HÁZI FELADAT

Szoftver laboratórium 3.

Dokumentáció

Bunta Roland Árpád

BI5UYV

Feladat ismertetése:

Egy memória játékot fogok készíteni.

Ez azt jelenti, hogy egy van 5x4 (20) lap lefordítva és ezek között 10 pár található, ami azt jelenti, hogy ezeken a kettő lapokon ugyanaz a szimbólum (pl: szám) szerepel.

Egy játékos játssza, és felfordít 2 lapot. Ha a két lapon szereplő szimbólumok megegyeznek, akkor azok felfordítva maradnak és a többi lap közül folytathatja a további két lap felfordítását. Ha viszont nem azonosak a szimbólumok, akkor azok visszafordítódnak.

A játéknak akkor van vége, hogyha az összes lap fel van fordítva.

Ehhez egy menüt fogok létre hozni, amiben meg lehet tekinteni a rekordokat is, ami a szerint állít sorrendet, hogy milyen rövid idő alatt sikerült megnyerni a játékot.

USE-CASE:

A kezdés után a Menübe kerül a felhasználó.

Menü:

- új játék -> elindul egy új memória játék
- rekordok -> kiírja a rekordokat
- kilépés -> kilép a programból

Játék:

- újrajátszás -> új játék kezdődik
- vissza -> visszalép a menübe

Rekordok:

- vissza -> visszalép a menübe

Megoldási ötlet:

SWING-alapú GUI-val valósítom meg a grafikai megjelenítést. Egy 2D tömböt készítek JButton-ökből, amit egy JPanel-re rakok GridLayout-ot használva.

A tömb elemeit fájlból olvasom be, amibe meg lehet majd adni milyen szimbólum párokat kelljen keresni. Mivel a tömb 4szer 5 vagyis 20 elemű, de 2es párokban vannak, ezért a fájlban csak 10 elem lesz és ezeket fogom elhelyezni 2 helyre véletlenszerűen.

Amikor fordítja a lapokat és a pár második lapját is felfordította, de nem egyeznek meg a szimbólumok, akkor a második lap is egy ideig felfordítva marad, hogy megjegyezhesse a játékos.

Egy időzítő is lesz benne, ami méri mikor nyerte meg a játékot. Ilyenkor meg lehet adni egy nevet ami az idővel együtt eltárolódik a rekordokba, amit egy fájlba mentek ki és olvasok be. Elől szerepel a legrövidebb idővel rendelkező és véges számú adatot tárol, tehát ha az utolsó időnél lassabb és nem engedek bele több elemet, akkor nem kerül bele.

Serializáció is lesz benne, ki lehet menteni a gombok értékeit.

Gyűjtemény keretrendszert alkalmazok, melyekben külön valósítom meg az osztályokat.

↑ ↑ ↑

Specifikáció

_____

Változások:

A játékban 4x4es táblát csináltam, mivel ez még hamar megoldhatóvá teszi és később, ha több szintet szeretnénk megvalósítani ez a kiindulás.

Az értéket nem egy felhasználótól megadott fájltól olvasom be, mert lehetnek benne olyan értékek, amik nagyon hosszúak, vagy olyan szimbólumok, amit nem lehet megjeleníteni minden gépen. Ezért egy ArrayList-be számokat.

Két lap felfordítása után, ha megegyezik a két lap, akkor a háttere színe megváltozik fehérről feketére, ezzel is jobban szemléltetve, hogy azok megvannak. Ha viszont nem egyeznek meg, akkor fordulnak vissza, amikor egy újabb lapot kiválasztott, így több ideje van megjegyezni azokat.

