

## Vírus szimuláció

Készítette Doxygen 1.9.0



<b>1. Prog3: Vírus szimuláció nagyházfeladat terv</b>	<b>1</b>
1.1. Az ötlet leírása . . . . .	1
<b>2. Programozói dokumentáció kiegészítés</b>	<b>3</b>
2.1. A programban található osztályok . . . . .	3
2.1.1. simulator package . . . . .	3
2.1.1.1. Simulation Player . . . . .	4
2.1.1.2. Simulation Statistics Store . . . . .	4
2.1.1.3. Simulation Template . . . . .	4
2.1.2. simulator components . . . . .	4
2.1.2.1. Dot . . . . .	4
2.1.2.2. dotTypes . . . . .	4
2.1.2.3. Point . . . . .	4
2.1.2.4. SimulationMap . . . . .	4
2.1.3. Tests . . . . .	4
2.1.3.1. Dot Test . . . . .	4
2.1.3.2. Point Test . . . . .	5
2.1.3.3. Simulation Template Test . . . . .	5
2.1.4. UI . . . . .	5
2.1.4.1. Main . . . . .	5
2.1.4.2. Sim Editor Controller . . . . .	5
2.1.4.3. Sim Statistics Controller . . . . .	5
2.1.4.4. Simulation Player Controller . . . . .	5
2.2. A programban található interfész . . . . .	5
2.2.1. Steppable . . . . .	5
2.3. Fordítási tudnivalók . . . . .	5
2.4. Egyéb tudnivalók . . . . .	5
<b>3. Prog3: Vírus szimuláció nagyházfeladat specifikáció</b>	<b>7</b>
3.1. Az ötlet leírása . . . . .	7
3.2. A program funkcionalitása a felhasználó szemszögéből . . . . .	7
3.3. Megoldási ötlet (vázlat) . . . . .	9
<b>4. Felhasználói dokumentáció</b>	<b>11</b>
4.1. A program általános leírása . . . . .	11
4.2. A program funkcionalitása a felhasználó szemszögéből . . . . .	11
4.3. A programban használt fájlkezelés . . . . .	14
<b>5. Hierarchikus mutató</b>	<b>15</b>
5.1. Osztályhierarchia . . . . .	15
<b>6. Osztálymutató</b>	<b>17</b>
6.1. Osztálylista . . . . .	17

<b>7. Osztályok dokumentációja</b>	<b>19</b>
7.1. simulatorComponents.Dot osztályreferencia	19
7.1.1. Részletes leírás	21
7.1.2. Konstruktorkok és destruktorkok dokumentációja	21
7.1.2.1. Dot() [1/3]	21
7.1.2.2. Dot() [2/3]	21
7.1.2.3. Dot() [3/3]	21
7.1.3. Tagfüggvények dokumentációja	22
7.1.3.1. bounceBack()	22
7.1.3.2. clone()	22
7.1.3.3. die()	23
7.1.3.4. draw()	23
7.1.3.5. drawCenters()	23
7.1.3.6. getLocation()	23
7.1.3.7. getRadius()	24
7.1.3.8. getType()	24
7.1.3.9. heal()	24
7.1.3.10. hitBy()	24
7.1.3.11. infectedBy()	25
7.1.3.12. init()	25
7.1.3.13. initVelocity()	25
7.1.3.14. isCollidedWith() [1/2]	26
7.1.3.15. isCollidedWith() [2/2]	26
7.1.3.16. isOutOfWindow()	26
7.1.3.17. moveBack()	27
7.1.3.18. refresh()	27
7.1.3.19. remove()	27
7.1.3.20. setHealChance()	27
7.1.3.21. setInfChance()	28
7.1.3.22. setLocation()	28
7.1.3.23. setMortChance()	28
7.1.3.24. step()	29
7.1.4. Adattagok dokumentációja	29
7.1.4.1. healChance	29
7.1.4.2. infChance	29
7.1.4.3. location	29
7.1.4.4. mass	29
7.1.4.5. mortChance	29
7.1.4.6. radius	30
7.1.4.7. sinceDead	30
7.1.4.8. sinceInfection	30
7.1.4.9. type	30

7.1.4.10. velocity	30
7.2. Tests.DotTest osztályreferencia	30
7.2.1. Részletes leírás	30
7.2.2. Tagfüggvények dokumentációja	31
7.2.2.1. isCollidedWith()	31
7.3. simulatorComponents.dotTypes felsoroló referencia	31
7.3.1. Részletes leírás	32
7.3.2. Adattagok dokumentációja	32
7.3.2.1. Dead	32
7.3.2.2. Healthy	32
7.3.2.3. Infectious	32
7.3.2.4. Neutral	32
7.3.2.5. None	32
7.4. UI.Main osztályreferencia	33
7.4.1. Részletes leírás	33
7.4.2. Tagfüggvények dokumentációja	33
7.4.2.1. main()	34
7.4.2.2. start()	34
7.5. simulatorComponents.Point osztályreferencia	34
7.5.1. Részletes leírás	35
7.5.2. Konstruktorkok és destruktorkok dokumentációja	36
7.5.2.1. Point() [1/2]	36
7.5.2.2. Point() [2/2]	36
7.5.3. Tagfüggvények dokumentációja	36
7.5.3.1. add() [1/2]	36
7.5.3.2. add() [2/2]	37
7.5.3.3. calcDisplacement()	37
7.5.3.4. calcDistance()	37
7.5.3.5. divide()	38
7.5.3.6. dotProduct()	38
7.5.3.7. getX()	38
7.5.3.8. getY()	38
7.5.3.9. isOutOfCanvas()	39
7.5.3.10. isOutOfCanvasBottom()	39
7.5.3.11. isOutOfCanvasLeft()	39
7.5.3.12. isOutOfCanvasRight()	40
7.5.3.13. isOutOfCanvasTop()	40
7.5.3.14. multiply()	40
7.5.3.15. subtract() [1/2]	41
7.5.3.16. subtract() [2/2]	41
7.5.4. Adattagok dokumentációja	41
7.5.4.1. x	41

7.5.4.2.	y	42
7.6.	Tests.PointTest osztályreferencia	42
7.6.1.	Részletes leírás	43
7.6.2.	Tagfüggvények dokumentációja	43
7.6.2.1.	add()	43
7.6.2.2.	calcDisplacement()	43
7.6.2.3.	calcDistance()	43
7.6.2.4.	divide()	43
7.6.2.5.	divideByZero()	43
7.6.2.6.	dotProduct()	43
7.6.2.7.	multiply()	44
7.6.2.8.	setUp()	44
7.6.2.9.	subtract()	44
7.6.3.	Adattagok dokumentációja	44
7.6.3.1.	p0	44
7.7.	UI.SimEditorController osztályreferencia	45
7.7.1.	Részletes leírás	46
7.7.2.	Konstruktorok és destruktorok dokumentációja	46
7.7.2.1.	SimEditorController()	46
7.7.3.	Tagfüggvények dokumentációja	47
7.7.3.1.	addManyDotsPressed()	47
7.7.3.2.	clearCanvas()	47
7.7.3.3.	createDotOnMousePosition()	47
7.7.3.4.	healSliderChanged()	47
7.7.3.5.	infSliderChanged()	47
7.7.3.6.	initialize()	48
7.7.3.7.	mortalitySliderChanged()	48
7.7.3.8.	openSerializedSimulationTemplate()	48
7.7.3.9.	openSimulationTemplate()	48
7.7.3.10.	redraw()	48
7.7.3.11.	serializeSimulationTemplate()	49
7.7.3.12.	setTypeOfDotOnMousePositionToDead()	49
7.7.3.13.	setTypeOfDotOnMousePositionToHealthy()	49
7.7.3.14.	setTypeOfDotOnMousePositionToInfectious()	49
7.7.3.15.	setTypeOfDotOnMousePositionToNeutral()	49
7.7.3.16.	speedSliderChanged()	49
7.7.3.17.	startSimulationPlayer()	49
7.7.3.18.	startSimulationPlayerFromFile()	50
7.7.4.	Adattagok dokumentációja	50
7.7.4.1.	healField	50
7.7.4.2.	healSlider	50
7.7.4.3.	img	50

7.7.4.4.	<a href="#">infField</a>	50
7.7.4.5.	<a href="#">infSlider</a>	51
7.7.4.6.	<a href="#">manyDotsComboBox</a>	51
7.7.4.7.	<a href="#">manyDotsField</a>	51
7.7.4.8.	<a href="#">mortField</a>	51
7.7.4.9.	<a href="#">mortSlider</a>	51
7.7.4.10.	<a href="#">pane</a>	51
7.7.4.11.	<a href="#">radius</a>	51
7.7.4.12.	<a href="#">selectedType</a>	51
7.7.4.13.	<a href="#">simulationTemplate</a>	52
7.7.4.14.	<a href="#">speedField</a>	52
7.7.4.15.	<a href="#">speedSlider</a>	52
7.7.4.16.	<a href="#">stage</a>	52
7.8.	<a href="#">UI.SimStatisticsController osztályreferencia</a>	52
7.8.1.	<a href="#">Részletes leírás</a>	53
7.8.2.	<a href="#">Konstruktorok és destruktorok dokumentációja</a>	54
7.8.2.1.	<a href="#">SimStatisticsController()</a>	54
7.8.3.	<a href="#">Tagfüggvények dokumentációja</a>	54
7.8.3.1.	<a href="#">initialize()</a>	54
7.8.3.2.	<a href="#">updateChart()</a>	54
7.8.4.	<a href="#">Adattagok dokumentációja</a>	54
7.8.4.1.	<a href="#">chart</a>	55
7.8.4.2.	<a href="#">deaths</a>	55
7.8.4.3.	<a href="#">heals</a>	55
7.8.4.4.	<a href="#">infections</a>	55
7.8.4.5.	<a href="#">population</a>	55
7.8.4.6.	<a href="#">sss</a>	55
7.9.	<a href="#">simulatorComponents.SimulationMap osztályreferencia</a>	56
7.9.1.	<a href="#">Részletes leírás</a>	56
7.9.2.	<a href="#">Konstruktorok és destruktorok dokumentációja</a>	57
7.9.2.1.	<a href="#">SimulationMap()</a>	57
7.9.3.	<a href="#">Tagfüggvények dokumentációja</a>	57
7.9.3.1.	<a href="#">draw()</a>	57
7.9.3.2.	<a href="#">hitBy()</a>	57
7.9.3.3.	<a href="#">init()</a>	58
7.9.3.4.	<a href="#">isCollidedWith() [1/2]</a>	58
7.9.3.5.	<a href="#">isCollidedWith() [2/2]</a>	58
7.9.3.6.	<a href="#">isOutOfWindow()</a>	59
7.9.3.7.	<a href="#">moveBack()</a>	59
7.9.3.8.	<a href="#">refresh()</a>	60
7.9.3.9.	<a href="#">step()</a>	60
7.10.	<a href="#">simulator.SimulationPlayer osztályreferencia</a>	60

7.10.1. Részletes leírás	62
7.10.2. Konstruktork és destruktorok dokumentációja	62
7.10.2.1. SimulationPlayer()	62
7.10.3. Tagfüggvények dokumentációja	62
7.10.3.1. addDeadDot()	62
7.10.3.2. addHealedDot()	63
7.10.3.3. addInfectedDot()	63
7.10.3.4. addSteppable()	63
7.10.3.5. changePlayAndPause()	63
7.10.3.6. currTickIncrease()	63
7.10.3.7. exit()	63
7.10.3.8. forwardOneStep()	64
7.10.3.9. getIncubationPeriod()	64
7.10.3.10. getRemove()	64
7.10.3.11. getRemoveTime()	64
7.10.3.12. moveDotsFromOutOfWindow()	64
7.10.3.13. refresh()	65
7.10.3.14. removeSteppable()	65
7.10.3.15. run()	65
7.10.3.16. sendData()	65
7.10.3.17. speedDown()	66
7.10.3.18. speedUp()	66
7.10.4. Adattagok dokumentációja	66
7.10.4.1. addInTheEnd	66
7.10.4.2. canvas	66
7.10.4.3. currTick	66
7.10.4.4. deadCnt	66
7.10.4.5. healedCnt	66
7.10.4.6. infectedCnt	67
7.10.4.7. millisecondsElapsed	67
7.10.4.8. minPeriod	67
7.10.4.9. neutralCnt	67
7.10.4.10. oneTickInMs	67
7.10.4.11. paused	67
7.10.4.12. removeInTheEnd	67
7.10.4.13. sendDataPeriod	67
7.10.4.14. sss	68
7.10.4.15. stepables	68
7.10.4.16. timer	68
7.11. UI.SimulationPlayerController osztályreferencia	68
7.11.1. Részletes leírás	69
7.11.2. Konstruktork és destruktorok dokumentációja	69



