Software-Ontwerp

Iteratie 2

Reniers V. - Devlieghere J. - Castel D. - Pante S.

KU Leuven

March 19, 2013

Inhoud

Inleiding

Thema's die aan bod komen:

- High-Level bespreking van het ontwerp.
- Onderdelen in detail bekeken.
- GRASP en design patterns.
- Uitbreidbaarheid van het ontwerp.
- Test cases.

Rolverdeling

Rolverdeling

Iteratie 2:

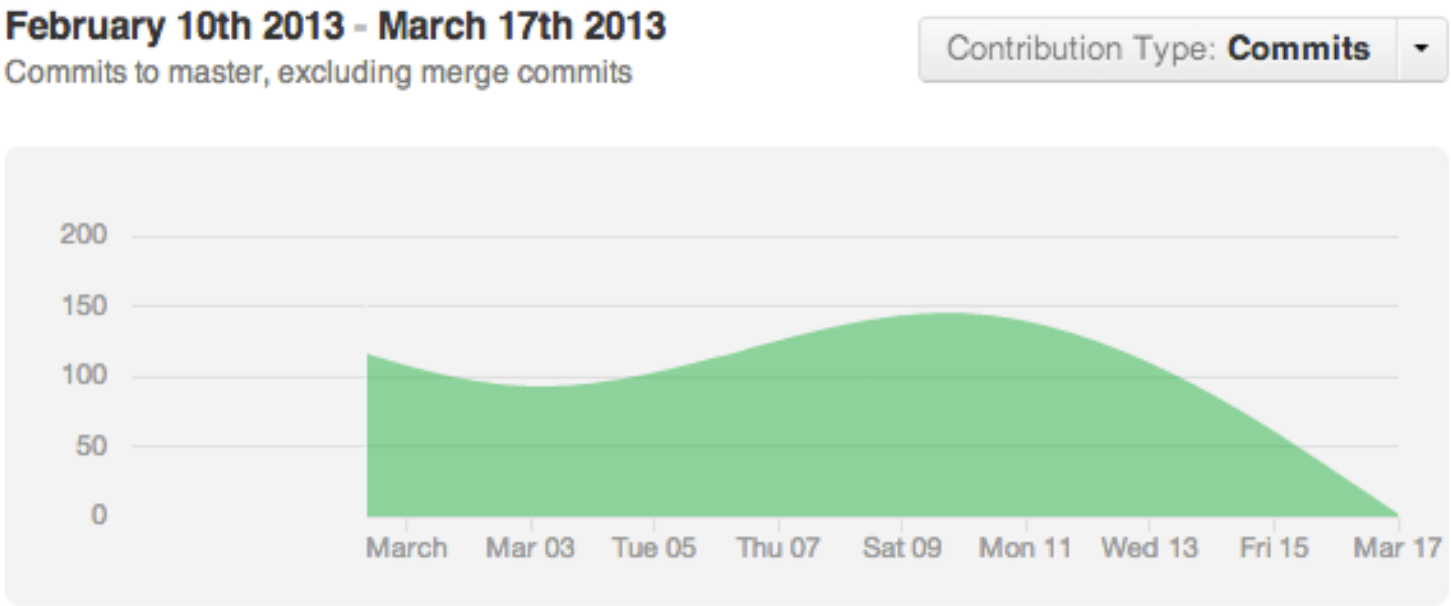
- Lead Designer: Dieter Castel
- Lead Tester: Vincent Reniers

Iteratie 3:

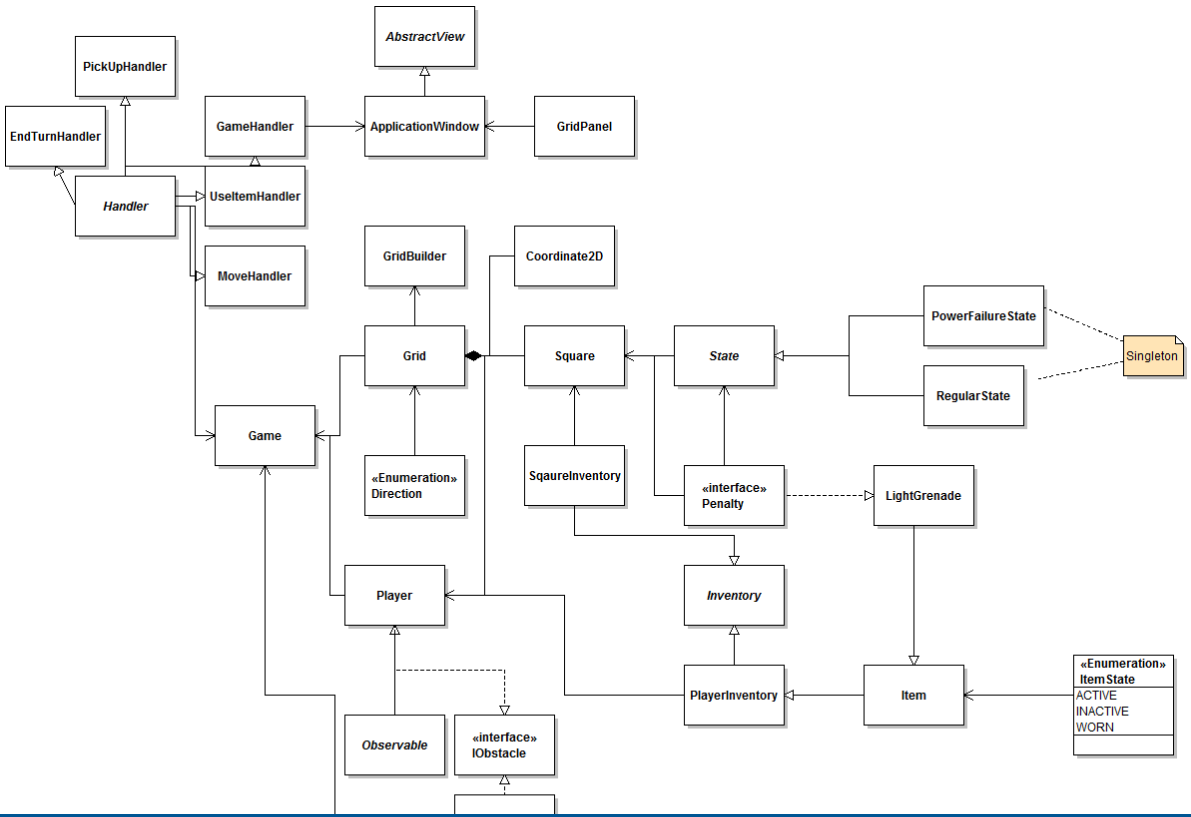
- Lead Designer: Jonas Devlieghere
- Lead Tester: Stefan Pante
- Domain Modeler: Vincent Reniers

