Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Prof. Dr. Chris Biemann
Seid M. Yimam

Language Technology Lab
Universität Hamburg

Outline

- Organizational Stuff
- Introduction to UIMA
- Introduction to MAVEN
- Software Setup
- Further Reading

Outline

- Organizational Stuff
- Introduction to UIMA
- Introduction to MAVEN
- Software Setup
- Further Reader

Practice Class

- When: Tuesday 12:15 – 13:45
- Where: G102
- Who: Seid M. Yimam (yimam@informatik.uni-hamburg.de) F-415

Exercises

- Exercises should be submitted before the next tutorial
- Machine learning and final projects will have separate deadlines
- Machine learning projects and Final projects can be done in groups
- Exercise will be out of 100 points and later converted to 50 points.

Exercises

- You must use your own computer to do the exercises
- All exercises will use **Java** as programming language using **UIMA**

Course Outline

Week	Topic
1	UIMA Setup
2-4	Getting to know UIMA components
5	ClearTk-ML setup and introduction, POS tagging example
6	Machine learning project: NER or chunking
7	Feature engineering
8	Machine learning project presentations
9	Project: setup of provided standard pipeline
10	Project idea presentations (small groups)
11-13	Supervision of project groups (by appointment)
14	Project results presentations

Outline

- Organizational Stuff
- Introduction to UIMA
- Introduction to MAVEN
- Software Setup
- Further Reader

What is UIMA ?

- UIMA = Unstructured Information Management Architecture

- A component-based architecture for **analysis** of unstructured information (e.g., natural language text)
  - ***structured information*** – intended meaning is unambiguous and explicitly represented – e.g. relational database table
  - ***unstructured information*** - information whose intended meaning is only loosely implied by its form – e.g natural language document, voice…
  - "**Analysis**" means deriving a structure from the unstructured data

- Works like an assembly line:
  - take the raw material
  - refine it step by step
  - drive off with a nice car

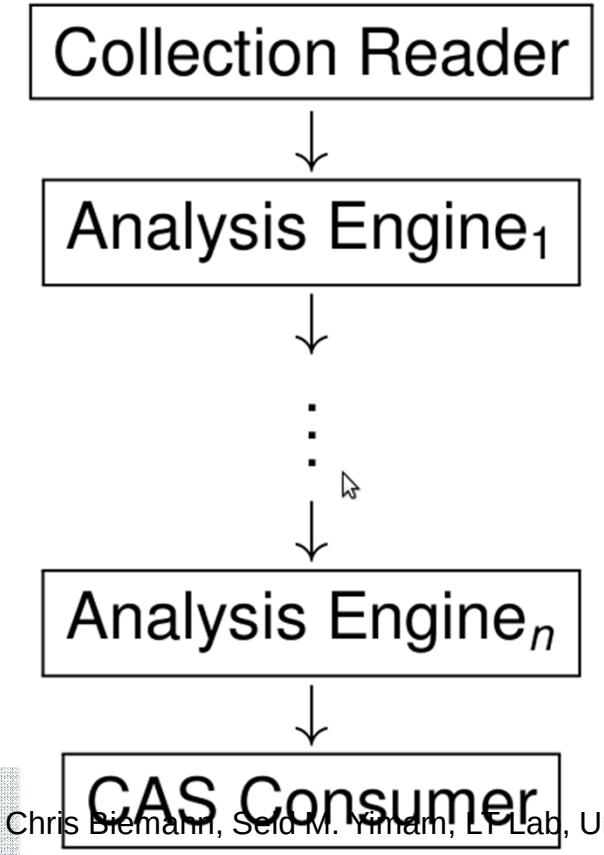
History of UIMA

- Originally developed by IBM
- Apache UIMA: Free Software Java and C++ implementations of the UIMA specification

- Used in many academic and commercial contexts:
  - **biomedical NLP** systems
  - **ClearTK** (statistical machine learning for NLP)
  - **DKPro** (general NLP framework)
  - **Watson** (question answering)

UIMA Pipeline

- An aggregation of UIMA components
- Flow of data from each component



Collection Reader

- Iterates over a collection of documents (e.g. from a folder)

Reader

Collection Reader

- Creates an empty data structure (Common Analysis System (CAS)), which holds objects, values and properties

