NagyHF Dokumentáció – Ulrich Flóra, M62ZXD

Turmeszkek

A programom egy sejtautomata, amely áll egy 2D-s rácsból plusz ezen egy hangyából. A hangya a saját aktuális állapota a sejt aktuális állapota alapján dönt arról, hogy a neki előzetesen megadott szabályok közül melyiket alkalmazza. Az src package a következő osztályokból áll:

- ***Cell*** : A sejteket reprezentáló osztály, amely segítségével létrehozom a szimuláció alapját képező táblát. Egy sejt állapotát tárolja továbbá a konstruktora az alapértelmezett színt fehérre állítja.
- ***Main***: A Main osztály, innen indul a felhasználói felület.
- ***Turmite***: A Hangyát/Termeszt reprezentáló osztály, amely tárolja a hanyga irányát, állapotát és a rá vonatkozó szabályokat.
- ***TurmiteFrame***: A felhasználói felülethez tartozó osztály, amelyet a JFrame-ből származtattam.
- ***TurmiteTable***: A szimuláció táblája, amelyet a JFrame-ből származtattam.

A Turmite osztály

A Hangyát/Termeszt reprezentáló osztály, amely tárolja a hangya irányát, állapotát és a rá vonatkozó szabályokat. A konstruktorban a szabályok kétdimenziós tömb első sora is inicializálódik hisz ezeket a felhasználó nem változtathatja meg. Az első sor tartalmazza ugyanis az összes lehetőséget, amely előfordulhat abból a szempontból, hogy a hangya és a sejt, amin áll milyen állapotban vannak. (00 = 0, 01 = 1, 10 = 2, 11 = 3). Így minden oszlop egy különböző szabályt jelent : (állapot-cellaérték)-irány-újérték-újállapot, sorrendben ahogy az a specifikációban is adott. A hangya iránya a következőképpen van tárolva: 0 = fel, 1 = le, 2 = jobb, 3 = bal.

A TurmiteFrame osztály

- *TurmiteFrame():* A GUI konstruktora

- *static Turmite regiTurmInit():* Ha a felhasználó az egyel korábbi hangyát szeretné újra szimulálni akkor hívódik meg ez a függvény. Egy txt fájlból olvassa ki az előző hangya adatait, azonban a kapott mátrixot transzponálni kell. Visszatérési értéke: Turmite , a kapott hangya. Throws: FileNotFoundException

- *void saveTurmToFile(Turmite turmite):* Ahhoz, hogy lehessen régebbi hangyát használni el kell menteni az adott hangya szabályait. A paraméterként kapott hangya szabályait írja ki egy txt fájlba, az első sort kivéve hisz azok minden hangyában ugyanazok, inkább jelölésre szolgálnak. Paraméterek: turmite, a mentendő hangya. Throws: IOException

- *void setDelay(int delay) :* Átállítja a szimuláció sebességét.

- *Turmite turminit():*Inicializálja a hangyát a felhasználó által megadott paraméterek szerint, amelyeket a comboboxok és a textfieldek segítségével állít be. A hangya szabályit tartalmazó mátrixot tölti fel, állapotgépek segítségével. A felhasználói felületen a 4 sor jelenti a "rules" tömb 4 oszlopát, természetesen a megfelelő sorrendben.

Így először az irányra/fordulásra vonatkozó sort tölti fel. Majd az új állapota vonatkozót, végül pedig a sejt új állapotára vonatkozót. Továbbá elmenti a kapott szabályokat a saveTurmToFile függvény segítségével, amely ahhoz kell majd, hogy képes legyen a program régebbi hangya visszatöltésére. Visszatérési értéke: Turmite, a kapott hangya. Throws: IOException.

- *void tableCloser():* A TurmiteTable frameket bezáró függvény, mivel azokat a felhasználónak nem kell bezárnia. Hisz a gombok (start, old) megnyomásával kell ennek megtörténnie.

A TurmiteTable osztály

- *public TurmiteTable():* A tábla konstruktora, amely egy CellTable-t is létrehoz. Throws: InterruptedException

- *public Cell getCell(int x, int y):* Visszatér a megadott koordináták helyén szereplő sejttel. A "pos" vagyis a sejteket tartalmazó tömbben a keresett sejt indexe, ezt abból adódóan kell a "y * cols + x" képlettel kiszámolni, hogy a sejtek egy egydimenziós tömbben vannak eltárolva. Paraméterek: x - , a táblán az x azaz vízszintes koordináta ,y - , a táblán az y azaz függőleges koordináta. Visszatérési érték: Cell, a keresett sejt.