Werkverdeling

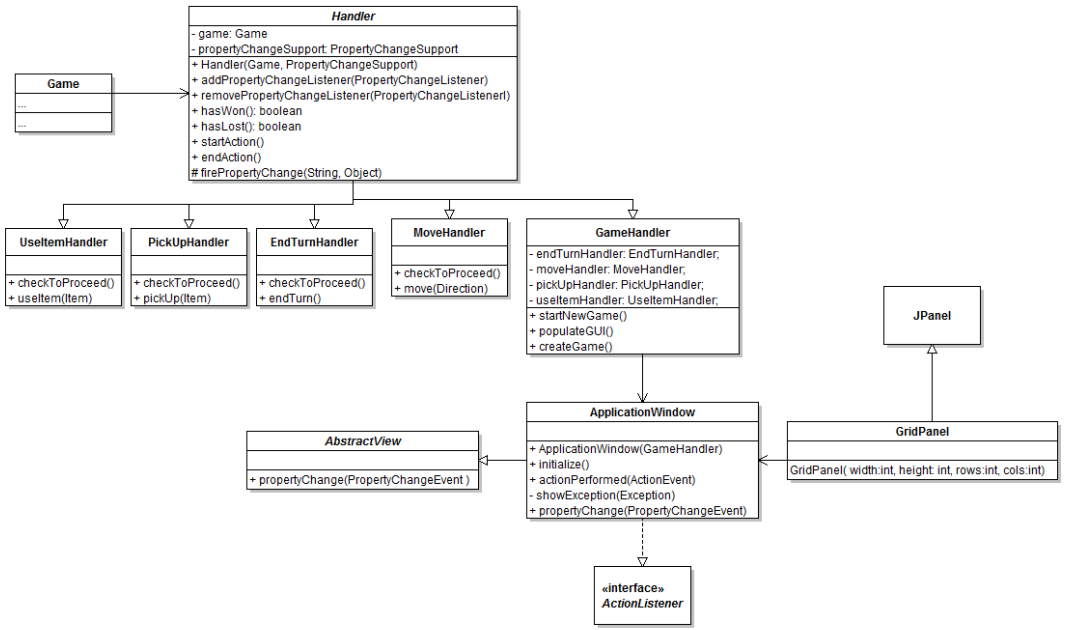
Iteratie 2: 1 maart - 15 maart



MVC


AbstractView
PickUpHandler
EndTurnHandler
GameHandler
ApplicationWindow
GridPanel
Handler
UseItemHandler
MoveHandler
GridBuilder
Coordinate2D
Grid
Square
State
PowerFailureState
Singleton
RegularState
Game
«Enumeration»
Direction
SqaureInventory
«interface»
Penalty
LightGrenade
Player
Inventory
PlayerInventory
Item
«Enumeration»
ItemState
ACTIVE
INACTIVE
WORN
Observable
«interface»
IObstacle

Handlers and view



Handlers and view

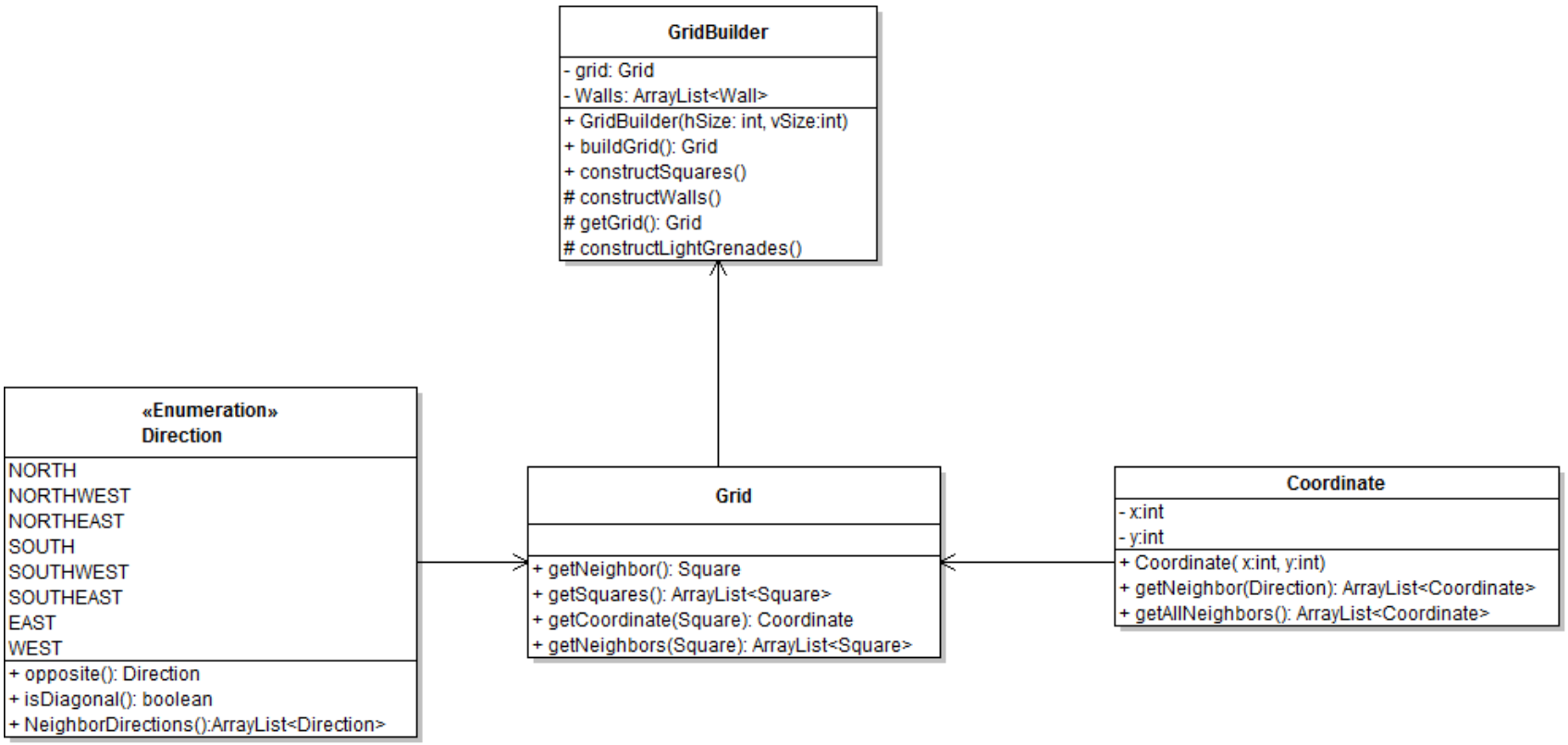
MVA (Model-View-Adapter)

