Snake

A programozás alapjai 3 házi feladat

Keresztes Csenge

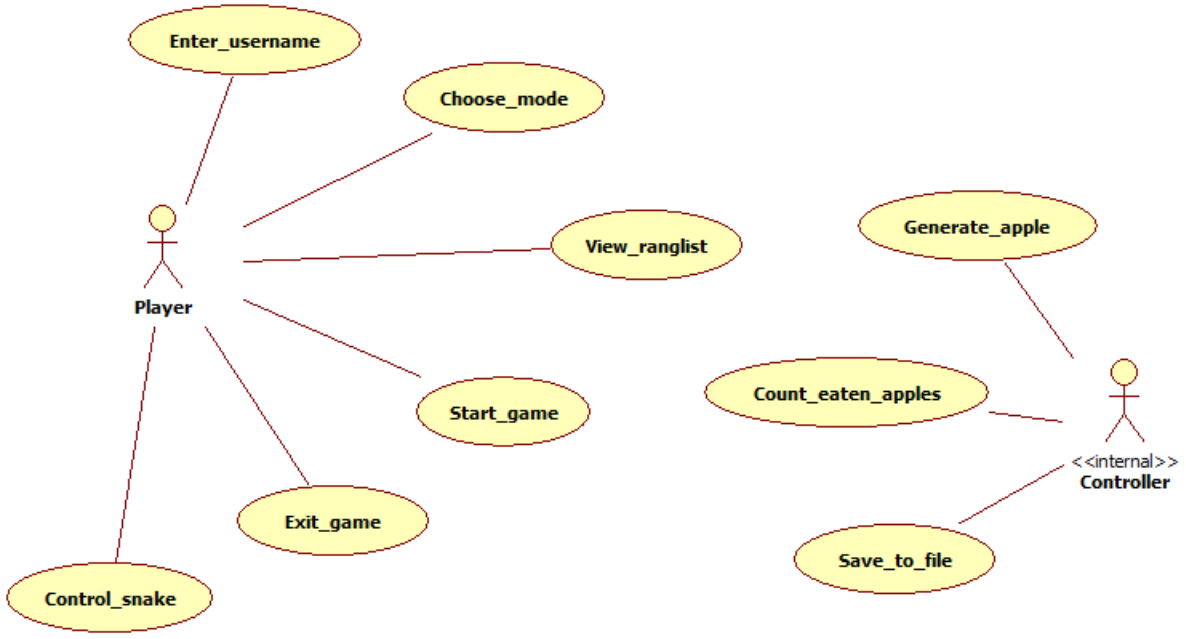
B8XWST

1 A FELADAT LEÍRÁSA

A feladat egy grafikus felületű játék elkészítése, amely igen beszélő nevet kapott: Snake (magyarul kígyó). A Snake játék lényege, hogy a felhasználó irányít egy kígyót egy pályán. A pályán mindig van egy véletlenszerűen elhelyezett alma. A felhasználó célja az, hogy a kígyót úgy irányítsa, hogy nekimenjen a pályán levő almának és így megegye azt. Amint a kígyó megevett egy almát, az alma eltűnik a pályáról és helyette egy másik megjelenik valahol máshol, illetve a kígyó teste is nő egy egységet. Minden egyes alma megevésénél a score növekszik eggyel, a cél a minél nagyobb score elérése. Ha a felhasználó a kígyót úgy irányítja, hogy az nekimegy a pálya szélének, vagy önmagának, akkor a játék véget ér. A kígyót megállítani nem lehet. A program indításakor megjelenik egy ablak, amelyben a felhasználó beírhat egy felhasználónevet , kiválaszthatja, hogy a három nehézségi szint közül (easy, medium, hard) melyiket szeretné játszani. Ugyanezen az ablakon van még egy start gomb is, amelyre kattintással megjelenik egy másik ablak és a játék elindul. A felhasználó a játék során a kígyót a navigációs billentyűk segítségével irányíthatja fel, le, jobbra, illetve balra. Ha a játék a fentebb említett két mód valamelyikén véget ér, az ablakon megjelenik a „GAME OVER”, a “Highscore: nev:pont”, illetve a „SCORE: pont” felirat, ahol a “nev” annak a felhasználónak a neve, aki eddig a legtöbb pontot szerezte meg (eltekintve a játék nehézségi szintjétől). A pont mindkét esetben a felhasználó által a játék végéig sikeresen elkapott gyümölcsök számát mutatja. A játék penete közben illetve végeztével is található az ablak alján egy “Ranglist” gomb, amely megnyomásával megjelenik ugyanazon az ablakon egy ranglista, amely az eddigi játékosok nevét, az általuk játszott játék nehézségi szintjét, illetve a megszerzett pontot tartalmazza, nehézségi szint szerint csoportosítva, majd pontszám szerint csökkenő sorrendben rendezve. Az ablakot a címsorban található (x) ikonnal lehet bezárni, de mivel a kezdeti “menu-ablak” sem záródott be, így akár újra lehet játszani is, a program újbóli lefuttatása nélkül.

2 USE-CASE-EK

2.1 USE-CASE DIAGRAM



2.2 USE-CASE LEÍRÁSOK

Cím	Enter_username
Leírás	A felhasználó a menüablakban beírja a felhasználónevét az erre kijelölt területre.
Aktorok	Player

Cím	Choose_mode
Leírás	A felhasználó a menüablakban kiválasztja a nehézségi szintet (JComboBoxból).
Aktorok	Player
Főforgatókönyv	A felhasználó kiválasztja a nehézségi szintet.
Alternatív forgatókönyv	1. „easy” szintet választja 2. „medium” szintet választja 3. „hard” szintet választja

Cím	Start_game
Leírás	A felhasználó a menüablakban rákattint a „START” gombra és a játék elindul.
Aktorok	Player

Cím	Exit_game
Leírás	A felhasználó a program futtatásától kezdve bármikor rákattinthat akár a menüablak, akár a fő(játék)ablak címsorában lévő piros (x) ikonra, amely hatására bezáródnak az ablakok, a program futása leáll.
Aktorok	Player

Cím	Control_snake
Leírás	A felhasználó a navigációs billenytűk segítségével irányítja a snake-et.
Aktorok	Player
Főforgatókönyv	A felhasználó irányítja a snake-et.
Alternatív forgatókönyv	1. a felfelé nyíl megnyomásával a kígyó felfelé mozog, kivéve, ha addig lefelé mozgott 2. a jobbra nyíl megnyomásával a kígyó jobbra mozog, kivéve, ha addig balra mozgott 3. a lefelé nyíl megnyomásával a kígyó lefelé mozog, kivéve, ha addig felfelé mozgott 4. a balra nyíl megynomásával a kígyó balra mozog, kivéve, ha addig jobbra mozgott

