

Telussa, N.Y. (17068193) Jordie Verbakel (16057155) Tim Wapenaar (17030439)

PROJECT DOOLHOF

# Inleiding

Voor het projecten moeten wij een doolhof realiseren in Netbeans waarbij een spelen kan lopen met de pijltjes toetsen, sleutels kan oppakken en deuren met de juiste sleutels openmaken. Een speler kan meerdere keren gebruikt worden maar een sleutel kan maar één deur openen.

In het begin van het project zijn we begonnen met een analyse diagram en een design diagram die bedoelt zijn voor de opdracht gever en de programmeur. Tijdens het spelen van het spel moet de speler het level kunnen resetten hierbij reset het spel, het level die je momenteel aan het spelen bent. Zodra je drie levels hebt gehaald is het spel klaar en krijgt als volgt een bericht dat je het spel uitgespeeld hebt.

Ook hebben wij tijdens het maken van het spel en toen het af was een JUnit test op alle functies gegooid die we konden doen. (Functies met Graphics G weggelaten).

Als laatst hebben wij nog documentatie geschreven met behulp van de JavaDocs, deze kan je bekijken door:

1. Het zip bestand uit te pakken
2. /mazegame/dist/javadoc/index.html
3. Index.html openen in webbrowser

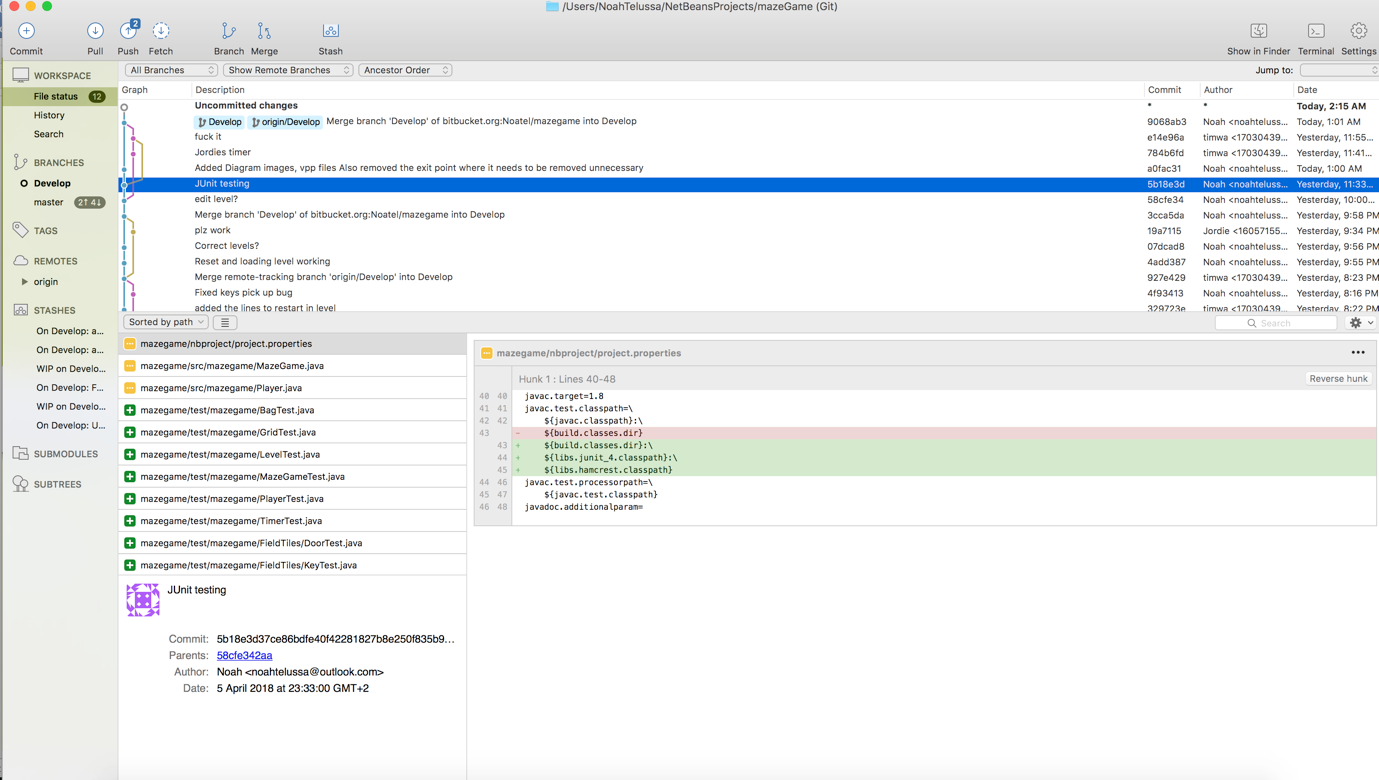
## Hoe zijn we te werk gegaan?

