NHZ Skeleton

2025.04.28

0.8.0

Készítette : Albitz Csanád

1. Hierarchikus mutató

1.1. Osztályhierarchia

Majdnem (de nem teljesen) betu″rendbe szedett leszármazási lista:

2. Osztálymutató

2.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

3. Fájlmutató

3.1. Fájllista

Az összes dokumentált fájl listája rövid leírásokkal:

4. Osztályok dokumentációja

4.1. Boss osztályreferencia

Publikus tagfüggvények

- **Boss** ()

  Alap konstruktor.
- Boss (Karakter &karakter, double special)

  Boss konstruktora karakter és speciális megadásával.
- Boss (const Boss &boss)

  Boss másoló konstruktor.
- void special (Karakter &k)

  Speciális támadás A megtámadott minion nem tud visszatámadni.
- std::string **getName** ()

Publikus tagfüggvények a(z) Karakter osztályból származnak

- **Karakter** ()

  Karakter alap konstruktora.
- Karakter (const Karakter &v)

  Karakter másoló konstruktor.
- Karakter (const char *nev, int mana, char ikon, double hp, double maxhp, bool aktiv)

  Karakter konstruktor adatokkal.
- virtual void sebzodik (double sebzes, Karakter *tamado)

  Támadás A karakterre meghatározott mértéku˝ sebzést okoz. A minion védelemmel csökkentheti a sebzo˝dést, de a boss nem.
- virtual void vedelemValt (double d)

  Védelem változtatása A védelem maximális értéke nincsen korlátozva, de negatívba nem mehet.
- bool regen (int hp)

  gyógyítás
- bool getAktiv ()

  Aktív lekérése.
- virtual void reaktiv ()

  Újra aktiválás Újra aktívra állítja a karaktert. Ezt minden kör végén meghívódik.
- Karakter & operator= (const Karakter &karakter)

  Értékadó operátor.
- void **halal** ()

  Halál Ha elfogyott az élete az adott karakternek, üres karakterré válik.
- double elet ()

  Élet lekérdezése pusztán a teszteléshez szükséges.

Publikus tagfüggvények a(z) Kartya osztályból származnak

- **Kartya** ()

  Kartya alap konstruktora.
- Kartya (const char *nev, int mana, char ikon)

  Konstruktor adattagokkal.
- Kartya (const Kartya &k)

  Másoló konstruktor.
- virtual bool kijatszas (int *mana, Kartya *kiv)

  Kártya kijátszása.
- Kartya & operator= (const Kartya &kartya)

  Értékadó operátor.
- void manaKiir (std::ostream &os) const

Mana kiírása.

- void nevKiir (std::ostream &os) const

  Név kiírása.

- void ikonKiir (std::ostream &os) const

  Ikon kiírása.

- virtual **~Kartya** ()

  Kártya destruktor

További örökölt tagok

Védett attribútumok a(z) Karakter osztályból származnak

- double **hp**
- double **maxHp**
- bool **aktiv**

Védett attribútumok a(z) Kartya osztályból származnak

- std::string **nev**
- int **manaKoltseg**
- char **ikon**

Statikus védett attribútumok a(z) Kartya osztályból származnak

- static const int **maxNevMeret**
- static const int **maxManaMeret**

4.1.1. Konstruktorok és destruktorok dokumentációja

Boss() [1/2]

Boss::Boss (
Karakter &*karakter*, double *special*)

Boss konstruktora karakter és speciális megadásával.

Paraméterek

karakter	

Boss() [2/2]

Boss::Boss (
const Boss & *boss*)

Boss másoló konstruktor.

Paraméterek

boss	másolandó

4.1.2. Tagfüggvények dokumentációja

special()

void Boss::special (
Karakter & *k*)

Speciális támadás A megtámadott minion nem tud visszatámadni.

Paraméterek

k	A célpont

4.2. Jatekos osztályreferencia

Publikus tagfüggvények

- **Jatekos** ()

  Játékos alap konstruktora.
- Jatekos (Boss boss, KartyaTarolo huzoPakli, KartyaTarolo kezPakli, KartyaTarolo MinionPakli, KartyaTarolo csomag, int maxMana)

  Játékos konstruktor betöltéshez.
- Jatekos (Boss boss, size_t MinionPakliMeret, size_t kezMeret, KartyaTarolo csomag, int maxMana)

  Játékos konstruktor Játék kezdetekor, használandó. Paraméterklnt a kötelezo˝ adatok vannak csak megadva.
- Jatekos (const Jatekos &j)

  Játékos másoló konstruktor.
- void **kezfeltolt** ()

  Kézben levo˝ kártyák feltöltése a húzópaklibóly.
- void **ujKor** ()

  Új kör indítása újra aktiválja a minionokat.
- void **huzopakliKever** ()

  Húzópakli keverés A csomag lapjait megkeveri és belerakja a húzópakliba. A megkevert pakliból feltölti a húzópaklit.
- bool Kijatszas (Kartya ∗k1, Kartya ∗k2)

  Kártya kijátszása.
- Boss & Getboss ()

  Boss getter.
- KartyaTarolo & getTarolo (TaroloTipus tipus)

  Tároló getter.