- Model en View communiceren niet rechtstreeks
- Handlers zijn **mediating controllers**
- *ApplicationWindow* implementeert *PropertyChangeListener*

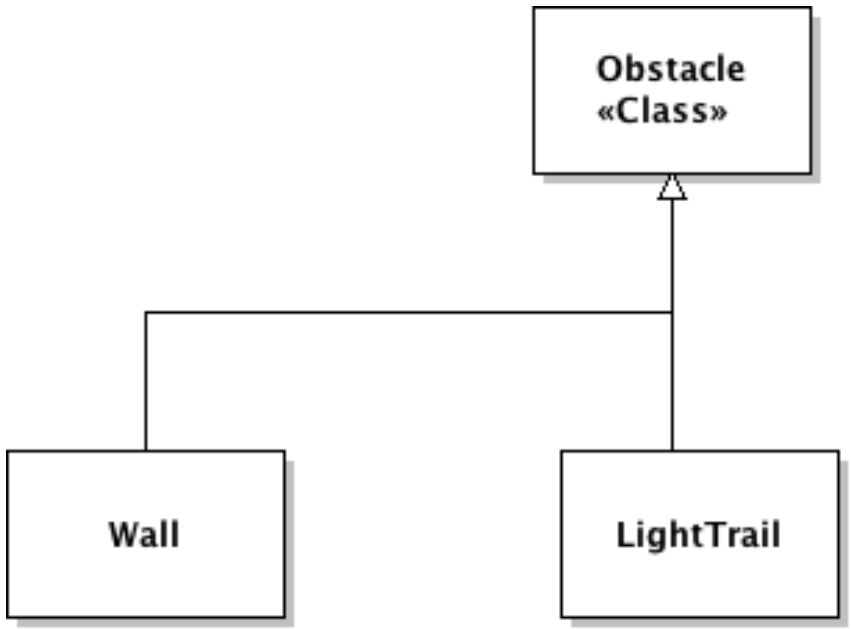
Handlers

- Handler voor elke Use-Case
- Geen GUI controller meer

Grid


GridBuilder
- grid: Grid
- Walls: ArrayList<Wall>
+ GridBuilder(hSize: int, vSize:int)
+ buildGrid(): Grid
+ constructSquares()
# constructWalls()
# getGrid(): Grid
# constructLightGrenades()
«Enumeration»
Direction
NORTH
NORTHWEST
NORTHEAST
SOUTH
SOUTHWEST
SOUTHEAST
EAST
WEST
+ opposite(): Direction
+ isDiagonal(): boolean
+ NeighborDirections():ArrayList<Direction>
Grid
+ getNeighbor(): Square
+ getSquares(): ArrayList<Square>
+ getCoordinate(Square): Coordinate
+ getNeighbors(Square): ArrayList<Square>
Coordinate
- x:int
- y:int
+ Coordinate(x:int, y:int)
+ getNeighbor(Direction): ArrayList<Coordinate>
+ getAllNeighbors(): ArrayList<Coordinate>

Obstacle interface

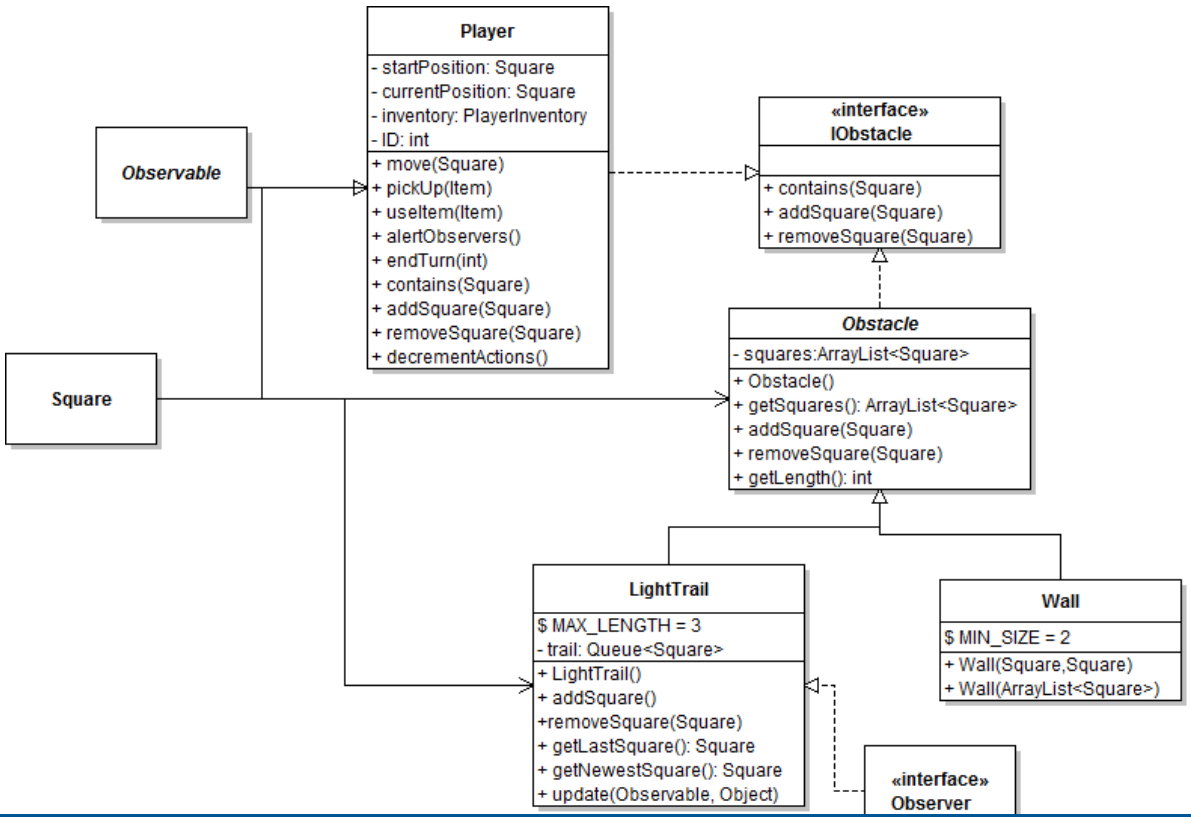


Obstacle

- Interface *IObstacle*
- Abstracte klasse *Obstacle* implementeert *IObstacle*
  - *LightTrail* implementeert *Obstacle*
  - *Wall* implementeert *Obstacle*
- *Player* implementeert *IObstacle*
- *Square* kan *Obstacle* bevatten

  LightTrail implementeert de *Observer* interface.

