Snake

Programozás alapjai 3. - házi feladat

Mészáros Krisztina

NQC2BA

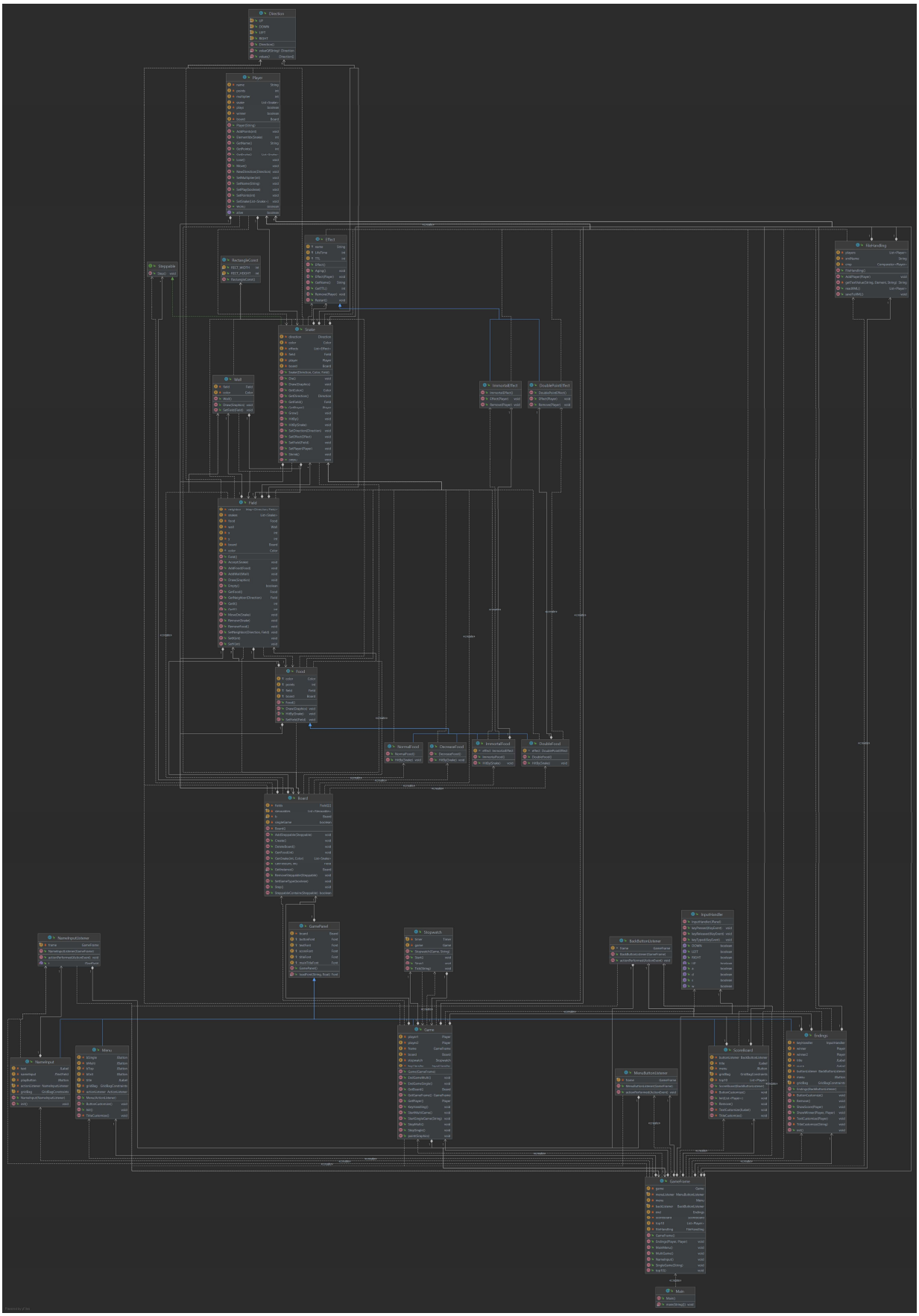
1 FELHASZNÁLÓI KÉZIKÖNYV

A játék a klasszikus kígyójáték idegen nevén a Snake, melyet szinte mindenki ismer, hiszen már a régebbi nyomógombos telefonokra is rég megjelent. A játék főszereplője egy kígyó melyet a játékos irányít, a célja pedig, hogy minél több ételt szedjen fel a pályáról, melyek véletlenszerű helyeken jelennek meg.

