

  Ein kleiner Heiltrank, der 5 LP wiederherstellen kann. Abgeleitet von der Heiltrank-Klasse.
- class UserInterface

  Klasse für das User Interface. Hier sind alle Methoden enthalten, die das Interface darstellen.

4.3 TerminalArena.Helpers-Namensbereichsreferenz

Klassen

- class Dice

  Repräsentiert einen Würfel mit konfigurierbarer Seitenanzahl.

4.4 TerminalArena.Interfaces-Namensbereichsreferenz

Klassen

- interface IEntity

  Interface für die Kämpferklasse. Eigenschaften und Methoden, die für alle Kämpfer identisch sein sollen.
- interface IItem

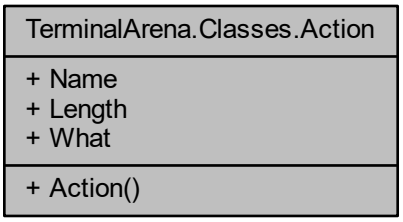
  Beschreibt einen Gegenstand.

5 Klassen-Dokumentation

5.1 TerminalArena.Classes.Action Klassenreferenz

Eine ausführbare Aktion.

Zusammengehörigkeiten von TerminalArena.Classes.Action:



Öffentliche Methoden

- Action (string name, string what)

  Konstruktor einer Aktion

Propertys

- string Name `[get, set]`

  Der Name der Aktion. Dieser wird im Interface angezeigt
- int Length `[get]`

  Die Länge des Aktionsnamens
- string What `[get]`

  Eine Zeichenfolge, die die Aktion identifiziert

5.1.1 Ausführliche Beschreibung

Eine ausführbare Aktion.

Definiert in Zeile 7 der Datei Action.cs.

5.1.2 Beschreibung der Konstruktoren und Destruktoren

5.1.2.1 Action() `TerminalArena.Classes.Action.Action (`

```
string name,
string what )
```

Konstruktor einer Aktion

Parameter

name	Der Name der Aktion
what	Der Identifikator

Definiert in Zeile 27 der Datei Action.cs.

```
00028         {
00029             Name = name;
00030             Length = name.Length;
00031             What = what;
00032         }
```

5.1.3 Dokumentation der Propertys

5.1.3.1 Length `int TerminalArena.Classes.Action.Length [get]`

Die Länge des Aktionsnamens

Definiert in Zeile 16 der Datei Action.cs.

```
00016 { get; }
```

5.1.3.2 Name `string TerminalArena.Classes.Action.Name [get], [set]`

Der Name der Aktion. Dieser wird im Interface angezeigt

Definiert in Zeile 12 der Datei Action.cs.

```
00012 { get; set; }
```

5.1.3.3 What `string TerminalArena.Classes.Action.What [get]`

Eine Zeichenfolge, die die Aktion identifiziert

Definiert in Zeile 20 der Datei Action.cs.

```
00020 { get; }
```

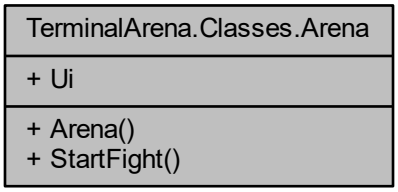
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Action.cs

5.2 TerminalArena.Classes.Arena Klassenreferenz

Repräsentiert die Arena für einen Kampf.

Zusammengehörigkeiten von TerminalArena.Classes.Arena:



Öffentliche Methoden

- Arena ()

  Erstellt die Arena.
- void StartFight ()

  Beginnt den Kampf gegen alle Gegner in der Liste

Propertys

- UserInterface Ui `[get, set]`

  Das Benutzerinterface

5.2.1 Ausführliche Beschreibung

Repräsentiert die Arena für einen Kampf.

Definiert in Zeile 10 der Datei Arena.cs.

5.2.2 Beschreibung der Konstruktoren und Destruktoren

5.2.2.1 Arena() `TerminalArena.Classes.Arena.Arena ( )`

Erstellt die Arena.

Definiert in Zeile 24 der Datei Arena.cs.

```
00025         {
00026             GenerateEnemies();
00027             SetupPlayer();
00028         }
```

5.2.3 Dokumentation der Elementfunktionen

5.2.3.1 StartFight() `void TerminalArena.Classes.Arena.StartFight ( )`

Beginnt den Kampf gegen alle Gegner in der Liste

Definiert in Zeile 33 der Datei Arena.cs.

```
00034         {
00035             var choice = 0;
00036             while (_player.IsAlive() && choice == 0)
00037             {
00038                 Ui.DrawActionBorder();
00039                 Ui.ClearActionArea();
00040 
00041                 Fight fight = new Fight(_enemies.First(), _player, Ui);
00042                 fight.DoFight();
00043 
00044                 Ui.ClearContentArea();
00045                 choice = Ui.PrintMenu("Nächste Runde?", "Nächste Runde", "Beenden");
00046                 if (choice == 1 || !_player.IsAlive())
00047                 {
00048                     break;
00049                 }
00050 
00051                 _enemies.Remove(_enemies.First());
00052             }
00053         }
```

5.2.4 Dokumentation der Propertys

5.2.4.1 Ui `UserInterface TerminalArena.Classes.Arena.Ui [get], [set]`

Das Benutzerinterface

Definiert in Zeile 15 der Datei Arena.cs.

```
00015 { get; set; }
```

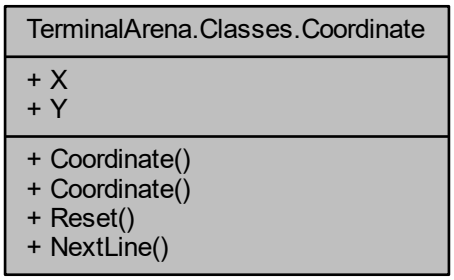
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Arena.cs

5.3 TerminalArena.Classes.Coordinate Klassenreferenz

Klasse, um eine Koordinate in der Konsole darzustellen.

Zusammengehörigkeiten von TerminalArena.Classes.Coordinate:



Öffentliche Methoden

- Coordinate (int x, int y)

  Erstellt eine X/Y-Koordinate mit den angegebenen Werten.
- Coordinate ()

  Erstellt eine Koordinate aus der momentanen Cursorposition.
- void Reset ()

  Setzt den Cursor auf die X/Y-Koordinate zurück.
- void NextLine ()

  Setzt die Koordinate eine Zeile herunter.

Propertys

- int X `[get, set]`

  Position von Links
- int Y `[get, set]`

  Position von Oben

5.3.1 Ausführliche Beschreibung

Klasse, um eine Koordinate in der Konsole darzustellen.

Definiert in Zeile 8 der Datei Coordinate.cs.

5.3.2 Beschreibung der Konstruktoren und Destruktoren

5.3.2.1 Coordinate() `[1/2]` `TerminalArena.Classes.Coordinate.Coordinate (`

```
int x,
int y )
```

Erstellt eine X/Y-Koordinate mit den angegebenen Werten.

Parameter

x	X (oder left)
y	Y (oder top)

Definiert in Zeile 25 der Datei Coordinate.cs.

```
00026         {
00027             X = x;
00028             Y = y;
00029         }
```

5.3.2.2 Coordinate() `[2/2]` `TerminalArena.Classes.Coordinate.Coordinate ( )`

Erstellt eine Koordinate aus der momentanen Cursorposition.

Definiert in Zeile 34 der Datei Coordinate.cs.

```
00035         {
00036             X = Console.CursorLeft;
00037             Y = Console.CursorTop;
00038         }
```

5.3.3 Dokumentation der Elementfunktionen

5.3.3.1 NextLine() void TerminalArena.Classes.Coordinate.NextLine ( )

Setzt die Koordinate eine Zeile herunter.

Definiert in Zeile 51 der Datei Coordinate.cs.

```
00052         {
00053             Y += 1;
00054         }
```

5.3.3.2 Reset() void TerminalArena.Classes.Coordinate.Reset ( )

Setzt den Cursor auf die X/Y-Koordinate zurück.