7.11.2.1. <code>SimulationPlayerController()</code>	69
7.11.3. Tagfüggvények dokumentációja	70
7.11.3.1. <code>initialize()</code>	70
7.11.3.2. <code>playAndPausePressed()</code>	70
7.11.3.3. <code>redraw()</code>	70
7.11.3.4. <code>speedDownPressed()</code>	70
7.11.3.5. <code>speedUpPressed()</code>	70
7.11.3.6. <code>statisticsPressed()</code>	71
7.11.3.7. <code>stepPressed()</code>	71
7.11.4. Adattagok dokumentációja	71
7.11.4.1. <code>img</code>	71
7.11.4.2. <code>pane</code>	71
7.11.4.3. <code>reDrawCallCnt</code>	71
7.11.4.4. <code>simulationPlayer</code>	71
7.11.4.5. <code>simulationTemplate</code>	72
7.11.4.6. <code>sss</code>	72
7.11.4.7. <code>stage</code>	72
7.12. <code>simulator.SimulationStatisticsStore</code> osztályreferencia	72
7.12.1. Részletes leírás	73
7.12.2. Konstruktorkok és destruktorkok dokumentációja	73
7.12.2.1. <code>SimulationStatisticsStore()</code>	73
7.12.3. Tagfüggvények dokumentációja	73
7.12.3.1. <code>addDeathsChange()</code>	73
7.12.3.2. <code>addHealChange()</code>	74
7.12.3.3. <code>addInfectionChange()</code>	74
7.12.3.4. <code>addPopulationChange()</code>	74
7.12.3.5. <code>clearAll()</code>	75
7.12.3.6. <code>clearDeathsQueue()</code>	75
7.12.3.7. <code>clearHealsQueue()</code>	75
7.12.3.8. <code>clearInfectionsQueue()</code>	75
7.12.3.9. <code>clearPopulationQueue()</code>	75
7.12.3.10. <code>getDeathsQueue()</code>	75
7.12.3.11. <code>getHealsQueue()</code>	76
7.12.3.12. <code>getInfectionsQueue()</code>	76
7.12.3.13. <code>getPopulationQueue()</code>	76
7.12.4. Adattagok dokumentációja	76
7.12.4.1. <code>deathsQueue</code>	76
7.12.4.2. <code>healsQueue</code>	77
7.12.4.3. <code>infectionsQueue</code>	77
7.12.4.4. <code>populationQueue</code>	77
7.13. <code>simulator.SimulationTemplate</code> osztályreferencia	77
7.13.1. Részletes leírás	78

7.13.2. Konstruktork és destruktork dokumentációja	78
7.13.2.1. SimulationTemplate() [1/2]	79
7.13.2.2. SimulationTemplate() [2/2]	79
7.13.3. Tagfüggvények dokumentációja	79
7.13.3.1. addDot()	79
7.13.3.2. clone()	79
7.13.3.3. createDot()	80
7.13.3.4. getDots()	80
7.13.3.5. getHealChance()	80
7.13.3.6. getInfChance()	81
7.13.3.7. getMortChance()	81
7.13.3.8. getSpeedOfDot()	81
7.13.3.9. refresh()	81
7.13.3.10. setHealChance()	82
7.13.3.11. setInfection()	82
7.13.3.12. setMortality()	82
7.13.3.13. setSpeed()	82
7.13.4. Adattagok dokumentációja	83
7.13.4.1. dots	83
7.13.4.2. healChance	83
7.13.4.3. infChance	83
7.13.4.4. mortChance	83
7.13.4.5. speedOfDot	83
7.14. Tests.SimulationTemplateTest osztályreferencia	84
7.14.1. Részletes leírás	84
7.14.2. Tagfüggvények dokumentációja	84
7.14.2.1. createDot()	84
7.15. simulator.Steppable interfészreferencia	84
7.15.1. Részletes leírás	85
7.15.2. Tagfüggvények dokumentációja	85
7.15.2.1. draw()	85
7.15.2.2. hitBy()	85
7.15.2.3. init()	85
7.15.2.4. isCollidedWith() [1/2]	86
7.15.2.5. isCollidedWith() [2/2]	86
7.15.2.6. isOutOfWindow()	87
7.15.2.7. moveBack()	87
7.15.2.8. refresh()	87
7.15.2.9. step()	88
<b>Tárgymutató</b>	<b>89</b>

## 1. fejezet

# Prog3: Vírus szimuláció nagyházfeladat terv

### 1.1. Az ötlet leírása

A program vírus terjedését szimulálja. A pálya egy téglalap lenne, ahol 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.



## Programozói dokumentáció kiegészítés

Az osztályok részletesebb leírása a forráskódban található javadoc commentben, illetve a mellékelt pdf-ben található.

#### **2.1.1.1. Simulation Player**

Egy szimuláció lejátszásához szükséges adatokat és függvényeket tárolja.

#### **2.1.1.2. Simulation Statistics Store**

Puffer tároló a statisztikai adatok tárolására.

#### **2.1.1.3. Simulation Template**

Egy szimuláció előkészítéséhez szükséges adatokat tárolja.

### **2.1.2. simulator components**

A szimulátorban felhasználható alkotóelemeket tartalmazza.

#### **2.1.2.1. Dot**

Pötty, ami egy embert reprezentál a pályán.

#### **2.1.2.2. dotTypes**

Enum, ami a pöttyök típusait tárolja

#### **2.1.2.3. Point**

A pályán lévő dolgok helyzetét, illetve bizonyos esetekben az origó és a pont közé rajzolható vectort valósítja meg. Az osztály tartalmaz a pontokon és vectorokon végezhető műveleteket is.

#### **2.1.2.4. SimulationMap**

A pálya hátterét reprezentálja. Legfontosabb feladata a Canvas letörlése.

### **2.1.3. Tests**

A programhoz tartozó Junit tesztek tartalmazza a package.

#### **2.1.3.1. Dot Test**

A Dotokra megírt tesztek tartalmazza.

#### 2.1.3.2. Point Test

A pontokra megírt tesztet tartalmazza.

#### 2.1.3.3. Simulation Template Test

A Simulation Template-re megírt tesztet tartalmazza.

### 2.1.4. UI

A grafikus megjelenítéssel kapcsolatos osztályokat tartalmazza. Az itt található osztályok mind javaFX-et használnak.

#### 2.1.4.1. Main

A program belépési pontja.

#### 2.1.4.2. Sim Editor Controller

Szimuláció sablonjának szerkesztéséhez szükséges függvényeket tartalmazza, amelyek a felhasználóval kommunikálnak.

#### 2.1.4.3. Sim Statistics Controller

A szimuláció megjelenítéséért felel. Közvetlen kapcsolatban áll a felhasználóval.

#### 2.1.4.4. Simulation Player Controller

A szimuláció lejátszásához szükséges függvényeket tárolja. Közvetlen kapcsolatban áll a felhasználóval.

## 2.2. A programban található interfész

### 2.2.1. Steppable

Ez az interfész valósítja meg az összes léptethető dolgot a játékban.

## 2.3. Fordítási tudnivalók

A program futtatásához szükség van a megfelelő javafx SDK-ra (javafx-sdk-11.0.2) [link](#). Illetve a teszteléshez Junit 4-re és a hozzá tartozó függőségekre (hamcrest-core-1.3.jar) A programhoz mellékeltem az eclipse projekt-fájlokat.

## 2.4. Egyéb tudnivalók

A program forrásainak készítése közben javadoc kommentet alkalmaztam. Az ebből Doxygen segítségével készített pdf fájlt mellékeltem a projekthez.

A programot az IntelliJ IDEA IDE-vel készítettem. Az ablakok létrehozásához SceneBuildert használtam.





## 3. fejezet

# Prog3: Vírus szimuláció nagyházfeladat specifikáció

### 3.1. Az ötlet leírása

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

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

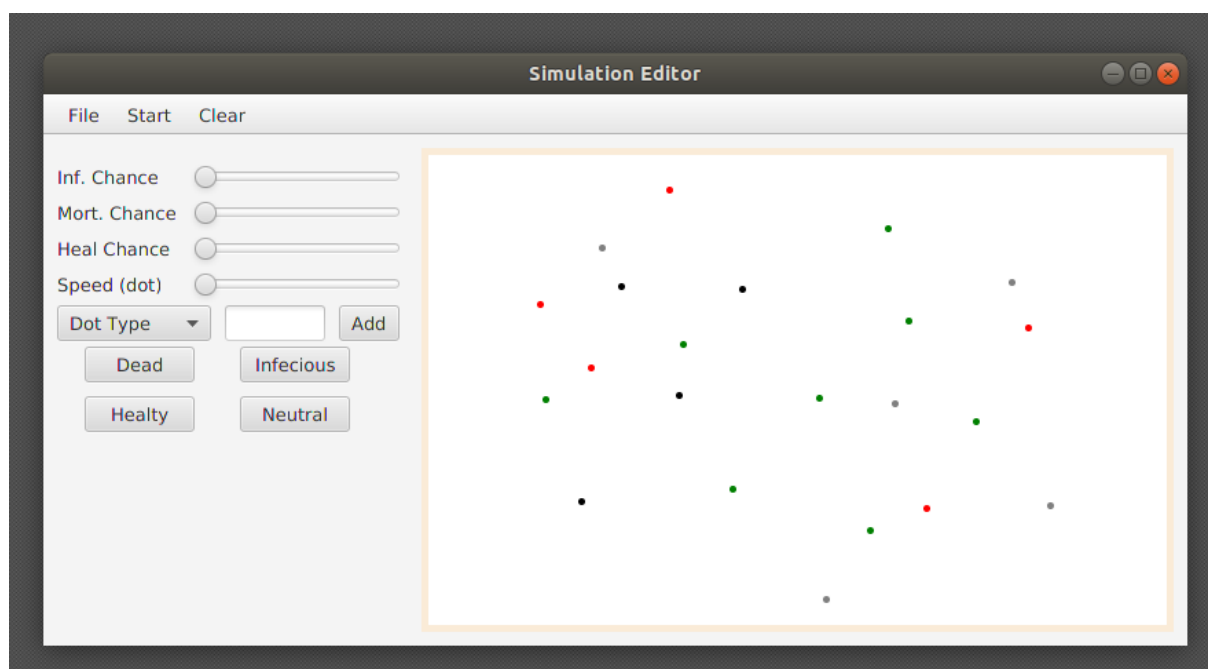
A különböző faktorokat csúszkákkal lehetne állítani: pl: pötty sebessége, milyen eséllyel fertőz, halálozási esély, gyógyulási idő stb. Ha a pötty falnak ütközik, vagy másik pöttyel, akkor visszapattan. A programhoz tartozik egy diagram is, ami a pöttyökről mutat statisztikát.

### 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 slider)
- n db pötty felhelyezése véletlenszerűen a pályára (Add button)
- pöttyök egyesével történő felhelyezése a pályára, egér kattintás alapján (4 db button)
- A pálya kezdeti értékeinek törlése (Clear)
- Szimuláció elindítása
  - Kezdeti értékek alapján(Start >Start)



1. ábra - Szimuláció előkészítése.