A kígyó minden egyes összeszedett étel után nőni fog eggyel. A kihívás a játékban, hogy a játékosnak mindeközben el kell kerülnie azt, hogy saját magába vagy a pálya szélén a falba harapjon. Hogy egy kicsit megcsavarjam a dolgokat, időről-időre megjelennek különleges kaják is, amik befolyásolják a játék menetét: az egyik csökkenti a kígyó hosszát, a másik egy rövid időre lehetővé teszi, hogy ne tudjon magába harapni, illetve egy harmadik hatására duplázódni fog egy rövid időre a továbbiakban felszedett csalik pontértéke.

Egyjátékos módban értelemszerűen a nyilak segítségével lehetséges irányítani a kígyót a megjelenő pályán.

2 OSZTÁLYDIAGRAM



3 STRUKTURÁLIS LEÍRÁS

3.1 AZ OSZTÁLYOK LEÍRÁSA

3.1.1 GameFrame

Felelősségek

Az osztály egyetlen felelőssége az ablak kezelése: méretei(a hozzáadott JPanelhez igazodik), láthatósága, felosztása (vagy épp a nem felosztása).

Attribútumok

- Game: game	A játék maga.
- MenuButtonListener: menuListener	A neve is adja, a menü gombjait kezeli.
- Menu: menu	A menü megjelenéséért felel.
- BackButtonListener: backListener	A visszagomb(oka)t kezeli, van ilyen a toplistánál, és minden játék végén.
- Endings: end	A játék végén megjelenő „képért” felel.
- ScoreBoard: scoreBoard	A toplista megjelenéséért felel.
- List<Player>: top10	A beolvasott, és később megjelenített legjobb 10.
- FileHandling: fileHandling	A fájlkezeléséért felel (beolvasás, és kimentés).

Metódusok

+ Endings(p1: Player, p2: Player)	A végefeliratot intézi.
+ MainMenu()	A főmenüt kezeli.
+ MultiGame()	A kétjátékos módot kezeli.
+ NameInput()	Egyjátékos mód esetén a név bevitelét kezeli.
+ SingleGame(s: String)	Egyjátékos módot kezeli, átadja a bevitt nevet.
+ top10()	A toplista megjelenítését kezeli.

3.1.2 Game

Felelősségek

Feladata a játék(ok) menetének levezénylése, elindítja, futtatja és befejezi a játéko(ka)t, illetve futás közben egy másik osztály (InputHandler) segítségével kezeli a billentyűket.

Attribútumok

- InputHandler: keyHandler	A billentyűket kezelő listener.
- Player: player1	A játékos, ő semiképp nem null.
- Player: player2	Ő csak egyjátékos mód esetén null.
- GameFrame: frame	A játék ablaka, vele lehet meghívni, majd az vége címet.
- Board: board	Az első és egyetlen eredeti példánya a pályának, segítségével lehet majd legenerálni, és lépni.

Metódusok

+ EndGameMulti()	Befejezi a Multi Game-t, megállítja a stoppert, meghívja az Ending-et.