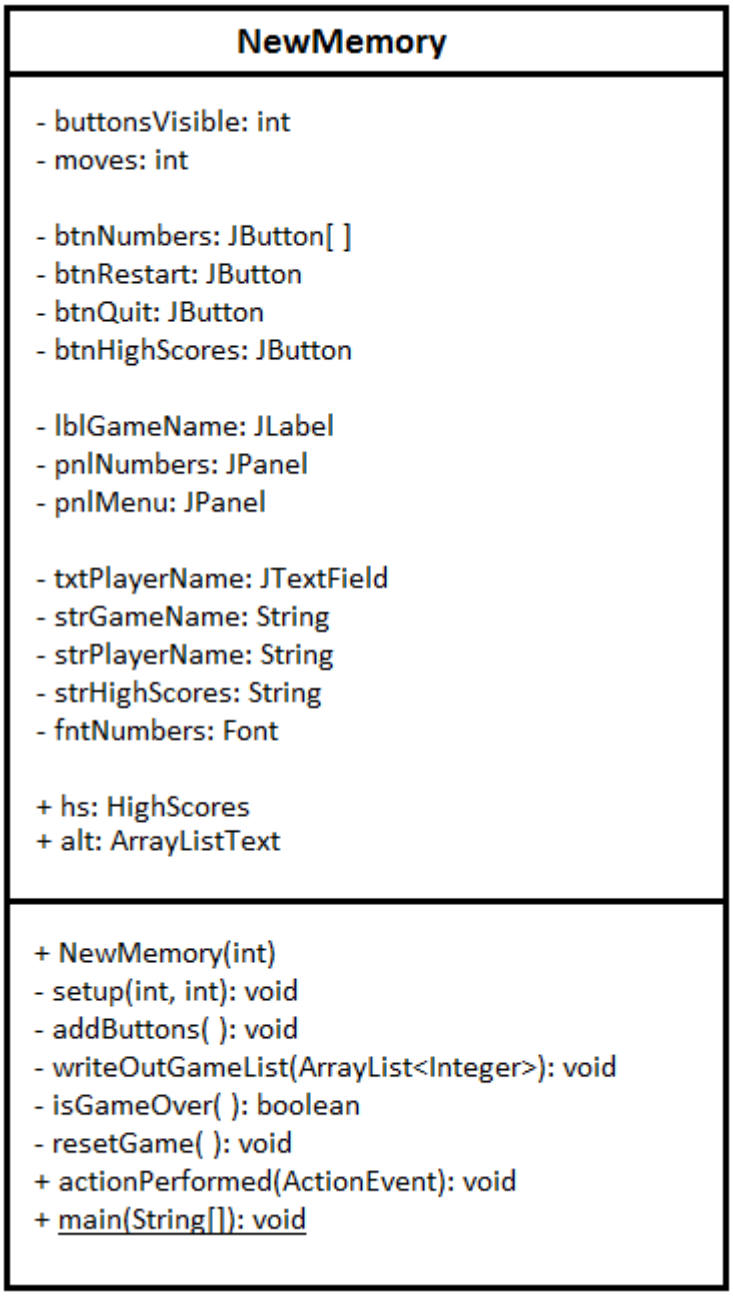
A Rekordokba a játékosok lépés alapján kerülnek bele, mivel sokkal értelmesebb a rangsor állítás, ez teszteli tényleg a memóriát, nem az ha gyorsan végig kattintgat. Maximum 10 eredmény kerülhet be a Rekordokba, ha azonos a lépésszámuk, akkor az lesz előrébb aki újabban játszott. Egy lépésnek számít két lap kiválasztása.

Tehát az időzítő helyett egy lépésszámláló lesz, ami akkor jelenik meg, ha elkezdte a játékot. Addig a Memória Játék felirat látható a helyén.

OSZTÁLYOK

A változók kifejtése a kódban megtalálható, a függvényeket itt is leírom részletesebben.

NewMemory.java



Változók:

buttonsVisible: int
moves: int

btnNumbers: JButton[ ]
btnRestart: JButton
btnQuit: JButton
btnHighScores: JButton

lblGameName: JLabel
pnlNumbers: JPanel
pnlMenu: JPanel

txtPlayerName: JTextField
strGameName: String
strPlayerName: String
strHighScores: String
fntNumbers: Font

hs: HighScores
alt: ArrayListText

Függvények:

NewMemory(int) - konstruktor, int: alap esetben 16ot adunk meg, ez lesz a pálya mérete

setup(int, int) - felállítjuk a pályát, az átadott paraméterek lesznek a felbontás,
nem lehet újraméretezni, rátesszük a menü panelt, és erre a jön a játék neve, ami a játék elindítása után a lépések számát írja majd ki, a gombok: rekordok, újrajátszás, kilépés, „Játékos neve” szövegmező, és egy írható szövegmező amibe írhatja a játékos a nevét

addButtons( ) - itt létrehozzuk a gombokat, háttér fehér, még text nincs rajtuk, actionlistenerhez hozzáadjuk

writeOutGameList(ArrayList<Integer>) - ez kiírja a gombok értékét, rendezetten 4x4es alakban így ez könnyen megfeleltethető a játéktáblának

isGameOver( ) - akkor van vége a játéknak ha az összes gomb párját már megtaláltuk, de ha van olyan aminek még nem akkor nincs vége

resetGame( ) - ha új játék kezdődött akkor a gombok színe fehér lesz, text nincs bennük, ilyenkor újragenerálja a számokat és szerializálás után kiíratja a konzolra az értékeket