4.2.1. Konstruktorok és destruktorok dokumentációja

Jatekos() [1/3]

Jatekos::Jatekos (
Boss *boss*,
KartyaTarolo *huzoPakli*,
KartyaTarolo *kezPakli*, KartyaTarolo *MinionPakli*, KartyaTarolo *csomag*,
int *maxMana*)

Játékos konstruktor betöltéshez.

Paraméterek

boss	
huzoPakli	
kezPakli	
MinionPakli	
csomag	
maxMana	

Jatekos() [2/3]

Jatekos::Jatekos (
Boss *boss*,
size_t *MinionPakliMeret*, size_t *kezMeret*, KartyaTarolo *csomag*,
int *maxMana*)

Játékos konstruktor Játék kezdetekor, használandó. Paraméterklnt a kötelezo˝ adatok vannak csak megadva.

Paraméterek

boss	
MinionPakliMeret	
kezMeret	
csomag	
maxMana	

Jatekos() [3/3]

Jatekos::Jatekos (
const Jatekos & *j*)

Játékos másoló konstruktor.

Paraméterek

j	másolandó

4.2.2. Tagfüggvények dokumentációja

Getboss()

Boss & Jatekos::Getboss ()

Boss getter.

Visszatérési érték

getTarolo()

KartyaTarolo & Jatekos::getTarolo (
TaroloTipus *tipus*)

Tároló getter.

Paraméterek

tipus	

Visszatérési érték

Kijatszas()

bool Jatekos::Kijatszas (
Kartya * *k1*, Kartya *
k2)

Kártya kijátszása.

Paraméterek

k1	kijátszandó kártya
k2	ahova kijátszák a kártyát

Visszatérési érték

sikeres volt-e a kijátszás

Ez a dokumentáció az osztályról a következo˝ fájl alapján készült:

- jatekos.h

4.3. Karakter osztályreferencia

A Karakter osztály származási diagramja:

Publikus tagfüggvények

- **Karakter** ()

  Karakter alap konstruktora.
- Karakter (const Karakter &v)

  Karakter másoló konstruktor.
- Karakter (const char *nev, int mana, char ikon, double hp, double maxhp, bool aktiv)

  Karakter konstruktor adatokkal.
- virtual void sebzodik (double sebzes, Karakter *tamado)

  Támadás A karakterre meghatározott mértéku˝ sebzést okoz. A minion védelemmel csökkentheti a sebzo˝dést, de a boss nem.
- virtual void vedelemValt (double d)

  Védelem változtatása A védelem maximális értéke nincsen korlátozva, de negatívba nem mehet.

- bool regen (int hp)

  gyógyítás
- bool getAktiv ()

  Aktív lekérése.
- virtual void reaktiv ()

  Újra aktiválás Újra aktívra állítja a karaktert. Ezt minden kör végén meghívódik.
- Karakter & operator= (const Karakter &karakter)

  Értékadó operátor.
- void **halal** ()

  Halál Ha elfogyott az élete az adott karakternek, üres karakterré válik.
- double elet ()

  Élet lekérdezése pusztán a teszteléshez szükséges.

Publikus tagfüggvények a(z) Kartya osztályból származnak

- **Kartya** ()

  Kartya alap konstruktora.
- Kartya (const char *nev, int mana, char ikon)

  Konstruktor adattagokkal.
- Kartya (const Kartya &k)

  Másoló konstruktor.
- virtual bool kijatszas (int *mana, Kartya *kiv)

  Kártya kijátszása.
- Kartya & operator= (const Kartya &kartya)

  Értékadó operátor.
- void manaKiir (std::ostream &os) const

  Mana kiírása.
- void nevKiir (std::ostream &os) const

  Név kiírása.
- void ikonKiir (std::ostream &os) const

  Ikon kiírása.
- virtual **~Kartya** ()

  Kártya destruktor.

Védett attribútumok

- double **hp**
- double **maxHp**
- bool **aktiv**

Védett attribútumok a(z) Kartya osztályból származnak

- std::string **nev**
- int **manaKoltseg**
- char **ikon**

További örökölt tagok

Statikus védett attribútumok a(z) Kartya osztályból származnak

- static const int **maxNevMeret**
- static const int **maxManaMeret**

4.3.1. Konstruktorok és destruktorok dokumentációja

Karakter() [1/2]

Karakter::Karakter(
const Karakter & *v*)

Karakter másoló konstruktor.

Paraméterek

v	a másolandó karakter

Karakter() [2/2]

Karakter::Karakter(
const char * *nev*, int
mana,
char *ikon*, double
hp, double*maxhp*,
bool *aktiv*)

Karakter konstruktor adatokkal.

Paraméterek

nev	karakter neve
mana	karakter kijátszásához szükséges manaszint (egy bossbál ez elhanyagolható, mivel bossból csak egy van, ami a játék elejéto˝l kezdve létezik)
ikon	
hp	
maxhp	
aktiv	

4.3.2. Tagfüggvények dokumentációja

elet()

double Karakter::elet ()

Élet lekérdezése pusztán a teszteléshez szükséges.

Visszatérési érték

S karakter élete.

getAktiv()

bool Karakter::getAktiv ()

Aktív lekérése.