Definiert in Zeile 43 der Datei Coordinate.cs.

```
00044         {
00045             Console.SetCursorPosition(X, Y);
00046         }
```

5.3.4 Dokumentation der Propertys

5.3.4.1 X int TerminalArena.Classes.Coordinate.X [get], [set]

Position von Links

Definiert in Zeile 13 der Datei Coordinate.cs.

```
00013 { get; set; }
```

5.3.4.2 Y int TerminalArena.Classes.Coordinate.Y [get], [set]

Position von Oben

Definiert in Zeile 18 der Datei Coordinate.cs.

```
00018 { get; set; }
```

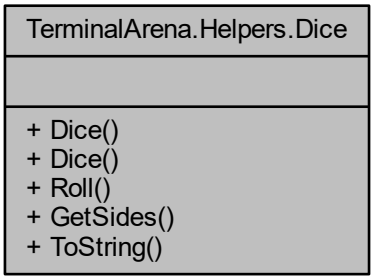
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Coordinate.cs

5.4 TerminalArena.Helpers.Dice Klassenreferenz

Repräsentiert einen Würfel mit konfigurierbarer Seitenanzahl.

Zusammengehörigkeiten von TerminalArena.Helpers.Dice:



Öffentliche Methoden

- Dice (int sides)

  Erstellt einen Würfel mit der gewünschten Seitenzahl.
- Dice ()

  Erstellt einen Würfel mit 6 Seiten.
- int Roll ()

  Würfelt mit dem Würfel.
- int GetSides ()

  Gibt die Anzahl der Seiten zurück.
- override string ToString ()

  Gibt eine Beschreibung des "Dice"-Objekts als Zeichenfolge zurück.

5.4.1 Ausführliche Beschreibung

Repräsentiert einen Würfel mit konfigurierbarer Seitenanzahl.

Definiert in Zeile 8 der Datei Dice.cs.

5.4.2 Beschreibung der Konstruktoren und Destruktoren

5.4.2.1 Dice() [1/2] TerminalArena.Helpers.Dice.Dice (int *sides*)

Erstellt einen Würfel mit der gewünschten Seitenzahl.

Parameter

sides	Seiten des Würfels

Definiert in Zeile 17 der Datei Dice.cs.

```
00018         {
00019             _sides = sides;
00020             _random = new Random();
00021         }
```

5.4.2.2 Dice() [2/2] TerminalArena.Helpers.Dice.Dice ( )

Erstellt einen Würfel mit 6 Seiten.

Definiert in Zeile 26 der Datei Dice.cs.

```
00027         {
00028             _sides = 6;
00029             _random = new Random();
00030         }
```

5.4.3 Dokumentation der Elementfunktionen

5.4.3.1 GetSides() int TerminalArena.Helpers.Dice.GetSides ( )

Gibt die Anzahl der Seiten zurück.

Rückgabe

int

Definiert in Zeile 45 der Datei Dice.cs.

```
00046         {
00047             return _sides;
00048         }
```

5.4.3.2 Roll() int TerminalArena.Helpers.Dice.Roll ( )

Würfelt mit dem Würfel.

Rückgabe

int

Definiert in Zeile 36 der Datei Dice.cs.

```
00037         {
00038             return _random.Next(1, _sides + 1);
00039         }
```

5.4.3.3 ToString() override string TerminalArena.Helpers.Dice.ToString ( )

Gibt eine Beschreibung des "Dice"-Objekts als Zeichenfolge zurück.

Rückgabe

string

Definiert in Zeile 54 der Datei Dice.cs.

```
00055         {
00056             return $"Würfele mit einem {_sides}-seitigen Würfel";
00057         }
```

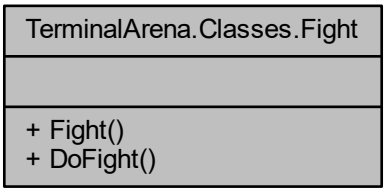
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Dice.cs

5.5 TerminalArena.Classes.Fight Klassenreferenz

Diese Klasse repräsentiert einen Kampf.

Zusammengehörigkeiten von TerminalArena.Classes.Fight:



Öffentliche Methoden

- Fight (Fighter enemy, Fighter player, UserInterface ui)
  Initialisiert einen Kampf zwischen zwei Kämpfern.
- void DoFight ()
  Führt den Kampf aus, bis einer der Kämpfer stirbt.

5.5.1 Ausführliche Beschreibung

Diese Klasse repräsentiert einen Kampf.

Definiert in Zeile 11 der Datei Fight.cs.

5.5.2 Beschreibung der Konstruktoren und Destruktoren

5.5.2.1 Fight() TerminalArena.Classes.Fight.Fight (Fighter *enemy*, Fighter *player*, UserInterface *ui*)

Initialisiert einen Kampf zwischen zwei Kämpfern.

Parameter

enemy	Der Gegner
player	Der Spieler
ui	Eine Instanz des Interface

Definiert in Zeile 24 der Datei Fight.cs.

```
00025         {
00026             _enemy = enemy;
00027             _player = player;
00028             _dice = new Dice(2);
00029             _ui = ui;
00030         }
```

5.5.3 Dokumentation der Elementfunktionen

5.5.3.1 DoFight() `void TerminalArena.Classes.Fight.DoFight ( )`

Führt den Kampf aus, bis einer der Kämpfer stirbt.

Definiert in Zeile 48 der Datei Fight.cs.

```
00049         {
00050             Fighter first = _player;
00051             Fighter second = _enemy;
00052 
00053             _ui.DrawTitle($"{_player} vs {_enemy}");
00054 
00055             bool secondGoesFirst = _dice.Roll() <= _dice.GetSides() / 2;
00056             if (secondGoesFirst)
00057             {
00058                 first = _enemy;
00059                 second = _player;
00060             }
00061 
00062             while (first.IsAlive() && second.IsAlive())
00063             {
00064                 string action;
00065                 if (first.Actions.Count > 0)
00066                 {
00067                     action = _ui.PrintActions(first.Actions);
00068                     first.DoAction(action);
00069                 }
00070                 _ui.ClearContentArea();
00071                 first.Attack(second);
00072                 PrintFighter(first);
00073                 PrintFighter(second);
00074                 PrintMessage(first.Message);
00075                 PrintMessage(second.Message);
00076 
00077                 if (second.IsAlive())
00078                 {
00079                     if (second.Actions.Count > 0)
00080                     {
00081                         action = _ui.PrintActions(second.Actions);
00082                         second.DoAction(action);
00083                     }
00084                     _ui.ClearContentArea();
00085                     second.Attack(first);
00086                     PrintFighter(first);
00087                     PrintFighter(second);
00088                     PrintMessage(second.Message);
00089                     PrintMessage(first.Message);
00090                 }
00091             }
00092         }
```

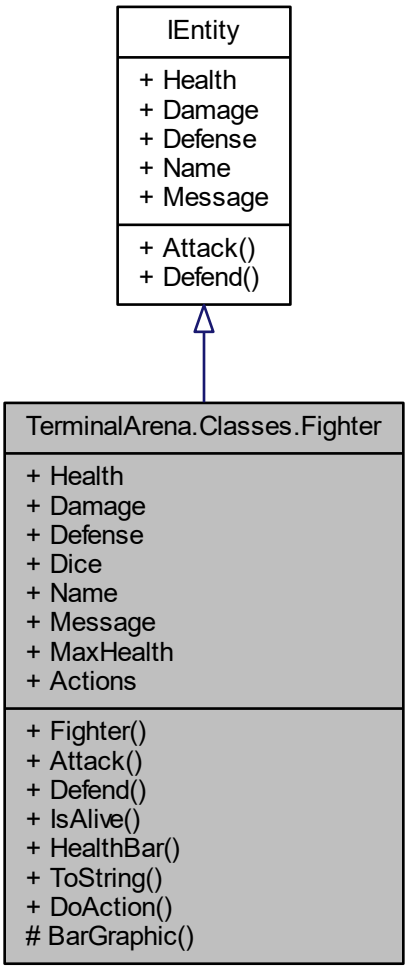
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Fight.cs

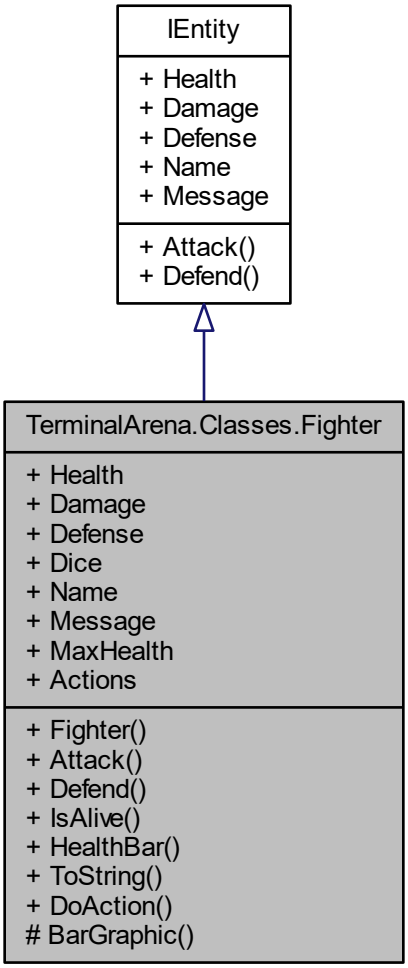
5.6 TerminalArena.Classes.Fighter Klassenreferenz

Die Implementierung des IEntity-Interfaces. Hier wird ein Kämpfer erstellt.