Cím	View_ranglist
Leírás	A felhasználó, miután vesztett, rákattinthat az ablak alján lévő „Ranglist” gombra és megtekintheti a ranglistát egy táblázatban.
Aktorok	Player

Cím	Generate_apple
Leírás	A Controller random x és y koordinátákra (a pályán belül) generál egy almát a játék indításakor, illetve ha a kígyó megette azt.
Aktorok	Controller

Cím	Count_eaten_apples
Leírás	A Controller a játék során számolja az adott felhasználó által megevett almák számát.
Aktorok	Controller

Cím	Save_to_file
Leírás	A Controller a játékos nevét, a nehézség szintjét és az elért pontszámot a „players.dat” fájlba menti. Rekorddöntés esetén a felhasználó nevét és a pontszámát a „highest.txt” fájlba menti.
Aktorok	Controller

3 STRUKTURÁLIS LEÍRÁS

3.1 AZ OSZTÁLYOK LEÍRÁSA

3.1.1 Snake

Felelősségek

A játékban a kígyónak felel meg. A kígyó egységekből áll, ha a kígyó felszed egy almát, tehát a feje találkozik az almával, akkor a kígyó egy egységgel megnő. Ha önmagába megy bele vagy a pályát reprezentáló négyzet valamelyik falába, akkor a kígyó meghal, a játék véget ér.

Attribútumok

private int x	A kígyó fejének kezdeti, horizontális, x koordinátája.
private int y	A kígyó fejének kezdeti, vertikális, y koordinátája.
private int applesEaten	A kígyó által megevett almák száma.
private Direction direction	A kígyó aktuális iránya, az hogy éppen melyik irányban mozog.
public ArrayList<Point> location	A kígyó egyes elemeinek a koordinátáit Point típusú ArrayListben tároltam, tehát pl. a location[0] felel meg a kígyó fejének, a location[1] pedig a feje utáni egységnek. A Point típus mindig 2 koordinátából adatokat tárol, tehát a kígyó minden egységének helyét is 2 koordinátával jellemezhetjük.

Metódusok

public void setx(int n)	Az x attribútum settere, beállítja az értékét a paraméterként kapott értékre.
public void sety(int m)	Az y attribútum settere, beállítja az értékét a paraméterként kapott értékre.
public int getx()	Az x attribútum gettere, visszaadja az értékét.
public int gety()	Az y attribútum gettere, visszaadja az értékét.
public void setdirection(Direction d)	A direction attribútum settere, beállítja az értékét a paraméterként kapott értékre (irányba).

public Direction getdirection()	A direction attribútum gettere, visszaadja az értékét (irányt).
public int getapplesEaten()	Az applesEaten attribútum gettere, visszaadja a kígyó által megevett almák számát.
public void setapplesEaten(int n)	Az applesEaten attribútum settere, beállítja az értékét a paraméterként kapott értékre.
public Snake()	A kígyó konstruktora, ami beállítja, hogy a kígyó feje alapértelmezetten a pálya (20,80) koordinátájú egységén helyezkedjen el. Beállítja a következő testegység helyét is (20, 79)-re, mivel a kígyó úgy indul el, hogy van egy feje és egy testeleme is. A konstruktor továbbá beállítja, hogy a kígyó jobbra menjen alapértelmezetten és a kezdeti applesEaten értékét is 0-ra állítja.
public Snake(int x, int y)	Ez is a snake konstruktora, csap paraméterekkel, ahol az x illetve az y az attribútomoknál leírtakat jelentik. Tehát a pályá egy tetszőleges helyhez is hozzá tudjuk adni a kígyót. A paraméternélküli konstruktorhoz hasonlóan beállítja az attribútumok alapértelmezett értékét.
public void collidedwithApple(Apple a)	Ez a metódus ellenőrzi, hogy a kígyó ütközött-e azzal az almával,amelyiket paraméterként megkapta, vagy sem. Ha a kígyó fejének x koordinátái és y koordinátái megegyeznek az alma x és y koordinátáival, akkor ütköztek. Ekkor meghívja a kígyó tailgrow() függvényét, illetve 1-gyel megnöveli az applesEaten értékét.
public void tailgrow()	A kígyó hosszát egy egységgel megnövelő metódus, amelyet a collidedwithApple(a) metódus hív meg ha a kígyó ütközött az almával. Ilyenkor a kígyó koordinátáit tároló location-höz hozzáadódik egy pont. Pontosan annak a pontnak a koordinátája lesz ez a pont, ami korábban a kígyó utolsó(leghátsó) testelemének volt a koordinátája, hiszen a kígyó mindeközben mozog.
public boolean selfcollision()	Ez a kígyó egyik Boolean visszatérési értékkel rendelkező függvénye, amely ellenőrzi, hogy a kígyó beleütközött-e magába, tehát a fejének a koordinátái megegyeznek-e a testének bármelyik koordinátáival (ezért van for ciklusban). Ha beleütközött, igaz értékkel tér vissza, máskülönben hamissal.
public boolean bordercollision()	A kígyó másik Boolean visszatérési értékkel rendelkező függvénye, amely azt ellenőrzi, hogy a kígyó a megadott pálya kerateinek neki ment-e. Ha nekiment bármelyiknek (vagy a felsőnek, vagy a jobb szélsőnek, vagy az alsónak vagy a bal szélsőnek), akkor igaz értékkel tér vissza, ha ezek egyike se következett be, akkor hamissal. A

	megadott pályahatárok a grafikai megjelenítés érdekében lettek beállítva.
public void move()	Ez a függvény felel a kígyó mozgásáért. Egy for ciklus segítségével "végigshifteli" a kígyó testrészeit. Tehát pl a location[1] helyen lévő testrész új koordinátái megegyeznek a location[0] helyen lévő testrész(azaz a fej) koordinátáival. Egy switch-case segítségével vizsgáltam meg az egyes eseteket a attól függően, hogy a direction milyen értéket kapott . Ezt a felhasználó által lenyomott navigációs billenytűk határozzák meg. Minden esetben (up, down, right, left) a kígyó fejének vagy az x vagy az y koordinátáját növeli vagy csökkenti a tábla (Board) egy egységével az irányoknak megfelelően.
public void paint(Graphics g, Image head, Image body, Board board)	A kígyót kirajzoló függvény, amelynek vannak paraméterei. Mivel és a kígyó testének az egységeit és a fejét képpel ábrázolom, ezért van két Image parameter. A Graphics g is egy parameter, hiszen a GUI-nak erre szüksége van a megjelenítéshez. Paraméter továbbá a Board board is, mivel ez az osztály felel a játék közbeni grafikus dolgok megjelenítéséért, ő fogja majd meghívni a snake-nek ezt a függvényét.