A Start menüpont megnyitásával megnyílik 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 után 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. (Fertőzöttek, 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 képlet fogja számolni:  $v = 2^{\left\lfloor \left\lfloor x \right\rfloor \right\rfloor}$ , 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 megfertőződött személyeket jelképezik
- Healthy (Egészséges): A fertőzött pöttyök képesek a gyógyulásra előre beállított eséllyel
- Dead (Halott): A fertőzött pöttyök képesek meghalni előre beállított eséllyel

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

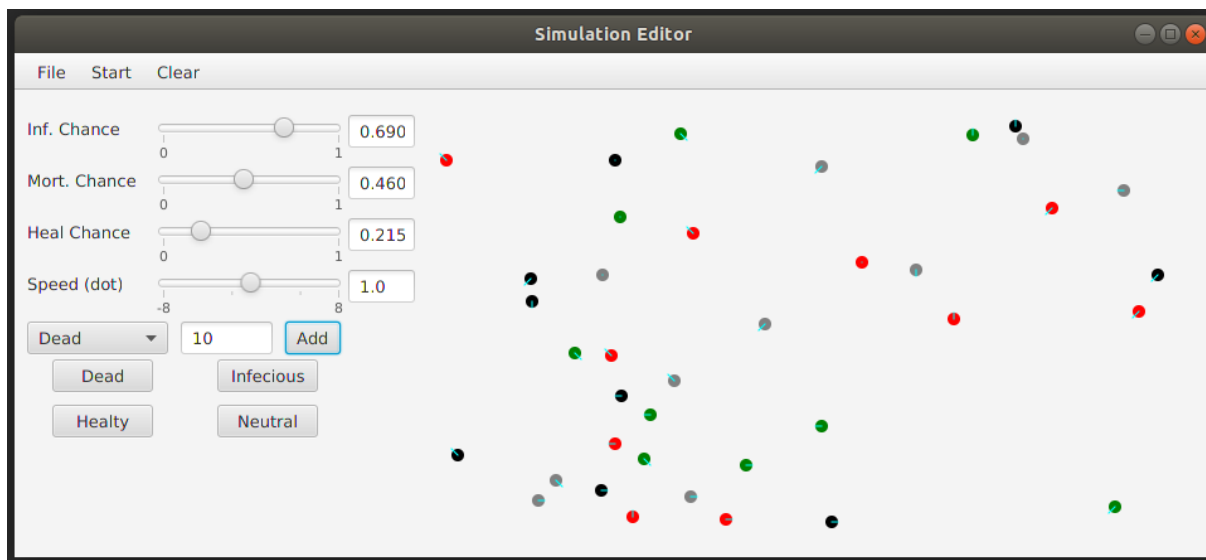
Egy fertőzött pötty másokat tovább tud fertőzni, illetve a lappangási idő letelte után (500 tick), lehetősége lesz gyógyulni, illetve meghalni a beállított valószínűség alapján.

### 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 funkciói:

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



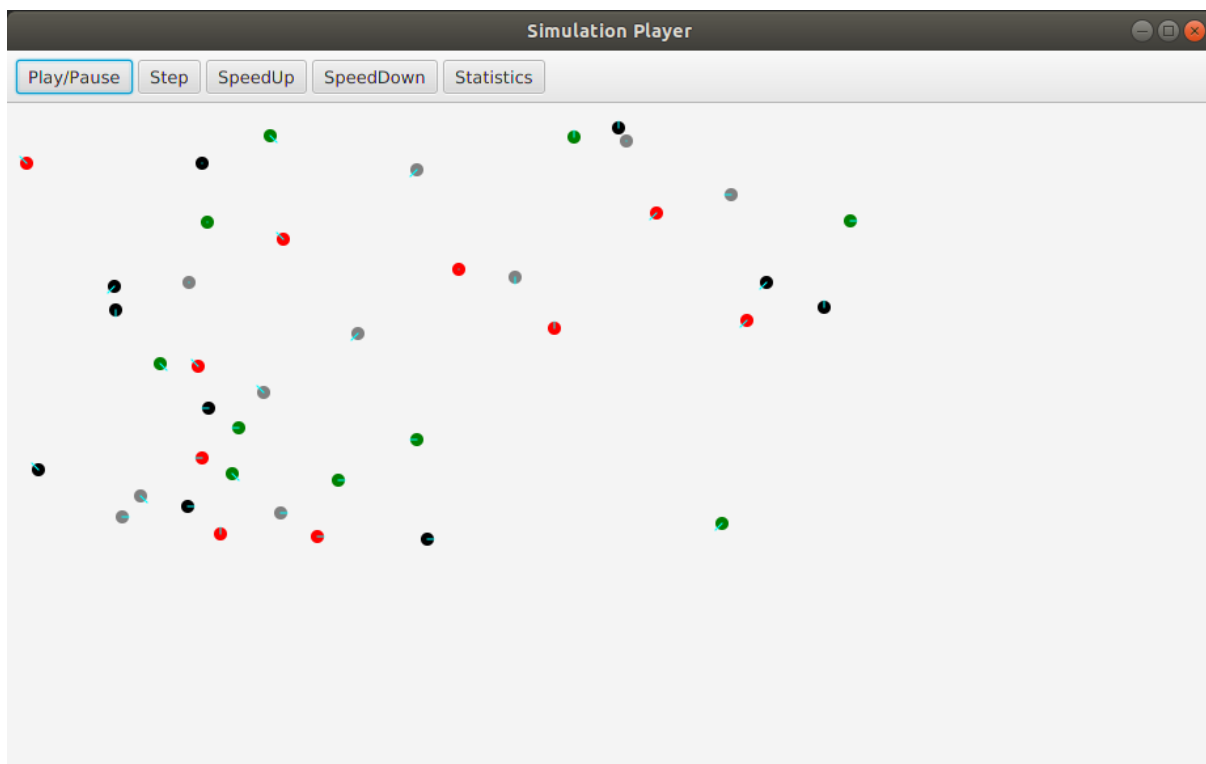
2.ábra - Szimuláció előkészítése.

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 egérgommbal kattintunk.

A kék vonal a sebesség vektort jelöli.

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

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



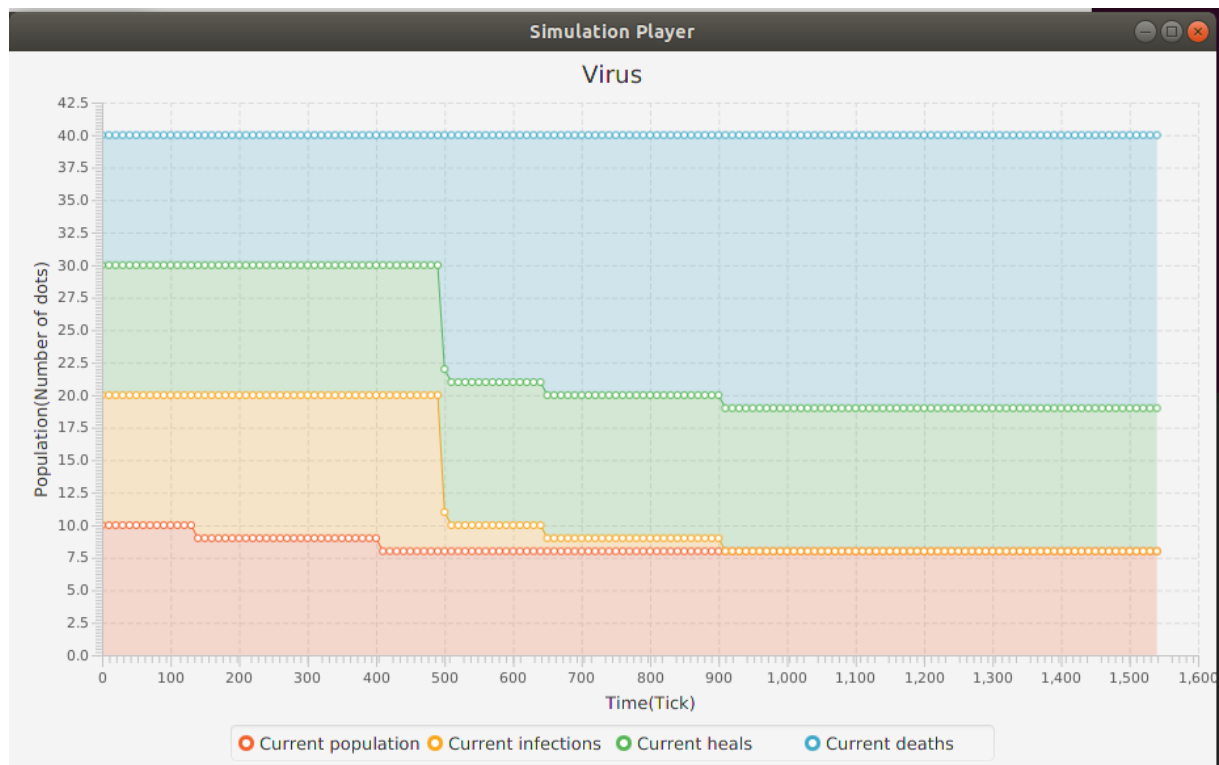
3.ábra - Vírus szimulálása

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 vagy lassabban az idő.

A Statistics gomb megnyomása után 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-enként frissül. Ha a statisztika bezárásra kerül, akkor az addig kapott adatok törlődnek.

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

Meghalás pillanatától számítva 150 tick múlva a halott pötty eltűnik.

3.ábra - Statisztika megjelenítése

A programrész funkciói:

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

### 4.3. A programban használt fájlkezelés

A szimuláció sablonjának fájlba történő mentésére és onnan való betöltésére van lehetőség. Ha rossz fájlt nyitunk meg, akkor a standard outputon hibaüzenetet kapunk. A szimuláció sablonjának betöltésére alkalmas fájlokra a felhasználónak kell emlékeznie.



## 5. fejezet

# Hierarchikus mutató

### 5.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

Cloneable	
simulator.SimulationTemplate . . . . .	77
simulatorComponents.Dot . . . . .	19
Tests.DotTest . . . . .	30
Tests.PointTest . . . . .	42
Serializable	
simulator.SimulationTemplate . . . . .	77
simulator.SimulationStatisticsStore . . . . .	72
Tests.SimulationTemplateTest . . . . .	84
simulator.Steppable . . . . .	84
simulatorComponents.Dot . . . . .	19
simulatorComponents.SimulationMap . . . . .	56
Application	
UI.Main . . . . .	33
Initializable	
UI.SimEditorController . . . . .	45
UI.SimStatisticsController . . . . .	52
UI.SimulationPlayerController . . . . .	68
Serializable	
simulatorComponents.Dot . . . . .	19
simulatorComponents.Point . . . . .	34
simulatorComponents.dotTypes . . . . .	31
TimerTask	
simulator.SimulationPlayer . . . . .	60



## 6. fejezet

# Osztálymutató

### 6.1. Osztálylista

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

<a href="#">simulatorComponents.Dot</a>	19
<a href="#">Tests.DotTest</a>	30
<a href="#">simulatorComponents.dotTypes</a>	31
<a href="#">UI.Main</a>	33
<a href="#">simulatorComponents.Point</a>	34
<a href="#">Tests.PointTest</a>	42
<a href="#">UI.SimEditorController</a>	45
<a href="#">UI.SimStatisticsController</a>	52
<a href="#">simulatorComponents.SimulationMap</a>	56
<a href="#">simulator.SimulationPlayer</a>	60
<a href="#">UI.SimulationPlayerController</a>	68
<a href="#">simulator.SimulationStatisticsStore</a>	72
<a href="#">simulator.SimulationTemplate</a>	77
<a href="#">Tests.SimulationTemplateTest</a>	84
<a href="#">simulator.Steppable</a>	84

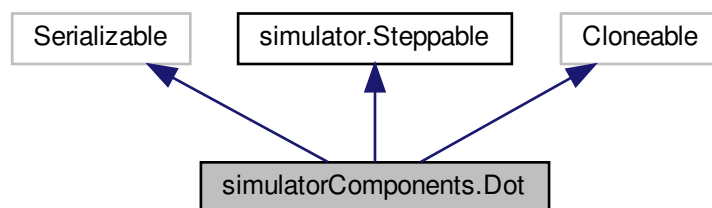


## 7. fejezet

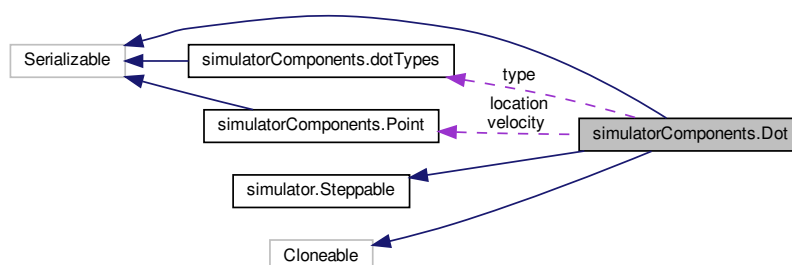
# Osztályok dokumentációja

### 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 interfeszeket.

### 7.1.2. Konstruktorok és destruktorok dokumentációja

#### 7.1.2.1. Dot() [1/3]

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

A potty konstruktora

Paraméterek

<i>x</i>	A potty x kordinataja
<i>y</i>	A potty y kordinataja
<i>r</i>	A potty sugara

#### 7.1.2.2. Dot() [2/3]

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

A potty konstruktora

Paraméterek

<i>x</i>	A potty x kordinataja
<i>y</i>	A potty y kordinataja
<i>r</i>	A potty sugara
<i>speed</i>	A potty 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 potty konstruktora

#### Paraméterek

<i>x</i>	A potty x koordinataja
<i>y</i>	A potty y koordinataja
<i>r</i>	A potty sugara
<i>speed</i>	A potty sebessége
<i>type</i>	A potty típusa
<i>infChance</i>	Masik pottyot ilyen eselyel fertoz, ha a <a href="#">Dot</a> fertozo
<i>mortChance</i>	A potty halalozasi eselye, ha mar megfertozodott es a virus lappangasi ideje lejart
<i>healChance</i>	A potty gyógyulási eselye, ha mar megfertozodott es a virus lappangasi ideje lejart

### 7.1.3. Tagfüggvények dokumentációja

#### 7.1.3.1. bounceBack()

```

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

```

A potty a Canvas szelen visszapattan

#### Paraméterek

<i>c</i>	A kapott Canvas
----------	-----------------

#### 7.1.3.2. clone()

```

Object simulatorComponents.Dot.clone ( ) throws CloneNotSupportedException

```

A potty által feluldefinialt clone metodus.

