Vírus szimuláció

Készítette Doxygen 1.9.0

1. fejezet

Prog3: Vírus szimuláció nagyházifeladat terv

1.1. Az ötlet leírása

A program vírus terjedését szimulálja. A pálya egy téglalap lenne, ahhol színes pöttyök tudnának ütközni(kontakt). 4 féle pötty található a pályán:

- Fekete: halott(nem mozog, idővel eltűnik)
- Piros: fertőző.
- Zöld: gyógyult.
- Fehér: semleges.

A különböző faktorokat csúszkákkal lehetne állítani: pl: pötty sebessége, milyen eséllyel fertőz, halálozási esély, gyógyulási idő stb. Ha a pötty falnak ütközik, vagy másik pöttyel, akkor visszapattan. A programhoz tartozik egy diagram is, ami a pöttyökről mutat statisztikát.

2. fejezet

Programozói dokumentáció

2.1. Fordítási tudnivalók

3. fejezet

Prog3: Vírus szimuláció nagyházifeladat specifikáció

3.1. Az ötlet leírása

A program vírus terjedését szimulálja. A pálya egy téglalap lenne, ahhol színes pöttyök tudnának ütközni(kontakt). 4 féle pötty található a pályán:

- Fekete: halott(nem mozog, idővel eltűnik)
- Piros: fertőző.
- Zöld: gyógyult.
- Fehér(szürke) semleges.

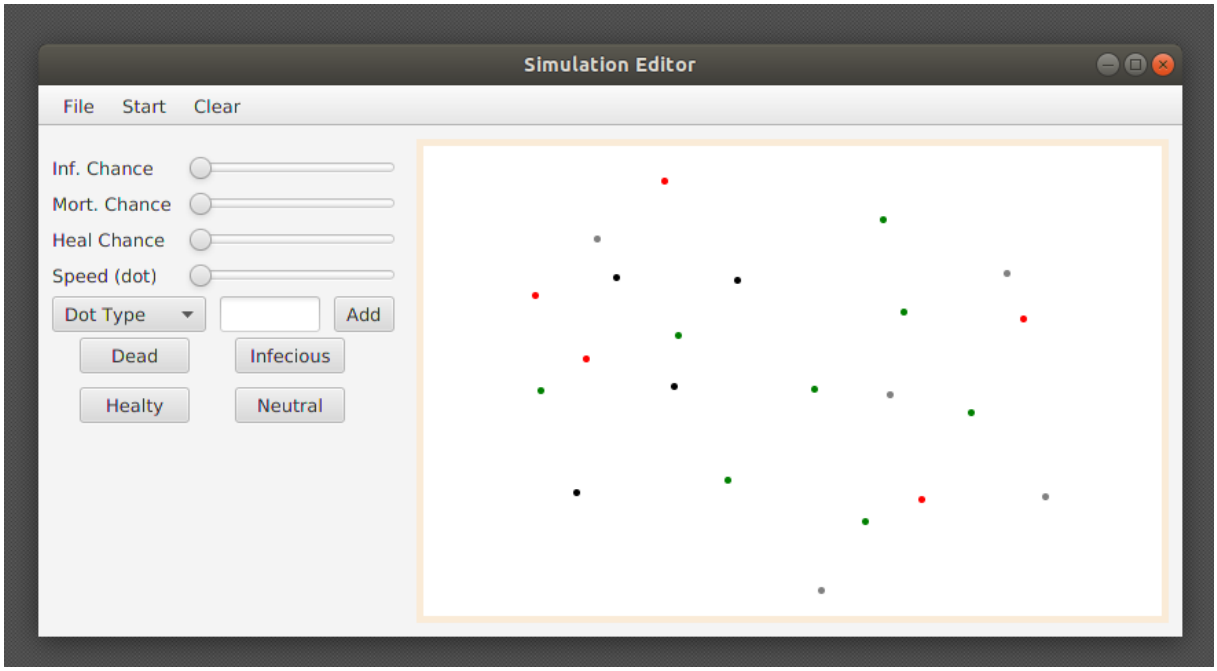
A különböző faktorokat csúszkákkal lehetne állítani: pl: pötty sebessége, milyen eséllyel fertőz, halálozási esély, gyógyulási idő stb. Ha a pötty falnak ütközik, vagy másik pöttyel, akkor visszapattan. A programhoz tartozik egy diagram is, ami a pöttyökről mutat statisztikát.

3.2. A program funkcionalitása a felhasználó szemszögéből

A program indítása után megnyílik a Simulation Editor (1. ábra)-hoz hasonló ablak. Itt lehetőségünk van egy szimuláció előkészítésére. A programrész funciói:

- Szimuláció előkészítése
  - Új (üres) szimuláció létrehozása(File>New)
  - Szimuláció betöltése fájlból (kezdeti értékek) (File>Open simulation)
  - Szimuláció mentése fájlba (kezdeti értékek) (File>Save simulation)
- Kezdeti értékek beállítása
  - Fertőzési esély (Infection Chance slider)
  - Halálozási esély (Mortality Chance slider)
  - Gyógyulási esély (Heal Chance slider)

  - Pötty sebessége(Speed slidWer)
  - n db pötty felhelyezése véletlenszerűen a pályára (Add button)
  - pöttyök egyessével történő felhelyezése a pályára, egér kattintás alapján (4db button)
  - A pálya kezdeti értékeinek törlése (Clear)
- Szimuláció elindítása
  - Kezdeti értékek alapján(Start >Start)



1. ábra - Szimuláció előkészítése.

A Start menüpont megnyitásával megnyillik egy új ablak, amiben a vírus szimulációja történik. Ez hasonlóan fog kinézni, mint az Editor. A programrész funkciói:

- Szimuláció kezelése
  - Idő elindítása, megállítása, gyorsítása, lassítása, léptetés egyesével
  - statisztika megnyitása

A Statistics gomb megnyomása utána megjelenik egy új ablakban a szimulációhoz tartozó statisztika. Ez szintén hasonlóan fog kinézni, mint az Editor.

- Szimuláció statisztikája
  - Olyan diagram(idő szerint), ahol ábrázolva vannak a fontos adatok. (Ferőzöttek, halottak, gyógyultak, stb)

3.3. Megoldási ötlet (vázlat)

A megoldáshoz JavaFX alapú GUI-t fogok használni. A mintaképen látható módon fogom ezt elkészíteni. A kezdeti értékek mentése és betöltése szerializálás segítségével fog történni. A program (legalább matematikai szempontból) lényeges részeihez JUnit tesztet fogok készíteni.

A csúszkákat $x=0..1$ -ig lehet állítani (kivéve sebesség csúszka), valós számra. A csúszkákhoz tartozó esemény bekövetkezésénél (pl.: ütközés) generálok egy véletlen számot $ 0..1 =r$ között(valós). Ha $ r<x$, akkor bekövetkezik az esemény (pl.: a kontakt megfertőződik).

A sebesség csúszkát -8 és +8 között lehet állítani. A keletkező sebesség szorzót(v) az alábbi képtlet fogja számolni: $ v = 2^{\left \lfloor{x}\right \rfloor } $, ahol $ x$ a csúszka által kapott valós szám.

Pötty mozgásának irányát egy 0 és 7 közt generált véletlen egész szám fogja meghatározni Kontaktálás esetén a pötty1 és pötty2 közepe között kevesebb mint 2r távolság van.

4. fejezet

Felhasználói dokumentáció

4.1. A program általános leírása

A program célja egy vírus terjedésének és lefolyásának szimulálása a lakosságon. A programban Pöttyök (Dot) jelképezi az egyes embereket. 4 féle Dot található a programban:

- Neutral(Semleges): A még meg nem fertőződött személyeket jelképezik
- Infectious(Fertőző): A már megfertőzött személyeket jelképezik
- Healthy(Egészséges): A már fertőző pöttyök képesek a gyógyulásra előre beállított eséllyel
- Dead(Halott): A már fertőző pöttyök képesek a meghalásra előre beállított eséllyel

A szimulációhoz egy téglalap tartozik. Ide lehet lehelyezni a pöttyöket. Ha két pötty ütközik egymással, akkor rugalmas ütközés megy végbe. Az ütközés jelképezi a kontaktálást. Ha egy semleges pötty fertőzöttel ütközik, akkor az előre beállított esély alapján van esély a megfertőződésre.

Egy fertőzött pötty másokat tovább tud fertőzni, illetve a lappangási idő letelte után (500 tick), lehetősége lesz gyógyulni, illetve meghalni a beállított valószínűség alapján.

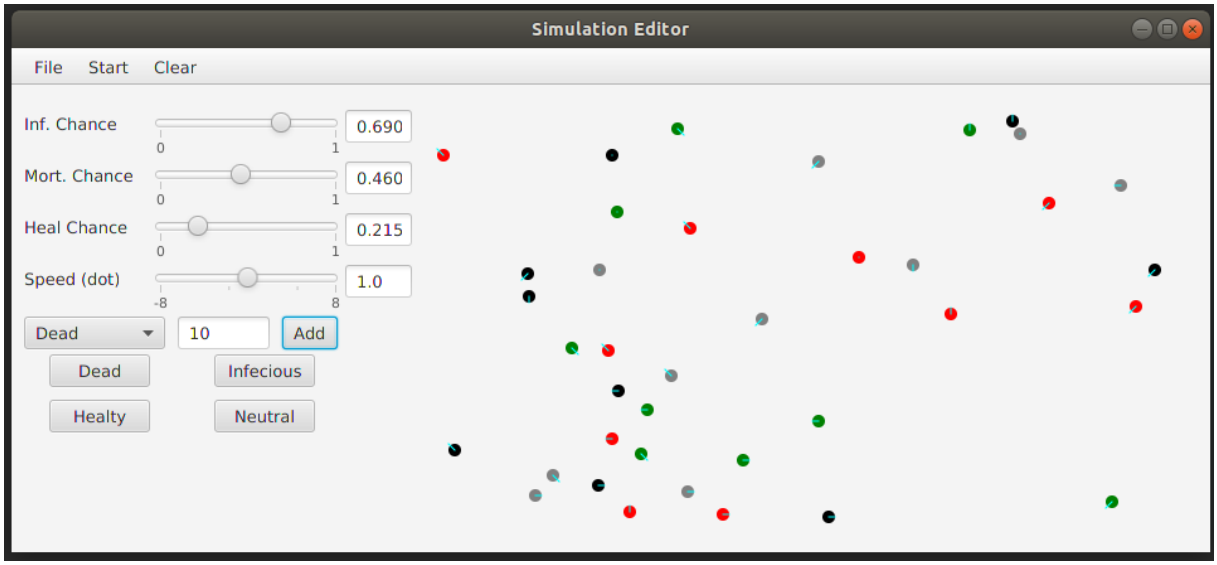
4.2. A program funkcionalitása a felhasználó szemszögéből

A program indítása után megnyílik a Simulation Editor (2. ábra) és betöltődik egy új üres szimuláció. Itt lehetőségünk van egy szimuláció előkészítésére. A programrész funciói:

- Szimuláció előkészítése
  - Szimuláció betöltése fájlból (kezdeti értékek) (File>Open simulation...)
  - Szimuláció mentése fájlba (kezdeti értékek) (File>Save simulation)
- Kezdeti értékek beállítása
  - Fertőzési esély (Infection Chance slider)
  - Halálozási esély (Mortality Chance slider)
  - Gyógyulási esély (Heal Chance slider)

  - Pötty sebessége(Speed slidWer)
  - n db pötty felhelyezése véletlenszerűen a pályára (Add button)
  - pöttyök egyessével történő felhelyezése a pályára, egér kattintás alapján (4db button)
  - A pálya kezdeti értékeinek törlése (Clear > Clear)

- Szimuláció elindítása
  - Kezdeti értékek alapján(Start > Start)