3.1.2 Apple

Felelősségek

A játékban lévő almának felel meg. Egyszerre csak egy alma van a pályán mindig, amellyel ha egy kígyó ütközik, az eltűnik és helyette egy másik generálódik. Az almák random (x,y) koordinátákra helyeződnek el.

Attribútumok

private int x	Az alma horizontális, x koordinátája.
private int y	Az alma vertikális, y koordinátája.
Random random	A random osztály, amelyet az Alma osztályba példányosítunk is, mivel ez fogja majd az alma koordinátáit random generálni.

Metódusok

public void setx(int n)	Az x attribútum settere, beállítja az értékét a paraméterként kapott értékre.
public void sety(int m)	Az y attribútum settere, beállítja az értékét a paraméterként kapott értékre.
public int getx()	Az x attribútum gettere, visszaadja az értékét.
public int gety()	Az y attribútum gettere, visszaadja az értékét.

public Apple()	Az alma parameter nélküli konstruktora, amely a random osztály példányának segítségével meghatározza az alma koordinátáit a pályán.
public void NewApple()	Lényegében ugyanazt csinálja, mint a konstruktor, csak ezt akkor hívjuk ha a kígyó ütközött az almával.
public Apple(int x, int y)	Az alma parameters konstruktora, amely beállítja az alma koordinátáit a paramétereknek megfelelően.
public void paint(Graphics g, Image apple, Board board)	Az almát kirajzoló függvény, amelyet majd a Board egy példánya fog meghívni. Az almát is egy kép segítségével szemléltettem, ezért kellett a paramterei közé is egy kép. Ez a függvény meghívja a g.drawImage(apple, this.x, this.y, board) függvényt is, amely segítségével a GUI valóban elhelyezi majd a képet a panelen (ami itt a board) Az alma x és y koordinátái is paraméterei a függvénynek, hiszen ez alapján tudja majd, hogy a pályán melyik egységbe kell helyezni a képet.

3.1.3 Direction

Felelősségek

Ez egy enum osztály, amely 4 konstanst tárol: up, right, down, left. Ezek a kígyó páylán lévő mozgási irányát reprezentálják. A navigációs billenytűknek is megfeleltethetőek, hiszen up=felfelé nyíl, right=jobbra nyíl, down=lefelé nyíl, left=balra nyíl.

Attribútumok

-

Metódusok

-

3.1.4 Game

Felelősségek

Ez az osztály felelős a játék működéséért, illetve a grafikus felület irányításáért, tehát ő hívja meg bemenetnek megfelelően a grafikus felületet megvalósító Frame, illetve Board osztály metódusait. A Ranglist metódusait is ő hívja meg. Ez az osztály indítja magát a játékot és hozza létre a megfelelő Snake, Apple, Frame, Ranglist és Board példányokat. Implementálja az ActionListenert, hiszen ennek az osztálynak a feladata az események kezelése, tehát pl. az is, hogy ahogy fut a játék, a megfelelő késleltetéssel mozgassa a kígyót, illetve a navigációs billenytűk lenyomását, és a fombokra kattintást is ő figyeli. (Természetesen a MenuFrame esetén nem, mivel azt nem tekintettem konkrétan a játék részének, abban csak a beállítások vannak és az játék-ablak megnyitása.)

Attribútumok

private String name	A felhasználónév attribútuma, amit majd a felhasználó a MenuFrame osztály által létrehozott ablakban ad meg az arra megfelelő JTextField mezőben.
private String mode	A nehézségi szintet jelölő attribútum, amit majd a felhasználó a MenuFrame osztály által létrehozott ablakban választ ki az arra megfelelő JComboBox segítségével.
static boolean running	Ez a Game osztály statikus Boolean változója, amely azt jelöli, hogy maga a játék(ahol a kígyó mozog és lehet irányítani) fut-e. Kezdetleges értéke false. Amíg a menu-ben a felhasználó rá nem kattint a Start gombra, addig ez az érték false. Illetve ahogy veszített a játékos és vége a játéknak, szintén visszavált false-ba.
private int delay	A menüben kiválasztott nehézségi szintnek megfelelő időbeni késleltetés attribútuma, amely befolyásolja, hogy a kígyó majd milyen gyorsan mozog.
Board panel	A játék futása során a grafikus felület megvalósításáért felelős osztály típusal ellátott változó.
Timer timer	Időzítő osztály egy változója. Ennek segítségével állítjuk be, hogy a program futása során az egyes eseménynek milyen időközönként következzenek be.
Apple apple	Alma osztály típusú változó
Snake snake	Snake típusú váltpzó
Frame frame	Frame osztály típusú változó, ami majd magát a játék ablakát jelenti számunkra.

Metódusok

public Snake getSnake()	A snake attribútum gettere, visszaadja a snake-et.
public static boolean getRunning()	A running attribútum gettere, visszaadja az értékét.
public static void setRunning(boolean run)	A running attribútum settere, beállítja az értékét a paraméterben kapott értékre.
public Game(String name, String mode)	A Game osztály konstruktora, paraméterekkel, amelyek meghatározzák, hogy a játék milyen felhasználónévvel és milyen nehézségi szinten induljon (ld. MenuFrame osztály). Ez a konstruktor továbbá példányosítja az Apple, Snake, Frame, Ranglist, illetve a Board osztályt, amelynek megfelelő értékeket is átad paraméterezés segítségével (ld.Frame illetve Board osztály) . A Board osztály pédányát Focusable-re állítja be, hiszen egy komponenshez csak így tudunk hozzárendelni egy keyListenert, amely majd figyeli, hogy melyik gombot