+ EndGameSingle()	Befejezi a Single Game-t.
+ StartSingleGame(s: String)	Elindítja a játékot, inicializálja a további szükséges változókat, pl. az átvett nevet is beállítja a játékosnak.
+ StartMultiGame(s: String)	Ugynaz, mint a Single-nél csak a duplája, és nincs név.
+ StepSingle()	Az óra ütésére (Tick) megnézi a lenyomott billentyűt, közvetetten továbblépteti a szükséges dolgokat.
+ StepMulti()	Ugyanaz, mint az előbbi csak két játékossal.
+ KeyHandling()	Kezeli a nyilakat, és a „w, a, s, d” gombokat.
+ paint(g: Graphics)	Egy Override, bejárja a mezőket és a koordinátáknak megfelelően kirajzolja a kockákat.

3.1.3 FileHandling

Felelősségek

Feladata a fájl beolvasása, és az adatok kimentése a program futása során többször is.

Attribútumok

- List<Player>: players	Az aktuális 10 legjobb listája, pontszám szerint csökkenő sorrendbe rendezve.
- String: xmlName	A fájl neve (a beolvasás és a kiírás ugyanazzal történik).
- Comparator<Player>: cmp	Egy komparátor, ami csökkenő sorrendbe teszi a játékosok listáját pontszámuk alapján.

Metódusok

+ AddPlayer(p: PLayer)	Megvizsgálja, hogy a kapott játékos felkerülhet-e a listára, és ha igen, akkor újra rendezi a listát és kimenti.
+ readXML()	Nevéből adódóan is, beolvassa a fájlt és az elemeket bepakolja a listába.
+ saveToXML()	Kimenti a lista elemeit a fájlba.
+ getTextValue(s1 String, e Element, s2 String)	Beveszi a kellő elemet, a tagnek megfelelően.

3.1.4 InputHandler

Felelősségek

Feladata a billentyűk kezelése.

Attribútumok

- UP: boolean	Fent-e a legutóbb lenyomott, gomb.
- DOWN: boolean	Lenti-e.
- LEFT: boolean	Bal-e.
- RIGHT: boolean	Vagy jobb.
- W: boolean	W-e (kétjátékos mód esetén)
- A: boolean	A.
- S: boolean	S.
- D: boolean	Vagy D.

Metódusok

+ isUp(): boolean	Visszaadja az UP értékét.
+ isDown(): boolean	Visszaadja az DOWN értékét.

+ isLeft(): boolean	Visszaadja az LEFT értékét.
+ isRight(): boolean	Visszaadja az RIGHT értékét.
+ isW(): boolean	Visszaadja az W értékét.
+ isS(): boolean	Visszaadja az S értékét.
+ isA(): boolean	Visszaadja az A értékét.
+ isD(): boolean	Visszaadja az D értékét.
+ keyPressed()	Beállítja az osztály változóinak a megfelelő értékeket a lenyomott billentyűknek megfelelően.

3.1.5 Stoppwatch

Felelősségek

Méri az időt, a fő feladata pedig, hogy adott időközönként „üt” egyet, amire minden léptethető dolog lépni fog majd.

Attribútumok

- timer: Timer	Az időzítő.
- game: Game	Konstruktorában veszi majd át, így tudja majd kifejteni rajta a Tick() a hatását.

Metódusok

+ Tick(s: String)	Meghívja adott időközönként a game megfelelő Step() függvényét.
+ Start()	Elindítja az időzítőt.
+ Stop()	Megállítja az időzítőt.

3.1.6 Board

Felelősségek

A tábla osztály, ahogy a neve is adja, felel mindenért, ami rajta történik. Tárolja a mezőket egy tömben, hogy könnyen hozzáférhető legyen. Itt generálódik le a pálya, rá a kígyó és az étkek, amikor kell.

Attribútumok

-fields: Field[40][40]	Gyakorlatilag maga a pálya.
-steppables: List<Steppable>	A léptethető (kígyó részek) listája.
- b: Board	Önmaga, ugyanis ez egy (és az egyetlen) Singleton osztály.
- singleGame: boolean	Másként kell kicsit viselkednie (pl. két kígyó van, és nem hozhat létre különleges étkeket).

