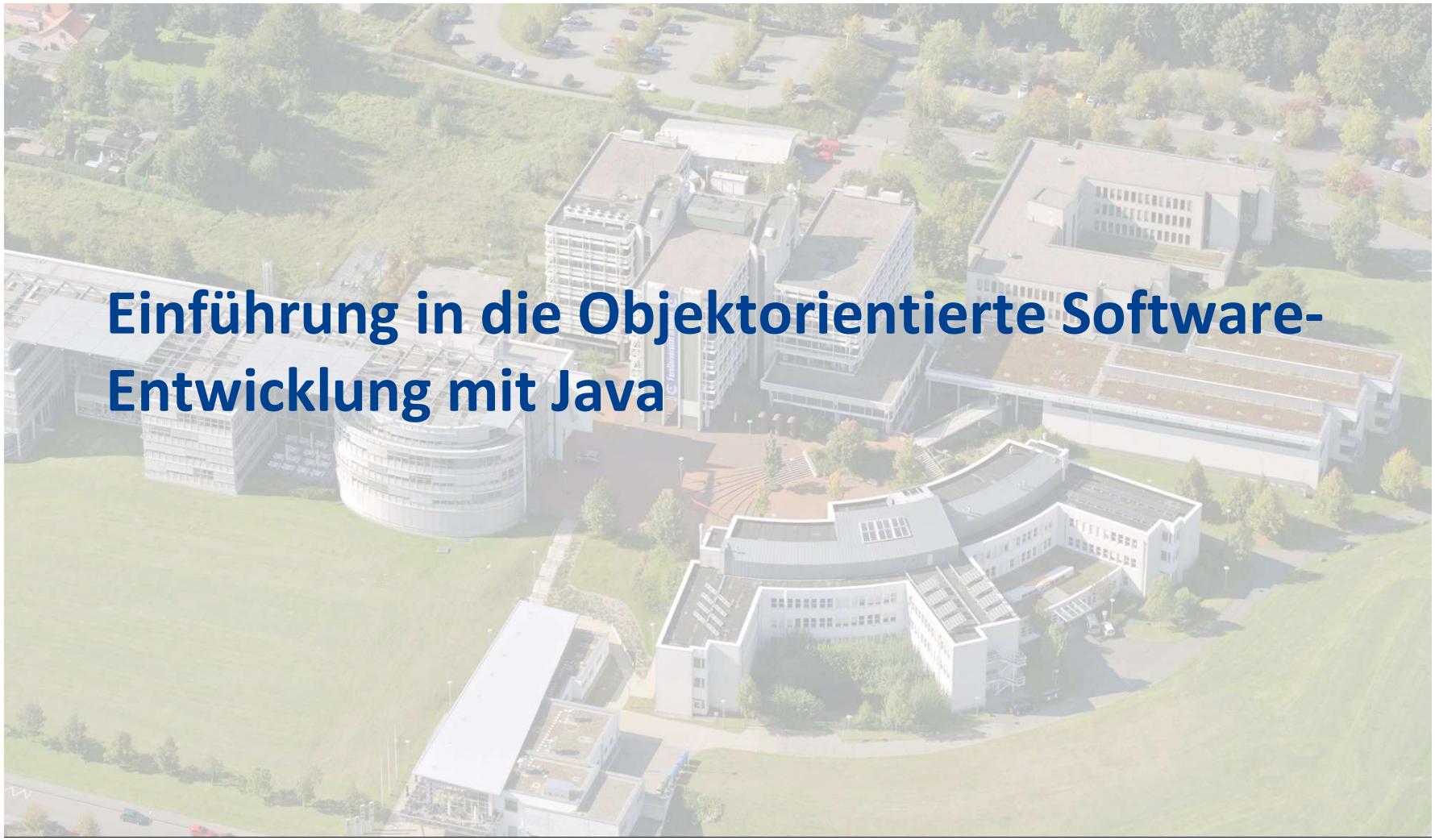


Einführung in die Objektorientierte Software-Entwicklung mit Java

Regionalzentrum Hamburg Michael Zöller

Agenda Samstag

Vorstellung und Überblick

Eclipse

Einstieg „FizzBuzz“

Unit-Testing mit Junit und Mockito

„FizzBuzz“ objektorientiert 1

Git

„FizzBuzz“ objektorientiert 2

„FizzBuzz“ objektorientiert 3

Maven

Euer Dozent

- **Michael Zöller**
- ...aus Hamburg
- …FeU-Alumni in Computer Science, BWL und Winf, ehemals Mentor
- …Software-Entwickler und-Architekt
- …Kontakt:
  - michael2.zoeller@gmail.com
  - @zettsystems auf twitter

