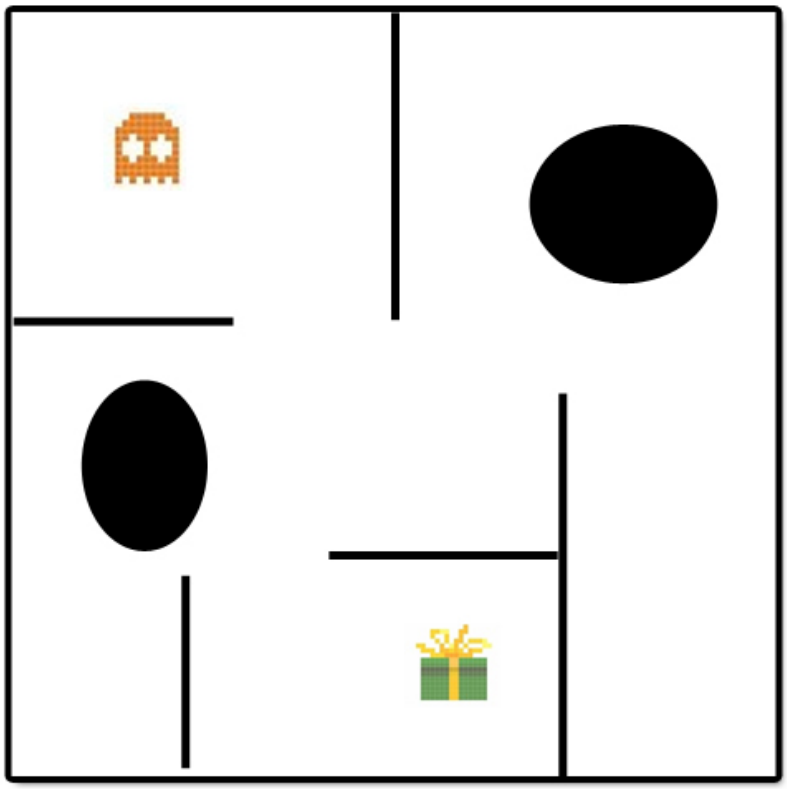
Search Robot



Software Architecture Document

Camille Zanni (zannc2)
Simon Gfeller (gfels4)

Inhalt

Contents

Abbildungsverzeichnis

Änderungsverzeichnis

Version	Datum	Beschreibung	Author
First Draft	7. Januar 2014	First Draft	gfels4/zannc2
Last Draft	9. Januar 2014	Last Draft	gfels4/zannc2

Einleitung

Das Projekt Search Robot wurde in zwei Unterprojekte Frontend und Robot aufgeteilt. So konnten zuerst das Frontend, welches den Editor enthält, implementiert werden. Die Roboterlogik konnte dann an das funktionierende Frontend anhängen und implementiert werden. Diese Aufteilung ist ebenfalls in der Struktur des Codes ersichtlich.

Hinweis: Für die bessere Lesbarkeit stehen alle Diagramme als Bild zur Verfügung.

Frontend

Use Cases in kürze

Für detaillierte Informationen nehmen Sie das Dokument *Use Case Frontend* zur Hand.

Hindernisse setzen

Ein Spieler setzt alle gewünschte Hindernisse (2D Hindernisse) auf dem Spielfeld.

Roboter setzen

Der Spieler setzt den Roboter auf das Spielfeld. Diese Position wird zugleich die Startposition des Roboters sein.

Ziel setzten

Der Spieler setzt das Ziel, das der Roboter suchen muss, auf das Spielfeld.

Roboter starten

Der Spieler startet das Spiel, d.h. der Roboter beginnt das Ziel zu suchen.

Suche abbrechen

Der Spieler unterbricht die Suche nach dem Ziel und kehrt zum Spielfeldeditor zurück

Spielfeld löschen

Der Spieler hat die Möglichkeit das Spiel jederzeit zurückzusetzten, d.h. Hindernisse, Ziel und Roboter werden vom Spielfeld entfernt.

Spielfeld importieren

Ein gespeichertes Spielfeld kann von einer Datei geladen werden.

Spielfeld exportieren

Das Spielfeld wird in einer Datei gespeichert werden.

Implementation

Field und View

Das Field dient als Container der Hindernisse. Die View verwaltet das Field und zeigt alle entsprechenden Hindernisse an, resp. sie führt das «Draw» aus.