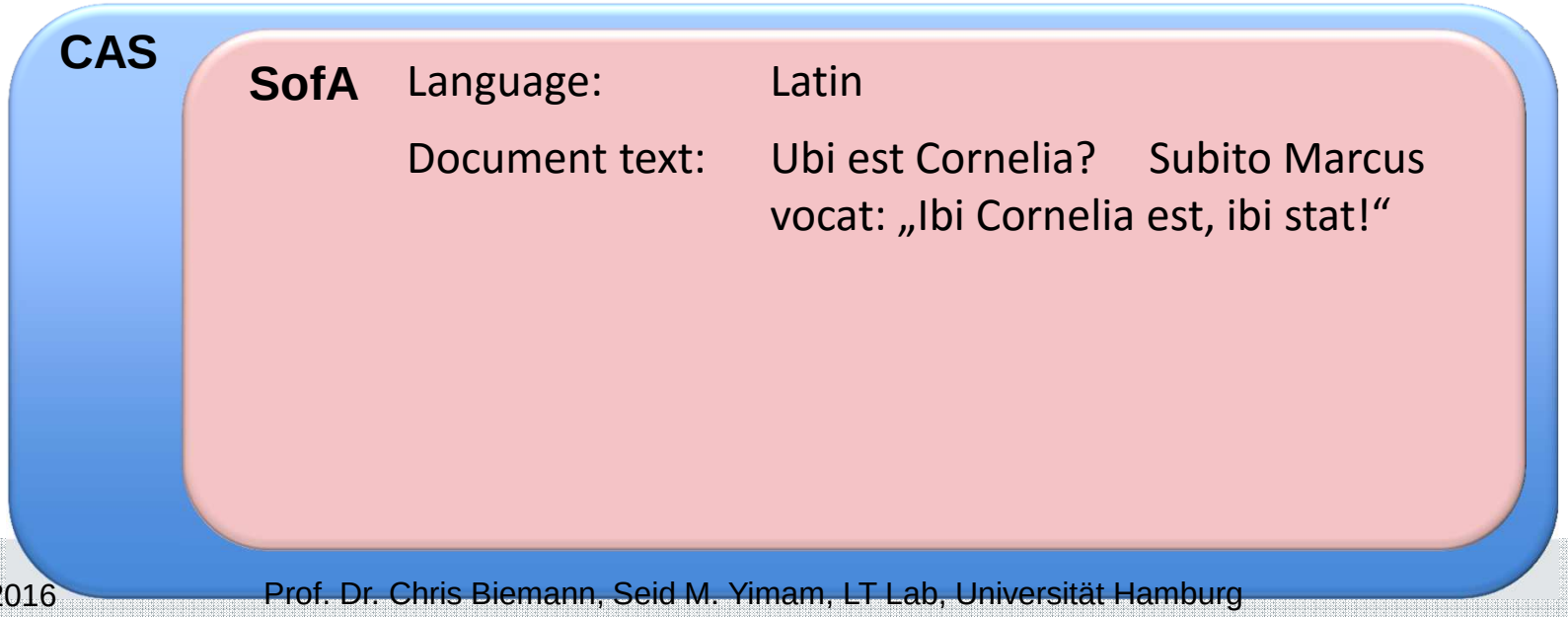
Collection Reader

- Each CAS has one or more **views**, each corresponding to a **Subject of Analysis (SofA)**

Collection Reader

- Set the document text to the CAS and some properties of the text (e.g. language, encoding, etc.)

Reader



Types

- To describe text (e.g. words) we need Types
- UIMA defines a few basic **types**
- Types have properties (called **features**)
  - Example: The type Person migh have a feature „Age“ and „Gender“
- Types can be extended to define arbitrarily rich domain- and application-specific type systems
- A type system defines the various kinds of objects that may be discovered by components which subscribe to that type system
- The (frequently subclassed) **Annotation type** is used to label regions of a document
- Annotations include “**begin**” and “**end**” features

Eclipse UIMA type system editor

Type System Definition

Types (or Classes)

The following types (classes) are defined in this analysis engine descriptor.
The grayed out items are imported or merged from other descriptors, and cannot be edited here. (To edit them, edit their source files).