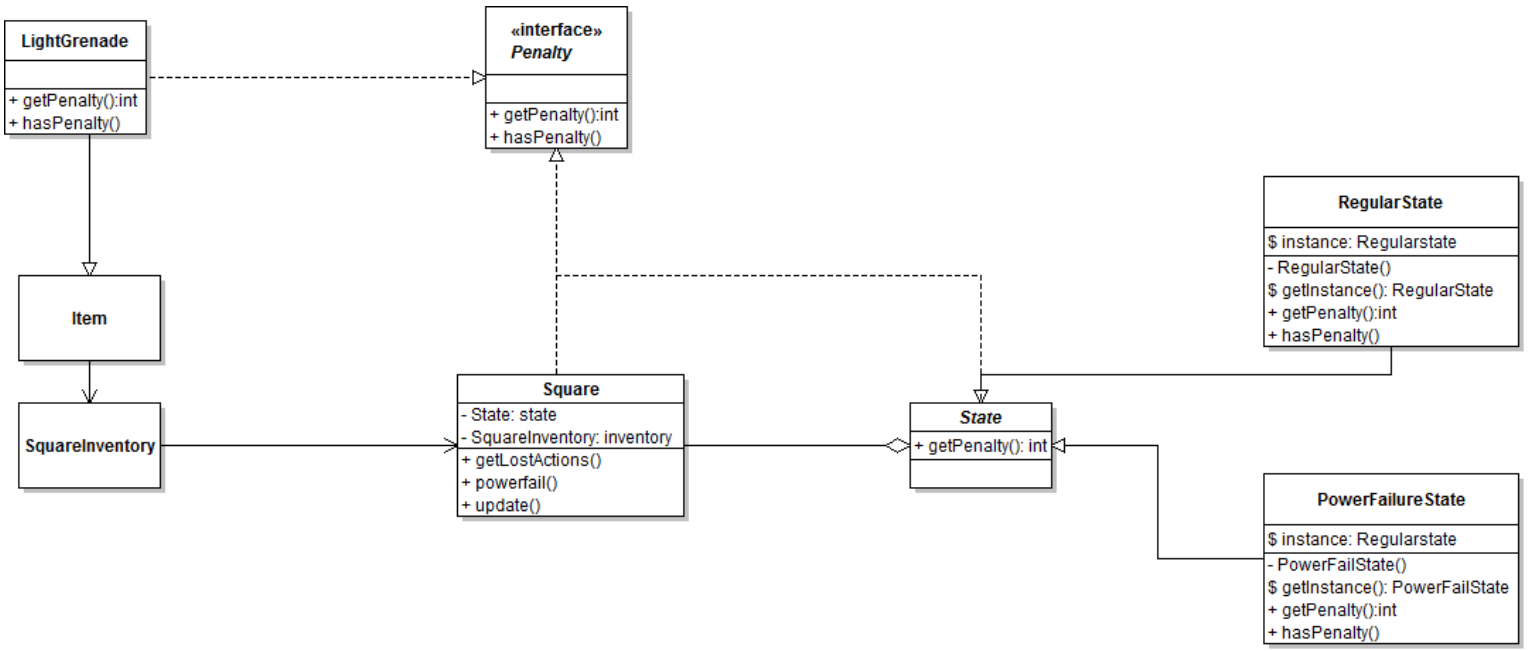
Obstacle


Player
- startPosition: Square
- currentPosition: Square
- inventory: PlayerInventory
- ID: int
+ move(Square)
+ pickUp(Item)
+ useItem(Item)
+ alertObservers()
+ endTurn(int)
+ contains(Square)
+ addSquare(Square)
+ removeSquare(Square)
+ decrementActions()
Observable
«interface»
IObstacle
+ contains(Square)
+ addSquare(Square)
+ removeSquare(Square)
Obstacle
- squares:ArrayList<Square>
+ Obstacle()
+ getSquares(): ArrayList<Square>
+ addSquare(Square)
+ removeSquare(Square)
+ getLength(): int
Square
LightTrail
$ MAX_LENGTH = 3
- trail: Queue<Square>
+ LightTrail()
+ addSquare()
+removeSquare(Square)
+ getLastSquare(): Square
+ getNewestSquare(): Square
+ update(Observable, Object)
Wall
$ MIN_SIZE = 2
+ Wall(Square,Square)
+ Wall(ArrayList<Square>)
«interface»
Observer