	nyomta le a felhasználó és annak megfelelően mozog majd kígyó. A konstruktorban még switch case segítségével beállítja, hogy a paraméterként kapott mode alapján (easy, medium vagy hard) milyen késleltetéssel(delay) fusson majd a játék, tehát a delay értékét is beállítja. Mindenzek után meghívja a Game osztály Start() metódusát.
public void Start()	A játék indításáért felelős metódus. Átállítja a running attribútum értékét true-ra, hiszen a játék elindul, illetve aTimer osztályt itt példányosítja a megfelelő delay megadásával, majd elindítja a timer-t.
public void checkCollisions()	Ez a metódus ellenőrzi, hogy a játékban lévő kígyó ütközött-e saját magával (a kígyó selfCollision() függvénye igaz értéket ad-e) vagy a pálya falaival (a snake borderCollision() függvénye igaz értéket ad-e). Ha ezek közül bármelyik is igaz, akkor a running attribútumot átállítja false-ra és leállítja a timert. Ezután meghívja a Ranglist osztály save_data(file) metódusát, hiszen el kell menteni a felhasználó nevét, a nehézségi szintet, illetve a megszerzett pontot az általam létrehozott “players.dat” fájlba. Még ebben a metódusban adunk hozzá a Frame “topb” , “Ranglista” nevű gomjához egy ActionListenert.Ennek köszönhetően, ha a játék már nem fut, akkor az alul lévő “Ranglista” gomra kattintva megjelenik egy táblázat (a táblázatot is ebben az ActionListenerben hívjuk meg), amelyen a felhasználó megtekintheti az addig játszottak nevét, a hozzájuk tartozó nehézségi módot, illetve az általuk megszerzett pontokat.
public void keyPressed(KeyEvent e)	Ez a függvény a public class MyKeyAdapter belső osztály felüldefiniált függvénye, amellyel beállítottam, hogy az egyes navigációs billenytűk lenyomására a kígyó direction változója az irányoknak megfelelő értéket/típust vegyen fel. De a kígyó nem fordulhat 180 fokot, hiszen akkor megenné magát és vége lenne a játéknak. Ezért minden egyes irányváltási kérelem(tehát a felhasználó általi nyíllenyomás) után egy if-clause-ban megvizsgáltam, hogy a jelenlegi irány az a kért irány 180 fokos elforgatotja, tehát ellentéte-e.(pl. bal iránykérés esetén nem lehet az aktuális irány job). Ha nem az ellentéte, akkor az irányváltás megtörténik. Különben semmi sem történik.

public void actionPerformed(ActionEvent e)	Mivel az implementált ActionListener osztály absztrakt, így az actionPerformed(ActionEvent) absztrakt függvényét felül kell definiáljuk, hogy működjön is a játék. Ebben a felüldefiniált függvényben határoztam meg, hogy ha a játék az fut, akkor milyen függvényeknek kell meghívódniuk, és ezek:
	snake.move();
	snake.collidedwithApple(apple);
	checkCollisions();
	Tehát amikor a játék fut, akkor (a delaynek megfelelő időközönként) a kígyót mozgatjuk, ellenőrizzük, hogy evett-e almát és ellenőrizzük, hogy ütközött-e (ami a játék végét jelentené).

3.1.5 Board

Felelősségek

A játék futása során, illetve a GameOver kiiratásakor a grafikus felület megjelenítéséért felelős osztály. Ez jeleníti meg a mozgó kígyót, a random koordinátákra generálódó almát, illetve magát a pályát, amelyen a kígyó mozog. (+a gameovert a végén). Az osztály a JPanel osztály egy leszármazottja, tehát örökli annak a funkcióit.

Attribútumok

static final int width=640	Statikus, nem megváltoztatható változó, amely az ablak szélességét fejezi ki.
static final int height=700	Statikus, nem megváltoztatható változó, amely az ablak magasságát fejezi ki.
static final int unit=20	Statikus, nem megváltoztatható változó, amely a pályán egy egység méretét fejezi ki.
private String mode	A felhasználónév attribútuma, amelyet a Board a Game osztálytól kap meg paraméterként a konstruktorában.
private String name	A felhasználó által választott nehézségi mód, amelyet a Board a Game osztálytól kap meg paraméterkét a konstruktorában.
private int highscore	A játék valaha elért legnagyobb pontszámát tárolj ez a változó.
Graphics g	A grafikus felület változója.
Apple apples	Az Alma osztály típusú változó.
Snake snake	A Snake osztály típusú változó.
Image apple	Image típusú változó, ami majd az alma képe lesz
Image head	Image típusú változó, ami majd a kígyó fejének képe lesz.
Image body	Image típusú változó, ami majd a kígyó testének az egységének képe lesz.

Metódusok

public static int getwidth()	A width attribútum gettere, visszaadja az értékét.
public static int getheight()	A height attribútum gettere, visszaadja az értékét.
public static int getunit()	A unit attribútum gettere, visszaadja az értékét.
public String getMode()	A mode attribútum gettere, visszaadja az értékét.
public void setMode(String Mode)	A mode attribútum settere, beállítja az értékét a paraméterként megkapott értékre.
public String getName()	A name attribútum gettere, visszaadja az értékét.
public void setName(String nam)	A name attribútum settere, beállítja az értékét a paraméterként megkapott értékre.
public int getHighscore()	A highscore attribútum gettere, visszaadja az értékét.
public void setHighscore(int high)	A highscore attribútum settere, beállítja az értékét a paraméterként megkapott értékre.
public Board(String name, String mode, Snake s, Apple a)	A Board osztály paraméteres konstruktora. A Game osztálytól kapja meg majd ezeket a paramétereket, hiszen az hívja meg a konstruktort. Tehát beállítódik a megfelelő felhasználónév, a nehézségi szint, a kígyó pédánya illetve az alma példánya. A konstruktor továbbá beállítja a panel méretét a megfelelő statikus változói segítségével, a panel háttérszínét sötétszürkére, valamint meghívja a Board loadImages() függvényét.
public void loadImages()	Ez a függvény betölti a megfelelő képfájlokat (az elérséi útjukkal megadva), majd hozzárendeli a megfelelő Image típusú attribútumokhoz azokat.
public void draw(Graphics g)	Ez a függvény rajzol a paraméterként megkapot g komponensre. Van benne egy if-else elágazás, miszerint ha a Game running változója true értékű, akkor beállítja a pálya méreteit illetve színét feketére, benyégyzethálózza sötétszürke színnel a a panel egészét. Az értékek a pálya méreténél a szép grafikus megjelenés miatt lettek úgy kiválasztva. Még ugyanezen az if-en belül meghívja az alma illetve a kígyó paint metódusát (ennek a segítségével fog megjelenni a képernyőn), valamint megfelelő szövegtípus-beállításokat végez el, hiszen a panelnak a tetején a játék mindig ki fogja írni a felhasználó nevét, a módot amiben játszik, illetve az általa szerzett pontokat, amik minden egyes alkalommal nőnek, amikor a kígyó megeszik egy almát. Ezt a 3 adatot egy Stringbe vettem fel, amit a FontMetrics osztály segítségével helyeztem el a panel felső részén, a címsor alatt. Az else ág akkor következik be, ha a Game running változója false értékű, tehát a játék már nem fut, mert a

