Programozási technológia   
3. Beadandó

Név: Magyar Viktor

Neptun kód: O9WEJJ

Dátum: 2024.05.10.

Feladat száma: 3

[1. Feladat leírása: 3](#_Toc162907818)

[2. Megoldási terv: 4](#_Toc162907819)

[2.1 Megoldás leírása 4](#_Toc162907820)

[2.2 Model komponens 4](#_Toc162907821)

[2.2.1 tictactoe.table package: 4](#_Toc162907822)

[2.2.2 tictactoe.gamecontrol package: 5](#_Toc162907823)

[2.2.3 tictactoe.gamemenu package: 6](#_Toc162907824)

[2.2.4 tictactoe.ObjectCompositionUtils 6](#_Toc162907825)

[2.4 Controller komponens 7](#_Toc162907826)

[2.4.1 tictactoe.applicationstates package 7](#_Toc162907827)

[2.4.2 tictactoe.exceptions: 7](#_Toc162907828)

[2.5 View komponens 8](#_Toc162907829)

[2.5.1 tictactoe.views package osztályai: 8](#_Toc162907830)

[2.5.2 tictactoe.views.mainmenu package: 9](#_Toc162907831)

[2.5.3 tictactoe.views.initializegameview: 10](#_Toc162907832)

[2.5.4 tictactoe.views.tictactoeview: 11](#_Toc162907833)

[Esemény-eseménykezelő párok 12](#_Toc162907834)

[Tesztelési Terv 12](#_Toc162907835)

# 1. Feladat leírása:

Készítsünk programot, amellyel egy labirintusból való kijutást játszhatunk. A játékos a labirintus bal alsó sarkában kezd, és a feladata, hogy minél előbb eljusson a jobb felső sarokba úgy, hogy négy irányba (balra, jobbra, fel, vagy le) mozoghat, és elkerüli a labirintus sárkányát. Minden labirintusban van több kijutási útvonal. A sárkány egy véletlenszerű kezdőpozícióból indulva folyamatosan bolyong a pályán úgy, hogy elindul valamilyen irányba, és ha falnak ütközik, akkor elfordul egy véletlenszerűen kiválasztott másik irányba. Ha a sárkány a játékossal szomszédos területre jut, akkor a játékos meghal. Mivel azonban a labirintusban sötét van, a játékos mindig csak 3 sugarú körben látja a labirintus felépítését, távolabb nem. Tartsuk számon, hogy a játékos mennyi labirintuson keresztül jutott túl és amennyiben elveszti az életét, mentsük el az adatbázisba az eredményét. Egy menüpontban legyen lehetőségünk a 10 legjobb eredménnyel rendelkező játékost megtekinteni, az elért pontszámukkal, továbbá lehessen bármikor új játékot indítani egy másik menüből. Ügyeljünk arra, hogy a játékos, vagy a sárkány ne falon kezdjenek.

# 2. Megoldási terv

## 2.1 Megoldás leírása

A feladat megoldását az Model-View architektúra alapján készítettem el. Ami azt jelenti, hogy a program szétválasztható két komponensre, ami a Model, és a View. Részletesen bemutatom a következő fejezetekben, hogy mely package-ek, illetve osztályok teszik ki a két komponenst.

## 2.2 Model komponens

### 2.2.1 labirinth.model.gamecontrol package

A képen szöveg, képernyőkép, szám, sor látható

Automatikusan generált leírás

A model komponens table csomagja tartalmazza a pálya logikáját. A Table és a Column osztály egy AbstractList generikus osztályból leszármazó osztály. Azért választottam ezt a lista fajtát, mert így a pálya nem módosítható, mivel csak a get és a size metódusait írtam felül a két osztályban. A Field osztály egy mezőt reprezentál és tartalmazza azt, hogy az üres-e, X, illetve O jel van-e rakva. Ennek van gettere, illetve settere is. A FieldChangeListener osztály egy eseményt reprezentál, amire fel lehet iratkozni. A feliratkozott példányok onSignalChanged metódusa, akkor hívódik meg, amikor a Field.setSignal metódus meghívódik.

2.2.2 labirinth.model.map package A képen szöveg, Párhuzamos, szám, dokumentum látható

Automatikusan generált leírás

Ez a játékmenet levezényléséért felelő csomag. A játékosok „utasításait” command pattern segítségével implementáltam. A Player osztály felelős a játékos adatainak lekéréséért és az utasítások elkészítéséért. A TableObserver osztály implementálja az ITableObserver interfészt. Ez az osztály ellenőrzi a játékmenet tábláját a játék közben, illetve állítja a MatchResult attribútumának értékét, amit le tudnak kérni kívűlről. A Match osztály pedig tartalmazza a táblát, játékosokat, a egy TableObserver példányt, illetve ezek segítségével vezényli a játékot. Két féle eseményfigyelő interfész van ebben a csomagban. Egy, ami a játékosok állapotváltozását figyeli, illetve egy, ami a játékmenet változásait figyeli.

