Theseus and the Minotaur

Gyömbér Péter

PIM313

Feladat rövid ismertetése (user manual)

A játék a jól ismert minotaurusz mítoszát igyekszik reprezentálni. A játékos egy karaktert, Theseus-t irányítja a labirintusban, azzal a céllal, hogy megölje az ott élő minotauruszt. A minotaurusz véletlenszerűen mászkál a labirintus belsejében, a játékosnak először meg kell találnia, majd meg is kell ölni azt. A megölés során lövedékekkel kell eltalálnia az ellenséget, míg az meg nem hal. A a minotaurusz nekimegy Theseus-nak, az eleterej lecsökken. Ha minden élet elfogyott, a játékos elveszti a játékot. Theseust különféle képességerősítők is segítik, melyek időnként megjelennek e labirintusban. A játék menürendszerrel is rendelkezik, valamint a játék állását is el lehet menteni, illetve visszatölteni azt.

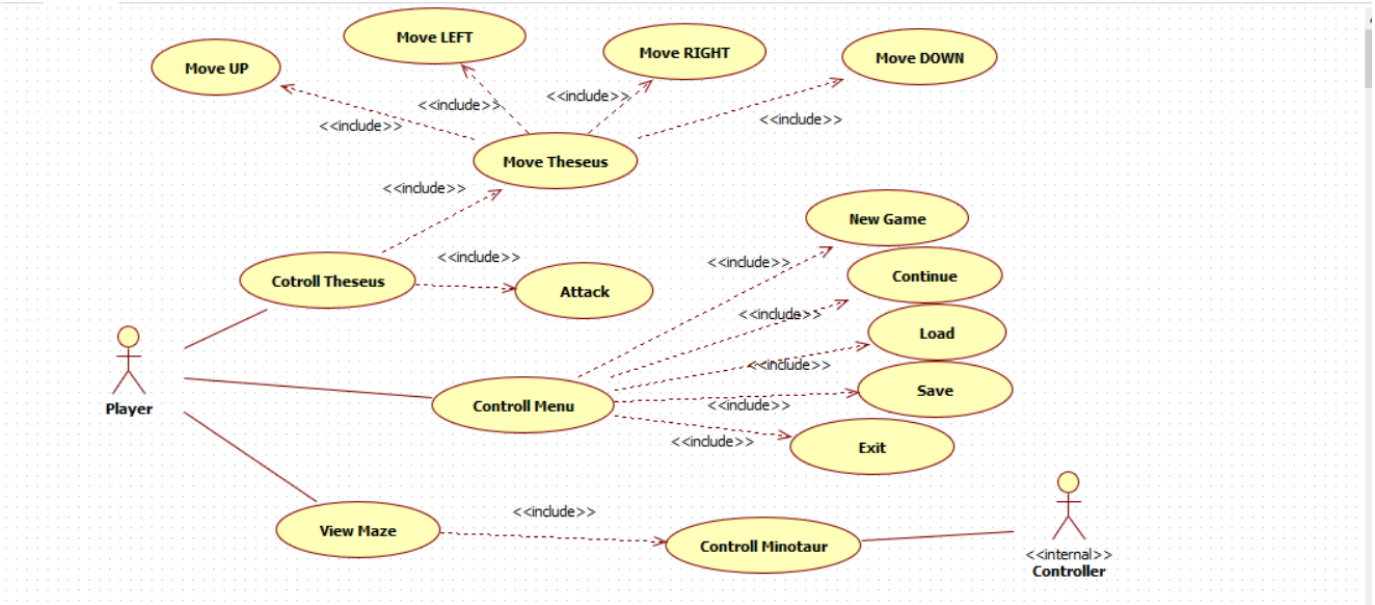
Bővítések a Specifikációhoz képest

- Ha a minotaurusz nekimegy Theseusnak, nem hal meg rögtön, csupán 1-el lecsökken az élete (kezdetben 3 van).
- Theseust képességerősítők is segítik, melyek véletlenszerűen jelennek meg a labirintusban.
- Theseus egy fonalat húz maga után, ami a az útvonalát mutatja, igyekezve ezzel a játék hangulát a mitószhoz közelíteni.
- A menü gobjai nem Jbutton-ök, hanem sajat készítésű gombok.
- Ha valamelyik karakter meghal, az a mozgását elveszti, de a többi dolog a labirintusban még tovább működik, tetszőlegesen indítható új játék.

Use-case-ek

A játékos négy irányba (föl, le, balra, jobbra) tudja mozgatni a karakterét. Ezt a „w, a, s, d” billentyűk lenyomásával teheti meg. Továbbá a „space” billentyű lenyomásával lőni is képes. A játékos természetesen a labirintust is meg tudja tekinteni, mely egy grafikus felület segítségével jelenítődik meg. A menüt az „escape” billentyű lenyomásával hozhatja be. Lehetősége van új játékot indítani, folytathatja az aktuálisan futó játékot, elmentheti, illetve visszatöltheti a legutoljára mentett állást. Egy további „exit” gomb is lehetőségére áll, a játék befejezésére. Az egyes opciók közöl az egér segítségével lehet választani.

