

## 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</b>	<b>3</b>
2.1. Fordítási tudnivalók . . . . .	3
<b>3. Prog3: Vírus szimuláció nagyházfeladat specifikáció</b>	<b>5</b>
3.1. Az ötlet leírása . . . . .	5
3.2. A program funkcionalitása a felhasználó szemszögéből . . . . .	5
3.3. Megoldási ötlet (vázlat) . . . . .	7
<b>4. Felhasználói dokumentáció</b>	<b>9</b>
4.1. A program általános leírása . . . . .	9
4.2. A program funkcionalitása a felhasználó szemszögéből . . . . .	9
<b>5. Hierarchikus mutató</b>	<b>13</b>
5.1. Osztályhierarchia . . . . .	13
<b>6. Osztálymutató</b>	<b>15</b>
6.1. Osztálylista . . . . .	15
<b>7. Osztályok dokumentációja</b>	<b>17</b>
7.1. simulatorComponents.Dot osztályreferencia . . . . .	17
7.1.1. Részletes leírás . . . . .	19
7.1.2. Konstruktork és destruktorok dokumentációja . . . . .	19
7.1.2.1. Dot() [1/3] . . . . .	19
7.1.2.2. Dot() [2/3] . . . . .	19
7.1.2.3. Dot() [3/3] . . . . .	19
7.1.3. Tagfüggvények dokumentációja . . . . .	20
7.1.3.1. bounceBack() . . . . .	20
7.1.3.2. clone() . . . . .	20
7.1.3.3. die() . . . . .	21
7.1.3.4. draw() . . . . .	21
7.1.3.5. drawCenters() . . . . .	21
7.1.3.6. getLocation() . . . . .	21
7.1.3.7. getRadius() . . . . .	22
7.1.3.8. getType() . . . . .	22
7.1.3.9. heal() . . . . .	22
7.1.3.10. hitBy() . . . . .	22
7.1.3.11. infectedBy() . . . . .	23
7.1.3.12. init() . . . . .	23
7.1.3.13. initVelocity() . . . . .	23
7.1.3.14. isCollidedWith() [1/2] . . . . .	24
7.1.3.15. isCollidedWith() [2/2] . . . . .	24

7.1.3.16.	<a href="#">isOutOfWindow()</a>	24
7.1.3.17.	<a href="#">moveBack()</a>	25
7.1.3.18.	<a href="#">refresh()</a>	25
7.1.3.19.	<a href="#">remove()</a>	25
7.1.3.20.	<a href="#">setHealChance()</a>	25
7.1.3.21.	<a href="#">setInfChance()</a>	26
7.1.3.22.	<a href="#">setLocation()</a>	26
7.1.3.23.	<a href="#">setMortChance()</a>	26
7.1.3.24.	<a href="#">step()</a>	27
7.1.4.	<a href="#">Adattagok dokumentációja</a>	27
7.1.4.1.	<a href="#">healChance</a>	27
7.1.4.2.	<a href="#">infChance</a>	27
7.1.4.3.	<a href="#">location</a>	27
7.1.4.4.	<a href="#">mass</a>	27
7.1.4.5.	<a href="#">mortChance</a>	27
7.1.4.6.	<a href="#">radius</a>	28
7.1.4.7.	<a href="#">sinceDead</a>	28
7.1.4.8.	<a href="#">sinceInfection</a>	28
7.1.4.9.	<a href="#">type</a>	28
7.1.4.10.	<a href="#">velocity</a>	28
7.2.	<a href="#">Tests.DotTest osztályreferencia</a>	28
7.2.1.	<a href="#">Részletes leírás</a>	28
7.2.2.	<a href="#">Tagfüggvények dokumentációja</a>	29
7.2.2.1.	<a href="#">isCollidedWith()</a>	29
7.3.	<a href="#">simulatorComponents.dotTypes felsoroló referencia</a>	29
7.3.1.	<a href="#">Részletes leírás</a>	30
7.3.2.	<a href="#">Adattagok dokumentációja</a>	30
7.3.2.1.	<a href="#">Dead</a>	30
7.3.2.2.	<a href="#">Healthy</a>	30
7.3.2.3.	<a href="#">Infectious</a>	30
7.3.2.4.	<a href="#">Neutral</a>	30
7.3.2.5.	<a href="#">None</a>	30
7.4.	<a href="#">UI.Main osztályreferencia</a>	31
7.4.1.	<a href="#">Részletes leírás</a>	31
7.4.2.	<a href="#">Tagfüggvények dokumentációja</a>	31
7.4.2.1.	<a href="#">main()</a>	32
7.4.2.2.	<a href="#">start()</a>	32
7.5.	<a href="#">simulatorComponents.Point osztályreferencia</a>	32
7.5.1.	<a href="#">Részletes leírás</a>	33
7.5.2.	<a href="#">Konstruktorok és destruktorok dokumentációja</a>	34
7.5.2.1.	<a href="#">Point() [1/2]</a>	34
7.5.2.2.	<a href="#">Point() [2/2]</a>	34

7.5.3.	Tagfüggvények dokumentációja	34
7.5.3.1.	add() [1/2]	34
7.5.3.2.	add() [2/2]	35
7.5.3.3.	calcDisplacement()	35
7.5.3.4.	calcDistance()	35
7.5.3.5.	divide()	36
7.5.3.6.	dotProduct()	36
7.5.3.7.	getX()	36
7.5.3.8.	getY()	36
7.5.3.9.	isOutOfCanvas()	37
7.5.3.10.	isOutOfCanvasBottom()	37
7.5.3.11.	isOutOfCanvasLeft()	37
7.5.3.12.	isOutOfCanvasRight()	38
7.5.3.13.	isOutOfCanvasTop()	38
7.5.3.14.	multiply()	38
7.5.3.15.	subtract() [1/2]	39
7.5.3.16.	subtract() [2/2]	39
7.5.4.	Adattagok dokumentációja	39
7.5.4.1.	x	39
7.5.4.2.	y	40
7.6.	Tests.PointTest osztályreferencia	40
7.6.1.	Részletes leírás	41
7.6.2.	Tagfüggvények dokumentációja	41
7.6.2.1.	add()	41
7.6.2.2.	calcDisplacement()	41
7.6.2.3.	calcDistance()	41
7.6.2.4.	divide()	41
7.6.2.5.	divideByZero()	41
7.6.2.6.	dotProduct()	41
7.6.2.7.	multiply()	42
7.6.2.8.	setUp()	42
7.6.2.9.	subtract()	42
7.6.3.	Adattagok dokumentációja	42
7.6.3.1.	p0	42
7.7.	UI.SimEditorController osztályreferencia	43
7.7.1.	Részletes leírás	44
7.7.2.	Konstruktorok és destruktorok dokumentációja	44
7.7.2.1.	SimEditorController()	44
7.7.3.	Tagfüggvények dokumentációja	45
7.7.3.1.	addManyDotsPressed()	45
7.7.3.2.	clearCanvas()	45
7.7.3.3.	createDotOnMousePosition()	45

7.7.3.4.	healSliderChanged()	45
7.7.3.5.	infSliderChanged()	45
7.7.3.6.	initialize()	46
7.7.3.7.	mortalitySliderChanged()	46
7.7.3.8.	openSerializedSimulationTemplate()	46
7.7.3.9.	openSimulationTemplate()	46
7.7.3.10.	redraw()	46
7.7.3.11.	serializeSimulationTemplate()	47
7.7.3.12.	setTypeOfDotOnMousePositionToDead()	47
7.7.3.13.	setTypeOfDotOnMousePositionToHealthy()	47
7.7.3.14.	setTypeOfDotOnMousePositionToInfectious()	47
7.7.3.15.	setTypeOfDotOnMousePositionToNeutral()	47
7.7.3.16.	speedSliderChanged()	47
7.7.3.17.	startSimulationPlayer()	47
7.7.3.18.	startSimulationPlayerFromFile()	48
7.7.4.	Adattagok dokumentációja	48
7.7.4.1.	healField	48
7.7.4.2.	healSlider	48
7.7.4.3.	img	48
7.7.4.4.	infField	48
7.7.4.5.	infSlider	49
7.7.4.6.	manyDotsComboBox	49
7.7.4.7.	manyDotsField	49
7.7.4.8.	mortField	49
7.7.4.9.	mortSlider	49
7.7.4.10.	pane	49
7.7.4.11.	radius	49
7.7.4.12.	selectedType	49
7.7.4.13.	simulationTemplate	50
7.7.4.14.	speedField	50
7.7.4.15.	speedSlider	50
7.7.4.16.	stage	50
7.8.	UI.SimStatisticsController osztályreferencia	50
7.8.1.	Részletes leírás	51
7.8.2.	Konstruktorok és destruktorok dokumentációja	52
7.8.2.1.	SimStatisticsController()	52
7.8.3.	Tagfüggvények dokumentációja	52
7.8.3.1.	initialize()	52
7.8.3.2.	updateChart()	52
7.8.4.	Adattagok dokumentációja	52
7.8.4.1.	chart	53
7.8.4.2.	deaths	53

7.8.4.3.	heals	53
7.8.4.4.	infections	53
7.8.4.5.	population	53
7.8.4.6.	sss	53
7.9.	simulatorComponents.SimulationMap osztályreferencia	54
7.9.1.	Részletes leírás	54
7.9.2.	Konstruktorok és destruktorok dokumentációja	55
7.9.2.1.	SimulationMap()	55
7.9.3.	Tagfüggvények dokumentációja	55
7.9.3.1.	draw()	55
7.9.3.2.	hitBy()	55
7.9.3.3.	init()	56
7.9.3.4.	isCollidedWith() [1/2]	56
7.9.3.5.	isCollidedWith() [2/2]	56
7.9.3.6.	isOutOfWindow()	57
7.9.3.7.	moveBack()	57
7.9.3.8.	refresh()	58
7.9.3.9.	step()	58
7.10.	simulator.SimulationPlayer osztályreferencia	58
7.10.1.	Részletes leírás	60
7.10.2.	Konstruktorok és destruktorok dokumentációja	60
7.10.2.1.	SimulationPlayer()	60
7.10.3.	Tagfüggvények dokumentációja	60
7.10.3.1.	addDeadDot()	60
7.10.3.2.	addHealedDot()	61
7.10.3.3.	addInfectedDot()	61
7.10.3.4.	addSteppable()	61
7.10.3.5.	changePlayAndPause()	61
7.10.3.6.	currTickIncrease()	61
7.10.3.7.	exit()	61
7.10.3.8.	forwardOneStep()	62
7.10.3.9.	getIncubationPeriod()	62
7.10.3.10.	getRemove()	62
7.10.3.11.	getRemoveTime()	62
7.10.3.12.	moveDotsFromOutOfWindow()	62
7.10.3.13.	refresh()	63
7.10.3.14.	removeSteppable()	63
7.10.3.15.	run()	63
7.10.3.16.	sendData()	63
7.10.3.17.	speedDown()	64
7.10.3.18.	speedUp()	64
7.10.4.	Adattagok dokumentációja	64

