**Fahrzeugeinsatzplaung**

# Entwicklung einer Anwendung zur Optimierung

Oliver Pohling, David Mittelstädt, Henrik Voß

9. Juli 2014

Was für ein Projekt!

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Problembeschreibung

## 1.2 Anforderungen

- Fahrzeug
  - Ein Fahrzeug kann einen Produkttyp transportieren
  - Geschwindigkeit
  - Kapazität
  - Zeitfenster
  - Start- und End-Depot
- Produkt
  - Name
- Auftrag
  - Beliebige Auf- und Ablade-Station
  - Zeitfenster
- Station
  - Namen
  - Koordinaten

# 2 Softwarearchitektur

## 2.1 Systemdesign

Das Systemdesign setzt sich im wesentlichen aus den folgenden vier Modulen zusammen:

1. Konfigurationsgenerierung
2. Konstruktionsverfahren
3. Genetisches Optimierungsverfahren
4. Ergebnisvisualisierung

Desweiteren sind zwei Datenmodelle definiert:

1. Konfiguration-Datenmodell
2. Ergebnis-Datenmodell

## 2.2 Konfigurations-Datenmodell

Das Konfiguration-Datenmodell beschreibt die Konfiguration, die vom Anwender zur Verfügung gestellt wird. Sie umfasst Stationen, Fahrzeuge, Produkte, Aufträge und Zeitfenster (siehe Abbildung 1).

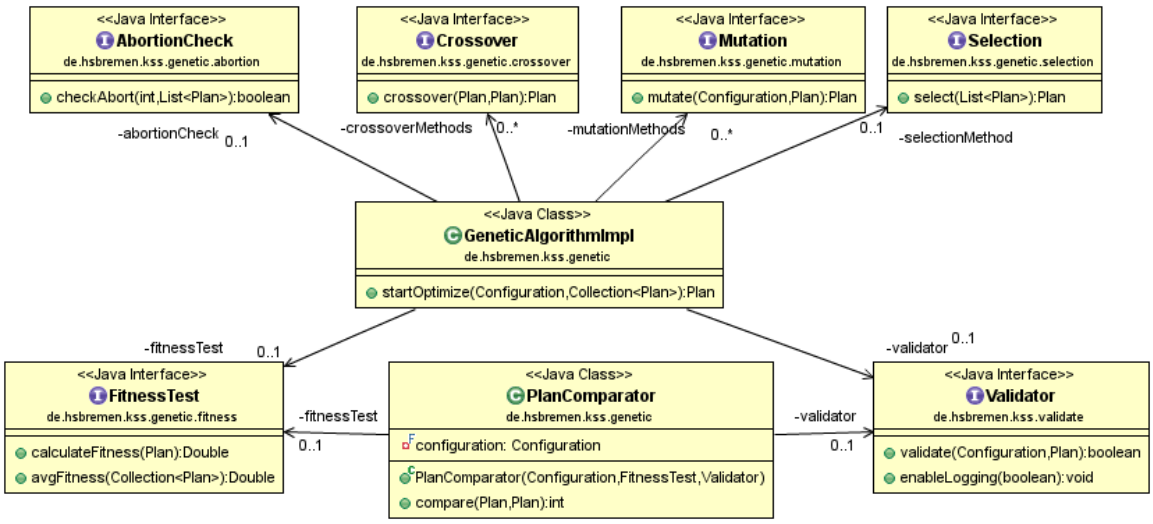


Abbildung 1: Klassendiagramm vom genetischen Algorithmus

## 2.3 Ergebnis-Datenmodell

## 2.4 Konfigurationsgenerierung

# 3 Umsetzung

## 3.1 Verwendete Bibliotheken

Im Rahmen der Entwicklung wurden diverse OpenSource - Bibliotheken eingeführt, um die Entwicklung zu beschleunigen und die Qualität der Software zu steigern. Im folgenden werden die verwendeten Bibliotheken beschrieben:

**Apache Commons Math** (Version 3.2, Apache License 2.0) [11] Stellt mathematische Funktionen zur Verfügung. Wird unteranderem für die Berechnung der Entfernungen verwendet.

**Apache Commons Lang** (Version 3.3.1, Apache License 2.0) [10]

**Apache Commons Collections** (Version 4.0, Apache License 2.0) [9] Erweitert das Java-Collections-Framework.

**Joda-Time** (Version 2.3, Apache License 2.0) [4]

**JUnit** (Version 4.11, Eclipse Public License) [5] JUnit ist die Standard-Bibliothek für Unit-Tests unter Java.

**Hamcrest** (Version 1.3, BSD 3-Clause) [3]

**Easymock** (Version 3.2, Apache License 2.0) [1]

**SL4J** (Version 1.7.6, MIT license) [8]

**Logback** (Version 1.1.1, Eclipse Public License v1.0 und LGPL 2.1) [7]

**Google Guave** (Version 17.0, Apache License 2.0) [2]

**JFreechart** (Version 1.0.17, LPGL) [6]

## 3.2 Grafische Oberfläche

# 4 Experimente

# 5 Ausblick

# 6 Fazit

## Literatur

[1] EasyMock contributors: *EasyMock*. http://easymock.org/. Version: 2014. – [Online; 9. Juli 2014]

[2] Google: *Guave*. https://code.google.com/p/guava-libraries/. Version: 2014. – [Online; 9. Juli 2014]

[3] hamcrest.org: *Hamcrest*. http://hamcrest.org/. Version: 2014. – [Online; 9. Juli 2014]

[4] Joda: *Joda-Time - Java date and time API*. http://www.joda.org/joda-time/. Version: 2014. – [Online; 9. Juli 2014]

[5] JUnit: *JUnit*. http://junit.org/. Version: 2014. – [Online; 9. Juli 2014]

[6] Object Refinery Limited: *JFreeChart*. http://www.jfree.org/jfreechart/. Version: 2014. – [Online; 9. Juli 2014]

[7] QOS.ch: *Logback Project*. http://logback.qos.ch/. Version: 2014. – [Online; 9. Juli 2014]

[8] QOS.ch: *Simple Logging Facade for Java (SLF4J)*. http://http://www.slf4j.org/. Version: 2014. – [Online; 9. Juli 2014]

[9] The Apache Software Foundation: *Commons Collections*. http://commons.apache.org/proper/commons-collections/. Version: 2014. – [Online; 9. Juli 2014]

[10] The Apache Software Foundation: *Commons Lang*. http://commons.apache.org/proper/commons-lang/. Version: 2014. – [Online; 9. Juli 2014]

[11] The Apache Software Foundation: *Commons Math: The Apache Commons Mathematics Library*. http://commons.apache.org/proper/commons-math/. Version: 2014. – [Online; 9. Juli 2014]