### 2.2.3 labirinth.model.utilities package

A képen szöveg, nyugta, diagram látható

Automatikusan generált leírás

A gamemenu csomag a különböző menüelemek logikáját valósítja meg. Az IMenuOption interfész doOption metódus implementálása adja meg, hogy az adott menü elem az mit fog csinálni, amikor megnyomják vagy kiválasztják, illetve a getTitle metódus, pedig a View által megjelenítendő szöveggel tér vissza.

### 2.2.4 labirinth.model.entities package

A képen szöveg, nyugta, diagram, Párhuzamos látható

Automatikusan generált leírás

Ez az osztály különböző Singleton objektumokat tartalmaz. Tartalmazza a különböző View-okat, illetve az alapértelmezett IMatchFactory-t implementáló osztály egy példányát.

### 2.2.5 labirinth.model.gamestates package

A képen szöveg, képernyőkép, sor, diagram látható

Automatikusan generált leírás

### 2.2.6 labirinth.model.resources package

A képen szöveg, nyugta, Betűtípus, sor látható

Automatikusan generált leírás

## 2.3 View komponens

### 2.5.1 labirinth.view package

A képen szöveg, Betűtípus, nyugta, sor látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus, nyugta látható

Automatikusan generált leírás

### 2.5.2 labirinth.view.scorelist package

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

A MainWindow osztály a JFrame osztályból származik és innen indul el az alkalmazás. Ez egy Singleton osztály. A main metódusban a MainWindow.getInstance metódusát hívjuk csak meg. A megfelelő View osztályokat egy CardLayout-ba töltjük be a konstruktorban, illetve az onViewChanged metódus segítségével válthatunk a nézet-ek között. Ezt a metódust az ApplicationState.onStateInitialize metódusa hívja meg.

A MenuButton általánosan a menü gombokat reprezentáló osztály, ami egy IMenuOption-t kap meg és a doOption metódusát hívja meg a gomb megnyomásakor.

A View osztály egy generikus osztály, ami a különböző nézetek ősosztálya.

### 2.5.3 labirinth.view.mainmenu package

A képen szöveg, képernyőkép, nyugta, Betűtípus látható

Automatikusan generált leírás

A MainMenuView a View<InMainMenu> osztályából származik le, így az initialize egy InMainMenu példányt kap. Ez alapján lekéri tőle az IMenu attribútumát, majd MenuButton gombokat helyez el egymás alá középre.

### 2.5.4 labirinth.view.startgamemenu package

A képen szöveg, képernyőkép, sor, szám látható

Automatikusan generált leírás

A InitializeGameView a View<InInitializeGameMenu> osztályából származik le, így az initialize egy InInitializeGameMenu példányt kap. Ez alapján inicializálja a különböző vezérlőket, illetve a JTextFieldListener osztály segítségével a JTextField-ekhez hozzárendeli az InInitializeGameMenu különböző settereit. Tehát, amikor a text field-ek értéke változik, akkor a controller példány értéke is változik, ami kivételt dob, ha valamilyen input nem megfelelő. Ekkor piros keretet kap a text field.

### 2.5.5 labirinth.view.game package

A képen szöveg, nyugta, Párhuzamos, diagram látható

Automatikusan generált leírás

Ez a csomag tartalmazza a játékmenet nézetéhez szükséges osztályokat. A FieldButton reprezentálja a mezőket, ami egy field-et kap a konstruktorában és feliratkozik a FieldChangeListener.onSignalChanged eseményére. A signalMap attribútum segítségével változtatja át String típusúra az adott Signal objektumot és azt állítja be a saját szövegének, amikor változik a Signal attribútum. A PlayerPanel a játékos releváns adatait megjelenítő panel, a ThrowButton, pedig a pálya feletti gombokat reprezentálja, amit megnyomva a játékos ledobja az adott oszlopba a saját jelét. A TicTacToeGameView ezen osztályok segítségével készíti el a játék nézetét.