Use-case Diagramm



Osztályok

Az osztályok részletes leírása megtalálható a forráskódban kommentek szintjén. Minden osztály előtt szerepel annak a feladata és egyéb tervezői ötletek is. A beszédes változó- és függvénynevek mellett továvábbá azok részletes leírása is dokumentálva van. Ebből kifolyólag az alábbiakban csak az osztályok nevei, valamint azok feladatai vannak felsorolva. A könnyebb áttekinthetés érdekében az osztályok változói és metódusai szándékosan nem szerepelnek.

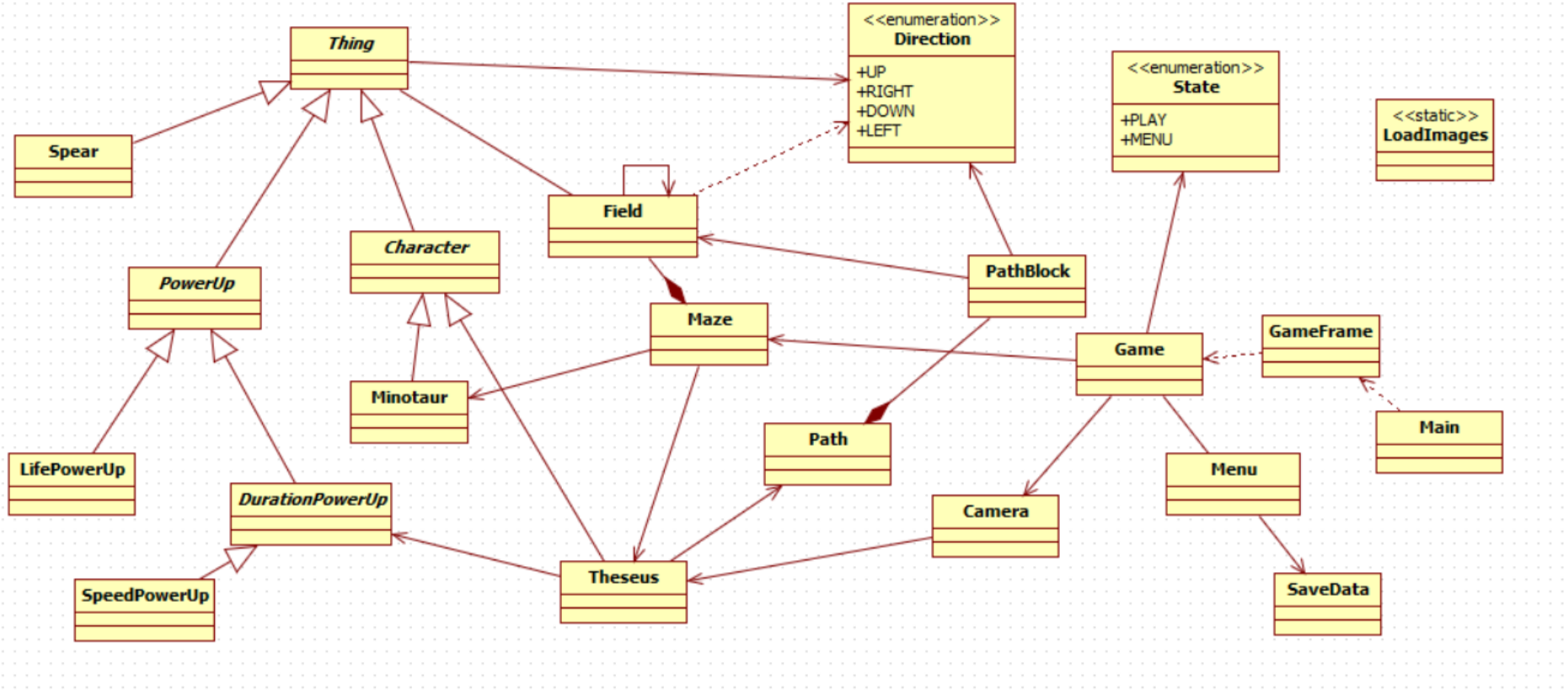
- Main
  - Létrehoz egy GameFrame-et, majd a megadott méretekkel futtatja rajta a Játékot.
- LoadImages
  - A képek betöltéséért felel. Tartalmazza továbbá a játékban előforduló egységes, fix méretet is. Változói elérhetőek a többi osztályból.
- SaveData
  - A mentés megvalosításáért felel. A Labirintust és a benne levő egyes Dolgokat Stringek mátrxává alakítja, majd ebből állítja vissza. Implementálja a Serializable-t, melynek segítségével a fájlba írást és olvasást valosítja meg. A kódolás fontosságát az egyszerű Serializalas közben felmerülő problémak (pl.: túl nagy méret) teszik indokolttá.
- Game
  - A játék motorja, mely kezeli a felhasználói inputokat és a játék állapotait. Folyamatosan frissíti az egyes Dolgokat és kirajzolj azokat a képernyőre. JPanel-ből öröklődik így a GameFrame-hez hozzáadható. Implementálja a

KeyListener és ActionListener interfészeket, melyek billentyűlenyomasokért és az időzítő használhatóságáért szükségesek. A Játék osztály tárolja a Labirintust, a Kamerát, a Menüt, az egyes irányokat és a játék állapotát.