Visszatérési érték

aktív-e az állapot.

operator=()

Karakter & Karakter::operator= (
const Karakter & *karakter*)

Értékadó operátor.

Paraméterek

karakter	Amelyik karakter adatait másolja

Visszatérési érték

Aktuális objektum referencia

reaktiv()

virtual void Karakter::reaktiv () [virtual]

Újra aktiválás Újra aktívra állítja a karaktert. Ezt minden kör végén meghívódik. Újraimplementáló leszármazottak: Minion.

regen()

bool Karakter::regen (
int *hp*)

gyógyítás

Paraméterek

hp	gyógyítás mértéke

Visszatérési érték

sebzodik()

virtual void Karakter::sebzodik (
double *sebzes*,
Karakter * *tamado*) [virtual]

Támadás A karakterre meghatározott mértéku″ sebzést okoz. A minion védelemmel csökkentheti a sebzo″dést, de a boss nem.

Paraméterek

sebzes	a sebzés mértéke
tamado	a támadó

Újraimplementáló leszármazottak: Minion.

vedelemValt()

virtual void Karakter::vedelemValt (
double *d*) [virtual]

Védelem változtatása A védelem maximális értéke nincsen korlátozva, de negatívba nem mehet.

Paraméterek

d	a védelem mértékét megváltoztatásának mértéke

Újraimplementáló leszármazottak: Minion.

Ez a dokumentáció az osztályról a következo˝ fájl alapján készült:

- karakter.h

4.4. Kartya osztályreferencia

Publikus tagfüggvények

- **Kartya** ()

  Kartya alap konstruktora.
- Kartya (const char *nev, int mana, char ikon)

  Konstruktor adattagokkal.
- Kartya (const Kartya &k)

  Másoló konstruktor.
- virtual bool kijatszas (int *mana, Kartya *kiv)

  Kártya kijátszása.
- Kartya & operator= (const Kartya &kartya)

  Értékadó operátor.
- void manaKiir (std::ostream &os) const

  Mana kiírása.
- void nevKiir (std::ostream &os) const

  Név kiírása.
- void ikonKiir (std::ostream &os) const

  Ikon kiírása.
- virtual ~**Kartya** ()

  Kártya destruktor.

Védett attribútumok

- std::string **nev**
- int **manaKoltseg**
- char **ikon**Statikus védett attribútumok
- static const int **maxNevMeret**
- static const int **maxManaMeret**

4.4.1. Konstruktorok és destruktorok dokumentációja

Kartya() [1/2]

Kartya::Kartya (
const char * *nev*, int
mana,
char *ikon*) Konstruktor

adattagokkal. **Paraméterek**

nev	A kártya neve
mana	A kártya kijátszásának manaköltsége
ikon	A karakter megjelenítése során megjeleno˝ ikon

Kartya() [2/2]

Kartya::Kartya (
const Kartya & *k*)

Másoló konstruktor.

Paraméterek

k	A másolandó objektum

4.4.2. Tagfüggvények dokumentációja

ikonKiir()

void Kartya::ikonKiir (
std::ostream & *os*) const

Ikon kiírása.

Paraméterek

os	Az output stream ahova kiírja

kijatszas()

virtual bool Kartya::kijatszas (
int * *mana*,
Kartya * *kiv*) [virtual]

Kártya kijátszása.Paraméterek

mana	A jelenleg rendelkezésre álló mana mennyisége.
kiv	A kiválasztott célpont kártya, amire kifejti hatását.

Visszatérési érték

Kijátszható-e a rendelkezésre álló manából.

Újraimplementáló leszármazottak: Minion.

manaKiir()

void Kartya::manaKiir (
std::ostream & *os*) const

Mana kiírása.

Paraméterek

os	Az output stream ahova kiírja

nevKiir()

void Kartya::nevKiir (
std::ostream & *os*) const

Név kiírása.

Paraméterek

os	Az output stream ahova kiírja

operator=()

Kartya & Kartya::operator= (
const Kartya & *kartya*)

Értékadó operátor.

Paraméterek

kartya	Amelyik kártya adatait másolja

Visszatérési érték

Aktuális objektum referencia

Ez a dokumentáció az osztályról a következo˝ fájl alapján készült:

- kartya.h

4.5. KartyaTarolo osztályreferencia

Publikus tagfüggvények

- **KartyaTarolo** ()
  Tároló alapkonstruktora.
- KartyaTarolo (size_t kapacitas)
  Kártya tároló kapacítás szerinti konstruktora.
- KartyaTarolo (const KartyaTarolo &t)
  Kártyatároló másolókonstruktora.
- void randomBeszur (Kartya ∗kartya)
  Feltöltés A kártyapakli összekeveréséért felel.
- void berak (Kartya ∗kartya, size_t index)
  Soros feltöltés Csak sorba feltöltött tároló esetén mu˝ködik (pl fájlbetöltésnél)
- Kartya ∗ kihuz (size_t index)
  Kihúzás megadott indexro˝l.
- size_t getMeret () const
  Tárolóban levo˝ tényleges elemek számának lekérése.
- size_t getKapacitas () const
  Tároló kapacításának lekérése.
- Kartya ∗ operator[ ] (size_t index)
  Indexelo˝ operátor.
- void **kiurites** ()
  Tároló ürítése.
- **~KartyaTarolo** ()
  Tároló destruktora.