Omdat wij drieën al eerder hebben samengewerkt (Namelijk met het solitair spel in de eerste week) hebben we al ervaring met samenwerking. Door onze goede samenwerking hebben wij de eerste plaats binnen weten zien te krijgen, we hebben als volgt gewerkt.

### Version Control

Omdat we graag de code die we gemaakt hebben willen delen hebben we een Version control genaamd Bitbucket aangemaakt. Bitbucket kan je code opslaan en delen met andere die je toegang hebt gegeven.

We hebben de code op Bitbucket gekregen door een programma genaamd sourcetree. Met sourcetree kan je code die jij lokaal hebt veranderd op de Bitbucket neerzetten die een ander team lid weer kan pakken en mergen.



Beeld van Sourcetree

### Taakverdeling

De taakverdeling hebben we net zoals het andere project gedaan namelijk een trello board gebruikt. We hebben eerst gekeken vanuit de analyse diagram welke taken er gedaan moeten worden. Eerst hadden we op het board simpele taken neergezet zoals:” het aanmaken van level 1” maar werd snel complexer.

Ook hebben we het board in verschillende taken verdeeld namelijk :

* To do
* Doing
* On Hold
* Done

#### To Do

In To Do gaan we alle taken inzetten die nog gedaan moeten worden

#### Doing

Bij doing gaan we neerzetten welke taken we nu mee bezig zijn.

