A PROGRAMOZÁS ALAPJAI 3 (BMEVIIIAB00)

Gyöngyösi Péter

M80ILY

HÁZI FELADAT DOKUMENTÁCIÓ

Feladat rövid szöveges ismertetése:

A **sudoku (szúdoku)** egy logikai játék, melyben megadott szabályok szerint számjegyeket kell elhelyezni egy táblázatban. A legközönségesebb változatáról lesz szó a következőkben. Egy 9x9-es táblázaton belül van 9 db 3x3-as résztáblázat. Minden résztáblázatot 1,2,3,4,5,6,7,8,9 számokkal kell kitölteni úgy, hogy az egész 9x9-es táblázat minden sorában és minden oszlopában az 1…9 számok mindegyike egyszer forduljon elő. A résztáblázatokban is egy szám csak egyszer szerepelhet. A rejtvény készítője a megoldhatóság érdekében előre ki szokta tölteni néhány szükséges számmal a 9x9-es táblázat bizonyos celláit – általában úgy, hogy csak egy megoldása létezzen a rejtvénynek. A rejtvényfejtő feladata kitölteni a maradék cellákat a fentebb írt feltételeknek megfelelően.

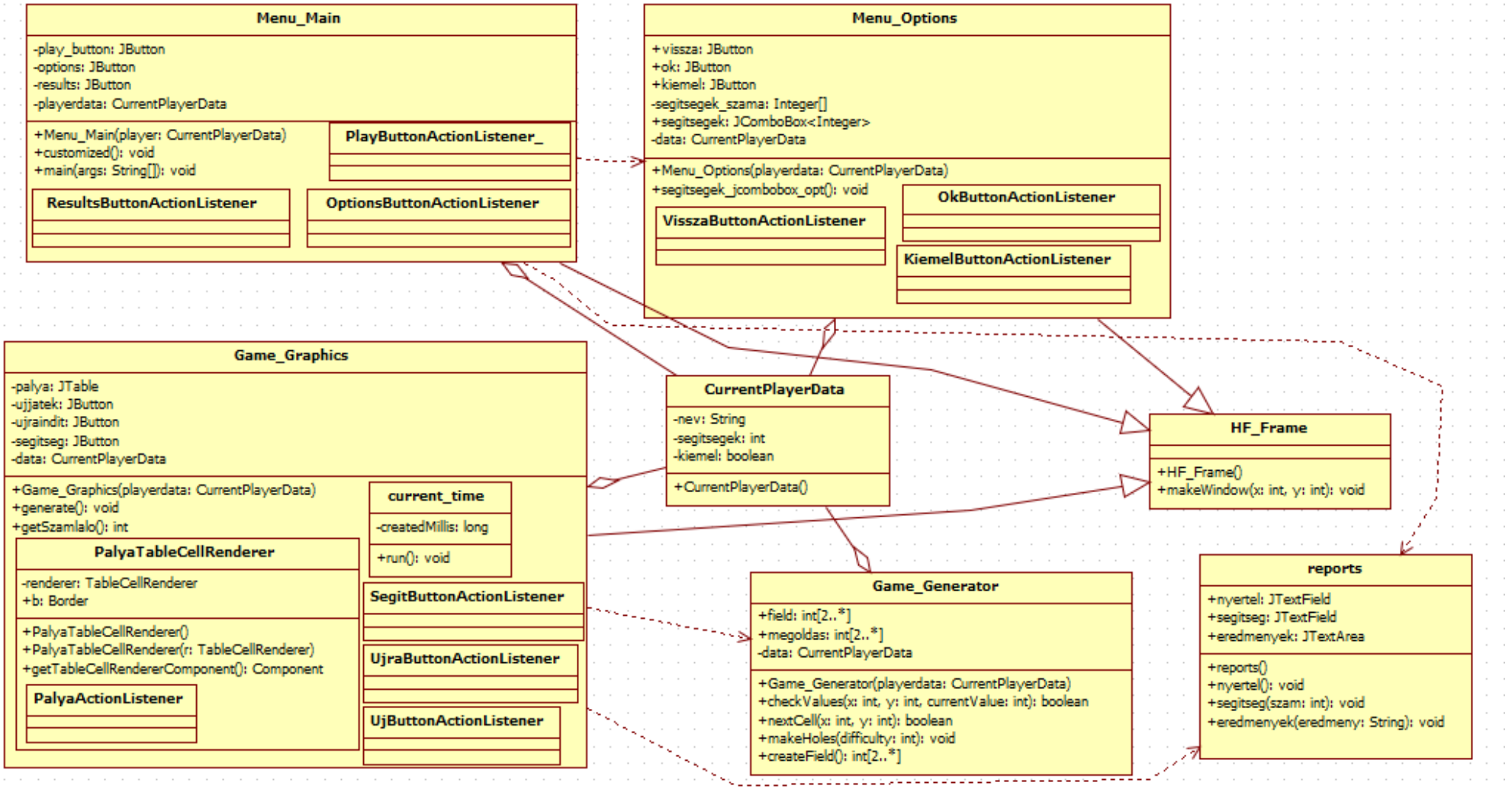
	3							
			1	9	5			
		8					6	
8				6				
4			8					1
				2				
	6					2	8	
			4	1	9			5
							7	