4.5.1. Konstruktorok és destruktorok dokumentációja

KartyaTarolo() [1/2]

KartyaTarolo::KartyaTarolo (
size_t *kapacitas*)

Kártya tároló kapacítás szerinti konstruktora.

Paraméterek

kapacitas	Az a méret amekkorára szükség lesz.

KartyaTarolo() [2/2]

KartyaTarolo::KartyaTarolo (
const KartyaTarolo & *t*)

Kártyatároló másolókonstruktora.

Paraméterek

a	másolandó tároló referenciája

4.5.2. Tagfüggvények dokumentációja

berak()

void KartyaTarolo::berak (
Kartya * *kartya*, size_t
index)

Soros feltöltés Csak sorba feltöltött tároló esetén mu˝ködik (pl fájlbetöltésnél)

Paraméterek

kartya	A következo˝ elem
index	A hely ahova berakja az adott elemet

getKapacitas()

size_t KartyaTarolo::getKapacitas () const

Tároló kapacításának lekérése.

Visszatérési érték

Tároló kapacítása

getMeret()

size_t KartyaTarolo::getMeret () const

Tárolóban levo˝ tényleges elemek számának lekérése.

Visszatérési érték

Ezeknek a száma

kihuz()

Kartya * KartyaTarolo::kihuz (
size_t *index*)

Kihúzás megadott indexro˝l.

Paraméterek

index	A megadott index

Visszatérési érték

A kihúzott kártya adatai.

operator[]()

Kartya * KartyaTarolo::operator[] (
size_t *index*)

Indexelo˝ operátor.

Paraméterek

index	

Visszatérési érték

randomBeszur()

void KartyaTarolo::randomBeszur (
Kartya * *kartya*)

Feltöltés A kártyapakli összekeveréséért felel.

Paraméterek

Kartya	A behelyezendo˝ kártya

Ez a dokumentáció az osztályról a következo˝ fájl alapján készült:

- kartyaTarolo.h

4.6. Kurzor osztályreferencia

Publikus tagfüggvények

- **Kurzor** ()

  Kurzor alapkonstruktor.
- Kurzor (const Jatekos &p1, const Jatekos &p2, int jatekos, int fazis)

  Kurzor konstruktor adattagokkal.
- void lepes (irany ir)

  Lépés kezelése A feladata vezérelni a fazis1Lepes és fazis2Lepes függvényt.
- void fazis1Lepes (irany ir)

  Elso˝ fázis mozgás Ekkor történik akártyák kijátszása a kézbo˝l.
- void fazis2Lepes (irany ir)

  Második fázis mozgás Ekkor történik a minionokkal illetve a bossal való támadás.
- void **kivalaszt** ()

  Kiválaszt Akkor hívódik meg, amikor a felhasználó arra a kártyára mozgatta a kurzort, amelyiket ki akarja választani.
- void **kovFazis** ()

  Következo˝ fázis Akkor hívódik meg, amikor a felhasználó már nem kíván több dolgot csinálni az aktuális fázisban.
- Jatekos & aktJatekos ()

  Aktuális játékos getter.

4.6.1. Konstruktorok és destruktorok dokumentációja

Kurzor()

Kurzor::Kurzor (
const Jatekos & *p1*, const Jatekos & *p2*, int *jatekos*, int *fazis*)

Kurzor konstruktor adattagokkal.

Paraméterek

p1	egyes számú játékos
p2	kettes számú játékos
jatekos	melyik játékos van éppen körön
fazis	a játékos melyik fázisban van

4.6.2. Tagfüggvények dokumentációja

aktJatekos()

Jatekos & Kurzor::aktJatekos ()

Aktuális játékos getter.

Visszatérési érték

Aktuális játékos referencia

fazis1Lepes()

void Kurzor::fazis1Lepes (
irany *ir*)

Elso″ fázis mozgás Ekkor történik akártyák kijátszása a kézbo″l.

Paraméterek

ir	inputnak kapott lépés irány

fazis2Lepes()

void Kurzor::fazis2Lepes (
irany *ir*)

Második fázis mozgás Ekkor történik a minionokkal illetve a bossal való támadás.

Paraméterek

ir	inputnak kapott lépés irány

lepes()

void Kurzor::lepes (
irany *ir*)

Lépés kezelése A feladata vezérelni a fazis1Lepes és fazis2Lepes függvényt.

Paraméterek

ir	inputnak kapott lépés irány

Ez a dokumentáció az osztályról a következo″ fájl alapján készült:

- kurzor.h

4.7. Minion osztályreferencia

Publikus tagfüggvények

- **Minion** ()

  Minion alapkonstruktora.
- Minion (Karakter &k, double ero)

  Minion konstruktora karakter és ero˝ megadásával.
- Minion (Minion &m)

  Minion konstruktora minion referenciával.
- Minion & operator= (const Minion &minion)

  Értékadó operátor.
- void sebzodik (double sebzes, Karakter ∗tamado) override

  Sebzo˝dés Ellentétben a bossal, a minion rendelkezik védelemmel, amely csökkenti a rá kijátszott sebzés mértékét. A minion (hacsak nem halott) vissza is tud támadni.
- void tamadas (Karakter ∗celpont)