- Menu
  - A Játék Menüje. Segítségével indítható új játék, menthető el az aktuális, illetve tölthető be a legutoljára mentett. Tartalmazza a Game-et, amihez tartozik, továbbá egy SaveData-t, ami a mentéshez szükséges. Implementálja a MouseListenert, mivel egérkattintással választhatunk a Menü opciói közül.
- State
  - A játék két különböző állapotát definiálja.
- Camera
  - A kamera felelős azért, hogy a játékban éppen mely obiektumok jelenjenek meg a képernyőn.
- GameFrame
  - A program ablaka, ezert a JFrame-ből öröklődik. Fix szélessége és magassága van, mely a többi osztály számara is látható, és egy saját címe.
- Direction
  - A játékban előforduló négy különböző iránytípust (balra, jobbra, fel, le) definiálja.
- Field
  - A játékban egy Mezőt reprezentál. A labirintus Mezőkből épül fel, melyeken az egyes Dolgok elhelyezkedhetnek. A Mező lehet fal (ha fal, lehet növényes vagy sima), tárolja a négy különböző irányban lévő szomszédját (ha van), a sajat (X,Y) koordinátaját és a rajta lévő Dolgok listáját.
- Maze
  - A Labirintus Mezőkből épül fel, benne mozognak az egyes Dolgok.
- Path
  - A játékban egy Ösvényt (útvonalat) reprezentál. Tárolja a megfelelő Mezőket a megadott sorrendben, illetve azt is, hogy azt melyik irányból jövet érték el. Ehhez a PathBlock osztályt használja fel, ami képes az elemeket egyszerűbben kezelni.
- PathBlock
  - A Path egy blokkja. Rendelkezik egy Mezővel és egy Iránnyal.
- Thing
  - A játékban lévő Dolgok abstract osztálya. Minden Dolog ebből szarmazik, így rendelkezik a Dolog tulajdonságaival, amit a játékban elvárunk.
- Character
  - A Karakter egy abstract osztály, ami szintén leszármazik az abstract Dolog osztályból. Feladata, hogy a ténylegesen élő egységeket együtt lehessen kezelni, illetve könnyen bövíteni azokat. A legtobb atributumát örokli, de kibővül az eleterő és isAlive változókkal.
- Minotaur

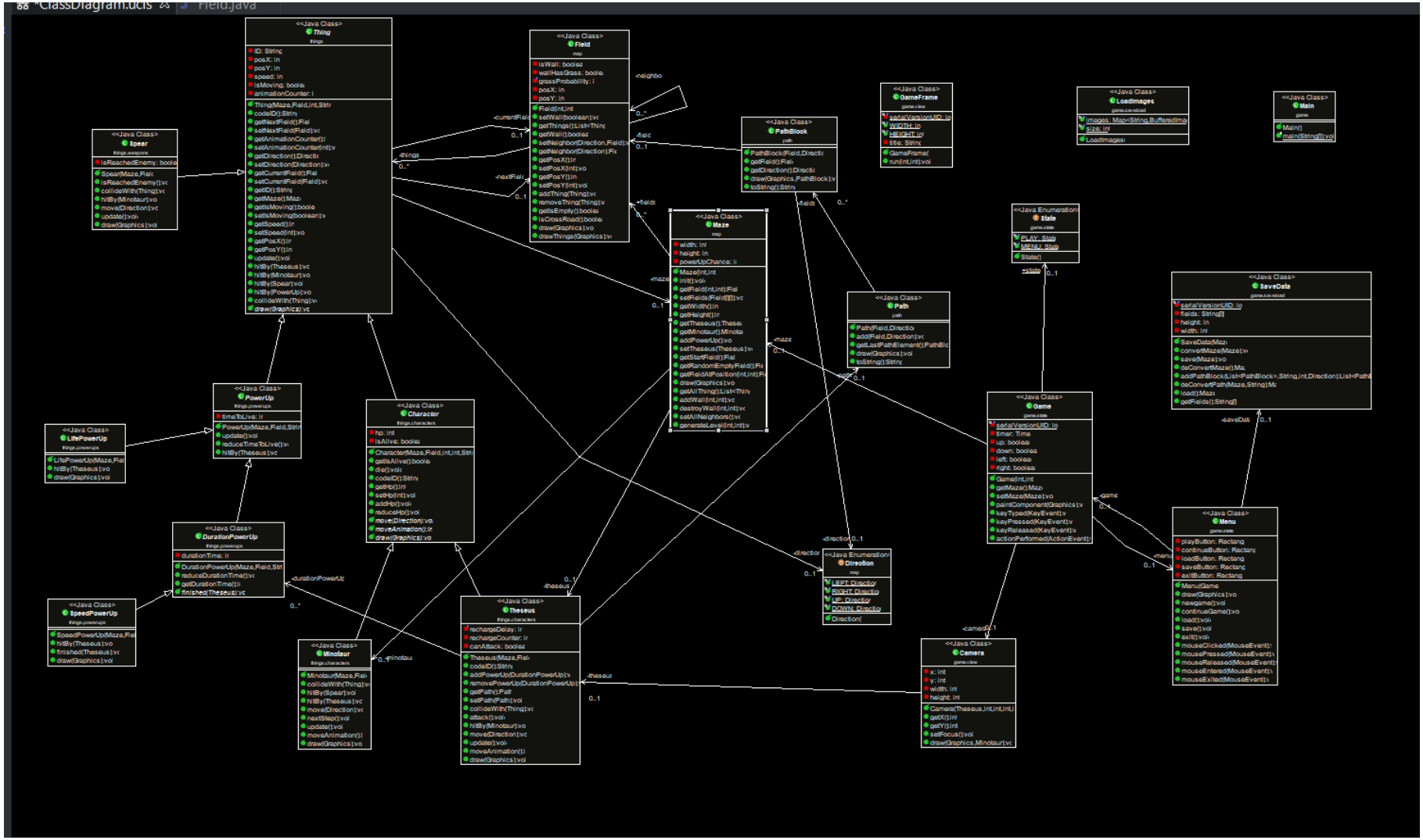
    - A Minotaurusz egy Karakter a játékbn, ezert örökődik Karakterből. A Minotaurusz Thesesu ellensége, őt kell megölni a játék megnyeréséhez.
- Theseus
    - A játékos Theseust iranyítja a Labirintusban. A Karakter leszármazottja. Theseus képes mozogni, illetve Dárdát hajítani. Célja, hogy megölje a Minotauruszt, de közben vigyáznia kell, hisz meg is halhat tőle.
- DurationPowerUp
    - Egy abstract osztály, minden DurationPowerUp belőle származik le. A PowerUp-ból öröklődik, azzal bővítve azt, hogy a hatása csak egy bizonyos ideig érvényes, utana lejár.
- LifePowerUp
    - Theseus életerejét képes növelni. A PowerUp-bol öröklődik.
- PowerUp
    - A PowerUp egy abstarct osztály, ami minden képességnövelő Dolgot magábafoglal. Leszármazottja a Dolog ősosztálynak.
- SpeedPowerUp
    - Theseus sebességét képes növelni egy bizonyos ideig. Leszármazottja a DurationPowerUp-nak.
- Spear
    - Egy Dárdát reprezentál a játékban. Theseus hozhatja létre, melynek hatására elkezd repülni, míg valaminek neki nem ütközik. Leszármazottja a Dolog ősosztálynak.