7.10.4.1. addInTheEnd . . . . .	64
7.10.4.2. canvas . . . . .	64
7.10.4.3. currTick . . . . .	64
7.10.4.4. deadCnt . . . . .	64
7.10.4.5. healedCnt . . . . .	64
7.10.4.6. infectedCnt . . . . .	65
7.10.4.7. millisecondsElapsed . . . . .	65
7.10.4.8. minPeriod . . . . .	65
7.10.4.9. neutralCnt . . . . .	65
7.10.4.10. oneTickInMs . . . . .	65
7.10.4.11. paused . . . . .	65
7.10.4.12. removeInTheEnd . . . . .	65
7.10.4.13. sendDataPeriod . . . . .	65
7.10.4.14. sss . . . . .	66
7.10.4.15. stepables . . . . .	66
7.10.4.16. timer . . . . .	66
7.11. UI.SimulationPlayerController osztályreferencia . . . . .	66
7.11.1. Részletes leírás . . . . .	67
7.11.2. Konstruktork és destruktorok dokumentációja . . . . .	67
7.11.2.1. SimulationPlayerController() . . . . .	67
7.11.3. Tagfüggvények dokumentációja . . . . .	68
7.11.3.1. initialize() . . . . .	68
7.11.3.2. playAndPausePressed() . . . . .	68
7.11.3.3. redraw() . . . . .	68
7.11.3.4. speedDownPressed() . . . . .	68
7.11.3.5. speedUpPressed() . . . . .	68
7.11.3.6. statisticsPressed() . . . . .	69
7.11.3.7. stepPressed() . . . . .	69
7.11.4. Adattagok dokumentációja . . . . .	69
7.11.4.1. img . . . . .	69
7.11.4.2. pane . . . . .	69
7.11.4.3. reDrawCallCnt . . . . .	69
7.11.4.4. simulationPlayer . . . . .	69
7.11.4.5. simulationTemplate . . . . .	70
7.11.4.6. sss . . . . .	70
7.11.4.7. stage . . . . .	70
7.12. simulator.SimulationStatisticsStore osztályreferencia . . . . .	70
7.12.1. Részletes leírás . . . . .	71
7.12.2. Konstruktork és destruktorok dokumentációja . . . . .	71
7.12.2.1. SimulationStatisticsStore() . . . . .	71
7.12.3. Tagfüggvények dokumentációja . . . . .	71
7.12.3.1. addDeathsChange() . . . . .	71



7.12.3.2. addHealChange()	72
7.12.3.3. addInfectionChange()	72
7.12.3.4. addPopulationChange()	72
7.12.3.5. clearAll()	73
7.12.3.6. clearDeathsQueue()	73
7.12.3.7. clearHealsQueue()	73
7.12.3.8. clearInfectionsQueue()	73
7.12.3.9. clearPopulationQueue()	73
7.12.3.10. getDeathsQueue()	73
7.12.3.11. getHealsQueue()	74
7.12.3.12. getInfectionsQueue()	74
7.12.3.13. getPopulationQueue()	74
7.12.4. Adattagok dokumentációja	74
7.12.4.1. deathsQueue	74
7.12.4.2. healsQueue	75
7.12.4.3. infectionsQueue	75
7.12.4.4. populationQueue	75
7.13. simulator.SimulationTemplate osztályreferencia	75
7.13.1. Részletes leírás	76
7.13.2. Konstruktorkok és destruktorkok dokumentációja	76
7.13.2.1. SimulationTemplate() [1/2]	77
7.13.2.2. SimulationTemplate() [2/2]	77
7.13.3. Tagfüggvények dokumentációja	77
7.13.3.1. addDot()	77
7.13.3.2. clone()	77
7.13.3.3. createDot()	78
7.13.3.4. getDots()	78
7.13.3.5. getHealChance()	78
7.13.3.6. getInfChance()	79
7.13.3.7. getMortChance()	79
7.13.3.8. getSpeedOfDot()	79
7.13.3.9. refresh()	79
7.13.3.10. setHealChance()	80
7.13.3.11. setInfection()	80
7.13.3.12. setMortality()	80
7.13.3.13. setSpeed()	80
7.13.4. Adattagok dokumentációja	81
7.13.4.1. dots	81
7.13.4.2. healChance	81
7.13.4.3. infChance	81
7.13.4.4. mortChance	81
7.13.4.5. speedOfDot	81

7.14. Tests.SimulationTemplateTest osztályreferencia . . . . .	82
7.14.1. Részletes leírás . . . . .	82
7.14.2. Tagfüggvények dokumentációja . . . . .	82
7.14.2.1. createDot() . . . . .	82
7.15. simulator.Steppable interfészreferencia . . . . .	82
7.15.1. Részletes leírás . . . . .	83
7.15.2. Tagfüggvények dokumentációja . . . . .	83
7.15.2.1. draw() . . . . .	83
7.15.2.2. hitBy() . . . . .	83
7.15.2.3. init() . . . . .	83
7.15.2.4. isCollidedWith() [1/2] . . . . .	84
7.15.2.5. isCollidedWith() [2/2] . . . . .	84
7.15.2.6. isOutOfWindow() . . . . .	85
7.15.2.7. moveBack() . . . . .	85
7.15.2.8. refresh() . . . . .	85
7.15.2.9. step() . . . . .	86
<b>Tárgymutató</b>	<b>87</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.



## **2. fejezet**

# **Programozói dokumentáció**

### **2.1. Fordítási tudnivalók**



## 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.

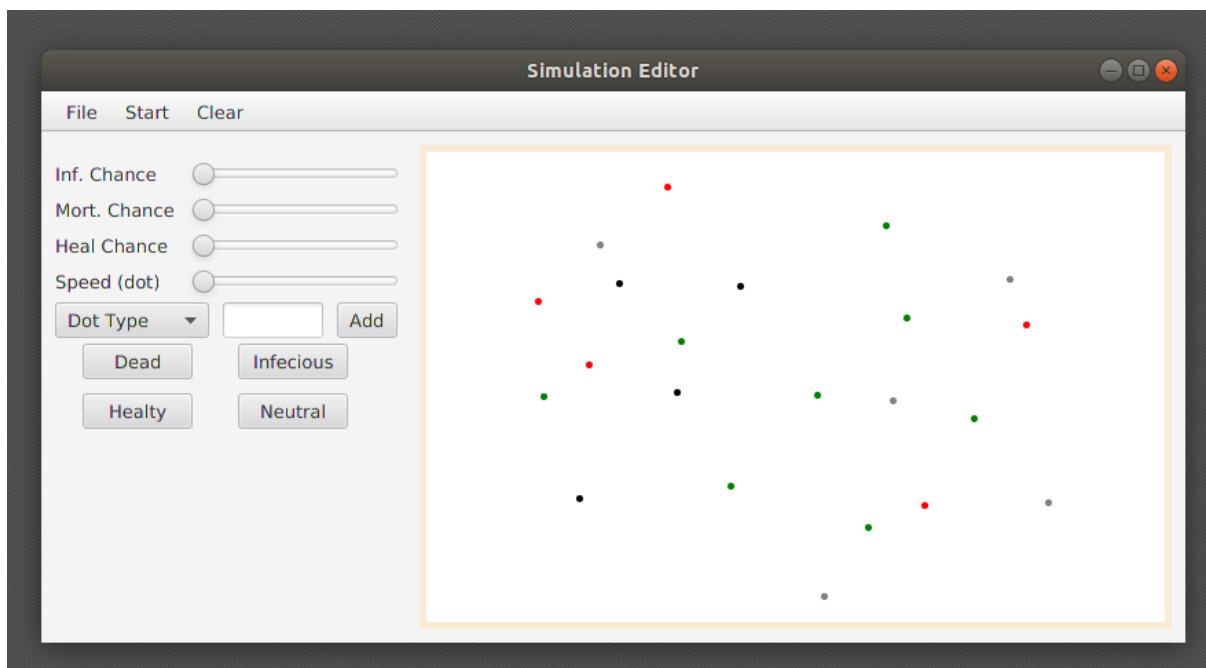
A különböző faktorokat csúszkákkal lehetne állítani: pl: pötty sebessége, milyen eséllyel fertőz, halálozási esély, gyógyulási idő stb. Ha a pötty falnak ütközik, vagy másik pöttyel, akkor visszapattan. A programhoz tartozik egy diagram is, ami a pöttyökről mutat statisztikát.

### 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 slidWer)
- 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)
- 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. (Ferőzöttek, halottak, gyógyultak, stb)



### 3.3. Megoldási ötlet (vázlat)

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

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

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

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



## 4. fejezet

# Felhasználói dokumentáció

### 4.1. A program általános leírása

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

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

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

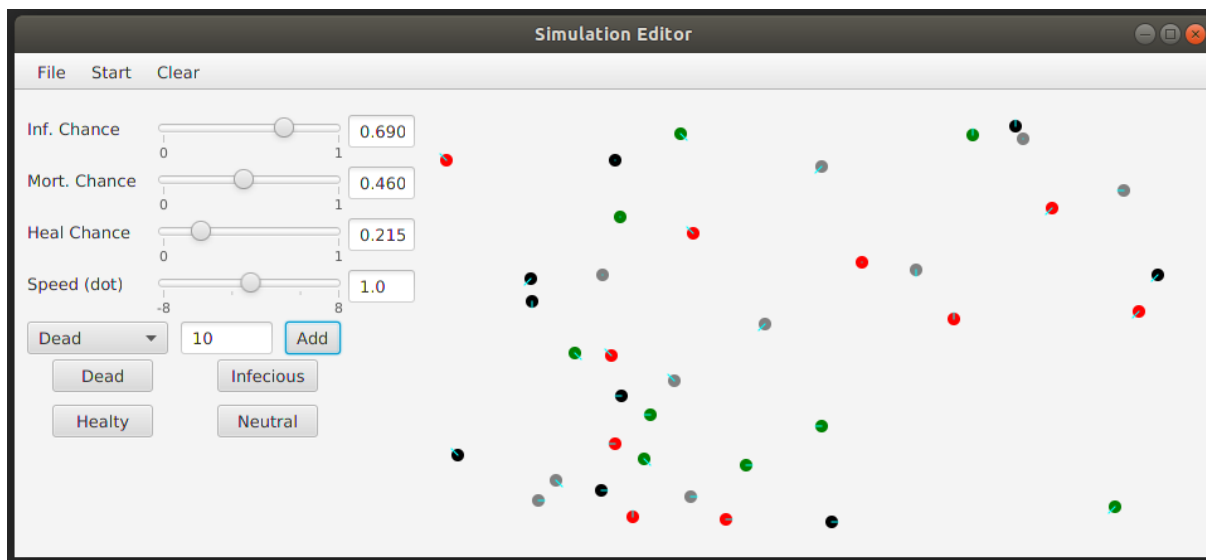
Egy fertőző pötty másokat tovább tud fertőzni, illetve a lappangási idő letelte után (500 tick), lehetősége lesz gyógyulni, illetve meghalni a beállított valószínűség alapján.