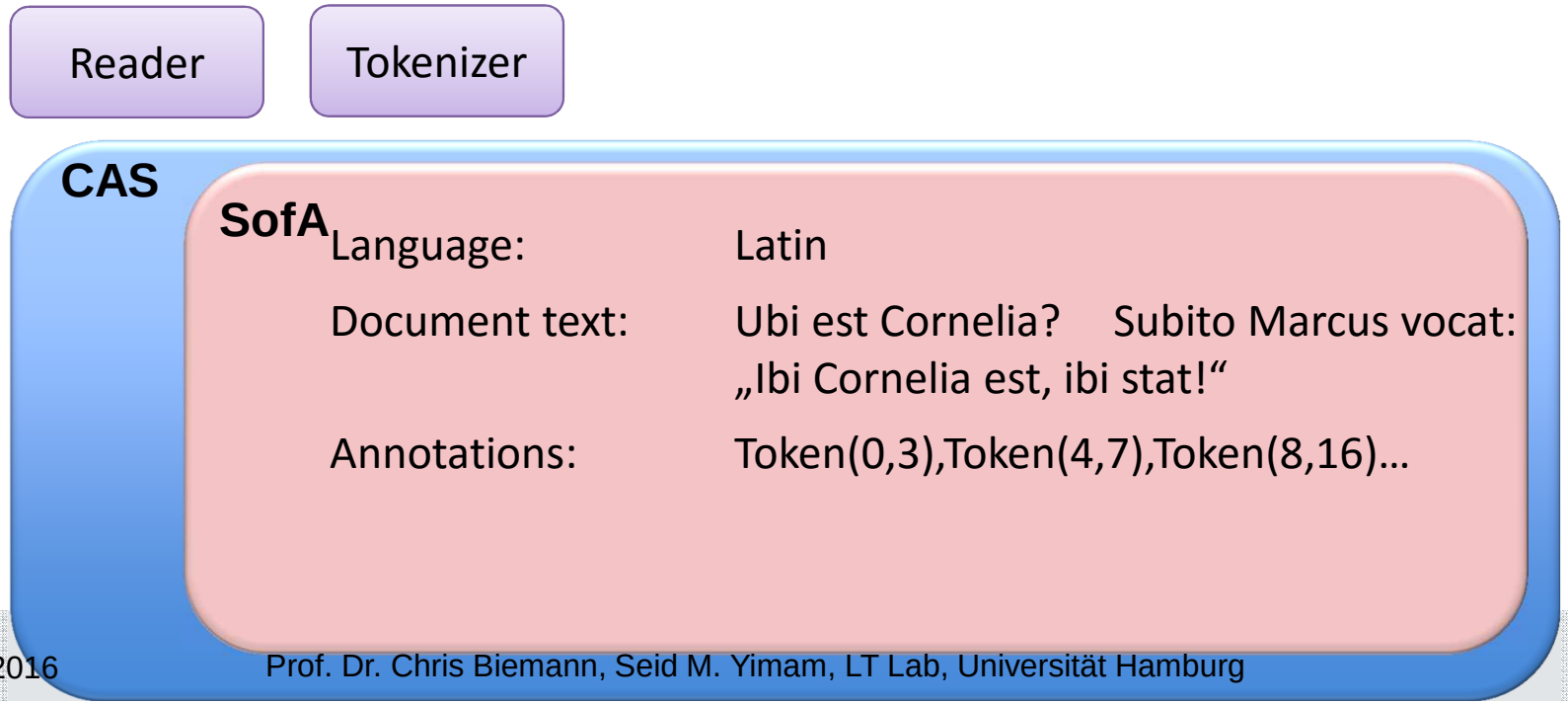
Type Name or Feature Name	SuperType or Range
org.apache.uima.examples.tokenizer.Sentence	uima.tcas.Annotation
org.apache.uima.examples.tokenizer.Token	uima.tcas.Annotation
org.apache.uima.tutorial.DateAnnot	org.apache.uima.tutorial.DateTimeAnnot
org.apache.uima.tutorial.DateTimeAnnot	uima.tcas.Annotation
shortDateString	uima.cas.String
org.apache.uima.tutorial.Meeting	uima.tcas.Annotation
room	org.apache.uima.tutorial.RoomNumber
date	org.apache.uima.tutorial.DateAnnot
startTime	org.apache.uima.tutorial.TimeAnnot
endTime	org.apache.uima.tutorial.TimeAnnot
org.apache.uima.tutorial.RoomNumber	uima.tcas.Annotation
org.apache.uima.tutorial.TimeAnnot	org.apache.uima.tutorial.DateTimeAnnot