A játék kezdetén a sudoku-tábla.

A „szúdoku” név egy hosszabb japán kifejezés rövidítése. Az eredeti név jelentése: „a számjegyek csak egyszer szerepelhetnek”. A szimmetrikus (egymásba tükrözhető, forgatható) megoldások megkülönböztetésétől eltekintve, a 9×9-es szúdoku lehetséges kitöltéseinek száma

6 670 903 752 021 072 936 960 azaz közel 6,7 trilliárd.

AZ OSZTÁLYDIAGRAM



Gyöngyösi Péter
M80ILY

A megvalósított metódusok leírása:

Menu_Main.java:

class Menu_Main

Menu_Main(player: CurrentPlayerData)

Ebben a konstruktorban hívom meg a komponensek tulajdonságait beállító függvényeket, majd adom hozzá a JFrame-hez őket. Ezenkívül Layout-ot és hátteret állítok be.

play_button_opt(): void

A *play_button* JButton-t allocálja és tulajdonságait állítja be.

options_button_opt(): void

Az *options* JButton-t allocálja és tulajdonságait állítja be.

results_button_opt(): void

Az *results* JButton-t allocálja és tulajdonságait állítja be.

sudoku_jtextfield_opt(): void

Az *sudoku* JTextField-et allocálja és tulajdonságait állítja be.

kiiras_jtextfield_opt(): void

Az *kiiras* JTextField-et allocálja és tulajdonságait állítja be.

default_(): void

Beállítja a program indulásakor az alapértelmezett beállításokat. Későbbiekben akkor hívódik majd meg, ha a játékos neve: 'Unknown'. A *kiiras* és a *play_button* komponensek tulajdonságait változtatja meg (szöveg, szín).

customized(): void

A testreszabott beállításokkor hívódik meg (amennyiben a játékos neve nem 'Unknown'). A *kiiras* és a *play_button* komponensek tulajdonságait változtatja meg (szöveg, szín).

class PlayButtonActionListener

mouseClicked(e: MouseEvent): void

Bezárja az ablakot. Generál egy pályát, majd új ablakot nyit a meghatározott tulajdonságoknak megfelelően.

mouseEntered(e: MouseEvent): void

Megvastagítja a gomb szegélyét, ha fölé visszük az egeret.

mouseExited(e: MouseEvent): void

Megvékonyítja a gomb szegélyét, ha elvisszük róla az egeret.

mousePressed(e: MouseEvent): void

Megváltoztatja a háttérszínét, ha nyomva tartjuk.

mouseReleased(e: MouseEvent): void

Visszaváltoztatja az eredeti háttérszínét.