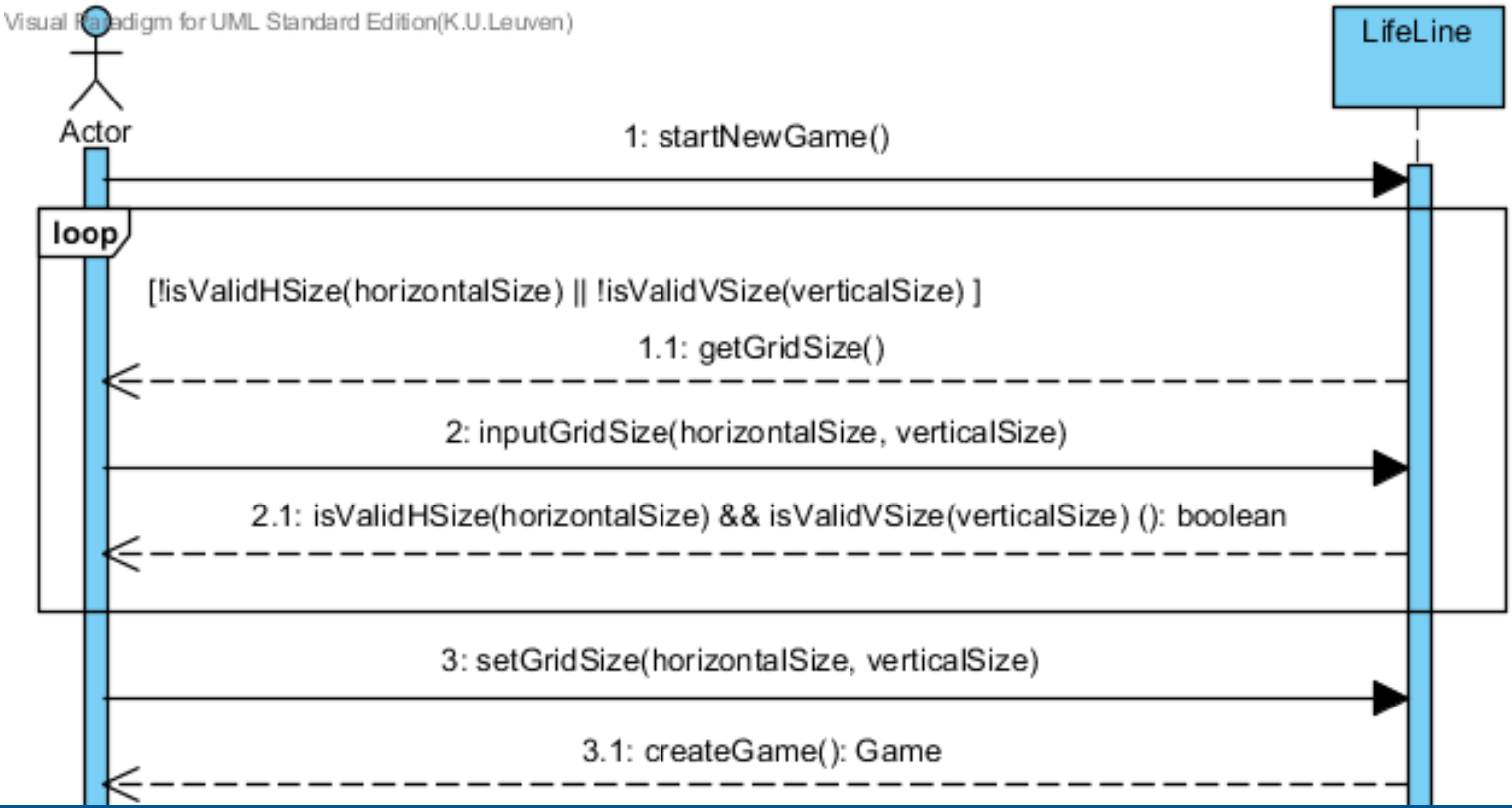
States and Penalty

- State Pattern
  - Square heeft meerdere toestanden: *RegularState*, *PowerFailureState*
  - Square zorgt voor overgang van staat
- Chain of Responsibility (Command)
  - State bepaalt eigen penalty
  - LightGrenade bepaalt eigen penalty
  - Square is eigenaar van concept penalty