### 2.5.6 labirinth.view.gameover package

A képen szöveg, nyugta, képernyőkép, sor látható

Automatikusan generált leírás

# 3. Esemény-eseménykezelő párok

|  |  |
| --- | --- |
| **Esemény** | **Eseménykezelő** |
| A játékos az egyik oszlopban elhelyezi a saját jelét | ThrowListener |
| A játékos megnyom egy menü gombot | MenuButton |
| A játékos beír valamit az egyik szövegmezőbe | JTextFieldListener |
| Egy mező jele megváltozik a pályán | FieldChangeListener |
| Játékosok váltakozása a nézetben játékmenet közben | PlayerListener |
| Játék vége | MatchListener |
| A pályán egy oszlop betelik | MatchListener |
| A pályát alaphelyzetbe állítják | MatchListener |

# 4. Pályát generáló algoritmus leírása

# 5. Tesztelési Terv

|  |  |
| --- | --- |
| **Név** | **Lépések** |
| Játék kezdés teszt | * Indítsd el a programot * Nyomj a Start Game gombra * Nyomj a Start gombra   Elvárt működés:  A játékmenet elindul a player1 nevezetű játékos zöld színű, tehát ő a kezdő játékos, a tábla üres, illetve a tábla felett lévő gombok zöld kerettel rendelkeznek és megnyomhatóak. Bal felső saroknál két menüelem látszódik. |
| Játék vége teszt1 | * Indítsd el a programot * Nyomj a Start Game gombra * Nyomj a Start gombra * Nyomd meg az 1. oszlop gombját * Nyomd meg a 2. oszlop gombját * Nyomd meg a 2. oszlop gombját * Nyomd meg a 3. oszlop gombját * Nyomd meg a 3. oszlop gombját * Nyomd meg az 1. oszlop gombját * Nyomd meg a 3. oszlop gombját * Nyomd meg a 4. oszlop gombját * Nyomd meg a 4. oszlop gombját * Nyomd meg a 4. oszlop gombját * Nyomd meg a 4. oszlop gombját   Elvárt működés:  A lépések sikeresek és az utolsó lépés után jelzi a program a felhasználónak, hogy a player1 nevű játékos nyert 4 átlósan kirakott jellel. |
| Játék vége teszt2 | * Indítsd el a programot * Nyomj a Start Game gombra * Nyomj a Start gombra * Nyomd meg az 1. oszlop gombját * Nyomd meg a 1. oszlop gombját * Nyomd meg a 2. oszlop gombját * Nyomd meg a 2. oszlop gombját * Nyomd meg a 3. oszlop gombját * Nyomd meg az 3. oszlop gombját * Nyomd meg a 1. oszlop gombját * Nyomd meg a 4. oszlop gombját * Nyomd meg a 2. oszlop gombját * Nyomd meg a 4. oszlop gombját   Elvárt működés:  A lépések sikeresek és az utolsó lépés után jelzi a program a felhasználónak, hogy a player2 nevű játékos nyert 4 darab sorban kirakott jellel. |
| Játék vége teszt3 | * Indítsd el a programot * Nyomj a Start Game gombra * Nyomj a Start gombra * Nyomd meg az 1. oszlop gombját * Nyomd meg a 2. oszlop gombját * Nyomd meg a 1. oszlop gombját * Nyomd meg a 2. oszlop gombját * Nyomd meg a 1. oszlop gombját * Nyomd meg az 2. oszlop gombját * Nyomd meg a 1. oszlop gombját * Nyomd meg a 2. oszlop gombját * Nyomd meg a 3. oszlop gombját * Nyomd meg a 4. oszlop gombját * Nyomd meg a 3. oszlop gombját * Nyomd meg a 4. oszlop gombját * Nyomd meg a 3. oszlop gombját * Nyomd meg a 4. oszlop gombját * Nyomd meg a 3. oszlop gombját * Nyomd meg a 4. oszlop gombját   Elvárt működés:  A lépések sikeresek és az utolsó lépés után jelzi a program a felhasználónak, hogy a meccs döntetlen és látszódik, hogy a tábla teljesen betelt az X és az O jelekkel. |
| Kilépés a játékból teszt | * Indítsd el a programot * Nyomj az Exit Game gombra   Elvárt működés:  A program bezáródik, amikor megnyomják az Exit Game gombot. |
| Vissza a menübe teszt | * Indítsd el a programot * Nyomj a Start Game gombra * Nyomj a Start gombra * Nyomj a Menu gombra   Elvárt működés:  A játékmenetben a Menu gombra kattintva visszatér a program a főmenübe. |
| Vissza teszt | * Indítsd el a programot * Nyomj a Start Game gombra * Nyomj a Back gombra   Elvárt működés:  A meccs adatainak megadásakor a Back gombra kattintva visszatér a program a főmenübe. |
| Új játék kezdése teszt | * Indítsd el a programot * Nyomj a Start Game gombra * Nyomj a Start gombra * Nyomj a New Game gombra * Írd be a Column Number X Row Number mezőbe a 8x5 értéket * Nyomj a Start gombra   Elvárt működés:  Az első start gombra nyomáskor egy 4x4-es tábla jelenik meg, a második start gombra nyomáskor, pedig egy 8x5-ös tábla jelenik meg. |