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

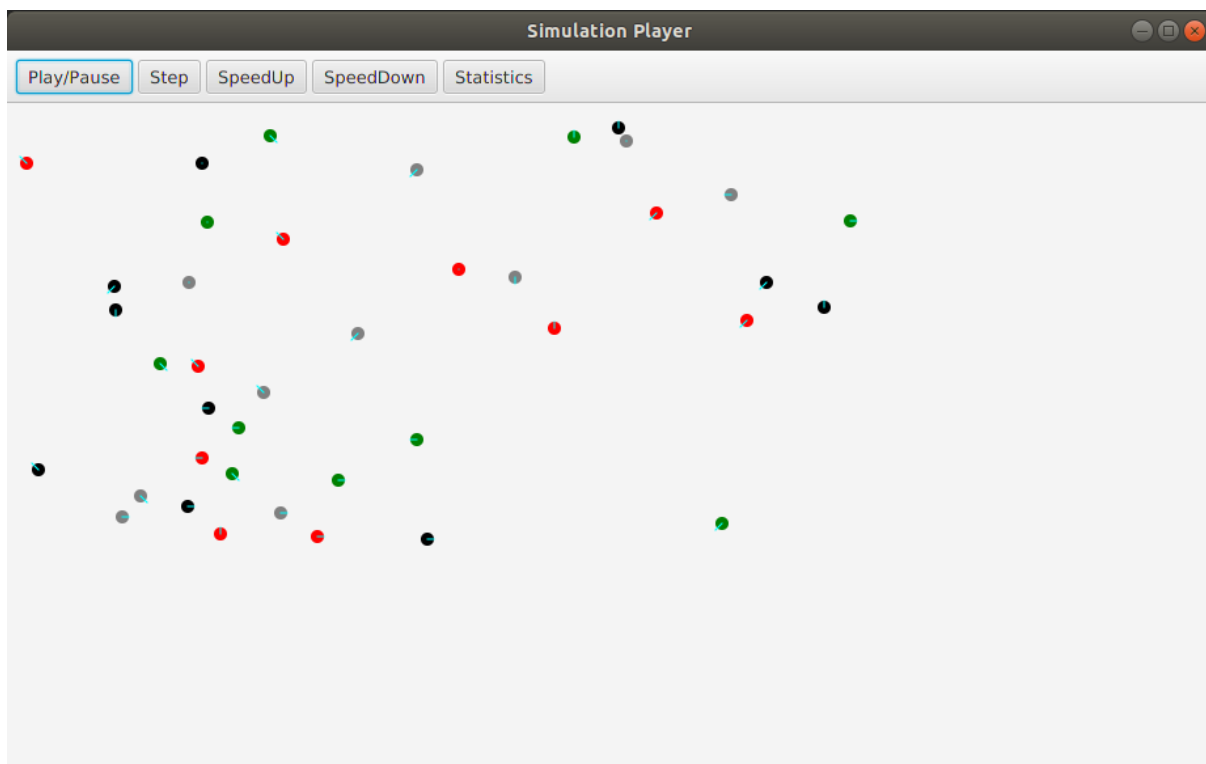


2.ábra - Szimuláció előkészítése.

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

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

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



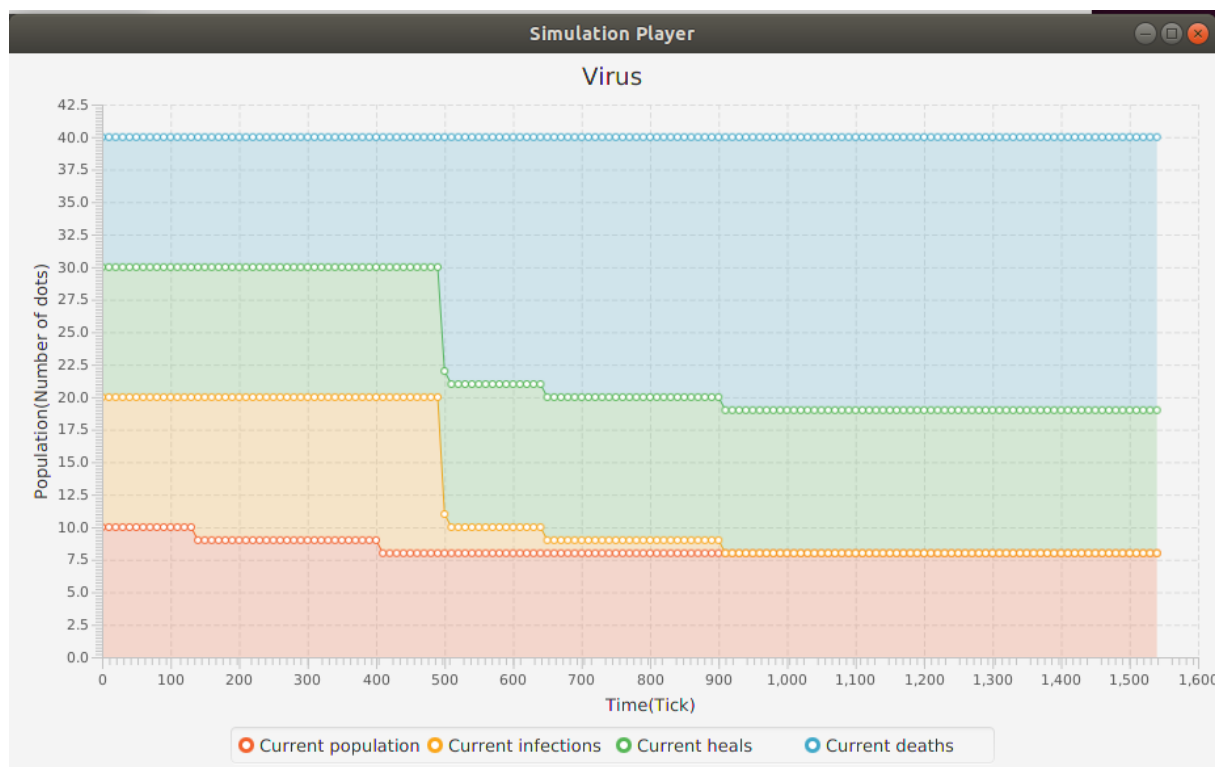
3.ábra - Vírus szimulálása

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 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 -ként frissül. Ha a statisztika bezárásra kerül, akkor az addig kapott adatok törölődnek.

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

### 3.ábra - Statisztika megjelenítése

A programrész funkciói:

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

## 5. fejezet

# Hierarchikus mutató

### 5.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

Cloneable	
simulator.SimulationTemplate . . . . .	75
simulatorComponents.Dot . . . . .	17
Tests.DotTest . . . . .	28
Tests.PointTest . . . . .	40
Serializable	
simulator.SimulationTemplate . . . . .	75
simulator.SimulationStatisticsStore . . . . .	70
Tests.SimulationTemplateTest . . . . .	82
simulator.Steppable . . . . .	82
simulatorComponents.Dot . . . . .	17
simulatorComponents.SimulationMap . . . . .	54
Application	
UI.Main . . . . .	31
Initializable	
UI.SimEditorController . . . . .	43
UI.SimStatisticsController . . . . .	50
UI.SimulationPlayerController . . . . .	66
Serializable	
simulatorComponents.Dot . . . . .	17
simulatorComponents.Point . . . . .	32
simulatorComponents.dotTypes . . . . .	29
TimerTask	
simulator.SimulationPlayer . . . . .	58





## 6. fejezet

# Osztálymutató

### 6.1. Osztálylista

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

<a href="#">simulatorComponents.Dot</a>	17
<a href="#">Tests.DotTest</a>	28
<a href="#">simulatorComponents.dotTypes</a>	29
<a href="#">UI.Main</a>	31
<a href="#">simulatorComponents.Point</a>	32
<a href="#">Tests.PointTest</a>	40
<a href="#">UI.SimEditorController</a>	43
<a href="#">UI.SimStatisticsController</a>	50
<a href="#">simulatorComponents.SimulationMap</a>	54
<a href="#">simulator.SimulationPlayer</a>	58
<a href="#">UI.SimulationPlayerController</a>	66
<a href="#">simulator.SimulationStatisticsStore</a>	70
<a href="#">simulator.SimulationTemplate</a>	75
<a href="#">Tests.SimulationTemplateTest</a>	82
<a href="#">simulator.Steppable</a>	82

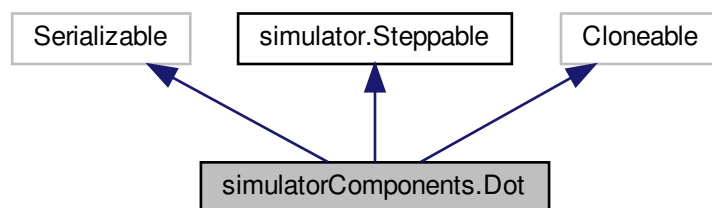


## 7. fejezet

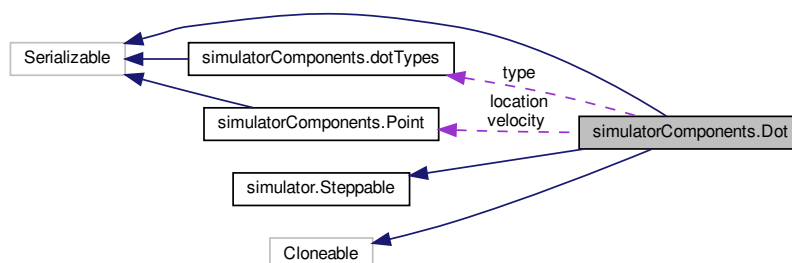
# Osztályok dokumentációja

### 7.1. simulatorComponents.Dot osztályreferencia

A simulatorComponents.Dot osztály származási diagramja:



A simulatorComponents.Dot osztály együttműködési diagramja:



## Publikus tagfüggvények

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

## Védett tagfüggvények

- void [remove](#) ()
- void [bounceBack](#) ([Canvas](#) c)

## Privát tagfüggvények

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

## Privát attribútumok

- [Point](#) [location](#) = null
- double [radius](#)
- [Point](#) [velocity](#)
- [dotTypes](#) [type](#)
- double [infChance](#)
- double [mortChance](#)
- double [healChance](#)
- double [mass](#)
- int [sinceInfection](#) = 0
- int [sinceDead](#) = 0

### 7.1.1. Részletes leírás

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

### 7.1.2. Konstruktorok és destruktorok dokumentációja

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

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

A pötty konstruktora

##### Paraméterek

<i>x</i>	A pötty x kordinátája
<i>y</i>	A pötty y kordinátája
<i>r</i>	A pötty sugara

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

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

A pötty konstruktora

##### Paraméterek

<i>x</i>	A pötty x kordinátája
<i>y</i>	A pötty y kordinátája
<i>r</i>	A pötty sugara
<i>speed</i>	A pötty sebessége

#### 7.1.2.3. Dot() [3/3]

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

```

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

```

A pötty konstruktora

#### Paraméterek

<i>x</i>	A pötty x kordinátája
<i>y</i>	A pötty y kordinátája
<i>r</i>	A pötty sugara
<i>speed</i>	A pötty sebessége
<i>type</i>	A pötty típusa
<i>infChance</i>	Másik pöttyöt ilyen eséllyel fertőz, ha a <a href="#">Dot</a> fertőző
<i>mortChance</i>	A pötty halálozási esélye, ha már megfertőződött és a vírus lappangási ideje lejárt
<i>healChance</i>	A pötty gyógyulási esélye, ha már megfertőződött és a vírus lappangási ideje lejárt

### 7.1.3. Tagfüggvények dokumentációja

#### 7.1.3.1. bounceBack()

```

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

```

A pötty a Canvas szélén visszapattan

#### Paraméterek

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

#### 7.1.3.2. clone()

```

Object simulatorComponents.Dot.clone ( ) throws CloneNotSupportedException

```

A pötty által felüldefiniált clone metódus.