Az osztályok, és azok legfontosabb kapcsolataik



Megjegyzés: Az ábrán a LoadImages a láthatóság kedvéért nincs összekötve semelyik osztállyal sem, hiszen ezt mindegyik osztály eléri.

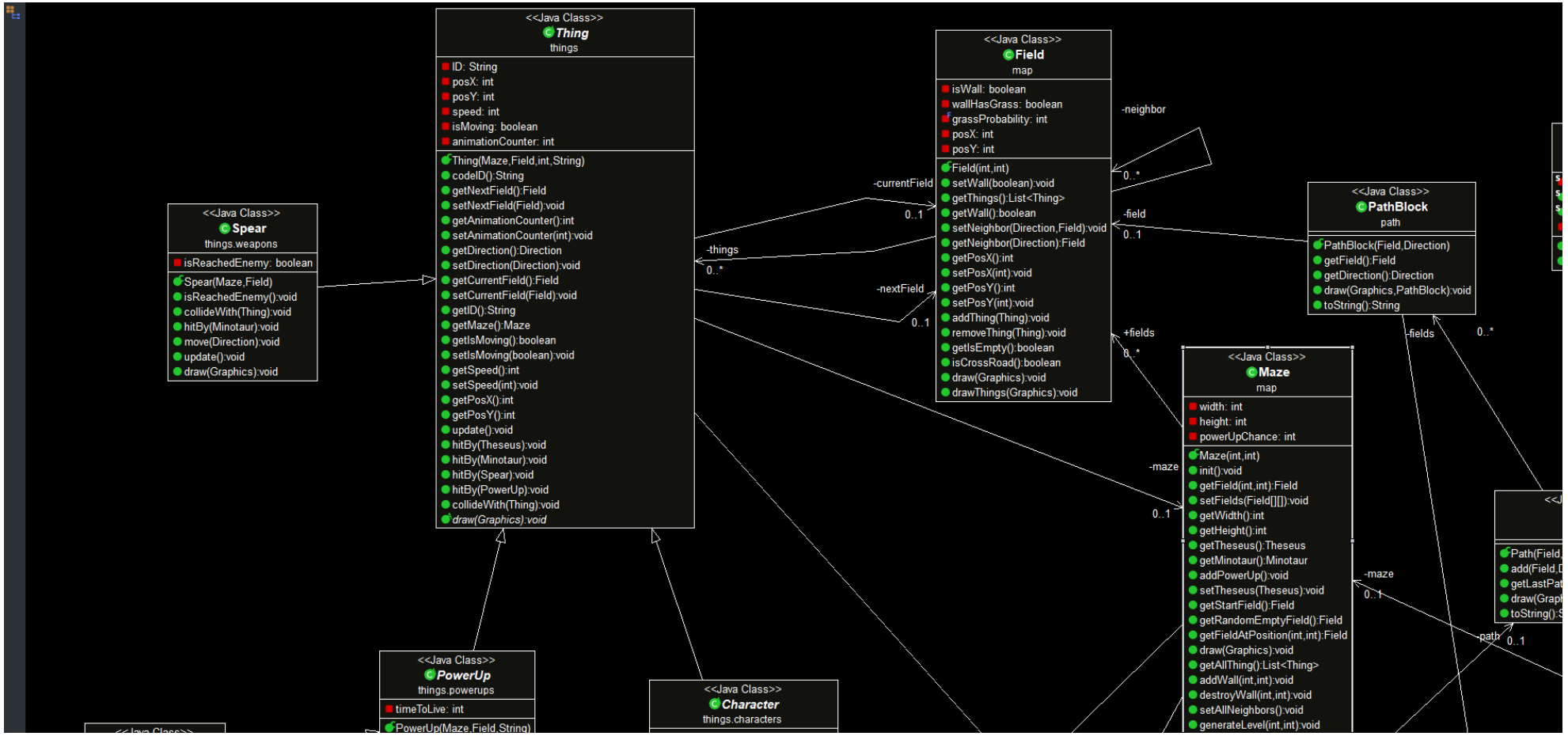
Osztálydiagramm