Klassendiagramm für TerminalArena.Classes.Fighter:



Zusammengehörigkeiten von TerminalArena.Classes.Fighter:



Öffentliche Methoden

- Fighter (string name, int health, int damage, int defense)

  Erstellt einen Kämpfer mit den angegebenen Attributen
- virtual void Attack (IEntity enemy)

  Greift den spezifizierten Gegner an
- void Defend (int damage)

  Versucht, sich gegen einen Angriff zu verteidigen
- bool IsAlive ()

  Gibt den Zustand des Kämpfers zurück
- string HealthBar ()

  Öffentliche Methode zur Darstellung einer Gesundheitsleiste
- override string ToString ()

  Liefert die Repräsentation des Fighter-Objekts als String
- void DoAction (string action)

  Führt die gewünschte Aktion aus und aktualisiert die Aktionen

Geschützte Methoden

- string BarGraphic (int current, int maximum)

  Erstellt aus gegenwärtiger und maximaler Resource eine Leiste Dient der grafischen Anzeige der Resource

Propertys

- int Health `[get, set]`

  Die Gesundheit des Kämpfers
- int Damage `[get, set]`

  Der Schaden, den der Kämpfer verursachen kann
- int Defense `[get, set]`

  Die Verteidigung des Kämpfers
- string Name `[get, set]`

  Der Name des Kämpfers
- string Message `[get, set]`

  Eine Nachricht über ein Ereignis
- List< Action > Actions `[get, set]`

  Eine Liste von Aktionen

5.6.1 Ausführliche Beschreibung

Die Implementierung des IEntity-Interfaces. Hier wird ein Kämpfer erstellt.

Definiert in Zeile 14 der Datei Fighter.cs.

5.6.2 Beschreibung der Konstruktoren und Destruktoren

5.6.2.1 Fighter() `TerminalArena.Classes.Fighter.Fighter (`

```
string name,
int health,
int damage,
int defense )
```

Erstellt einen Kämpfer mit den angegebenen Attributen

Parameter

name	Der Name des Kämpfers
health	Die Gesundheit des Kämpfers
damage	Der Schaden des Kämpfers
defense	Die Verteidigung des Kämpfers

Definiert in Zeile 74 der Datei Fighter.cs.

```
00075         {
00076             Name = name;
```

```
00077             MaxHealth = health;
00078             Health = health;
00079             Damage = damage;
00080             Defense = defense;
00081             Dice = new Dice(6);
00082             _inventory = new Inventory();
00083             _inventory.Items.Add(new SmallHealthPotion());
00084             _inventory.Items.Add(new MediumHealthPotion());
00085             _inventory.Items.Add(new LargeHealthPotion());
00086             Actions = new List<Action>();
00087         }
```

5.6.3 Dokumentation der Elementfunktionen

5.6.3.1 Attack() `virtual void TerminalArena.Classes.Fighter.Attack ( IEntity enemy ) [virtual]`

Greift den spezifizierten Gegner an

Parameter

enemy	Der anzugreifende Gegner

Implementiert TerminalArena.Interfaces.IEntity.

Definiert in Zeile 93 der Datei Fighter.cs.

```
00094         {
00095             int calculatedDamage = Damage + Dice.Roll();
00096             Message = ($"{Name} verursacht {calculatedDamage} Schaden.");
00097             enemy.Defend(calculatedDamage);
00098         }
```

5.6.3.2 BarGraphic() `string TerminalArena.Classes.Fighter.BarGraphic ( int current, int maximum ) [protected]`

Erstellt aus gegenwärtiger und maximaler Resource eine Leiste Dient der grafischen Anzeige der Resource

Parameter

current	Gegenwärtige Resource
maximum	Maximale Resource

Rückgabe

Eine Zeichenfolge, die die Resource repräsentiert

Definiert in Zeile 142 der Datei Fighter.cs.

```
00143         {
00144             string bar = "";
00145             int total = 20;
00146             double count = Math.Round(((double) current / maximum) * total);
```

```
00147             if (count == 0 && IsAlive())
00148             {
00149                 count = 1;
00150             }
00151
00152             for (int i = 0; i < count; i++)
00153             {
00154                 bar += "";
00155             }
00156
00157             return bar.PadRight(total);
00158         }
```

5.6.3.3 Defend() void TerminalArena.Classes.Fighter.Defend (int *damage*)

Versucht, sich gegen einen Angriff zu verteidigen

Parameter

damage	Zu verursachender Schaden

Implementiert TerminalArena.Interfaces.IEntity.

Definiert in Zeile 104 der Datei Fighter.cs.

```
00105         {
00106             string message;
00107             int damageTaken = damage - Dice.Roll();
00108             if (damageTaken > 0)
00109             {
00110                 Health -= damageTaken;
00111                 message = $"{Name} hat {damageTaken} Punkte Schaden genommen";
00112                 if (Health <= 0)
00113                 {
00114                     Health = 0;
00115                     message += " und ist gestorben";
00116                 }
00117             }
00118             else
00119             {
00120                 message = $"{Name} hat den Schaden geblockt";
00121             }
00122
00123             Message = message;
00124         }
```

5.6.3.4 DoAction() void TerminalArena.Classes.Fighter.DoAction (string *action*)

Führt die gewünschte Aktion aus und aktualisiert die Aktionen

Parameter

action	Die gewünschte Aktion

Definiert in Zeile 183 der Datei Fighter.cs.

```
00184         {
00185             switch (action)
00186             {
00187                 case "potion":
00188                 {
```

```
00189                     UsePotion();
00190                     break;
00191                 }
00192                 case "attack":
00193                     break;
00194                 case "flee":
00195                 {
00196                     Health = 0;
00197                     Message = $"{Name} hat aufgegeben.";
00198                     break;
00199                 }
00200             }
00201         }
```

5.6.3.5 HealthBar() string TerminalArena.Classes.Fighter.HealthBar ( )

Öffentliche Methode zur Darstellung einer Gesundheitsleiste

Rückgabe

Eine Zeichenfolge, die die Gesundheit repräsentiert

Definiert in Zeile 164 der Datei Fighter.cs.

```
00165         {
00166             return "[" + BarGraphic(Health, MaxHealth).Pastel(Color.Red).PastelBg(Color.DarkRed) +
00167                    $"] {Health} / {MaxHealth}";
00168         }
```

5.6.3.6 IsAlive() bool TerminalArena.Classes.Fighter.IsAlive ( )

Gibt den Zustand des Kämpfers zurück

Rückgabe

Lebendig ja/nein

Definiert in Zeile 130 der Datei Fighter.cs.

```
00131         {
00132             return Health > 0;
00133         }
```

5.6.3.7 ToString() override string TerminalArena.Classes.Fighter.ToString ( )

Liefert die Repräsentation des Fighter-Objekts als String

Rückgabe

(string) Fighter-Objekt

Definiert in Zeile 174 der Datei Fighter.cs.

```
00175         {
00176             return Name;
00177         }
```