#### Visszatérési érték

A pötty Object

## Kivételek

<i>CloneNotSupportedException</i>	kivételt dobhat(De nem fog, mert a <a href="#">Dot</a> szerializálható)
-----------------------------------	---

**7.1.3.3. die()**

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

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

**7.1.3.4. draw()**

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

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

## Paraméterek

<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 pötty közepe fölött megrajzolja a vectort. Debug célokra 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 pötty sugarának gettere

**Visszatérési érték**

A pötty sugara

**7.1.3.8. getType()**

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

A pötty típusának gettere

**Visszatérési érték**

A pötty típusa

**7.1.3.9. heal()**

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

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

**7.1.3.10. hitBy()**

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

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

**Paraméterek**

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



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

#### 7.1.3.11. infectedBy()

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

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

##### Paraméterek

<i>d</i>	A fertőzést okozó pötty
----------	-------------------------

#### 7.1.3.12. init()

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

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

##### Paraméterek

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

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

#### 7.1.3.13. initVelocity()

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

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

##### Paraméterek

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

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

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

A pötty ütközött-e a kapott steppable-el?

**Paraméterek**

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

**Visszatérési érték**

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

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

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

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

A pötty ütközött-e a kapott steppable-el?

**Paraméterek**

<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 pötty 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 )
```

Visszahúzza a pöttyöt a Canvas területére, ha kiment belőle

**Paraméterek**

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

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

**7.1.3.18. refresh()**

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

A pötty újrarajzolása

**Paraméterek**

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

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

**7.1.3.19. remove()**

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

A pötty hozzáadása a kör végén eltávolítandók listájához

**7.1.3.20. setHealChance()**

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

Gyógyulási esély settere

## Paraméterek

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

**7.1.3.21. setInfChance()**

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

Átfertőzései esély settere

## Paraméterek

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

**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 )
```

Halálozási esély settere

## Paraméterek

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

#### 7.1.3.24. step()

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

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

##### Paraméterek

c	A kapott Canvas
---	-----------------

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

#### 7.1.4. Adattagok dokumentációja

##### 7.1.4.1. healChance

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

A pötty gyógyulási esélye

##### 7.1.4.2. infChance

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

A pötty átfertőzésének esélye

##### 7.1.4.3. location

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

A pötty közepének pozíciója

##### 7.1.4.4. mass

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

A pötty tömege

##### 7.1.4.5. mortChance

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

A pötty halálozási esélye

#### 7.1.4.6. radius

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

A pötty sugara

#### 7.1.4.7. sinceDead

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

A halál pillanatától eltelt tickek száma

#### 7.1.4.8. sinceInfection

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

Fertőzés óta eltelt tickek száma

#### 7.1.4.9. type

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

A pötty típusa

#### 7.1.4.10. velocity

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

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

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- Dot.java

## 7.2. Tests.DotTest osztályreferencia

### Publikus tagfüggvények

- void `isCollidedWith` ()

#### 7.2.1. Részletes leírás

Dot tesztelése

## 7.2.2. Tagfüggvények dokumentációja

### 7.2.2.1. isCollidedWith()

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

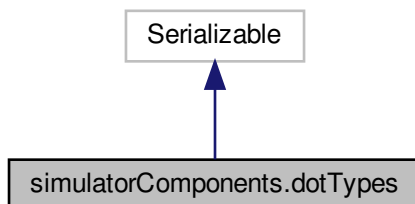
Ütközés tesztelése

Ez a dokumentáció az osztályról a következő fájl alapján készült:

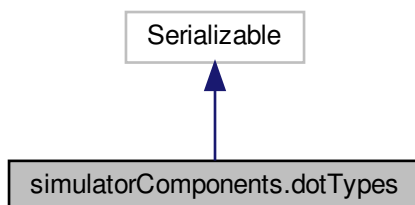
- DotTest.java

## 7.3. simulatorComponents.dotTypes felsoroló referencia

A simulatorComponents.dotTypes osztály származási diagramja:



A simulatorComponents.dotTypes osztály együttműködési diagramja:



## Publikus attribútumok

- [None](#)
- [Healthy](#)
- [Infectious](#)
- [Neutral](#)
- [Dead](#)

### 7.3.1. Részletes leírás

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

### 7.3.2. Adattagok dokumentációja

#### 7.3.2.1. Dead

```
simulatorComponents.dotTypes.Dead
```

Halott

#### 7.3.2.2. Healthy

```
simulatorComponents.dotTypes.Healthy
```

Egészséges

#### 7.3.2.3. Infectious

```
simulatorComponents.dotTypes.Infectious
```

Fertőző: Lappangási idő után meghalhat vagy meggyűgylhat

#### 7.3.2.4. Neutral

```
simulatorComponents.dotTypes.Neutral
```

Semleges: Sima személy

#### 7.3.2.5. None

```
simulatorComponents.dotTypes.None
```

Nincs: hiba kezelés miatt

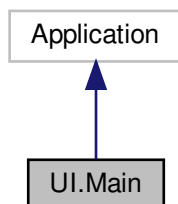
A dokumentáció ehhez az enum-hoz a következő fájl alapján készült:

- `dotTypes.java`

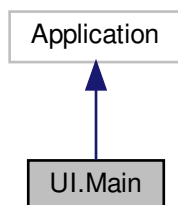


## 7.4. UI.Main osztályreferencia

Az UI.Main osztály származási diagramja:



Az UI.Main osztály együttműködési diagramja:



### Publikus tagfüggvények

- void `start` (Stage primaryStage) throws IOException

### Statikus publikus tagfüggvények

- static void `main` (String[] args)

#### 7.4.1. Részletes leírás

`Main` class. Ez a program belépési pontja.

#### 7.4.2. Tagfüggvények dokumentációja

#### 7.4.2.1. main()

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

Ez a program belépési pontja.

##### Paraméterek

<i>args</i>	parancssori argumentumok listája. Nem használok.
-------------	--

#### 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

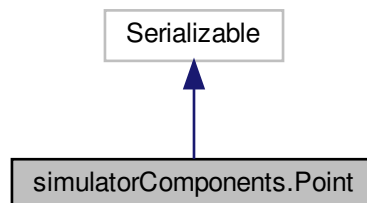
<i>IOException</i>	Akkor dobja, ha a simulationEditor.fxml nem található.
--------------------	--

Ez a dokumentáció az osztályról a következő fájl alapján készült:

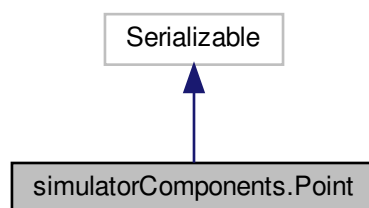
- Main.java

## 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 tárolása és ezeken műveletek végzése Abban az esetben, ha vektort fejez ki, akkor a Pont az Origóba 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 kordináta
<i>y</i>	A kapott y kordináta

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

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

[Point](#) osztály 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 hozzáad egy X és Y értéket

Paraméterek

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

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

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

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

#### Paraméterek

$p$	A kapott Pont
-----	---------------

### 7.5.3.3. calcDisplacement()

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

Két pont közötti elmozdulás számolása

#### Paraméterek

$p$	A kapott Pont
-----	---------------

#### Visszatérési érték

Az elmozdulás vector

### 7.5.3.4. calcDistance()

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

Két pont között számol távolságot, távolság =  $((x_1-x_2)^2+(y_1-y_2)^2)^{(1/2)}$  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 és Y kordinátáját elosztja az értékkel

##### Paraméterek

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

#### 7.5.3.6. dotProduct()

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

Skaláris szorzat számolását végzi két vector között

##### Paraméterek

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

##### Visszatérési érték

A kiszámolt skaláris szorzatot visszaadja.

#### 7.5.3.7. getX()

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

X getter

##### Visszatérési érték

x értéke

#### 7.5.3.8. getY()

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

y getter

##### Visszatérési érték

y értéke

### 7.5.3.9. isOutOfCanvas()

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

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

#### Paraméterek

<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-ról

#### 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-ról

#### 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-ról

**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-ról

**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 kordinátáját megszorozza a kapott értékkel



## 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 kordináták 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 kordináta a bal felső sarokban van a (0,0)

#### 7.5.4.2. y

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

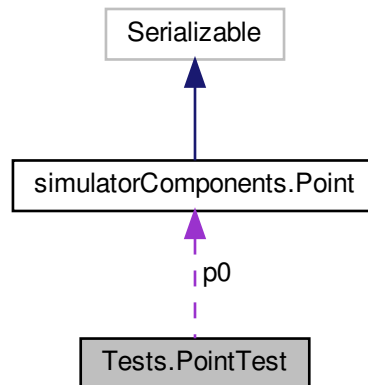
y kordináta a bal felső sarokban van a (0,0)

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- Point.java

## 7.6. Tests.PointTest osztályreferencia

A Tests.PointTest osztály együttműködési diagramja:



### Publikus tagfüggvények

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

### Csomag attribútumok

- `Point p0`

### 7.6.1. Részletes leírás

Point tesztelésére szolgál

### 7.6.2. Tagfüggvények dokumentációja

#### 7.6.2.1. add()

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

add teszt

#### 7.6.2.2. calcDisplacement()

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

calcDisplacement teszt

#### 7.6.2.3. calcDistance()

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

calcDistance teszt

#### 7.6.2.4. divide()

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

divide teszt

#### 7.6.2.5. divideByZero()

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

divide by Zero teszt

#### 7.6.2.6. dotProduct()

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

dotProduct teszt

#### 7.6.2.7. multiply()

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

multiply teszt

#### 7.6.2.8. setUp()

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

Create (0,0)

#### 7.6.2.9. subtract()

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

subtract teszt

### 7.6.3. Adattagok dokumentációja

#### 7.6.3.1. p0

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

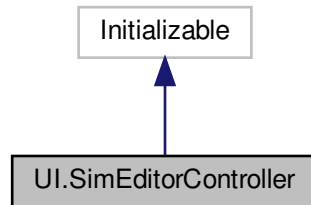
Origó

Ez a dokumentáció az osztályról a következő fájl alapján készült:

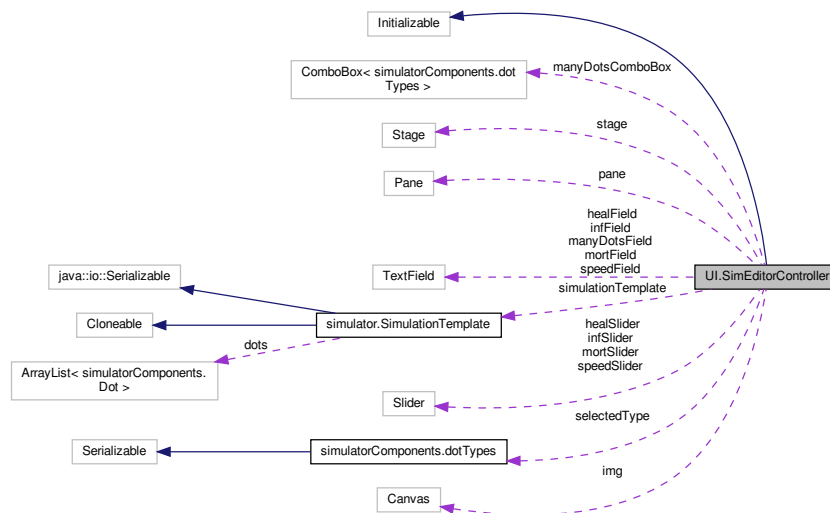
- PointTest.java

## 7.7. UI.SimEditorController osztályreferencia

Az UI.SimEditorController osztály származási diagramja:



Az UI.SimEditorController osztály együttműködési diagramja:



### Publikus tagfüggvények

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

### Privát tagfüggvények

- void [redraw](#) ()
- void [addManyDotsPressed](#) ()
- void [infSliderChanged](#) ()
- void [mortalitySliderChanged](#) ()

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

## Privát attribútumok

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

### 7.7.1. Részletes leírás

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

### 7.7.2. Konstruktorok és destruktorok dokumentációja

#### 7.7.2.1. SimEditorController()

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

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

## Paraméterek

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

#### 7.7.3.2. clearCanvas()

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

Canvas letörlését végzi. Új SimulationTemplate-et hoz létre.