#### On Hold

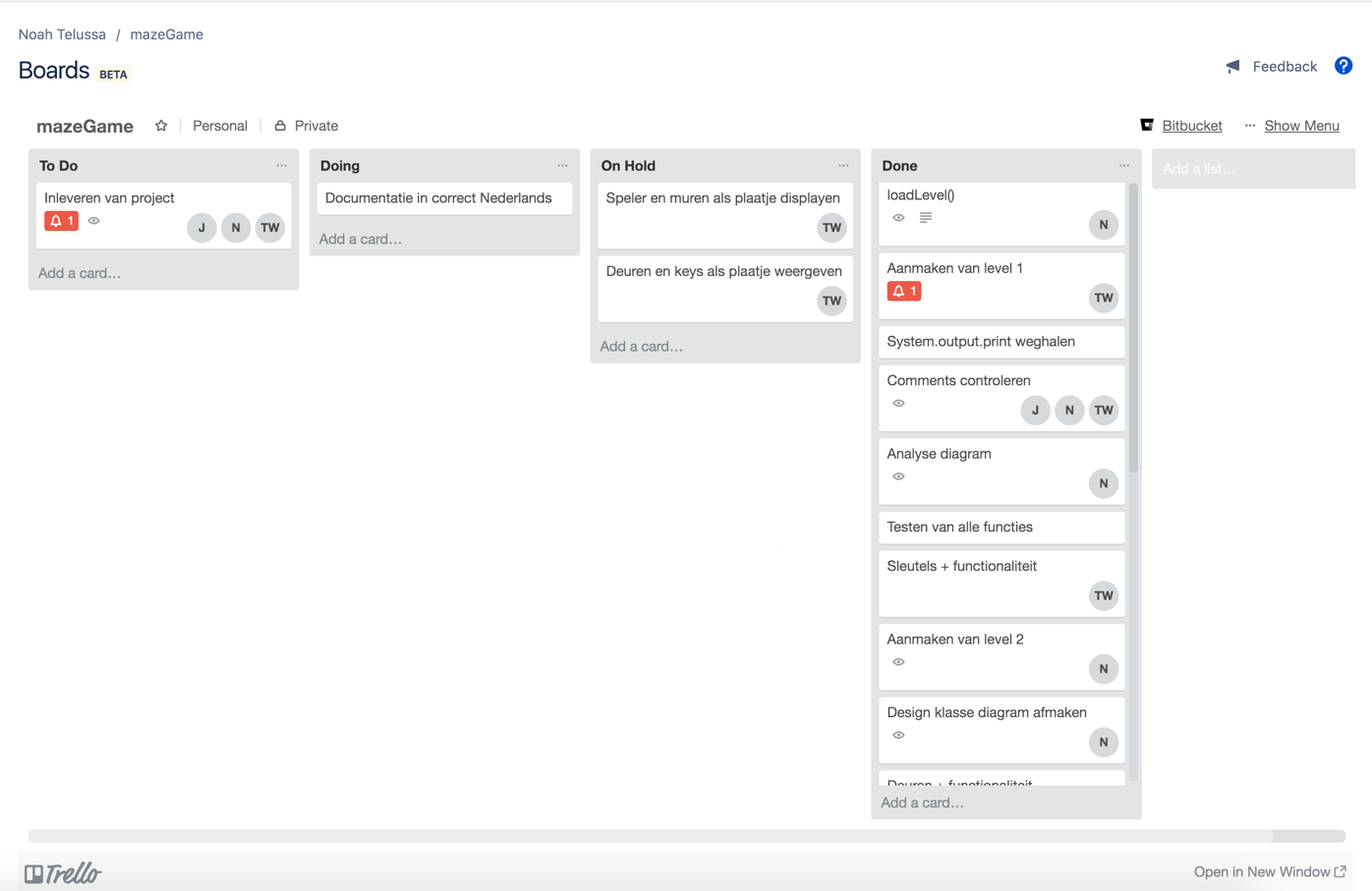
On hold zijn voor de extra dingen die we na het project kunnen doen zoals de muren en

Speler als een plaatje displayen.

#### Done

Alle taken die we al gedaan hebben

We hebben geprobeerd het zo eerlijk mogelijk te verdelen zodat iedereen evenveel werk heeft.

Trello board met taken

[Inleiding 1](#_Toc510745684)

[Hoe zijn we te werk gegaan? 2](#_Toc510745685)

[Version Control 2](#_Toc510745686)

[Taakverdeling 3](#_Toc510745687)

[Projecteisen 5](#_Toc510745688)

[Beschrijving hoe een level uit zou moeten zien 5](#_Toc510745689)

[Analyse diagram: 6](#_Toc510745690)

[Design diagram: 7](#_Toc510745691)

[JUnit4 testing 8](#_Toc510745692)

[Testen van de klassen 8](#_Toc510745693)

[Method getYPosition() 8](#_Toc510745694)