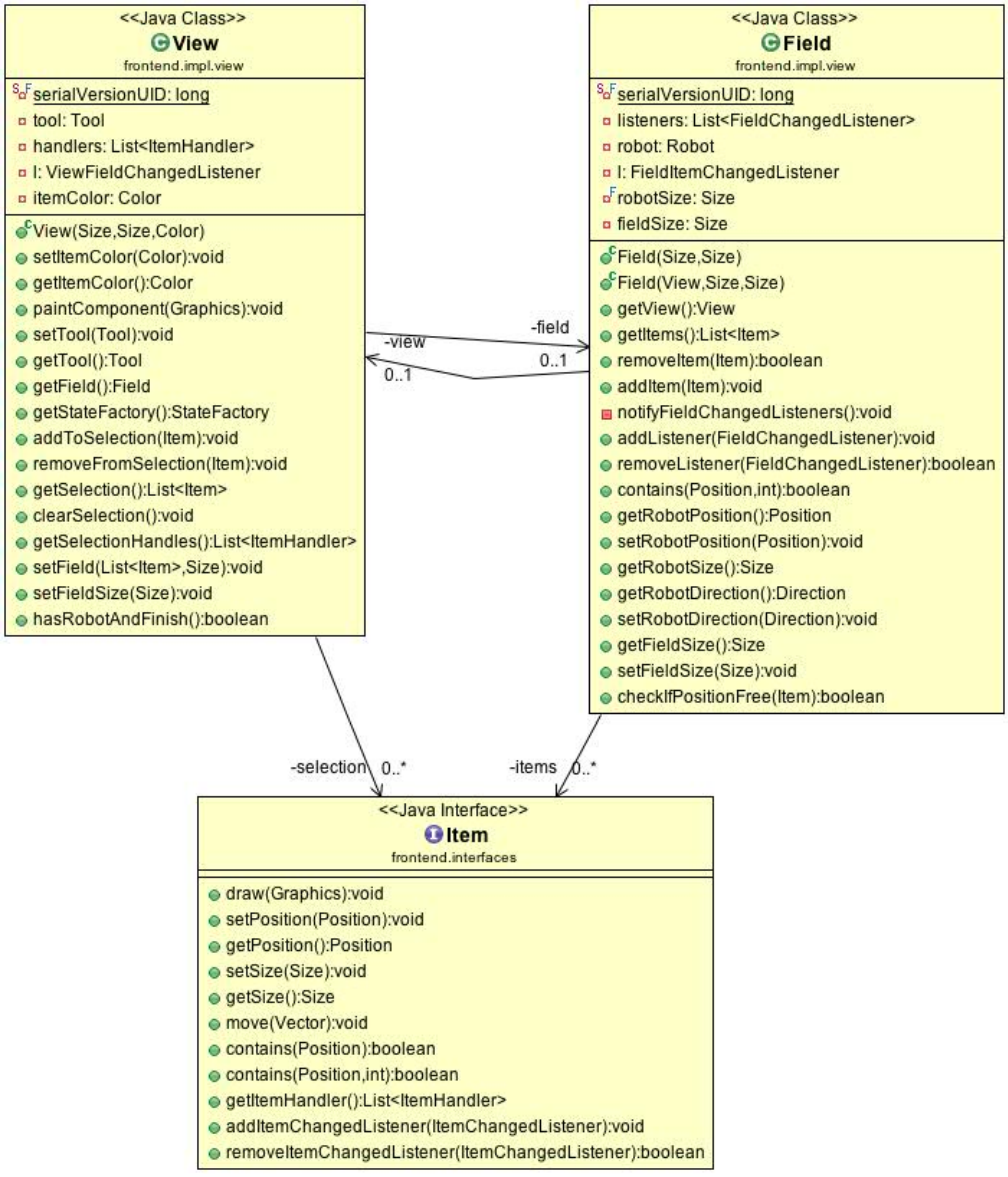


Abbildung 1: UML View und Field

Hindernisse

Zur Implementation der Hindernisse (Items) wurde das Decorator Pattern verwendet. So können ohne grosse Modifikationen weitere Hindernisse definiert werden.