#### 7.7.3.3. createDotOnMousePosition()

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

Egér kattintáskor meghívja a Dot létrehoz 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 inicializálója. 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

### 7.7.3.7. mortalitySliderChanged()

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

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

### 7.7.3.8. openSerializedSimulationTemplate()

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

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

### 7.7.3.9. openSimulationTemplate()

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

Deszerializálja a megadott simulationTemplate-et.

#### Visszatérési érték

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

### 7.7.3.10. redraw()

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

Feladata az ablak újrarajzolása.



**7.7.3.11. serializeSimulationTemplate()**

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

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

**7.7.3.12. setTypeOfDotOnMousePositionToDead()**

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

Beállítja az egér által lehelyezendő Dot típusát halottra.

**7.7.3.13. setTypeOfDotOnMousePositionToHealthy()**

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

Beállítja az egér által lehelyezendő Dot típusát egészségesre.

**7.7.3.14. setTypeOfDotOnMousePositionToInfectious()**

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

Beállítja az egér által lehelyezendő Dot típusát fertőzőre.

**7.7.3.15. setTypeOfDotOnMousePositionToNeutral()**

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

Beállítja az egér által lehelyezendő Dot típusát közömbösre.

**7.7.3.16. speedSliderChanged()**

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

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

**7.7.3.17. startSimulationPlayer()**

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

Létrehozza és elindítja a SimulationPlayer ablakot.

**Kivételek**

<i>IOException</i>	Kivételt dob, ha az fxml nem létezik.
--------------------	---------------------------------------

**7.7.3.18. startSimulationPlayerFromFile()**

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

Elindítja a SimulationPlayer-t közvetlenül fájlból.

**Kivételek**

<i>IOException</i>	Kivételt dob, ha az fxml nem létezik.
--------------------	---------------------------------------

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

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

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

**7.7.4.2. healSlider**

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

Slider, a gyógyulási esély beállítására.

**7.7.4.3. img**

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

Canvas, a Dot-ok jelennek meg.

**7.7.4.4. infField**

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

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

#### 7.7.4.5. infSlider

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

Slider, az átfertőzés esélyének beállítására.

#### 7.7.4.6. manyDotsComboBox

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

ComboBox, a User itt választja ki a Dot típusát

#### 7.7.4.7. manyDotsField

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

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

#### 7.7.4.8. mortField

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

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

#### 7.7.4.9. mortSlider

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

Slider, a halálozási esély beállítására.

#### 7.7.4.10. pane

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

Pane, ebben található a canvas, amire rajzolunk.

#### 7.7.4.11. radius

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

Pötty alapértelmezett sugara.

#### 7.7.4.12. selectedType

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

Gomb által kiválasztott Dot típusának tárolója.

#### 7.7.4.13. simulationTemplate

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

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

#### 7.7.4.14. speedField

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

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

#### 7.7.4.15. speedSlider

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

Slider, a Dot sebességének beállítására.

#### 7.7.4.16. stage

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

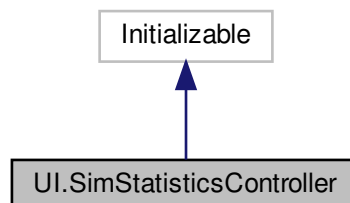
Stage eltárolása. Konstruktorban kapjuk az ablak létrehozása során. Ez alapján pozícionáljuk a többi ablakot.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

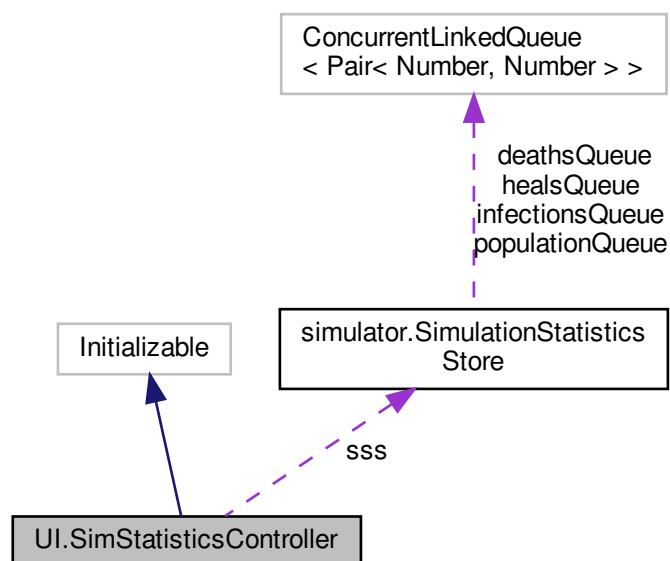
- SimEditorController.java

## 7.8. UI.SimStatisticsController osztályreferencia

Az UI.SimStatisticsController osztály származási diagramja:



Az UI.SimStatisticsController osztály együttműködési diagramja:



### Publikus tagfüggvények

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

### Csomag attribútumok

- [SimulationStatisticsStore](#) sss

### Privát attribútumok

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

#### 7.8.1. Részletes leírás

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

## 7.8.2. Konstruktorkok és destruktorkok 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 inicializálója. 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

### 7.8.3.2. updateChart()

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

Frissíti a grafikonon megjelenő adatokat

## 7.8.4. Adattagok dokumentációja

#### 7.8.4.1. chart

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

A kirajzolandó grafikus

#### 7.8.4.2. deaths

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

halálozási adatokat tároló XYChart Series

#### 7.8.4.3. heals

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

gyógyulási adatokat tároló XYChart Series

#### 7.8.4.4. infections

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

fertőzési adatokat tároló XYChart Series

#### 7.8.4.5. population

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

lakossági adatokat tároló XYChart Series

#### 7.8.4.6. sss

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

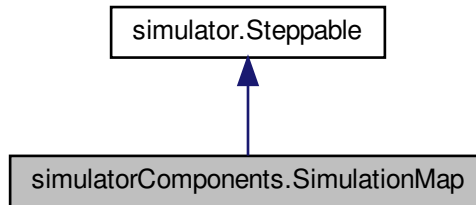
Közös store a simulationPlayerrel

Ez a dokumentáció az osztályról a következő fájl alapján készült:

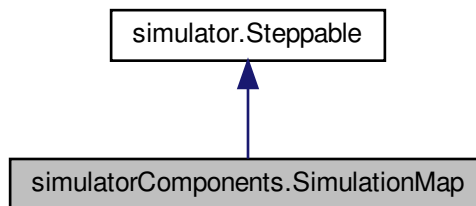
- SimStatisticsController.java

## 7.9. simulatorComponents.SimulationMap osztályreferencia

A simulatorComponents.SimulationMap osztály származási diagramja:



A simulatorComponents.SimulationMap osztály együttműködési diagramja:



### Publikus tagfüggvények

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

### 7.9.1. Részletes leírás

[SimulationMap](#) osztály A pályát, jelképezi. Feladata, hogy kör elején letörli magát



## 7.9.2. Konstruktorkok és destruktorkok dokumentációja

### 7.9.2.1. SimulationMap()

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

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

#### Paraméterek

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

## 7.9.3. Tagfüggvények dokumentációja

### 7.9.3.1. draw()

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

Letörli 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](#) Pályával ütközni, de a léptethetőség miatt szükséges.

#### 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 ( Tehát letörli magát)

##### Paraméterek

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

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

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

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

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

##### Paraméterek

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

##### Visszatérési érték

Mindig Hamis

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

#### 7.9.3.5. isCollidedWith() [2/2]

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

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

## Paraméterek

st	A kapott Másik Steppable
----	--------------------------

## Visszatérési érték

Mindig Hamis

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

**7.9.3.6. isOutOfWindow()**

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

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

## Paraméterek

c	A kapott Canvas
---	-----------------

## Visszatérési érték

Mindig Hamis

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

**7.9.3.7. moveBack()**

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

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

## Paraméterek

c	A kapott Canvas
---	-----------------

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

#### 7.9.3.8. refresh()

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

Frissíti a Canvas-t. Meghívja a [draw\(Canvas\)](#)-t

##### Paraméterek

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

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

#### 7.9.3.9. step()

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

Lépés során meghívja saját maga refresh függvényét.

##### Paraméterek

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

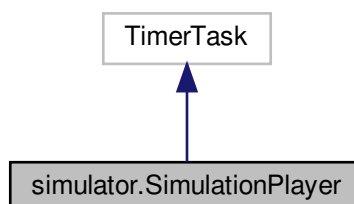
Megvalósítja a következőket: [simulator.Steppable](#).

Ez a dokumentáció az osztályról a következő fájl alapján készült:

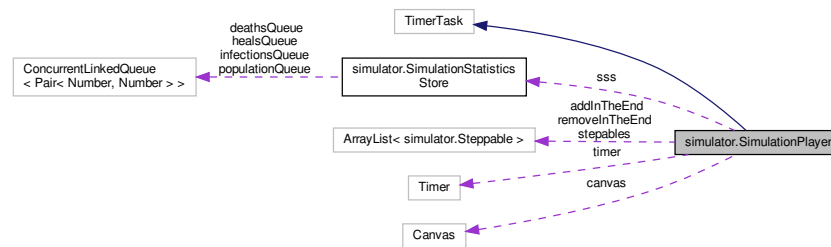
- SimulationMap.java

## 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 szimuláció lejátszásához szükséges adatokat és függvényeket tárolja.

