Házi feladat

Turmesz

1. Specifikáció

A „turmesz” egy 2D-Turing gép. Egy 2D rács van, amelynek minden cellája 1 vagy 0 állapotú. Ezen kívül van egy hangya (a „termesz”) ami egy cellán áll, vagyis van egy aktuális pozíciója és iránya, és minden lépésben végrehajt egy műveletet.

A hangyának meg kell adni egy programot, hogy az aktuális állapotának és a cella tartalmának (amin áll) megfelelően mit tegyen.

Egy lépés így néz ki:

1. Fordul 90 fok valamilyen sokszorosával → L[eft], R[ight], N[o turn], U[-turn]
2. Aktuális cellába beír valamit (0 vagy 1)
3. Előre lép egyet
4. Átmegy egy másik állapotba

A program specifikációját **[állapot-cellaérték-irány-új érték-új állapot]** formában kell megadni, erre egy példa:

0-0-L-1-0
0-1-U-0-1
1-0-R-0-0
1-1-N-1-1

A szoftver ellenőrizni fogja, hogy csak teljesen specifikált programot lehessen futtatni, vagyis minden lehetséges **újérték-újállapot párnak** legyen megfelelő **állapot-cellaérték specifikáció.**

A példa magyarázata:

Ha a hangya 0-ás állapotban van és a cellának amire lép az értéke 0 akkor 90 fokot balra fordul és a cella értékét 1-re változtatja, majd felveszi a 0-s állapotot és előre lép egyet.
Ha a hangya 0-s állapotban van és olyan cellára lép, aminek az értéke 1 akkor 180 fokkal elfordul, a cella értékét 0-ra változtatja és felveszi az 1-es állapotot. És így tovább.

Magába a programba be lehet majd tölteni a 2D rács elmentett állapotát és a hangya programját is, vagy kézzel lehet állítani a rács egyes celláinak állapotát és begépelni a hangya programját.

A szimulációt (lépések kezdete) el lehet indítani, bármikor leállítani és újraindítani, a rácsok tartalmát törölni és a szimuláció sebességét állítani.

Várható implementációs döntések:

(Lehet, hogy ezektől később el kell térnem, de egyelőre így képzelem el)

- A program beolvassa a felhasználó által begépelt hangya programot, azt betölti egy kétdimenziós String tömbbe, tokenizálja kötőjelek szerint, ellenőrzi, hogy teljesen specifikált-e, ha igen a Hangya objektumnak beállítja a tulajdonságait, ha nem, kiírja a hibát a felhasználónak.
- Maga a rács egy kétdimenziós Integer tömb lesz, így könnyen megadható a pozíció és tárolható az érték
- A rács módosításához lesz egy „Modify” gomb amire kattintva a rács szerkeszthető. Egy cellára kattintással annak az ellenkezőjére változik az értéke.
- A szimuláció sebessége balra és jobbra nyilakkal lesz állítható.
- A program ablaka két részre lesz osztva az egyikben látható a hangya programja, a másikban a rács.
- A hangya programja csak akkor szerkeszthető, ha a szimuláció éppen áll.
- A hangya programját a szimuláció minden indításakor újra betölti a hangyába.

2. Felhasználói leírás

A program ablaka két részre oszlik, jobb oldalon van a rács, amin a hangya mozog, alapesetben a hangya a rács közepéből indul. A bal oldalon van egy szövegdoboz, amibe a hangya programját kell írni [állapot-cellaérték-irány-új érték-új állapot] formában, egy parancsot egy sorba. Alatta egy sor piros szöveg van, ami visszajelzést ad a programtól. Ide írja, ha a beírt parancsok szintaxisa helytelen és ha mentünk, mi a mentett fájl neve.

Alatta van egy Save és egy Load gomb, a Save-el automatikusan választ fájl nevet év-hónap-nap_óra-perc-mp.ser formátumban amt ezután kiír. A Load-ra kattintva felugró ablakból lehet kiválasztani a betöltendő fájlt.