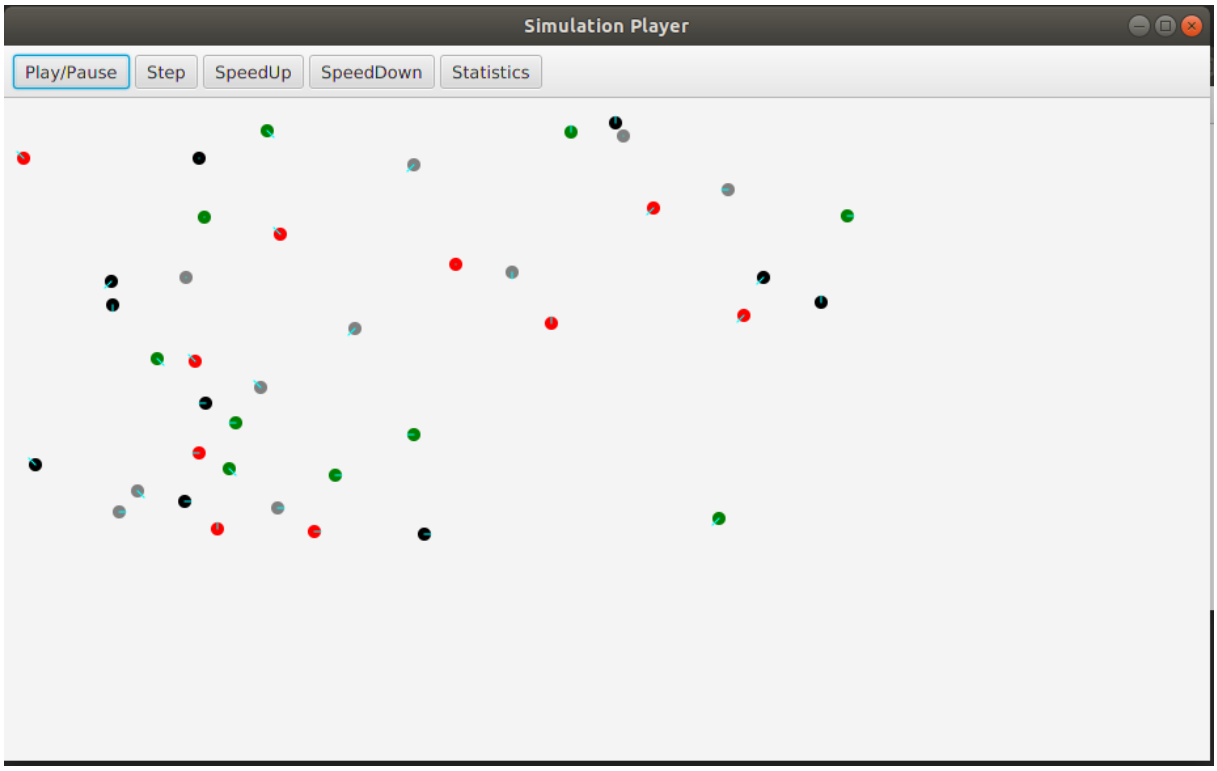


2.ábra - Szimuláció előkészítése.

A szimuláció előkészítéseként be kell állítanunk a csúszkák segítségével az esélyeket, majd pöttyöket kell hozzáadnunk a szimuláció sablonhoz. Pöttyöket kétféleképpen adhatunk hozzá. A legerdülő menüben kiválasztjuk a pötty típusát a mezőbe beírjuk a lerakandó pöttyök számát(Egész szám 0 és n között), majd rányomunk az Add gombra. A másik lehetőség, hogy először a 4 típust reprezentáló gombok egyikére kattintunk, majd a kívánt helyre mozgatva az egeret a bal egérgombbal kattintunk.

Ha elégedettek vagyunk a szimulálandó vírus sablonjával, akkor a sablont elmenthetjük a File>Save simulation segítségével.

A Start menüpont megnyitásával megnyillik egy új ablak, amiben a vírus szimulációja történik. (3.ábra)



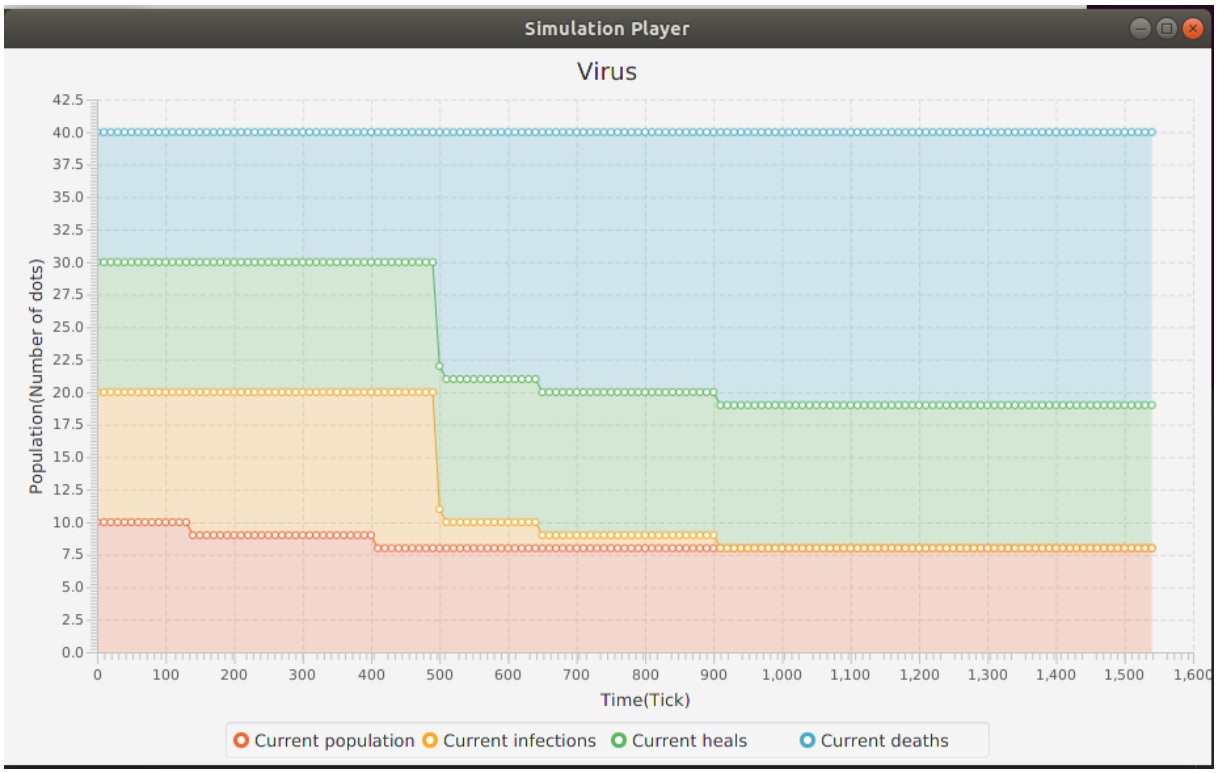
3.ábra - Vírus szimulálása

A programrész funkciói:

- Szimuláció kezelése
  - Idő elindítása megállítása (Play/Pause)
  - léptetés egyesével (Step)
  - Idő gyorsítása (SpeedUp)
  - Idő lassítása (SpeedDown)
  - statisztika megnyitása (Statistics)
- Szimuláció megjelenítése és lejátszása

A vírus terjedése körökre van osztva (Tick) a szimuláció sebessége nem befolyásolja ezt. Csupán a felhasználó számára telik két kör között gyorsabban az idő.

A Statistics gomb megnyomása utána megjelenik egy új ablakban a szimulációhoz tartozó statisztika. (4.ábra)



Ha a szimuláció már el lett indítva 10 Tickenként a pillanatnyi állás elküldésre kerül a grafikonhoz. A grafikon 100 ms -ként frissül. Ha a statisztika bezárásra kerül, akkor az addig kapott adatok törlődnek.

Ha a grafikont nem nyitjuk, meg csak a program futása után, az adatok akkor is megjelennek.

3.ábra - Statisztika megjelenítése

A programrész funkciói:

- Szimuláció statisztikája
  - Olyan diagram(tick szerint), ahol ábrázolva vannak a populáció, a fertőzöttek, a gyógyultak és a halottak

5. fejezet

Hierarchikus mutató

5.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

6. fejezet

Osztálymutató

6.1. Osztálylista

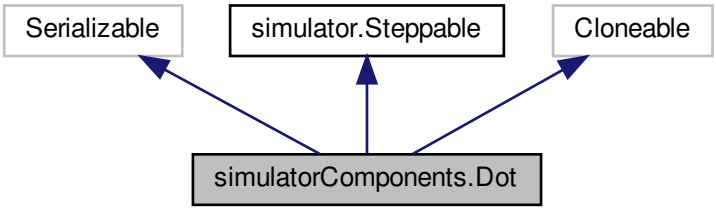
Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

7. fejezet

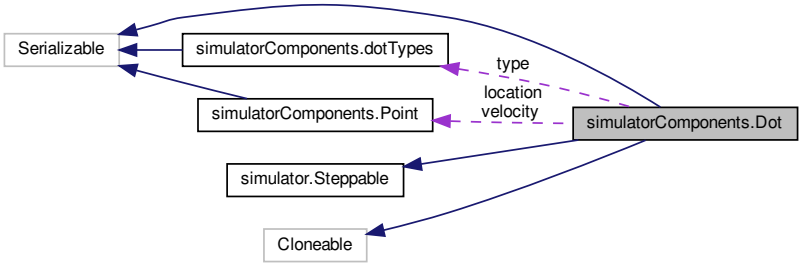
Osztályok dokumentációja

7.1. simulatorComponents.Dot osztályreferencia

A simulatorComponents.Dot osztály származási diagramja:



A simulatorComponents.Dot osztály együttműködési diagramja:



Publikus tagfüggvények

- Dot (double x, double y, double r)
- Dot (double x, double y, double r, double speed)
- Dot (double x, double y, double r, double speed, dotTypes type, double infChance, double mortChance, double healChance)
- Object clone () throws CloneNotSupportedException
- void initVelocity (double speed)
- Point getLocation ()
- void setLocation (Point location)
- void setHealChance (double heal)
- void setInfChance (double inf)
- void setMortChance (double mort)
- dotTypes getType ()
- double getRadius ()
- boolean isCollidedWith (Steppable st)
- boolean isCollidedWith (Dot dot)
- boolean isOutOfWindow (Canvas c)
- void hitBy (Dot d)
- void step (Canvas c)
- void moveBack (Canvas c)
- void init (Canvas c)
- void refresh (Canvas c)
- void draw (Canvas c)

Védett tagfüggvények

- void remove ()
- void bounceBack (Canvas c)

Privát tagfüggvények

- void infectedBy (Dot d)
- void heal ()
- void die ()
- void drawCenters (Dot dot, Canvas c)

Privát attribútumok

- Point location = null
- double radius
- Point velocity
- dotTypes type
- double infChance
- double mortChance
- double healChance
- double mass
- int sinceInfection = 0
- int sinceDead = 0

7.1.1. Részletes leírás

Dotokat reprezentáló osztály. Megvalósítja a Drawable, Serializable, Steppable és Clonable interfészeket.

7.1.2. Konstruktorok és destruktorok dokumentációja

7.1.2.1. Dot() [1/3]

```
simulatorComponents.Dot.Dot (
            double x,
            double y,
            double r )
```

A pötty konstruktora

Paraméterek

x	A pötty x kordinátája
y	A pötty y kordinátája
r	A pötty sugara

7.1.2.2. Dot() [2/3]

```
simulatorComponents.Dot.Dot (
            double x,
            double y,
            double r,
            double speed )
```

A pötty konstruktora

Paraméterek

x	A pötty x kordinátája
y	A pötty y kordinátája
r	A pötty sugara
speed	A pötty sebessége

7.1.2.3. Dot() [3/3]

```
simulatorComponents.Dot.Dot (
            double x,
```

```
            double y,
            double r,
            double speed,
            dotTypes type,
            double infChance,
            double mortChance,
            double healChance )
```

A pötty konstruktora

Paraméterek

x	A pötty x kordinátája
y	A pötty y kordinátája
r	A pötty sugara
speed	A pötty sebessége
type	A pötty típusa
infChance	Másik pöttyöt ilyen esélyel fertőz, ha a Dot fertőző
mortChance	A pötty halálozási esélye, ha már megfertőzödött és a vírus lappangási ideje lejárt
healChance	A pötty gyógyulási esélye, ha már megfertőzödött és a vírus lappangási ideje lejárt

7.1.3. Tagfüggvények dokumentációja

7.1.3.1. bounceBack()

```
void simulatorComponents.Dot.bounceBack (
            Canvas c )  [protected]
```

A pötty a Canvas szélén visszapattan

Paraméterek

c	A kapott Canvas

7.1.3.2. clone()

```
Object simulatorComponents.Dot.clone ( ) throws CloneNotSupportedException
```

A pötty által felüldefiniált clone metódus.