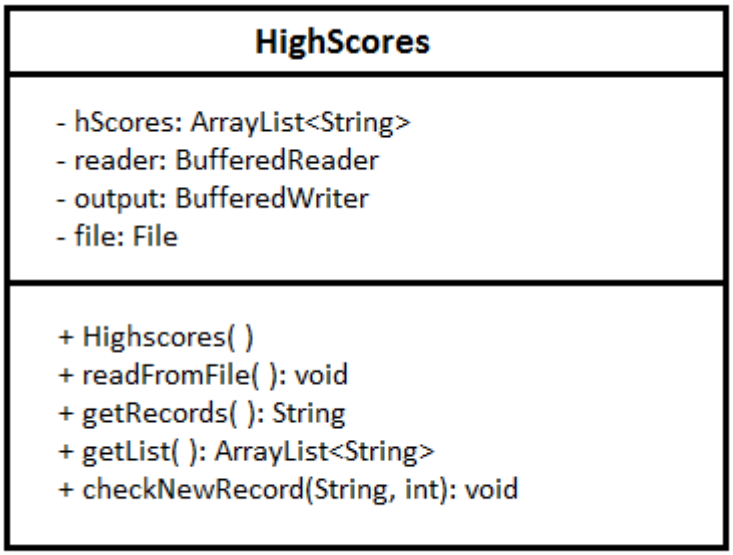
actionPerformed(Actionevent) – Eseménykezelő, esetek:

- quit gomb: kilépés
- restart gomb: új játék kezdése
- playerName szövegmezőbe írás: játékos nevének elmentése
- highScores gomb: JOptionPane.showMessage segítségével kiíródnak az eredmények, először a lépésszám majd a játékos neve
  pl. „12 lépés – Kiss János”

Ha nincs vége a játéknak, akkor ha egy kártyára kattintottunk akkor az értéke kiíródik és ha az utána kiválasztott lapnak ugyanaz az értéke akkor fekete színre változnak és kimentek a játékból.
Ha vége van a játéknak, akkor megnézi, hogy ezzel a lépésszámmal be kerülhet-e a rekordok közé és ha igen akkor berakja a megfelelő helyre(checkNewrecord). Ezután kiíródik a képernyőre egy választási lehetőség, új játék vagy kilépés.

main(String[ ]) - main metódus, NewMemory(16) hívódik meg.

HighScores.java



Változók:

hScores: ArrayList<String>
reader: BufferedReader
output: BufferedWriter
file: File

Függvények:

HighScores( ) – konstruktor

readFromFile( ) – beolvassa a fájlból az adatokat soronként, egy sor egy String lesz benne, kivételkezelés.

getRecords( ) – a StringBuilder segítségével az ArrayList elemeit szétszedem és belerakom egy Stringbe, új sorral elválasztva őket és ezt visszaadom.

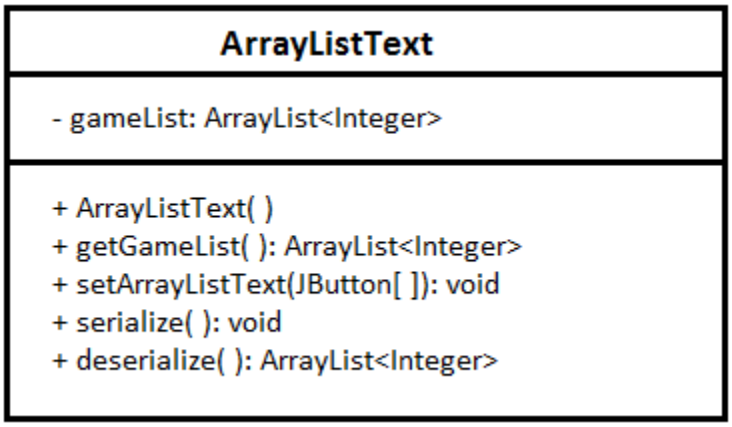
getList( ) – visszaadom a Listát

checkNewRecord(String, int) – paraméterként kap egy nevet és egy lépésszámot, amit le kell osztani 2vel mivel minden helyes kattintásra növekedik. Alapértelmezett játékos név: „Névtelen játékos”, ha nem volt elem a rekordok listában, akkor az új elem lesz az első. Forma pl. „10 lépés – név”.