#### Visszatérési érték

A potty Object



## Kivételek

<i>CloneNotSupportedException</i>	kivetelt dobhat(De nem fog, mert a Dot szerIALIZALHATO)
-----------------------------------	---

**7.1.3.3. die()**

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

A potty meghal Feladatai: A potty típusának megváltoztatása, SimulationPlayer fele jelzi, hogy egy potty meghalt

**7.1.3.4. draw()**

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

A pottyot kirajzoló függvény. Kirajzolja a pottyot, majd rajzolja a sebesség vektorát

## Paraméterek

<i>c</i>	A kapott Canvas
----------	-----------------

Megvalósítja a következőket: [simulator.Steppable](#).

**7.1.3.5. drawCenters()**

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

Két potty közepe fölött megrajzolja a vektort. Debug celokra használtam, de nem töröltem.

## Paraméterek

<i>dot</i>	A másik <a href="#">Dot</a>
<i>c</i>	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 potty sugarának gettere

**Visszatérési érték**

A potty sugara

**7.1.3.8. getType()**

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

A potty tipusának gettere

**Visszatérési érték**

A potty típusa

**7.1.3.9. heal()**

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

A potty meggyógyul Feladatai: A potty tipusának megváltoztatása, SimulationPlayer fele jelzi, hogy egy potty meggyógyult

**7.1.3.10. hitBy()**

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

A potty másik pottyel való utkozese során hivodik. Ez kezeli a statikus (pl.: ket potty fedne egymast) es dinamikus utkozest (rugalmas utkozes)

**Paraméterek**

<i>d</i>	A másik potty
----------	---------------

Megvalósítja a következőket: [simulator.Steppable](#).

#### 7.1.3.11. infectedBy()

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

Fertozes bekovetkezese Feladata, hogy ennek a pottynek a megfelelo adatait beallitsa, a SimulationPlayer fele jelzi, hogy uj fertozes tortent

##### Paraméterek

<i>d</i>	A fertozest okozo potty
----------	-------------------------

#### 7.1.3.12. init()

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

A pottyot inicializalja: Ha a Canvason kívül van visszahuzza ot, majd kirajzolja

##### Paraméterek

<i>c</i>	A kapott Canvas
----------	-----------------

Megvalósítja a következőket: [simulator.Steppable](#).

#### 7.1.3.13. initVelocity()

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

Sebesseg vektor inicializalasa a kapott sebesseg alapjan. A fuggveny a lehetseges iranyt veletlenszeruen generalja.

##### Paraméterek

<i>speed</i>	A kapott sebesseg
--------------	-------------------

**7.1.3.14. isCollidedWith()** [1/2]

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

A potty utkozott-e a kapott steppable-el?

**Paraméterek**

<i>dot</i>	A kapott potty
------------	----------------

**Visszatérési érték**

Igen vagy Nem, a ket potty kozepai kozott levo tavolsag es a sugarak osszege alapjan.

Megvalósítja a következőket: [simulator.Steppable](#).

**7.1.3.15. isCollidedWith()** [2/2]

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

A potty utkozott-e a kapott steppable-el?

**Paraméterek**

<i>st</i>	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 potty a canvason kívül van-e

**Paraméterek**

<i>c</i>	A kapott Canvas
----------	-----------------

**Visszatérési érték**

Igen vagy Nem

Megvalósítja a következőket: [simulator.Steppable](#).

**7.1.3.17. moveBack()**

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

Visszahuzza a pottyot a Canvas területére, ha kiment belöle

**Paraméterek**

<i>c</i>	A kapott Canvas
----------	-----------------

Megvalósítja a következőket: [simulator.Steppable](#).

**7.1.3.18. refresh()**

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

A potty újrarajzolása

**Paraméterek**

<i>c</i>	A kapott Canvas
----------	-----------------

Megvalósítja a következőket: [simulator.Steppable](#).

**7.1.3.19. remove()**

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

A potty hozzáadása a kor vegeen eltávolítandók listájához

**7.1.3.20. setHealChance()**

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

Gyógyulási esély settere

## Paraméterek

<i>heal</i>	A kapott gyógyulási esély
-------------	---------------------------

**7.1.3.21. setInfChance()**

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

atfertozesei esely settere

## Paraméterek

<i>inf</i>	A kapott atfertozesi esely
------------	----------------------------

**7.1.3.22. setLocation()**

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

Location setterje

## Paraméterek

<i>location</i>	A kapott hely
-----------------	---------------

**7.1.3.23. setMortChance()**

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

Halalozasi esely settere

## Paraméterek

<i>mort</i>	A halalozasi gyógyulási esély
-------------	-------------------------------

#### 7.1.3.24. step()

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

Lepes fuggveny Feladatai: Lappangasi ido vizsgalata, Halalozas utan eltelt ido vizsgalata, Palya szelevel torteno utkozes, potty leptetese

##### 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 potty gyógyulási esélye

#### 7.1.4.2. infChance

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

A potty atfertőzésének esélye

#### 7.1.4.3. location

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

A potty közepének pozíciója

#### 7.1.4.4. mass

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

A potty tömege

#### 7.1.4.5. mortChance

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

A potty halálzási esélye

#### 7.1.4.6. radius

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

A potty 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 potty típusa

#### 7.1.4.10. velocity

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

A potty sebesség vektorának végpontja (Mint 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 tesztelese



## 7.2.2. Tagfüggvények dokumentációja

### 7.2.2.1. isCollidedWith()

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

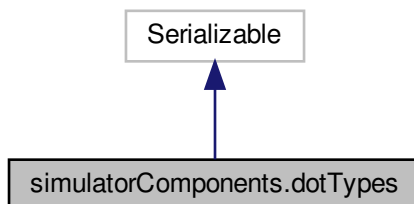
utkozes tesztelese

Ez a dokumentáció az osztályról a következő fájl alapján készült:

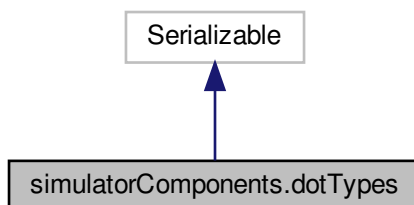
- DotTest.java

## 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 pottyok típusát tartja. 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 meggyógyulhat

#### 7.3.2.4. Neutral

```
simulatorComponents.dotTypes.Neutral
```

Semleges: Sima személy

#### 7.3.2.5. None

```
simulatorComponents.dotTypes.None
```

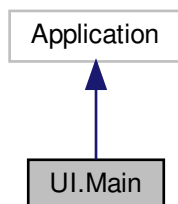
Nincs: hiba kezelés miatt

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

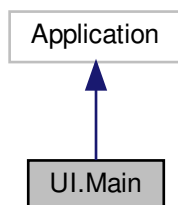
- `dotTypes.java`

## 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 belepési pontja.

#### 7.4.2. Tagfüggvények dokumentációja

#### 7.4.2.1. main()

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

Ez a program belepési pontja.

##### Paraméterek

<i>args</i>	parancssori argumentumok listaja. Nem hasznalom.
-------------	--

#### 7.4.2.2. start()

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

Az ablakot elindító függvény.

##### Paraméterek

<i>primaryStage</i>	JavaFX rendszertől kapott primary stage.
---------------------	--

##### Kivételek

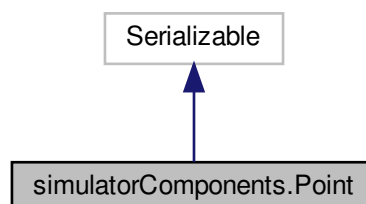
<i>IOException</i>	Akkor dobja, ha a simulationEditor.fxml nem található.
--------------------	--

Ez a dokumentáció az osztályról a következő fájl alapján készült:

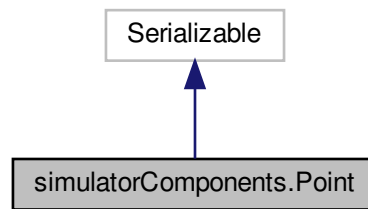
- Main.java

## 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 tarolása és ezeken műveletek végzése. Abban az esetben, ha vektort fejez ki, akkor a Pont az Origoba tolt vector végpontját jelenti.

## 7.5.2. Konstruktorkok és destruktorkok dokumentációja

### 7.5.2.1. Point() [1/2]

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

[Point](#) konstruktura

Paraméterek

<i>x</i>	A kapott x kordinata
<i>y</i>	A kapott y kordinata

### 7.5.2.2. Point() [2/2]

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

[Point](#) osztaly konstruktora

Paraméterek

<i>p</i>	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 hozzaad egy X es Y erteket

Paraméterek

<i>x</i>	A kapott x ertek
<i>y</i>	A kapott y ertek

### 7.5.3.2. add() [2/2]

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

Ket pontot összead x es y kordinatak 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 =  $((x_1-x_2)^2+(y_1-y_2)^2)^{1/2}$  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 es Y kordinatajat elosztja az ertekkel

##### Paraméterek

<i>val</i>	Az ertek, amivel leosztunk
------------	----------------------------

#### 7.5.3.6. dotProduct()

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

Skalaris szorzat szamolasat vegzi ket vector kozott

##### Paraméterek

<i>v</i>	A kapott vector
----------	-----------------

##### Visszatérési érték

A kiszamolt skalaris szorzatot visszaadja.

#### 7.5.3.7. getX()

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

X getter

##### Visszatérési érték

x erteke

#### 7.5.3.8. getY()

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

y getter

##### Visszatérési érték

y erteke



### 7.5.3.9. isOutOfCanvas()

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

Megvizsgálja, hogy a pont kilóg-e a Canvas-rol legalabb egy oldalon

#### Paraméterek

<i>c</i>	A kapott Canvas
<i>r</i>	A kapott korrekciós érték (általában a <a href="#">Dot</a> 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-rol

#### Paraméterek

<i>c</i>	A kapott Canvas
<i>r</i>	A kapott korrekciós érték (általában a <a href="#">Dot</a> 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-rol

#### Paraméterek

<i>c</i>	A kapott Canvas
<i>r</i>	A kapott korrekciós érték (általában a <a href="#">Dot</a> 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-rol

**Paraméterek**

<i>c</i>	A kapott Canvas
<i>r</i>	A kapott korrekciós érték (általában a <a href="#">Dot</a> 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-rol

**Paraméterek**

<i>c</i>	A kapott Canvas
<i>r</i>	A kapott korrekciós érték (általában a <a href="#">Dot</a> sugara)

**Visszatérési érték**

Igen vagy Nem

**7.5.3.14. multiply()**

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

Egy pont X és Y koordinátáit megszorozza a kapott értékekkel

## Paraméterek

<i>val</i>	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

<i>x</i>	A kapott x érték
<i>y</i>	A kapott y érték

**7.5.3.16. subtract()** [2/2]

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

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

## Paraméterek

<i>p</i>	A kapott Pont
----------	---------------

**7.5.4. Adattagok dokumentációja****7.5.4.1. x**

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

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

#### 7.5.4.2. y

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

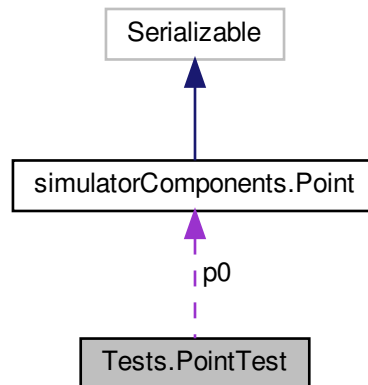
y kordinata a bal felso sarokban van a (0,0)