Visszatérési érték

A pötty Object

Kivételek

CloneNotSupportedException	kivételt dobhat(De nem fog, mert a Dot szerializálható)

7.1.3.3. die()

```
void simulatorComponents.Dot.die ( )  [private]
```

A pötty meghal Feladatai: A pötty típusának megváltoztatása, SimulationPlayer felé jelzi, hogy egy pötty meghalt

7.1.3.4. draw()

```
void simulatorComponents.Dot.draw (
            Canvas c )
```

A pöttyöt kirajzoló függvény. Kirajzolja a pöttyöt, majd rárajzolja a sebesség vektorát

Paraméterek

c	A kapott Canvas

Megvalósítja a következőket: simulator.Steppable.

7.1.3.5. drawCenters()

```
void simulatorComponents.Dot.drawCenters (
            Dot dot,
            Canvas c )  [private]
```

Két pötty közepe fölött megrajzolja a vectort. Debug célokra használtam, de nem töröltem.

Paraméterek

dot	A másik Dot
c	A kapott Canvas

7.1.3.6. getLocation()

```
Point simulatorComponents.Dot.getLocation ( )
```

Location getterje

Visszatérési érték

A hely

7.1.3.7. getRadius()

```
double simulatorComponents.Dot.getRadius ( )
```

A pötty sugarának gettere

Visszatérési érték

A pötty sugara

7.1.3.8. getType()

```
dotTypes simulatorComponents.Dot.getType ( )
```

A pötty típusának gettere

Visszatérési érték

A pötty típusa

7.1.3.9. heal()

```
void simulatorComponents.Dot.heal ( )  [private]
```

A pötty meggyógyul Feladatai: A pötty típusának megváltoztatása, SimulationPlayer felé jelzi, hogy egy pötty meggyógyult

7.1.3.10. hitBy()

```
void simulatorComponents.Dot.hitBy (
            Dot d )
```

A pötty másik pöttyel való ütközése során hívódik. Ez kezeli a statikus (pl.: két pötty fedné egymást) és dinamikus ütközést (rugalmas ütközés)

Paraméterek

d	A másik pötty

Megvalósítja a következőket: simulator.Steppable.

7.1.3.11. infectedBy()

```
void simulatorComponents.Dot.infectedBy (
            Dot d )  [private]
```

Fertőzés bekövetkezése Feladata, hogy ennek a pöttynek a megfelelő adatait beállítsa, a SimulationPlayer felé jelzi, hogy új fertőzés történt

Paraméterek

d	A fertőzést okozó pötty

7.1.3.12. init()

```
void simulatorComponents.Dot.init (
            Canvas c )
```

A pöttyöt inicializálja: Ha a Canvason kívül van visszahúzza őt, majd kirajzolja

Paraméterek

c	A kapott Canvas

Megvalósítja a következőket: simulator.Steppable.

7.1.3.13. initVelocity()

```
void simulatorComponents.Dot.initVelocity (
            double speed )
```

Sebesség vektor inicializálása a kapott sebesség alapján. A függvény a lehetséges irányt véletlenszerűen generálja.

Paraméterek

speed	A kapott sebesség

7.1.3.14. isCollidedWith() [1/2]

```
boolean simulatorComponents.Dot.isCollidedWith (
            Dot dot )
```

A pötty ütközött-e a kapott steppable-el?

Paraméterek

dot	A kapott pötty

Visszatérési érték

Igen vagy Nem, a két pötty közepei között lévő távolság és a sugarak összege alapján.

Megvalósítja a következőket: simulator.Steppable.

7.1.3.15. isCollidedWith() [2/2]

```
boolean simulatorComponents.Dot.isCollidedWith (
            Steppable st )
```

A pötty ütközött-e a kapott steppable-el?

Paraméterek

st	A kapott steppable

Visszatérési érték

Igen vagy Nem

Megvalósítja a következőket: simulator.Steppable.

7.1.3.16. isOutOfWindow()

```
boolean simulatorComponents.Dot.isOutOfWindow (
            Canvas c )
```

Megvizsgálja, hogy a pötty a canvason kívül van-e

Paraméterek

c	A kapott Canvas

Visszatérési érték

Igen vagy Nem

Megvalósítja a következőket: simulator.Steppable.

7.1.3.17. moveBack()

```
void simulatorComponents.Dot.moveBack (
            Canvas c )
```

Visszahúzza a pöttyöt a Canvas területére, ha kiment belőle

Paraméterek

c	A kapott Canvas

Megvalósítja a következőket: simulator.Steppable.

7.1.3.18. refresh()

```
void simulatorComponents.Dot.refresh (
            Canvas c )
```

A pötty újrarajzolása

Paraméterek

c	A kapott Canvas

Megvalósítja a következőket: simulator.Steppable.

7.1.3.19. remove()

```
void simulatorComponents.Dot.remove ( )  [protected]
```

A pötty hozzáadása a kör végén eltávolítandók listájához

7.1.3.20. setHealChance()

```
void simulatorComponents.Dot.setHealChance (
            double heal )
```

Gyógyulási esély settere

Paraméterek

heal	A kapott gyógyulási esély

7.1.3.21. setInfChance()

```
void simulatorComponents.Dot.setInfChance (
            double inf )
```

Átfertőzései esély settere

Paraméterek

inf	A kapott átfertőzési esély

7.1.3.22. setLocation()

```
void simulatorComponents.Dot.setLocation (
            Point location )
```

Location setterje

Paraméterek

location	A kapott hely

7.1.3.23. setMortChance()

```
void simulatorComponents.Dot.setMortChance (
            double mort )
```

Halálozási esély settere

Paraméterek

mort	A halálozási gyógyulási esély

7.1.3.24. step()

```
void simulatorComponents.Dot.step (
            Canvas c )
```

Lépés függvény Feladatai: Lappangási idő vizsgálata, Halálozás után eltelt idő vizsgálata, Pálya szélével történő ütközés, pötty léptetése

Paraméterek

c	A kapott Canvas

Megvalósítja a következőket: simulator.Steppable.

7.1.4. Adattagok dokumentációja

7.1.4.1. healChance

```
double simulatorComponents.Dot.healChance  [private]
```

A pötty gyógyulási esélye

7.1.4.2. infChance

```
double simulatorComponents.Dot.infChance  [private]
```

A pötty átfertőzésének esélye

7.1.4.3. location

```
Point simulatorComponents.Dot.location = null  [private]
```

A pötty közepének poziciója

7.1.4.4. mass

```
double simulatorComponents.Dot.mass  [private]
```

A pötty tömege

7.1.4.5. mortChance

```
double simulatorComponents.Dot.mortChance  [private]
```

A pötty halálozási esélye

7.1.4.6. radius

```
double simulatorComponents.Dot.radius  [private]
```

A pötty sugara

7.1.4.7. sinceDead

```
int simulatorComponents.Dot.sinceDead = 0  [private]
```

A halál pillanatától eltelt tickek száma

7.1.4.8. sinceInfection

```
int simulatorComponents.Dot.sinceInfection = 0  [private]
```

Fertőzés óta eltelt tickek száma

7.1.4.9. type

```
dotTypes simulatorComponents.Dot.type  [private]
```

A pötty típusa

7.1.4.10. velocity

```
Point simulatorComponents.Dot.velocity  [private]
```

A pötty sebesség vektorának végpontja (Mintha a Vector az (0,0)-ból mutatna velocity-ba)

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- Dot.java

7.2. Tests.DotTest osztályreferencia

Publikus tagfüggvények

- void isCollidedWith ()

7.2.1. Részletes leírás

Dot tesztelése

7.2.2. Tagfüggvények dokumentációja

7.2.2.1. isCollidedWith()

```
void Tests.DotTest.isCollidedWith ( )
```

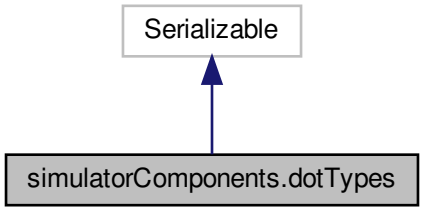
Ütközés tesztelése

Ez a dokumentáció az osztályról a következő fájl alapján készült:

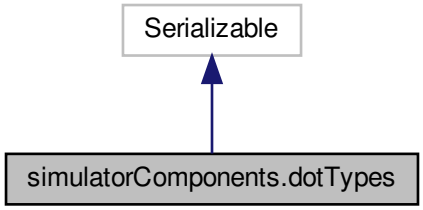
- DotTest.java

7.3. simulatorComponents.dotTypes felsoroló referencia

A simulatorComponents.dotTypes osztály származási diagramja:



A simulatorComponents.dotTypes osztály együttműködési diagramja:



Publikus attribútumok

- None
- Healthy
- Infectious
- Neutral
- Dead

7.3.1. Részletes leírás

Enum, ami a pöttyök típusát tárolja. Megvalósítja a Serializable interfészt

7.3.2. Adattagok dokumentációja

7.3.2.1. Dead

`simulatorComponents.dotTypes.Dead`

Halott

7.3.2.2. Healthy

`simulatorComponents.dotTypes.Healthy`

Egészséges

7.3.2.3. Infectious

`simulatorComponents.dotTypes.Infectious`

Fertőző: Lappangási idő után meghalhat vagy meggyúgyulhat

7.3.2.4. Neutral

`simulatorComponents.dotTypes.Neutral`

Semleges: Sima személy

7.3.2.5. None

`simulatorComponents.dotTypes.None`

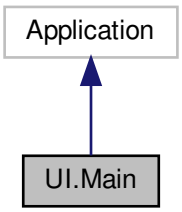
Nincs: hiba kezelés miatt

A dokumentáció ehhez az enum-hoz a következő fájl alapján készült:

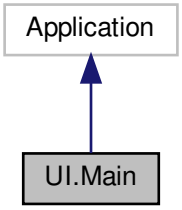
- dotTypes.java

7.4. UI.Main osztályreferencia

Az UI.Main osztály származási diagramja:



Az UI.Main osztály együttműködési diagramja:



Publikus tagfüggvények

- void start (Stage primaryStage) throws IOException

Statikus publikus tagfüggvények

- static void main (String[ ] args)

7.4.1. Részletes leírás

Main class. Ez a program belépési pontja.

7.4.2. Tagfüggvények dokumentációja

7.4.2.1. main()

```
static void UI.Main.main (
            String[] args )  [static]
```

Ez a program belépési pontja.

Paraméterek

args	parancssori argumentumok listája. Nem használom.

7.4.2.2. start()

```
void UI.Main.start (
            Stage primaryStage ) throws IOException
```

Az ablakot elindító függvény.

Paraméterek

primaryStage	JavaFX rendszertől kapott primary stage.

Kivételek

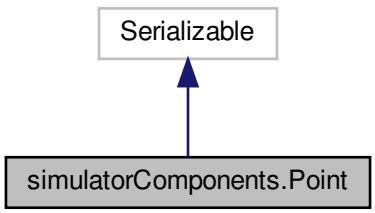
IOException	Akkor dobja, ha a simulationEditor.fxml nem található.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

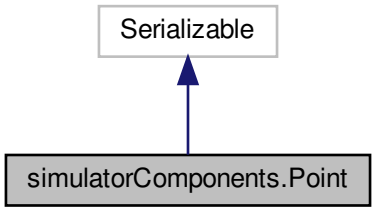
- Main.java

7.5. simulatorComponents.Point osztályreferencia

A simulatorComponents.Point osztály származási diagramja:



A simulatorComponents.Point osztály együttműködési diagramja:



Publikus tagfüggvények

- Point (double x, double y)
- Point (Point p)
- double getX ()
- double getY ()
- double calcDistance (Point p)
- double calcDisplacement (Point p)
- void add (Point p)
- void subtract (Point p)
- void multiply (double val)
- void divide (double val)
- double dotProduct (Point v)
- boolean isOutOfCanvas (Canvas c, double r)

Csomag függvények

- void add (double x, double y)
- void subtract (double x, double y)
- boolean isOutOfCanvasTop (Canvas c, double r)
- boolean isOutOfCanvasBottom (Canvas c, double r)
- boolean isOutOfCanvasLeft (Canvas c, double r)
- boolean isOutOfCanvasRight (Canvas c, double r)

Csomag attribútumok

- double x
- double y

7.5.1. Részletes leírás

Point osztály Feladata: az összetartozó x és y értékek tárolása és ezeken műveletek végzése Abban az esetben, ha vectort fejez ki, akkor a Pont az Origóba tolt vector végpontját jelenti.