  Karakter megtámadása Megtámadja a kiválasztott karaktert.
- void vedelemValt (double d)

  Védelem változtatása A támadások esetén elo˝ször a védelem csökken és csak aztán az élets.
- bool kijatszas (int ∗mana, Kartya ∗kiv)

  Minion lehelyezése.
- void reaktiv ()

  A minion reaktiválása A minion minden újraaktiválásnál elveszíti a védelmét is.
- double minionVedelem ()

  Védelem getter Tesztekzez használt getter.
- double minionhp ()

  élet getter Tesztekzez használt getter

Publikus tagfüggvények a(z) Karakter osztályból származnak

- **Karakter** ()

  Karakter alap konstruktora.
- Karakter (const Karakter &v)

  Karakter másoló konstruktor.
- Karakter (const char ∗nev, int mana, char ikon, double hp, double maxhp, bool aktiv)

  Karakter konstruktor adatokkal.
- bool regen (int hp)

  gyógyítás
- bool getAktiv ()

  Aktív lekérése.
- Karakter & operator= (const Karakter &karakter)

  Értékadó operátor.
- void **halal** ()

  Halál Ha elfogyott az élete az adott karakternek, üres karakterré válik.
- double elet ()

  Élet lekérdezése pusztán a teszteléshez szükséges.

Publikus tagfüggvények a(z) Kartya osztályból származnak

- **Kartya** ()

  Kartya alap konstruktora.
- Kartya (const char ∗nev, int mana, char ikon)

  Konstruktor adattagokkal.
- Kartya (const Kartya &k)

  Másoló konstruktor.
- Kartya & operator= (const Kartya &kartya)

Értékadó operátor.

- void manaKiir (std::ostream &os) const

  Mana kiírása.
- void nevKiir (std::ostream &os) const

  Név kiírása.
- void ikonKiir (std::ostream &os) const

  Ikon kiírása.
- virtual **~Kartya** ()

  Kártya destruktor.

További örökölt tagok

Védett attribútumok a(z) Karakter osztályból származnak

- double **hp**
- double **maxHp**
- bool **aktiv**

Védett attribútumok a(z) Kartya osztályból származnak

- std::string **nev**
- int **manaKoltseg**
- char **ikon**

Statikus védett attribútumok a(z) Kartya osztályból származnak

- static const int **maxNevMeret**
- static const int **maxManaMeret**

4.7.1. Konstruktorok és destruktorok dokumentációja

Minion() [1/2]

Minion::Minion (
Karakter & *k*, double *ero*)

Minion konstruktora karakter és ero″ megadásával.

Paraméterek

k	
ero	

Minion() [2/2]

Minion::Minion (
Minion & *m*)

Minion konstruktora minion referenciával.

4.7.2. Tagfüggvények dokumentációja

kijatszas()

bool Minion::kijatszas (
int * *mana*,
Kartya * *kiv*) [virtual]

Minion lehelyezése

Paraméterek

mana	
kiv	Az az üres karakterlap, ahova

Újraimplementált o″sök: Kartya.

minionhp()

double Minion::minionhp ()

élet getter Tesztekzez használt getter

Visszatérési érték

minion élete

minionVedelem()

double Minion::minionVedelem ()

Védelem getter Tesztekzez használt getter.

Visszatérési érték

védelem

operator=()

Minion & Minion::operator= (
const Minion & *minion*)

Értékadó operátor.

Paraméterek

minion	Amelyik minion adatait másolja

Visszatérési érték

Aktuális objektum referencia

reaktiv()

void Minion::reaktiv () [virtual]

A minion reaktiválása A minion minden újraaktiválásnál elveszíti a védelmét is. Újraimplementált o″sök: Karakter.

sebzodik()

void Minion::sebzodik (
double *sebzes*,
Karakter * *tamado*) [override], [virtual]

Sebzo″dés Ellentétben a bossal, a minion rendelkezik védelemmel, amely csökkenti a rá kijátszott sebzés mértékét. A minion (hacsak nem halott) vissza is tud támadni.

Paraméterek

sebzes	A minionra kifejtett sebzés mértéke
tamado	A karakter, aki megtámadta az adott miniont.

Újraimplementált o″sök: Karakter.

tamadas()

void Minion::tamadas (
Karakter * *celpont*)

Karakter megtámadása Megtámadja a kiválasztott karaktert.

Paraméterek

celpont	A célpont

vedelemValt()

void Minion::vedelemValt (
double *d*) [virtual]

Védelem változtatása A támadások esetén elo″ször a védelem csökken és csak aztán az élets.

Paraméterek

d	a védelem megváltoztatásának mértéke

Visszatérési érték

sikeres volt-e a védelem változtatása

Újraimplementált o″sök: Karakter.