Add Type
Add...
Edit...
Remove
Export...
JCasGen

- JCasGen creates Java classes out of the XML file

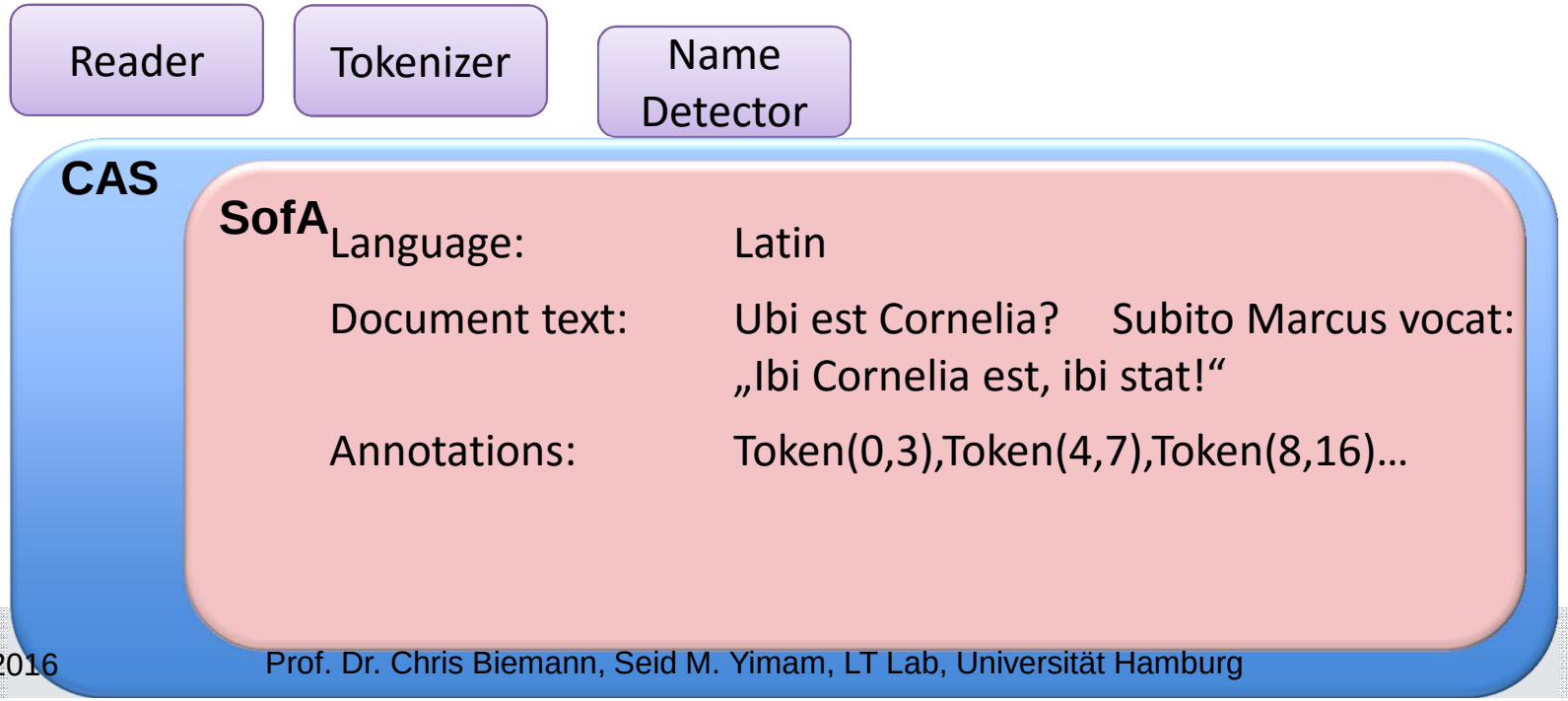
Analysis Engine

- Pass the cas to the next **Analysis Engine (AE)** „Tokenizer“
- Each AE could derive some structure and record it as an annotation