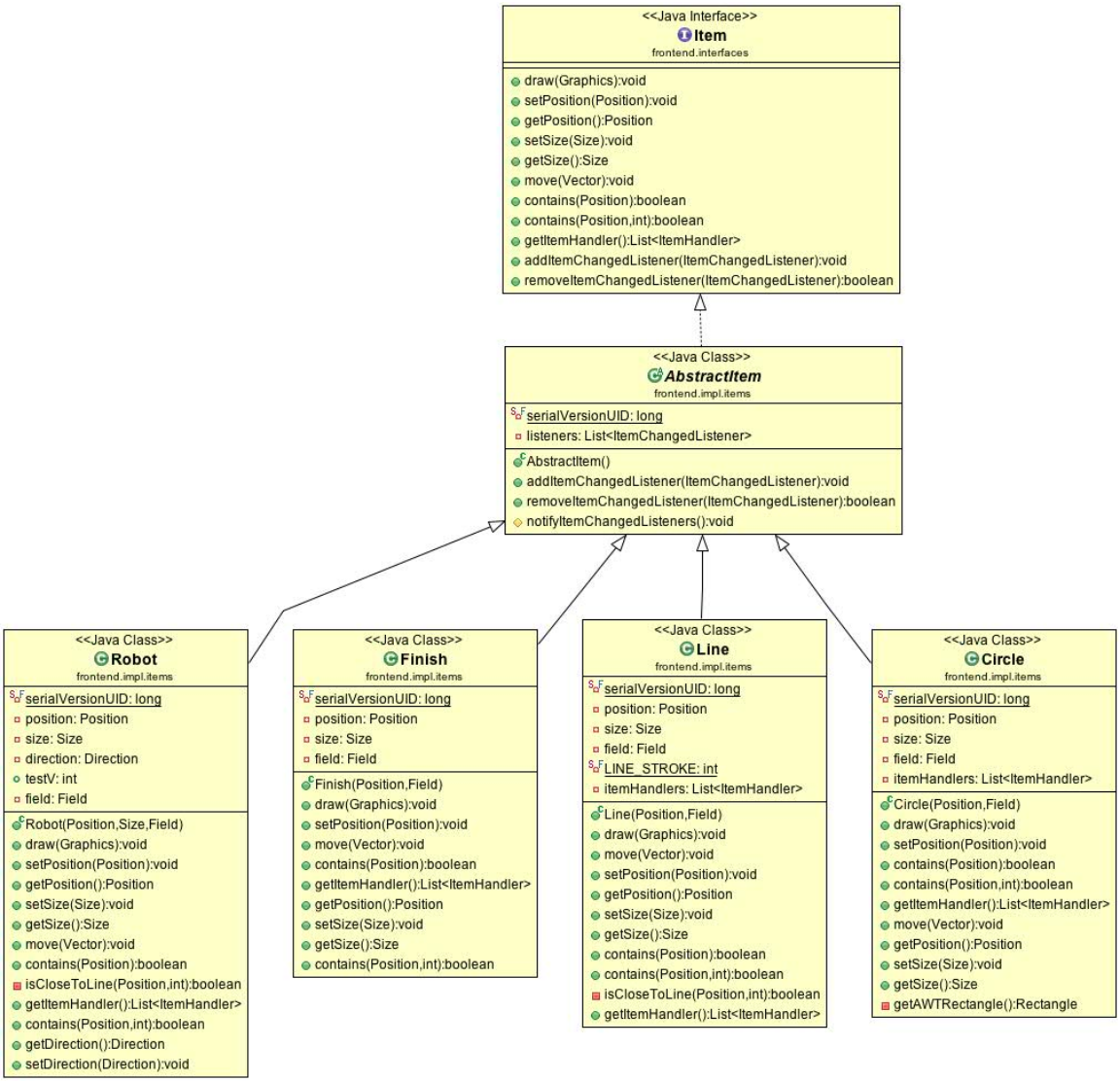


Abbildung 2: UML Hindernisse

Die Hindernisse werden jeweils durch das zugehörige Tool erstellt. Diese wurden anhand des Factory Patterns implementiert.