Ez a dokumentáció az osztályról a következo″ fájl alapján készült:

- minion.h

4.8. Varazslat osztályreferencia

A Varazslat osztály származási diagramja:

Publikus tagfüggvények

- **Varazslat** ()
  Alap konstruktor.
- Varazslat (Varazslat &v)
  Másoló konstruktor.
- Varazslat (const char *nev, int mana, char ikon, double sebzes, double gyogyitas, double vedelem)
  Konstruktor adattagokkal.
- bool kijatszas (int *mana, Karakter &kiv)
  kijátszás

Publikus tagfüggvények a(z) Kartya osztályból származnak

- **Kartya** ()
  Kartya alap konstruktora.
- Kartya (const char *nev, int mana, char ikon)
  Konstruktor adattagokkal
  Kartya (const Kartya &k)
  Másoló konstruktor.
- virtual bool kijatszas (int *mana, Kartya *kiv)
  Kártya kijátszása.
- Kartya & operator= (const Kartya &kartya)
  Értékadó operátor.
- void manaKiir (std::ostream &os) const
  Mana kiírása.
- void nevKiir (std::ostream &os) const
  Név kiírása.
- void ikonKiir (std::ostream &os) const
  Ikon kiírása.
- virtual **~Kartya** ()
  Kártya destruktor.

További örökölt tagok

Védett attribútumok a(z) Kartya osztályból származnak

- std::string **nev**
- int **manaKoltseg**
- char **ikon**

Statikus védett attribútumok a(z) Kartya osztályból származnak

- static const int **maxNevMeret**
- static const int **maxManaMeret**

4.8.1. Konstruktorok és destruktorok dokumentációja

Varazslat() [1/2]

Varazslat::Varazslat(
Varazslat & *v*)

Másoló konstruktor.

Paraméterek

v	másolandó objektum

Varazslat() [2/2]

Varazslat::Varazslat(
const char * *nev*, int
mana,
char *ikon*, double
sebzes, double*gyogyitas*,
double *vedelem*)

Konstruktor adattagokkal.

Paraméterek

nev	Varázslat neve
mana	Varázslat manaköltsége
ikon	Varázslat ikonja
sebzes	Varázslat sebzése
gyogyitas	Varázslat gyógyítása
vedelem	Varázslat védelem növelése

4.8.2. Tagfüggvények dokumentációja

kijatszas()

bool Varazslat::kijatszas(
int * *mana*, Karakter &
kiv)

kijátszás

Paraméterek

mana	A rendelkezésre álló mana mennyisége
kiv	a célpont

Visszatérési érték

skeres volt-e a kijátszás

Ez a dokumentáció az osztályról a következo˝ fájl alapján készült:

- varazslat.h

5. Fájlok dokumentációja

5.1. boss.h fájlreferencia

A játékosok fo″karaktere.

#include "karakter.h"
A boss.h definíciós fájl függési gráfja:

Osztályok

- class Boss

  Részletes leírás

A játékosok fo″karaktere.

Ezen karakter védelme a legfontosabb a játékban. Az a játékos veszít, akié meghal.

Szerzo″

Albitz Csanád

Dátum

2025-04-20

5.2. boss.h

Ugrás a fájl dokumentációjához.

```
00001
00011 #ifndef BOSS_H
00012 #define BOSS_H
00013 #include "karakter.h"
00014
00015 class Boss:public Karakter{ 00016
        double specialSebzes;
00017 public:
00019         Boss();
00022         Boss(Karakter&karakter,doublespecial); 00025
              Boss(const Boss &boss);
00026
00030         voidspecial(Karakter& k); 00031
00032         std::stringgetName(){returnthis->nev;} 00033 };
00034
00035 #endif
```

5.3. jatekos.h fájlreferencia

A két játékos adatait tároló osztály. Ő felel a játékosok személyes kártyáiért. A játéktáblán két játékos helyezkedik el, szimmetrikusan. Minden játékosnak van egy húzópaklija, egy minionokat tároló paklija, amiben a lehelyezett minionok vannak. A csomag az amibo˝ l az osztály felépíti a húzo˝ paklit. Minden körben meghatározott mennyiségu˝ mana áll rendelkezésére a játékosoknak. Ez minden körben egyel no˝ .

kurzor.h

#include "kartya.h" #include "boss.h"

#include "kartyaTarolo.h"

Osztályok

- class Jatekos

Enumerációk

- enum class TaroloTipus { **Kez** , **Huzo** , **Minionok** }

  A tároló kiválasztásában segít.

5.3.1. Részletes leírás

A két játékos adatait tároló osztály. Ő felel a játékosok személyes kártyáiért. A játéktáblán két játékos helyezkedik el, szimmetrikusan. Minden játékosnak van egy húzópaklija, egy minionokat tároló paklija, amiben a lehelyezett minionok vannak. A csomag az amibo˝ l az osztály felépíti a húzo˝ paklit. Minden körben meghatározott mennyiségu˝ mana áll rendelkezésére a játékosoknak. Ez minden körben egyel no˝ .