5.6.4 Dokumentation der Propertys

5.6.4.1 Actions `List<Action> TerminalArena.Classes.Fighter.Actions [get], [set]`

Eine Liste von Aktionen

Definiert in Zeile 61 der Datei Fighter.cs.

```
00061 { get; set; }
```

5.6.4.2 Damage `int TerminalArena.Classes.Fighter.Damage [get], [set]`

Der Schaden, den der Kämpfer verursachen kann

Definiert in Zeile 26 der Datei Fighter.cs.

```
00026 { get; set; }
```

5.6.4.3 Defense `int TerminalArena.Classes.Fighter.Defense [get], [set]`

Die Verteidigung des Kämpfers

Definiert in Zeile 31 der Datei Fighter.cs.

```
00031 { get; set; }
```

5.6.4.4 Health `int TerminalArena.Classes.Fighter.Health [get], [set]`

Die Gesundheit des Kämpfers

Definiert in Zeile 21 der Datei Fighter.cs.

```
00021 { get; set; }
```

5.6.4.5 Message `string TerminalArena.Classes.Fighter.Message [get], [set]`

Eine Nachricht über ein Ereignis

Definiert in Zeile 46 der Datei Fighter.cs.

```
00046 { get; set; }
```

5.6.4.6 Name `string TerminalArena.Classes.Fighter.Name [get], [set]`

Der Name des Kämpfers

Definiert in Zeile 41 der Datei Fighter.cs.

```
00041 { get; set; }
```

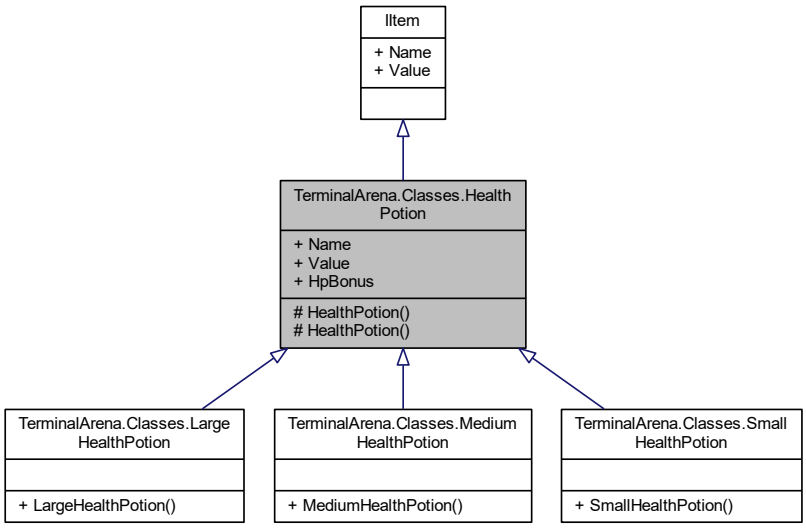
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Fighter.cs

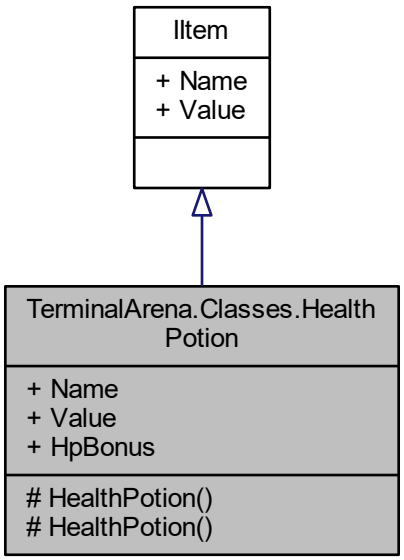
5.7 TerminalArena.Classes.HealthPotion Klassenreferenz

Abstrakte Klasse für Heiltränke.

Klassendiagramm für TerminalArena.Classes.HealthPotion:



Zusammengehörigkeiten von TerminalArena.Classes.HealthPotion:



Geschützte Methoden

- HealthPotion (int hpBonus)

  Erstellt einen Heiltrank mit der angegebenen Stärke.
- HealthPotion ()

  Erstellt einen Heiltrank mit der Stärke 10.

Propertys

- string Name `[get, set]`

  Der Name des Heiltranks
- int Value `[get, set]`

  Der Wert des Heiltranks
- int HpBonus `[get, set]`

  Die Anzahl an Lebenspunkten, die der Heiltrank wiederherstellt.

5.7.1 Ausführliche Beschreibung

Abstrakte Klasse für Heiltränke.

Definiert in Zeile 8 der Datei HealthPotion.cs.

5.7.2 Beschreibung der Konstruktoren und Destruktoren

5.7.2.1 HealthPotion() [1/2] TerminalArena.Classes.HealthPotion.HealthPotion (int *hpBonus*) [protected]

Erstellt einen Heiltrank mit der angegebenen Stärke.

Parameter

hpBonus	Die Stärke des Tranks

Definiert in Zeile 27 der Datei HealthPotion.cs.

```
00028         {
00029         }
```

5.7.2.2 HealthPotion() [2/2] TerminalArena.Classes.HealthPotion.HealthPotion ( ) [protected]

Erstellt einen Heiltrank mit der Stärke 10.

Definiert in Zeile 34 der Datei HealthPotion.cs.

```
00035         {
00036         }
```

5.7.3 Dokumentation der Propertys

5.7.3.1 HpBonus int TerminalArena.Classes.HealthPotion.HpBonus [get], [set]

Die Anzahl an Lebenspunkten, die der Heiltrank wiederherstellt.

Definiert in Zeile 21 der Datei HealthPotion.cs.

```
00021 { get; set; }
```

5.7.3.2 Name string TerminalArena.Classes.HealthPotion.Name [get], [set]

Der Name des Heiltranks

Definiert in Zeile 13 der Datei HealthPotion.cs.

```
00013 { get; set; }
```

5.7.3.3 Value int TerminalArena.Classes.HealthPotion.Value [get], [set]

Der Wert des Heiltranks

Definiert in Zeile 17 der Datei HealthPotion.cs.

```
00017 { get; set; }
```

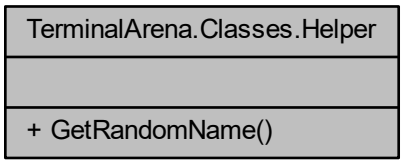
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- HealthPotion.cs

5.8 TerminalArena.Classes.Helper Klassenreferenz

Hilfsklasse. Primär für die Namensgenerierung der Kämpfer.

Zusammengehörigkeiten von TerminalArena.Classes.Helper:



Öffentliche, statische Methoden

- static string GetRandomName ()

  Gibt einen zufälligen Namen aus der names.json-Datei zurück.

5.8.1 Ausführliche Beschreibung

Hilfsklasse. Primär für die Namensgenerierung der Kämpfer.

Definiert in Zeile 11 der Datei Helper.cs.

5.8.2 Dokumentation der Elementfunktionen

5.8.2.1 GetRandomName() `static string TerminalArena.Classes.Helper.GetRandomName ( ) [static]`

Gibt einen zufäligen Namen aus der names.json-Datei zurück.

Rückgabe

string

Definiert in Zeile 17 der Datei Helper.cs.

```
00018         {
00019             StreamReader file = File.OpenText(@"names.json");
00020 
00021             List<string> names = JsonConvert.DeserializeObject<List<string»(file.ReadToEnd());
00022             return names[GetRandomNumber(0, names.Count)];
00023         }
```

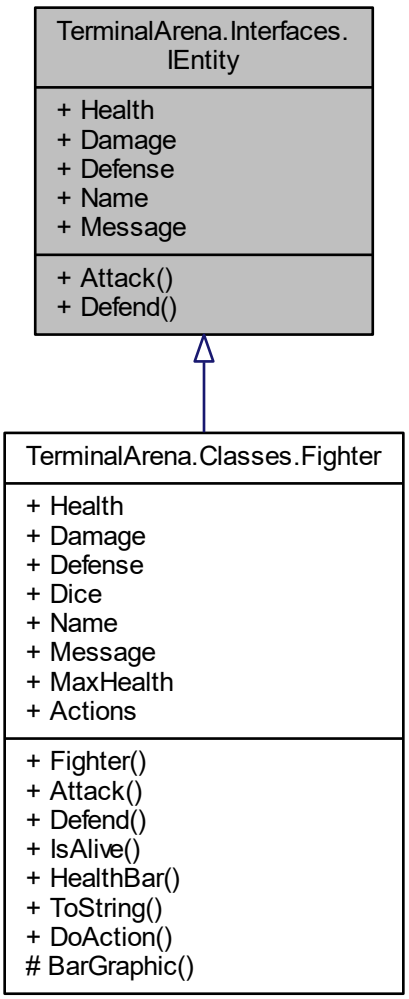
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Helper.cs

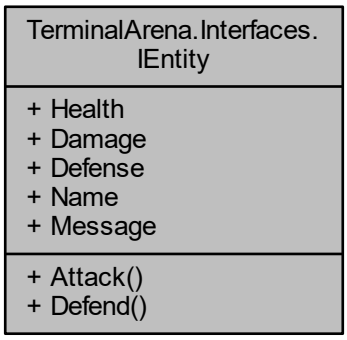
5.9 TerminalArena.Interfaces.IEntity Schnittstellenreferenz

Interface für die Kämpferklasse. Eigenschaften und Methoden, die für alle Kämpfer identisch sein sollen.