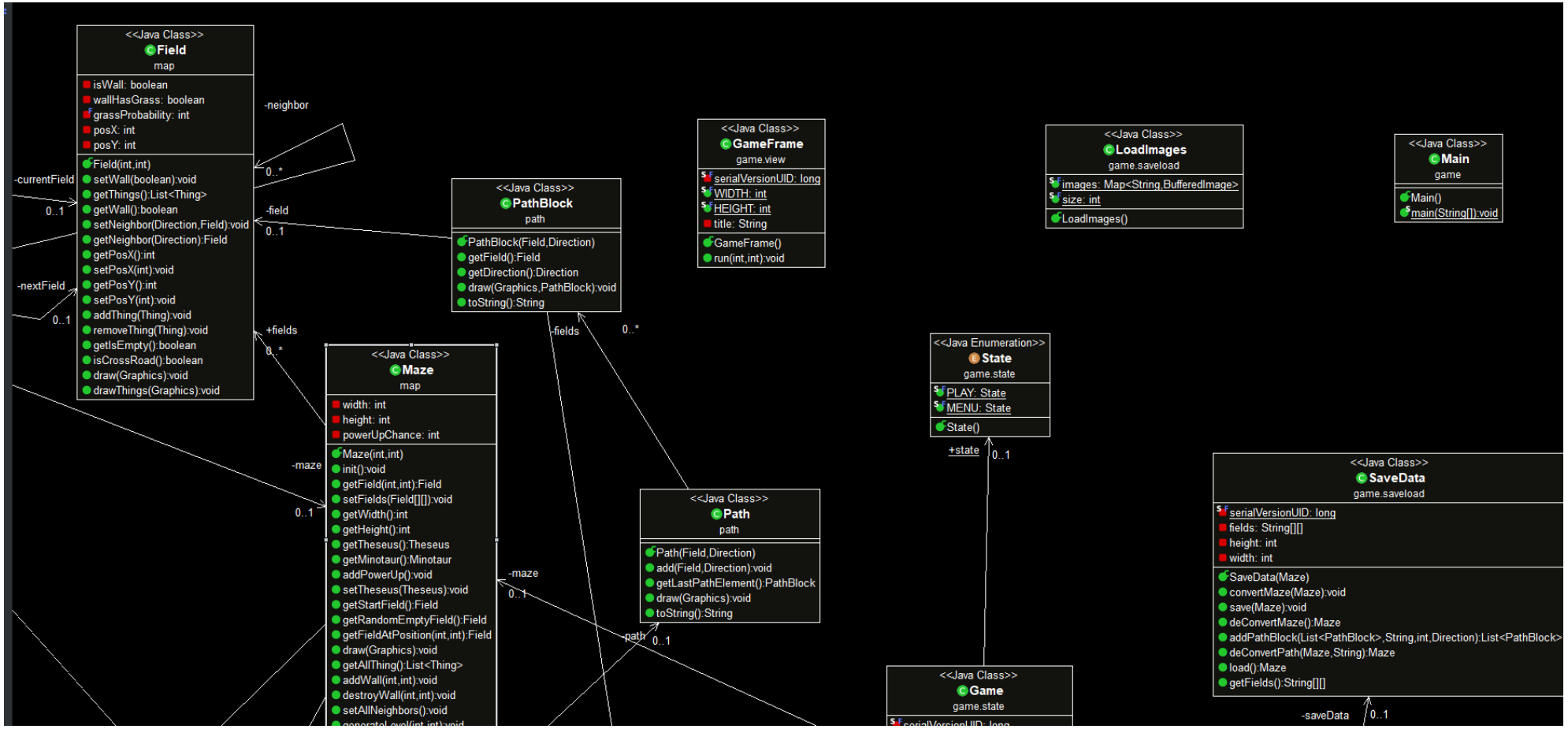
Megjegyzés: Az ábra az Eclipse osztélygenerátorával készült. Nem szerpelnek rajta a függőségek az olvashatóság kedvéért, ez indokolja az elszigetelődött osztályokat.

A láthatóság kedvéért 4 részre bontottam az osztálydiagrammot.