Szerző

Albitz Csanád

Dátum

2025-04-20

5.4. jatekos.h

Ugrás a fájl dokumentációjához.

```
00001
00016 #ifndef JATEKOS_H
00017 #define JATEKOS_H
00018
00019 #include "kartya.h"
00020 #include "boss.h"
00021 #include "kartyaTarolo.h"
00022
00024 enum class TaroloTipus { 00025
        Kez,
00026   Huzo,
00027   Minionok
00028 };
00029
00030 class Jatekos{
00032       int maxMana;
00034       int mana;
00036       Boss boss;
00037       KartyaTarolo kez;
00039       KartyaTarolo huzo;
00041       KartyaTarolo minionok;
00043       KartyaTarolo csomag;
00044
00045 public:
00047       Jatekos();
00055       Jatekos(Boss boss,KartyaTarolo huzoPakli,KartyaTarolo kezPakli,KartyaTarolo MinionPakli,KartyaTarolo
      csomag,int maxMana);
00063       Jatekos(Boss boss,size_t MinionPakliMeret,size_t kezMeret,KartyaTarolo csomag,int maxMana); 00066       Jatekos(const Jatekos&
j);
00067
00069       void kezfeltolt();
00070
00073       void ujKor();
00077       void huzopakliKever();
00078
00083       bool Kijatszas(Kartya* k1,Kartya* k2); 00084
00087   Boss& Getboss();
00088
00092   KartyaTarolo& getTarolo(TaroloTipus tipus); 00093
00094
00095 };
00096 #endif
```

5.5. karakter.h fájlreferencia

Karakterek.

#include "kartya.h"
A karakter.h definíciós fájl függési gráfja:
Osztályok

- class Karakter

5.5.1. Részletes leírás

Karakterek.

A karakterek definíciója. Ez egy absztrakt osztály, mely elo˝segíti a minion és boss definiálását.

Szerzo˝

Albitz Csanád

Dátum

2025-04-20

5.6. karakter.h

Ugrás a fájl dokumentációjához.

```
00001
00010 #ifndef KARAKTER_H
00011 #define KARAKTER_H
00012
00013 #include "kartya.h"
00014
00015 class Karakter : public Kartya 00016 {
00017 protected:
00018         double hp;
00019         double maxHp;
00020         bool aktiv;
00021
00022 public:
00024         Karakter();
00027         Karakter(const Karakter &v);
00036         Karakter(const char *nev, int mana, char ikon, double hp, double maxhp, bool aktiv); 00041                 virtual void
sebzodik(double sebzes,Karakter* tamado);
00045         virtual void vedelemValt(double d);
00049         bool regen(int hp);
00052         bool getAktiv();
00055         virtual void reaktiv();
00059         Karakter &operator=(const Karakter &karakter); 00062             void
halal();
00063
00067         double elet();
00068 };
00069 #endif
```

5.7. kartya.h fájlreferencia

Kartya osztály.

#include <string>
#include <iostream>
Osztályok

- class Kartya

5.7.1. Részletes leírás

Kartya osztály.

Absztrakt o˝s, amely tartalmazza a minden kártya általános adatait.

Szerzo˝

Albitz Csanád

Dátum

2025-04-16

5.8. kartya.h

Ugrás a fájl dokumentációjához.

```
00001
00010 #ifndef KARTYA_H
00011 #define KARTYA_H
00012
00013 #include <string>
00014 #include <iostream>
00015
00016 class Kartya
00017 {
00018 protected:
00019         const static int maxNevMeret; 00020
00021         const static int maxManaMeret; 00022
00023         std::string nev;
00024
00025         int manaKoltseg;
00026
00027         char ikon;
00028
00029 public:
00031         Kartya();
00036         Kartya(const char *nev, int mana, char ikon); 00039
              Kartya(const Kartya &k);
00044         virtual bool kijatszas(int* mana,Kartya* kiv); 00048        Kartya
&operator=(const Kartya &kartya);
00049
00052         void manaKiir(std::ostream& os)const; 00055 void
nevKiir(std::ostream& os) const; 00058          void
ikonKiir(std::ostream& os) const; 00059
00061         virtual ~Kartya();
00062
00063 };
00064 #endif
```

5.9. kartyaTarolo.h fájlreferencia

Osztályok

- class KartyaTarolo

5.9.1. Részletes leírás

Kártya tároló osztály.

A minionok, a húzópakli és a kézben levo″ paklit tároló heterogén kollekcióért felelo″s osztály A tároló mu″ködéséhez szükséges, hogy annyi elemmel hozzák létre, amennyit tárolni szeretne

Szerzo″

Albitz Csanád

Dátum

2025-04-20

5.10. kartyaTarolo.h

Ugrás a fájl dokumentációjához.

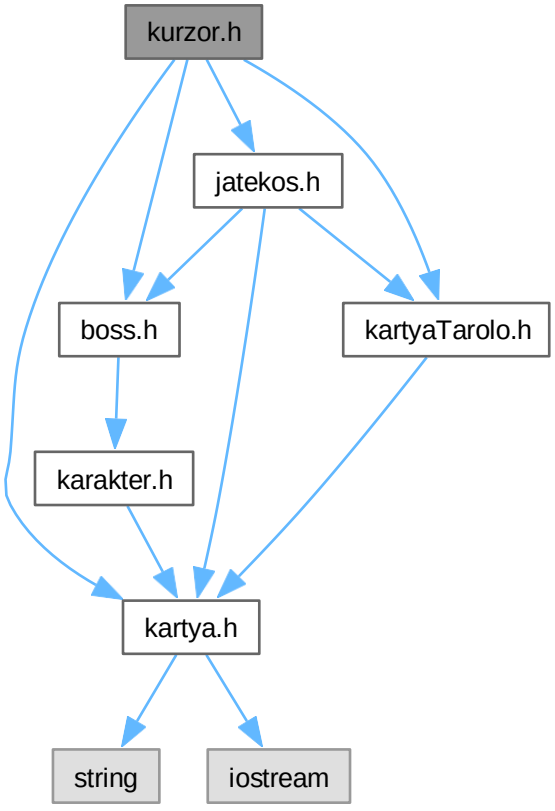
```
00001
00010 #ifndef KARTYATAROLO_H
00011 #define KARTYATAROLO_H
00012
00013 #include "kartya.h"
00014
00015 class KartyaTarolo{
00016         Kartya** tomb;
00017         size_t kapacitas;
00018         size_t meret;
00019 public:
00021         KartyaTarolo();
00024         KartyaTarolo(size_t kapacitas); 00027
         KartyaTarolo(const KartyaTarolo& t); 00031    void
randomBeszur(Kartya* kartya);
00036         void berak(Kartya* kartya,size_t index); 00040 Kartya*
kihuz(size_t index);
00043         size_t getMeret() const;
00046         size_t getKapacitas()const;
00050         Kartya* operator[](size_t index); 00052
         void kiurites();
00054         ~KartyaTarolo();
00055 };
00056 #endif
```