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 tesztelesere szolgal

### 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]
```

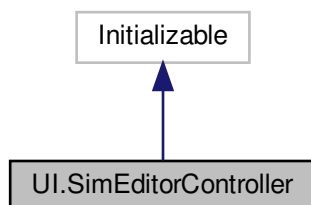
Origo

Ez a dokumentáció az osztályról a következő fájl alapján készült:

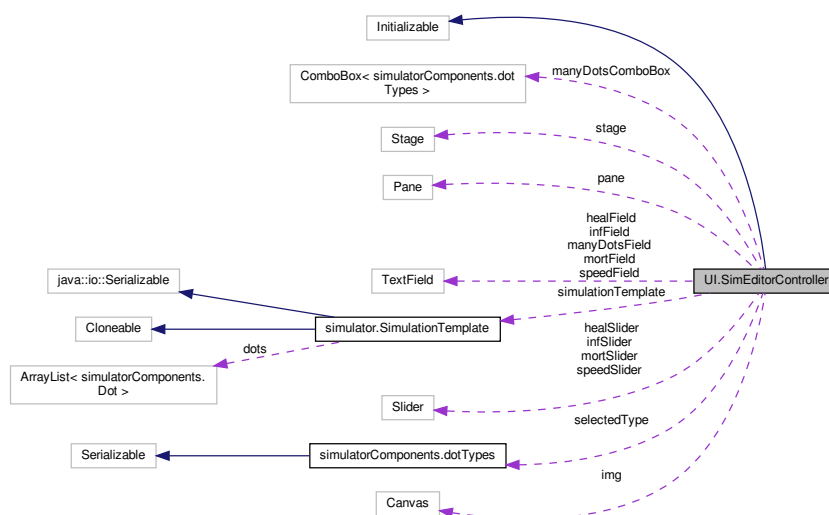
- PointTest.java

## 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`
- final double `radius` = 5.0
- 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`

### 7.7.1. Részletes leírás

Simulation Editor ablak kontroller osztaja. Feladata, hogy kezelje az ablakkal torteno User interakciokat

### 7.7.2. Konstruktorok és destruktorok dokumentációja

#### 7.7.2.1. `SimEditorController()`

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

A kontroller konstruktora. Letrehozza a kontrollert, beallitja a stage-et es a simulationTemplate-et



## Paraméterek

<i>st</i>	Stage, amit a Main-ben hozunk létre.
-----------	--------------------------------------

### 7.7.3. Tagfüggvények dokumentációja

#### 7.7.3.1. addManyDotsPressed()

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

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

#### 7.7.3.2. clearCanvas()

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

Canvas letörlesztet vegzi. új SimulationTemplate-et hoz létre.

#### 7.7.3.3. createDotOnMousePosition()

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

Eger kattintaskor meghívja a Dot létrehozó függvényt, és frissíti a Canvast.

## Paraméterek

<i>event</i>	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 inicializaloja. Feladata az ablakon talalhato elemek ertekeinek beallitasa.

##### Paraméterek

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

#### 7.7.3.7. mortalitySliderChanged()

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

Halalozasi esely beallito csuszka valtozasakor hivodik. Feladata, hogy ezt az erteket tovabbítsa a simulationTemplate-nek, és kiírja ezt a megfelelő TextFieldbe.

#### 7.7.3.8. openSerializedSimulationTemplate()

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

Meghívja a simulationTemplate-et deszerializalo fuggvenyt. SimulationTemplate-et beallítja a kapott ertekekre. A csuszkat és a hozzájuk tartozo TextField-et beallítja a megfelelő ertekekre. Ha null a kapott template a fuggvény visszater. A hibajelzes már az [openSimulationTemplate\(\)](#)-ben megtortent.

#### 7.7.3.9. openSimulationTemplate()

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

Deszerializálja a megadott simulationTemplate-et.

##### Visszatérési érték

SimulationTemplate visszatér egy deszerializált Template-et, vagy nullt 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 szerializalas segitsegevel fajlba. ertesiti a User-t, ha sikeres. ertesiti a felhasznalot, ha a fajl null vagy nem letezik.

**7.7.3.12. setTypeOfDotOnMousePositionToDead()**

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

Beallitja az eger altal lehelyezendo Dot tipusat halottra.

**7.7.3.13. setTypeOfDotOnMousePositionToHealthy()**

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

Beallitja az eger altal lehelyezendo Dot tipusat egeszsegesre.

**7.7.3.14. setTypeOfDotOnMousePositionToInfectious()**

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

Beallitja az eger altal lehelyezendo Dot tipusat fertozore.

**7.7.3.15. setTypeOfDotOnMousePositionToNeutral()**

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

Beallitja az eger altal lehelyezendo Dot tipusat kozombosre.

**7.7.3.16. speedSliderChanged()**

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

Sebesseg beallito csuszka valtozasakor hivodik. Feladata, hogy ezt az erteket tovabbitsa a simulationTemplate-nek, es kiirja ezt a megfelelo TextFieldbe.

**7.7.3.17. startSimulationPlayer()**

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

Letrehozza es elinditja a SimulationPlayer ablakot.

**Kivételek**

<i>IOException</i>	Kivetelt dob, ha az fxml nem letezik.
--------------------	---------------------------------------

**7.7.3.18. startSimulationPlayerFromFile()**

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

Elindítja a SimulationPlayer-t közvetlenül fajlbol.

**Kivételek**

<i>IOException</i>	Kivetelt dob, ha az fxml nem letezik.
--------------------	---------------------------------------

**7.7.4. Adattagok dokumentációja****7.7.4.1. healField**

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

TextField, itt jelezzuk vissza az Usernek a beallitott gyogyulas erteket.

**7.7.4.2. healSlider**

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

Slider, a gyugyulasi esely beallitasara.

**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 jelezzuk vissza az Usernek a beallitott infection erteket.

#### 7.7.4.5. infSlider

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

Slider, az attferozes eselyenek beallitasara.

#### 7.7.4.6. manyDotsComboBox

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

ComboBox, a User itt valasztja ki a Dot tipusat

#### 7.7.4.7. manyDotsField

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

TextField, a User itt adja meg a lerakni kivant Dotok, szamat. Csak Integer lehet, kulonben hibauzenet keletkezik

#### 7.7.4.8. mortField

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

TextField, itt jelezzuk vissza az Usernek a beallitott halalozas ertekeit.

#### 7.7.4.9. mortSlider

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

Slider, a halalozasi esely beallitasara.

#### 7.7.4.10. pane

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

Pane, ebben talalhato a canvas, amire rajzolunk.

#### 7.7.4.11. radius

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

Potty alapertelmezett sugara.

#### 7.7.4.12. selectedType

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

Gomb altal kivalasztott Dot tipusanak taroloja.

#### 7.7.4.13. simulationTemplate

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

SimulationTemplate-et tarolunk, az Editor ezt modositja. Ez tarolja a szimulaciohoz elinditasahoz szukseges adatokat.

#### 7.7.4.14. speedField

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

TextField, itt jelezzuk vissza az Usernek a beallitott sebesseg erteket.

#### 7.7.4.15. speedSlider

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

Slider, a Dot sebessegeinek beallitasara.

#### 7.7.4.16. stage

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

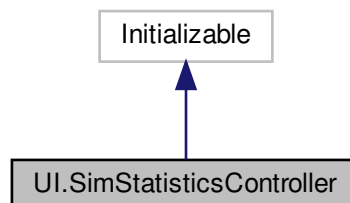
Stage eltarolasa. Konstruktorban kapjuk az ablak létrehozasa soran. Ez alapjan pozicionaljuk a tobbi ablakot.

Ez a dokumentacio az osztalyrol a kovetkezo fajl alapjan készült:

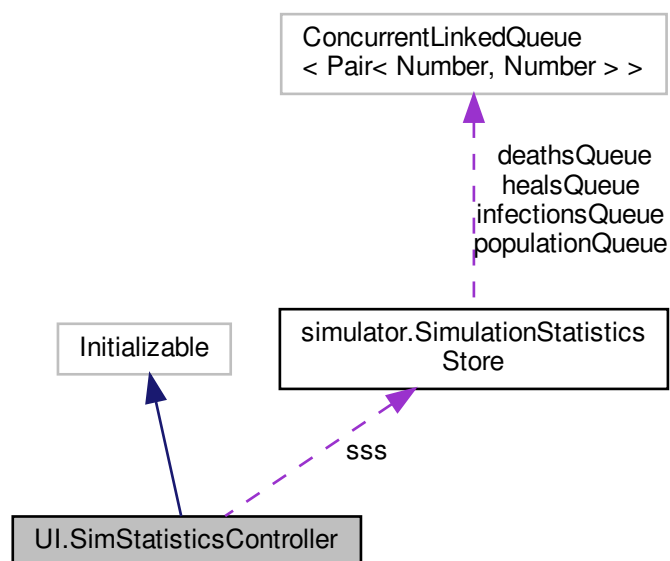
- SimEditorController.java

## 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 osztaja Feladata, hogy kezelje az ablakkal torteno User interakcioikat

## 7.8.2. Konstruktork és destruktorok dokumentációja

### 7.8.2.1. SimStatisticsController()

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

`SimStatisticsController` konstruktorja

#### Paraméterek

<i>st</i>	A kapott Stage
<i>sss</i>	A kapott SimulationStatisticsStore

## 7.8.3. Tagfüggvények dokumentációja

### 7.8.3.1. initialize()

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

Az ablak inicializaloja. Feladata az ablakon talalhato elemek ertekeinek beallitasa.

#### Paraméterek

<i>url</i>	JavaFX használja relatív utvonal meghatározása a root objectnek
<i>resourceBundle</i>	Azok a forrasok, amik a root object helyenek meghatározasahoz szuksegesek

### 7.8.3.2. updateChart()

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

Frissiti a grafikonon megjelenő adatokat

## 7.8.4. Adattagok dokumentációja



#### 7.8.4.1. chart

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

A kirajzolando grafikus

#### 7.8.4.2. deaths

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

halalozasi adatokat tarolo XYChart Series

#### 7.8.4.3. heals

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

gyogyulasi adatokat tarolo XYChart Series

#### 7.8.4.4. infections

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

fertozesi adatokat tarolo XYChart Series

#### 7.8.4.5. population

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

lakossagi adatokat tarolo XYChart Series

#### 7.8.4.6. sss

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

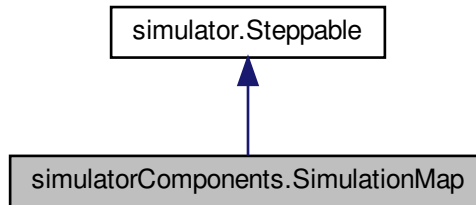
Kozos store a simulationPlayerrel

Ez a dokumentáció az osztályról a következő fájl alapján készült:

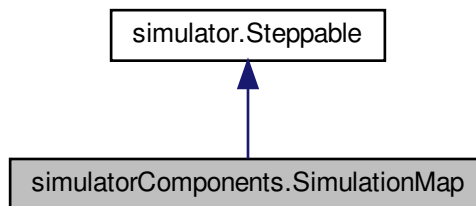
- SimStatisticsController.java

## 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, jelkepezi. Feladata, hogy kor elejen letorli magát

## 7.9.2. Konstruktorkok és destruktorkok dokumentációja

### 7.9.2.1. SimulationMap()

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

[SimulationMap](#) konstruktorja, meghívja a [SimulationMap](#) init függvényet

#### Paraméterek

<i>c</i>	A kapott canvas
----------	-----------------

## 7.9.3. Tagfüggvények dokumentációja

### 7.9.3.1. draw()

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

Letorli a Canvas-t

#### Paraméterek

<i>c</i>	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](#) Palyával utkozni, de a leptethetoseg miatt szukseges.

#### Paraméterek

<i>dot</i>	A kapott <a href="#">Dot</a>
------------	------------------------------

Megvalósítja a következőket: [simulator.Steppable](#).

#### 7.9.3.3. `init()`

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

Inicializálja a `SimulationMap`-et ( Tehat letorli magat)

##### Paraméterek

<code>c</code>	A kapott <code>Canvas</code>
----------------	------------------------------

Megvalósítja a következőket: [simulator.Steppable](#).

#### 7.9.3.4. `isCollidedWith()` [1/2]

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

Megvizsgálja, hogy tudott-e utkozni egy `Dot`-al. Mindig hamisat ad vissza, mert a palya nem utkozik, hanem a hatter szerepet tolti be.

##### Paraméterek

<code>dot</code>	A kapott <code>Dot</code>
------------------	---------------------------

##### 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 utkozni egy másik `Steppable`-el. Mindig hamisat ad vissza, mert a palya nem utkozik, hanem a hatter szerepet tolti be.

## Paraméterek

st	A kapott Masik 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 hatter nem tud az ablakon kívül esni. A leptethetoseg miatt szukseges.

## 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 )
```

Visszahuzza az objektumot a Canvas-ra, A leptethetoseg miatt szukseges.

