Ontwikkel Document

# Beschrijving systeem eisen.

Voor het speelveld hebben wij de volgende eisen uit het project beschrijving document kunnen halen.

* Het speelveld is een vierkant vlak.
* Als de speler het eindveld bereikt krijgt hij een melding.
* Het speelveld bevat obstakels zoals muren en barricades.

Voor de barricades hebben wij de volgende eisen uit het project beschrijving document kunnen halen.

* Barricades kunnen opengemaakt worden met sleutels.
* Sleutels werken alleen op barricades met de zelfde pincode als de sleutel.
* Als de speler een sleutel probeert te gebruiken op een barricade met een andere pincode, krijgt de speler een melding.

Voor de sleutels die het karakter oppakt hebben wij de volgende eisen uit het project beschrijving document kunnen halen.

* Het karakter kan/moet sleutels ompakken.
* Het karakter kan de sleutel die hij/zij bij zich heeft niet op de grond leggen(droppen).
* Het karakter kan maar 1 sleutel bij zich hebben.
* Sleutels kunnen maar 1 keer opgepakt worden. na een sleutel is opgepakt ligt deze niet meer op het veld.
* Een sleutel kan meerdere keren gebruikt worden.

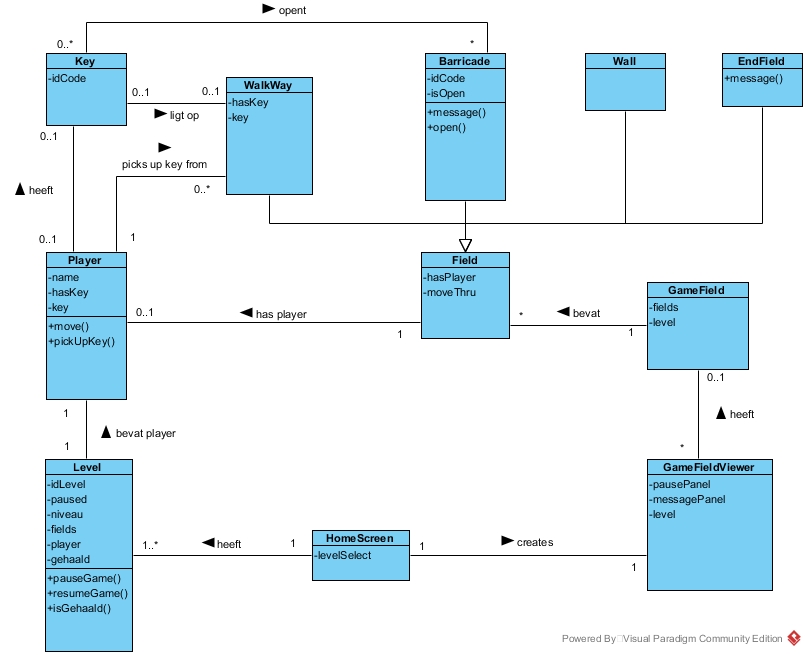
Dit is slechts een aannamen die wij hebben genomen aangezien we nergens konden vinden of de speler een keuze zou moeten krijgen om de sleutel op te pakken.

* Als de speler op een sleutel gaat staan terwijl hij al een sleutel heeft pakt hij de nieuwe sleutel op en verdwijnt de sleutel die hij had.

Voor het Karakter dat de speler bestuurd hebben wij devolgende eisen uit het project beschrijving document kunnen halen.

* De speler kan met behulp van de pijltjestoetsen op het toetsenbord zich verplaatsen
* Het moet mogelijk zijn om halverwege een spel, het hetzelfde speelveld opnieuw te starten
* Het spel moet uitgebreid kunnen worden