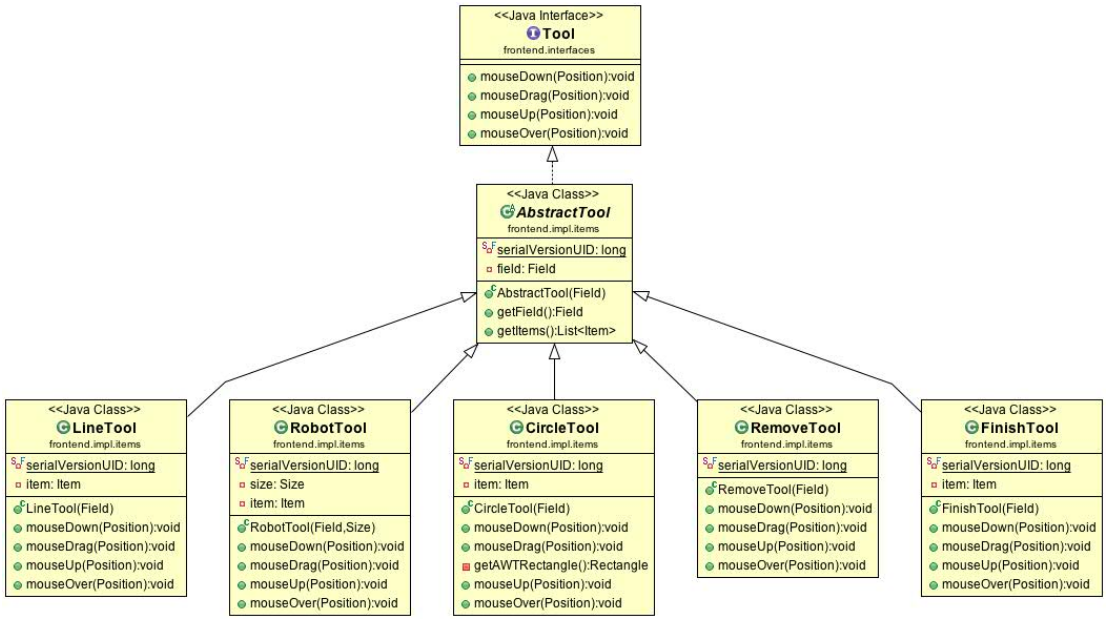


Abbildung 3: UML Tools

Zum vergrössern resp. verkleinern der Hindernisse wurden für die Hindernisse entsprechende Handler implementiert. Die Implementation wurde ebenfalls anhand des Decorator Patterns durchgeführt.

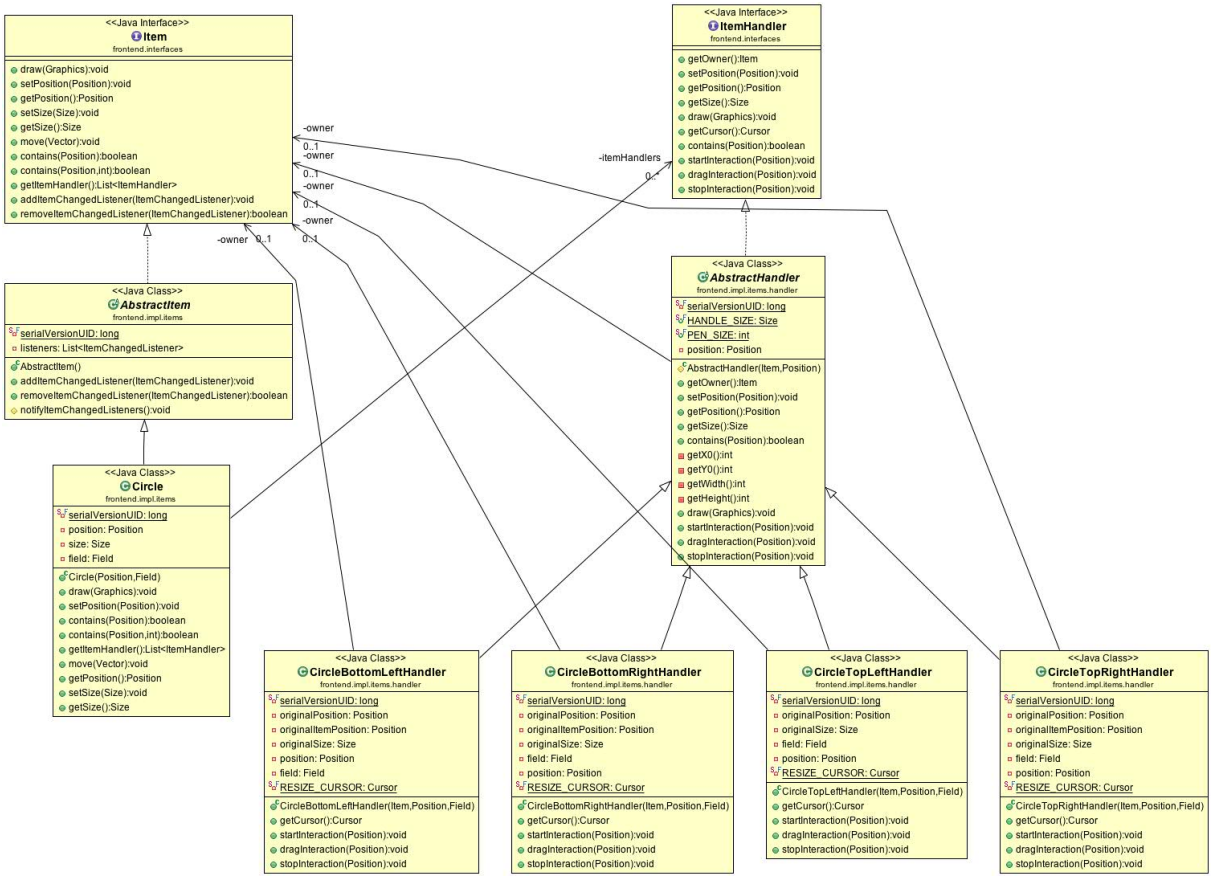


Abbildung 4: UML Handler (Beispiel Circle)

Mit der Vorlage des Observer Patterns wurde für jedes Hindernis ein Listener implementiert. Dieser informiert das Field (in welchem das Hindernis enthalten ist) über diese Änderungen. Das Field besitzt ebenfalls solche Observer, welche die View über allfällige Änderungen informiert. Die View führt schlussendlich dien entsprechende Aktion durch.