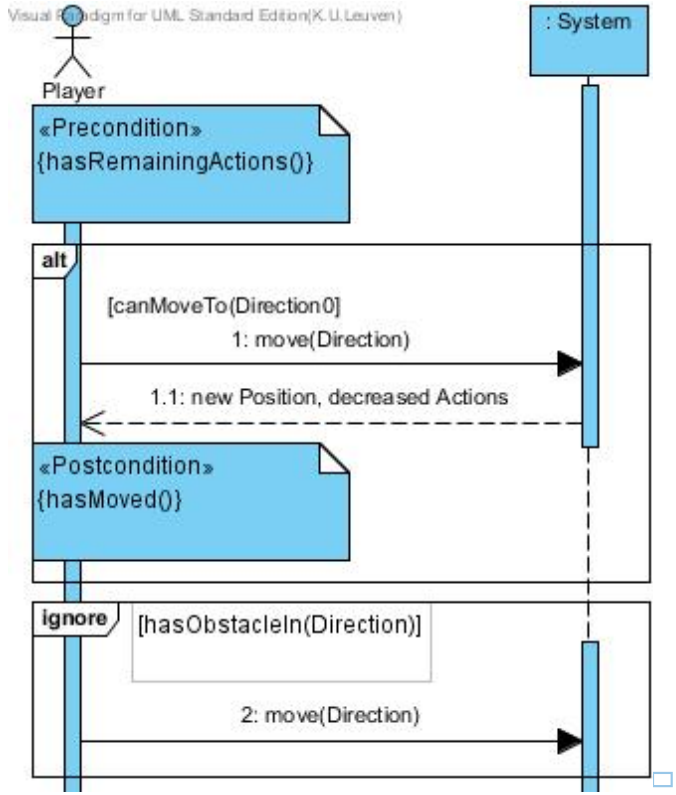
States and Penalty



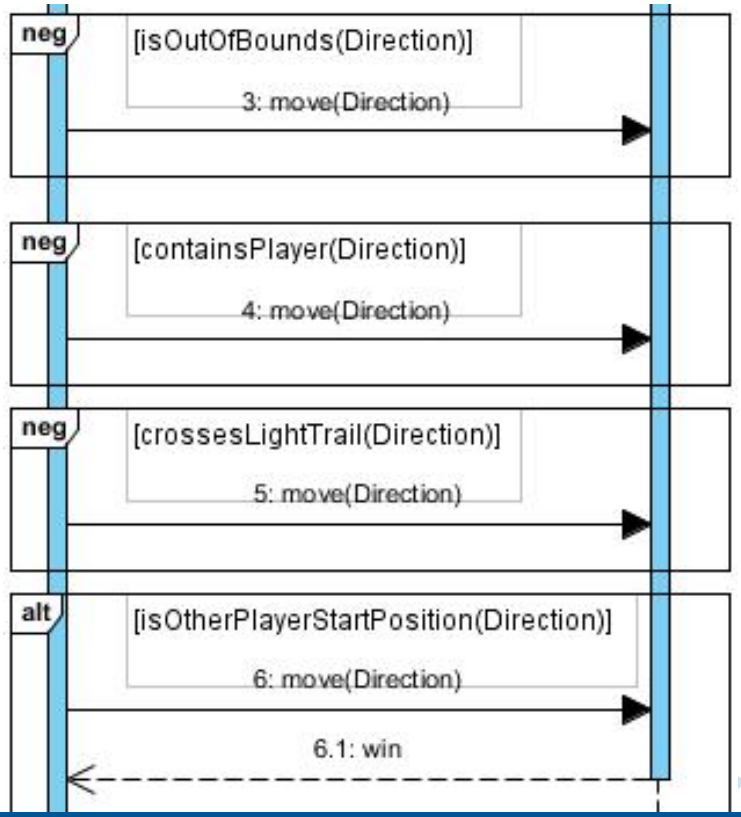
Start New Game



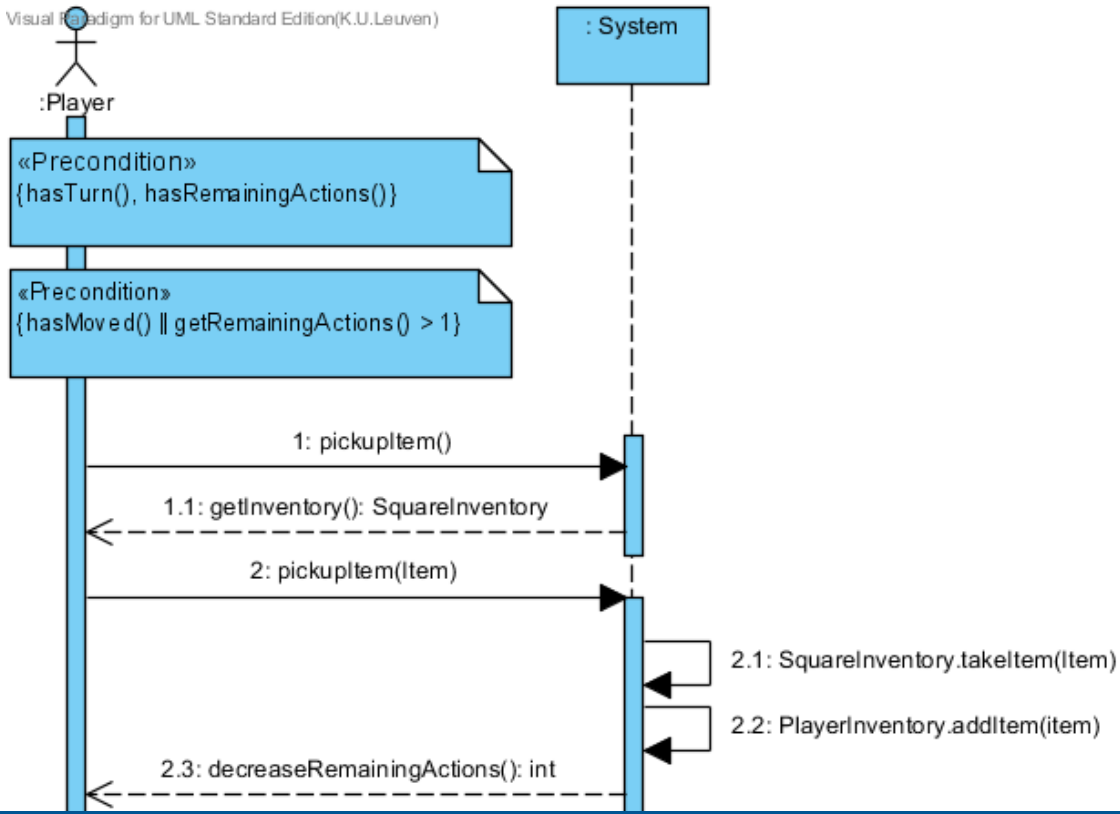
Move deel 1


Player
: System
«Precondition»
{hasRemainingActions()}
alt
[canMoveTo(Direction0]
1: move(Direction)
1.1: new Position, decreased Actions
«Postcondition»
{hasMoved()}
ignore
[hasObstacleIn(Direction)]
2: move(Direction)

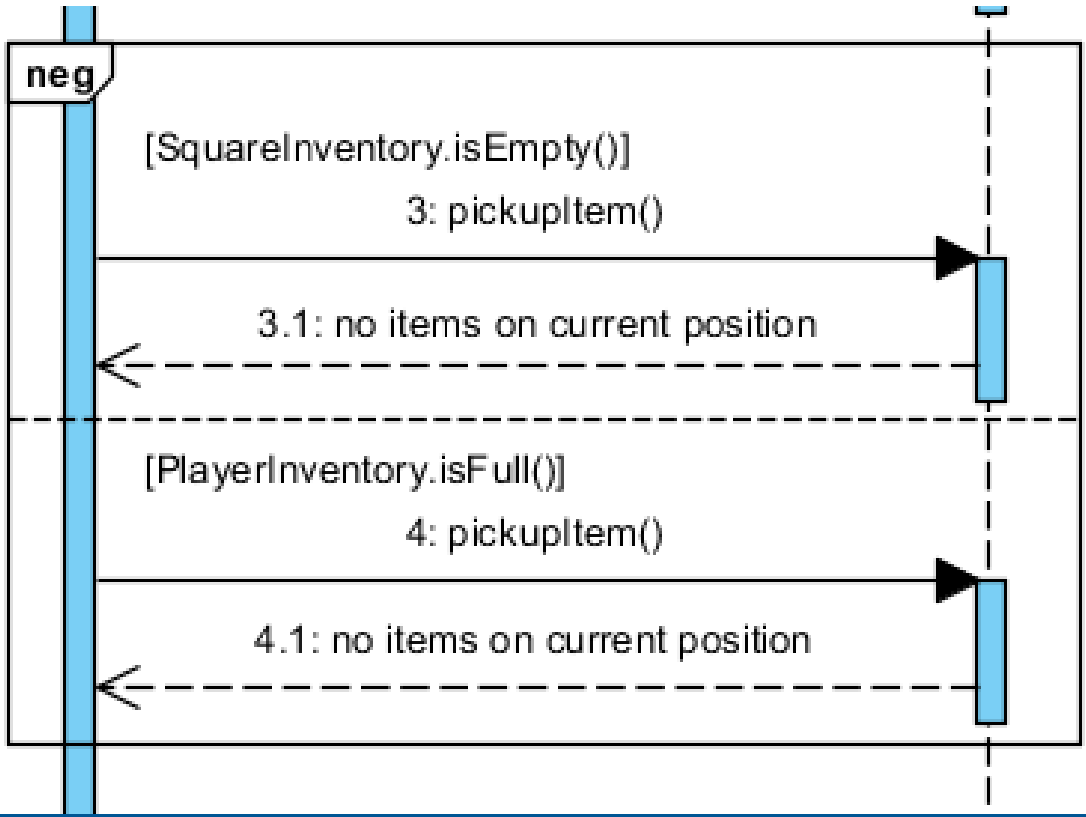
Move deel 2


neg
[isOutOfBounds(Direction)]
3: move(Direction)
neg
[containsPlayer(Direction)]
4: move(Direction)
neg
[crossesLightTrail(Direction)]
5: move(Direction)
alt
[isOtherPlayerStartPosition(Direction)]
6: move(Direction)
6.1: win