Viszont ha már voltak benne elemek, akkor végig kell ezeken menni fentről lefele, megnézni mi az első helyen lévő szám bennük és ezt kell összehasonlítani az újonnan érkező lépésszámmal. Ha az új kisebb vagy egyenlő, akkor az az elem elé beszúrjuk. Ilyenkor hátrébb csúszik a többi, a lista mérete nő, vagyis meg kell vizsgálni nem léptük-e át a 10es maximum méretet. Ha igen akkor a 11. elemet törölni kell. Ezután az új listát beírjuk a fájlba.

Ha végigértünk és mindegyik elemnél nagyobb volt, akkor berakjuk a végére, ha az elemszám még nem éri el a 10et. Ha eléri, nem csinálunk semmit.

ArrayListText.java



Változók:

gameList: ArrayList<Integer>

Függvények:

ArrayListText( ) – konstruktor

getGameList( ) – visszatér a játéklistával(integereket tartalmazó lista)

setArrayListText(Jbutton[ ]) – a játéklistát feltölti véletlen számokkal, a gomb tömb elemszámának figyelembe vételével. Pl. 16gombnál -> 1től 8ig számok, mindegyik kétszer szerepel. Shuffle – megkeveri az elemeket.

serialize( ) – szerializálja a játéklistát, így később bármikor vissza tudjuk kérni az adatokat, esetleg elmenteni ezeket. A *serialization.txt*-be írja ki.

deserialize( ) – a szerializált játéklistát visszaolvassa a fájlból, így felhasználó számára is olvasható formába kiíratható lesz.

Testing.java

Ebben található a JUnit-tal megoldott tesztelés.

TEST 1: ArrayListText tesztelése, ha létrehozok 16 gomb átadásával egy új számokat tároló listát, akkor pontosan 16 elem lesz benne.

TEST 2: Ha ennek a listának az elemeit megnézem, akkor az első elem értéke 0-nál nagyobb és 9nél kisebb lesz.

TEST 3: HighScores tesztelése, feltöltök egy eredmény listát, ha van benne legalább két elem, akkor az első két elem lépésszámát összehasonlítom, és az előrébb lévő elemnek lépésszáma kisebb vagy egyenlő lesz az utána lévő elem lépésszámához képest.

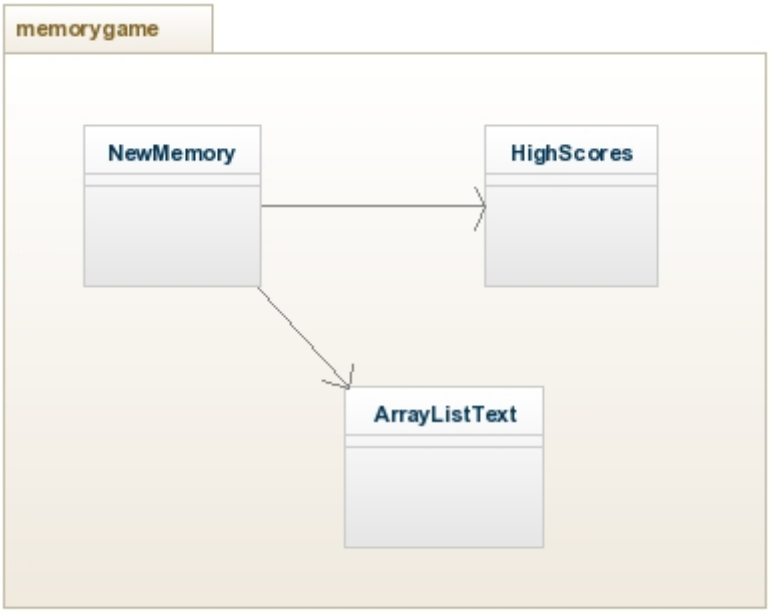
TEST 4: Ha a lista betelt, vagyis 10 elem van benne, akkor, ha jön egy új elem, akkor a lista mérete nem fog változni.

TEST 5: Ha az újonnan érkező elemnek kisebb a lépésszáma, mint az első elemnek, az új elem lesz az első helyen.

Fájlok:

rekordok.txt: ebből olvassunk be a rekordokat tartalmazó listába és változás esetét ebbe írjuk ki őket. Olvasható formában szerepelnek benne az adatok.

serialization.txt: ebbe kerül bele a számokat tartalmazó lista szerializációja, majd de-szerializációnál ki is írhatjuk.