Klassendiagramm für TerminalArena.Interfaces.IEntity:



Zusammengehörigkeiten von TerminalArena.Interfaces.IEntity:



Öffentliche Methoden

- void Attack (IEntity enemy)

  Greift ein Ziel an. Der Schaden und der Verteidigungswert des Gegners werden berechnet.
- void Defend (int damage)

  Verteidigung gegen einen Angriff. Hier wird entschieden, ob und wie viel Schaden verursacht wird.

Propertys

- int Health `[get, set]`

  Gesundheit des Kämpfers
- int Damage `[get, set]`

  Schadenswert des Kämpfers
- int Defense `[get, set]`

  Verteidigungswert des Kämpfers
- string Name `[get, set]`

  Der Name des Kämpfers
- string Message `[get, set]`

  Eine Nachricht über Ereignisse

5.9.1 Ausführliche Beschreibung

Interface für die Kämpferklasse. Eigenschaften und Methoden, die für alle Kämpfer identisch sein sollen.

Definiert in Zeile 7 der Datei IEntity.cs.

5.9.2 Dokumentation der Elementfunktionen

5.9.2.1 Attack() `void TerminalArena.Interfaces.IEntity.Attack (`

`IEntity` *enemy*)

Greift ein Ziel an. Der Schaden und der Verteidigungswert des Gegners werden berechnet.

Parameter

enemy	Der anzugreifende Gegner

Implementiert in TerminalArena.Classes.Fighter.

5.9.2.2 Defend() `void TerminalArena.Interfaces.IEntity.Defend ( int damage )`

Verteidigung gegen einen Angriff. Hier wird entschieden, ob und wie viel Schaden verursacht wird.

Parameter

damage	Der Schaden, der verursacht werden soll

Implementiert in TerminalArena.Classes.Fighter.

5.9.3 Dokumentation der Propertys

5.9.3.1 Damage `int TerminalArena.Interfaces.IEntity.Damage [get], [set]`

Schadenswert des Kämpfers

Definiert in Zeile 17 der Datei IEntity.cs.

```
00017 { get; set; }
```

5.9.3.2 Defense `int TerminalArena.Interfaces.IEntity.Defense [get], [set]`

Verteidigungswert des Kämpfers

Definiert in Zeile 22 der Datei IEntity.cs.

```
00022 { get; set; }
```

5.9.3.3 Health `int TerminalArena.Interfaces.IEntity.Health [get], [set]`

Gesundheit des Kämpfers

Definiert in Zeile 12 der Datei IEntity.cs.

```
00012 { get; set; }
```

5.9.3.4 Message `string TerminalArena.Interfaces.IEntity.Message [get], [set]`

Eine Nachricht über Ereignisse

Definiert in Zeile 32 der Datei IEntity.cs.

```
00032 { get; set; }
```

5.9.3.5 Name `string TerminalArena.Interfaces.IEntity.Name [get], [set]`

Der Name des Kämpfers

Definiert in Zeile 27 der Datei IEntity.cs.

```
00027 { get; set; }
```

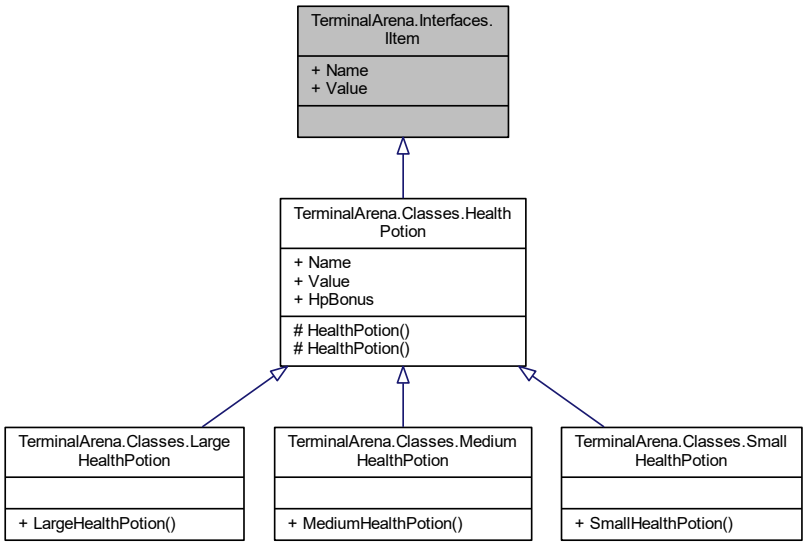
Die Dokumentation für diese Schnittstelle wurde erzeugt aufgrund der Datei:

- IEntity.cs

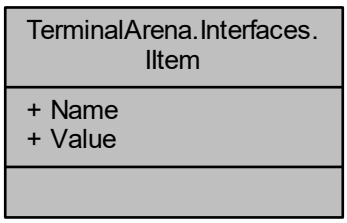
5.10 TerminalArena.Interfaces.IItem Schnittstellenreferenz

Beschreibt einen Gegenstand.

Klassendiagramm für TerminalArena.Interfaces.IItem:



Zusammengehörigkeiten von TerminalArena.Interfaces.IItem:



Propertys

- string Name `[get, set]`
  Der Name des Gegenstandes
- int Value `[get, set]`
  Der Wert des Gegenstandes

5.10.1 Ausführliche Beschreibung

Beschreibt einen Gegenstand.

Definiert in Zeile 6 der Datei IItem.cs.

5.10.2 Dokumentation der Propertys

5.10.2.1 Name `string TerminalArena.Interfaces.IItem.Name [get], [set]`

Der Name des Gegenstandes

Definiert in Zeile 11 der Datei IItem.cs.

```
00011 { get; set; }
```

5.10.2.2 Value `int TerminalArena.Interfaces.IItem.Value [get], [set]`

Der Wert des Gegenstandes

Definiert in Zeile 16 der Datei IItem.cs.

```
00016 { get; set; }
```

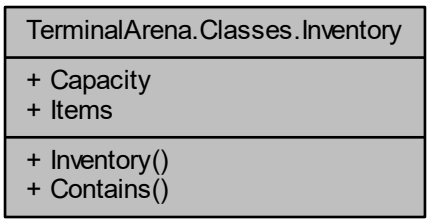
Die Dokumentation für diese Schnittstelle wurde erzeugt aufgrund der Datei:

- IItem.cs

5.11 TerminalArena.Classes.Inventory Klassenreferenz

Repräsentiert das Inventar eines Kämpfers.

Zusammengehörigkeiten von TerminalArena.Classes.Inventory:



Öffentliche Methoden

- Inventory (int capacity=10)

  Initialisiert ein Inventar mit einer Größe von 10 (falls nicht anders angegeben)
- bool Contains (Type itemType)

  Prüft, ob im Inventar ein Gegenstand des gesuchten Typs existiert, und gibt das Ergebnis zurück.

Öffentliche Attribute

- int Capacity => Items?.Count ?? 0

  Die Kapazität des Inventars.

Propertys

- List< IItem > Items `[get]`

  Eine Liste von Gegenständen.

5.11.1 Ausführliche Beschreibung

Repräsentiert das Inventar eines Kämpfers.

Definiert in Zeile 10 der Datei Inventory.cs.

5.11.2 Beschreibung der Konstruktoren und Destruktoren

5.11.2.1 Inventory() `TerminalArena.Classes.Inventory.Inventory ( int capacity = 10 )`

Initialisiert ein Inventar mit einer Größe von 10 (falls nicht anders angegeben)

Parameter

capacity	Die Kapazität des Inventars

Definiert in Zeile 26 der Datei Inventory.cs.

```
00027         {
00028             SetInventoryCapacity(capacity);
00029         }
```

5.11.3 Dokumentation der Elementfunktionen

5.11.3.1 Contains() `bool TerminalArena.Classes.Inventory.Contains ( Type itemType )`

Prüft, ob im Inventar ein Gegenstand des gesuchten Typs existiert, und gibt das Ergebnis zurück.

Parameter

itemType	Der Typ des Gegenstands

Rückgabe

Wahr oder Falsch

Definiert in Zeile 36 der Datei Inventory.cs.

```
00037         {
00038             foreach (IItem item in Items)
00039             {
00040                 if (item.GetType() == itemType)
00041                 {
00042                     return true;
00043                 }
00044             }
00045 
00046             return false;
00047         }
```

5.11.4 Dokumentation der Datenelemente

5.11.4.1 Capacity `int TerminalArena.Classes.Inventory.Capacity => Items?.Count ?? 0`

Die Kapazität des Inventars.