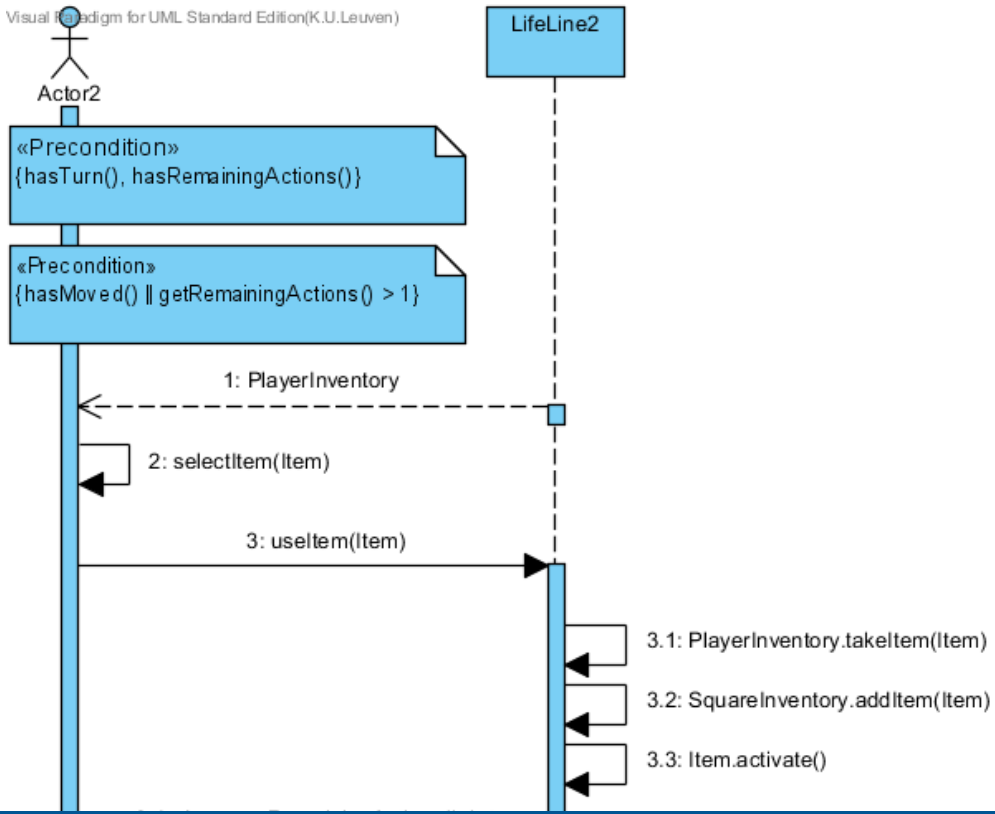
Pick Up Item deel 1



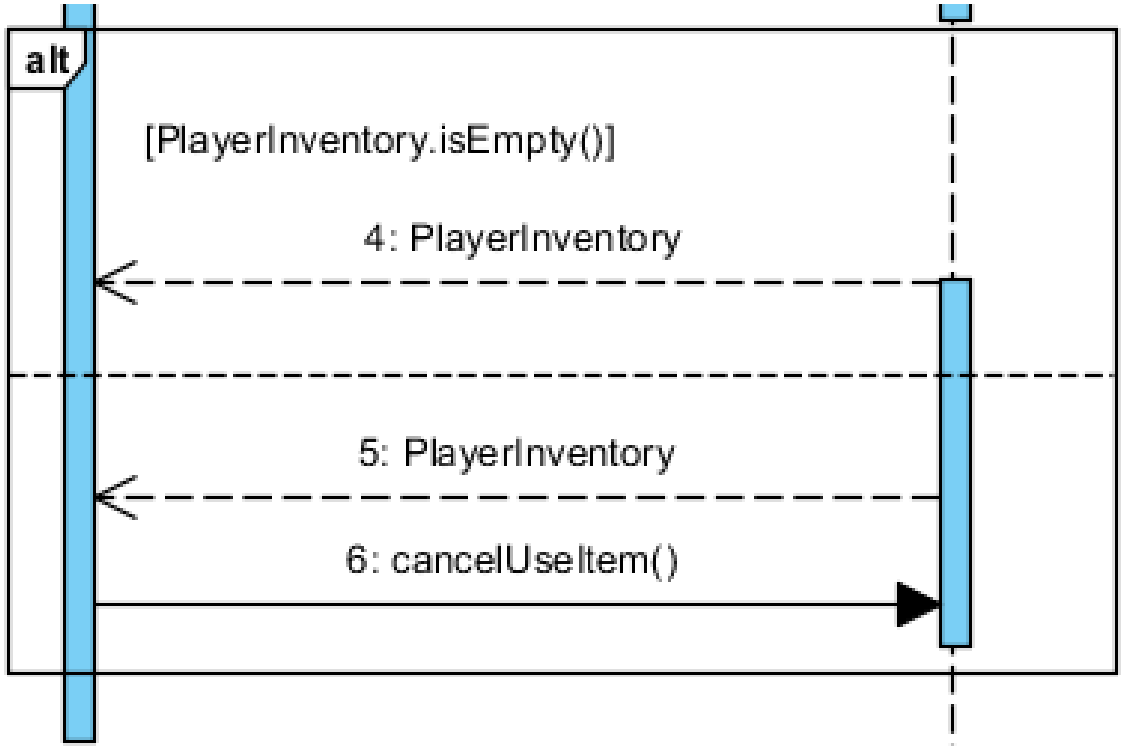
Pick Up Item deel 2



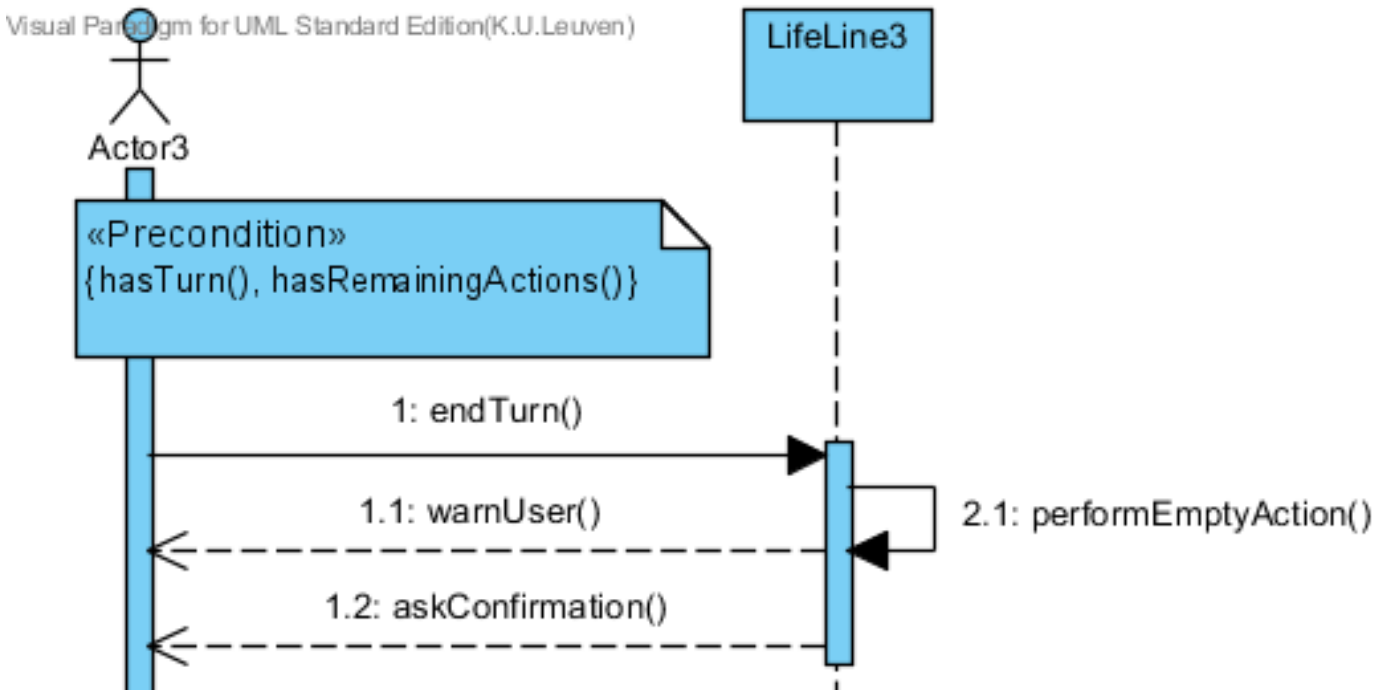