7.5.2. Konstruktorok és destruktorok dokumentációja

7.5.2.1. Point() [1/2]

```
simulatorComponents.Point.Point (
            double x,
            double y )
```

Point konstruktura

Paraméterek

x	A kapott x kordináta
y	A kapott y kordináta

7.5.2.2. Point() [2/2]

```
simulatorComponents.Point.Point (
            Point p )
```

Point osztály konstruktora

Paraméterek

p	A kapott Pont

7.5.3. Tagfüggvények dokumentációja

7.5.3.1. add() [1/2]

```
void simulatorComponents.Point.add (
            double x,
            double y )  [package]
```

Egy ponthoz hozzáad egy X és Y értéket

Paraméterek

x	A kapott x érték
y	A kapott y érték

7.5.3.2. add() [2/2]

```
void simulatorComponents.Point.add (
            Point p )
```

Két pontot összead x és y kordináták alapján, az eredmény az első operandusban tárolódik

Paraméterek

p	A kapott Pont

7.5.3.3. calcDisplacement()

```
double simulatorComponents.Point.calcDisplacement (
            Point p )
```

Két pont közötti elmozdulás számolása

Paraméterek

p	A kapott Pont

Visszatérési érték

Az elmozdulás vector

7.5.3.4. calcDistance()

```
double simulatorComponents.Point.calcDistance (
            Point p )
```

Két pont között számol távolságot, távolság = ((x1-x2)^2+(y1-y2)^2)^(1/2) képlet segítségével

Paraméterek

p	A kapott Pont

Visszatérési érték

A távolság, csak pozitív vagy 0 lehet

7.5.3.5. divide()

```
void simulatorComponents.Point.divide (
            double val )
```

Egy pont X és Y kordinátáját elosztja az értékkel

Paraméterek

val	Az érték, amivel leosztunk

7.5.3.6. dotProduct()

```
double simulatorComponents.Point.dotProduct (
            Point v )
```

Skaláris szorzat számolását végzi két vector között

Paraméterek

v	A kapott vector

Visszatérési érték

A kiszámolt skaláris szorzatot visszaadja.

7.5.3.7. getX()

```
double simulatorComponents.Point.getX ( )
```

X getter

Visszatérési érték

x értéke

7.5.3.8. getY()

```
double simulatorComponents.Point.getY ( )
```

y getter

Visszatérési érték

y értéke

7.5.3.9. isOutOfCanvas()

```
boolean simulatorComponents.Point.isOutOfCanvas (
            Canvas c,
            double r )
```

Megvizsgálja, hogy a pont kilóg-e a Canvas-ról legalább egy oldalon

Paraméterek

c	A kapott Canvas
r	A kapott korrekciós érték (általában a Dot sugara)

Visszatérési érték

Igen vagy Nem

7.5.3.10. isOutOfCanvasBottom()

```
boolean simulatorComponents.Point.isOutOfCanvasBottom (
            Canvas c,
            double r )  [package]
```

Megvizsgálja, hogy a pont alul kilóg-e a Canvas-ról

Paraméterek

c	A kapott Canvas
r	A kapott korrekciós érték (általában a Dot sugara)

Visszatérési érték

Igen vagy Nem

7.5.3.11. isOutOfCanvasLeft()

```
boolean simulatorComponents.Point.isOutOfCanvasLeft (
            Canvas c,
            double r )  [package]
```

Megvizsgálja, hogy a pont bal oldalt kilóg-e a Canvas-ról

Paraméterek

c	A kapott Canvas
r	A kapott korrekciós érték (általában a Dot sugara)

Visszatérési érték

Igen vagy Nem

7.5.3.12. isOutOfCanvasRight()

```
boolean simulatorComponents.Point.isOutOfCanvasRight (
            Canvas c,
            double r )  [package]
```

Megvizsgálja, hogy a pont jobb oldalt kilóg-e a Canvas-ról

Paraméterek

c	A kapott Canvas
r	A kapott korrekciós érték (általában a Dot sugara)

Visszatérési érték

Igen vagy Nem

7.5.3.13. isOutOfCanvasTop()

```
boolean simulatorComponents.Point.isOutOfCanvasTop (
            Canvas c,
            double r )  [package]
```

Megvizsgálja, hogy a pont felül kilóg-e a Canvas-ról

Paraméterek

c	A kapott Canvas
r	A kapott korrekciós érték (általában a Dot sugara)

Visszatérési érték

Igen vagy Nem

7.5.3.14. multiply()

```
void simulatorComponents.Point.multiply (
            double val )
```

Egy pont X és Y kordinátáját megszorozza a kapott értékkel

Paraméterek

val	A kapott érték

7.5.3.15. subtract() [1/2]

```
void simulatorComponents.Point.subtract (
            double x,
            double y )  [package]
```

Egy pontból kivon egy X és Y értéket

Paraméterek

x	A kapott x érték
y	A kapott y érték

7.5.3.16. subtract() [2/2]

```
void simulatorComponents.Point.subtract (
            Point p )
```

Két pontot kivon x és y kordináták alapján, az eredmény az első operandusban tárolódik

Paraméterek

p	A kapott Pont

7.5.4. Adattagok dokumentációja

7.5.4.1. x

```
double simulatorComponents.Point.x  [package]
```

x kordináta a bal felső sarokban van a (0,0)

7.5.4.2. y

```
double simulatorComponents.Point.y  [package]
```

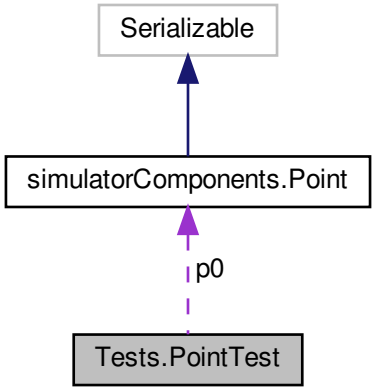
y kordináta a bal felső sarokban van a (0,0)

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- Point.java

7.6. Tests.PointTest osztályreferencia

A Tests.PointTest osztály együttműködési diagramja:



Publikus tagfüggvények

- void setUp ()
- void calcDistance ()
- void calcDisplacement ()
- void add ()
- void subtract ()
- void multiply ()
- void divide ()
- void divideByZero ()
- void dotProduct ()

Csomag attribútumok

- Point p0

7.6.1. Részletes leírás

Point tesztelésére szolgál

7.6.2. Tagfüggvények dokumentációja

7.6.2.1. add()

```
void Tests.PointTest.add ( )
```

add teszt

7.6.2.2. calcDisplacement()

```
void Tests.PointTest.calcDisplacement ( )
```

calcDisplacement teszt

7.6.2.3. calcDistance()

```
void Tests.PointTest.calcDistance ( )
```

calcDistance teszt

7.6.2.4. divide()

```
void Tests.PointTest.divide ( )
```

divide teszt

7.6.2.5. divideByZero()

```
void Tests.PointTest.divideByZero ( )
```

divide by Zero teszt

7.6.2.6. dotProduct()

```
void Tests.PointTest.dotProduct ( )
```

dotProduct teszt

7.6.2.7. multiply()

```
void Tests.PointTest.multiply ( )
```

multiply teszt

7.6.2.8. setUp()

```
void Tests.PointTest.setUp ( )
```

Create (0,0)

7.6.2.9. subtract()

```
void Tests.PointTest.subtract ( )
```

subtract teszt

7.6.3. Adattagok dokumentációja

7.6.3.1. p0

```
Point Tests.PointTest.p0  [package]
```

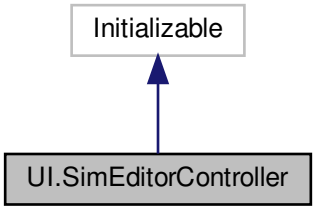
Origó

Ez a dokumentáció az osztályról a következő fájl alapján készült:

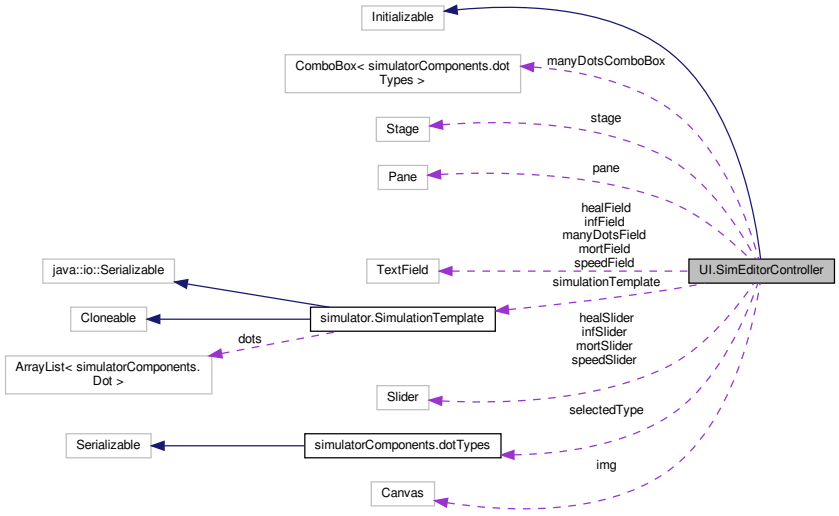
- PointTest.java

7.7. UI.SimEditorController osztályreferencia

Az UI.SimEditorController osztály származási diagramja:



Az UI.SimEditorController osztály együttműködési diagramja:



Publikus tagfüggvények

- SimEditorController (Stage st)
- void initialize (URL url, ResourceBundle resourceBundle)

Privát tagfüggvények

- void redraw ()
- void addManyDotsPressed ()
- void infSliderChanged ()
- void mortalitySliderChanged ()

- void healSliderChanged ()
- void speedSliderChanged ()
- void clearCanvas ()
- void createDotOnMousePosition (MouseEvent event)
- void setTypeOfDotOnMousePositionToDead ()
- void setTypeOfDotOnMousePositionToInfectious ()
- void setTypeOfDotOnMousePositionToHealthy ()
- void setTypeOfDotOnMousePositionToNeutral ()
- void serializeSimulationTemplate ()
- void openSerializedSimulationTemplate ()
- SimulationTemplate openSimulationTemplate ()
- void startSimulationPlayer () throws IOException
- void startSimulationPlayerFromFile () throws IOException

Privát attribútumok

- final Stage stage
- Canvas img
- Slider infSlider
- TextField infField
- Slider mortSlider
- TextField mortField
- Slider healSlider
- TextField healField
- Slider speedSlider
- TextField speedField
- Pane pane
- TextField manyDotsField
- ComboBox< dotTypes > manyDotsComboBox
- SimulationTemplate simulationTemplate
- simulatorComponents.dotTypes selectedType = simulatorComponents.dotTypes.None
- final double radius = 5.0

7.7.1. Részletes leírás

Simulation Editor ablak kontroller osztája. Feladata, hogy kezelje az ablakkal történő User interakciókat

7.7.2. Konstruktorok és destruktorok dokumentációja

7.7.2.1. SimEditorController()

```
UI.SimEditorController.SimEditorController (
            Stage st )
```

A kontroller konstruktora. Létrehozza a kontrollert, beállítja a stage-et és a simulationTemplate-et

Paraméterek

st	Stage, amit a Main-ben hozunk létre.

7.7.3. Tagfüggvények dokumentációja

7.7.3.1. addManyDotsPressed()

```
void UI.SimEditorController.addManyDotsPressed ( )  [private]
```

Több pötty véletlenszerű elhelyezését kezelő gomb megnyomása esetén hívódik. Feladata a TextField Integerré alakítása, ha ez nem megoldható hibaüzenetet küld. Ha a kapott szám Integerré alakítható, a megadott típus alapján meghívja n db alkalommal a Dot létrehozó függvényt.

7.7.3.2. clearCanvas()

```
void UI.SimEditorController.clearCanvas ( )  [private]
```

Canvas letörlését végzi. Új SimulationTemplate-et hoz létre.

7.7.3.3. createDotOnMousePosition()

```
void UI.SimEditorController.createDotOnMousePosition (
            MouseEvent event )  [private]
```

Egér kattintáskor meghívja a Dot létehoz függvényt, és frissíti a Canvast.