## 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

<i>c</i>	A kapott Canvas
----------	-----------------

Megvalósítja a következőket: [simulator.Steppable](#).

### 7.9.3.9. step()

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

Lepes során meghívja saját maga refresh függvényét.

#### Paraméterek

<i>c</i>	A kapott Canvas
----------	-----------------

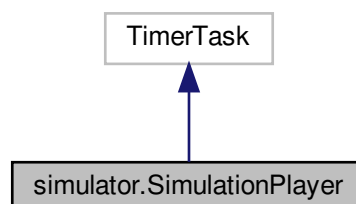
Megvalósítja a következőket: [simulator.Steppable](#).

Ez a dokumentáció az osztályról a következő fájl alapján készült:

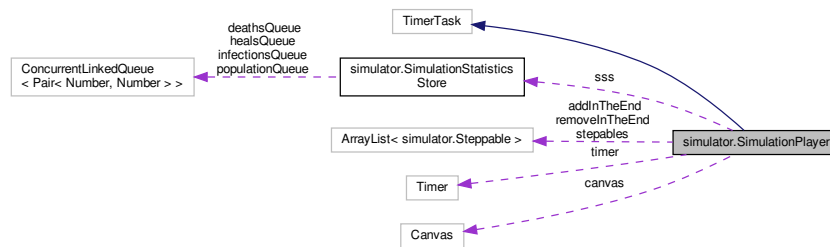
- SimulationMap.java

## 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 szimulacio lejatszasahoz szukseges adatokat es fuggvenyeket tarolja.

### 7.10.2. Konstruktorkok és destruktorkok dokumentációja

#### 7.10.2.1. SimulationPlayer()

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

[SimulationPlayer](#) konstruktora

#### Paraméterek

<i>sim</i>	A kapott <a href="#">SimulationTemplate</a>
<i>canvas</i>	A kapott Canvas
<i>sss</i>	A kapott statisztika tarolo, ami pufferkent funkcional

### 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]
```

Hozzaadja a [Steppable](#) dolgot a hozzáadandoak listájához

Paraméterek

<i>st</i>	A hozzáadando <a href="#">Steppable</a>
-----------	---

#### 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 lepteti a kórt

#### 7.10.3.7. exit()

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

Kilepeskor lezárja és üríti a szükséges dolgokat

#### 7.10.3.8. forwardOneStep()

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

Egy lepest szimulál le. Eloszor mindenki lép. Utána az utkozések jönnek, majd eltávolítjuk és hozzáadjuk a szükséges Steppable-oket, majd Frissítjük a Canvast. Vegül a jelentlegi Tick noveleset vegezzuk.

#### 7.10.3.9. getIncubationPeriod()

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

Lappangasi idot visszado fuggveny

##### Visszatérési érték

Lappangasi ido Tickekben merve

#### 7.10.3.10. getRemove()

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

Viszzadja az eltavolitandoak listajat

##### Visszatérési érték

Az eltavolitandoak listaja

#### 7.10.3.11. getRemoveTime()

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

Halott Dot eltunesi idejet hatarozza meg

##### Visszatérési érték

Eltunesi ido Tickekben merve

#### 7.10.3.12. moveDotsFromOutOfWindow()

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

A Canvasrol kilogo dotokat visszarakja a Canvas-ra

## Paraméterek

<i>img</i>	A kapott Canvas
------------	-----------------

**7.10.3.13. refresh()**

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

Frissíti a teljes szimulacio tartalmat Az első [Steppable](#) mindig a pályá maga

## Paraméterek

<i>c</i>	A kapott Canvas
----------	-----------------

**7.10.3.14. removeSteppable()**

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

Hozzaadja a [Steppable](#) dolgot az eltávolítandók listájához

## Paraméterek

<i>st</i>	Az eltávolítando <a href="#">Steppable</a>
-----------	--

**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 erzett idő telest

#### 7.10.3.18. speedUp()

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

Gyorsítja a felhasználó által erzett idő telest

### 7.10.4. Adattagok dokumentációja

#### 7.10.4.1. addInTheEnd

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

Olyan leptethető dolgok tárolója, amit hozzá kell adni a leptethető 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 egészséges Dotok száma

**7.10.4.6. infectedCnt**

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

ebben fertozo Dotok szama

**7.10.4.7. millisecondsElapsed**

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

Szimulacio kezdete ota eltelt ido

**7.10.4.8. minPeriod**

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

Ket kor kozott eltelt minimalis periodusido

**7.10.4.9. neutralCnt**

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

ebben semleges Dotok szama

**7.10.4.10. oneTickInMs**

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

Egy Tick Ms-ban merve

**7.10.4.11. paused**

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

Le van-e szuneteltetve a szimulacio

**7.10.4.12. removeInTheEnd**

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

Olyan leptetheto dolgok taroloja, amit el kell tavolitani a steppables kozul a Tick vegén

**7.10.4.13. sendDataPeriod**

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

Adatkuldes gyakorisaga

#### 7.10.4.14. sss

`SimulationStatisticsStore` `simulator.SimulationPlayer.sss` [package]

Szimulacio Statisztikat tarolo osztaly Pufferkent funkcional

#### 7.10.4.15. stepables

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

Leptetheto dolgok taroloja

#### 7.10.4.16. timer

`Timer` `simulator.SimulationPlayer.timer` [package]

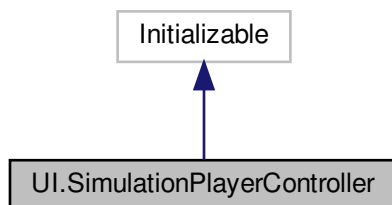
Idozito, a koronkent torteno lepesert felel

Ez a dokumentáció az osztályról a következő fájl alapján készült:

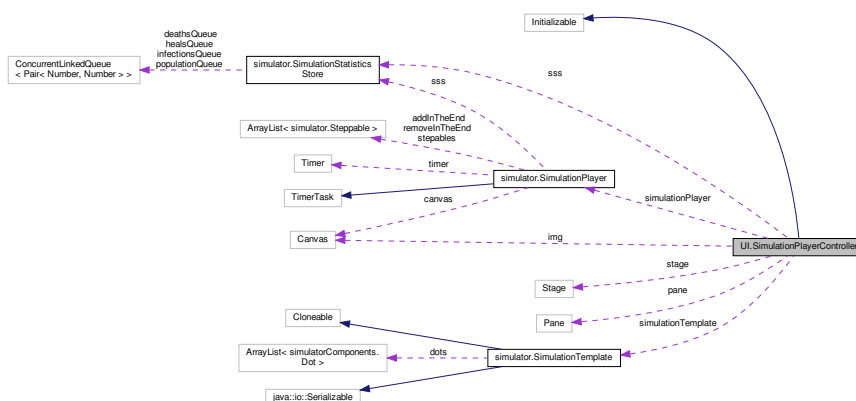
- `SimulationPlayer.java`

## 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 osztaja. Feladata, hogy kezelje az ablakkal torteno User interakciokat

### 7.11.2. Konstruktorok és destruktorok dokumentációja

#### 7.11.2.1. SimulationPlayerController()

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

[SimulationPlayerController](#) konstruktora

## Paraméterek

<i>st</i>	A kapott Stage
<i>sim</i>	A kapott SimulationTemplate

### 7.11.3. Tagfüggvények dokumentációja

#### 7.11.3.1. initialize()

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

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

## Paraméterek

<i>url</i>	JavaFX használja relatív útvonal meghatározása a root objectnek
<i>resourceBundle</i>	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

<i>IOException</i>	kivetelt 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 tarol

#### 7.11.4.5. simulationTemplate

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

Futatando szimulacio adatait tarolja

#### 7.11.4.6. sss

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

Statisztika puffer tara

#### 7.11.4.7. stage

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

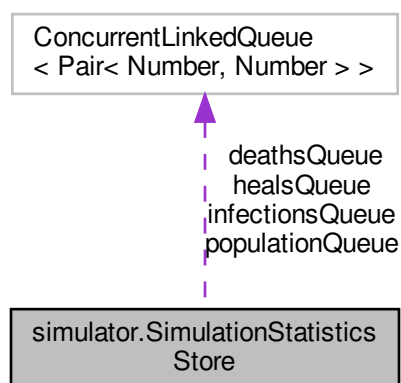
Stage eltarolasa. Konstruktorban kapjuk az ablak létrehozasa soran. Ez alapjan pozicionaljuk a tobbi ablakot.

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 ConcurrentLinkedQueue<>()
- ConcurrentLinkedQueue< Pair< Number, Number > > [deathsQueue](#) = new ConcurrentLinkedQueue<>()
- ConcurrentLinkedQueue< Pair< Number, Number > > [infectionsQueue](#) = new ConcurrentLinkedQueue<>()
- ConcurrentLinkedQueue< Pair< Number, Number > > [healsQueue](#) = new ConcurrentLinkedQueue<>()

### 7.12.1. Részletes leírás

Puffer tarolo a stasztikai adatok tarolasara

### 7.12.2. Konstruktorkok és destruktorkok 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 )
```

Hozzaad egy uj erteket a deathsQueue-hoz

## Paraméterek

<i>time</i>	Idobelyeg (Tick-ben)
<i>val</i>	Az aktualis ertekek

**7.12.3.2. addHealChange()**

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

Hozzaad egy uj ertekek a healsQueue-hoz

## Paraméterek

<i>time</i>	Idobelyeg (Tick-ben)
<i>val</i>	Az aktualis ertekek

**7.12.3.3. addInfectionChange()**

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

Hozzaad egy uj ertekek a infectionsQueue-hoz

## Paraméterek

<i>time</i>	Idobelyeg (Tick-ben)
<i>val</i>	Az aktualis ertekek

**7.12.3.4. addPopulationChange()**

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

Hozzaad egy uj ertekek a populationQueue-hoz

## Paraméterek

<i>time</i>	Idobelyeg (Tick-ben)
<i>val</i>	Az aktualis ertekek

**7.12.3.5. clearAll()**

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

uriti az osszes Queue-t

**7.12.3.6. clearDeathsQueue()**

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

uriti a deathsQueue-t

**7.12.3.7. clearHealsQueue()**

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

uriti a healsQueue-t

**7.12.3.8. clearInfectionsQueue()**

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

uriti az infectionsQueue-t

**7.12.3.9. clearPopulationQueue()**

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

uriti a populationQueue-t

**7.12.3.10. getDeathsQueue()**

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.getDeathsQueue ( )
```

Visszadja a deathsQueue-t

Visszatérési érték

deathsQueue

#### 7.12.3.11. getHealsQueue()

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.getHealsQueue ( )
```

Visszadja a healsQueue-t

**Visszatérési érték**

healsQueue

#### 7.12.3.12. getInfectionsQueue()

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.getInfectionsQueue ( )
```

Visszadja az infectionsQueue-t

**Visszatérési érték**

infectionsQueue

#### 7.12.3.13. getPopulationQueue()