Metódusok

+ Create()	Létrehozz a mezőket, és beállítja a megfelelőket egymásnak szomszédnak.
+ GetInstance()	Statikus metódus a Singleton osztály egy példányának létrehozására. Ha nem létezik létrehozza, ha létezik (vagy a létrehozás után), Önmagát adja vissza.
+ GenSnake(columnt: int, c: Color): List<Snake>	Egy kígyó létrehozása az adott oszlopba, és színnel, listába rendezése és visszaadása.
+ AddSteppable(s: Steppable)	Egy léptethető dolgot ad a listához.
+ RemoveSteppable (s: Steppable)	Elvesz egy léptethető dolgot a listából.

+ SteppableContains (s:Steppabe): boolean	Boolean értékkel visszatérve megmondja, hogy a keresett elem benne van-e
+ GenFood()	Kaják generálása, helyük véletlenszerű sorsolása, ami nem eshet egybe kígyótestrésszel sem, annak nem lenne sok értelme.
+ GetField(x: int, y: int)	Visszaadja a pálya kért elemét.
+ Step()	Léptethető dolgok léptetése a listán végiglépkedve.
+ DeleteBoard()	Törli a tábla eredeti példányát.

3.1.7 Food

Felelősségek

A játékban felszedhető étkeket testesíti meg.

Attribútumok

- Color: color	Az étek színe, melyre kirajzoláskor lesz majd szükség.
- int: points	Az étek pontértéke.
- Field: field	A mező melyen éppen áll.
- Board: board	A pálya egy példánya.

Metódusok

+ GetPoints(): int	Visszaadja a csali pontértékét.
+ HitBy(s: Snake)	A kígyóval való “ütközést” kezeli. Meghívja rajta a Grow-t, a játékosának, pedig átadja a pontértékét, illetve szól a board-nak, hogy újat kellene generálni.
+ SetField(f: Field)	Beállít neki egy mezőt, amin állhat.
+ Draw(g: Graphics)	Kirajzolja az ablakba.

3.1.8 NormalFood

Felelősségek

Nincs érdemi eltérése az ősétől, a Food osztálytól.

Attribútumok

- points: int	Pontértéke = 1.
- color: Color	A színe, RGB(248, 168, 153).

Metódusok

+ HitBy(s: Snake)	Növeli a játékos pontszámát, és növeli a kígyó méretét is.

3.1.9 DecreaseFood

Felelősségek

Különleges étek, amit, ha sikerül felszednie a játékosnak, akkor csökken a kígyó mérete.

Attribútumok

- points: int	Pontértéke = 3.
- color: Color	A színe, RGB(181, 239, 206).

Metódusok

+ HitBy(s: Snake)	Növeli a játékos pontszámát, és csökkenti a kígyó méretét is.

3.1.10 ImmortalFood

Felelősségek

Egy időre a kígyó nem fog tudni magába harapni, ha ezt felszedi.

Attribútumok

- points: int	Pontértéke = 3.
- color: Color	A színe, RGB(187, 147, 205).

Metódusok

+ HitBy(s: Snake)	Egy időre az adott kígyó nem fog tudni magába harapni, hozzáadja a megfelelő effektet a kígyóhoz(igazából csak a fejéhez, de az pont elég).

3.1.11 DoubleFood

Felelősségek

Eléri, hogy a játékos által felszedett csalik egy időre dupla pontot érjenek.

Attribútumok

- points: int	Pontértéke = 3.
- color: Color	A színe, RGB(226, 207, 132).

Metódusok

+ HitBy(s: Snake)	Az adott kígyó játékosával eléri, hogy duplázódjanak a megszerzett pontok egy időre.

3.1.12 Direction

Felelősségek

Ez az enumeráció egy mező (Field) lehetséges szomszédos irányait, illetve a kígyó (Snake) haladási irányát reprezentálja: fel (UP), le (DOWN), balra (LEFT) vagy jobbra (RIGHT).