Paraméterek

event	A kapott MouseEvent

7.7.3.4. healSliderChanged()

```
void UI.SimEditorController.healSliderChanged ( )  [private]
```

Gyógyulási esély beállító csúszka változásakor hívódik. Feladata, hogy ezt az értéket továbbítsa a simulation↩ Template-nek, és kiírja ezt a megfelelő TextFieldbe.

7.7.3.5. infSliderChanged()

```
void UI.SimEditorController.infSliderChanged ( )  [private]
```

Fertőzési esély beállító csúszka változásakor hívódik. Feladata, hogy ezt az értéket továbbítsa a simulation↩ Template-nek, és kiírja ezt a megfelelő TextFieldbe.

7.7.3.6. initialize()

```
void UI.SimEditorController.initialize (
            URL url,
            ResourceBundle resourceBundle )
```

Az ablak inicializálója. Feladata az ablakon található elemek értékeinek beállítása.

Paraméterek

url	JavaFX használja relatív útvonal meghatározása a root objectnek
resourceBundle	Azok a források, amik a root object helyének meghatározásához szükségesek

7.7.3.7. mortalitySliderChanged()

```
void UI.SimEditorController.mortalitySliderChanged ( )  [private]
```

Halálozási esély beállító csúszka változásakor hívódik. Feladata, hogy ezt az értéket továbbítsa a simulation↩ Template-nek, és kiírja ezt a megfelelő TextFieldbe.

7.7.3.8. openSerializedSimulationTemplate()

```
void UI.SimEditorController.openSerializedSimulationTemplate ( )  [private]
```

Meghívja a simulationTemplate-et deszerializáló függvényt. SimulationTemplate-et beállítja a kapott értékre. A csúszkákat és a hozzájuk tartozó TextField-et beállítja a megfelelő értékre. Ha null a kapott template a függvény visszatér. A hibajelzés már az openSimulationTemplate()-ben megtörtént.

7.7.3.9. openSimulationTemplate()

```
SimulationTemplate UI.SimEditorController.openSimulationTemplate ( )  [private]
```

Deszerializálja a megadott simulationTemplate-et.

Visszatérési érték

SimulationTemplate visszadja egy deszerializált Template-et, vagy nullt-t ha nem sikerült a folyamat.

7.7.3.10. redraw()

```
void UI.SimEditorController.redraw ( )  [private]
```

Feladata az ablak újrarajzolása.

7.7.3.11. serializeSimulationTemplate()

```
void UI.SimEditorController.serializeSimulationTemplate ( )  [private]
```

Elmenti a simulationTemplate-et szerializálás segítségével fájlba. Értesíti a User-t, ha sikeres. Értesíti a felhasználót, ha a fájl null vagy nem létezik.

7.7.3.12. setTypeOfDotOnMousePositionToDead()

```
void UI.SimEditorController.setTypeOfDotOnMousePositionToDead ( )  [private]
```

Beállítja az egér által lehelyezendő Dot típusát halottra.

7.7.3.13. setTypeOfDotOnMousePositionToHealthy()

```
void UI.SimEditorController.setTypeOfDotOnMousePositionToHealthy ( )  [private]
```

Beállítja az egér által lehelyezendő Dot típusát egészségesre.

7.7.3.14. setTypeOfDotOnMousePositionToInfectious()

```
void UI.SimEditorController.setTypeOfDotOnMousePositionToInfectious ( )  [private]
```

Beállítja az egér által lehelyezendő Dot típusát fertőzőre.

7.7.3.15. setTypeOfDotOnMousePositionToNeutral()

```
void UI.SimEditorController.setTypeOfDotOnMousePositionToNeutral ( )  [private]
```

Beállítja az egér által lehelyezendő Dot típusát közömbösre.

7.7.3.16. speedSliderChanged()

```
void UI.SimEditorController.speedSliderChanged ( )  [private]
```

Sebesség beállító csúszka változásakor hívódik. Feladata, hogy ezt az értéket továbbítsa a simulationTemplatenek, és kiírja ezt a megfelelő TextFieldbe.

7.7.3.17. startSimulationPlayer()

```
void UI.SimEditorController.startSimulationPlayer ( ) throws IOException  [private]
```

Létrehozza és elindítja a SimulationPlayer ablakot.

Kivételek

IOException	Kivételt dob, ha az fxml nem létezik.

7.7.3.18. startSimulationPlayerFromFile()

```
void UI.SimEditorController.startSimulationPlayerFromFile ( ) throws IOException  [private]
```

Elindítja a SimulationPlayer-t közvetlenül fájlból.

Kivételek

IOException	Kivételt dob, ha az fxml nem létezik.

7.7.4. Adattagok dokumentációja

7.7.4.1. healField

```
TextField UI.SimEditorController.healField  [private]
```

TextField, itt jelezzük vissza az Usernek a beállított gyógyulás értékét.

7.7.4.2. healSlider

```
Slider UI.SimEditorController.healSlider  [private]
```

Slider, a gyúgyulási esély beállítására.

7.7.4.3. img

```
Canvas UI.SimEditorController.img  [private]
```

Canvas, a Dot-ok jelennek meg.

7.7.4.4. infField

```
TextField UI.SimEditorController.infField  [private]
```

TextField, itt jelezzük vissza az Usernek a beállított infection értékét.

7.7.4.5. infSlider

```
Slider UI.SimEditorController.infSlider  [private]
```

Slider, az átfertőzés esélyének beállítására.

7.7.4.6. manyDotsComboBox

```
ComboBox<dotTypes> UI.SimEditorController.manyDotsComboBox  [private]
```

ComboBox, a User itt választja ki a Dot típusát

7.7.4.7. manyDotsField

```
TextField UI.SimEditorController.manyDotsField  [private]
```

TextField, a User itt adja meg a lerakni kívánt Dotok, számát. Csak Integer lehet, különben hibaüzenet keletkezik

7.7.4.8. mortField

```
TextField UI.SimEditorController.mortField  [private]
```

TextField, itt jelezzük vissza az Usernek a beállított halálozás értékét.

7.7.4.9. mortSlider

```
Slider UI.SimEditorController.mortSlider  [private]
```

Slider, a halálozási esély beállítására.

7.7.4.10. pane

```
Pane UI.SimEditorController.pane  [private]
```

Pane, ebben található a canvas, amire rajzolunk.

7.7.4.11. radius

```
final double UI.SimEditorController.radius = 5.0  [private]
```

Pötty alapértelmezett sugara.

7.7.4.12. selectedType

```
simulatorComponents.dotTypes UI.SimEditorController.selectedType = simulatorComponents.dotTypes.None
[private]
```

Gomb által kiválasztott Dot típusának tárolója.

7.7.4.13. simulationTemplate

```
SimulationTemplate UI.SimEditorController.simulationTemplate  [private]
```

SimulationTemplate-et tárolunk, az Editor ezt módosítja. Ez tárolja a szimulációhoz elindításához szükséges adatokat.

7.7.4.14. speedField

```
TextField UI.SimEditorController.speedField  [private]
```

TextField, itt jelezzük vissza az Usernek a beállított sebesség értékét.

7.7.4.15. speedSlider

```
Slider UI.SimEditorController.speedSlider  [private]
```

Slider, a Dot sebességének beállítására.

7.7.4.16. stage

```
final Stage UI.SimEditorController.stage  [private]
```

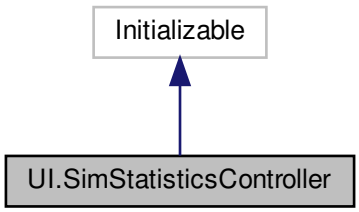
Stage eltárolása. Konstruktorban kapjuk az ablak létrehozása során. Ez alapján pozícionáljuk a többi ablakot.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

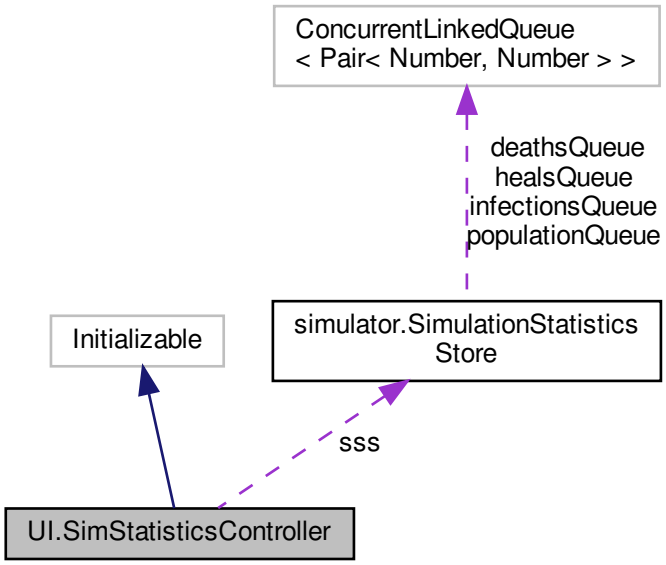
- SimEditorController.java

7.8. UI.SimStatisticsController osztályreferencia

Az UI.SimStatisticsController osztály származási diagramja:



Az UI.SimStatisticsController osztály együttműködési diagramja:



Publikus tagfüggvények

- SimStatisticsController (Stage st, SimulationStatisticsStore sss)
- void initialize (URL url, ResourceBundle resourceBundle)
- void updateChart ()

Csomag attribútumok

- SimulationStatisticsStore sss

Privát attribútumok

- StackedAreaChart< Number, Number > chart
- final XYChart.Series< Number, Number > population = new XYChart.Series<>()
- final XYChart.Series< Number, Number > deaths = new XYChart.Series<>()
- final XYChart.Series< Number, Number > infections = new XYChart.Series<>()
- final XYChart.Series< Number, Number > heals = new XYChart.Series<>()

7.8.1. Részletes leírás

Simulation Statistics Kontroller osztája Feladata, hogy kezelje az ablakkal történő User interakciókat

7.8.2. Konstruktorok és destruktorok dokumentációja

7.8.2.1. SimStatisticsController()

```
UI.SimStatisticsController.SimStatisticsController (
            Stage st,
            SimulationStatisticsStore sss )
```

SimStatisticsController konstrukturja

Paraméterek

st	A kapott Stage
sss	A kapott SimulationStatisticsStore

7.8.3. Tagfüggvények dokumentációja

7.8.3.1. initialize()

```
void UI.SimStatisticsController.initialize (
            URL url,
            ResourceBundle resourceBundle )
```

Az ablak inicializálója. Feladata az ablakon található elemek értékeinek beállítása.