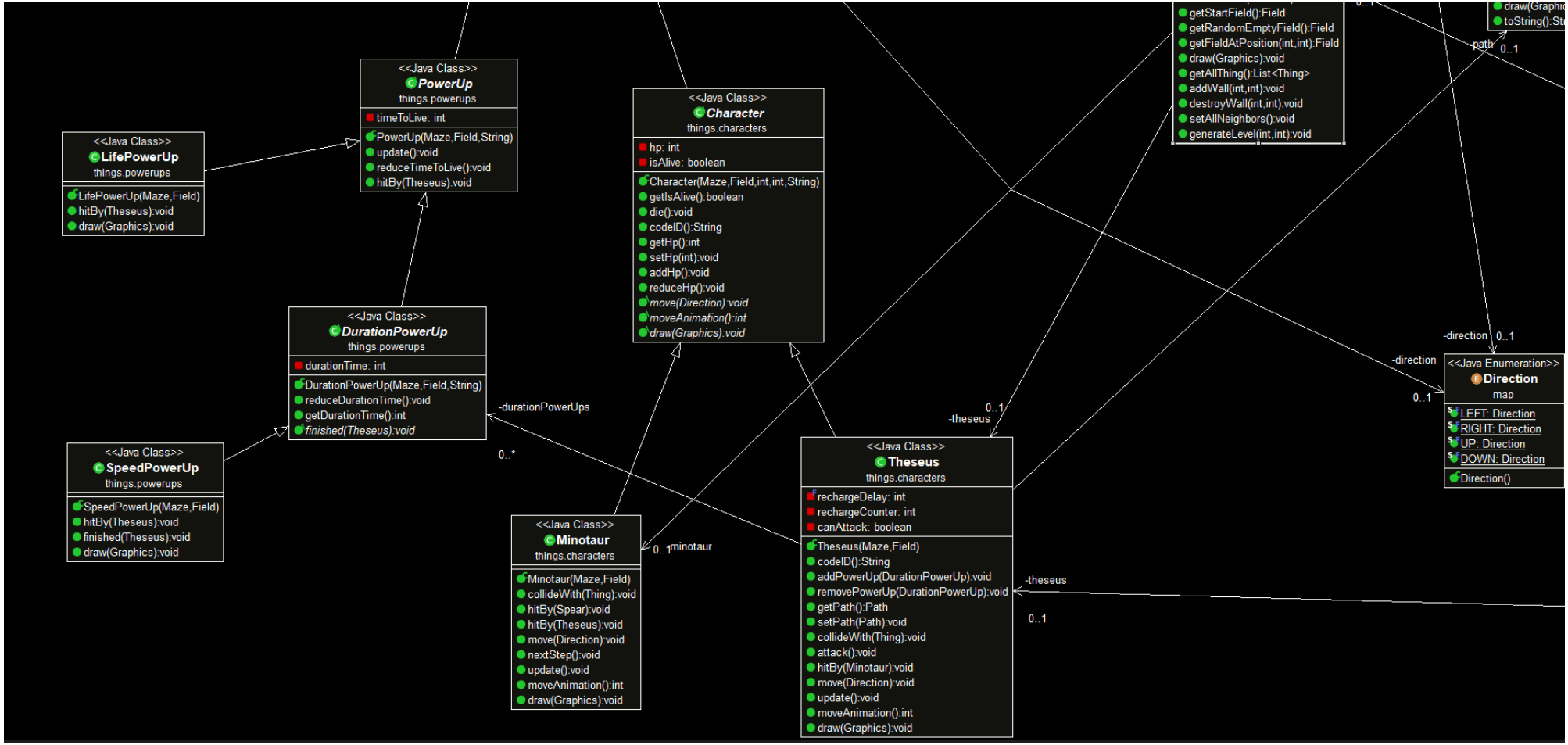
Bal felső



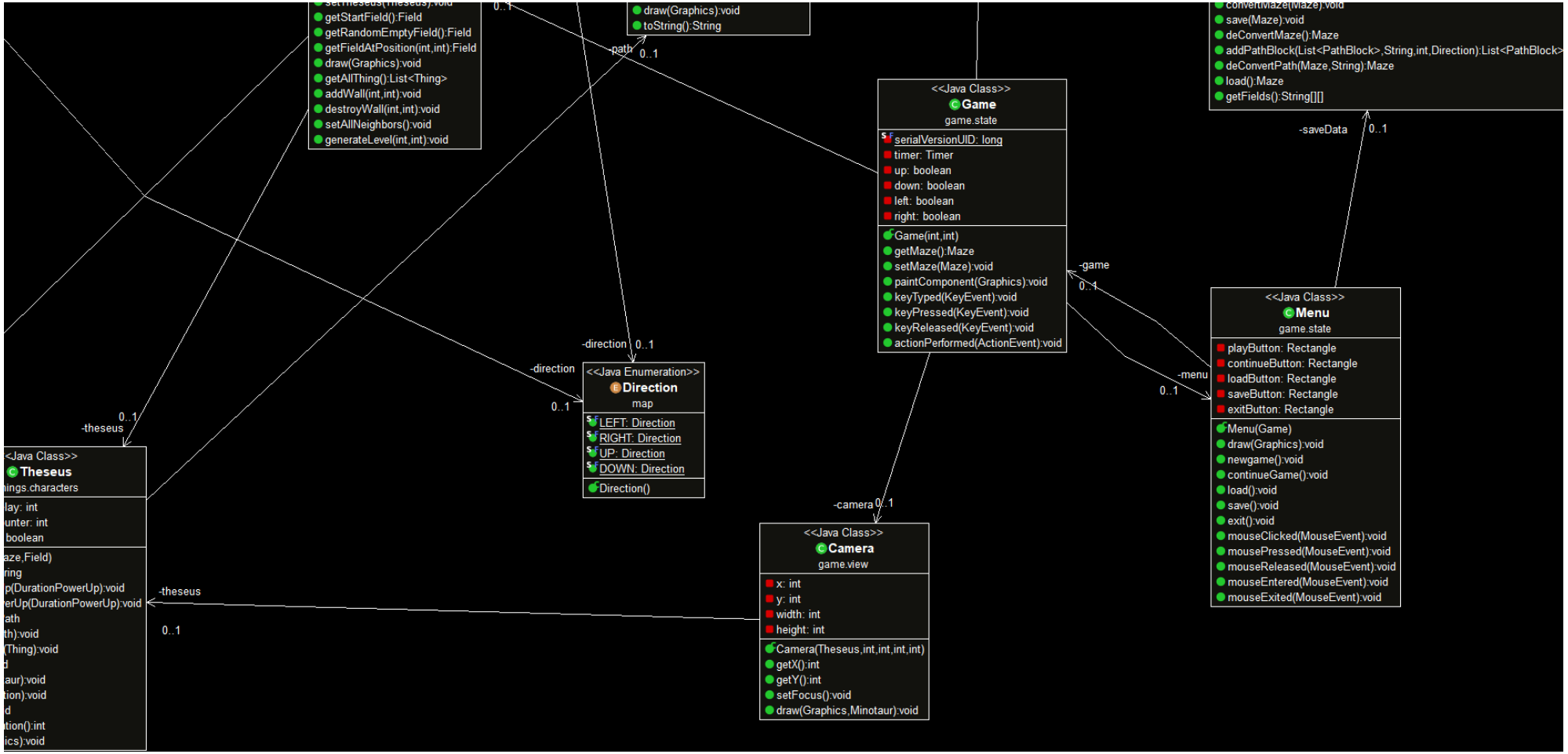
Jobb felső


<<Java Class>>
Field
map
isWall: boolean
wallHasGrass: boolean
grassProbability: int
posX: int
posY: int
Field(int,int)
setWall(boolean):void
getThings():List<Thing>
getWall():boolean
setNeighbor(Direction,Field):void
getNeighbor(Direction):Field
getPosX():int
setPosX(int):void
getPosY():int
setPosY(int):void
addThing(Thing):void
removeThing(Thing):void
getIsEmpty():boolean
isCrossRoad():boolean
draw(Graphics):void
drawThings(Graphics):void
-neighbor
0..*
-currentField
0..1
-nextField
0..1
-field
0..1
<<Java Class>>
PathBlock
path
PathBlock(Field,Direction)
getField():Field
getDirection():Direction
draw(Graphics,PathBlock):void
toString():String
<<Java Class>>
GameFrame
game.view
serialVersionUID: long
WIDTH: int
HEIGHT: int
title: String
GameFrame()
run(int,int):void
<<Java Class>>
LoadImages
game.saveload
images: Map<String,BufferedImage>
size: int
LoadImages()
<<Java Class>>
Main
game
Main()
main(String[]):void
+fields
0..*
-fields
0..*
<<Java Class>>
Maze
map
width: int
height: int
powerUpChance: int
Maze(int,int)
init():void
getField(int,int):Field
setFields(Field[][]):void
getWidth():int
getHeight():int
getTheseus():Theseus
getMinotaur():Minotaur
addPowerUp():void
setTheseus(Theseus):void
getStartField():Field
getRandomEmptyField():Field
getFieldAtPosition(int,int):Field
draw(Graphics):void
getAllThing():List<Thing>
addWall(int,int):void
destroyWall(int,int):void
setAllNeighbors():void
-maze
0..1
-maze
0..1
<<Java Class>>
Path
path
Path(Field,Direction)
add(Field,Direction):void
getLastPathElement():PathBlock
draw(Graphics):void
toString():String
path
0..1
<<Java Enumeration>>
State
game.state
PLAY: State
MENU: State
State()
+state
0..1
<<Java Class>>
SaveData
game.saveload
serialVersionUID: long
fields: String[][]
height: int
width: int
SaveData(Maze)
convertMaze(Maze):void
save(Maze):void
deConvertMaze():Maze
addPathBlock(List<PathBlock>,String,int,Direction):List<PathBlock>