3.1.13 Field

Felelősségek

Ezekből a Fieldekből épül fel a pálya. Tudja, éppen mi áll rajta, legyen az kígyó (Snake), étek (Food) vagy akár fal (Wall). Ezeket ütközteti, amikor egy új kígyó lép rá.

Attribútumok

- neighbor: Map<Direction, Field>	A szomszédokat tárolja, ahol a kulcs az irány, értéke pedig a szomszédos mező az adott irányban.
- snakes: List<Snake>	A rajta álló kígyók listája.
- food: Food	A rajta lévő kaja (ha van).
- wall: Wall	A rajta lévő fal (ha van).
- x: int	X koordinátája, a kirajzoláshoz.
- y: int	Y koordinátája, a kirajzoláshoz.
- board: Board	A tábla egy példánya.
- color: Color	A mezők színe.

Metódusok

- SetX()	Beállítja az X koordinátát.

- GetX(): int	Visszaadja az X koordinátát.
- SetY()	Beállítja az Y koordinátát.
- GetY(): int	Visszaadja az Y koordinátát.
- Accept(s: Snake)	Hozzáadja az adott kígyót a listájához, illetve ütközteti a magán levő elemekkel (ha vannak ilyenek).
- Empty(): boolean	Boolean értékkel visszatérve, megmondja üres-e az adott mező (csalétket csak üres mezőre kellene generálni.)
- MoveOn(s: Snake)	Eltávolítja a listájából a kígyót, amikor tovább indul, illetve átadja a kígyót a megfelelő irányba a szomszédjának.
- Remove(s: Snake)	Eltávolítja a listájából a kígyót.
- AddFood(f: Food)	Étket ad az attribútumhoz.
- GetFood(): Food	Visszaadja az étket (ha van, ha nem null-t).
- RemoveFood()	Törli az étket a mezőről.
- AddWalll(w: Wall)	Falat ad a mezőhöz.
- GetNeighbor (d: Direction): Field	Visszaadja a kért szomszédot, a megfelelő irányból.
- SetNeighbor(d:Direction, f:Field)	Beállaítja a szomszédot a megfelelő irányba.
- Draw(g: Graphics)	Kirajzolja a mezőn lévő elemet, vagy magát a mezőt, ha üres.

3.1.14 Player

Felelősségek

A játékos. Nevét a bemeneten kapja a játékostól, kígyót pedig a Board generál neki, amit majd irányítani tud a gombokkal. Az osztály feladata a név és a pontszám tárolása, illetve a teljes kígyóé, ez kapcsolja össze az elemeket.

Attribútumok

- name: String	A játékos neve.
- points: int	A játékos pontszáma.
- snake: List<Snake>	A játkos kígyója.
- board: Board	A tábla egy példánya.
- multiplier: int	A pontok aktuális szorzója (alapjáraton 1).
- plays: boolean	True, amikor a játékos még él.
- winner: boolean	Ha veszít, hamisra állítódik. Kétjátékos esetén van lényege.

Metódusok

+ GetSnake(): List<Snake>	Visszaadja a teljes kígyót, vagyis egy listát az elemeivel sorban.
+ NewDirection(d: Direction)	Beállít a fejnek egy új irányt.
+ Move()	Végig lép az elemeken, és átadja az előttük levő irányát (ebbe az irányba fognak továbblépni).
+ AddPoints(p: int)	Növeli a pontszámát az adott számmal.
+ Lose()	Hamisra állítja a winner attribútumot.
+ Won(): boolean	A winner gettere.

3.1.15 Effect

Felelősségek

Effekt, ami egy kígyón tudja kifejteni a hatását.

Attribútumok

- name: String	Az effekt neve, egyedi, azonosítóként is szolgál.
- LifeTime: int	Az effekt élettartama, ez egy fix szám.
- TTL: int	Az effekt aktuális életkora, folyamatosan csökken.