Paraméterek

url	JavaFX használja relatív útvonal meghatározása a root objectnek
resourceBundle	Azok a források, amik a root object helyének meghatározásához szükségesek

7.8.3.2. updateChart()

```
void UI.SimStatisticsController.updateChart ( )
```

Frissíti a grafikonon megjelenő adatokat

7.8.4. Adattagok dokumentációja

7.8.4.1. chart

```
StackedAreaChart<Number,Number> UI.SimStatisticsController.chart  [private]
```

A kirajzolandó grafikus

7.8.4.2. deaths

```
final XYChart.Series<Number,Number> UI.SimStatisticsController.deaths = new XYChart.Series<>()
[private]
```

halálozási adatokat tároló XYChart Series

7.8.4.3. heals

```
final XYChart.Series<Number,Number> UI.SimStatisticsController.heals = new XYChart.Series<>()
[private]
```

gyógyulási adatokat tároló XYChart Series

7.8.4.4. infections

```
final XYChart.Series<Number,Number> UI.SimStatisticsController.infections = new XYChart.↩
Series<>()  [private]
```

fertőzési adatokat tároló XYChart Series

7.8.4.5. population

```
final XYChart.Series<Number,Number> UI.SimStatisticsController.population = new XYChart.↩
Series<>()  [private]
```

lakossági adatokat tároló XYChart Series

7.8.4.6. sss

```
SimulationStatisticsStore UI.SimStatisticsController.sss  [package]
```

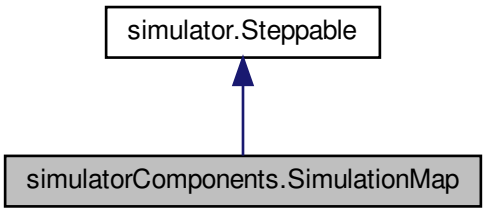
Közös store a simulationPlayerrel

Ez a dokumentáció az osztályról a következő fájl alapján készült:

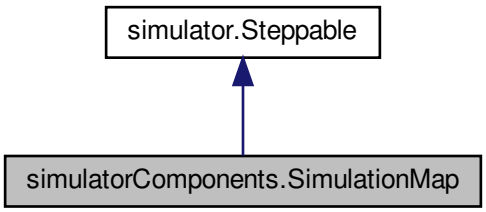
- SimStatisticsController.java

7.9. simulatorComponents.SimulationMap osztályreferencia

A simulatorComponents.SimulationMap osztály származási diagramja:



A simulatorComponents.SimulationMap osztály együttműködési diagramja:



Publikus tagfüggvények

- SimulationMap (Canvas c)
- void step (Canvas c)
- void init (Canvas c)
- boolean isCollidedWith (Steppable st)
- boolean isCollidedWith (Dot dot)
- void hitBy (Dot dot)
- boolean isOutOfWindow (Canvas c)
- void moveBack (Canvas c)
- void refresh (Canvas c)
- void draw (Canvas c)

7.9.1. Részletes leírás

SimulationMap osztály A pályát, jelképezi. Feladata, hogy kör elején letörli magát

7.9.2. Konstruktorok és destruktorok dokumentációja

7.9.2.1. SimulationMap()

```
simulatorComponents.SimulationMap.SimulationMap (
            Canvas c )
```

SimulationMap konstruktorja, meghívja a SimulationMap init függvényét

Paraméterek

c	A kapott canvas

7.9.3. Tagfüggvények dokumentációja

7.9.3.1. draw()

```
void simulatorComponents.SimulationMap.draw (
            Canvas c )
```

Letörli a Canvas-t

Paraméterek

c	A kapott Canvas

Megvalósítja a következőket: simulator.Steppable.

7.9.3.2. hitBy()

```
void simulatorComponents.SimulationMap.hitBy (
            Dot dot )
```

Kezeli, hogy mi történik, ha egy Dot eltalálja. Semmi, mert nem tud egy Dot Pályával ütközni, de a léptethetőség miatt szükséges.

Paraméterek

dot	A kapott Dot

Megvalósítja a következőket: simulator.Steppable.

7.9.3.3. init()

```
void simulatorComponents.SimulationMap.init (
            Canvas c )
```

Inicializálja a SimulationMap-et (Tehát letörli magát)

Paraméterek

c	A kapott Canvas

Megvalósítja a következőket: simulator.Steppable.

7.9.3.4. isCollidedWith() [1/2]

```
boolean simulatorComponents.SimulationMap.isCollidedWith (
            Dot dot )
```

Megvizsgálja, hogy tudott-e ütközni egy Dot-al. Mindig hamisat ad vissza, mert a pálya nem ütközik, hanem a háttér szerepét tölti be.

Paraméterek

dot	A kapott Dot

Visszatérési érték

Mindig Hamis

Megvalósítja a következőket: simulator.Steppable.

7.9.3.5. isCollidedWith() [2/2]

```
boolean simulatorComponents.SimulationMap.isCollidedWith (
            Steppable st )
```

Megvizsgálja, hogy tudott-e ütközni egy másik Steppable-el. Mindig hamisat ad vissza, mert a pálya nem ütközik, hanem a háttér szerepét tölti be.

Paraméterek

st	A kapott Másik Steppable

Visszatérési érték

Mindig Hamis

Megvalósítja a következőket: simulator.Steppable.

7.9.3.6. isOutOfWindow()

```
boolean simulatorComponents.SimulationMap.isOutOfWindow (
            Canvas c )
```

Megvizsgálja, hogy aza ablakon kívül esik-e. A háttér nem tud az ablakon kívül esni. A léptethetőség miatt szükséges.

Paraméterek

c	A kapott Canvas

Visszatérési érték

Mindig Hamis

Megvalósítja a következőket: simulator.Steppable.

7.9.3.7. moveBack()

```
void simulatorComponents.SimulationMap.moveBack (
            Canvas c )
```

Visszahúzza az objektumat a Canvas-ra, A léptethetőség miatt szükséges.

Paraméterek

c	A kapott Canvas

Megvalósítja a következőket: simulator.Steppable.

7.9.3.8. refresh()

```
void simulatorComponents.SimulationMap.refresh (
            Canvas c )
```

Frissíti a Canvas-t. Meghívja a draw(Canvas)-t

Paraméterek

c	A kapott Canvas

Megvalósítja a következőket: simulator.Steppable.

7.9.3.9. step()

```
void simulatorComponents.SimulationMap.step (
            Canvas c )
```

Lépés során meghívja saját maga refresh függvényét.

Paraméterek

c	A kapott Canvas

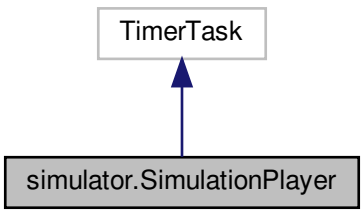
Megvalósítja a következőket: simulator.Steppable.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

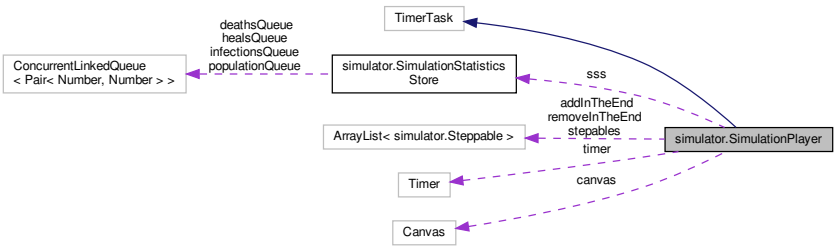
- SimulationMap.java

7.10. simulator.SimulationPlayer osztályreferencia

A simulator.SimulationPlayer osztály származási diagramja:



A simulator.SimulationPlayer osztály együttműködési diagramja:



Publikus tagfüggvények

- SimulationPlayer (SimulationTemplate sim, Canvas canvas, SimulationStatisticsStore sss)
- void moveDotsFromOutOfWindow (Canvas img)
- void refresh (Canvas c)
- void run ()
- void changePlayAndPause ()
- void forwardOneStep ()
- void exit ()
- void speedUp ()
- void speedDown ()

Statikus publikus tagfüggvények

- static int getIncubationPeriod ()
- static int getRemoveTime ()
- static void removeSteppable (Steppable st)
- static void addSteppable (Steppable st)
- static ArrayList< Steppable > getRemove ()
- static void addInfectedDot ()
- static void addHealedDot ()
- static void addDeadDot ()

Csomag attribútumok

- SimulationStatisticsStore sss
- int minPeriod
- Timer timer
- boolean paused
- Canvas canvas
- int currTick
- int millisecondsElapsed
- int sendDataPeriod = 10

Statikus csomag attribútumok

- static ArrayList< Steppable > stepables
- static ArrayList< Steppable > removeInTheEnd
- static ArrayList< Steppable > addInTheEnd
- static int infectedCnt
- static int deadCnt
- static int healedCnt
- static int neutralCnt
- static int oneTickInMs

Privát tagfüggvények

- void currTickIncrease ()
- void sendData ()

7.10.1. Részletes leírás

SimulationPlayer osztály. Egy szimuláció lejátszásához szükséges adatokat és függvényeket tárolja.

7.10.2. Konstruktorok és destruktorok dokumentációja

7.10.2.1. SimulationPlayer()

```
simulator.SimulationPlayer.SimulationPlayer (
            SimulationTemplate sim,
            Canvas canvas,
            SimulationStatisticsStore sss )
```

SimulationPlayer konstruktora

Paraméterek

sim	A kapott SimulationTemplate
canvas	A kapott Canvas
sss	A kapott statisztika tároló, ami pufferként funkcionál

7.10.3. Tagfüggvények dokumentációja

7.10.3.1. addDeadDot()

```
static void simulator.SimulationPlayer.addDeadDot ( )  [static]
```

Új fertőzött esetén növeli a halottak és csökkenti a fertőző Dotok számát

7.10.3.2. addHealedDot()

```
static void simulator.SimulationPlayer.addHealedDot ( )  [static]
```

Új gyógyult esetén növeli a gyógyultakat és csökkenti a fertőző Dotok számát

7.10.3.3. addInfectedDot()

```
static void simulator.SimulationPlayer.addInfectedDot ( )  [static]
```

Új fertőzött esetén növeli a fertőzötteket és csökkenti a semleges Dotok számát

7.10.3.4. addSteppable()

```
static void simulator.SimulationPlayer.addSteppable (
            Steppable st )  [static]
```

Hozzáadja a Steppable dolgot a hozzáadandóak listájához

Paraméterek

st	A hozzáadandó Steppable

7.10.3.5. changePlayAndPause()

```
void simulator.SimulationPlayer.changePlayAndPause ( )
```

negálja a paused változó értékét

7.10.3.6. currTickIncrease()

```
void simulator.SimulationPlayer.currTickIncrease ( )  [private]
```

Elküldti az adatokat a SimulationStatisticsStore-nak, majd lépteti a kört

7.10.3.7. exit()

```
void simulator.SimulationPlayer.exit ( )
```

Kilépéskor lezárja és üríti a szükséges dolgokat

7.10.3.8. forwardOneStep()

```
void simulator.SimulationPlayer.forwardOneStep ( )
```

Egy lépést szimulál le. Először mindenki lép. Utána az ütközések jönnek, majd eltávolítjuk és hozzáadjuk a szükséges Steppable-öket, majd Frissítjük a Canvast. Végül a jelentlegi Tick növelését végezzük.

7.10.3.9. getIncubationPeriod()

```
static int simulator.SimulationPlayer.getIncubationPeriod ( )  [static]
```

Lappangási időt visszadó függvény

Visszatérési érték

Lappangási idő Tickekben mérve

7.10.3.10. getRemove()

```
static ArrayList<Steppable> simulator.SimulationPlayer.getRemove ( )  [static]
```

Viszzadja az eltávolítandóak listáját

Visszatérési érték

Az eltávolítandóak listája

7.10.3.11. getRemoveTime()

```
static int simulator.SimulationPlayer.getRemoveTime ( )  [static]
```

Halott Dot eltűnési idejét határozza meg

Visszatérési érték

Eltűnési idő Tickekben mérve

7.10.3.12. moveDotsFromOutOfWindow()

```
void simulator.SimulationPlayer.moveDotsFromOutOfWindow (
            Canvas img )
```

A Canvasról kilógó dotokat visszarakja a Canvas-ra

Paraméterek

img	A kapott Canvas

7.10.3.13. refresh()

```
void simulator.SimulationPlayer.refresh (
            Canvas c )
```

Frissíti a teljes szimuláció tartalmát Az első Steppable mindig a pálya maga

Paraméterek

c	A kapott Canvas

7.10.3.14. removeSteppable()

```
static void simulator.SimulationPlayer.removeSteppable (
            Steppable st )  [static]
```

Hozzáadja a Steppable dolgot az eltávolítandók listájához

Paraméterek

st	Az eltávolítandó Steppable

7.10.3.15. run()

```
void simulator.SimulationPlayer.run ( )
```

Timer hívja minPeriod időközönként. Ha nincs szüneteltetve, akkor oneTickInMs ms-ként megtesz egy lépést

7.10.3.16. sendData()

```
void simulator.SimulationPlayer.sendData ( )  [private]
```

Minden sendDataPeriod-ban elküldi a SimulationStatisticsStore-nak az éppen aktuális adatokat

7.10.3.17. speedDown()

```
void simulator.SimulationPlayer.speedDown ( )
```

Lassítja a felhasználó által érzett idő telését

7.10.3.18. speedUp()

```
void simulator.SimulationPlayer.speedUp ( )
```

Gyorsítja a felhasználó által érzett idő telését

7.10.4. Adattagok dokumentációja

7.10.4.1. addInTheEnd

```
ArrayList<Steppable> simulator.SimulationPlayer.addInTheEnd  [static], [package]
```

Olyan léptethető dolgok tárolója, amit hozzá kell adni a léptethető dolgok közé a Tick végén

7.10.4.2. canvas

```
Canvas simulator.SimulationPlayer.canvas  [package]
```

A canvas, amire rajzolunk

7.10.4.3. currTick

```
int simulator.SimulationPlayer.currTick  [package]
```

Éppen aktuális kör sorszáma

7.10.4.4. deadCnt

```
int simulator.SimulationPlayer.deadCnt  [static], [package]
```