A továbbiakban a hasonló osztályoknak (implementálják a MouseListener-t) a függvényeit akkor részletezem, ha a fentiektől eltérő viselkedést mutatnak!

class OptionsButtonActionListener

mouseClicked(e: MouseEvent): void

Bezárja az ablakot. Új ablakot nyit, ahol be lehet állítani, milyen tulajdonságokkal szeretnénk játszani.

class ResultsButtonActionListener

mouseClicked(e: MouseEvent): void

A *results.txt* fájlból megjeleníti a játék történetében nyert játszmák játékos neveit és a hozzájuk tartozó időt.

Menu_Options.java

class Menu_Options

Menu_Options(playerdata: CurrentPlayerData)

Ebben a konstruktorban hívom meg a komponensek tulajdonságait beállító függvényeket, majd adom hozzá a JFrame-hez őket. Ezenkívül Layout segítségével elhelyezem a komponenseket.

testreszab_jtextfield_opt(): void

A *testreszab* JTextField tulajdonságit állítja be, ill. foglal memóriát.

vissza_button_opt(): void

A *vissza* JButton tulajdonságit állítja be, ill. foglal memóriát.

nev_jtextfield_opt(): void

A *nev* JTextField tulajdonságit állítja be, ill. foglal memóriát.

nevinfo_jtextfield_opt(): void

A *nevinfo* JTextField tulajdonságit állítja be, ill. foglal memóriát.

ok_jbutton_opt(): void

Az *ok* JButton tulajdonságit állítja be, ill. foglal memóriát.

segitsegekinfo_jtextfield_opt(): void

A *segitsegekinfo* JTextField tulajdonságit állítja be, ill. foglal memóriát.

segitsegek_jcombobox_opt(): void

A *segitsegek* JComboBox tulajdonságit állítja be, feltölti 1 és 5 közöti számokkal, ill. foglal memóriát.

kiemel_jbutton_opt(): void

A *kiemel* JButton tulajdonságit állítja be, ill. foglal memóriát.

class VisszaButtonActionListener

mouseClicked(e: MouseEvent): void

Bezárja az ablakot. Megnyitja a kezdőoldalt, az alapértelmezett beállításokkal.

class OkButtonActionListener

mouseClicked(e: MouseEvent): void

Bezárja az ablakot. Megnyitja a kezdőoldalt, a testreszabott beállításokkal.

class KiemelButtonActionListener // NEM VALÓSULT MEG

mouseClicked(e: MouseEvent): void

Beállítja, hogy játék közben ha egy szám fölé visszük a kurzort, az összes hasonló számot megnagyobbítsa.

Game_Generator.java

class Game_Generator

Game_Generator(playerdata: CurrentPlayerData)

Paraméterként kapott adatot elmenti, a mátrixokat inicializálja.

checkValues(x: int, y: int, checkValue: int): boolean

Megvizsgálja, hogy a paraméterként kapott szám szerepelhet-e a paraméterként kapott koordinátákon. Megvizsgálja a sort/oszlopot, ill. a 3x3-as mátrixokat is. Igaz értékkel tér vissza, amennyiben a szám szerepelhet, hamissal, ha nem.

nextCell(x: int, y: int): boolean

Feltölti a mátrixot 0 és 9 közötti számokkal. Feltételnek használja az előző függvényt. Rekurzívan meghívja önmagát, mindig a következő cellára. Ha már nem tud számokat generálni, hogy helyes legyen, de még nem töltötte fel teljesen a mátrixot, rekurzívan meghívja az összes korábbi függvényt false értékkel, majd kezdi elölről az egészet odáig, amíg fel nem töltötte a mátrixot. Vélhetőleg nagyon sokszor fut le.

makeHoles(difficulty: int): void

Generál véletlen helyeket a mátrixban. Paraméterként megadhatjuk, hogy hány üres helyet (0-ásat) csináljon. A felhasználói felületen erre nincsen lehetőség.

createField(): int[][]

A fenti függvények segítségével létrehozza a megoldandó sudoku mátrixot, ezenkívül létrehoz egy másik mátrixot és beállítja, melyek voltak a gép által helyén hagyott számok, és melyek a felhasználó által bevittek.

getColumnCount(): int
getRowCount(): int
getValueAt(rowIndex: int, columnIndex: int): Object
setValueAt(value: Object, row: int, col: int): void
isCellEditable(row: int, col: int): boolean
getColumnClass(c: int): Object

A fenti függvények beállítják, hogy a JTable-be fel lehessen tölteni, illetve beállítják, mit lehet módosítani és mit nem. Ezenkívül egy megoldás mátrixszal folyamatosan összeveti a jelenlegi táblát és ha egyezés van kiírja fájlba az eredményt.

Game_Graphics.java

class Game_Graphics

Game_Graphics(playerdata: CurrentPlayerData)

Ebben a konstruktorban hívom meg a komponensek tulajdonságait beállító függvényeket, majd adom hozzá a JFrame-hez őket. Ezenkívül Layout-ot, hátteret állítok be és időzítőt indítok el.

generate(): void

A JTable-t feltöltöm a Game_Generator segítségével.

counter_jtextfield_opt(): void

A *counter* JTextField tulajdonságit állítja be, ill. foglal memóriát.

palya_jtable_opt(): void

A *palya* JTable tulajdonságit állítja be, az alapértelmezett megjelenítését (renderer) átállítom ill. foglal memóriát.

ujjatek_jbutton_opt(): void

A *ujjatek* JButton tulajdonságit állítja be, ill. foglal memóriát.

segitseg_jbutton_opt(): void

A *segitseg* JButton tulajdonságit állítja be, ill. foglal memóriát.

ujraindit_jbutton_opt(): void

A *ujraindit* JButton tulajdonságit állítja be, ill. foglal memóriát.

class current_time

run(): void

Időzítő beállításai. Az időzítő leáll, ha a tábla helyesen ki van töltve.

getSzamlalo(): static int

Visszaadja a számláló értékét.

class PalyaTableCellRenderer

PalyaTableCellRenderer()
PalyaTableCellRenderer(r: TableCellRenderer)
highlightCell(szam: int): void // NEM VALÓSULT MEG
getTableCellRendererComponent(palya: JTable, value: Object, isSelected: boolean, hasFocus: boolean, row:int, col: int): Component

A *palya* JTable megjelenítését szabályozzák. Szegélyek, hátterek, betűszínek, méretek egyedi beállítások szerint.

class UjButtonActionListener

mouseClicked(e: MouseEvent): void

Bezárja az ablakot. Visszalép a főmenübe.

class SegitButtonActionListener

mouseClicked(e: MouseEvent): void

Megmondja, hány darab hiba van jelenleg a kitöltésben.

class UjraButtonActionListener

mouseListener(e: MouseEvent): void

Kitörli az eddig bevitt értékeket és újraindítja az időzítőt.

HF_Frame.java

class HF_Frame

makeWindow(x: int, y: int): void

Létrehozza az ablakot paraméteresen és középre igazítja. Ebből az osztályból származnak le a fenti osztályok.

CurrentPlayerData.java

class CurrentPlayerData

CurrentPlayerData()

Beállítja az alapértelmezett tulajdonságokat ('Unknown', 3, false);

Továbbiakban getter/setter függvények. Nem részletezem.

reports.java

class reports

nyertel(): void

Meghívódik, és kis ablakban jelzi, hogy nyertél, amennyiben helyesen raktad ki a táblát.

segitseg(szam: int): void

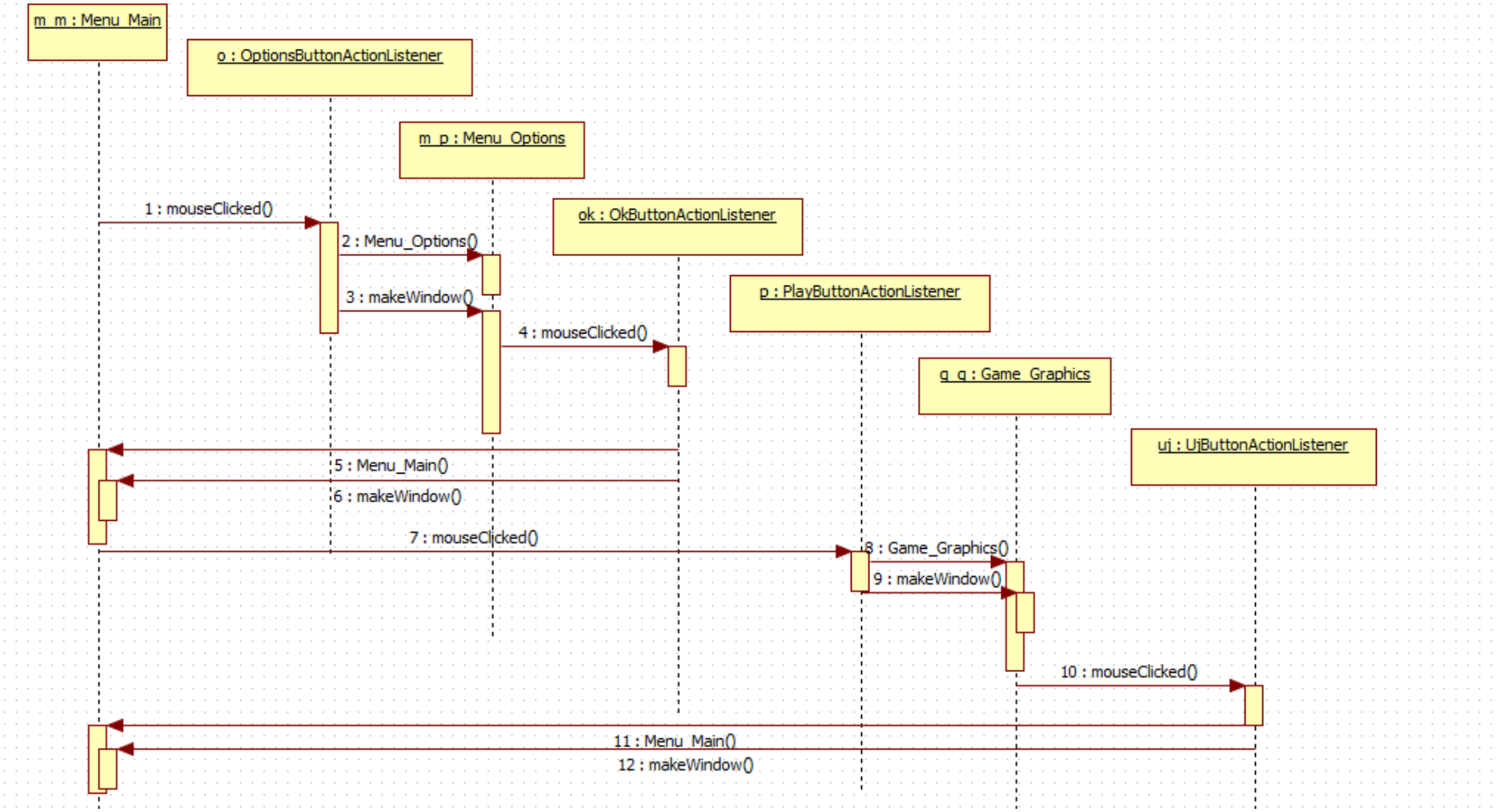
Meghívódik, és kis ablakban jelzi, hány hibás kitöltésed van eddig.