```
ConcurrentLinkedQueue<Pair<Number, Number> > simulator.SimulationStatisticsStore.getPopulationQueue ( )
```

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]
```

halalozasi 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.infectionsQueue = new ConcurrentLinkedQueue<>() [package]
```

fertőzési adatok puffere

#### 7.12.4.4. populationQueue

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

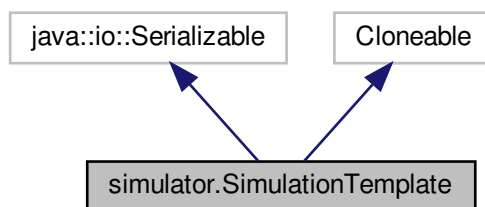
population adatok puffere. Azert erre esett a választásom, mert így a Concurrent Modification Exception-t el tudom kerülni, mert történhet olyan, hogy éppen olvasom a Puffert, mikor a vegere már kerül az új adat

Ez a dokumentáció az osztályról a következő fájl alapján készült:

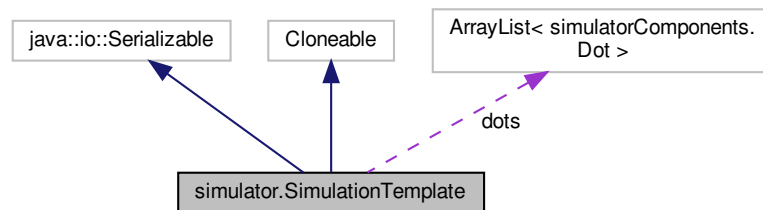
- SimulationStatisticsStore.java

### 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. Szimulacio elindításához kapcsolatos adatokat tarol. Megvalósítja a java.io.Serializable és Cloneable interfészeket

### 7.13.2. Konstruktorkok és destruktorkok dokumentációja



#### 7.13.2.1. SimulationTemplate() [1/2]

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

[SimulationTemplate](#) konstruktura Beallítja az alapértelmezett értékeket és létrehozza az objektumot.

#### 7.13.2.2. SimulationTemplate() [2/2]

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

Simulation template copy konstruktora

Paraméterek

<i>st</i>	a masolando <a href="#">SimulationTemplate</a>
-----------	--

### 7.13.3. Tagfüggvények dokumentációja

#### 7.13.3.1. addDot()

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

Hozzaad egy Dot-ot a Dotokat tartalmazó listához

Paraméterek

<i>d</i>	A kapott Dot
----------	--------------

#### 7.13.3.2. clone()

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

clone függvény feluldefiníálása. Celja, hogy a [SimulationTemplate](#) masolasat megvalosítsa.

Visszatérési érték

Visszadja az objektum masolatát(deep copy)

## Kivételek

<i>CloneNotSupportedException</i>	kivetelt dobhat (de nem fog, mert a dot es a <a href="#">SimulationTemplate</a> is klonozható)
-----------------------------------	--

**7.13.3.3. createDot()**

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

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

## Paraméterek

<i>type</i>	A létrehozandó Dot típusa
<i>x</i>	A létrehozandó Dot x koordinátája
<i>y</i>	A létrehozandó Dot y koordinátája
<i>r</i>	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 ( )
```

Fertozesi esely getter fuggvenye

**Visszatérési érték**

A fertozesi esely

**7.13.3.7. getMortChance()**

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

Halalozasi esely getter fuggvenye

**Visszatérési érték**

A kapott halalozasi esely

**7.13.3.8. getSpeedOfDot()**

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

A sebesseg getter fuggvenye

**Visszatérési érték**

A sebessege a Dot-nak

**7.13.3.9. refresh()**

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

Frissiti a Canvas tartalmat

**Paraméterek**

c	A kapott canvas
---	-----------------

#### 7.13.3.10. setHealChance()

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

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

##### Paraméterek

<i>heal</i>	A kapott gyógyulási esély
-------------	---------------------------

#### 7.13.3.11. setInfection()

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

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

##### Paraméterek

<i>inf</i>	A kapott fertőzési esély
------------	--------------------------

#### 7.13.3.12. setMortality()

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

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

##### Paraméterek

<i>mort</i>	A kapott halálozási esély
-------------	---------------------------

#### 7.13.3.13. setSpeed()

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

Sebesesség esély setter függvénye Beallítja a kapott sebességet a jövőben létrejövő Dot-okra.

## Paraméterek

<i>speed</i>	A kapott sebesseg
--------------	-------------------

### 7.13.4. Adattagok dokumentációja

#### 7.13.4.1. dots

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

Dot-okat tarolo ArrayList.

#### 7.13.4.2. healChance

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

Gyogyulasi esely

#### 7.13.4.3. infChance

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

atfertozesi esely

#### 7.13.4.4. mortChance

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

Halalozasi esely

#### 7.13.4.5. speedOfDot

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

Dot sebessege

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 ( )
```

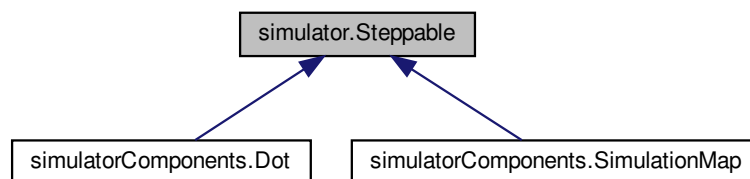
create Dot tesztje

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 leptethető dolgokat valósítja meg

### 7.15.2. Tagfüggvények dokumentációja

#### 7.15.2.1. draw()

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

Kirajzolja a leptethető dolgot a Canvason

Paraméterek

<i>c</i>	A kapott Canvas
----------	-----------------

Megvalósítják a következők: [simulatorComponents.SimulationMap](#) és [simulatorComponents.Dot](#).

#### 7.15.2.2. hitBy()

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

Dottal való utközést lekezelő függvény

Paraméterek

<i>dot</i>	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

<i>c</i>	A kapott Canvas
----------	-----------------

Megvalósítják a következők: [simulatorComponents.SimulationMap](#) és [simulatorComponents.Dot](#).

**7.15.2.4. isCollidedWith()** [1/2]

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

utkozott-e a kapott Dot-al?

## Paraméterek

<i>dot</i>	A kapott Masik 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 )
```

utkozott-e a kapott Steppable-el?