### 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 tároló, ami pufferként funkcionál

### 7.10.3. Tagfüggvények dokumentációja

#### 7.10.3.1. [addDeadDot](#)()

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

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

#### 7.10.3.2. addHealedDot()

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

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

#### 7.10.3.3. addInfectedDot()

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

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

#### 7.10.3.4. addSteppable()

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

Hozzáadja a [Steppable](#) dolgot a hozzáadandóak listájához

Paraméterek

<i>st</i>	A hozzáadandó <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 lépteti a kört

#### 7.10.3.7. exit()

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

Kilépéskor lezárja és üríti a szükséges dolgokat

#### 7.10.3.8. forwardOneStep()

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

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

#### 7.10.3.9. getIncubationPeriod()

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

Lappangási időt visszatérő függvény

##### Visszatérési érték

Lappangási idő Tickeken mérve

#### 7.10.3.10. getRemove()

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

Viszadja az eltávolítandók listáját

##### Visszatérési érték

Az eltávolítandók listája

#### 7.10.3.11. getRemoveTime()

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

Halott Dot eltűnési idejét határozza meg

##### Visszatérési érték

Eltűnési idő Tickeken mérve

#### 7.10.3.12. moveDotsFromOutOfWindow()

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

A Canvasról kilógó 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 szimuláció tartalmát Az első [Steppable](#) mindig a pálya maga

## Paraméterek

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

**7.10.3.14. removeSteppable()**

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

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

## Paraméterek

<i>st</i>	Az eltávolítandó <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 érzett idő telését

#### 7.10.3.18. speedUp()

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

Gyorsítja a felhasználó által érzett idő telését

### 7.10.4. Adattagok dokumentációja

#### 7.10.4.1. addInTheEnd

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

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

#### 7.10.4.2. canvas

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

A canvas, amire rajzolunk

#### 7.10.4.3. currTick

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

Éppen aktuális kör sorszáma

#### 7.10.4.4. deadCnt

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

Éppen halott Dotok száma

#### 7.10.4.5. healedCnt

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

Éppen egészséges Dotok száma

**7.10.4.6. infectedCnt**

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

Éppen fertőző Dotok száma

**7.10.4.7. millisecondsElapsed**

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

Szimuláció kezdete óta eltelt idő

**7.10.4.8. minPeriod**

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

Két kör között eltelt minimális periódusidő

**7.10.4.9. neutralCnt**

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

Éppen semleges Dotok száma

**7.10.4.10. oneTickInMs**

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

Egy Tick Ms-ban mérve

**7.10.4.11. paused**

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

Le van-e szüneteltetve a szimuláció

**7.10.4.12. removeInTheEnd**

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

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

**7.10.4.13. sendDataPeriod**

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

Adatküldés gyakorisága

#### 7.10.4.14. sss

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

Szimuláció Statisztikát tároló osztály Pufferként funkcionál

#### 7.10.4.15. stepables

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

Léptethető dolgok tárolója

#### 7.10.4.16. timer

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

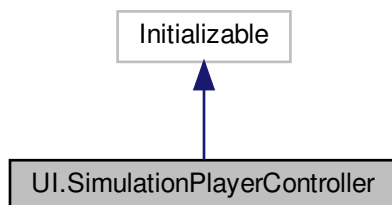
Időzítő, a körönként történő lépésért felel

Ez a dokumentáció az osztályról a következő fájl alapján készült:

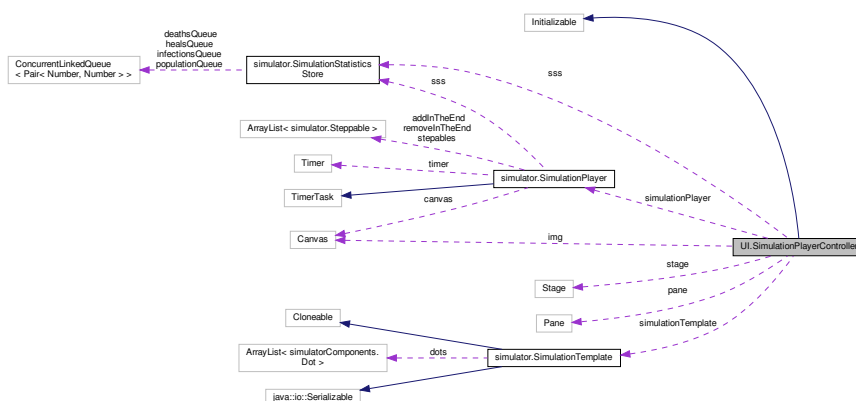
- `SimulationPlayer.java`

## 7.11. UI.SimulationPlayerController osztályreferencia

Az `UI.SimulationPlayerController` osztály származási diagramja:



Az `UI.SimulationPlayerController` osztály együttműködési diagramja:



## Publikus tagfüggvények

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

## Csomag függvények

- [SimulationPlayerController](#) (Stage st, [SimulationTemplate](#) sim)

## Csomag attribútumok

- Stage [stage](#)
- [SimulationPlayer](#) [simulationPlayer](#)
- [SimulationTemplate](#) [simulationTemplate](#)
- [SimulationStatisticsStore](#) [sss](#)

## Privát tagfüggvények

- void [redraw](#) ()

## Privát attribútumok

- Canvas [img](#)
- Pane [pane](#)
- int [reDrawCallCnt](#) = 0

### 7.11.1. Részletes leírás

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

### 7.11.2. Konstruktorok és destruktork 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 inicializálója. 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>	kivételt dobhat, de ha az fxm1 fájl jó helyen van nem fog
--------------------	---

### 7.11.3.7. stepPressed()

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

Step gomb megnyomásának kezelése

## 7.11.4. Adattagok dokumentációja

### 7.11.4.1. img

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

Canvas amire rajzolunk

### 7.11.4.2. pane

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

Pane, ebben található a canvas, amire rajzolunk.

### 7.11.4.3. reDrawCallCnt

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

Hányszor hívtuk meg a redraw függvényt. reDraw működéséhez szükséges

### 7.11.4.4. simulationPlayer

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

simulationPlayer-t tárol

#### 7.11.4.5. simulationTemplate

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

Futatandó szimuláció adatait tárolja

#### 7.11.4.6. sss

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

Statisztika puffer tára

#### 7.11.4.7. stage

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

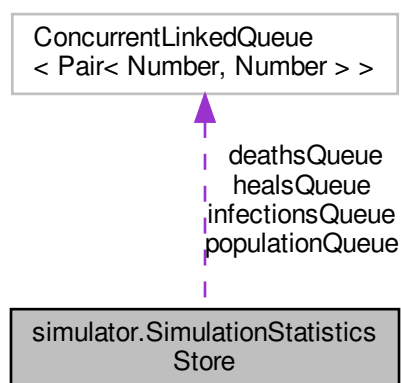
Stage eltárolása. Konstruktorban kapjuk az ablak létrehozása során. Ez alapján pozícionáljuk a többi ablakot.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- `SimulationPlayerController.java`

### 7.12. simulator.SimulationStatisticsStore osztályreferencia

A `simulator.SimulationStatisticsStore` osztály együttműködési diagramja:





## Publikus tagfüggvények

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

## Csomag attribútumok

- ConcurrentLinkedQueue< Pair< Number, Number > > [populationQueue](#) = new 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 tároló a statisztikai adatok tárolására

### 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 )
```

Hozzáad egy új értéket a deathsQueue-hoz

## Paraméterek

<i>time</i>	Időbélyeg (Tick-ben)
<i>val</i>	Az aktuális érték

**7.12.3.2. addHealChange()**

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

Hozzáad egy új értéket a healsQueue-hoz

## Paraméterek

<i>time</i>	Időbélyeg (Tick-ben)
<i>val</i>	Az aktuális érték

**7.12.3.3. addInfectionChange()**

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

Hozzáad egy új értéket a infectionsQueue-hoz

## Paraméterek

<i>time</i>	Időbélyeg (Tick-ben)
<i>val</i>	Az aktuális érték

**7.12.3.4. addPopulationChange()**

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

Hozzáad egy új értéket a populationQueue-hoz

## Paraméterek

<i>time</i>	Időbélyeg (Tick-ben)
<i>val</i>	Az aktuális érték

**7.12.3.5. clearAll()**

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

Üríti az összes Queue-t

**7.12.3.6. clearDeathsQueue()**

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

Üríti a deathsQueue-t

**7.12.3.7. clearHealsQueue()**

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

Üríti a healsQueue-t

**7.12.3.8. clearInfectionsQueue()**

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

Üríti az infectionsQueue-t

**7.12.3.9. clearPopulationQueue()**

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

Üríti a populationQueue-t

**7.12.3.10. getDeathsQueue()**

```
ConcurrentLinkedListQueue<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]
```

halálozási adatok puffere

#### 7.12.4.2. healsQueue

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

gyógyulási adatok puffere

#### 7.12.4.3. infectionsQueue

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

fertőzési adatok puffere

#### 7.12.4.4. populationQueue

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

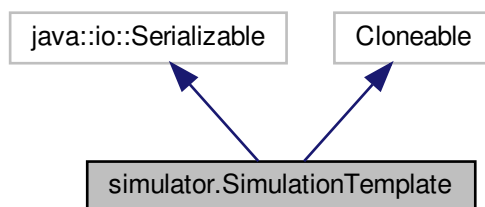
population adatok puffere. Azért erre esett a választásom, mert így a Concurrent Modification Exception-t el tudom kerülni, mert történhet olyan, hogy éppen olvasom a Puffert, mikor a végére már kerül az új adat

Ez a dokumentáció az osztályról a következő fájl alapján készült:

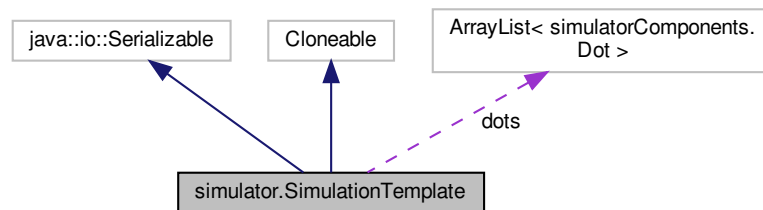
- SimulationStatisticsStore.java

### 7.13. simulator.SimulationTemplate osztályreferencia

A simulator.SimulationTemplate osztály származási diagramja:



A simulator.SimulationTemplate osztály együttműködési diagramja:



## Publikus tagfüggvények

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

## Privát attribútumok

- ArrayList< [Dot](#) > [dots](#)
- double [infChance](#)
- double [mortChance](#)
- double [healChance](#)
- double [speedOfDot](#)

### 7.13.1. Részletes leírás

[SimulationTemplate](#) osztály. Szimuláció elindításához kapcsolatos adatokat tárol. Megvalósítja a java.io.Serializable és Cloneable interfészeket

### 7.13.2. Konstruktorkok és destruktorkok dokumentációja

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

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

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

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

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

Simulation template copy konstruktora

Paraméterek

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

### 7.13.3. Tagfüggvények dokumentációja

#### 7.13.3.1. addDot()

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

Hozzáad egy Dot-ot a Dotokat tartalmazó listához

Paraméterek

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

#### 7.13.3.2. clone()

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

clone függvény felüldefiniálása. Célja, hogy a [SimulationTemplate](#) másolását megvalósítsa.

Visszatérési érték

Visszadja az objektum másolatát(deep copy)

## Kivételek

<i>CloneNotSupportedException</i>	kivételt dobhat (de nem fog, mert a dot és a <a href="#">SimulationTemplate</a> is klónozzható)
-----------------------------------	---

**7.13.3.3. createDot()**

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

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

## Paraméterek

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

Fertőzési esély getter függvénye

**Visszatérési érték**

A fertőzési esély

**7.13.3.7. getMortChance()**

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

Halálozási esély getter függvénye

**Visszatérési érték**

A kapott halálozási esély

**7.13.3.8. getSpeedOfDot()**

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

A sebesség getter függvénye

**Visszatérési érték**

A sebessége a Dot-nak

**7.13.3.9. refresh()**

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

Frissíti a Canvas tartalmát

**Paraméterek**

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

#### 7.13.3.10. setHealChance()

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

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

##### Paraméterek

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

#### 7.13.3.11. setInfection()

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

Fertőzőési esély setter függvénye Beállítja a kapott fertőzőési esélyt a jövőben létrejövő Dot-okra.

##### Paraméterek

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

#### 7.13.3.12. setMortality()

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

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

##### Paraméterek

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

#### 7.13.3.13. setSpeed()

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

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

## Paraméterek

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

### 7.13.4. Adattagok dokumentációja

#### 7.13.4.1. dots

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

Dot-okat tároló ArrayList.

#### 7.13.4.2. healChance

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

Gyógyulási esély

#### 7.13.4.3. infChance

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

Átfertőzési esély

#### 7.13.4.4. mortChance

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

Halálozási esély

#### 7.13.4.5. speedOfDot

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

Dot sebessége

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- SimulationTemplate.java

## 7.14. Tests.SimulationTemplateTest osztályreferencia

### Publikus tagfüggvények

- void `createDot` ()

#### 7.14.1. Részletes leírás

Simulation Template osztály tesztje

#### 7.14.2. Tagfüggvények dokumentációja

##### 7.14.2.1. createDot()

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

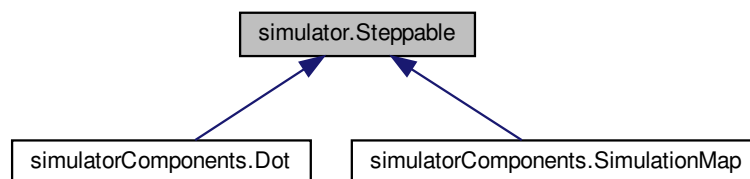
create Dot tesztje

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- SimulationTemplateTest.java

## 7.15. simulator.Steppable interfészreferencia

A simulator.Steppable osztály származási diagramja:



### Publikus tagfüggvények

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

### 7.15.1. Részletes leírás

Interfész, amely a léptethető dolgokat valósítja meg

### 7.15.2. Tagfüggvények dokumentációja

#### 7.15.2.1. draw()

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

Kirajzolja a léptethető dolgot a Canvason

Paraméterek

<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 ütközést 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 )
```

Ütközött-e a kapott Dot-al?