Alatta a „<<” gombbal lassítani „>>” gyorsítani lehet a hangyát. A Start gombbal elindul a hangya a rácson a beírt parancsokkal. A Stop gombbal megáll a hangya és a Restart gombbal törlődnek a rács változtatásai és visszaáll alapállapotba.

Ha a rácson valamelyik négyzetrácsra kattintunk az állapota megváltozik.

A zip fájlban van 2 tesztfájl, testfile2000.ser és testfile3000.ser.

3. Implementáció

Main.java osztály

Létrehozza az alkalmazást

public static void main(String[] args)

App osztály

Ősosztálya a JFrame, létrehozza az alkalmazás ablakát, elhelyezi az elemeit, kezeli az időzítőt

public App(String s) – létrehozza a JFrame-t és beállítja a címének az adott Stringet és elrendezi benne a komponenseket. A komponensekhez hozzáadja a MouseListenerjeiket.

public void startTimer(int t, App a) – létrehoz egy Timert és elindítja az adott int-el mint periódus

public void stopTimer() – megállítja az időzítőt

public void setUpGrid(int num) – előkészíti a rácsot, a rácsok számát beállítja az átadott intre

Grid osztály

Ősosztálya a JPanel, létrehozza a rácsot, kirajzolja amikor szükséges, beállítja a hangyát hozzá.

public Grid(int num) – létrehozza a rácsot az adott rácsszámmal

public void setUp() – az rács alapterülete alapján előkészíti a rácsok számát egy sorban, oszlopban, alapállapotba állítja

public void paintComponent(Graphics g) - alap JComponent függvény overrideja, az adott változók alapján kirajzolja a rácsot, bele a hangyát

public Ant getAnt() – Visszaadja a rácshoz tartozó hangyát

Ant osztály

Ősosztálya a Block, implementálja a Serializable-t, feldolgozza és tárolja a hangya programját, lépteti a hangyát.

public Ant() – létrehozza a hangyát, az állapotát és az irányát 0-ra állítja

public void setXY(int x, int y) – beállítja a hangya x és y pozícióját

public Command[] getCmnd() – visszaadja a hangya parancsait

public void setCmnd(Command[] c) – beállítja a parancsokat

public void setRect(Rectangle rec) – beállítja a hangya téglalapját

public void setGrid(Grid g) – beállítja a hangya melyik rácshoz tartozik

public void setBlock(Block b) – beállítja hangya éppen melyik négyzetrácson áll

public int programmeAnt(String cmd) – feldolgozza és betölti a kapott stringet az hangya Command változójába, ha 0 értéket ad vissza nem volt mit beolvasnia, ha 1-et akkor a program szintaktikailag helyes, ha ekkor sem tért még vissza akkor, 2-t ad, vagyis be tudta tölteni a programot, de valami szintaktikailag hibás volt.

public boolean checkProg() – visszaadja hogy a beolvasott program szintaktikailag helyes-e.

public void step() – lépteti eggyel a hangyát a Commandjai szerint

public int getX() – visszaadja az x pozícióját

public int getY() – visszaadja az y pozícióját

Block osztály

Implementálja a Serializable interfészt, tárolja egy blokk állapotát és téglalapját

public void switchState() – ellentétjére változtatja a blokk állapotát

Command osztály

Implementálja a Serializable interfészt, tárol egy hangya parancsot

public Command(int aS, int bS, int d, int bN, int aN) – létrehozza és inicializálja a a tárolt adatokat

public void setArray() – A változókat sorrendben egy tömbbe tárolja, könnyebb visszaolvasásért.

public Integer[] getArray() – visszaadja a tömböt

Idozito osztály

Ősosztálya a TimerTask, beállítja, hogy időközönként lépjen egyet a hangya.