deConvertPath(Maze,String):Maze
load():Maze
getFields():String[][]
-saveData
0..1
<<Java Class>>
Game
game.state

Bal alsó


<<Java Class>>
PowerUp
things.powerups
timeToLive: int
PowerUp(Maze,Field,String)
update():void
reduceTimeToLive():void
hitBy(Theseus):void
<<Java Class>>
LifePowerUp
things.powerups
LifePowerUp(Maze,Field)
hitBy(Theseus):void
draw(Graphics):void
<<Java Class>>
Character
things.characters
hp: int
isAlive: boolean
Character(Maze,Field,int,int,String)
getIsAlive():boolean
die():void
codeID():String
getHp():int
setHp(int):void
addHp():void
reduceHp():void
move(Direction):void
moveAnimation():int
draw(Graphics):void
getStartField():Field
getRandomEmptyField():Field
getFieldAtPosition(int,int):Field
draw(Graphics):void
getAllThing():List<Thing>
addWall(int,int):void
destroyWall(int,int):void
setAllNeighbors():void
generateLevel(int,int):void
-path
0..1
<<Java Class>>
DurationPowerUp
things.powerups
durationTime: int
DurationPowerUp(Maze,Field,String)
reduceDurationTime():void
getDurationTime():int
finished(Theseus):void
-durationPowerUps
0..*
-direction
0..1
<<Java Enumeration>>
Direction
map
LEFT: Direction
RIGHT: Direction
UP: Direction
DOWN: Direction
Direction()
<<Java Class>>
SpeedPowerUp
things.powerups
SpeedPowerUp(Maze,Field)
hitBy(Theseus):void
finished(Theseus):void
draw(Graphics):void
<<Java Class>>
Minotaur
things.characters
Minotaur(Maze,Field)
collideWith(Thing):void
hitBy(Spear):void
hitBy(Theseus):void
move(Direction):void
nextStep():void
update():void
moveAnimation():int
draw(Graphics):void
0..1
-minotaur
<<Java Class>>
Theseus
things.characters
rechargeDelay: int
rechargeCounter: int
canAttack: boolean
Theseus(Maze,Field)
codeID():String
addPowerUp(DurationPowerUp):void
removePowerUp(DurationPowerUp):void
getPath():Path
setPath(Path):void
collideWith(Thing):void
attack():void
hitBy(Minotaur):void
move(Direction):void
update():void
moveAnimation():int
draw(Graphics):void
0..1
-theseus
-theseus
0..1