Definiert in Zeile 20 der Datei Inventory.cs.

5.11.5 Dokumentation der Propertys

5.11.5.1 Items `List<IItem> TerminalArena.Classes.Inventory.Items [get]`

Eine Liste von Gegenständen.

Definiert in Zeile 15 der Datei Inventory.cs.

```
00015 { get; private set; }
```

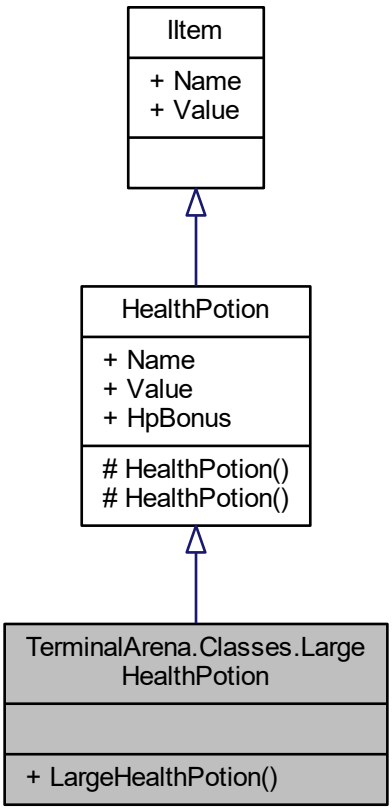
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Inventory.cs

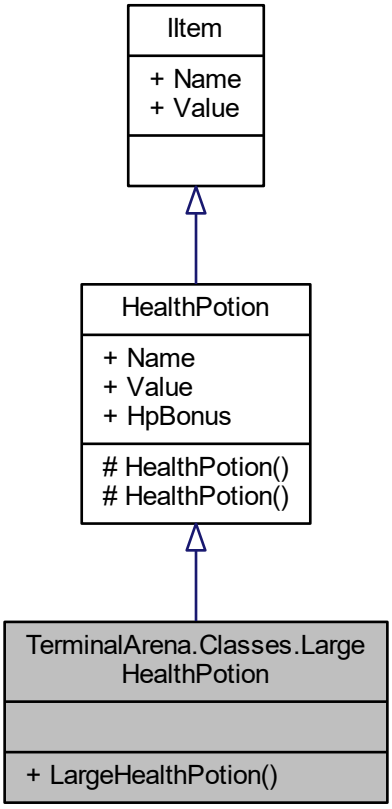
5.12 TerminalArena.Classes.LargeHealthPotion Klassenreferenz

Ein mittlerer Heiltrank, der 50 LP wiederherstellen kann. Abgeleitet von der Heiltrank-Klasse.

Klassendiagramm für TerminalArena.Classes.LargeHealthPotion:



Zusammengehörigkeiten von TerminalArena.Classes.LargeHealthPotion:



Öffentliche Methoden

- LargeHealthPotion ()

  Erstellt einen Heiltrank mit der Stärke 50

Weitere Geerbte Elemente

5.12.1 Ausführliche Beschreibung

Ein mittlerer Heiltrank, der 50 LP wiederherstellen kann. Abgeleitet von der Heiltrank-Klasse.

Siehe auch

HealthPotion

Definiert in Zeile 8 der Datei LargeHealthPotion.cs.

5.12.2 Beschreibung der Konstruktoren und Destruktoren

5.12.2.1 LargeHealthPotion() `TerminalArena.Classes.LargeHealthPotion.LargeHealthPotion ( )`

Erstellt einen Heiltrank mit der Stärke 50

Definiert in Zeile 13 der Datei LargeHealthPotion.cs.

```
00013                                   : base(50)
00014         {
00015             Name = "Großer Heiltrank";
00016             Value = 30;
00017         }
```

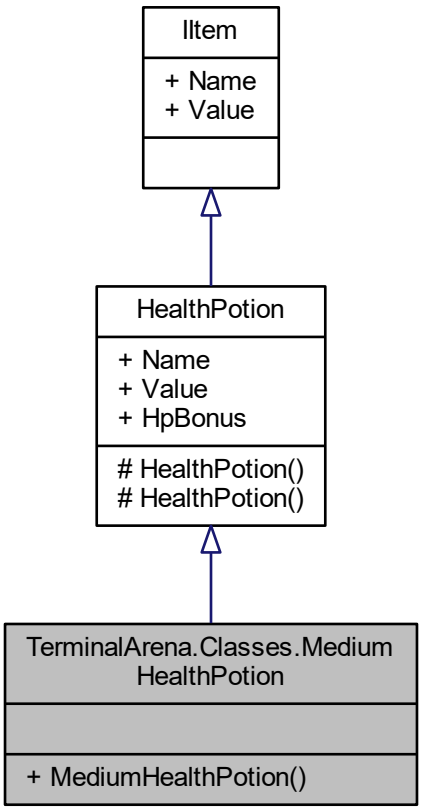
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- LargeHealthPotion.cs

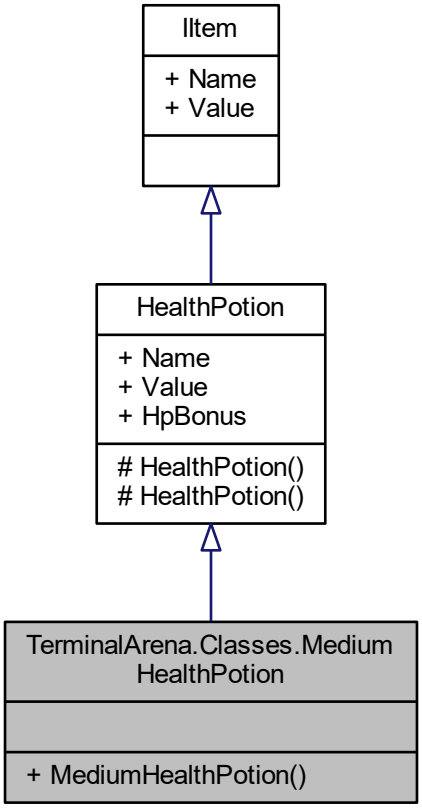
5.13 TerminalArena.Classes.MediumHealthPotion Klassenreferenz

Ein mittlerer Heiltrank, der 20 LP wiederherstellen kann. Abgeleitet von der Heiltrank-Klasse.

Klassendiagramm für TerminalArena.Classes.MediumHealthPotion:



Zusammengehörigkeiten von TerminalArena.Classes.MediumHealthPotion:



Öffentliche Methoden

- MediumHealthPotion ()

  Erstellt einen Heiltrank mit der Stärke 20

Weitere Geerbte Elemente

5.13.1 Ausführliche Beschreibung

Ein mittlerer Heiltrank, der 20 LP wiederherstellen kann. Abgeleitet von der Heiltrank-Klasse.

Siehe auch

HealthPotion

Definiert in Zeile 8 der Datei MediumHealthPotion.cs.

5.13.2 Beschreibung der Konstruktoren und Destruktoren

5.13.2.1 MediumHealthPotion() `TerminalArena.Classes.MediumHealthPotion.MediumHealthPotion ( )`

Erstellt einen Heiltrank mit der Stärke 20

Definiert in Zeile 13 der Datei MediumHealthPotion.cs.