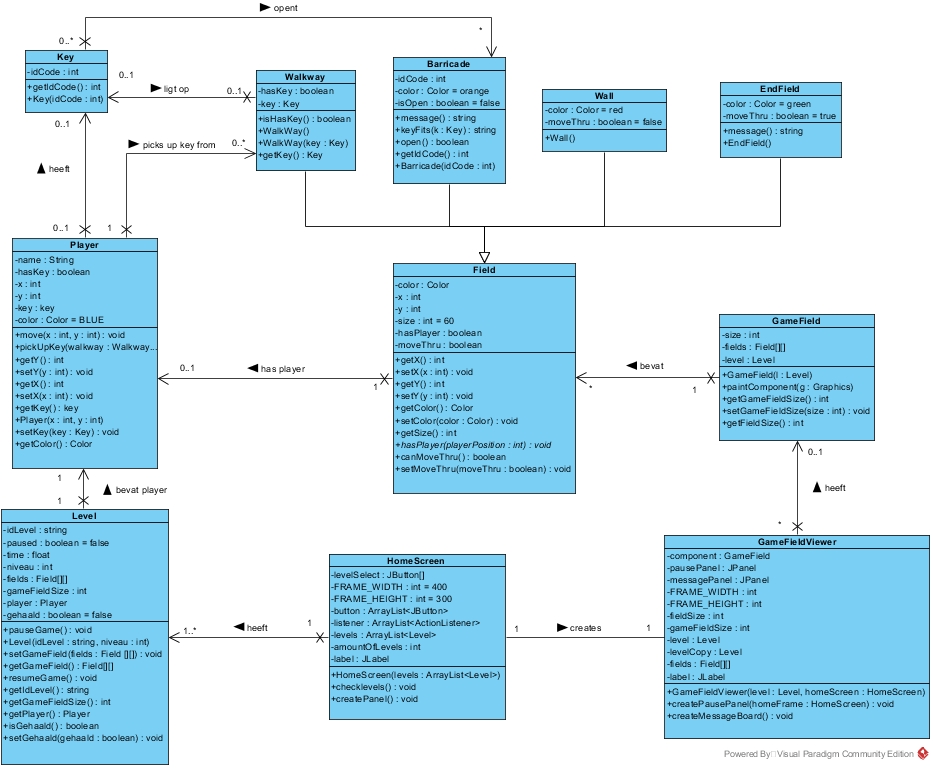
# Analysis klasse diagram



We hebben eerst gekeken wat er allemaal op het veld moest komen. Hierbij vonden wij de speler en een aantal verschillende velden. Van de velden hebben wij een super klasse gemaakt om ze te groeperen. Ook zijn er sleutels in het speel veld, omdat die niet altijd op het veld liggen hebben wij er voor gekozen om het object aan de speler of het veld mee te geven. Aangezien de speler maar 1 sleutel bij zich kon hebben vonden wij het niet nootzakelijk om alle sleutels te blijven onthouden.

Ook heeft een spel een start scherm nodig waar je een level kan selecteren om het spel te starten. Elk level heeft een speelveld met daarop een speler. Dat word door de homescreen mee gegeven aan de gamefieldviewer om het spel zichtbaar te maken. de viewer maakt vervolgens met de velden van het speelveld het zichtbare component gamefield aan. Het spel kan beginnen.

# Design Klasse diagram

****

Tijdens het programmeren kwamen we er achter dat om functies werkend te maken ze gegevens van de andere objecten nodig hadden. Wij hebben om die reden besloten om de velden en de player objecten in een level object te zetten zodat ze gemakkelijk naar elkaar konden navigeren. Vervolgens hebben we het level mee gegeven aan de gamefieldviewer waar de grafische kant zoals de technische kant van het spel.

Wij hebben besloten om de technische kant in de keylistener van gamefieldviewer te maken omdat daar tijdens het lopen de positie van de speler wordt bepaald. En in de plaats van die positie ergens anders weer op te vragen, hebben wij besloten om gelijk te controleren op welk veld de speler staat en welke er om hem heen liggen. Hierdoor weet de keylistener meteen wat de opties van de speler zijn.

# Code coverage

Code coverage houdt in het percentage van de code die getest is, en dus een kleine kans op fouten bevat. Code coverage kan je bereken door het aantal “lines of tested code” te delen door het totaal aantal regels in je code x100%.

“Lines of tested code”/”Lines of code”\*100%

In ons geval is het percentage code die we getest hebben met Unit testen:   
24 / 980 \*100% = 2.45%

In ons geval hebben we de constructor methode van Classe Level getest voor:   
24 / 24 \*100% = 100%

# Decision coverage

Bij decision coverage is het belangrijk dat je alle mogelijke uitkomsten test bij true en false. Je kan het zien als een aantal if-statements achter elkaar. Decision coverage wordt ook vaak verstaan als branch coverage omdat de uit komst van een beslissing (“true” or “false”) een branch wordt genoemt. Een decision is een if-statement of een loop control-statement.

Bijvorbeeld:

Read A

Read B //Read A,B 1A

IF A+B > 10 THEN //A+B > 10 2 B/C

Print "A+B is Large" //"A+B is Large" 3D

ENDIF // E

If A > 5 THEN // A > 5 4 F/G

Print "A Large" //"A Large" 5H

ENDIF

Om decision coverage te berekenen moet je kijken naar het minste aantal wegen die je moet nemen om alle beslissingen af te gaan.

1. 1A-2C-3D-E-4G-5H
2. 1A-2B-E-4F

Alle beslissingen zijn nu genomen dus de decision coverage is 2.