## Paraméterek

<i>dot</i>	A kapott Másik Dot
------------	--------------------

## Visszatérési érték

Igen vagy Nem

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

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

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

Ütközött-e a kapott Steppable-el?

## Paraméterek

<i>st</i>	A kapott Másik <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 léptethető dolog a Canvason kívül tartózkodik-e?

##### Paraméterek

c	A kapott Canvas
---	-----------------

##### Visszatérési érték

Igen vagy Nem

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

#### 7.15.2.7. moveBack()

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

Visszahúzza a léptethető dolgot a Canvasra

##### Paraméterek

c	A kapott Canvas
---	-----------------

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

#### 7.15.2.8. refresh()

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

Frissíti a léptethető dolgot a Canvason

##### Paraméterek

c	A kapott Canvas
---	-----------------

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

#### 7.15.2.9. step()

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

Egy körben végzendő lépések

##### Paraméterek

<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, [34](#), [35](#)
  - Tests.PointTest, [41](#)
- addDeadDot
  - simulator.SimulationPlayer, [60](#)
- addDeathsChange
  - simulator.SimulationStatisticsStore, [71](#)
- addDot
  - simulator.SimulationTemplate, [77](#)
- addHealChange
  - simulator.SimulationStatisticsStore, [72](#)
- addHealedDot
  - simulator.SimulationPlayer, [61](#)
- addInfectedDot
  - simulator.SimulationPlayer, [61](#)
- addInfectionChange
  - simulator.SimulationStatisticsStore, [72](#)
- addInTheEnd
  - simulator.SimulationPlayer, [64](#)
- addManyDotsPressed
  - UI.SimEditorController, [45](#)
- addPopulationChange
  - simulator.SimulationStatisticsStore, [72](#)
- addSteppable
  - simulator.SimulationPlayer, [61](#)
- bounceBack
  - simulatorComponents.Dot, [20](#)
- calcDisplacement
  - simulatorComponents.Point, [35](#)
  - Tests.PointTest, [41](#)
- calcDistance
  - simulatorComponents.Point, [35](#)
  - Tests.PointTest, [41](#)
- canvas
  - simulator.SimulationPlayer, [64](#)
- changePlayAndPause
  - simulator.SimulationPlayer, [61](#)
- chart
  - UI.SimStatisticsController, [52](#)
- clearAll
  - simulator.SimulationStatisticsStore, [73](#)
- clearCanvas
  - UI.SimEditorController, [45](#)
- clearDeathsQueue
  - simulator.SimulationStatisticsStore, [73](#)
- clearHealsQueue
  - simulator.SimulationStatisticsStore, [73](#)
- clearInfectionsQueue
  - simulator.SimulationStatisticsStore, [73](#)
- clearPopulationQueue
  - simulator.SimulationStatisticsStore, [73](#)
- clone
  - simulator.SimulationTemplate, [77](#)
  - simulatorComponents.Dot, [20](#)
- createDot
  - simulator.SimulationTemplate, [78](#)
  - Tests.SimulationTemplateTest, [82](#)
- createDotOnMousePosition
  - UI.SimEditorController, [45](#)
- currTick
  - simulator.SimulationPlayer, [64](#)
- currTickIncrease
  - simulator.SimulationPlayer, [61](#)
- Dead
  - simulatorComponents.dotTypes, [30](#)
- deadCnt
  - simulator.SimulationPlayer, [64](#)
- deaths
  - UI.SimStatisticsController, [53](#)
- deathsQueue
  - simulator.SimulationStatisticsStore, [74](#)
- die
  - simulatorComponents.Dot, [21](#)
- divide
  - simulatorComponents.Point, [35](#)
  - Tests.PointTest, [41](#)
- divideByZero
  - Tests.PointTest, [41](#)
- Dot
  - simulatorComponents.Dot, [19](#)
- dotProduct
  - simulatorComponents.Point, [36](#)
  - Tests.PointTest, [41](#)
- dots
  - simulator.SimulationTemplate, [81](#)
- draw
  - simulator.Steppable, [83](#)
  - simulatorComponents.Dot, [21](#)
  - simulatorComponents.SimulationMap, [55](#)
- drawCenters
  - simulatorComponents.Dot, [21](#)
- exit
  - simulator.SimulationPlayer, [61](#)
- forwardOneStep

- simulator.SimulationPlayer, 61
- getDeathsQueue
  - simulator.SimulationStatisticsStore, 73
- getDots
  - simulator.SimulationTemplate, 78
- getHealChance
  - simulator.SimulationTemplate, 78
- getHealsQueue
  - simulator.SimulationStatisticsStore, 73
- getIncubationPeriod
  - simulator.SimulationPlayer, 62
- getInfChance
  - simulator.SimulationTemplate, 78
- getInfectionsQueue
  - simulator.SimulationStatisticsStore, 74
- getLocation
  - simulatorComponents.Dot, 21
- getMortChance
  - simulator.SimulationTemplate, 79
- getPopulationQueue
  - simulator.SimulationStatisticsStore, 74
- getRadius
  - simulatorComponents.Dot, 22
- getRemove
  - simulator.SimulationPlayer, 62
- getRemoveTime
  - simulator.SimulationPlayer, 62
- getSpeedOfDot
  - simulator.SimulationTemplate, 79
- getType
  - simulatorComponents.Dot, 22
- getX
  - simulatorComponents.Point, 36
- getY
  - simulatorComponents.Point, 36
- heal
  - simulatorComponents.Dot, 22
- healChance
  - simulator.SimulationTemplate, 81
  - simulatorComponents.Dot, 27
- healedCnt
  - simulator.SimulationPlayer, 64
- healField
  - UI.SimEditorController, 48
- heals
  - UI.SimStatisticsController, 53
- healSlider
  - UI.SimEditorController, 48
- healSliderChanged
  - UI.SimEditorController, 45
- healsQueue
  - simulator.SimulationStatisticsStore, 74
- Healthy
  - simulatorComponents.dotTypes, 30
- hitBy
  - simulator.Steppable, 83
  - simulatorComponents.Dot, 22
- simulatorComponents.SimulationMap, 55
- img
  - UI.SimEditorController, 48
  - UI.SimulationPlayerController, 69
- infChance
  - simulator.SimulationTemplate, 81
  - simulatorComponents.Dot, 27
- infectedBy
  - simulatorComponents.Dot, 23
- infectedCnt
  - simulator.SimulationPlayer, 64
- infections
  - UI.SimStatisticsController, 53
- infectionsQueue
  - simulator.SimulationStatisticsStore, 75
- Infectious
  - simulatorComponents.dotTypes, 30
- infField
  - UI.SimEditorController, 48
- infSlider
  - UI.SimEditorController, 48
- infSliderChanged
  - UI.SimEditorController, 45
- init
  - simulator.Steppable, 83
  - simulatorComponents.Dot, 23
  - simulatorComponents.SimulationMap, 56
- initialize
  - UI.SimEditorController, 45
  - UI.SimStatisticsController, 52
  - UI.SimulationPlayerController, 68
- initVelocity
  - simulatorComponents.Dot, 23
- isCollidedWith
  - simulator.Steppable, 84
  - simulatorComponents.Dot, 23, 24
  - simulatorComponents.SimulationMap, 56
  - Tests.DotTest, 29
- isOutOfCanvas
  - simulatorComponents.Point, 36
- isOutOfCanvasBottom
  - simulatorComponents.Point, 37
- isOutOfCanvasLeft
  - simulatorComponents.Point, 37
- isOutOfCanvasRight
  - simulatorComponents.Point, 38
- isOutOfCanvasTop
  - simulatorComponents.Point, 38
- isOutOfWindow
  - simulator.Steppable, 84
  - simulatorComponents.Dot, 24
  - simulatorComponents.SimulationMap, 57
- location
  - simulatorComponents.Dot, 27
- main
  - UI.Main, 31

- manyDotsComboBox
  - UI.SimEditorController, [49](#)
- manyDotsField
  - UI.SimEditorController, [49](#)
- mass
  - simulatorComponents.Dot, [27](#)
- millisecondsElapsed
  - simulator.SimulationPlayer, [65](#)
- minPeriod
  - simulator.SimulationPlayer, [65](#)
- mortalitySliderChanged
  - UI.SimEditorController, [46](#)
- mortChance
  - simulator.SimulationTemplate, [81](#)
  - simulatorComponents.Dot, [27](#)
- mortField
  - UI.SimEditorController, [49](#)
- mortSlider
  - UI.SimEditorController, [49](#)
- moveBack
  - simulator.Steppable, [85](#)
  - simulatorComponents.Dot, [25](#)
  - simulatorComponents.SimulationMap, [57](#)
- moveDotsFromOutOfWindow
  - simulator.SimulationPlayer, [62](#)
- multiply
  - simulatorComponents.Point, [38](#)
  - Tests.PointTest, [41](#)
- Neutral
  - simulatorComponents.dotTypes, [30](#)
- neutralCnt
  - simulator.SimulationPlayer, [65](#)
- None
  - simulatorComponents.dotTypes, [30](#)
- oneTickInMs
  - simulator.SimulationPlayer, [65](#)
- openSerializedSimulationTemplate
  - UI.SimEditorController, [46](#)
- openSimulationTemplate
  - UI.SimEditorController, [46](#)
- p0
  - Tests.PointTest, [42](#)
- pane
  - UI.SimEditorController, [49](#)
  - UI.SimulationPlayerController, [69](#)
- paused
  - simulator.SimulationPlayer, [65](#)
- playAndPausePressed
  - UI.SimulationPlayerController, [68](#)
- Point
  - simulatorComponents.Point, [34](#)
- population
  - UI.SimStatisticsController, [53](#)
- populationQueue
  - simulator.SimulationStatisticsStore, [75](#)
- radius
  - simulatorComponents.Dot, [27](#)
  - UI.SimEditorController, [49](#)
- redraw
  - UI.SimEditorController, [46](#)
  - UI.SimulationPlayerController, [68](#)
- reDrawCallCnt
  - UI.SimulationPlayerController, [69](#)
- refresh
  - simulator.SimulationPlayer, [63](#)
  - simulator.SimulationTemplate, [79](#)
  - simulator.Steppable, [85](#)
  - simulatorComponents.Dot, [25](#)
  - simulatorComponents.SimulationMap, [57](#)
- remove
  - simulatorComponents.Dot, [25](#)
- removeInTheEnd
  - simulator.SimulationPlayer, [65](#)
- removeSteppable
  - simulator.SimulationPlayer, [63](#)
- run
  - simulator.SimulationPlayer, [63](#)
- selectedType
  - UI.SimEditorController, [49](#)
- sendData
  - simulator.SimulationPlayer, [63](#)
- sendDataPeriod
  - simulator.SimulationPlayer, [65](#)
- serializeSimulationTemplate
  - UI.SimEditorController, [46](#)
- setHealChance
  - simulator.SimulationTemplate, [79](#)
  - simulatorComponents.Dot, [25](#)
- setInfChance
  - simulatorComponents.Dot, [26](#)
- setInfection
  - simulator.SimulationTemplate, [80](#)
- setLocation
  - simulatorComponents.Dot, [26](#)
- setMortality
  - simulator.SimulationTemplate, [80](#)
- setMortChance
  - simulatorComponents.Dot, [26](#)
- setSpeed
  - simulator.SimulationTemplate, [80](#)
- setTypeOfDotOnMousePositionToDead
  - UI.SimEditorController, [47](#)
- setTypeOfDotOnMousePositionToHealthy
  - UI.SimEditorController, [47](#)
- setTypeOfDotOnMousePositionToInfectious
  - UI.SimEditorController, [47](#)
- setTypeOfDotOnMousePositionToNeutral
  - UI.SimEditorController, [47](#)
- setUp
  - Tests.PointTest, [42](#)
- SimEditorController
  - UI.SimEditorController, [44](#)
- SimStatisticsController