# Projecteisen

1. Moet een spelveld hebben dat bestaat uit vierkanten vlakken

2. Er is een speler

3. De speler moet kunnen bewegen / van vlak kunnen veranderen

4. Een startpunt waar de speler begint in het begin van de level

5. Er is een eindpunt waar de speler naar toe moet kunnen lopen

6. Bepaalde vlakken zijn muren en barricades

7. De speler moet niet door de muren kunnen

8. De speler kan alleen langs de barricades met een sleutel

9. Er is een sleutel

10. De speler moet de sleutel kunnen bewaren

11. Een zak is nodig om de sleutel te bewaren

12. Er zijn verschillende soorten barricades

13. Er zijn meerdere sleutels

14. Je kan maar 1 sleutel te gelijk vast houden

15. Je kan een sleutel maar 1 keer gebruiken

16. Je moet halverwege opnieuw kunnen beginnen

17. Je moet meerdere levels toe kunnen voegen

18. De barricades zijn oranje

19. De meuren zijn grijs

20. Het eindpunt is groen

21. Je moet een melding krijgen als je het eindpunt hebt bereikt

## Beschrijving hoe een level uit zou moeten zien

Er wordt een scherm met vlakken ingeladen met 1 groenvlak (het eindpunt), grijze vlakken die de weg naar het groenvlak en de sleutels wijzen, oranje vlakken die de weg naar het groenvlak blokkeren, en zelfgekozen kleur of een afbeelding van een sleutel voor het sleutel vlak.

De speler probeert het eindpunt te bereiken, maar omdat barricades in de weg staan moet hij zich eerst naar een sleutel zien te leiden die hij dan kan gebruiken om de barricades te open.

Wanneer de speler het eindpunt uiteindelijk heeft bereikt dan gaat hij automatisch door naar het volgende level, om ook daar weer naar het eindpunt zien te komen.

Dit gaat zo door totdat er geen levels meer zijn.

Wanneer een speler tijden een level niet meer het overzicht heeft of hij zit ergens vast kan het level opnieuw beginnen door op de reset knop te drukken.

# Analyse diagram:

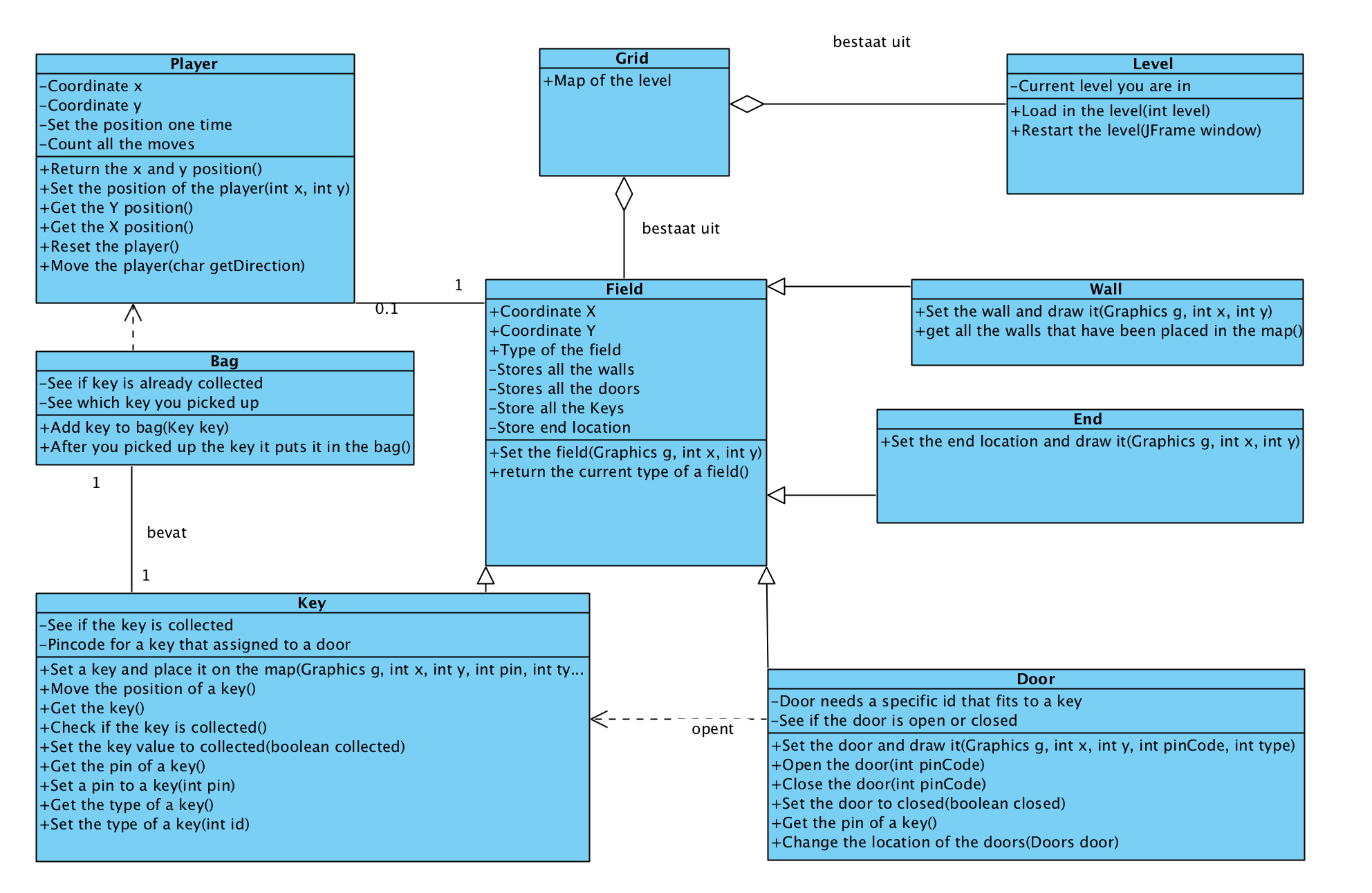
Hieronder bevindt zich de Analyse diagram die we gemaakt hebben. We hebben eerst de werkwoorden uit de opdracht gehaald als volgt zijn we aan het denken geweest hoe een speler bijvoorbeeld kan lopen, welke waarde een veld heeft en wat voor functies die moet hebben.