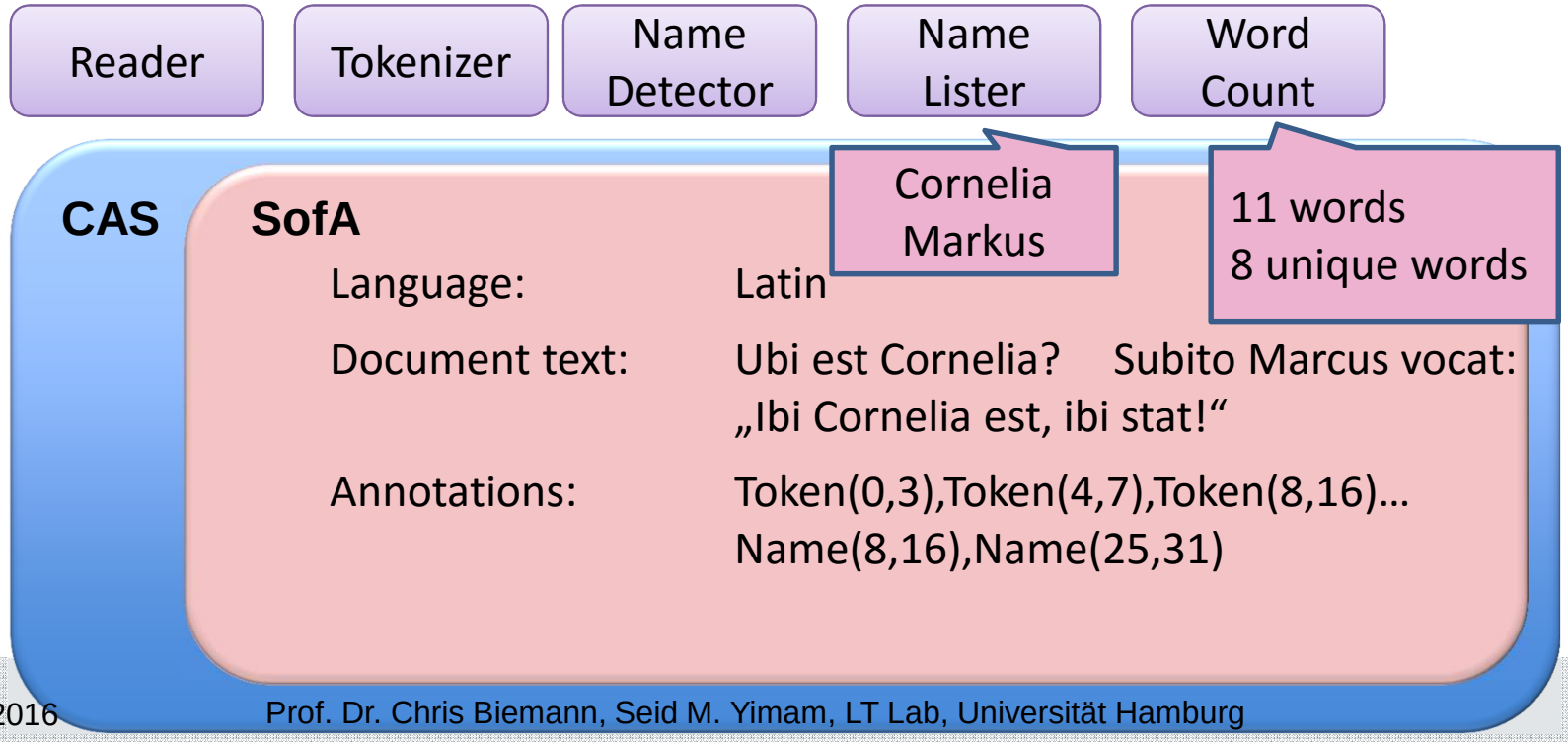
Analysis Engine

- Pass the cas to the next **Analysis Engine (AE)** „Tokenizer“
- Each AE could derive some structure and record it as an annotation



Consumer

- **CAS Consumer** do the „final“ processing
- Normally they are used to analyze, store and display Annotations of interest



Outline

- Organizational Stuff
- Introduction to UIMA
- Introduction to MAVEN
- Software Setup
- Further Reader

Maven

- Maven is a Yiddish term meaning *„accumulator of knowledge“*
- A **build** management tool
- Developed by Apache
- Primarily developed for Java development