- UI.SimStatisticsController, 52
- SimulationMap
  - simulatorComponents.SimulationMap, 55
- SimulationPlayer
  - simulator.SimulationPlayer, 60
- simulationPlayer
  - UI.SimulationPlayerController, 69
- SimulationPlayerController
  - UI.SimulationPlayerController, 67
- SimulationStatisticsStore
  - simulator.SimulationStatisticsStore, 71
- SimulationTemplate
  - simulator.SimulationTemplate, 76, 77
- simulationTemplate
  - UI.SimEditorController, 49
  - UI.SimulationPlayerController, 69
- simulator.SimulationPlayer, 58
  - addDeadDot, 60
  - addHealedDot, 61
  - addInfectedDot, 61
  - addInTheEnd, 64
  - addSteppable, 61
  - canvas, 64
  - changePlayAndPause, 61
  - currTick, 64
  - currTickIncrease, 61
  - deadCnt, 64
  - exit, 61
  - forwardOneStep, 61
  - getIncubationPeriod, 62
  - getRemove, 62
  - getRemoveTime, 62
  - healedCnt, 64
  - infectedCnt, 64
  - millisecondsElapsed, 65
  - minPeriod, 65
  - moveDotsFromOutOfWindow, 62
  - neutralCnt, 65
  - oneTickInMs, 65
  - paused, 65
  - refresh, 63
  - removeInTheEnd, 65
  - removeSteppable, 63
  - run, 63
  - sendData, 63
  - sendDataPeriod, 65
  - SimulationPlayer, 60
  - speedDown, 63
  - speedUp, 64
  - sss, 65
  - stepables, 66
  - timer, 66
- simulator.SimulationStatisticsStore, 70
  - addDeathsChange, 71
  - addHealChange, 72
  - addInfectionChange, 72
  - addPopulationChange, 72
  - clearAll, 73
  - clearDeathsQueue, 73
  - clearHealsQueue, 73
  - clearInfectionsQueue, 73
  - clearPopulationQueue, 73
  - deathsQueue, 74
  - getDeathsQueue, 73
  - getHealsQueue, 73
  - getInfectionsQueue, 74
  - getPopulationQueue, 74
  - healsQueue, 74
  - infectionsQueue, 75
  - populationQueue, 75
  - SimulationStatisticsStore, 71
- simulator.SimulationTemplate, 75
  - addDot, 77
  - clone, 77
  - createDot, 78
  - dots, 81
  - getDots, 78
  - getHealChance, 78
  - getInfChance, 78
  - getMortChance, 79
  - getSpeedOfDot, 79
  - healChance, 81
  - infChance, 81
  - mortChance, 81
  - refresh, 79
  - setHealChance, 79
  - setInfection, 80
  - setMortality, 80
  - setSpeed, 80
  - SimulationTemplate, 76, 77
  - speedOfDot, 81
- simulator.Steppable, 82
  - draw, 83
  - hitBy, 83
  - init, 83
  - isCollidedWith, 84
  - isOutOfWindow, 84
  - moveBack, 85
  - refresh, 85
  - step, 85
- simulatorComponents.Dot, 17
  - bounceBack, 20
  - clone, 20
  - die, 21
  - Dot, 19
  - draw, 21
  - drawCenters, 21
  - getLocation, 21
  - getRadius, 22
  - getType, 22
  - heal, 22
  - healChance, 27
  - hitBy, 22
  - infChance, 27
  - infectedBy, 23
  - init, 23

- initVelocity, [23](#)
- isCollidedWith, [23](#), [24](#)
- isOutOfWindow, [24](#)
- location, [27](#)
- mass, [27](#)
- mortChance, [27](#)
- moveBack, [25](#)
- radius, [27](#)
- refresh, [25](#)
- remove, [25](#)
- setHealChance, [25](#)
- setInfChance, [26](#)
- setLocation, [26](#)
- setMortChance, [26](#)
- sinceDead, [28](#)
- sinceInfection, [28](#)
- step, [26](#)
- type, [28](#)
- velocity, [28](#)
- simulatorComponents.dotTypes, [29](#)
  - Dead, [30](#)
  - Healthy, [30](#)
  - Infectious, [30](#)
  - Neutral, [30](#)
  - None, [30](#)
- simulatorComponents.Point, [32](#)
  - add, [34](#), [35](#)
  - calcDisplacement, [35](#)
  - calcDistance, [35](#)
  - divide, [35](#)
  - dotProduct, [36](#)
  - getX, [36](#)
  - getY, [36](#)
  - isOutOfCanvas, [36](#)
  - isOutOfCanvasBottom, [37](#)
  - isOutOfCanvasLeft, [37](#)
  - isOutOfCanvasRight, [38](#)
  - isOutOfCanvasTop, [38](#)
  - multiply, [38](#)
  - Point, [34](#)
  - subtract, [39](#)
  - x, [39](#)
  - y, [39](#)
- simulatorComponents.SimulationMap, [54](#)
  - draw, [55](#)
  - hitBy, [55](#)
  - init, [56](#)
  - isCollidedWith, [56](#)
  - isOutOfWindow, [57](#)
  - moveBack, [57](#)
  - refresh, [57](#)
  - SimulationMap, [55](#)
  - step, [58](#)
- sinceDead
  - simulatorComponents.Dot, [28](#)
- sinceInfection
  - simulatorComponents.Dot, [28](#)
- speedDown
  - simulator.SimulationPlayer, [63](#)
- speedDownPressed
  - UI.SimulationPlayerController, [68](#)
- speedField
  - UI.SimEditorController, [50](#)
- speedOfDot
  - simulator.SimulationTemplate, [81](#)
- speedSlider
  - UI.SimEditorController, [50](#)
- speedSliderChanged
  - UI.SimEditorController, [47](#)
- speedUp
  - simulator.SimulationPlayer, [64](#)
- speedUpPressed
  - UI.SimulationPlayerController, [68](#)
- sss
  - simulator.SimulationPlayer, [65](#)
  - UI.SimStatisticsController, [53](#)
  - UI.SimulationPlayerController, [70](#)
- stage
  - UI.SimEditorController, [50](#)
  - UI.SimulationPlayerController, [70](#)
- start
  - UI.Main, [32](#)
- startSimulationPlayer
  - UI.SimEditorController, [47](#)
- startSimulationPlayerFromFile
  - UI.SimEditorController, [48](#)
- statisticsPressed
  - UI.SimulationPlayerController, [68](#)
- step
  - simulator.Steppable, [85](#)
  - simulatorComponents.Dot, [26](#)
  - simulatorComponents.SimulationMap, [58](#)
- stepables
  - simulator.SimulationPlayer, [66](#)
- stepPressed
  - UI.SimulationPlayerController, [69](#)
- subtract
  - simulatorComponents.Point, [39](#)
  - Tests.PointTest, [42](#)
- Tests.DotTest, [28](#)
  - isCollidedWith, [29](#)
- Tests.PointTest, [40](#)
  - add, [41](#)
  - calcDisplacement, [41](#)
  - calcDistance, [41](#)
  - divide, [41](#)
  - divideByZero, [41](#)
  - dotProduct, [41](#)
  - multiply, [41](#)
  - p0, [42](#)
  - setUp, [42](#)
  - subtract, [42](#)
- Tests.SimulationTemplateTest, [82](#)
  - createDot, [82](#)
- timer
  - simulator.SimulationPlayer, [66](#)

- type
  - simulatorComponents.Dot, 28
- UI.Main, 31
  - main, 31
  - start, 32
- UI.SimEditorController, 43
  - addManyDotsPressed, 45
  - clearCanvas, 45
  - createDotOnMousePosition, 45
  - healField, 48
  - healSlider, 48
  - healSliderChanged, 45
  - img, 48
  - infField, 48
  - infSlider, 48
  - infSliderChanged, 45
  - initialize, 45
  - manyDotsComboBox, 49
  - manyDotsField, 49
  - mortalitySliderChanged, 46
  - mortField, 49
  - mortSlider, 49
  - openSerializedSimulationTemplate, 46
  - openSimulationTemplate, 46
  - pane, 49
  - radius, 49
  - redraw, 46
  - selectedType, 49
  - serializeSimulationTemplate, 46
  - setTypeOfDotOnMousePositionToDead, 47
  - setTypeOfDotOnMousePositionToHealthy, 47
  - setTypeOfDotOnMousePositionToInfectious, 47
  - setTypeOfDotOnMousePositionToNeutral, 47
  - SimEditorController, 44
  - simulationTemplate, 49
  - speedField, 50
  - speedSlider, 50
  - speedSliderChanged, 47
  - stage, 50
  - startSimulationPlayer, 47
  - startSimulationPlayerFromFile, 48
- UI.SimStatisticsController, 50
  - chart, 52
  - deaths, 53
  - heals, 53
  - infections, 53
  - initialize, 52
  - population, 53
  - SimStatisticsController, 52
  - sss, 53
  - updateChart, 52
- UI.SimulationPlayerController, 66
  - img, 69
  - initialize, 68
  - pane, 69
  - playAndPausePressed, 68
  - redraw, 68
  - reDrawCallCnt, 69
  - simulationPlayer, 69
  - SimulationPlayerController, 67
  - simulationTemplate, 69
  - speedDownPressed, 68
  - speedUpPressed, 68
  - sss, 70
  - stage, 70
  - statisticsPressed, 68
  - stepPressed, 69
- updateChart
  - UI.SimStatisticsController, 52
- velocity
  - simulatorComponents.Dot, 28
- x
  - simulatorComponents.Point, 39
- y
  - simulatorComponents.Point, 39