	játékos vesztett. Ebben az ágban meghívódik a Board gameOver(Graphics g) függvénye.
public void gameOver(Graphics g)	Ez a függvény is a megjelenítésért felelős, mindent megjelenít a paraméterként kapott grafikus komponensen amit akkor kell, mikor a játéknak vége van. Elvégez egy fájl beolvasást (ha nem létezik a megadott fájl, akkor létre is hozza azt) és egy fájlba mentést is BufferedReader illetve PrintWriter osztályok segítségével. Miután betöltötte/létrehozta a fájlt, ellenőrzi, hogy van-e már benne valami. Ha nincs, akkor beleírja az aktuális felhasználó nevét és az általa megszerzett pontot az alábbi formában: nev:pont. Így tároljuk a megadott txt fájlban lévő, éppen aktuális addigi legmagasabb pontott szerzett felhasználó nevét és a hozzá tartozó pontszámot. Ezután beolvassa a fájlt, pontosabban a sorait(habár cask egy van). És a benne található String-et “széttördeli” 2 részre. Az Integer.parseInt(line.split(":")[1]) függvény segítségével megkapjuka pontszámot, a line.split(“:”)[0] segítségével pediga felhasználónevet. Ezután megvizsgáljuk, hogy a beolvasott pontszám nagyobb-e, mint az aktuális játékos által szerzett pontszám. Ha nagyobb, akkor a fájlban nem módusul semi. Ha kisebb, akkor viszont a fájlban található String kicserélődik az aktuális játékos nevére és megszerzett pontjára, hiszen az less az új highscore. Ez a gameOver(g) függvény továbbá beállítja g.setColor() illetve g.setFont() függvények segítségével a panelra kiirandó Stringek betűtípusát. Megjelenítem az aktuális highscoret tartó játékos nevét és pontszámát, az aktuális játékhoz tartozó játékos pontszámát, illetve a “Game Over…” feliratot, amely jelzi a játék végét.

public void paintComponent(Graphics g)	Ez egy automatikusan hívódó függvény, amely meghívja az alábbi függvényeket:
	super.paintComponent(g);
	draw(g);
	repaint(); (ami frissíti a megjelenítést)
	Ez a függvény a kulcsa mindennek, ha ez nem lenne, akkor nem jelenne meg semi. Ez jeleníti meg ténylegesen a g komponens elemeit.

3.1.6 MenuFrame

Felelősségek

Ez az osztály a JFrame osztály leszármazottja. A program indításakor megjelenő ablakot reprezentálja a rajta lévő gombbal, JTextFielddel és JComboBoxszal együtt. Ezzel az osztállyal kezdődik minden, hiszen a Main-ben ezt hívjuk meg.

Attribútumok

private Object[] mode	Ez a tömb a nehézségi szinteknek megfelelő Stringeket tartalmaz: “easy”, “medium”, “hard”.
private JTextField tf	JTextField típusú attribútum, amelybe bele lehet írnia a felhasználónak.
private JComboBox cb	JComboBox típusű attribútum, amely majd 3 legördülő választási lehetőséget kínál fel.
private JButton b	A képernyőn lévő gomb, amire majd lehet kattintani. JButton típusú attribútum.
Image icon	Image típusú változó, amely majd a címsorban lévő ikon képe less.

Metódusok

private void initComponents()	Ebben a függvényben azok a beállítások vannak, amelyeket majd a MenuFrame konstruktorában meghívunk. Példányosítja a JTextField-et és beállítja az abba beírható maximális karakterek számát 15re, beállítja a setForeground() és a setFont() függvénnyel, hogy amikor a felhasználó oda begépeli a felhasználónevét, az hogy jelenjen meg, milyen betűtípussal, stb. A szövegdoboznak háttér/kitöltési színt is állít be, illetve a keretein is változtat. A függvény példányosítja a JComboBox-ot is, annak is beállít háttérszínt illetve a szöveg színét és típusát is beállítja. Példányosítja a JButton-t is, amelyen a “START” felirat lesz majd látható. A gombhoz a b.addActionListener(new AddButtonListener()) metódussál adunk hozzá egy ActionListenert, hiszen ha majd a felhasználó a gombra kattint akkor történnie is kell valaminek. Még ebben a függvényben létrehozunk egy Jpanel-t, illetve 2 JLabel-t, amelyek majd megjelenítik az ablakona JTextField doboz előtt, hogy “Name: “, illetve a JComboBox előtt, hogy “Mode: ” A JLabel-öknek beállítjuk a betűszínt és típust, valamin a JPanel háttérszínét feketére. Ezután hozzáadjuk a panelhez a megfelelő változókat. Amilyen sorrendben vannak hozzáadva, olyan sorrendben jelennek meg a

	képernyőn is. A panelt hozzáadja a függvény a MenuFrame-hez is a this.add(multiPlayer) metódussal (ahol a multiplayer a panel neve). A this.setVisible(true) függvény segítségével láthatóvá tesszük a MenuFrame ablakot és a hozzáadott komponenseket egyaránt.
public void actionPerformed(ActionEvent e)	Ez a függvény a final class AddButtonListener implements ActionListener belső osztály egy felüldefiniált függvénye, amely megadja, hogy mi történik ha a felhasználó rákattint a "START" gombra. Ilyenkor létrehozza a Game osztály egy pédányát a Game osztály konstruktorával, amelyben a paraméterek: tf.getText() – ez a felhasználó által beírt felhasználónevet (String name-t) jelölő parameter cb.getSelectedItem().toString() – ez a felhasználó által választott nehézségi szintet (String mode-ot) jelölő paraméter
public MenuFrame()	A MenuFrame paraméter nélküli konstruktora, amely beállítja, hogy a címsorban "Menu" cím szerepeljen, beállítja, hogy a címsorban lévő (x) ikonra kattintva az ablak bezáródjon, beállítja az ablak minimális méretét(amit tehát lehet nagyobbra "húzni" ha megjelent), illetve meghívja a MenuFrame initComponents() függvényét.