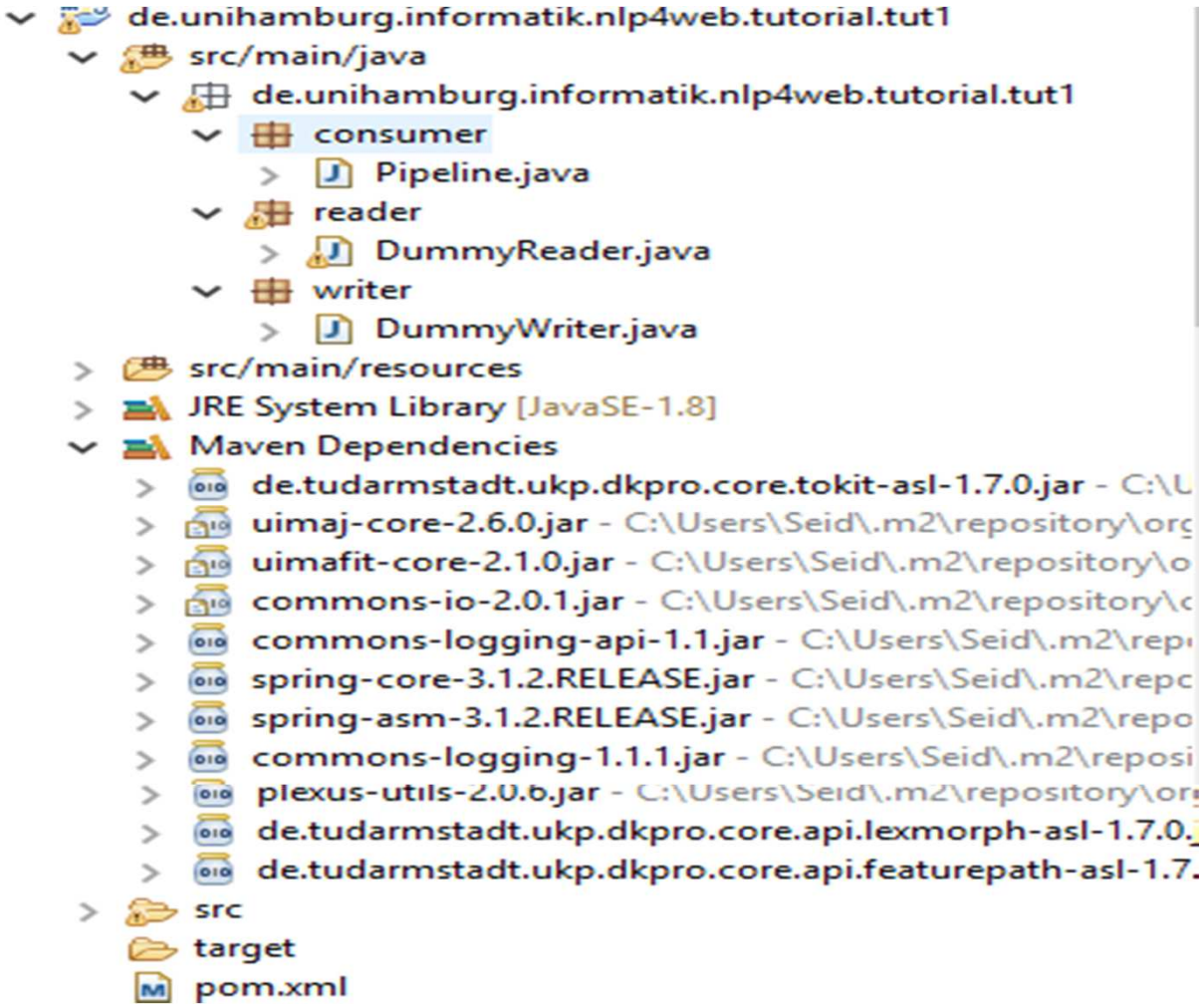
Maven Components

- Project object model (**POM**)
  - Describes all configurations of a particular project
    - Project name, project owner, dependencies on other projects
  - Defined in the file **pom.xml**
- **Plugins**
  - A set of goals to be executed
  - used to: create jar files, create war files, compile code, unit test code, create project documentation, and on and on.
- Build life cycles
  - A list of named **phases which give an order to goal execution**
- **Dependency management** model
  - Project dependencies are stored on **repository servers**