Jobb alsó


getStartField():Field
getRandomEmptyField():Field
getFieldAtPosition(int,int):Field
draw(Graphics):void
getAllThing():List<Thing>
addWall(int,int):void
destroyWall(int,int):void
setAllNeighbors():void
generateLevel(int,int):void
draw(Graphics):void
toString():String
path
0..1
<<Java Class>>
Game
game.state
serialVersionUID: long
timer: Timer
up: boolean
down: boolean
left: boolean
right: boolean
Game(int,int)
getMaze():Maze
setMaze(Maze):void
paintComponent(Graphics):void
keyTyped(KeyEvent):void
keyPressed(KeyEvent):void
keyReleased(KeyEvent):void
actionPerformed(ActionEvent):void
save(Maze):void
deConvertMaze():Maze
addPathBlock(List<PathBlock>,String,int,Direction):List<PathBlock>
deConvertPath(Maze,String):Maze
load():Maze
getFields():String[][]
-saveData
0..1
-game
0..1
-direction
0..1
<<Java Enumeration>>
Direction
map
LEFT: Direction
RIGHT: Direction
UP: Direction
DOWN: Direction
Direction()
-theseus
0..1
<<Java Class>>
Theseus
-camera
0..1
<<Java Class>>
Camera
game.view
x: int
y: int
width: int
height: int
Camera(Theseus,int,int,int,int)
getX():int
getY():int
setFocus():void
draw(Graphics,Minotaur):void
-menu
0..1
<<Java Class>>
Menu
game.state
playButton: Rectangle
continueButton: Rectangle
loadButton: Rectangle
saveButton: Rectangle
exitButton: Rectangle
Menu(Game)
draw(Graphics):void
newgame():void
continueGame():void
load():void
save():void
exit():void
mouseClicked(MouseEvent):void
mousePressed(MouseEvent):void
mouseReleased(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void

Adatszerkezetek

A program során főként egyszerű típusokat, beépített osztályokat, illetve az általam létrehozott osztályokat használtam. Készítettem két enumerációt, egyiket az egyes irányok, másikat a játék állapotainak a kezelésére. Igyekeztem kihasználni az OOP adta lehetőségeket, így több abstrack osztályt is létrehoztam, melyeket leszármazottjait egységesen tudtam kezelni. Ezen osztályok közül kiemelendő a „Things", mivel a legtöbb osztály belőle öröklődik. Így például a mezők egységesen tudtak tárolni a rajta lévő dolgokat (heterogén kollekcióként), de emellett még számos előnnyel járt. Fix méretű helyeken tömböket (pl.: labirintus mátrixa), míg előre nem belátható számú dolgok tárolására listákat használtam, tipikusan az ArrayList-et. Amikor két dolgot szerettem volna egymáshoz rendelni, HashMap-et alkalmaztam. Ilyen volt például a mezők egyes irányokban lévő szomszédjainak tárolása is. Emellett a képek betöltésénél is egy nevet rendeltem (String) a képekhez, aminek köszönhetően később könnyebben tudtam rájuk hivatkozni. Segédosztályokat is alkalmaztam, amiket persze máshogy is meg lehetett volna valósítani. Ilyen volt például a LoadImages (képek betöltésének segítése), SaveData (Serializálás megtámogatása) vagy a PathBlock (egy „ösvényelem" elszeparált kezelése).

Tesztosztályok

A tesztelés során törekedtem a fontosabb osztályok, valamint azok kritikusabb metódusainak a tesztelésére. Ehhez a JUnitTest paraméteres tesztjeit használtam. Az alábbi 4 osztály, összesen 13 metódusát teszteltem.

- MazeTest
  - A Labirintus főbb metódusait teszteli.
- MinotaurTest
  - A Minotaurusz főbb metódusait teszteli.
- SaveDataTest
  - A SaveData főbb metódusait teszteli.
- TheseusTest
  - Theseus főbb metódusait teszteli.

Fájlok szerkezete

A forrásfájlok az „src" nevű mappában találhatók meg, azon belül további „package"-ekre bontva, tartalmuk szerint hierarchikusan. A játékhoz felhasznált képek az „images" nevű könyvtárban vannak. Megtalálható továbbá Serializálás során létrejövő „game.txt" nevű fájl is, ami a legutoljára elmetett állapotot tárolja. Később ebből tölthető be és folytatható a játék.