Ábrák, diagramok, példák

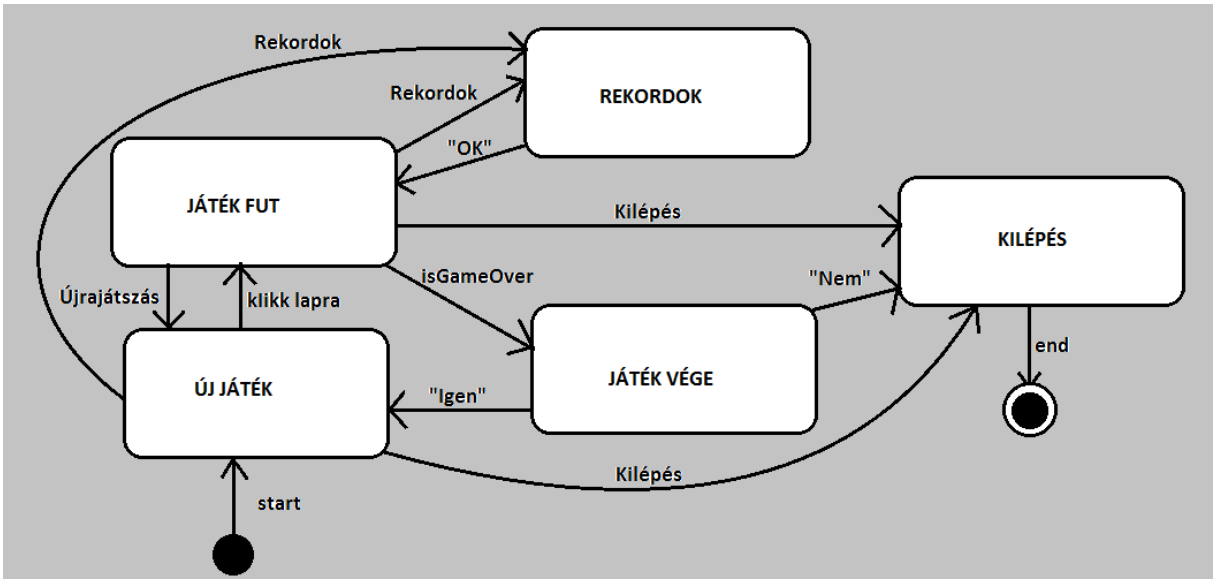
Menüben:
Új Játék:
- klikk lapra: elindul a játék
- Rekordok: megjeleníti a rekordokat
- Kilépés: kilép a játékból

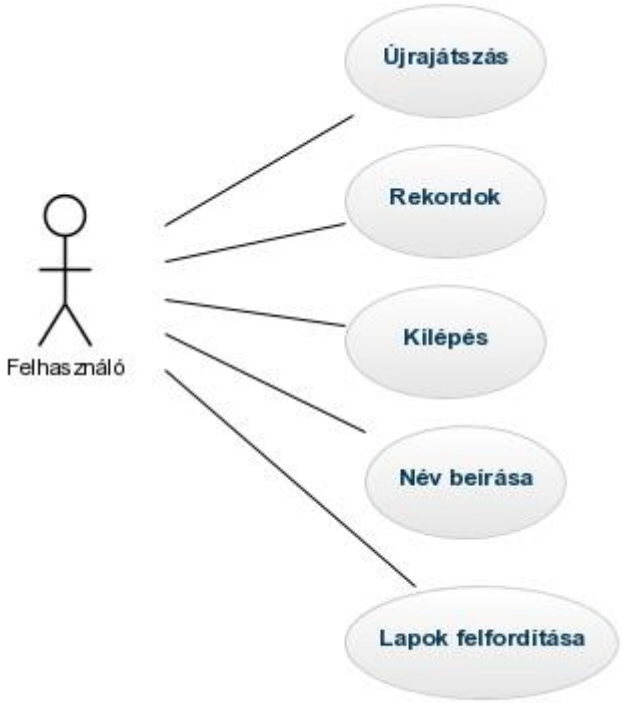
Játék fut:
- Újrajátszás: új játék indul
- Rekordok: megjeleníti a rekordokat
- Kilépés: kilép a játékból
- isGameOver: ezt a felhasználó nem éri el közvetlen! Ha megtalálta az összes párt, akkor a játéknak vége, de még nem lép ki a program