3.1.7 Frame

Felelősségek

A JFrame osztály leszármazottja, ő maga az ablak, ami megjelenik miután a Menu ablakban a felhasználó rákattintott a „START" gombra. A Frame a Game osztály konstruktorában példányosodik, az hívja meg a megfelelő paramétereket átadva neki. A Frame osztály elrendezés-menedzsere CardLayout, mert ezzel az ablakon különböző panelek jeleníthetőek meg, lehet közöttük „váltani", nem kell több ablakot létrehozni.

Attribútumok

private Board panel	Board típusú változó. Ez lesz az egyik egejelnítendő panel, ezen fog maga a játék megvalósulni.
private JPanel panelCont	JPanel típusú változó. Ez lesz majd a konténer, amiben tároljuk a különböző paneleket.
private JPanel top	JPanel típusú változó. Ez lesz a másik panel, ezen fog megjelenni a táblázat.
private JButton topb	JButton típusú változó, tehát egy gomb. Példányosítás után ere kattintva váltódik panelé

private CardLayout cl	CardLayout típusú változó. Ő az ablak elrendezés-menedzsere
private Image icon;	Image típusú változó, ebbe fogjuk majd betölteni azt a képet, amely az ablak címsorában lévő ikonként jelenik meg.

Metódusok

public JButton getTopb()	A topb attribútum gettere, visszaadja a topb-t.
public CardLayout getCl()	A cl attribútum gettere, visszaadja a cl-t.
public JPanel getPanelCont()	A panelCont attribútum gettere, visszaadja a panelCont-ot.
public Frame(Board pan)	Az ablak konstruktora, amelyben vannak paraméterek. Mivel a Frame osztályt a Game osztály konstruktora példányosítja, így a Game osztály átadja neki az általa példányosított Board panel-elt, hiszen a CardLayout segítségével ez lesz az első panel amit megjelenít. A konstruktorban panel attribútuma tehát megkapja a konstruktor pan paramétrét, a többi attribútum (az icon-on kívül) pedig pédányosodik, a gombon lévő felirat “Ranglist” less, hiszen ezzel lehet majd átváltani a “top” panelra, ahol megjelenik majd a táblázat. A icon = Toolkit.getDefaultToolkit().getImage("src/resources/snake.png"); this.setIconImage(icon); parancsok segítségével betöltjük az ablak ikonképét a megadott elérési útnak megfelelően, majd beállítjuk a címsor icókonképét erre. Beállítjuk, hogy a panelCont elrendezése a cl (CardLayout) legyen, majd a Topb gomb megjelenését állítjuk be. A “panel” elrendezését BorderLayout tipusúra állítjuk be, mert így lehet szépen az ablak aljába elhelyezni a Topb gombot. A “top” panel elrendezését is BorderLayout tipusúra állítjuk be, ezen fog majd megjeleni a táblázat. getPanelCont().add(panel, "1"); - ezzel hozzáadjuk a panelkonténerhez a panelt, “1”-es String-gel jelölve a későbbi rövidebb jelölés érdekében. getPanelCont().add(top, "2"); - ezzel hozzáadjuk a panelkonténerhez a top-ot, “2”-es String-gel jelölve. getCl().show(getPanelCont(), "1"); - ez a parancs azt csinálja, hogy amikor az ablak megnyílik akkor alapból először a panel fog megjelenni, mivel ő kapta az “1”-es jelölést (amúgy is a panel a Board típusú, tehát logikus, hogy az jelenik meg, mert ezen megy a játék). A továbbiakban a konstruktor hozzáadja az ablakhoz a panelkonténert(amivel együtt automatikusan hozzáadódik a panel és a top is), beállítja, hogy az ablak címsorában a “SNAKE”

	jelenjen meg, (x) ikonra kattintva záródjon be az ablak, ne lehessen átméretezni, kerüljön középre a képernyőn.

3.1.8 Player

Felelősségek

A Player egy szerializálható osztály, amely valóban egy játékosnak felel meg, annak az adatait (tehát név, nehézségi szint, pontszám) tárolja.

Attribútumok

private String name	String tipusú attribútum, a felhasználó neve.
private String mode	String tipusú attribútum, a felhasználó által kiválasztott nehézségi szint.
private int score	int típusú attribútum, a felhasználó által elért pontszám.

Metódusok

public String getName()	A name attribútum gettere, visszaadja az értékét.
public String getMode()	A mode attribútum gettere, visszaadja az értékét.
public int getScore()	A score attribútum gettere, visszaadja az értékét.
public void setName(String Name)	A name attribútum settere, beállítja a paraméterként kapott értékre.
public void setMode(String Mode)	A mode attribútum settere, beállítja a paraméterként kapott értékre.
public void setScore(int Score)	A score attribútum settere, beállítja a paraméterként kapott értékre.
public Player(String Name, String Mode, int Score)	A Player osztály konstruktora. A paramétereinek megfelelően beállítja az attribútumainak az értékét.

3.1.9 PlayerData

Felelősségek

Ez az osztály az AbstractTableModel leszármazottja, belőle fog elkészülni a ranglistás táblázat.

Attribútumok

private List<Player> players	Listában tároljuk a játékosok adatait. Példányosításkor ez egy ArrayList lesz

Metódusok

public List<Player> getPlayers()	A players attribútum gettere. Visszaadja a játékosok listáját.
public int getRowCount()	A sorok számának a gettere, visszaadja a táblázatban lévő sorok számát. Felüldefiniált függvény.

public int getColumnCount()	Az oszlopok számának a gettere, visszaadja a táblázatban lévő oszlopok számát.
public String getColumnName(int column)	Az oszlop nevének a gettere. A paraméterként megadott oszlopszámhoz tartozó oszlopnevet adja vissza.
public Class<?> getColumnClass(int columnIndex)	A paraméterként megadott sorszámhoz tartozó oszlopban lévő fájlok osztályát adja vissza, getter.
public boolean isCellEditable(int rowIndex, int columnIndex)	A paraméterként megadott sor- és oszlopindexhez tartozó cella szerkeszthetőségének megfelelően tér vissza true-val vagy false-szal.
public void setValueAt(Object aValue, int rowIndex, int columnIndex)	Setter, a a paraméterként megadott értéket állítja be a sor-es oszlopindexhez tartozó cellának. Azért Object a típusa az értéknek, mert így a különböző osztálytípusú oszlopokra is érvényes.
public Object getValueAt(int rowIndex, int columnIndex)	Getter, a paraméterként kapott sor-és oszlopsorszámhoz tartozó cella értékét adja vissza.
public void addPlayer(String Name, String Mode, int Score)	Hozzáad egy új játékost a listához, a paraméterként kapott változóknak megfelelő adatokkal.