Éppen halott Dotok száma

7.10.4.5. healedCnt

```
int simulator.SimulationPlayer.healedCnt  [static], [package]
```

Éppen egészségese Dotok száma

7.10.4.6. infectedCnt

```
int simulator.SimulationPlayer.infectedCnt  [static], [package]
```

Éppen fertőző Dotok száma

7.10.4.7. millisecondsElapsed

```
int simulator.SimulationPlayer.millisecondsElapsed  [package]
```

Szimuláció kezdete óta eltelt idő

7.10.4.8. minPeriod

```
int simulator.SimulationPlayer.minPeriod  [package]
```

Két kör között eltelt minimális periódusidő

7.10.4.9. neutralCnt

```
int simulator.SimulationPlayer.neutralCnt  [static], [package]
```

Éppen semleges Dotok száma

7.10.4.10. oneTickInMs

```
int simulator.SimulationPlayer.oneTickInMs  [static], [package]
```

Egy Tick Ms-ban mérve

7.10.4.11. paused

```
boolean simulator.SimulationPlayer.paused  [package]
```

Le van-e szüneteltetve a szimuláció

7.10.4.12. removeInTheEnd

```
ArrayList<Steppable> simulator.SimulationPlayer.removeInTheEnd  [static], [package]
```

Olyan léptethető dolgok tárolója, amit el kell távolítani a steppables közül a Tick végén

7.10.4.13. sendDataPeriod

```
int simulator.SimulationPlayer.sendDataPeriod = 10  [package]
```

Adatküldés gyakorisága

7.10.4.14. sss

`SimulationStatisticsStore simulator.SimulationPlayer.sss [package]`

Szimuláció Statisztikát tároló osztály Pufferként funkcionál

7.10.4.15. stepables

`ArrayList<Steppable> simulator.SimulationPlayer.stepables [static], [package]`

Léptethető dolgok tárolója

7.10.4.16. timer

`Timer simulator.SimulationPlayer.timer [package]`

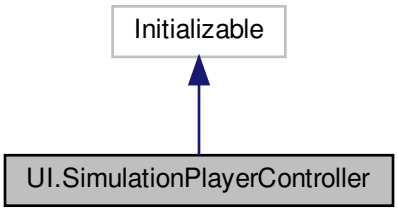
Időzítő, a körönként történő lépésért felel

Ez a dokumentáció az osztályról a következő fájl alapján készült:

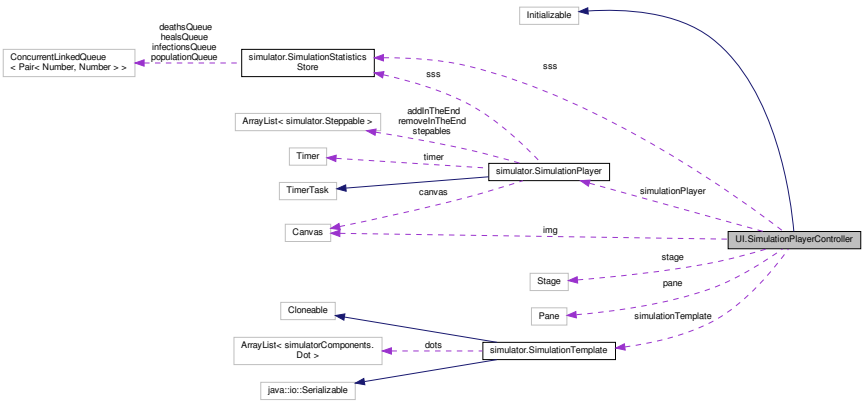
- SimulationPlayer.java

7.11. UI.SimulationPlayerController osztályreferencia

Az UI.SimulationPlayerController osztály származási diagramja:



Az UI.SimulationPlayerController osztály együttműködési diagramja:



Publikus tagfüggvények

- void initialize (URL url, ResourceBundle resourceBundle)
- void playAndPausePressed ()
- void stepPressed ()
- void statisticsPressed () throws IOException
- void speedUpPressed ()
- void speedDownPressed ()

Csomag függvények

- SimulationPlayerController (Stage st, SimulationTemplate sim)

Csomag attribútumok

- Stage stage
- SimulationPlayer simulationPlayer
- SimulationTemplate simulationTemplate
- SimulationStatisticsStore sss

Privát tagfüggvények

- void redraw ()

Privát attribútumok

- Canvas img
- Pane pane
- int reDrawCallCnt = 0

7.11.1. Részletes leírás

SimulationPlayer ablak kontroller osztája. Feladata, hogy kezelje az ablakkal történő User interakciókat

7.11.2. Konstruktorok és destruktorok dokumentációja

7.11.2.1. SimulationPlayerController()

```
UI.SimulationPlayerController.SimulationPlayerController (
            Stage st,
            SimulationTemplate sim )  [package]
```

SimulationPlayerController konstruktora

Paraméterek

st	A kapott Stage
sim	A kapott SimulationTemplate

7.11.3. Tagfüggvények dokumentációja

7.11.3.1. initialize()

```
void UI.SimulationPlayerController.initialize (
            URL url,
            ResourceBundle resourceBundle )
```

Az ablak inicializálója. Feladata az ablakon található elemek értékeinek beállítása.

Paraméterek

url	JavaFX használja relatív útvonal meghatározása a root objectnek
resourceBundle	Azok a források, amik a root object helyének meghatározásához szükségesek

7.11.3.2. playAndPausePressed()

```
void UI.SimulationPlayerController.playAndPausePressed ( )
```

playAndPause gomb megnyomásának kezelése

7.11.3.3. redraw()

```
void UI.SimulationPlayerController.redraw ( )  [private]
```

Az ablak újrarajzolása

7.11.3.4. speedDownPressed()

```
void UI.SimulationPlayerController.speedDownPressed ( )
```

speedDown gomb megnyomásnak kezelése

7.11.3.5. speedUpPressed()

```
void UI.SimulationPlayerController.speedUpPressed ( )
```

speedUp gomb megnyomásnak kezelése

7.11.3.6. statisticsPressed()

```
void UI.SimulationPlayerController.statisticsPressed ( ) throws IOException
```

Statistics gomb megnyomásának kezelése

Kivételek

IOException	kivételt dobhat, de ha az fxml fájl jó helyen van nem fog

7.11.3.7. stepPressed()

```
void UI.SimulationPlayerController.stepPressed ( )
```

Step gomb megnyomásának kezelése

7.11.4. Adattagok dokumentációja

7.11.4.1. img

```
Canvas UI.SimulationPlayerController.img  [private]
```

Canvas amire rajzolunk

7.11.4.2. pane

```
Pane UI.SimulationPlayerController.pane  [private]
```

Pane, ebben található a canvas, amire rajzolunk.

7.11.4.3. reDrawCallCnt

```
int UI.SimulationPlayerController.reDrawCallCnt = 0  [private]
```

Hányszor hívtuk meg a redraw függvényt. reDraw működéséhez szükséges

7.11.4.4. simulationPlayer

```
SimulationPlayer UI.SimulationPlayerController.simulationPlayer  [package]
```

simulationPlayer-t tárol

7.11.4.5. simulationTemplate

```
SimulationTemplate UI.SimulationPlayerController.simulationTemplate  [package]
```

Futatandó szimuláció adatait tárolja

7.11.4.6. sss

```
SimulationStatisticsStore UI.SimulationPlayerController.sss  [package]
```

Statisztika puffer tára

7.11.4.7. stage

```
Stage UI.SimulationPlayerController.stage  [package]
```

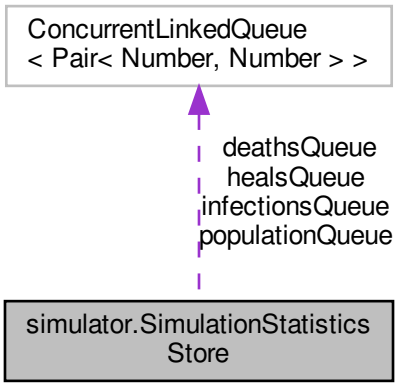
Stage eltárolása. Konstruktorban kapjuk az ablak létrehozása során. Ez alapján pozícionáljuk a többi ablakot.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- SimulationPlayerController.java

7.12. simulator.SimulationStatisticsStore osztályreferencia

A simulator.SimulationStatisticsStore osztály együttműködési diagramja:



Publikus tagfüggvények

- SimulationStatisticsStore ()
- void addPopulationChange (Number time, Number val)
- void addDeathsChange (Number time, Number val)
- void addInfectionChange (Number time, Number val)
- void addHealChange (Number time, Number val)
- ConcurrentLinkedQueue< Pair< Number, Number > > getInfectionsQueue ()
- ConcurrentLinkedQueue< Pair< Number, Number > > getPopulationQueue ()
- ConcurrentLinkedQueue< Pair< Number, Number > > getDeathsQueue ()
- ConcurrentLinkedQueue< Pair< Number, Number > > getHealsQueue ()
- void clearInfectionsQueue ()
- void clearPopulationQueue ()
- void clearDeathsQueue ()
- void clearHealsQueue ()
- void clearAll ()

Csomag attribútumok

- ConcurrentLinkedQueue< Pair< Number, Number > > populationQueue = new ConcurrentLinked↩ Queue<>()
- ConcurrentLinkedQueue< Pair< Number, Number > > deathsQueue = new ConcurrentLinkedQueue<>()
- ConcurrentLinkedQueue< Pair< Number, Number > > infectionsQueue = new ConcurrentLinked↩ Queue<>()
- ConcurrentLinkedQueue< Pair< Number, Number > > healsQueue = new ConcurrentLinkedQueue<>()

7.12.1. Részletes leírás

Puffer tároló a stasztikai adatok tárolására

7.12.2. Konstruktorok és destruktorok dokumentációja

7.12.2.1. SimulationStatisticsStore()

```
simulator.SimulationStatisticsStore.SimulationStatisticsStore ( )
```

SimulationStatisticsStore konstruktor

7.12.3. Tagfüggvények dokumentációja

7.12.3.1. addDeathsChange()

```
void simulator.SimulationStatisticsStore.addDeathsChange (
            Number time,
            Number val )
```

Hozzáad egy új értéket a deathsQueue-hoz

Paraméterek

time	Időbélyeg (Tick-ben)
val	Az aktuális érték

7.12.3.2. addHealChange()

```
void simulator.SimulationStatisticsStore.addHealChange (
            Number time,
            Number val )
```

Hozzáad egy új értéket a healsQueue-hoz

Paraméterek

time	Időbélyeg (Tick-ben)
val	Az aktuális érték

7.12.3.3. addInfectionChange()

```
void simulator.SimulationStatisticsStore.addInfectionChange (
            Number time,
            Number val )
```

Hozzáad egy új értéket a infectionsQueue-hoz

Paraméterek

time	Időbélyeg (Tick-ben)
val	Az aktuális érték

7.12.3.4. addPopulationChange()

```
void simulator.SimulationStatisticsStore.addPopulationChange (
            Number time,
            Number val )
```

Hozzáad egy új értéket a populationQueue-hoz

Paraméterek

time	Időbélyeg (Tick-ben)
val	Az aktuális érték

7.12.3.5. clearAll()

```
void simulator.SimulationStatisticsStore.clearAll ( )
```

Üríti az összes Queue-t

7.12.3.6. clearDeathsQueue()

```
void simulator.SimulationStatisticsStore.clearDeathsQueue ( )
```

Üríti a deathsQueue-t

7.12.3.7. clearHealsQueue()

```
void simulator.SimulationStatisticsStore.clearHealsQueue ( )
```

Üríti a healsQueue-t

7.12.3.8. clearInfectionsQueue()

```
void simulator.SimulationStatisticsStore.clearInfectionsQueue ( )
```

Üríti az infectionsQueue-t

7.12.3.9. clearPopulationQueue()

```
void simulator.SimulationStatisticsStore.clearPopulationQueue ( )
```

Üríti a populationQueue-t

7.12.3.10. getDeathsQueue()

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.getDeaths↩
Queue ( )
```

Visszadja a deathsQueue-t

Visszatérési érték

deathsQueue

7.12.3.11. getHealsQueue()

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.getHeals↩
Queue ( )
```