Nadat we de werkwoorden hebben uitgeschreven die we nodig hadden gingen we kijken wat voor methods we nodig hadden en wat ze gingen doen. We begonnen bij simpele methods zoals get the X en Y Position maar tijdens het programmeren paste we de Analyse diagram aan en design omdat we steeds complexere dingen tegen kwamen (Zoals een sleutel op de map laten spawnen).

De keuze die we gemaakt hebben is dat elk speciaal veld of het nou een Wall, Door of key is ze extende allemaal van de hoofdveld namelijk klassen Field. De klassen field heeft de basiswaarden die de andere klassen overnemen.

Wat we ook hebben is een bag. Als de Player een Key oppakt dan wordt hij automatisch in de Bag gestopt. Omdat je maar 1 key mag vasthouden hebben we zo bedacht dat als de Player nog een key oppakt dat de vorige dan wordt verwijderd.

Het laden van het Level wordt gedaan in de Level en Grid. Met de functie Loadlevel wordt level1, 2 of 3 ingeladen in de Grid Klassen. Wat de level klasse ook heeft is het restarten van een level.



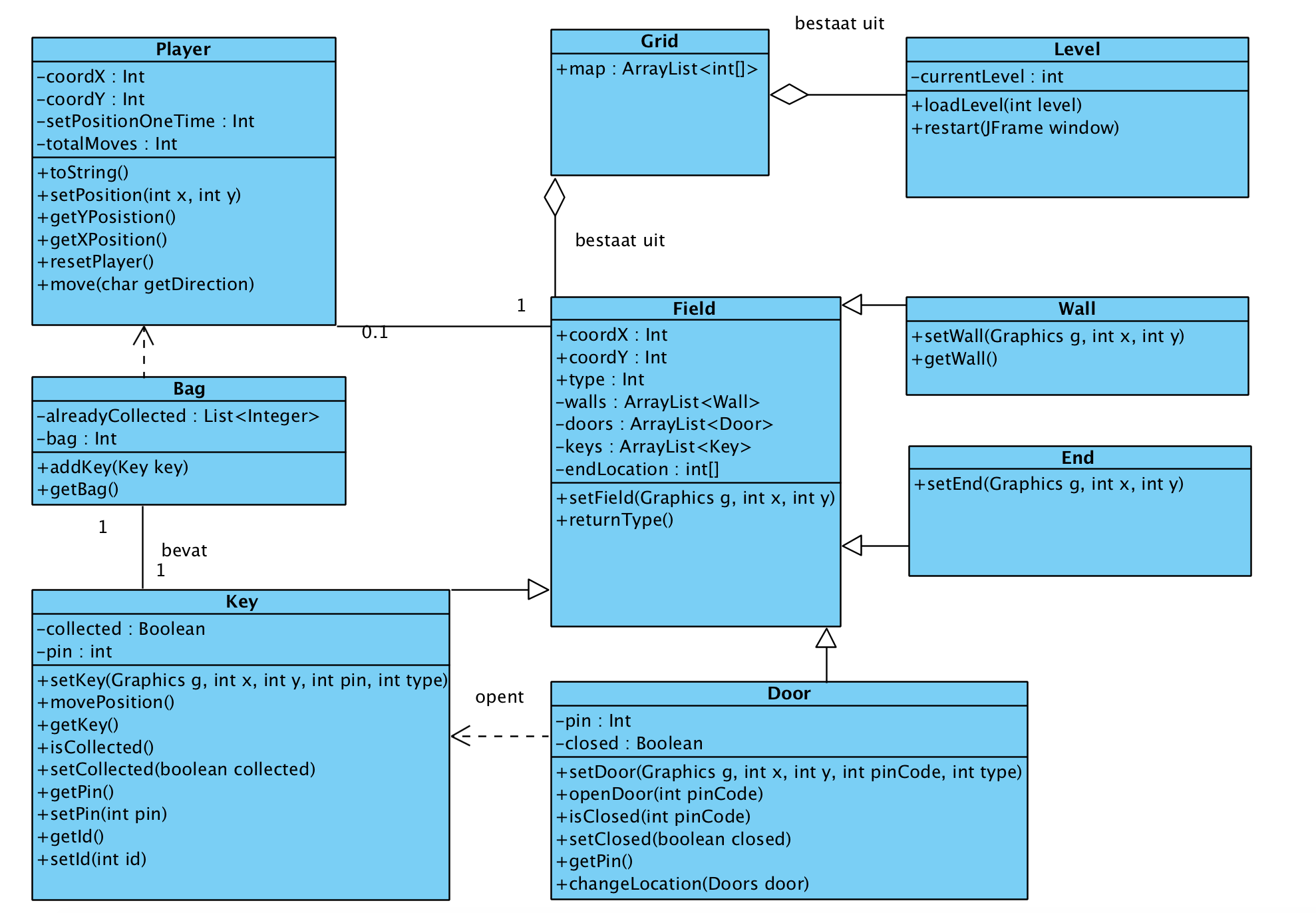
Analyse Diagram

# 

# Design diagram:

Zoals hierboven beschreven gingen we eerst aan de slag met de Analyse Diagram. De Design Diagram vonden we lastiger dan de Analyse diagram, dit kwam omdat we in het begin niet wisten welke soorten types gingen gebruiken (bijvoorbeeld Int,string,ArrayList).

Tijdens het project hebben we die types bewerkt zodat we een compleet design diagram hebben gemaakt.



Design Diagram

# JUnit4 testing

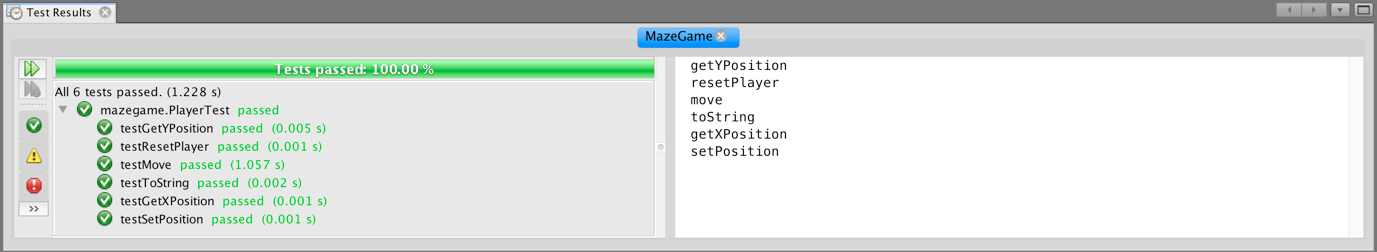
Het project dat we hebben gemaakt hebben we ook een JUnit testing gedaan bij alle klassen. Bij sommige klassen die de Graphics g gebruiken (Klassen waar je op de JFrame moet tekenen) hebben we overgeslagen. Dit komt omdat JUnit4 dit niet kon doen tenzij je een aparte package inlaadt op een complexe manier.