3.1.10 Main

Felelősségek

Ez az osztály felelős a program végrehajthatóságáért, futásáért. Innen indul a program.

Attribútumok: -

Metódusok

public static void main(String[] args)	Main függvény, amely példányosítja a MenuFrame osztályt.

3.1.11 Ranglist

Felelősségek

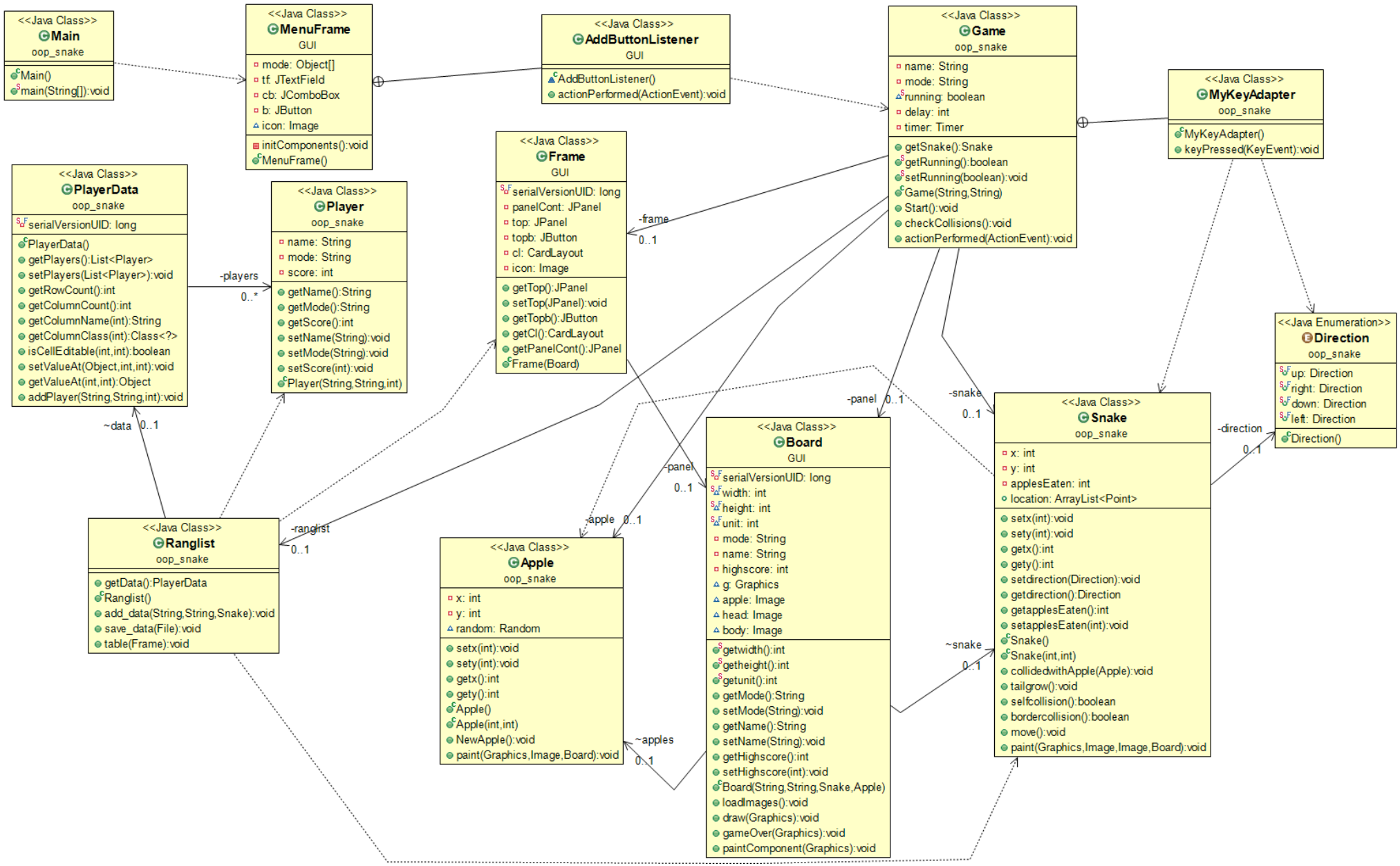
Ez az osztály felelős a „player.data” fájl beolvasásáért, ami tárolja az eddig játszott felhasználók neveit a játékmóddal és elért pontszámukkal együtt, továbbá felelős a játékosok ezen listájához új játékos adásáért, a „player.dat” fájl mentéséért, valamint a játékosok listájából készült táblázat létrehozásáért is.

Attribútumok

private PlayerData data	PlayerData típusú változó, amelybe majd beolvassuk az adott fájlban lévő adatokat. Ebből lesz majd a ranglistát megvalósító táblázat.

Metódusok

public Ranglist()	A Ranglist konstruktora. A PlayerData data egy try-catch blokkban példányodosik, hiszen a blokkon belül beolvassuk ObjectInputStream segítségével a megadott fájl adatait a data-ba. Fájlbeolvasásnál gyakran előfordulnak kivételek, ezért kezeljük őket a blokk segítségével.
public void add_data(String name, String mode, Snakes)	Ez a függvény hozzáad egy új adatrekordot a korábban már említett data-hoz. A paramétereiből az első kettő már eleve meghatározza a felhasználó nevét és a nehézségi szintet, a Snake pedig, mint parameter azért kell, mert a játék során megszerzett pontok egyenlőek a Snake által megevett almák számával. Így tehát már könnyen használhatjuk az alábbi parancsot: data.addPlayer(name, mode, s.getapplesEaten()); Ezt az add_data függvényt is majd a Game példánya hívja meg ha vége a játéknak, hiszen akkor kell hozzáadni egy játékost és az adatait a listához, ha már nem nő tovább a pontszáma.
public void save_data(File file)	Ez a függvény a megadott fájlba menti el ObjectOutputStream segítségével az adatokat a data listából. (kiírja őket oda). Ezt a függvényt is a Game példánya hívja meg.
public void table(Frame frame)	A függvény paraméterként kapja a Game által példányosított frame-et, hiszen abban az ablakba fog megjelenni, a frame-nek a top paneljén.Ez a függvény elkészít egy JTable-t a data lista alapján. Beállítja a táblázatban lévő szöveg színét, típusát, méretét, a cellák háttérszínét, a szegélyek színét, valamint külön az oszlopneveket is formázza nagyobbra illetve félkövérre. A DefaultTableCellRenderer osztályt pédányosítja, majd ennek segítségével középre igazítja minden cella tartalmát a szebb megjelenés érdekében. Mivel ez a táblázat majd egy ranglistaként szolgál, ezért rendezve van. Elsősorban a “Mode” szerint betűrendben, mert így olyan mintha csoportosítva lenne. A másodlagos rendezési kulcs a “Score”, ott a rendezés csökkenő, tehát mindig a legnagyobb értékhez tartozó adatok vannak legelől mindhárom nehézségi mós esetén is. A rendezéshez a TableRowSorter osztályt használtam, a rendezési kulcsokat ArrayList-ben tároltam el. Végül a sort() függvény rendzei a táblázatot a megadott adatoknak megfelelően.