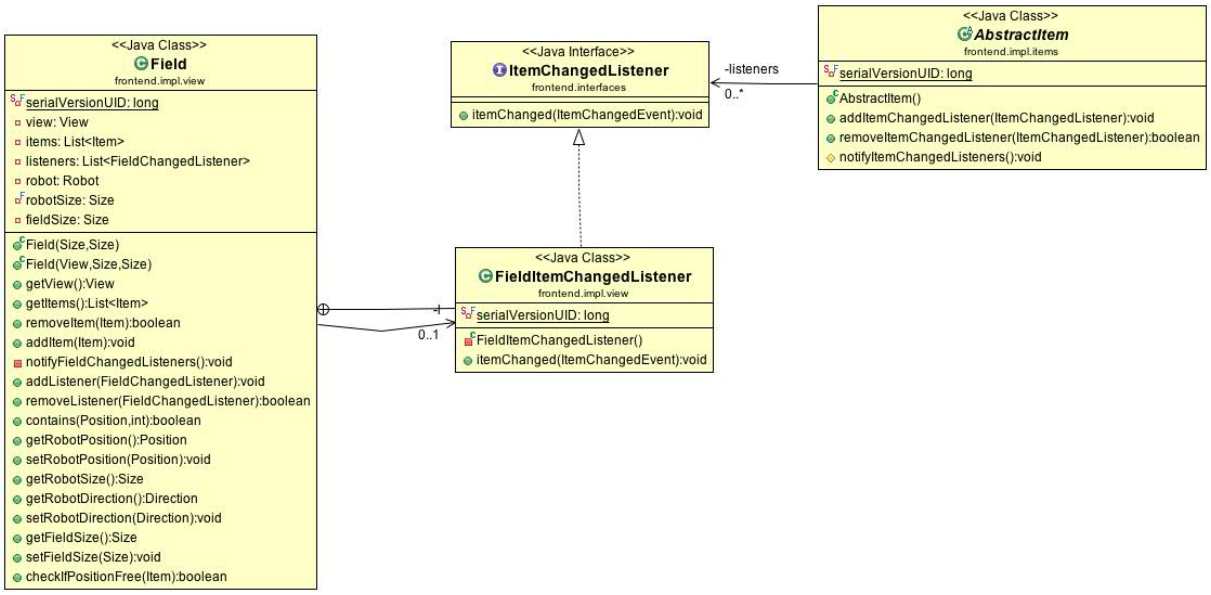


Abbildung 5: UML Observer Hindernisse

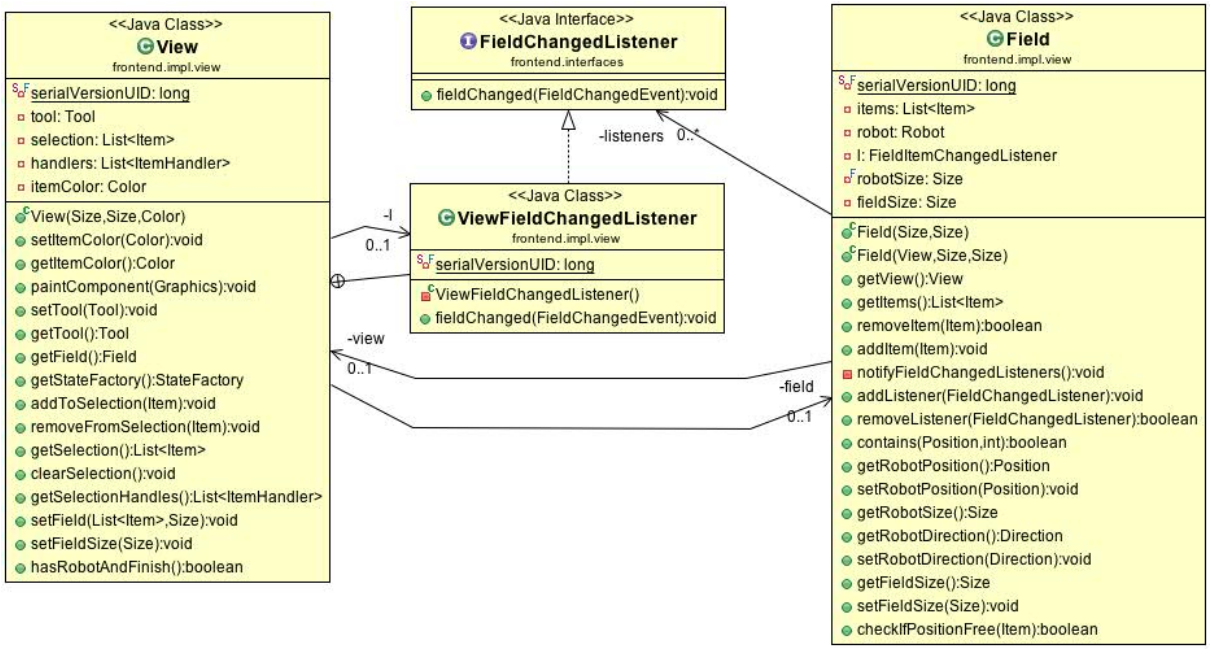


Abbildung 6: UML Observer Field

Das Verschieben und Bearbeiten der Hindernisse verwaltet das Selection Tool. Dieses verarbeitet anhand diverser States (gemäss dem State Pattern) die Mausaktivitäten.