- *public void step():* Eldönti, hogy melyik szabályt használja a program a hanyga 4 lehetséges szabályai közül majd meghívja az ahhoz a konkrét szabályhoz tartozó függvényt. Ezt a sejt állapota és a hangya állapota alapján teszi, mivel ez egy sejtautomata.

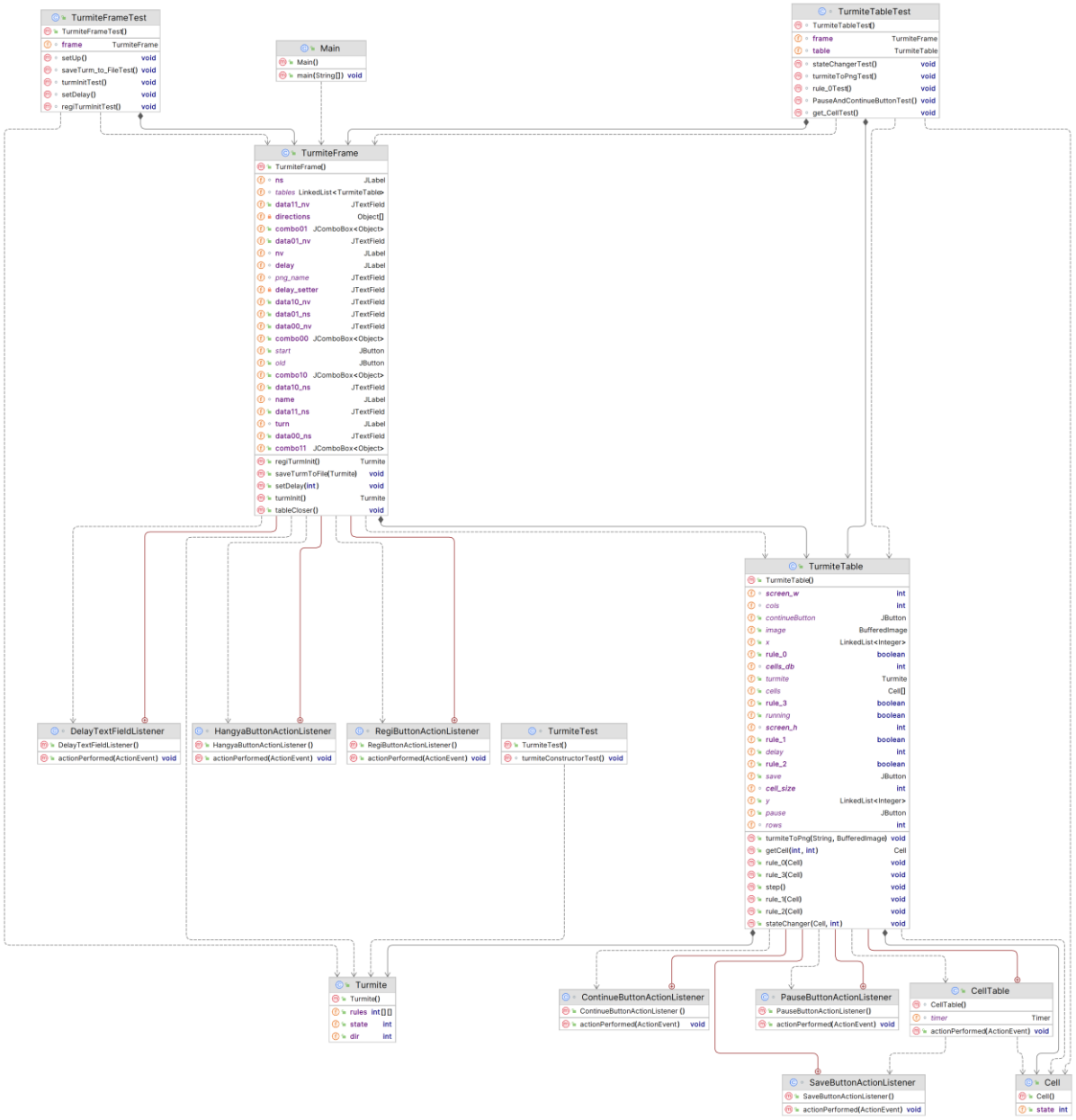
- *public void rule_0(Cell c), public void rule_1(Cell c) , public void rule_2(Cell c), public void rule_3(Cell c):* A 4 szabályhoz tartozó 4, működési elvében megegyező, azonban viselkedésben nyilvánvalóan különböző függvény. A lépéseket természetesen a hangyában tárolt szabályok tömb alapján végzik az állapotgépek. A lépések sorrendje a következő:
  - sejt állapotának megváltoztatása
  - hangya saját állapotának megváltoztatása
  - fordulás
  - lépés

  Paraméterek: c , az aktuális sejt, amelyen a hangya áll.