public Idozito(Ant a, App f) – Létrehozza az Idozitot és beállítja a hangyát hozzá

public void run() – beállítja hogy ha elindul az időzítő lépjen a hangya és ha ki akar lépni a rácsból leállítja az időzítőt

blockClicked osztály

A MouseListener interfészt implementálja, beállítja mi történjen egy blokkra kattintva

blockClicked(Grid gr) – létrehozza a példányt és beállítja melyik Gridhez tartozik

public void mouseClicked(MouseEvent e) – azt a pontot használva mai kattintva lett, vé megy a blokkokon hogy valamelyikben benne volt-e és ha igen annak megváltoztatja az állapotát

readClicked osztály

A MouseListener interfészt implementálja, beállítja mi történjen a Read-re kattintva

readClicked(App fr) – létrehoz egy példányt és beállítja melyik Apphoz tartozik

public void mouseClicked(MouseEvent arg0) – átadja a szövegdoboz szövegét a hangyának majd programmeAnt függvény visszatérési értéke alapján kiíratja piros szöveggel az eredményt

restartClicked osztály

A MouseListener interfészt implementálja, beállítja mi történjen a Restartra kattintva

restartClicked(App fr) – létrehoz egy példányt és beállítja melyik Apphoz tartozik

public void mouseClicked(MouseEvent arg0) – meghívja a az App Gridjének setUp() függvényét ami alaphelyzetbe állítja a rácsot

startClicked osztály

A MouseListener interfészt implementálja, beállítja mi történjen a Startra kattintva

StartClicked(App fr) – létrehoz egy példányt és beállítja melyik Apphoz tartozik

public void mouseClicked(MouseEvent arg0) – elindítja az App időzítőjét

stopClicked osztály

A MouseListener interfészt implementálja, beállítja mi történjen a Stopra kattintva

StopClicked(App fr) – létrehoz egy példányt és beállítja melyik Apphoz tartozik

public void mouseClicked(MouseEvent arg0) – leállítja az App időzítőjét

decTime osztály

A MouseListener interfészt implementálja, beállítja mi történjen a „>>”-ra kattintva

decTime (App fr) – létrehoz egy példányt és beállítja melyik Apphoz tartozik

public void mouseClicked(MouseEvent arg0) – leállítja a régit és létrehoz egy új időzítőt az Appnak, amiben a periódust csökkenti App.timeDif-el (ha a jelenlegi idő nagyobb mint timeDif)

incTime osztály

A MouseListener interfészt implementálja, beállítja mi történjen a „<<”-ra kattintva

incTime (App fr) – létrehoz egy példányt és beállítja melyik Apphoz tartozik

public void mouseClicked(MouseEvent arg0) – leállítja a régit és létrehoz egy új időzítőt az Appnak, amiben a periódust növeli App.timeDif-el.

saveClicked osztály

A MouseListener interfészt implementálja, beállítja mi történjen Save-re kattintva

saveClicked(App fr) – létrehozza a példányt és beállítja melyik Apphoz tartozik

public void mouseClicked(MouseEvent e) – megformázza a jelenlegi dátumot, majd azt fájlnévként használva szerializál egy SavedStuff példányt

loadClicked osztály

A MouseListener interfészt implementálja, beállítja mi történjen Load-ra kattintva.

saveClicked(App fr) – létrehozza a példányt és beállítja melyik Apphoz tartozik

public void mouseClicked(MouseEvent e) – megnyit egy JFileChoosert és az abban kiválasztott fájlt visszatölti egy SavedStuff példányba. A SavedStuff adatait ezután arra használja, hogy átállítsa a Grid adatait.

4. Tesztek

AntTest.java

public void commandLengthTest() – teszteli, hogy jól olvassa-e be a hangya hogy hány sorból áll az input

public void commandTest() – teszteli, hogy a hangya az input adatait helyesen olvassa és tárolja el

public void stepTest() – teszteli , hogy a hangya az adott parancsok alapján jól tud-e lépni

LoadTest.java

public void sqrTest() – teszteli, hogy a Grid helyesen változtatja-e meg és tölti be a mentett állapotot ha a jelenlegi és a betöltött rácsok száma nem egyezik meg