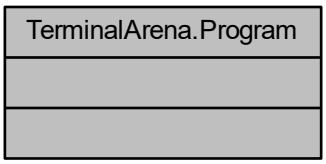
```
00013                                   : base(20)
00014         {
00015             Name = "Mittlerer Heiltrank";
00016             Value = 12;
00017         }
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- MediumHealthPotion.cs

5.14 TerminalArena.Program Klassenreferenz

Zusammengehörigkeiten von TerminalArena.Program:



5.14.1 Ausführliche Beschreibung

Definiert in Zeile 9 der Datei Program.cs.

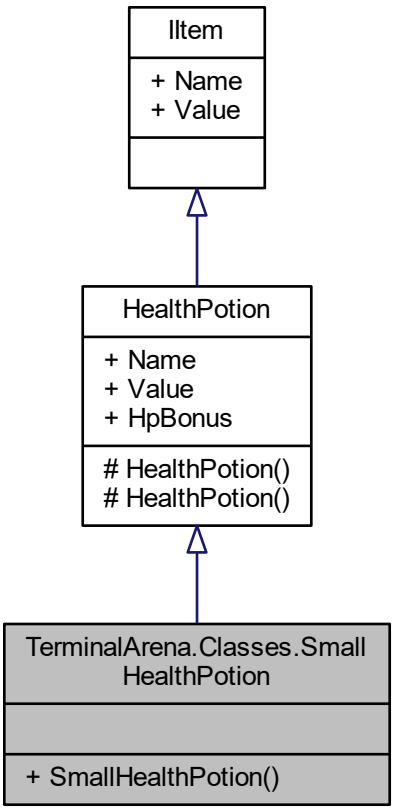
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Program.cs

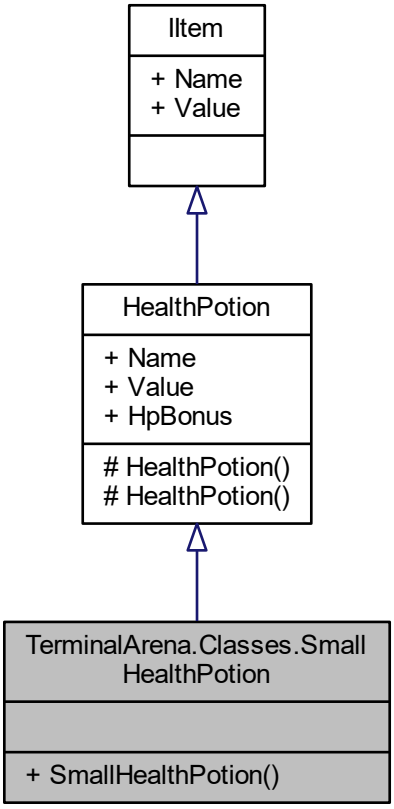
5.15 TerminalArena.Classes.SmallHealthPotion Klassenreferenz

Ein kleiner Heiltrank, der 5 LP wiederherstellen kann. Abgeleitet von der Heiltrank-Klasse.

Klassendiagramm für TerminalArena.Classes.SmallHealthPotion:



Zusammengehörigkeiten von TerminalArena.Classes.SmallHealthPotion:



Öffentliche Methoden

- SmallHealthPotion ()

  Erstellt einen Heiltrank mit der Stärke 5

Weitere Geerbte Elemente

5.15.1 Ausführliche Beschreibung

Ein kleiner Heiltrank, der 5 LP wiederherstellen kann. Abgeleitet von der Heiltrank-Klasse.

Siehe auch

HealthPotion

Definiert in Zeile 8 der Datei SmallHealthPotion.cs.

5.15.2 Beschreibung der Konstruktoren und Destruktoren

5.15.2.1 SmallHealthPotion() `TerminalArena.Classes.SmallHealthPotion.SmallHealthPotion ( )`

Erstellt einen Heiltrank mit der Stärke 5

Definiert in Zeile 13 der Datei SmallHealthPotion.cs.

```
00013                                   : base(5)
00014         {
00015             Name = "Kleiner Heiltrank";
00016             Value = 5;
00017         }
```

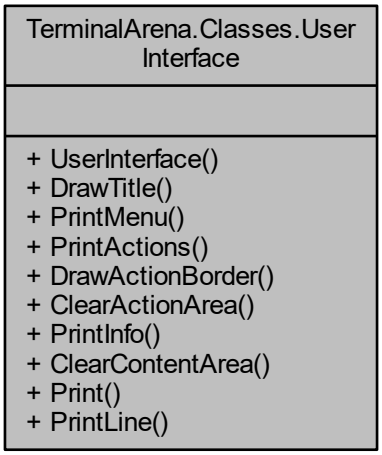
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- SmallHealthPotion.cs

5.16 TerminalArena.Classes.UserInterface Klassenreferenz

Klasse für das User Interface. Hier sind alle Methoden enthalten, die das Interface darstellen.

Zusammengehörigkeiten von TerminalArena.Classes.UserInterface:



Öffentliche Methoden

- UserInterface (string title)

  Löscht die Inhalte der Konsole, legt globale Eigenschaften fest und zeichnet Rahmen und Titel.
- void DrawTitle (string title)

  Schreibt den Titel des Fensters.
- int PrintMenu (string title, params string[ ] options)

  Zeigt ein Menü mit den gegebenen Optionen an. Gibt den Index der gewählten Option zurück.
- string PrintActions (List< Action > actions)

  Gibt eine Liste von Aktionen als Menü aus.
- void DrawActionBorder ()

  Zeichnet den Rand des Aktionsfeldes
- void ClearActionArea ()

  Löscht den Inhalt des Aktionsbereiches
- void PrintInfo ()

  Zeigt eine Information über das Programm an.
- void ClearContentArea ()

  Löscht den Inhaltsteil des Fensters (alles unterhalb des Titels und innerhalb der Rahmen).
- void Print (string message)

  Gibt eine Zeichenfolge im Inhaltsbereich aus, ohne in die nächste Zeile zu springen (Wie Console.Write).
- void PrintLine (string message=null)

  Gibt eine Zeichenfolge im Inhaltsbereich aus und springt in die nächste Zeile (wie Console.WriteLine). Ohne Zeichenfolge wird der Cursor nur in die nächste Zeile gesetzt.

5.16.1 Ausführliche Beschreibung

Klasse für das User Interface. Hier sind alle Methoden enthalten, die das Interface darstellen.

Definiert in Zeile 10 der Datei UserInterface.cs.

5.16.2 Beschreibung der Konstruktoren und Destruktoren

5.16.2.1 UserInterface() `TerminalArena.Classes.UserInterface.UserInterface ( string title )`

Löscht die Inhalte der Konsole, legt globale Eigenschaften fest und zeichnet Rahmen und Titel.

Definiert in Zeile 37 der Datei UserInterface.cs.

```
00038         {
00039             Console.Clear();
00040             SetupConsole();
00041             DrawOuterBorder();
00042 
00043             _contentHeight =
00044                 (Console.WindowHeight / 3) * 2 - TitleHeight - Padding -
00045                 2; // Fensterhöhe / 3 * 2 - Titel + Trenner - Abstand oben & unten - Rahmen oben &
    unten
00046             _contentWidth =
00047                 Console.WindowWidth - 2 - 2 * Padding; // Fensterbreite - Rahmen links & rechts -
    Abstand links & rechts
00048 
00049             _contentArea = new[]
00050             {
00051                 new Coordinate(1 + Padding, TitleHeight + Padding + 1), // Inhaltsfläche links oben
```

```
00052                 new Coordinate(_contentWidth, _contentHeight) // Inhaltsfläche rechts unten
00053             };
00054
00055
00056             _actionHeight = Console.WindowHeight - _contentHeight - Padding * 2 - 2 - TitleHeight;
00057             _actionWidth = _contentWidth;
00058
00059             _actionArea = new[]
00060             {
00061                 new Coordinate(1 + Padding, Console.WindowHeight - _actionHeight + Padding),
00062                 new Coordinate(Console.WindowWidth - 1 - Padding, Console.WindowHeight - 1 - Padding)
00063             };
00064
00065             DrawTitle(title);
00066         }
```

5.16.3 Dokumentation der Elementfunktionen

5.16.3.1 ClearActionArea() `void TerminalArena.Classes.UserInterface.ClearActionArea ( )`

Löscht den Inhalt des Aktionsbereiches

Definiert in Zeile 309 der Datei UserInterface.cs.

```
00310         {
00311             Coordinate currentPosition = new Coordinate();
00312             Coordinate position = new Coordinate(_actionArea[0].X, _actionArea[0].Y);
00313             for (int i = position.Y; i < _actionArea[1].Y; i++)
00314             {
00315                 Console.SetCursorPosition(position.X, i);
00316                 Console.Write(new string(' ', _contentWidth));
00317                 position.NextLine();
00318             }
00319
00320             Console.SetCursorPosition(_actionArea[0].X, _actionArea[0].Y);
00321             currentPosition.Reset();
00322         }
```

5.16.3.2 ClearContentArea() `void TerminalArena.Classes.UserInterface.ClearContentArea ( )`

Löscht den Inhaltsteil des Fensters (alles unterhalb des Titels und innerhalb der Rahmen).

Definiert in Zeile 344 der Datei UserInterface.cs.