- *public void stateChanger(Cell c, int col):* A hangya és a sejt állapotát megváltoztató függvény. Paraméterek: c , az aktuális sejt, amelyen a hangya áll. col , a szabály oszlopa, amelyet alkalmazni kell.

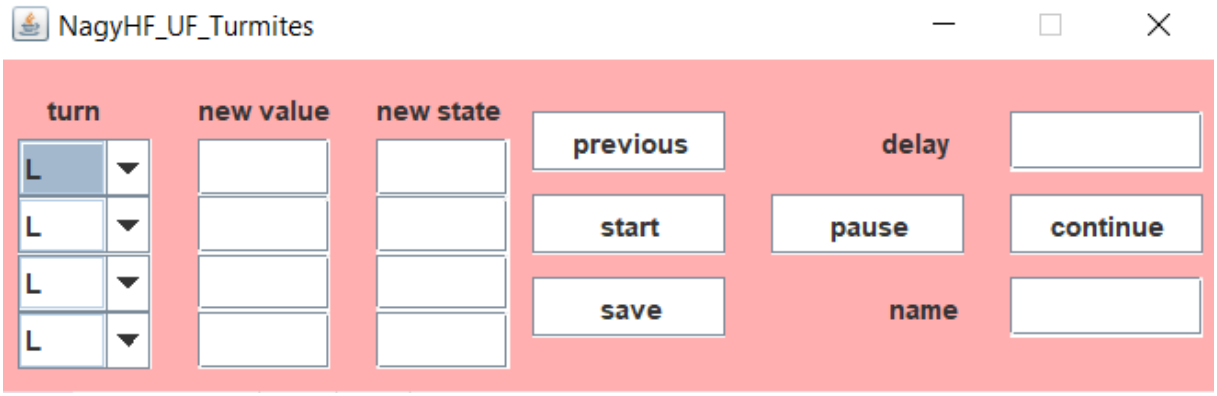
- *public void turmiteToPng(String name):* A szimulációt egy png fájlként elmentő függvény. A GUI-ban meg kell adni a kép nevét is. A képet a "pictures" mappába menti, amely a házit is tartalmazó könyvtárban található. Paraméterek: name , a felhasználó által választott képnév. Throws: IOException.

A projekt osztálydiagrammja



Felhasználói kézikönyv

A program indításakor a következő ablak jelenik meg:



Ezen az ablakon állíthatóak be a szimuláció paraméterei. A **turn** oszlopban a hangya fordulási iránya állítható be következő jelentésekkel: L[eft], R[ight], N[o turn], U[-turn]. A **new value** oszlopban a sejt új értéke állítható be, ha 0-t írunk akkor halott a sejt, ha 1-est akkor élő. Más ide nem írható be. A **new state** oszlop ehhez hasonló itt a hangya új állapota állítható.

Minden sor egy aktuális állapotot jelent, amiben a hangya a szimuláció során tartózkodik. Így az első sorban beállított értékek akkor lesznek felhasználva, ha a hangya állapota 0 és, a sejt értéke is 0. a második sorhoz a 0-1, a harmadikhoz az 1-0, az utolsóhoz pedig az 1-1 páros tartozik.

A felület jobb oldalán található több gomb és két szöveges bemeneti mező.

- ***previous***: Az előző hangya paramétereivel indítja el a szimulációt.
- ***delay felirat melletti mező***: A szimuláció sebessége állítható itt. Egy egész számot vár a program, minél nagyobb annál több idő fog eltelni két lépés között. A mezőt nem kötelező kitölteni.
- ***start***: A hangya paraméterinek beállítása után ezzel a gombbal indítható el a szimuláció.
- ***pause***: a szimulációt szüneteltető gomb
- ***continue***: a szimuláció folytatását teszi lehetővé
- ***save***: a szimuláció által generált táblát menti le egy png fájlba, amely nevét is ki lehet választani. Ezt a gombot célszerű a pause gomb használata után, azaz álló szimuláció során használni.
- ***name utáni mező***: itt adható meg a kép neve, amelyet a pictures mappába ment a program.