kurzor.h fájlreferencia

Kurzor osztály.

#include "kartya.h" #include "boss.h" #include"kartyaTarolo.h" #include "jatekos.h"

A kurzor.h definíciós fájl függési gráfja:



Osztályok

- class Kurzor

Enumerációk

- enum **irany** { **jobbra** , **balra** , **fel** , **le** }

5.10.1. Részletes leírás

Kurzor osztály.

Ő felel a felhasználótól érkezo˝ inputok feldolgozásáért. Nyomonköveti a játék állását. Egyszerre funkcionál játék managerként és inputmanagerként.

Szerzo˝

Albitz Csanád

Dátum

2025-04-20

5.11. kurzor.h

Ugrás a fájl dokumentációjához.

```
00001
00012 #ifndef KURZOR_H
00013 #define KURZOR_H
00014
00015 #include "kartya.h"
00016 #include "boss.h"
00017 #include "kartyaTarolo.h"
00018 #include "jatekos.h"
00019
00020 enum irany
00021 {
00022         jobbra,
00023         balra,
00024         fel,
00025         le
00026 };
00027
00028 class Kurzor
00029 {
00030         Jatekos p1;
00031         Jatekos p2;
00032         Kartya *sel1;
00033         Kartya *sel2;
00036         Kartya *mov;
00039         int jatekos;
00040         int fazis;
00047         int selSzint;
00048
00049 public:
00051         Kurzor();
00057         Kurzor(const Jatekos& p1,const Jatekos& p2,int jatekos,int fazis); 00061 void lepes(irany ir);
00065    void   fazis1Lepes(irany   ir);   00069    void
fazis2Lepes(irany ir); 00072      void kivalaszt();
00075      void kovFazis();
00078      Jatekos& aktJatekos();
00079 };
00080
00081 #endif
```

5.12. minion.h fájlreferencia

Osztályok

- class Minion

5.12.1. Részletes leírás

Minion osztály.

Minionok viselkedését írja le. A minionok az asztalon elhelyezkedo˝ (vagy még kártyaként jelenlevo˝) karakterek.

Szerzo˝

Albitz Csanád

Dátum

2025-04-20

5.13. minion.h

Ugrás a fájl dokumentációjához.

```
00001
00010 #ifndef MINION_H
00011 #define MINION_H
00012
00013 #include "karakter.h"
00014
00015 class Minion : public Karakter 00016 {
00017 private:
00018         double ero;
00019         double vedelem;
00020
00021 public:
00023         Minion();
00027         Minion(Karakter &k, double ero); 00030
              Minion(Minion &m);
00031
00035         Minion &operator=(const Minion &minion); 00036
00042         void sebzodik(double sebzes, Karakter *tamado) override; 00046                 void
tamadas(Karakter *celpont);
00051         void vedelemValt(double d);
00055         bool kijatszas(int *mana, Kartya *kiv); 00058         void
reaktiv();
00059
00060
00064         double minionVedelem();
00068         double minionhp();
00069 };
00070
00071 #endif
```

5.14. varazslat.h fájlreferencia

Varázslat osztály.

#include "kartya.h" #include "karakter.h"

Osztályok

- class Varazslat

5.14.1. Részletes leírás

Varázslat osztály.

Kartya egy alosztálya. Célja az egyszeri hatású mágiák definiálása.

Szerző

Albitz Csanád

Dátum

2025-04-20

5.15. varazslat.h

Ugrás a fájl dokumentációjához.

```
00001
00010 #ifndef VARAZSLAT_H
00011 #define VARAZSLAT_H
00012
00013 #include "kartya.h"
00014 #include "karakter.h"
00015
00016 class Varazslat:public Kartya{ 00017
                double sebzes;
00018           double gyogyitas;
00019           double vedelem;
00020 public:
00022           Varazslat();
00025           Varazslat(Varazslat& v);
00033           Varazslat(const char *nev, int mana, char ikon,double sebzes,double gyogyitas,double vedelem); 00038          bool kijatszas(int*
mana,Karakter& kiv);
00039 };
00040
00041 #endif
```

Tárgymutató