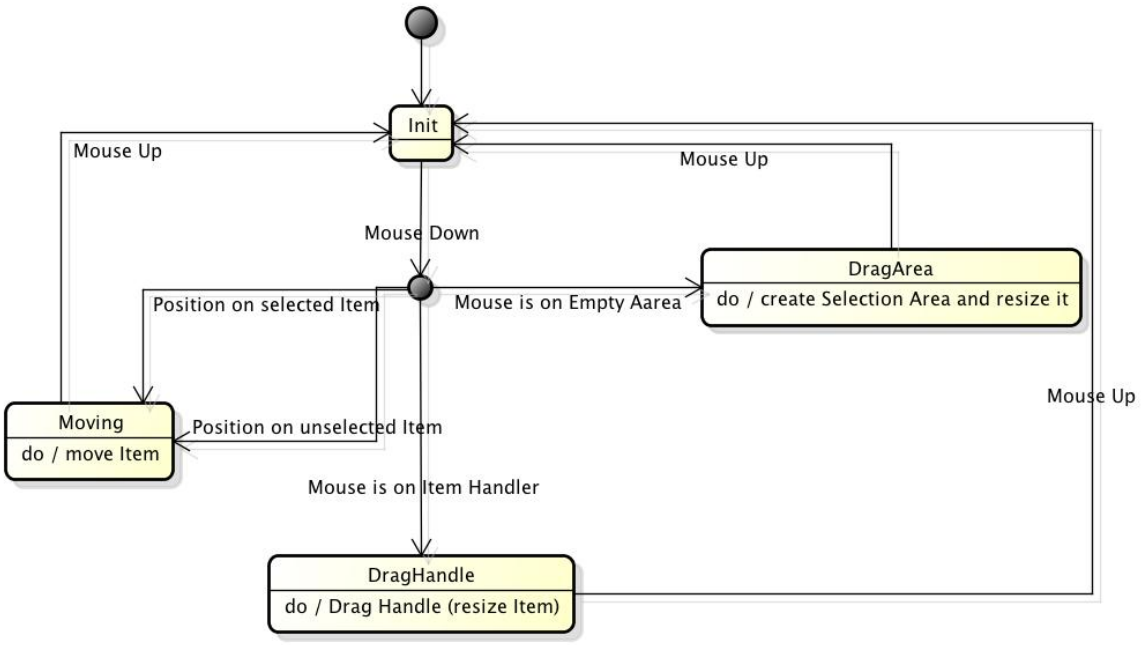


Abbildung 7: Selection State Diagramm

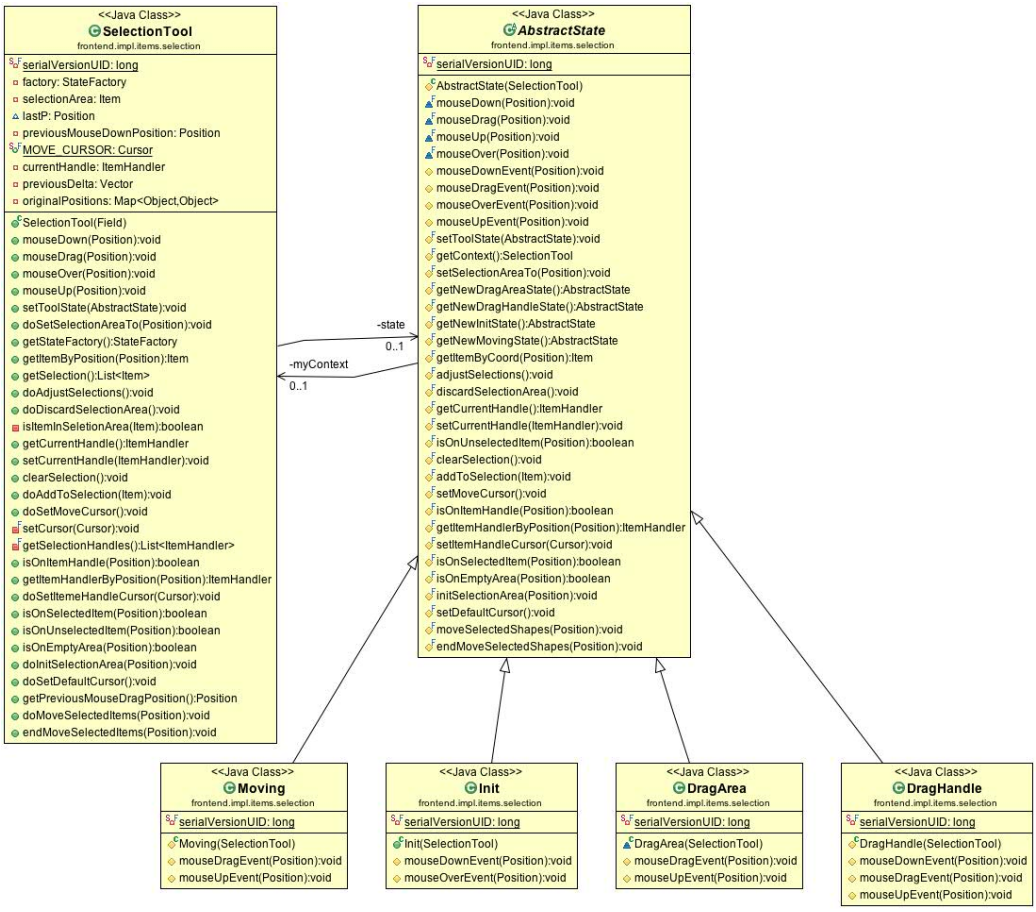


Abbildung 8: UML Selection

Robot

Use Cases in kürze

Für detaillierte Informationen nehmen Sie das Dokument *Use Case Robot* zur Hand.

Ziel suchen

Der Roboter sucht nach einem bestimmten Algorithmus das Spielfeld ab, d.h. Er kann sich fortbewegen, jeweils -90° und +90° scannen und die Hindernisse und die Spielrandfläche so erforschen.

Implementation

Die Roboter suche wird mit Hilfe eines Treads gestartet und falls nötig abgebrochen.

Strategie

Die Strategie wurde anhand des Strategy Patterns umgesetzt. So können ohne grosse Modifikationen neue Strategien hinzugefügt werden.