Rekordok:
- „OK”: visszalép az előző állapotba ahonnan jött

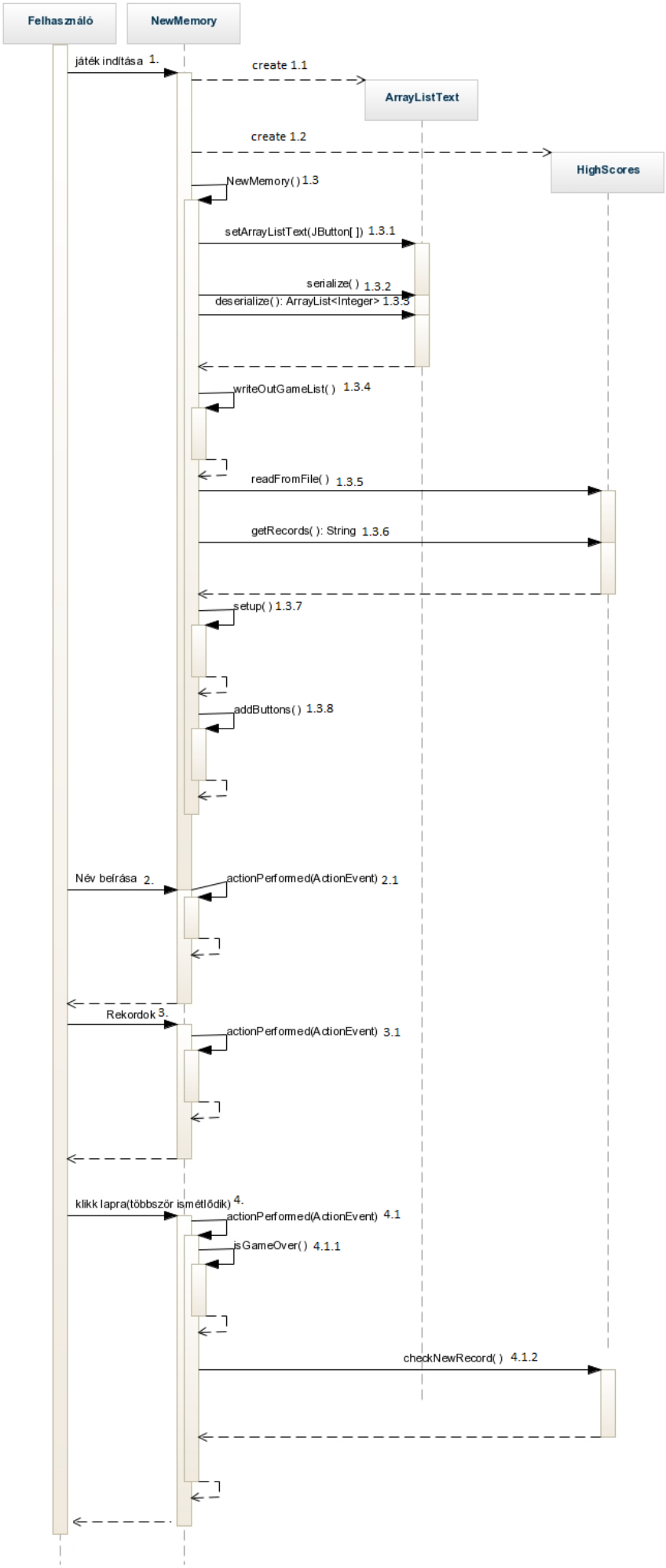
Játék vége: (feltesz egy kérdést, hogy szeretne-e új játékot kezdeni)
- „Igen”: új játék indítása
- „Nem”: kilép a programból

Kilépés:
- innen egyből bezárja a programot





Példa a játék használatára:
A program elindítása után a játékos beírja a nevét, ezután rákattint az eredményekre, majd miután megnézte azokat ráment az okéra és folytatja a játékot. Amikor megtalálta az összes párt, akkor miután felugrott az új játékra hívó ablak, ráment a nemre és a program kilépett.


Felhasználó
NewMemory
játék indítása 1.
create 1.1
ArrayListText
create 1.2
HighScores
NewMemory( ) 1.3
setArrayListText(JButton[ ]) 1.3.1
serialize( ) 1.3.2
deserialize( ): ArrayList<Integer> 1.3.3
writeOutGameList( ) 1.3.4
readFromFile( ) 1.3.5
getRecords( ): String 1.3.6
setup( ) 1.3.7
addButtons( ) 1.3.8
Név beírása 2.
actionPerformed(ActionEvent) 2.1
Rekordok 3.
actionPerformed(ActionEvent) 3.1
klikk lapra(többször ismétlődik) 4.
actionPerformed(ActionEvent) 4.1
isGameOver( ) 4.1.1
checkNewRecord( ) 4.1.2

Képek a játék állapotairól: