Spark

Kovács Bertalan
RHDBLM

2022. 11. 22.

Tartalom

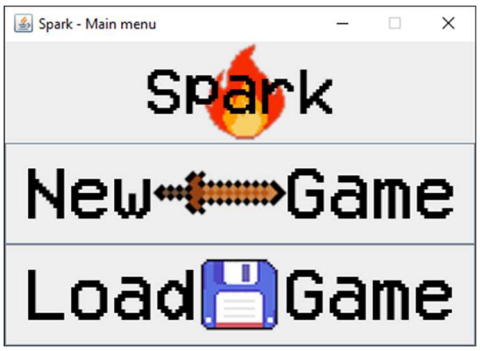
Felhasználói kézikönyv

A program kinézete és használata

A program két főbb nézettel rendelkezik, valamint egyik nézeten belül található kettő másik alnézet is. Ezek a következők:

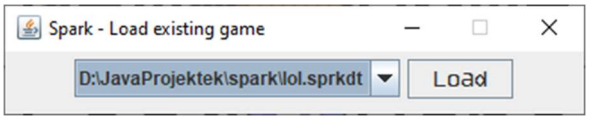
- Főmenü
  - Új játék kezdése
  - Meglévő játék betöltése
- A játék nézete

A program felhasználója ezen nézetek között navigálhat nyomógombok segítségével. A különböző nézetek az alábbi képeken láthatóak:

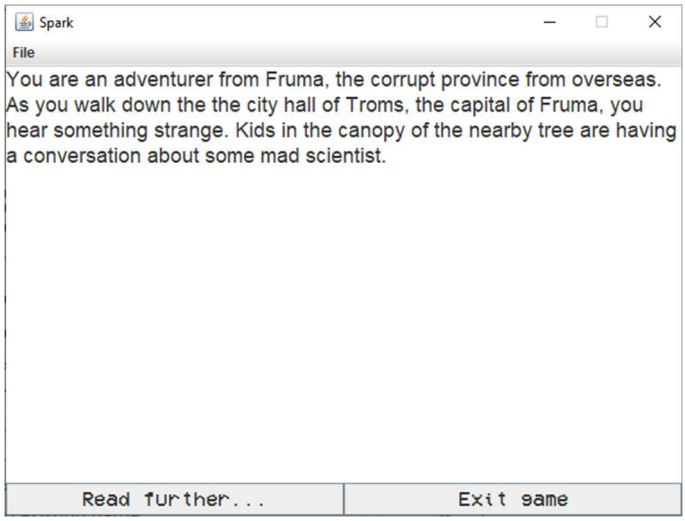


1. kép
A főmenü nézete

2. kép
Az új játék nézete



3. kép
A meglévő játék betöltése nézet



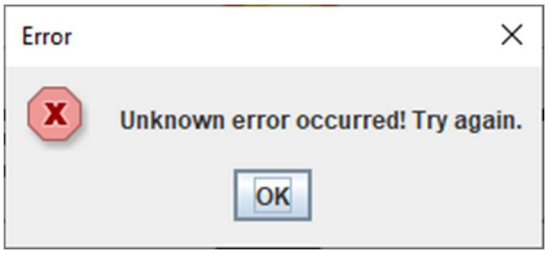
4. kép
A játék nézete

A főbb nézetek részletesebb leírása a következő oldalon szerepel.

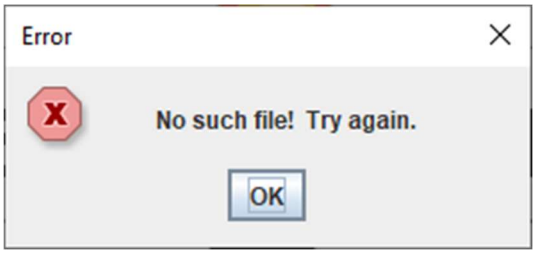
A főmenü nézete és használata

A főmenüben a felhasználó két gomb közül választhat. A „New Game” gomb megnyomása után az új játék kezdése ablak ugrik fel (ld.: 2. kép), ahol a felhasználó beírhat egy , a program követelményeinek megfelelő, játékfájl nevét és ezután, az ablakban szereplő „New Game” gombot megnyomva, kezdhet egy új játékot.

Hibás játékfájl vagy nem létező játékfájl esetén a program a következő hibát dobja:



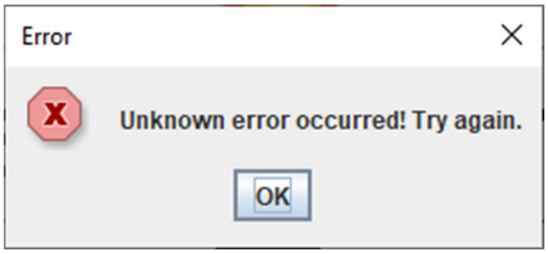
5. kép
A hibás fájl hibája



6. kép
Nem létező fájl hibája

A második gombot választva a felhasználó betölthet egy játékot és a meglévő játék betöltése ablak ugrik fel (ld.: 3. kép). Itt a felhasználó betölthet már mentett játékállásokat. A legördülőmenüből (ld.: 8. kép) a helyes fájlt kiválasztva és a „Load Game” gombot megnyomva a játék betölti a mentett állást.

Ha valamiért nem tudja a játék az adott fájlt betölteni, ezt a hibaüzenetet kaphatjuk:



7. kép
Hiba meglévő játék betöltésekor

8. kép
A kiválasztható, betölthető, játékfájlok listája

Mindkét opciónál, ha sikeres a fájlbetöltés, a játék nézetéhez kerülünk (ld.: 4. kép).

A játék nézete és használata

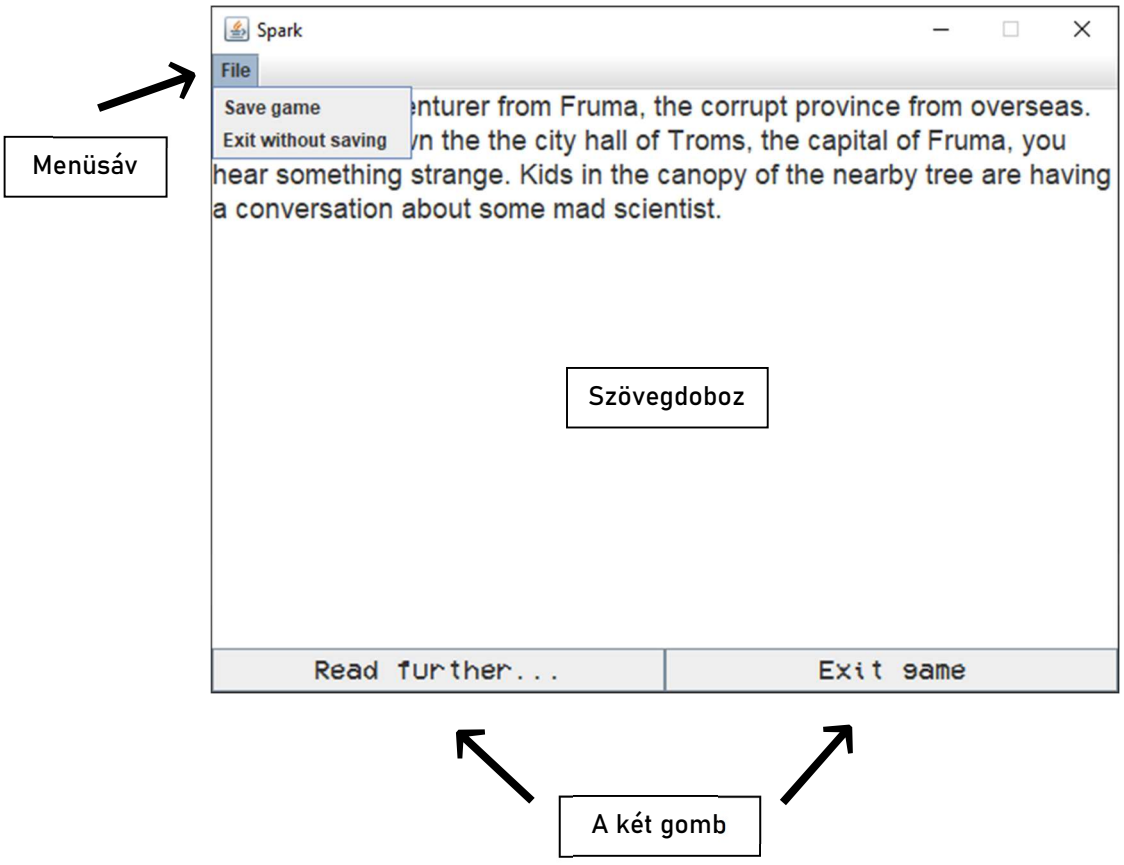
A játéknézet elérését az előző fejezet tárgyalta. Ebben a részben a játéknézetről lesz szó.

A játéknézetben négy főbb elem található. A menüsáv, a szövegdoboz és két gomb (ld.: 9. kép).

A menüsávon, a „File” fülecskére kattintva, a felhasználó el tudja menteni a jelenlegi játékállását („Save Game” címke), illetve ki tud lépni bármikor a játékból, a játék mentése nélkül.

A szövegdobozban a játékos a jelenlegi küldetés szövegét láthatja. Ezen szöveg alapján kell döntéseket hoznia, hogy a játékban tovább léphessen.

A fentebb említett döntéseket az ablak alján szereplő két gomb egyikének kiválasztásával hozhatja a játékos.



9. kép
A játéknézet részeinek bemutatása

Programozói dokumentáció

Az osztályok leírása

Application

Ez a program fő osztálya, innen indul a program. Singleton osztály.

Attribútumai:

-

Metódusai:

+main(args: String[]): void	A szabványos Java belépési pontja a programnak. A függvény létrehoz egy SparkFrame objektumot

Quest

A program küldetésosztálya. Tárolja a küldetés alaptulajdonságait. Absztrakt osztály. Szerializálható osztály.

Attribútumai:

-ID: int	A küldetés sorszáma
-Desc: String	A küldetés leírása
-OptionA: String	Az „A” opció szövege
-OptionB: String	A „B” opció szövege
-JumpA: int	Az „A” opció választásának hatására adott küldetéssorszám
-JumpB: int	A „B” opció választásának hatására adott küldetéssorszám

Metódusai:

+Quest(ID: int, Desc: String, OptionA: String, OptionB: String, JumpA: int, JumpB: int)	Konstruktor
+getID(): int	Visszaadja a küldetés ID attribútumát
+getDesc(): String	Visszaadja a küldetés Desc attribútumát
+getOptionA(): String	Visszaadja a küldetés OptionA attribútumát
+getOptionB(): String	Visszaadja a küldetés OptionB attribútumát
+getJumpA(): int	Visszaadja a küldetés JumpA attribútumát
+getJumpB(): int	Visszaadja a küldetés JumpB attribútumát

BasicQuest

Egy olyan osztály, amely maradéktalanul implementálja a Quest absztrakt osztályt.

Attribútumai:

-

Metódusai:

+BasicQuest(ID: int, Desc: String, OptionA: String, OptionB: String, JumpA: int, JumpB: int)	Konstruktor

RandomQuest

Egy olyan küldetésosztály, melynek véletlenszerű leírása van.

Attribútumok:

-description: List<String>	A lehetséges leírások kollekciója

Metódusok:

+RandomQuest(ID: int, Desc: String, OptionA: String, OptionB: String, JumpA: int, JumpB: int)	Konstruktor
+getDesc(): String	Visszaad egy véletlen leírást a lehetséges leírások közül.

VisitableQuest

Egy olyan küldetésosztály, mely egy alternatív leírást és egy alternatív ugrás-küldetés-sorszámot kínál, ha a küldetés már látogatva volt.

Attribútumok:

-alternatedesc: String	A küldetés alternatív leírása
-visited: boolean	Látogatottsági flag
-alternatejump: int	Az alternatív ugrás-küldetés-sorszám
-rnd: Random	Véletlenszám generáláshoz szükséges objektum
-opts: String[]	Az alternatív ugrás lehetséges szövegei

Metódusok:

+VisitableQuest(ID: int, Desc: String, OptionA: String, OptionB: String, JumpA: int, JumpB: int, alternatedescription: String, alternatejump: int)	Konstruktor
+getOptionA(): String	Ha nem látogatott a küldetés, akkor az eredeti „A” szöveget adja vissza, ha látogatott akkor egy alternatív ugrás szöveget
+getOptionB(): String	Ha nem látogatott a küldetés, akkor az eredeti „B” szöveget adja vissza, ha látogatott akkor egy alternatív ugrás szöveget
+getDesc(): String	Ha nem volt látogatott a küldetés, akkor az eredeti leírást adja vissza, ha látogatott volt, akkor pedig az alternatív leírást
+getJumpA(): int	Ha nem volt látogatott a küldetés, akkor a tényleges „A” ugrás értéket adja vissza és látogatottra állítja a küldetést, ha már látogatott volt, akkor az alternatív ugrás értékét
+getJumpB(): int	Ha nem volt látogatott a küldetés, akkor a tényleges „B” ugrás értéket adja vissza és látogatottra állítja a küldetést, ha már látogatott volt, akkor az alternatív ugrás értékét

QuestQueue

A küldetések tárolásáért és a köztük történő átmenetért felelős osztály. Szerializálható, singleton osztály.

Attribútumok:

-queue: List<Quest>	A küldetéseket tároló heterogén kollekció
-current_state: int	A jelenlegi állapotot tükröző változó, alapértelmezett értéke: 0

Metódusok:

+add(quest: Quest): void	Hozzáadja a paraméterként kapott küldetés objektumot a kollekcióhoz
+sort(): void	Rendezi a kollekciót küldetéssorszám szerinti növekvő sorrendben
+chooseA(): void	Kiválasztja a jelenlegi küldetés „A" ugrását és erre a változóra állítja a jelenlegi állapotot, ha ez az „A" ugrás -1 értékű, kilép
+chooseB(): void	Kiválasztja a jelenlegi küldetés „B" ugrását és erre a változóra állítja a jelenlegi állapotot, ha ez az „B" ugrás -1 értékű, kilép
-at(index: int): Quest	Visszaadja a paraméterként kapott állapotban található küldetés objektumot
+getCurrent(): Quest	Visszaadja a jelenlegi állapotban található küldetés objektumot

GameFile

A játék fájlkezeléséért felelős osztály. Singleton osztály.

Attribútumok:

-queue: QuestQueue	A küldetések QuestQueue típusú objektum

Metódusok:

+getQueue(). QuestQueue	Visszaadja a queue attribútumot
+newGame(name: String): void	Megnyitja a paraméterül kapott fájlt és betölti[1] a küldetéseket a queue tagváltozóba
	IOException kivételt dob, ha nem sikerült megnyitni a fájlt
+loadGame(name: File): void	Beolvassa a paraméterül kapott fájlból a benne szereplő QuestQueue típusú objektumot és betölti a queue tagváltozóba
	IOException kivételt dob, ha nem sikerült megnyitni a fájlt ClassNotFoundException kivételt dob, ha nem QuestQueue típusú objektumot olvas be
+saveGame(name: String): void	Kimenti a paraméterül kapott néven, 'sprkdt' fájlkiterjesztéssel, a játékállást
	IOException kivételt dob, ha nem sikerült kimenteni a játékot

[1] A program nem ellenőrzi a fájl szemantikai helyességét.

SparkJButton

A legtöbbet használt gomb a programban. Szürke háttere, „VCR OSD MONO[2]” betűtípusa, és állítható betűmérete és megjelenő szövege van.

Attribútumok:

-

Metódusok:

+SparkJButton(text:String, fontsize: int)	Konstruktor A paraméterül kapott szövegű és méretű JButton objektumot konstruál.

SparkFrame

A program grafikus megjelenítő főosztálya. Tartalmazza beágyazott osztályként a MainMenu és a GameFrame osztályokat. Singleton osztály.

Attribútumok:

-GAME: GameFile	A játék játékfájlja
-mainMenu: MainMenu	A játék főmenü nézet objektuma
-gameFrame: GameFrame	A játék játéknézet objektuma

Metódusok:

+SparkFrame()	Konstruktor Megpróbálja betölteni a „VCR OSD MONO” betűtípust, ha nem sikerül, akkor halad tovább
-initComponents(): void	Inicializálja a komponenseket a megfelelő beállításokkal Előhívja a főmenü nézet objektumát

MainMenu

A játék főmenü nézetéért felelős osztály. Örököl a szabványos JFrame osztályból. Tartalmazza a newGAME és loadGAME osztályokat beágyazott osztályként.

Attribútumok:

-newGame: JButton	Az új játék kezdése almenü gombja
-loadGame: JButton	A játékbetöltés almenü gombja
-newGAME: newGAME	Az új játék kezdése grafikus felület kezeléséért felelős objektum
-loadGAME: loadGAME	A játékbetöltés grafikus felület kezeléséért felelős objektum

Metódusok:

+MainMenu()	Konstruktor Beállítja az ablak nevét
-initComponents(): void	Inicializálja a komponenseket a megfelelő beállításokkal Elhelyezi és betölti a gombokat, valamint a program logóját a képernyőn Beállítja a megfelelő ActionListener-eket

[2] A program legalább is megpróbálja az adott betűstílust betölteni

newGAME

Az új játék kezdés nézetért felelős grafikus osztály. Örököl a szabványos JFrame osztályból.

Attribútumok:

-filename: JTextField	Szövegmező ahova betöltendő fájl neve írandó
-start: JButton	Az új játék kezdése gomb

Metódusok:

+newGAME()	Konstruktor Beállítja az ablak nevét
-initComponents(): void	Inicializálja a komponenseket a megfelelő beállításokkal Elhelyezi a gombot és a szövegmezőt Beállítja a megfelelő ActionListener-eket

loadGAME

A meglévő játék betöltéséért felelős grafikus osztály. Örököl a szabványos JFrame osztályból.

Attribútumok:

-load: JButton	A meglévő játék betöltése gomb
-options: JComboBox	A lehetséges, betölthető, játékfájlok legördülőmenüje

Metódusok:

+loadGAME()	Konstruktor Beállítja az ablak nevét
-initComponents(): void	Inicializálja a komponenseket a megfelelő beállításokkal Elhelyezi a gombot és a legördülőmenüt, valamint belerakja a legördülőmenübe a lehetséges játékfájlok kollekcióját Beállítja a megfelelő ActionListener-eket

GameFrame

Magáért a játék nézetért felelős grafikus osztály. Örököl a szabványos JFrame osztályból.

Attribútumok:

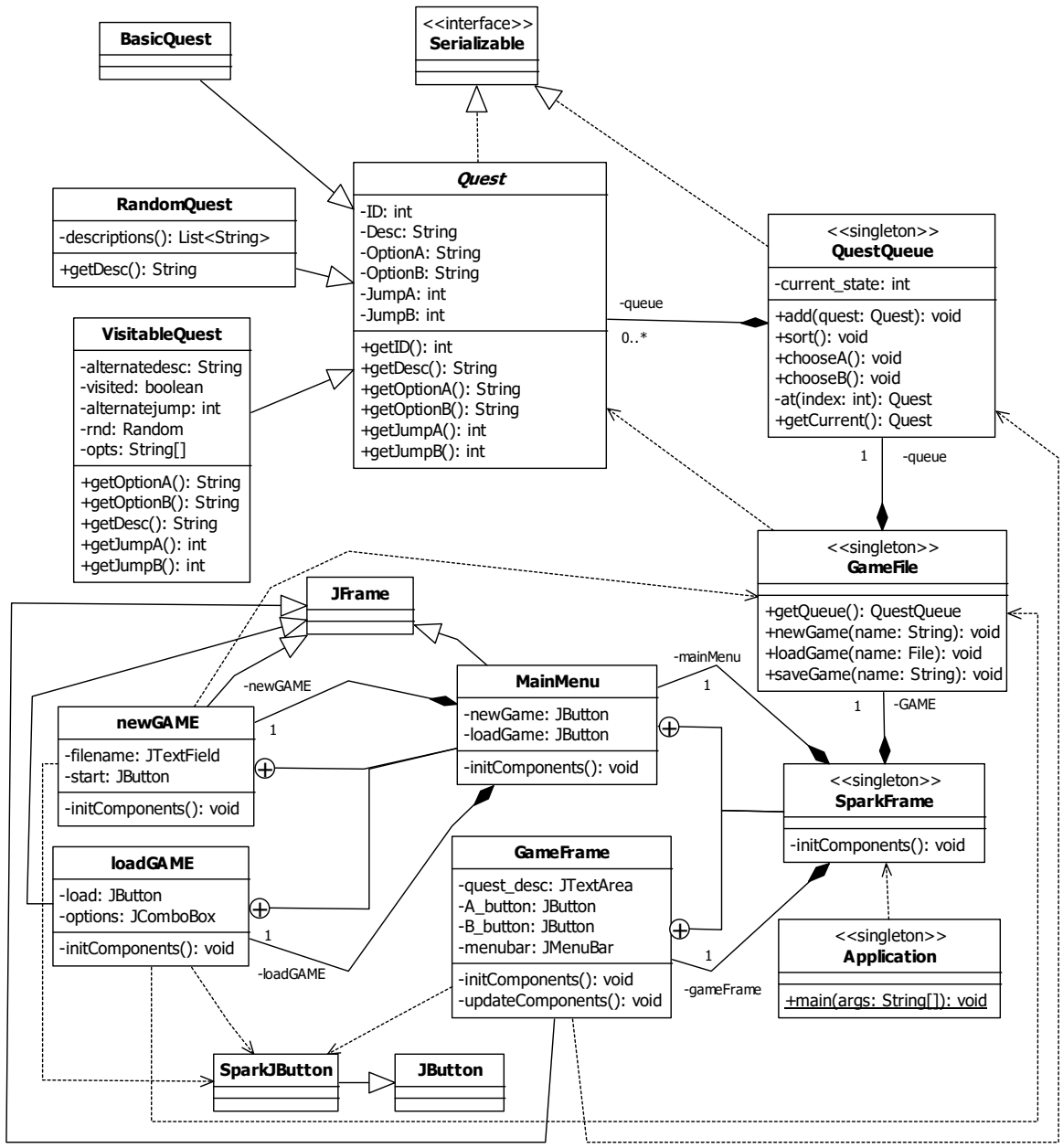
-quest_desc: JTextArea	A küldetésleírás szövegdoboza
-A_button: JButton	Az „A” opció kiválasztásának gombja
-B_button: JButton	A „B” opció kiválasztásának gombja
-menubar: JMenuBar	Az ablak menüszalaga

Metódusok:

+GameFrame()	Konstruktor Beállítja az ablak nevét
-initComponents(): void	Inicializálja a komponenseket a megfelelő beállításokkal Elhelyezi a gombokat, a menüszalagot és a szövegdobozt Létrehozza a menüszalag menüelemeit Beállítja a megfelelő ActionListener-eket
-updateComponents(): void	Frissíti a komponensek beállításait Kitörli az előző ActionListener-eket és újakat vesz fel helyettük Beállítja a szövegmezőben és a gombokon szereplő szöveget

Osztálydiagram

A fentebb felsorolt és kifejtett osztályok osztálydiagramja alább látható:



Tesztelés

A program teszteléséhez a JUnit 4-es verzióját használtam. Az egyes tesztosztályok és felépítésük alább olvasható:

GameFileTest osztály

Attribútumok:

-game: GameFile	A tesztelni kívánt osztály példányosított objektuma

Metódusok:

@Before

+init(): void	Létrehoz egy GameFile objektumot és eltárolja a game attribútumban

@Test

+testNewGameThrow(): void	Teszteli, hogy dob-e az osztály newGame(name: String) függvénye nem létező fájl esetén
+testLoadGameThrow(): void	Teszteli, hogy dob-e az osztály loadGame(name: File) függvénye nem létező fájl esetén
+testSaveGameThrow(): void	Teszteli, hogy dob-e az osztály saveGame(name: String) függvénye érvénytelen paraméter értékkor
+testNewGameNoThrow(): void	Teszteli, hogy nem dob-e az osztály newGame(name: String) függvénye létező fájl esetén
+testLoadGameNoThrow(): void	Teszteli, hogy nem dob-e az osztály loadGame(name: File) függvénye létező fájl esetén

QuestTest osztály

Attribútumok:

-

Metódusok:

@Test

+testAlternateDescription(): void	Teszteli, hogy egy VisitableQuest leírásának értéke koherens-e az osztály eredeti működésével
+testAlternateJump(): void	Teszteli, hogy egy VisitableQuest ugrásainak értéke koherens-e az osztály eredeti működésével

QuestQueueTest osztály

Attribútumok:

-queue: QuestQueue	A tesztelni kívánt osztály példányosított objektuma
-basic: BasicQuest	Egyszerű küldetés objektum
-random: RandomQuest	Random küldetés objektum

Metódusok:

@Before

+init(): void	Létrehoz egy QuestQueue objektumot és eltárolja a queue attribútumban Létrehoz egy egyszerű és egy random példaküldetést

@Test

+testQueueAdd(): void	Teszteli, hogy, ha hozzáadunk a QuestQueue-hoz egy objektumot, szerepel-e benne
+testChoose(): void	Teszteli, hogy jól működik-e az állapotátmenet a QuestQueue-ban
+testSort (): void	Teszteli, hogy jól rendez-e a QuestQueue

Fájlformátum

A program helyes működéséhez a szükséges a megfelelő fájlformátum használata, valamint szükségesek még egyéb fájlok.

A fájlformátum

A program helyes működéséhez elengedhetetlen a jó fájlformátum. A program nem ellenőrzi a fájlok szemantikai helyességét. A program ugyan nem követeli meg, hogy a küldetések azonosítójuk szerint rendezettek legyenek, de a fájl írójának az alábbi alapvető előírásoknak meg kell felelnie:

- Minden küldetés attribútumot **szigorúan** pontosvesszővel kell elválasztani
- A küldetések szövegeiben **nem** szerepelhet pontosvessző
- Minden küldetésnek **külön sorban** kell elhelyezkednie és minden küldetés sorának **teljesnek** kell lennie (nem lehet félbehagyott küldetésdefiníció)
- A küldetések azonosítójának számozása 0-tól indul és 2 147 483 647-ig tart

Ezen szabályok betartásával lesz csak elvárt a működés. Fontos még megjegyezni azt, hogy a 0 azonosítóval rendelkező küldetés az első küldetés, amit a program megjelenít és, hogy a -1-es küldetésazonosító a játék valamelyik kilépési pontja, viszont természetesen -1 azonosítóval rendelkező küldetés nem létezhet a játék fájlban.

Küldetések formátuma

A küldetések első két attribútuma megegyezik. Az első szám a küldetés azonosítója, a második karakter a küldetés típusa, ami lehet:

- Basic – „B” vagy „b” karakter
- Random – „R” vagy „r” karakter
- Visitable – „V” vagy „v” karakter

Basic küldetés

Az egyszerű vagy „basic” küldetés formátuma itt látható:

```
0;B;Lorem ipsum. Dolor sit amet?;Ack;1;Nak;2
```

Az első két attribútum leírása már szerepelt fentebb. A harmadik attribútum egy karakterlánc, mely a küldetés leírása. A negyedik az „A” opció leírása és sorban utána az ötödik az „A” opció választása esetén történő ugrás küldetésazonosítója[3]. Az utolsó két attribútum A „B” opció leírása és a „B” opció választása esetén történő ugrás küldetésazonosítója.

[3] Ezt úgy kell elképzelni, hogy, ha a kiválasztjuk az „A” opciót és az ugrásazonosító például 3, akkor a következő megjelenített küldetés a 3-as azonosítójú lesz.

Random küldetés

A véletlenleírású vagy „random” küldetés formátuma itt látható:

```
0;R;Random thing#Other random thing#...#Last random thing;Ack;1;Nak;2
```

Az attribútumok azonosak az egyszerű küldetés attribútumaival, itt annyi a különbség, hogy a harmadik attribútum, küldetés leírása attribútum, szövege elválasztható „#” karakterekkel, melyek mentén a program eldarabolja a szöveget, és a program véletlenszerűen sorsol egy leírást az eldarabolt szövegből.

Visitable küldetés

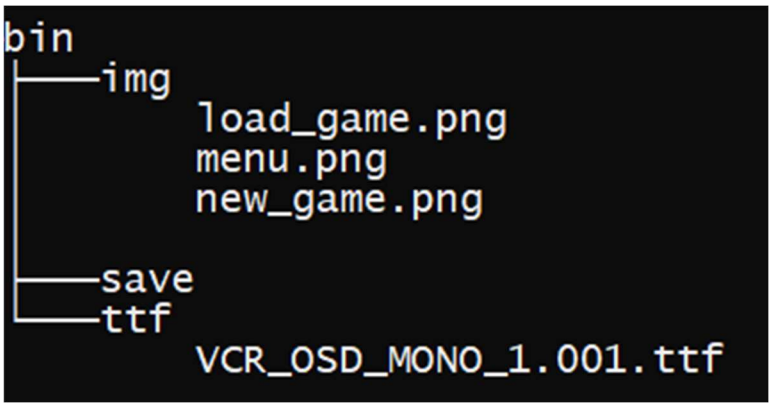
A látogatható vagy „visitable” küldetés formátuma itt látható:

```
0;V;Lorem ipsum. Dolor sit amet?;Ack;1;Nak;2;I was here.;1
```

Az első hét attribútum megegyezik az egyszerű küldetés attribútumaival. A nyolcadik attribútum akkor íródik ki, ha a küldetést már látogatta a felhasználó. Ennek példájára, ha már látogatva volt a küldetés a kilencedik attribútum az az ugrásazonosító, mely mindegyik válaszlehetőségre megtörténik.

A szükséges fájlok és struktúrájuk

A program esztétikus kinézetéhez különböző képfájlokra és egy betűtípusra van szükség. A fájlokat és a könyvtárszerkezetet az alábbi kép (10. kép) mutatja be:



10. kép
A fájlstruktúra

A szükséges fájlok helyes szerkezettel és a program forráskódja itt található. Az esetleges fájlhiány vagy hibás könyvtárstruktúra nem okoz gondot a program működésében. Az egyetlen probléma, amit maga után von, hogy nem lesz teljesen esztétikus a kezelőfelület.

Fordítás és futtatás

A program fordításához a JDK 18-as verziója szükséges. A programot az IntelliJ IDEA fejlesztőkörnyezet alapértelmezett fordítójával kell fordítani. Ha ezek a követelmények teljesülnek, akkor a program az elvárt módon működik.