# 800Anwendungsumgebung

* Windows 10 64 bit
* IntelliJ IDEA
* Java JDK 8 32 bit (achten sie darauf, dass es Probleme geben kann, wenn parallel eine 64 bit Version installiert ist!)
* Visual Studio Express 2015 Desktop Edition
* AIMSUN 8
* Python 2.7.6 (64 bit)
* SQL Server 2014 Standard (optional)

# Umgebungsvariablen

Um wichtige Include-Dateien finden zu können, werden Umgebungsvariablen verwendet.

Unter *Systemsteuerung -> System -> Erweiterte Systemeinstellungen ->Umgebungsvariablen* können Sie Systemvariablen bearbeiten und anlegen.

* Umgebungsvariable JAVA\_HOME anlegen:

zeigt auf das JDK, z. B. *%PROGRAMME%\Java\JDK\_1.8.0*

* Umgebungsvariable CLASSPATH anlegen:

Pfad zum Ordner *Aimsun-Modell/java* des OTC-Projektes angeben.

* Umgebungsvariable PYTHONPATH anlegen: z.B. *%PATH\_TO\_AIMSUN%\programming\Scripting\libs\*
* Umgebungsvariable PATH erweitern:

Pfad zu Python

Pfad zum bin-Ordner des JDK (%PROGRAMME%\Java\jdk\bin)

Pfad zum Ordner plugins im AIMSUN-Ordner

Pfad zum Hauptordner von AIMSUN

# Externe Libraries

Ein weiterer wichtiger Schritt ist es, alle verwendeten Java-Libraries, welche sich im Projektordner OTC\Java\libs\ befinden, in den jre\lib\ext\ Ordner des JRE zu kopieren. So sind die Libraries auf jeden Fall von den zum Einsatz kommenden Java-Klassen auffindbar.

Der Pfad zum ext-Ordner im JRE kann wie folgt lauten:

%PROGRAMME%\Java\jre8\lib\ext\

## IntelliJ Idea

Importieren Sie das Java-Projekt (Ordner java):

* *File -> Open* um Java-Ordner navigieren und OK
* Build-Ordner: AimsunModel/java
* Pakete JUnit und Annotations einbinden

## Visual Studio

Das Visual-Studio-Projekt für die API (Schnittstelle zwischen AIMSUN und Java-Projekt) sind so eingestellt, dass kompilierte Dateien an die richtigen Stellen gelegt werden. Die API-DLL muss kompiliert werden.

Öffnen sie die entsprechende Solution-Datei (aapi.sln) und erstellen sie die DLLs *JavaEventManager* (Release) und *Release* (Erstellen -> Projektmappe neu erstellen).

## AIMSUN

*javalauncher.dll* im Ordner AIMSUN-Modell enthält ein Python-Modul zum Starten der JVM aus einem Skript heraus. Die DLL muss in den Ordner im PYTHONPATH kopiert werden.

Um Layer 2 mit AIMSUN zu starten, muss die Datei *opt\_InternalFTC8.bat* ausgeführt werden. Denken Sie daran, für jedes ANG-Netz eine passende Batchdatei anzulegen, da das gewählte Netz angegeben werden muss.

# Start eines Optimierungslaufes

1. AIMSUN mit einem ANG-Modell starten (hier direkt ANG-File durch Doppelklick starten und nicht erst AIMSUN und dann laden, da sonst das Arbeitsverzeichnis falsch gesetzt wird!).

Replication über Rechtsklick *Batch simulation* starten.

Man kann auch einen Average auswählen, alle gewählten Replications im Average (zugänglich über Properties) auf *not simulated* setzen, *simulate pending replications*.

1. Die Java-OTC-GUI sollte sich öffnen und die Simulation auf Echtzeit verlangsamt starten.
2. Nun die Batch-Datei für Layer 2 ausführen. Das kann man nicht vorher machen, weil dann die RMI-Registry noch nicht verfügbar ist. Aufpassen, dass hier das richtige ANG-File passend zum ausgeführten Netzwerk angegeben ist!
3. Um zu beschleunigen, kann man, sobald Ebene 2 alle Fenster aufgebaut, in der Layer 1 GUI via *Action -> Manual unbusy* die Simulation beschleunigen. Sie sollte kurz verzögert werden, wenn Optimierungsläufe auf Ebene 2 angestoßen werden.

Tipp: Erhöhen sie die Anzahl der Simulation Threads in den Einstellungen eines Experiments um die Simulation zu beschleunigen!

# Debugging

## Java – Seite (OTC Manager)

In IntelliJ IDEA über Run 🡪 Edit Configurationseine neue Remote Java Application über das Plus erstellen.

**Transport**: Socket

**Debugger mode**: Attach

**Host**: localhost

**Port**: 8000

Im ANG-File unter Scenario->Replication mit rechtsklick die Eigenschaften öffnen und unter Kenngrößen bei Attach Java Debugger eine 2 (oder 1) eintragen.

Jetzt zuerst die Replication starten, dann in der Java-Entwicklungsumgebung den Debugger mit Run 🡪 Debug anhängen.

## C - Seite (Aimsun API)

Im Menü *Debuggen* auf *Starten* klicken (oder *F5* drücken). Es startet AIMSUN (eine Warnung, dass die geladene Anwendung keine Symbole für den Debugger enthält, kann man ignorieren). Hier ist manuell das ANG-Modell zu laden und die Simulation zu starten. Erst jetzt wird die DLL geladen und der Debugger kann eingreifen.

Falls dies nicht klappt, ist die häufigste Ursache, dass vom Modell die falsche DLL (also eine DLL, die nicht gegen die Debug-DLL gelinkt wurde) geladen wird. Dann vorher die Debug-DLL erst neu erstellen und den Debug-Vorgang erneut starten.

Es können Breakpoints gesetzt werden, sowie mit F5, F10 und F11 Schrittweise durch den Code gelaufen werden.