## Testen van de klassen

Bij het testen van de klassen heb ik er eentje uitgekozen die ik gecoverage heb, dit is namelijk de Player klassen.

Beschrijf de geplande testen en de bevindingen (invoer, verwachte uitvoer, werkelijke uitvoer) en geef aan waarom code coverage bereikt is.

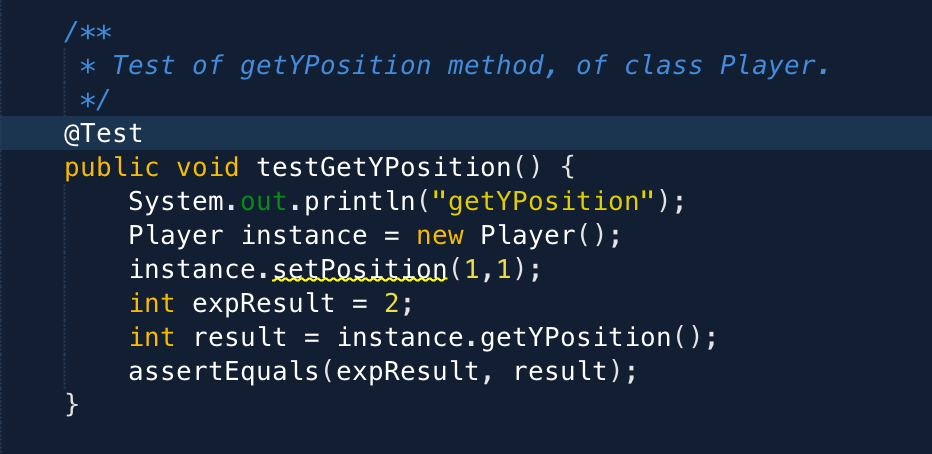
Hieronder bevindt zich de output van de JUnit4 testing die we gedaan hebben met de bijbehorende functies.

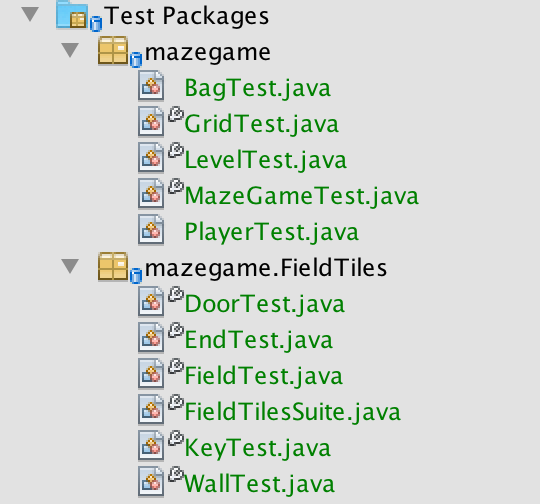


### Method getYPosition()

De getYPosition haalt hij de Y Positie op van de speler, ik maak hier een speler aan met “Player instance = new Player();”

Als je een nieuw player aanmaakt is de standaard y positie op 1.

Omdat we een setposition(1,1) eronder plakken voert hij de code coordY (1) + 1 uit. Dit zorgt ervoor dat de resultaat die we krijgen, de coördinaten van coördinaat Y op twee staat. We bevestigen dit met de functie assetEquals, hiermee kunnen we twee waardes vergelijken met elkaar. instance.getYPosition() is onze eerste waarde die we gaan vergelijken en we vergelijken het met onze expected resultaat (Wat twee is).



Zodra je het Netbeans project opent staat het in de Test Packages onder de mapjes mazegame en de FieldTiles.