Use Item deel 1



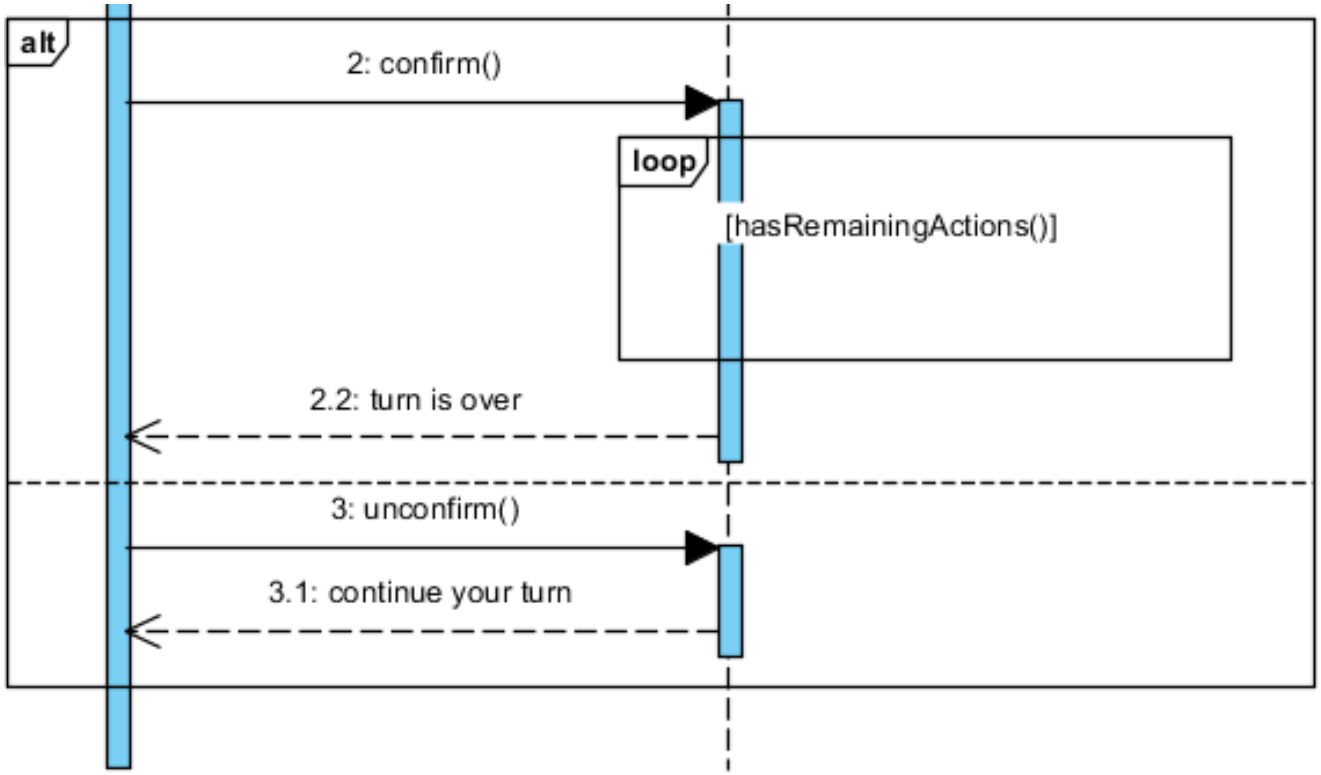
Use Item deel 2



End Turn deel 1



End Turn deel 2



Besluit

Bedankt voor uw aandacht.