## Paraméterek

<i>st</i>	A kapott Masik <a href="#">Steppable</a>
-----------	--

## 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 leptetheto dolog a Canvason kívül tartozkodik-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 )
```

Visszahuzza a leptetheto 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 leptetheto 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 korben vegzendo lepesek

##### Paraméterek

<i>c</i>	A kapott Canvas
----------	-----------------

Megvalósítják a következők: [simulatorComponents.SimulationMap](#) és [simulatorComponents.Dot](#).

Ez a dokumentáció az interfésről a következő fájl alapján készült:

- Steppable.java

# Tárgymutató

- add
  - simulatorComponents.Point, [36](#), [37](#)
  - Tests.PointTest, [43](#)
- addDeadDot
  - simulator.SimulationPlayer, [62](#)
- addDeathsChange
  - simulator.SimulationStatisticsStore, [73](#)
- addDot
  - simulator.SimulationTemplate, [79](#)
- addHealChange
  - simulator.SimulationStatisticsStore, [74](#)
- addHealedDot
  - simulator.SimulationPlayer, [63](#)
- addInfectedDot
  - simulator.SimulationPlayer, [63](#)
- addInfectionChange
  - simulator.SimulationStatisticsStore, [74](#)
- addInTheEnd
  - simulator.SimulationPlayer, [66](#)
- addManyDotsPressed
  - UI.SimEditorController, [47](#)
- addPopulationChange
  - simulator.SimulationStatisticsStore, [74](#)
- addSteppable
  - simulator.SimulationPlayer, [63](#)
- bounceBack
  - simulatorComponents.Dot, [22](#)
- calcDisplacement
  - simulatorComponents.Point, [37](#)
  - Tests.PointTest, [43](#)
- calcDistance
  - simulatorComponents.Point, [37](#)
  - Tests.PointTest, [43](#)
- canvas
  - simulator.SimulationPlayer, [66](#)
- changePlayAndPause
  - simulator.SimulationPlayer, [63](#)
- chart
  - UI.SimStatisticsController, [54](#)
- clearAll
  - simulator.SimulationStatisticsStore, [75](#)
- clearCanvas
  - UI.SimEditorController, [47](#)
- clearDeathsQueue
  - simulator.SimulationStatisticsStore, [75](#)
- clearHealsQueue
  - simulator.SimulationStatisticsStore, [75](#)
- clearInfectionsQueue
  - simulator.SimulationStatisticsStore, [75](#)
- clearPopulationQueue
  - simulator.SimulationStatisticsStore, [75](#)
- clone
  - simulator.SimulationTemplate, [79](#)
  - simulatorComponents.Dot, [22](#)
- createDot
  - simulator.SimulationTemplate, [80](#)
  - Tests.SimulationTemplateTest, [84](#)
- createDotOnMousePosition
  - UI.SimEditorController, [47](#)
- currTick
  - simulator.SimulationPlayer, [66](#)
- currTickIncrease
  - simulator.SimulationPlayer, [63](#)
- Dead
  - simulatorComponents.dotTypes, [32](#)
- deadCnt
  - simulator.SimulationPlayer, [66](#)
- deaths
  - UI.SimStatisticsController, [55](#)
- deathsQueue
  - simulator.SimulationStatisticsStore, [76](#)
- die
  - simulatorComponents.Dot, [23](#)
- divide
  - simulatorComponents.Point, [37](#)
  - Tests.PointTest, [43](#)
- divideByZero
  - Tests.PointTest, [43](#)
- Dot
  - simulatorComponents.Dot, [21](#)
- dotProduct
  - simulatorComponents.Point, [38](#)
  - Tests.PointTest, [43](#)
- dots
  - simulator.SimulationTemplate, [83](#)
- draw
  - simulator.Steppable, [85](#)
  - simulatorComponents.Dot, [23](#)
  - simulatorComponents.SimulationMap, [57](#)
- drawCenters
  - simulatorComponents.Dot, [23](#)
- exit
  - simulator.SimulationPlayer, [63](#)
- forwardOneStep

- simulator.SimulationPlayer, 63
- getDeathsQueue
  - simulator.SimulationStatisticsStore, 75
- getDots
  - simulator.SimulationTemplate, 80
- getHealChance
  - simulator.SimulationTemplate, 80
- getHealsQueue
  - simulator.SimulationStatisticsStore, 75
- getIncubationPeriod
  - simulator.SimulationPlayer, 64
- getInfChance
  - simulator.SimulationTemplate, 80
- getInfectionsQueue
  - simulator.SimulationStatisticsStore, 76
- getLocation
  - simulatorComponents.Dot, 23
- getMortChance
  - simulator.SimulationTemplate, 81
- getPopulationQueue
  - simulator.SimulationStatisticsStore, 76
- getRadius
  - simulatorComponents.Dot, 24
- getRemove
  - simulator.SimulationPlayer, 64
- getRemoveTime
  - simulator.SimulationPlayer, 64
- getSpeedOfDot
  - simulator.SimulationTemplate, 81
- getType
  - simulatorComponents.Dot, 24
- getX
  - simulatorComponents.Point, 38
- getY
  - simulatorComponents.Point, 38
- heal
  - simulatorComponents.Dot, 24
- healChance
  - simulator.SimulationTemplate, 83
  - simulatorComponents.Dot, 29
- healedCnt
  - simulator.SimulationPlayer, 66
- healField
  - UI.SimEditorController, 50
- heals
  - UI.SimStatisticsController, 55
- healSlider
  - UI.SimEditorController, 50
- healSliderChanged
  - UI.SimEditorController, 47
- healsQueue
  - simulator.SimulationStatisticsStore, 76
- Healthy
  - simulatorComponents.dotTypes, 32
- hitBy
  - simulator.Steppable, 85
  - simulatorComponents.Dot, 24
- simulatorComponents.SimulationMap, 57
- img
  - UI.SimEditorController, 50
  - UI.SimulationPlayerController, 71
- infChance
  - simulator.SimulationTemplate, 83
  - simulatorComponents.Dot, 29
- infectedBy
  - simulatorComponents.Dot, 25
- infectedCnt
  - simulator.SimulationPlayer, 66
- infections
  - UI.SimStatisticsController, 55
- infectionsQueue
  - simulator.SimulationStatisticsStore, 77
- Infectious
  - simulatorComponents.dotTypes, 32
- infField
  - UI.SimEditorController, 50
- infSlider
  - UI.SimEditorController, 50
- infSliderChanged
  - UI.SimEditorController, 47
- init
  - simulator.Steppable, 85
  - simulatorComponents.Dot, 25
  - simulatorComponents.SimulationMap, 58
- initialize
  - UI.SimEditorController, 47
  - UI.SimStatisticsController, 54
  - UI.SimulationPlayerController, 70
- initVelocity
  - simulatorComponents.Dot, 25
- isCollidedWith
  - simulator.Steppable, 86
  - simulatorComponents.Dot, 25, 26
  - simulatorComponents.SimulationMap, 58
  - Tests.DotTest, 31
- isOutOfCanvas
  - simulatorComponents.Point, 38
- isOutOfCanvasBottom
  - simulatorComponents.Point, 39
- isOutOfCanvasLeft
  - simulatorComponents.Point, 39
- isOutOfCanvasRight
  - simulatorComponents.Point, 40
- isOutOfCanvasTop
  - simulatorComponents.Point, 40
- isOutOfWindow
  - simulator.Steppable, 86
  - simulatorComponents.Dot, 26
  - simulatorComponents.SimulationMap, 59
- location
  - simulatorComponents.Dot, 29
- main
  - UI.Main, 33

- manyDotsComboBox
  - UI.SimEditorController, [51](#)
- manyDotsField
  - UI.SimEditorController, [51](#)
- mass
  - simulatorComponents.Dot, [29](#)
- millisecondsElapsed
  - simulator.SimulationPlayer, [67](#)
- minPeriod
  - simulator.SimulationPlayer, [67](#)
- mortalitySliderChanged
  - UI.SimEditorController, [48](#)
- mortChance
  - simulator.SimulationTemplate, [83](#)
  - simulatorComponents.Dot, [29](#)
- mortField
  - UI.SimEditorController, [51](#)
- mortSlider
  - UI.SimEditorController, [51](#)
- moveBack
  - simulator.Steppable, [87](#)
  - simulatorComponents.Dot, [27](#)
  - simulatorComponents.SimulationMap, [59](#)
- moveDotsFromOutOfWindow
  - simulator.SimulationPlayer, [64](#)
- multiply
  - simulatorComponents.Point, [40](#)
  - Tests.PointTest, [43](#)
- Neutral
  - simulatorComponents.dotTypes, [32](#)
- neutralCnt
  - simulator.SimulationPlayer, [67](#)
- None
  - simulatorComponents.dotTypes, [32](#)
- oneTickInMs
  - simulator.SimulationPlayer, [67](#)
- openSerializedSimulationTemplate
  - UI.SimEditorController, [48](#)
- openSimulationTemplate
  - UI.SimEditorController, [48](#)
- p0
  - Tests.PointTest, [44](#)
- pane
  - UI.SimEditorController, [51](#)
  - UI.SimulationPlayerController, [71](#)
- paused
  - simulator.SimulationPlayer, [67](#)
- playAndPausePressed
  - UI.SimulationPlayerController, [70](#)
- Point
  - simulatorComponents.Point, [36](#)
- population
  - UI.SimStatisticsController, [55](#)
- populationQueue
  - simulator.SimulationStatisticsStore, [77](#)
- radius
  - simulatorComponents.Dot, [29](#)
  - UI.SimEditorController, [51](#)
- redraw
  - UI.SimEditorController, [48](#)
  - UI.SimulationPlayerController, [70](#)
- reDrawCallCnt
  - UI.SimulationPlayerController, [71](#)
- refresh
  - simulator.SimulationPlayer, [65](#)
  - simulator.SimulationTemplate, [81](#)
  - simulator.Steppable, [87](#)
  - simulatorComponents.Dot, [27](#)
  - simulatorComponents.SimulationMap, [59](#)
- remove
  - simulatorComponents.Dot, [27](#)
- removeInTheEnd
  - simulator.SimulationPlayer, [67](#)
- removeSteppable
  - simulator.SimulationPlayer, [65](#)
- run
  - simulator.SimulationPlayer, [65](#)
- selectedType
  - UI.SimEditorController, [51](#)
- sendData
  - simulator.SimulationPlayer, [65](#)
- sendDataPeriod
  - simulator.SimulationPlayer, [67](#)
- serializeSimulationTemplate
  - UI.SimEditorController, [48](#)
- setHealChance
  - simulator.SimulationTemplate, [81](#)
  - simulatorComponents.Dot, [27](#)
- setInfChance
  - simulatorComponents.Dot, [28](#)
- setInfection
  - simulator.SimulationTemplate, [82](#)
- setLocation
  - simulatorComponents.Dot, [28](#)
- setMortality
  - simulator.SimulationTemplate, [82](#)
- setMortChance
  - simulatorComponents.Dot, [28](#)
- setSpeed
  - simulator.SimulationTemplate, [82](#)
- setTypeOfDotOnMousePositionToDead
  - UI.SimEditorController, [49](#)
- setTypeOfDotOnMousePositionToHealthy
  - UI.SimEditorController, [49](#)
- setTypeOfDotOnMousePositionToInfectious
  - UI.SimEditorController, [49](#)
- setTypeOfDotOnMousePositionToNeutral
  - UI.SimEditorController, [49](#)
- setUp
  - Tests.PointTest, [44](#)
- SimEditorController
  - UI.SimEditorController, [46](#)
- SimStatisticsController

- UI.SimStatisticsController, 54
- SimulationMap
  - simulatorComponents.SimulationMap, 57
- SimulationPlayer
  - simulator.SimulationPlayer, 62
- simulationPlayer
  - UI.SimulationPlayerController, 71
- SimulationPlayerController
  - UI.SimulationPlayerController, 69
- SimulationStatisticsStore
  - simulator.SimulationStatisticsStore, 73
- SimulationTemplate
  - simulator.SimulationTemplate, 78, 79
- simulationTemplate
  - UI.SimEditorController, 51
  - UI.SimulationPlayerController, 71
- simulator.SimulationPlayer, 60
  - addDeadDot, 62
  - addHealedDot, 63
  - addInfectedDot, 63
  - addInTheEnd, 66
  - addSteppable, 63
  - canvas, 66
  - changePlayAndPause, 63
  - currTick, 66
  - currTickIncrease, 63
  - deadCnt, 66
  - exit, 63
  - forwardOneStep, 63
  - getIncubationPeriod, 64
  - getRemove, 64
  - getRemoveTime, 64
  - healedCnt, 66
  - infectedCnt, 66
  - millisecondsElapsed, 67
  - minPeriod, 67
  - moveDotsFromOutOfWindow, 64
  - neutralCnt, 67
  - oneTickInMs, 67
  - paused, 67
  - refresh, 65
  - removeInTheEnd, 67
  - removeSteppable, 65
  - run, 65
  - sendData, 65
  - sendDataPeriod, 67
  - SimulationPlayer, 62
  - speedDown, 65
  - speedUp, 66
  - sss, 67
  - stepables, 68
  - timer, 68
- simulator.SimulationStatisticsStore, 72
  - addDeathsChange, 73
  - addHealChange, 74
  - addInfectionChange, 74
  - addPopulationChange, 74
  - clearAll, 75
  - clearDeathsQueue, 75
  - clearHealsQueue, 75
  - clearInfectionsQueue, 75
  - clearPopulationQueue, 75
  - deathsQueue, 76
  - getDeathsQueue, 75
  - getHealsQueue, 75
  - getInfectionsQueue, 76
  - getPopulationQueue, 76
  - healsQueue, 76
  - infectionsQueue, 77
  - populationQueue, 77
  - SimulationStatisticsStore, 73
- simulator.SimulationTemplate, 77
  - addDot, 79
  - clone, 79
  - createDot, 80
  - dots, 83
  - getDots, 80
  - getHealChance, 80
  - getInfChance, 80
  - getMortChance, 81
  - getSpeedOfDot, 81
  - healChance, 83
  - infChance, 83
  - mortChance, 83
  - refresh, 81
  - setHealChance, 81
  - setInfection, 82
  - setMortality, 82
  - setSpeed, 82
  - SimulationTemplate, 78, 79
  - speedOfDot, 83
- simulator.Steppable, 84
  - draw, 85
  - hitBy, 85
  - init, 85
  - isCollidedWith, 86
  - isOutOfWindow, 86
  - moveBack, 87
  - refresh, 87
  - step, 87
- simulatorComponents.Dot, 19
  - bounceBack, 22
  - clone, 22
  - die, 23
  - Dot, 21
  - draw, 23
  - drawCenters, 23
  - getLocation, 23
  - getRadius, 24
  - getType, 24
  - heal, 24
  - healChance, 29
  - hitBy, 24
  - infChance, 29
  - infectedBy, 25
  - init, 25

- initVelocity, 25
- isCollidedWith, 25, 26
- isOutOfWindow, 26
- location, 29
- mass, 29
- mortChance, 29
- moveBack, 27
- radius, 29
- refresh, 27
- remove, 27
- setHealChance, 27
- setInfChance, 28
- setLocation, 28
- setMortChance, 28
- sinceDead, 30
- sinceInfection, 30
- step, 28
- type, 30
- velocity, 30
- simulatorComponents.dotTypes, 31
  - Dead, 32
  - Healthy, 32
  - Infectious, 32
  - Neutral, 32
  - None, 32
- simulatorComponents.Point, 34
  - add, 36, 37
  - calcDisplacement, 37
  - calcDistance, 37
  - divide, 37
  - dotProduct, 38
  - getX, 38
  - getY, 38
  - isOutOfCanvas, 38
  - isOutOfCanvasBottom, 39
  - isOutOfCanvasLeft, 39
  - isOutOfCanvasRight, 40
  - isOutOfCanvasTop, 40
  - multiply, 40
  - Point, 36
  - subtract, 41
  - x, 41
  - y, 41
- simulatorComponents.SimulationMap, 56
  - draw, 57
  - hitBy, 57
  - init, 58
  - isCollidedWith, 58
  - isOutOfWindow, 59
  - moveBack, 59
  - refresh, 59
  - SimulationMap, 57
  - step, 60
- sinceDead
  - simulatorComponents.Dot, 30
- sinceInfection
  - simulatorComponents.Dot, 30
- speedDown
  - simulator.SimulationPlayer, 65
- speedDownPressed
  - UI.SimulationPlayerController, 70
- speedField
  - UI.SimEditorController, 52
- speedOfDot
  - simulator.SimulationTemplate, 83
- speedSlider
  - UI.SimEditorController, 52
- speedSliderChanged
  - UI.SimEditorController, 49
- speedUp
  - simulator.SimulationPlayer, 66
- speedUpPressed
  - UI.SimulationPlayerController, 70
- sss
  - simulator.SimulationPlayer, 67
  - UI.SimStatisticsController, 55
  - UI.SimulationPlayerController, 72
- stage
  - UI.SimEditorController, 52
  - UI.SimulationPlayerController, 72
- start
  - UI.Main, 34
- startSimulationPlayer
  - UI.SimEditorController, 49
- startSimulationPlayerFromFile
  - UI.SimEditorController, 50
- statisticsPressed
  - UI.SimulationPlayerController, 70
- step
  - simulator.Steppable, 87
  - simulatorComponents.Dot, 28
  - simulatorComponents.SimulationMap, 60
- stepables
  - simulator.SimulationPlayer, 68
- stepPressed
  - UI.SimulationPlayerController, 71
- subtract
  - simulatorComponents.Point, 41
  - Tests.PointTest, 44
- Tests.DotTest, 30
  - isCollidedWith, 31
- Tests.PointTest, 42
  - add, 43
  - calcDisplacement, 43
  - calcDistance, 43
  - divide, 43
  - divideByZero, 43
  - dotProduct, 43
  - multiply, 43
  - p0, 44
  - setUp, 44
  - subtract, 44
- Tests.SimulationTemplateTest, 84
  - createDot, 84
- timer
  - simulator.SimulationPlayer, 68

- type
  - simulatorComponents.Dot, 30
- UI.Main, 33
  - main, 33
  - start, 34
- UI.SimEditorController, 45
  - addManyDotsPressed, 47
  - clearCanvas, 47
  - createDotOnMousePosition, 47
  - healField, 50
  - healSlider, 50
  - healSliderChanged, 47
  - img, 50
  - infField, 50
  - infSlider, 50
  - infSliderChanged, 47
  - initialize, 47
  - manyDotsComboBox, 51
  - manyDotsField, 51
  - mortalitySliderChanged, 48
  - mortField, 51
  - mortSlider, 51
  - openSerializedSimulationTemplate, 48
  - openSimulationTemplate, 48
  - pane, 51
  - radius, 51
  - redraw, 48
  - selectedType, 51
  - serializeSimulationTemplate, 48
  - setTypeOfDotOnMousePositionToDead, 49
  - setTypeOfDotOnMousePositionToHealthy, 49
  - setTypeOfDotOnMousePositionToInfectious, 49
  - setTypeOfDotOnMousePositionToNeutral, 49
  - SimEditorController, 46
  - simulationTemplate, 51
  - speedField, 52
  - speedSlider, 52
  - speedSliderChanged, 49
  - stage, 52
  - startSimulationPlayer, 49
  - startSimulationPlayerFromFile, 50
- UI.SimStatisticsController, 52
  - chart, 54
  - deaths, 55
  - heals, 55
  - infections, 55
  - initialize, 54
  - population, 55
  - SimStatisticsController, 54
  - sss, 55
  - updateChart, 54
- UI.SimulationPlayerController, 68
  - img, 71
  - initialize, 70
  - pane, 71
  - playAndPausePressed, 70
  - redraw, 70
  - reDrawCallCnt, 71
  - simulationPlayer, 71
  - SimulationPlayerController, 69
  - simulationTemplate, 71
  - speedDownPressed, 70
  - speedUpPressed, 70
  - sss, 72
  - stage, 72
  - statisticsPressed, 70
  - stepPressed, 71
- updateChart
  - UI.SimStatisticsController, 54
- velocity
  - simulatorComponents.Dot, 30
- x
  - simulatorComponents.Point, 41
- y
  - simulatorComponents.Point, 41