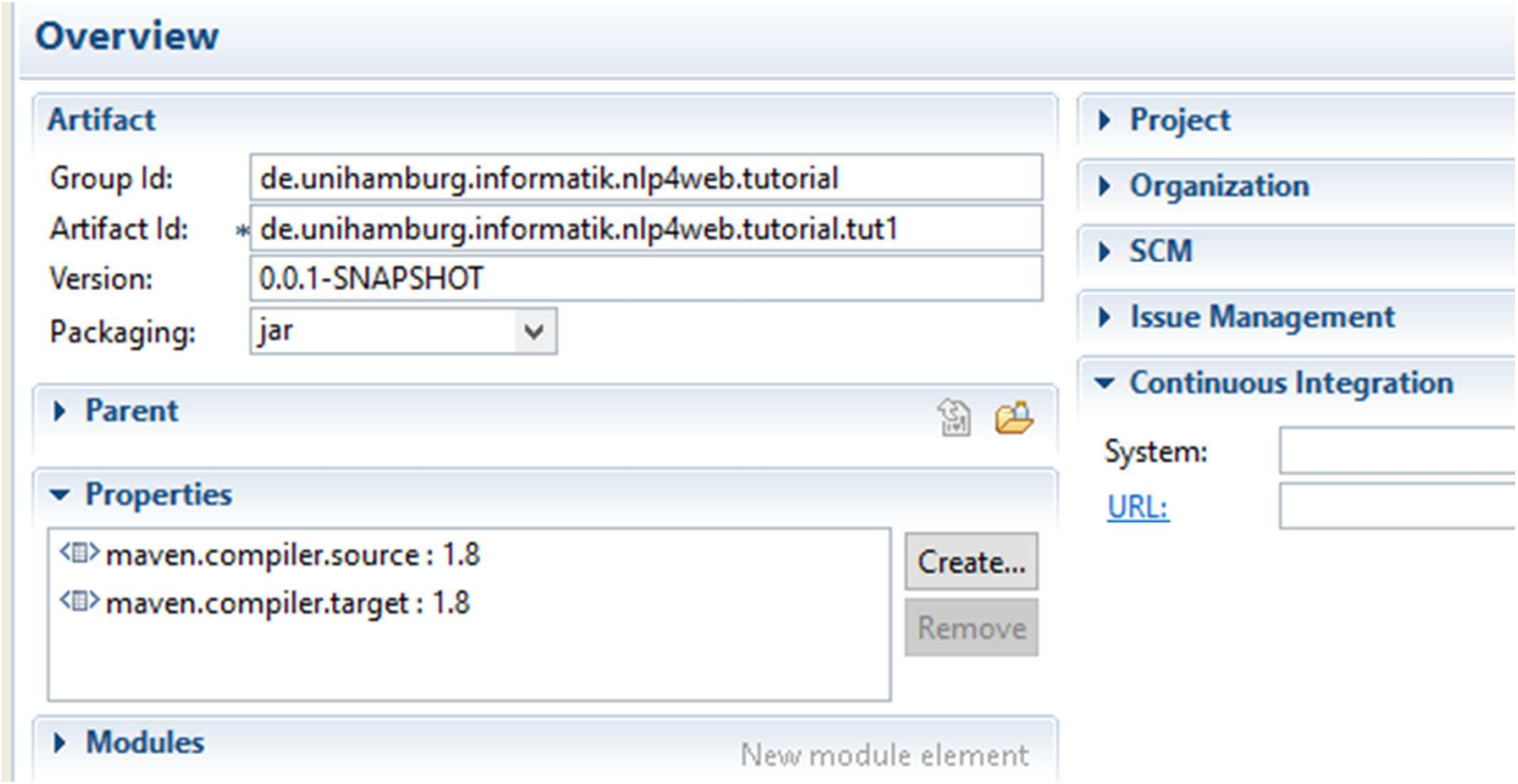
Maven Repository

- Store a collections of **artifacts used by Maven during dependency resolution** for a project
- An artifact is usually bundled as a JAR file containing the binary library or executable
- **Project coordinates** are a group of values which uniquely identify a project:
  - **Group ID**: The entity responsible for the artifact
    - e.g., de.unihamburg.lt.nlp4web
  - **Artifact ID**: The name of the artifact
    - e.g., de.unihamburg.lt.nlp4web.tut01
  - **Version**: The version number of the artifact
    - e.g., 1.0.0

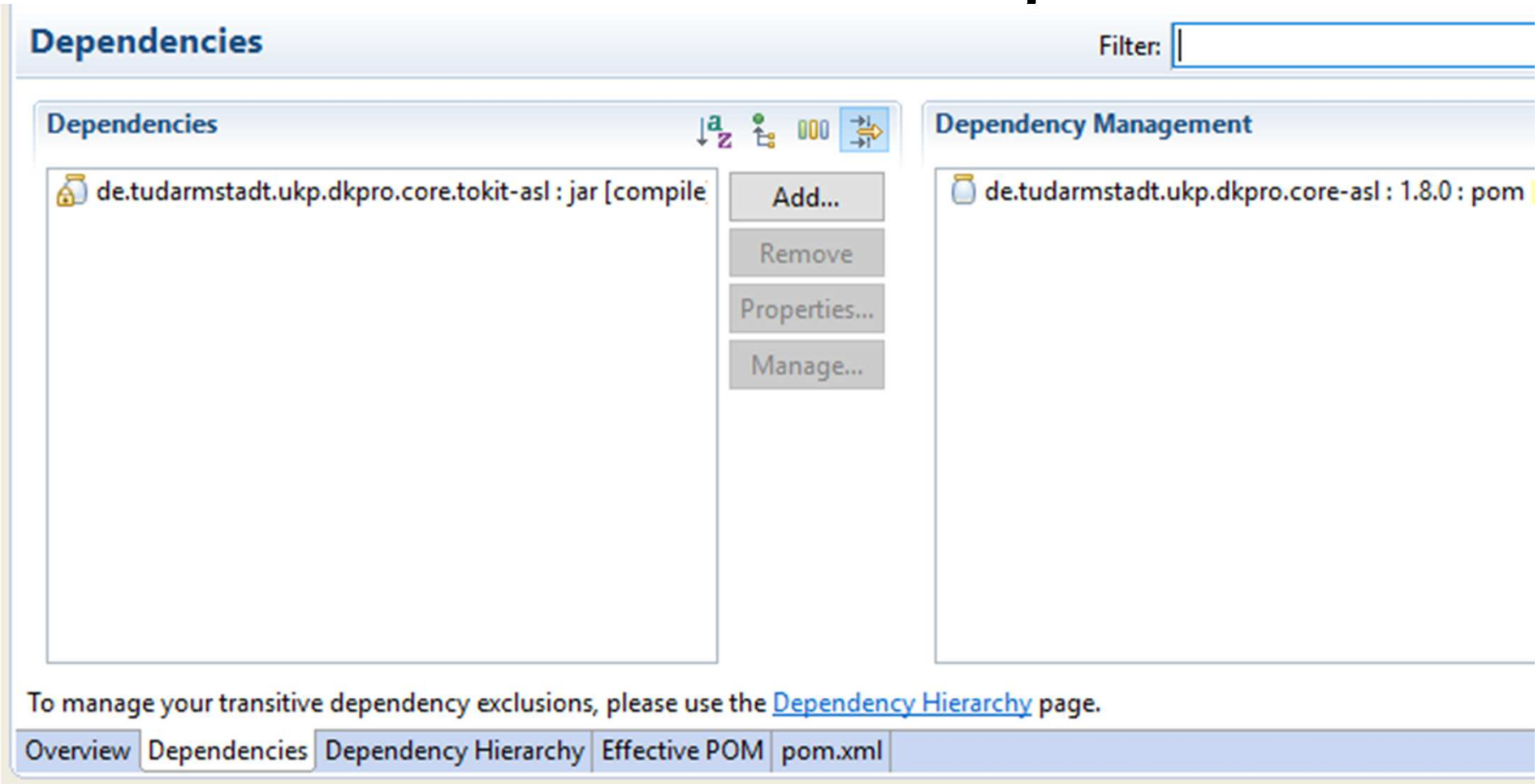
Maven projects in Eclipse: Package Explorer