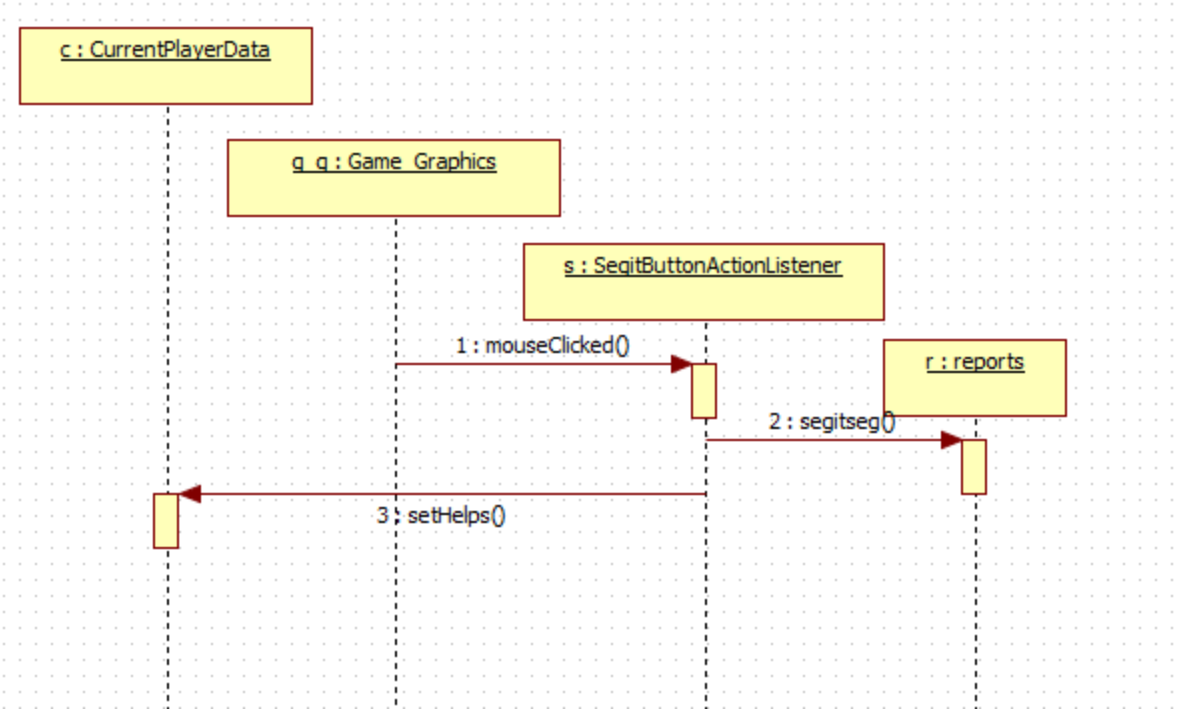
eredmenyek(eredmeny: String): void

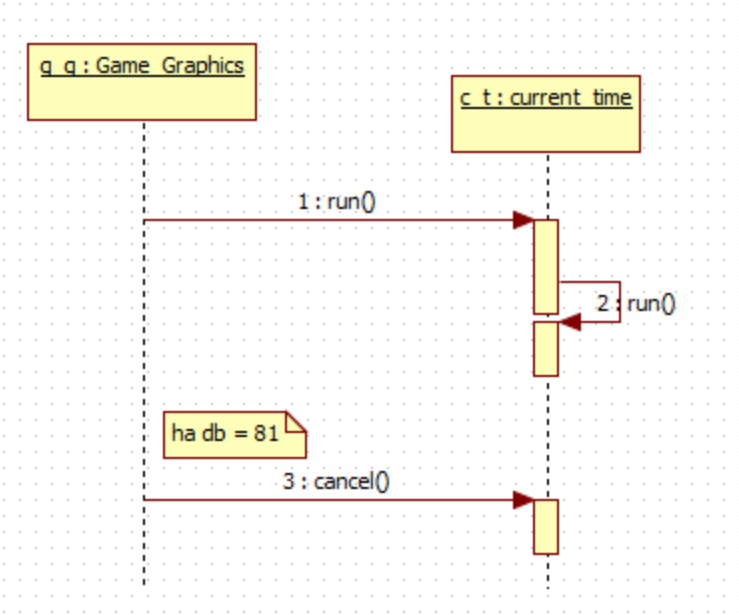
Meghívódik a menüben és ablakban kiírja a fájlban tárolt eredményeket.

A programhoz tartozó junit tesztek: *GameTest.java, MenuTest.java, ReportsTest.java*.

SZEKVENCIA DIAGRAMOK







Gyöngyösi Péter
M80ILY