3.2 OSZTÁLYDIAGRAM

4 BEMENETEK ÉS KIMENETEK

4.1 BEMENETEK

Miután a felhasználó kiválasztotta a menüablakból a kívánt beállításokat és rányomott a start gombra, a cp1250 kódolású „players.dat” fájlt beolvassa a program, hiszen annak tartalmát meg kell majd jelenítenie táblázatos formában, ha a felhasználó rákattint a „Ranglist” gombra. Game over esetén pedig a „highest.txt” fájlt olvassa be a program, mivel össze kell hasonlítania az aktuálisan játszott felhasználó adatait az eddigi rekordot tartó felhasználóéval. Ha az aktuális pontszáma nagyobb, akkor a fájlban is átírja a Stringet.

4.2 KIMENETEK

A játék végén minden játékos és a hozzá tartozó adat (játék nehézségi szintje, elért pontszám) a „players.dat” fájlba mentődik. Ha a játékos rekordot döntött, akkor a „highest.txt” fájlba mentődik el a neve és a pontszáma az alábbi formában: nev:pontszam

5 FELHASZNÁLÓI KÉSZIKÖNYV

A program indítása után megjelenik egy kis ablak a bal felső sarokban. Ezen található egy felirat: „Name:”. A felirat után van egy fekete, zöldes keretekkel ellátott téglalap, ebbe bele kell kattintani és begépelni a felhasználónevet, amit a játék során használni akar, lehetőleg 8 karakternél ne legyen hosszabb. A téglalap mellett található még egy felirat: „Mode:”. A felirattól jobbra található egy zöld téglalap, amelyen azt olvashatja, hogy „easy”. Ennek a téglalapnak a jobb szélén van egy kisebb, álló téglalap, amiben pedig egy fekete, lefele mutató nyíl. Ha erre rákattint, akkor megjelenik 3 téglalap alatta a következő feliratokkal: „easy”, „medium”, „hard”. Ezekre való kattintással lehet beállítani, hogy a játék könnyű, közepesen nehéz vagy nehéz legyen. „easy” módban a kígyó nagyon lassan mozog, így könnyű életben tartani. „medium” módban már sokkal gyorsabban mozog, itt már nehezebb nem veszíteni. „hard” módban pedig kifejezetten nehéz életben tartani a kígyót, mert annyira gyorsan mozog. Tehát a 3 opció közül az egyikre rá kell kattintani, majd ettől a téglalaptól is jobbra található egy „START” feliratú téglalap, amire ha rákattint, a játék elindul. A játék indulása megjelenik egy másik, nagyobb ablak a képernyő közepén, ezután már irányíthatja is a kígyót a pályán. A kígyót csak a navigációs billentyűk segítségével tudja irányítani. A navigációs billentyűk a billentyűzeten a nyilak. A kígyó mindig arra mozog, amilyen irányú nyilat nyomott be éppen (ez alól kivétel, hogyha pont ellentétes irányú nyilat nyom le mint amilyen irányba a kígyó éppen mozog). A kígyó magától megy, tehát csak a kanyarodásoknál szükséges nyomni a billentyűzetete, nem kell végig. A pálya fekete és rajta sötétszürke négyzetrács (vonalak függőlegesen és vízszintesen) vannak. Csak ezen a területen mozoghat a kígyó. A pályán mindig egy alma, valamelyik négyzetben. Az Ön célja az, hogy ennek az almának nekimenjem a kígyóval, mert ilyenkor a kígyó megeszi az almát, az ön pontszáma 1-gyel nő, a megevett alma eltűnik és egy új megjelenik helyette valahol a pályán. A játék során a „játék-ablak” felső részén

folyamatosan látja az ön által megadott felhasználónevet(„Name: xy), a módot amelyet a másik ablakban kiválasztott és amelyben épp játszik (Mode: xy) , illetve a pontszámát(Score: xy), ami mindig eggyel növekszik, ha a kígyó megevett egy almát. Ha a kígyó a pálya valamelyik szélének (tehát a szürke, nem pedig fekete és négyzetrácsos rész) nekimegy, vagy a saját testébe megy bele (pl. egy visszafele-kanyarodásnál, mint egy bezárt hurok), akkor veszített és megjelenik a „game over” képernyő. Ezen a képernyőn felül megjelenik a „Highscore: x:y” felirat, ahol az x azt a felhasználót jelenti, aki a mindenkori, játékmódtól független legmagasabb pontott gyűjtötte eddig a snake játékban, az y pedig az ezáltal a felhasználó által megszerzett pontot jelöli. Alatta van a „Score: x) felirat, amely pedig az ön által a lejátszott játékban elért pontot jelöli. Ez alatt pedig van egy „Game Over…” felirat is, ami jelzi önnek, hogy vége a játéknak, vesztett. Még az ablak alján található egy „Ranglist” feliratú téglalap(ami egy gomb), amelyre ha rákattint, megtekintheti a ranglistát, amiben az összes, eddig a játékkal játszott felhasználó neve, a mód, amiben játszottak, illetve az általuk megszerzett pontszám is megtalálható táblázatba rendezve, mód szerint csoportosítva és pontszám szerint csökkenő sorrendbe rendezve. Ez a „Ranglist” gomb a játék során is ott van alul végig, de hiába kattintana rá, olyankor nem tekintheti meg a ranglistát. Ha ki szeretne lépni a programból, csak kattintson a 2 ablak közül bármelyiknek az (x) ikonjára, amit a jobb oldali felső sarokban talál meg. Ha viszont szeretné még játszani, akkor csak annyi a dolga, hogy azokat a beállításokat, amiket ennek a kézikönyvnek az elején leírtunk beállítja, hiszen az a bal oldali „menü-ablak” a játék során sem tűnt el. De akár a beállítások megváltoztatása nélkül is újraindul a játék, ha rákattint a”START” gombra.