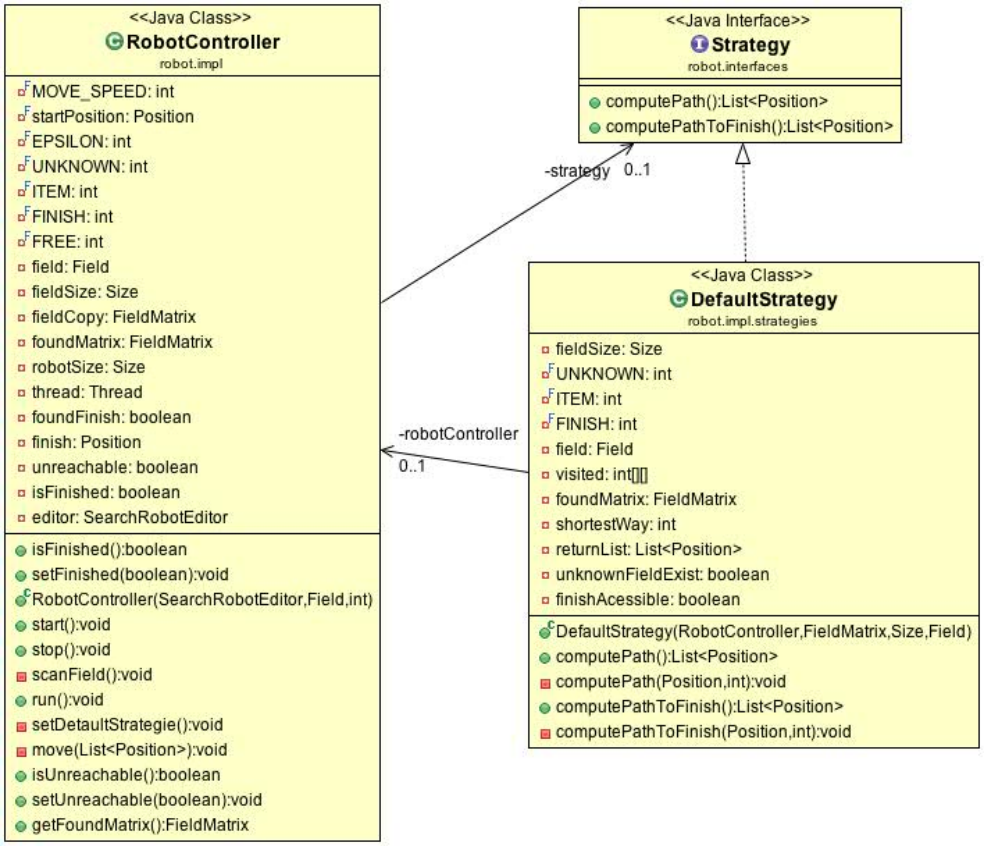


Abbildung 9: UML Strategie

Strategie

Die Default Strategie die wir implementiert haben funktioniert nach folgendem Sequenze Diagram.

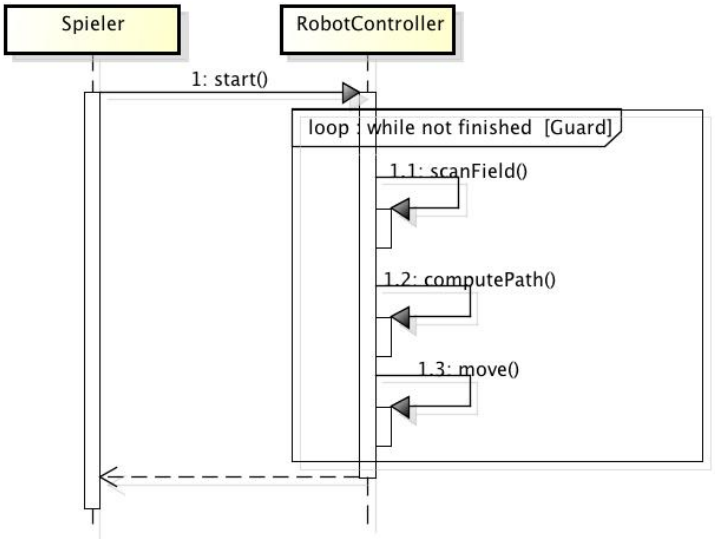