Maven projects in Eclipse: POM Editor Overview



Maven projects in Eclipse: POM Editor Dependencies



Outline

- Organizational Stuff
- Introduction to UIMA
- Introduction to MAVEN
- Software Setup
- Further Reader

Java Development Kit

- If you don't already have one, download and install a Java Development Kit such as OpenJDK 8 or Java SE 8 JDK: (You can use Java 7 if you like)
  - Debian
    - **sudo add-apt-repository ppa:openjdk-r/ppa**
    - **sudo apt-get update**
    - **sudo apt-get install openjdk-8-jdk**
    - **sudo update-alternatives --config java** (if you have multiple java)
  - Fedora **su -c yum install java-1.8.0-openjdk.x86_64** (we don't check this)
  - Other Go to http://jdk6.java.net/download.html

Eclipse

- Download and install the version appropriate for your system from https://eclipse.org/downloads/
- We use Version:**Neon.1a Release (4.6.1)** but you can use earlier versions if you have existing installation – it should work

Maven

- Maven will be accessed via Eclipse plugin
- We do **not** need to install a standalone version

Eclipse Plugins:

- Install the following plugins
- Go to Help→ install New Software…
- Add the following (restart might required in between)
  - Apache UIMA Eclipse tooling and runtime support http://www.apache.org/dist/uima/eclipse-update-site

Run the sample project

- Download and unpack this tutorial's sample project from STiNE or get it from a USB.
- File → Import… → Maven → Existing Maven Projects → Next
- Browse for and select the directory you unpacked
- Select the pom.xml file and click Finish
- The project will now appear in your Package Explorer.
- Navigate to the file **Pipeline.java** and open it.
- Run → Run As → Java Application
- Open the Progress tab to confirm that the dependencies are being retrieved
- Open the Console tab to view the output
- Examine **Pipeline.java** and the other source files to see if you can understand how they work

Outline

- Organizational Stuff
- Introduction to UIMA
- Introduction to MAVEN
- Software Setup
- Further Reading

Exercise 1 – 2 Points

1. Modify the **DummyReader.java** so that in stead of dummy text, it will add sentences from a file. The file news.txt is available under **src/main/resources**
   - How do you read a file from the resource folder?
     - HINT:
       - ClassLoader classLoader = getClass().getClassLoader();
       - File file = **new File(classLoader.getResource(FILENAME).getFile());**
   - How do you get sentences from the file?
     - ASSUME a sentence is given per line
2. Check the result you found from DummyWriter.java when you run Pipline.java. Is the number of sentence the same with the number of lines you had in news.txt? Why do you think they are different?
   - HINT: See the first Analysis engine (AnalysisEngine seg ....)

Further Reading

- Apache UIMA website
  http://uima.apache.org/
  - Everything about Apache UIMA
- Apache UIMA Overview and Setup, Chapter 2
  http://uima.apache.org/d/uimaj-2.9.0/overview_and_setup.pdf
  - An introduction to UIMA similar to this lecture
- Apache UIMA Documentation
  http://uima.apache.org/documentation.html
  - More UIMA tutorials, setup guides, reference manuals, etc.
  - Beware: many of the techniques used for defining AEs, AAEs, etc. described there have been superseded by uimaFIT
- Apache Maven
  http://maven.apache.org/
  - What is Maven?, Feature Summary, Getting Started Tutorial