```
00345         {
00346             Coordinate position = new Coordinate(_contentArea[0].X, _contentArea[0].Y);
00347
00348             for (int i = 0 + 1; i < _contentArea[1].Y; i++)
00349             {
00350                 Console.SetCursorPosition(position.X, position.Y);
00351                 Console.Write(new string(' ', _contentWidth));
00352                 position.NextLine();
00353             }
00354
00355             Console.SetCursorPosition(_contentArea[0].X, _contentArea[0].Y);
00356         }
```

5.16.3.3 DrawActionBorder() `void TerminalArena.Classes.UserInterface.DrawActionBorder ( )`

Zeichnet den Rand des Aktionsfeldes

Definiert in Zeile 282 der Datei UserInterface.cs.

```
00283         {
00284             Coordinate currentPosition = new Coordinate();
00285             int x = 0;
00286             int y = (Console.WindowHeight / 3) * 2;
00287 
00288             Coordinate left = new Coordinate(x, y);
00289             Coordinate right = new Coordinate(Console.WindowWidth - 1, y);
00290 
00291             for (int i = x; i < Console.WindowWidth - 1; i++)
00292             {
00293                 Console.SetCursorPosition(i, y);
00294                 Console.Write(Horizontal);
00295             }
00296 
00297             Console.SetCursorPosition(left.X, left.Y);
00298             Console.Write(VerticalLeft);
00299 
00300             Console.SetCursorPosition(right.X, right.Y);
00301             Console.Write(VerticalRight);
00302 
00303             currentPosition.Reset();
00304         }
```

5.16.3.4 DrawTitle() `void TerminalArena.Classes.UserInterface.DrawTitle ( string title )`

Schreibt den Titel des Fensters.

Parameter

title	Der Titel

Definiert in Zeile 106 der Datei UserInterface.cs.

```
00107         {
00108             ClearTitleArea();
00109             Coordinate position = new Coordinate(_contentArea[0].X, _contentArea[0].Y);
00110 
00111             CenterString(title, new Coordinate(1, 1));
00112 
00113             Console.SetCursorPosition(0, _contentArea[0].Y - 1 - Padding);
00114 
00115             string separator = VerticalLeft + new string(Horizontal, Console.WindowWidth - 2) +
      VerticalRight;
00116             Console.Write(separator);
00117 
00118             position.Reset();
00119         }
```

5.16.3.5 Print() `void TerminalArena.Classes.UserInterface.Print ( string message )`

Gibt eine Zeichenfolge im Inhaltsbereich aus, ohne in die nächste Zeile zu springen (Wie Console.Write).

Parameter

message	Zeichenfolge

Definiert in Zeile 371 der Datei UserInterface.cs.

```
00372         {
00373             Coordinate pos = new Coordinate();
00374 
00375             Console.SetCursorPosition(pos.X, pos.Y);
00376             Console.Write(message);
00377         }
```

5.16.3.6 PrintActions() `string TerminalArena.Classes.UserInterface.PrintActions ( List< Action > actions )`

Gibt eine Liste von Aktionen als Menü aus.

Parameter

actions	Eine Liste von Aktionen

Rückgabe

Die gewählte Aktion

Definiert in Zeile 204 der Datei UserInterface.cs.

```
00205         {
00206             Console.SetCursorPosition(_actionArea[0].X, _actionArea[0].Y);
00207             ConsoleKey key;
00208             int currentSelection = 0;
00209 
00210             int widestAction = actions.Select(action => action.Length).Prepend(0).Max() + 2;
00211             int actionsPerLine = _actionWidth / widestAction;
00212 
00213             do
00214             {
00215                 ClearActionArea();
00216 
00217                 for (int i = 0; i < actions.Count; i++)
00218                 {
00219                     Console.SetCursorPosition(_actionArea[0].X + i % actionsPerLine * widestAction,
00220                         _actionArea[0].Y + i / actionsPerLine);
00221 
00222                     if (i == currentSelection)
00223                     {
00224                         Console.ForegroundColor = ConsoleColor.Cyan;
00225                     }
00226 
00227                     Console.Write(actions[i].Name);
00228                     Console.ResetColor();
00229                 }
00230 
00231                 key = Console.ReadKey(true).Key;
00232 
00233                 switch (key)
00234                 {
00235                     case ConsoleKey.UpArrow:
00236                     {
00237                         if (currentSelection >= actionsPerLine)
00238                         {
00239                             currentSelection -= actionsPerLine;
00240                         }
00241 
00242                         break;
00243                     }
00244 
00245                     case ConsoleKey.DownArrow:
00246                     {
00247                         if (currentSelection + actionsPerLine < actions.Count)
00248                         {
00249                             currentSelection += actionsPerLine;
00250                         }
00251 
00252                         break;
00253                     }
00254                     case ConsoleKey.LeftArrow:
```

```
00255                     {
00256                         if (currentSelection % actionsPerLine > 0)
00257                         {
00258                             currentSelection--;
00259                         }
00260 
00261                         break;
00262                     }
00263 
00264                     case ConsoleKey.RightArrow:
00265                     {
00266                         if (currentSelection % actionsPerLine < actionsPerLine - 1)
00267                         {
00268                             currentSelection++;
00269                         }
00270 
00271                         break;
00272                     }
00273                 }
00274             } while (key != ConsoleKey.Enter);
00275 
00276             return actions[currentSelection].What;
00277         }
```

5.16.3.7 PrintInfo() `void TerminalArena.Classes.UserInterface.PrintInfo ( )`

Zeigt eine Information über das Programm an.

Definiert in Zeile 327 der Datei UserInterface.cs.

```
00328         {
00329             Coordinate currentPosition = new Coordinate();
00330             ClearContentArea();
00331             CenterString("Terminal Arena", currentPosition);
00332             currentPosition.NextLine();
00333             CenterString("v. 1", currentPosition);
00334             currentPosition.NextLine();
00335             CenterString("Press enter to return to the menu", currentPosition);
00336             do
00337             {
00338             } while (Console.ReadKey(true).Key != ConsoleKey.Enter);
00339         }
```

5.16.3.8 PrintLine() `void TerminalArena.Classes.UserInterface.PrintLine ( string message = null )`

Gibt eine Zeichenfolge im Inhaltsbereich aus und springt in die nächste Zeile (wie Console.WriteLine). Ohne Zeichenfolge wird der Cursor nur in die nächste Zeile gesetzt.

Parameter

message	Zeichenfolge

Definiert in Zeile 384 der Datei UserInterface.cs.

```
00385         {
00386             Coordinate pos = new Coordinate();
00387             Console.SetCursorPosition(pos.X, pos.Y);
00388             Console.Write(message);
00389             Console.SetCursorPosition(pos.X, pos.Y + 1);
00390         }
```

5.16.3.9 PrintMenu() `int TerminalArena.Classes.UserInterface.PrintMenu ( string title, params string[] options )`

Zeigt ein Menü mit den gegebenen Optionen an. Gibt den Index der gewählten Option zurück.

Parameter

title	Der Titel des Menüs
options	Eine Liste von Optionen

Rückgabe

Index der gewählten Option

Definiert in Zeile 135 der Datei UserInterface.cs.

```
00136         {
00137             ClearContentArea();
00138             ClearTitleArea();
00139 
00140             DrawTitle(title);
00141 
00142             int currentSelection = 0;
00143             ConsoleKey key;
00144             do
00145             {
00146                 Coordinate position =
00147                     new Coordinate(Console.WindowWidth - _contentWidth, Console.WindowHeight -
      _contentHeight);
00148 
00149                 for (int i = 0; i < options.Length; i++)
00150                 {
00151                     if (i == currentSelection)
00152                     {
00153                         Console.ForegroundColor = ConsoleColor.Red;
00154                     }
00155 
00156                     CenterString(options[i], position);
00157                     position.NextLine();
00158                     Console.ResetColor();
00159                 }
00160 
00161 
00162                 key = Console.ReadKey(true).Key;
00163 
00164                 switch (key)
00165                 {
00166                     case ConsoleKey.DownArrow:
00167                     {
00168                         if (currentSelection + 1 > options.Length - 1)
00169                         {
00170                             currentSelection = 0;
00171                         }
00172                         else
00173                         {
00174                             currentSelection += 1;
00175                         }
00176 
00177                         break;
00178                     }
00179 
00180                     case ConsoleKey.UpArrow:
00181                     {
00182                         if (currentSelection - 1 < 0)
00183                         {
00184                             currentSelection = options.Length - 1;
00185                         }
00186                         else
00187                         {
00188                             currentSelection -= 1;
00189                         }
00190 
00191                         break;
00192                     }
00193                 }
00194             } while (key != ConsoleKey.Enter);
```

```
00195
00196             return currentSelection;
00197         }
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- UserInterface.cs