Visszadja a healsQueue-t

Visszatérési érték

healsQueue

7.12.3.12. getInfectionsQueue()

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.getInfections↩
Queue ( )
```

Visszadja az infectionsQueue-t

Visszatérési érték

infectionsQueue

7.12.3.13. getPopulationQueue()

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.getPopulation↩
Queue ( )
```

Visszadja a populationQueue-t

Visszatérési érték

populationQueue

7.12.4. Adattagok dokumentációja

7.12.4.1. deathsQueue

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.deathsQueue
= new ConcurrentLinkedQueue<>()  [package]
```

halálozási adatok puffere

7.12.4.2. healsQueue

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.healsQueue =
new ConcurrentLinkedQueue<>()  [package]
```

gyógyulási adatok puffere

7.12.4.3. infectionsQueue

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.infections↩
Queue = new ConcurrentLinkedQueue<>()  [package]
```

fertőzési adatok puffere

7.12.4.4. populationQueue

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.population↩
Queue = new ConcurrentLinkedQueue<>()  [package]
```

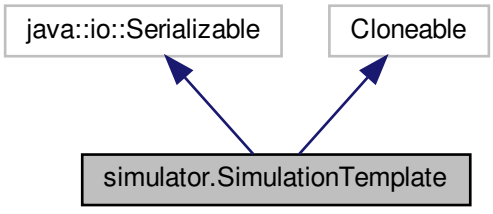
population adatok puffere. Azért erre esett a választásom, mert így a Concurrent Modification Exception-t el tudom kerülni, mert történthet olyan, hogy éppen olvasom a Puffert, mikör a végére már kerül az új adat

Ez a dokumentáció az osztályról a következő fájl alapján készült:

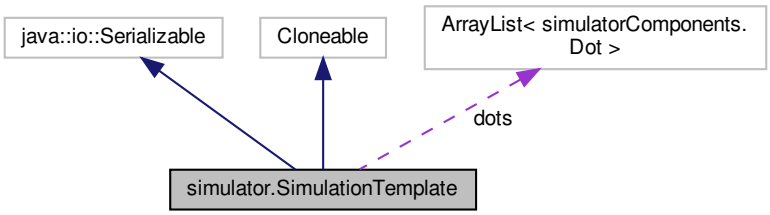
- SimulationStatisticsStore.java

7.13. simulator.SimulationTemplate osztályreferencia

A simulator.SimulationTemplate osztály származási diagramja:



A simulator.SimulationTemplate osztály együttműködési diagramja:



Publikus tagfüggvények

- SimulationTemplate ()
- SimulationTemplate (SimulationTemplate st)
- Object clone () throws CloneNotSupportedException
- double getInfChance ()
- void setInfection (double inf)
- double getMortChance ()
- void setMortality (double mort)
- double getHealChance ()
- void setHealChance (double heal)
- double getSpeedOfDot ()
- void setSpeed (double speed)
- ArrayList< Dot > getDots ()
- void addDot (Dot d)
- void createDot (dotTypes type, double x, double y, double r)
- void refresh (Canvas c)

Privát attribútumok

- ArrayList< Dot > dots
- double infChance
- double mortChance
- double healChance
- double speedOfDot

7.13.1. Részletes leírás

SimulationTemplate osztály. Szimuláció elindításához kapcsolatos adatokat tárol. Megvalósítja a java.io.Serializable és Cloneable interfészeket

7.13.2. Konstruktorok és destruktorok dokumentációja

7.13.2.1. SimulationTemplate() [1/2]

```
simulator.SimulationTemplate.SimulationTemplate ( )
```

SimulationTemplate konstruktura Beállítja az alapértelmezett értékeket és létrehozza az objektumot.

7.13.2.2. SimulationTemplate() [2/2]

```
simulator.SimulationTemplate.SimulationTemplate (
            SimulationTemplate st )
```

Simulation template copy konstruktora

Paraméterek

st	a másolandó SimulationTemplate

7.13.3. Tagfüggvények dokumentációja

7.13.3.1. addDot()

```
void simulator.SimulationTemplate.addDot (
            Dot d )
```

Hozzáad egy Dot-ot a Dotokat tartalmazó listához

Paraméterek

d	A kapott Dot

7.13.3.2. clone()

```
Object simulator.SimulationTemplate.clone ( ) throws CloneNotSupportedException
```

clone függvény felüldefiniálása. Célja, hogy a SimulationTemplate másolását megvalósítsa.

Visszatérési érték

Visszadja az objektum másolatát(deep copy)

Kivételek

CloneNotSupportedException	kivételt dobhat (de nem fog, mert a dot és a SimulationTemplate is klónozható)

7.13.3.3. createDot()

```
void simulator.SimulationTemplate.createDot (
            dotTypes type,
            double x,
            double y,
            double r )
```

Létrehoz a megadott paraméterek alapján egy Dot-ot és hozzáadja a Dot listához

Paraméterek

type	A létrehozandó Dot típusa
x	A létrehozandó Dot x kordinátája
y	A létrehozandó Dot y kordinátája
r	A létrehozandó Dot sugara

7.13.3.4. getDots()

```
ArrayList<Dot> simulator.SimulationTemplate.getDots ( )
```

Dot lista getter függvénye

Visszatérési érték

a dotokat tartalmazó Array list

7.13.3.5. getHealChance()

```
double simulator.SimulationTemplate.getHealChance ( )
```

Gyógyulási esély getter függvénye

Visszatérési érték

A gyógyulási esély

7.13.3.6. getInfChance()

```
double simulator.SimulationTemplate.getInfChance ( )
```

Fertőzési esély getter függvénye

Visszatérési érték

A fertőzési esély

7.13.3.7. getMortChance()

```
double simulator.SimulationTemplate.getMortChance ( )
```

Halálozási esély getter függvénye

Visszatérési érték

A kapott halálozási esély

7.13.3.8. getSpeedOfDot()

```
double simulator.SimulationTemplate.getSpeedOfDot ( )
```

A sebesség getter függvénye

Visszatérési érték

A sebessége a Dot-nak

7.13.3.9. refresh()

```
void simulator.SimulationTemplate.refresh (
            Canvas c )
```

Frissíti a Canvas tartalmát

Paraméterek

c	A kapott canvas

7.13.3.10. setHealChance()

```
void simulator.SimulationTemplate.setHealChance (
            double heal )
```

Gyógyulási esély setter függvénye Beállítja a kapott gyógyulási esélyt a jövőben létrejövő Dot-okra.

Paraméterek

heal	A kapott gyógyulási esély

7.13.3.11. setInfection()

```
void simulator.SimulationTemplate.setInfection (
            double inf )
```

Fertőzési esély setter függvénye Beállítja a kapott fertőzési esélyt a jövőben létrejövő Dot-okra.

Paraméterek

inf	A kapott fertőzési esély

7.13.3.12. setMortality()

```
void simulator.SimulationTemplate.setMortality (
            double mort )
```

Halálozási esély setter függvénye Beállítja a kapott halálozási esélyt a jövőben létrejövő Dot-okra.

Paraméterek

mort	A kapott halálozási esély

7.13.3.13. setSpeed()

```
void simulator.SimulationTemplate.setSpeed (
            double speed )
```

Sebesség esély setter függvénye Beállítja a kapott sebességet a jövőben létrejövő Dot-okra.

Paraméterek

speed	A kapott sebesség

7.13.4. Adattagok dokumentációja

7.13.4.1. dots

```
ArrayList<Dot> simulator.SimulationTemplate.dots  [private]
```

Dot-okat tároló ArrayList.

7.13.4.2. healChance

```
double simulator.SimulationTemplate.healChance  [private]
```

Gyógyulási esély

7.13.4.3. infChance

```
double simulator.SimulationTemplate.infChance  [private]
```

Átfertőzési esély

7.13.4.4. mortChance

```
double simulator.SimulationTemplate.mortChance  [private]
```

Halálozási esély

7.13.4.5. speedOfDot

```
double simulator.SimulationTemplate.speedOfDot  [private]
```

Dot sebessége

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- SimulationTemplate.java

7.14. Tests.SimulationTemplateTest osztályreferencia

Publikus tagfüggvények

- void createDot ()

7.14.1. Részletes leírás

Simulation Template osztály tesztje

7.14.2. Tagfüggvények dokumentációja

7.14.2.1. createDot()

```
void Tests.SimulationTemplateTest.createDot ( )
```

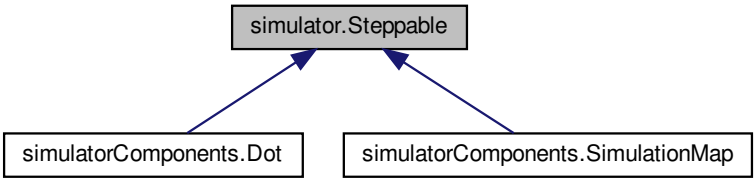
create Dot tesztje

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- SimulationTemplateTest.java

7.15. simulator.Steppable interfészreferencia

A simulator.Steppable osztály származási diagramja:



Publikus tagfüggvények

- void step (Canvas c)
- void init (Canvas c)
- boolean isCollidedWith (Steppable st)
- boolean isCollidedWith (Dot dot)
- void hitBy (Dot dot)
- boolean isOutOfWindow (Canvas c)
- void refresh (Canvas c)
- void draw (Canvas c)
- void moveBack (Canvas c)

7.15.1. Részletes leírás

Interfész, amely a léptethető dolgokat valósítja meg

7.15.2. Tagfüggvények dokumentációja

7.15.2.1. draw()

```
void simulator.Steppable.draw (
            Canvas c )
```

Kirajzolja a léptethető dolgot a Canvason

Paraméterek

c	A kapott Canvas

Megvalósítják a következők: simulatorComponents.SimulationMap és simulatorComponents.Dot.

7.15.2.2. hitBy()

```
void simulator.Steppable.hitBy (
            Dot dot )
```

Dottal való ütközést lekezelő függvény

Paraméterek

dot	A kapott Dot

Megvalósítják a következők: simulatorComponents.SimulationMap és simulatorComponents.Dot.

7.15.2.3. init()

```
void simulator.Steppable.init (
            Canvas c )
```

Szimuláció elején végzendő lépések

Paraméterek

c	A kapott Canvas

Megvalósítják a következők: simulatorComponents.SimulationMap és simulatorComponents.Dot.

7.15.2.4. isCollidedWith() [1/2]

```
boolean simulator.Steppable.isCollidedWith (
            Dot dot )
```

Ütközött-e a kapott Dot-al?

Paraméterek

dot	A kapott Másik Dot

Visszatérési érték

Igen vagy Nem

Megvalósítják a következők: simulatorComponents.SimulationMap és simulatorComponents.Dot.

7.15.2.5. isCollidedWith() [2/2]

```
boolean simulator.Steppable.isCollidedWith (
            Steppable st )
```

Ütközött-e a kapott Steppable-el?

Paraméterek

st	A kapott Másik Steppable

Visszatérési érték

Igen vagy Nem

Megvalósítják a következők: simulatorComponents.SimulationMap és simulatorComponents.Dot.

7.15.2.6. isOutOfWindow()

```
boolean simulator.Steppable.isOutOfWindow (
            Canvas c )
```

A léptethető dolog a Canvason kívül tartózkodik-e?

Paraméterek

c	A kapott Canvas

Visszatérési érték

Igen vagy Nem

Megvalósítják a következők: simulatorComponents.SimulationMap és simulatorComponents.Dot.

7.15.2.7. moveBack()

```
void simulator.Steppable.moveBack (
            Canvas c )
```

Visszahúzza a léptethető dolgot a Canvasra

Paraméterek

c	A kapott Canvas

Megvalósítják a következők: simulatorComponents.SimulationMap és simulatorComponents.Dot.

7.15.2.8. refresh()

```
void simulator.Steppable.refresh (
            Canvas c )
```

Frissíti a léptethető dolgot a Canvason

Paraméterek

c	A kapott Canvas

Megvalósítják a következők: simulatorComponents.SimulationMap és simulatorComponents.Dot.

7.15.2.9. step()

```
void simulator.Steppable.step (
            Canvas c )
```

Egy körben végzendő lépések

Paraméterek

c	A kapott Canvas

Megvalósítják a következők: simulatorComponents.SimulationMap és simulatorComponents.Dot.

Ez a dokumentáció az interfészről a következő fájl alapján készült:

- Steppable.java

Tárgymutató