Abbildung 10: SD Strategie

Die Wegberechnung (*computePath*) sucht jeweils den kürzesten Weg zu einem unbekannten Feld.

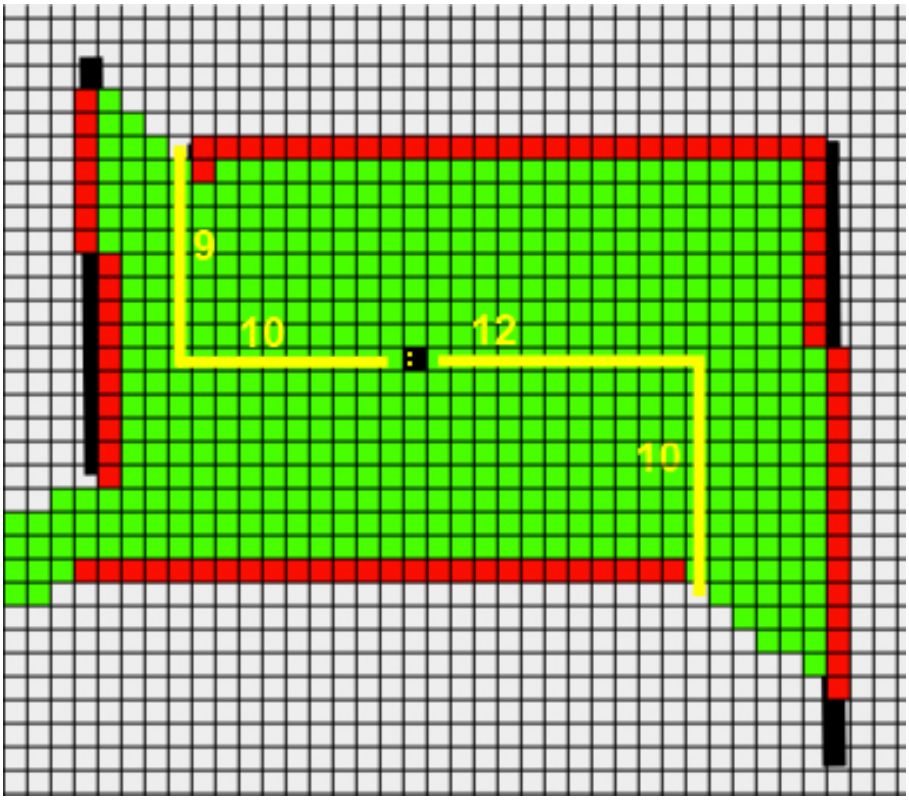


Abbildung 11: Visualisierung Wegberechnung