Metódusok

+ Aging()	Csökken az effekt TTL-je.
+ Effect(p: Player)	Ha az effekt játékoson kell kifejtse a hatását.
+ Remove(p: Player)	Bármilyen hatása is volt, visszacsinálja.
+ Restart()	A TTL-t újra egyenlővé teszi a LifeTime-mal, amikor a kígyó ugyanazon hatású kaját szedi fel mielőtt még letelne az előző ideje.

3.1.16 DoublePointEffect

Felelősségek

Effekt, ami a játékosnál kétszer annyi pontot eredményez, egy időre.

Attribútumok

- name: String	Az effekt neve, egyedi, azonosítóként is szolgál.
- LifeTime: int	Az effekt élettartama, ez egy fix szám.
- TTL: int	Az effekt aktuális életkora, folyamatosan csökken.

Metódusok

+ Effect(p: Player)	Kifejti a hatását a játékoson.
+ Remove(p: Player)	Eltávolítja a hatását.

3.1.17 ImmortalEffect

Felelősségek

Csinálni nem csinál sokat, csak tárolja inkább az élettartamát, nevét.

Attribútumok

- name: String	Az effekt neve, egyedi, azonosítóként is szolgál.
- LifeTime: int	Az effekt élettartama, ez egy fix szám.
- TTL: int	Az effekt aktuális életkora, folyamatosan csökken.

Metódusok

-

3.1.18 Snake

Felelősségek

A játék szerves része, a kígyó (gyakorlatilag csak egy eleme) egy iránnyal és saját mezővel rendelkező dolog, ami egy listába tömörül a játékosnál, így ezzel lehet igazából játszani.

Attribútumok

- direction: Direction	Tárolja, hogy a kígyó az adott pillanatban merre tart.
- color: Color	A kígyó színét tárolja, ami majd kirajzoláshoz fog kelleni.
- field: Field	A mező, melyen a kígyó éppen áll.
- player: Player	A játékos, akihez a kígyó tartozik.
- board: Board	A pálya, amin a mezők, ezeken pedig a kígyó van. Azért kellett egy példánya, hogy a kígyó tudjon “szólni”, ha megevett egy étket és újat kell generálni.
- effects: List<Effect>	A kígyón aktuálisan a hatásukat kifejtő effektek listája.

Metódusok

+ SetDirection(d: Direction)	Beállítja a kígyó irányát.
+ GetDirection(): Direction	Visszaadja a kígyó irányát.
+ GetColor(): Color	Visszaadja a kígyó színét.
+ SetField(f: Field)	Beállítja a kígyó mezőjét.
+ GetField(): Field	Visszaadja a kígyó mezőjét.
+ SetPlayer(p: Player)	Beállítja a kígyó játékosát.
+ HitBy(s: Snake)	Két kígyó ütközését kezeli.
+ HitBy()	A kígyó falba ütközését kezeli.
+ Grow()	A kígyó növekedését kezeli, a kaja megevésének hatására.
+ Shrink()	A kígyó zsugorodását kezeli, az egyik speciális kaja megevésének hatására.
+ Die()	A kígyó halálát, a játék végét hozza el.
+ Draw(g: Graphics)	A kígyó kirajzolását kezeli.
+ Step()	A kígyó előre (a saját, beállított irányába való) lépését kezeli.

3.1.19 Steppable

Felelősségek

Egy interface, ami így összefogja a léptethető elemeket.

3.1.20 Wall

Felelősségek

A pálya szélén lévő fal.

Attribútumok

- field: Field	A mező, amin a fal áll.
- color: Color	A fal színe.

Metódusok

+ Draw(g: Graphics)	Kirajzolja a falat.

3.1.21 GamePanel

Felelősségek

A kirajzolás fő felelőse, minden egyes elem kirajzolása innen fog elindulni, amikor lépésenként meghívódik a repaint().

Attribútumok

- board: Board	A tábla egy példánya.

Metódusok

+ paint()	Végiglépkedve a tábla mezőin mindet kirajzolja.