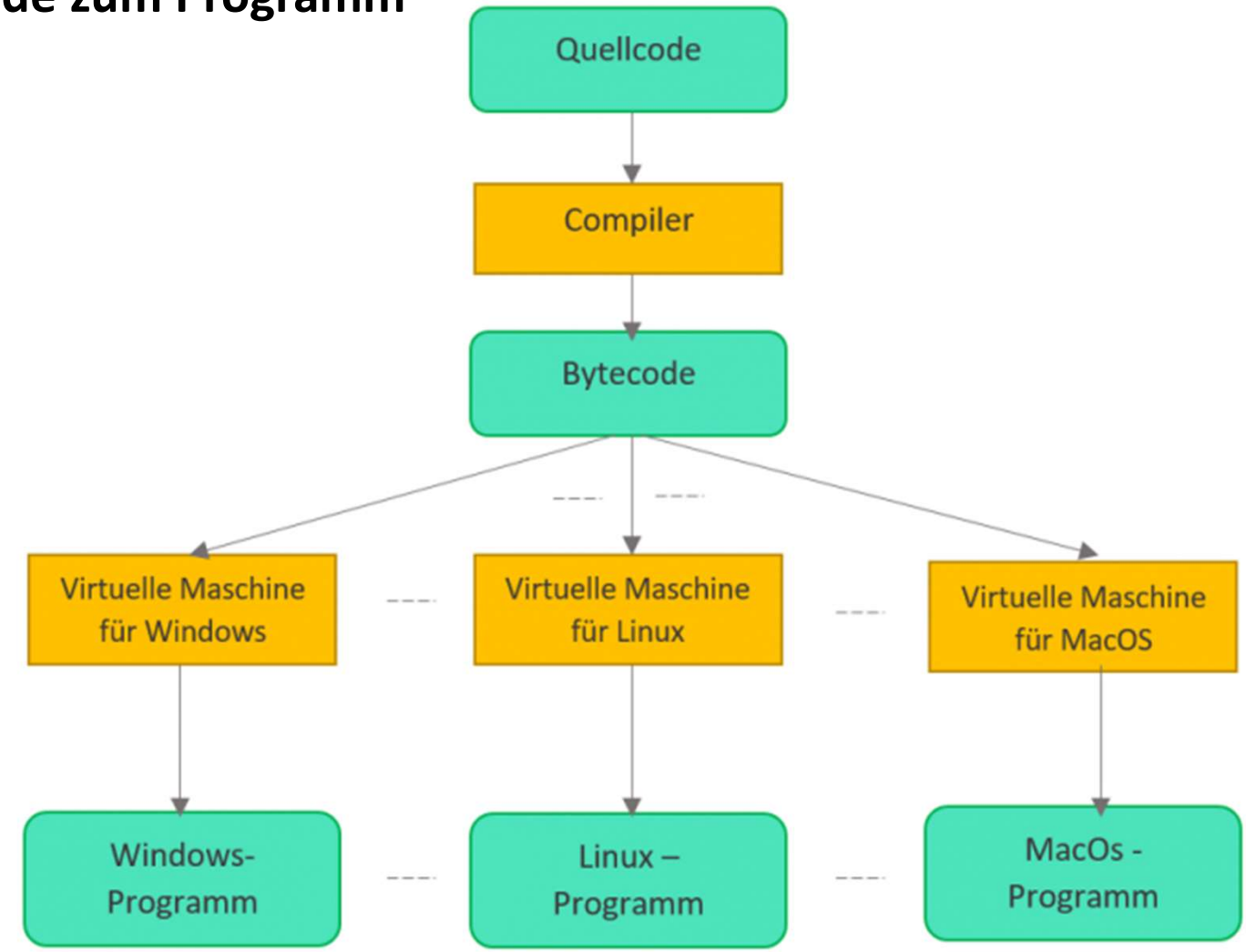
Ist Java installiert?

- Cmd
- java –versions
- javac
- „Installation“ angucken
- Ggf. installieren
- Hinweis auf Path und JAVA_HOME
- Unterschied path und classpath

Infos über Java

- Entstanden 1995, aktuelle Version 13 (gerade erschienen)
- Seit Jahren immer eine der angesagtesten Programmiersprachen in den Umfragen
- Objekt-orientiert, seit Version 8 gibt es „funktionalen Zucker“, in Java 10 wurde ein JS-mäßiges var als „untypisierter“ Typ eingeführt
- Seit Java 9 erscheinen halbjährlich neue Versionen
- OracleJDK für Business ab Java 11 kostenpflichtig (OpenJdk als Alternative)

Vom Code zum Programm



Aufgabe 1

- JDK in Version 8/11 installieren bzw. check ob es da ist (java –version, javac)

Agenda Samstag

Vorstellung und Überblick

Eclipse

Einstieg „FizzBuzz“

Unit-Testing mit Junit und Mockito

„FizzBuzz“ objektorientiert 1

Git

„FizzBuzz“ objektorientiert 2

„FizzBuzz“ objektorientiert 3

Maven

Ist Eclipse installiert?

- Ggf. installieren

Infos über Eclipse

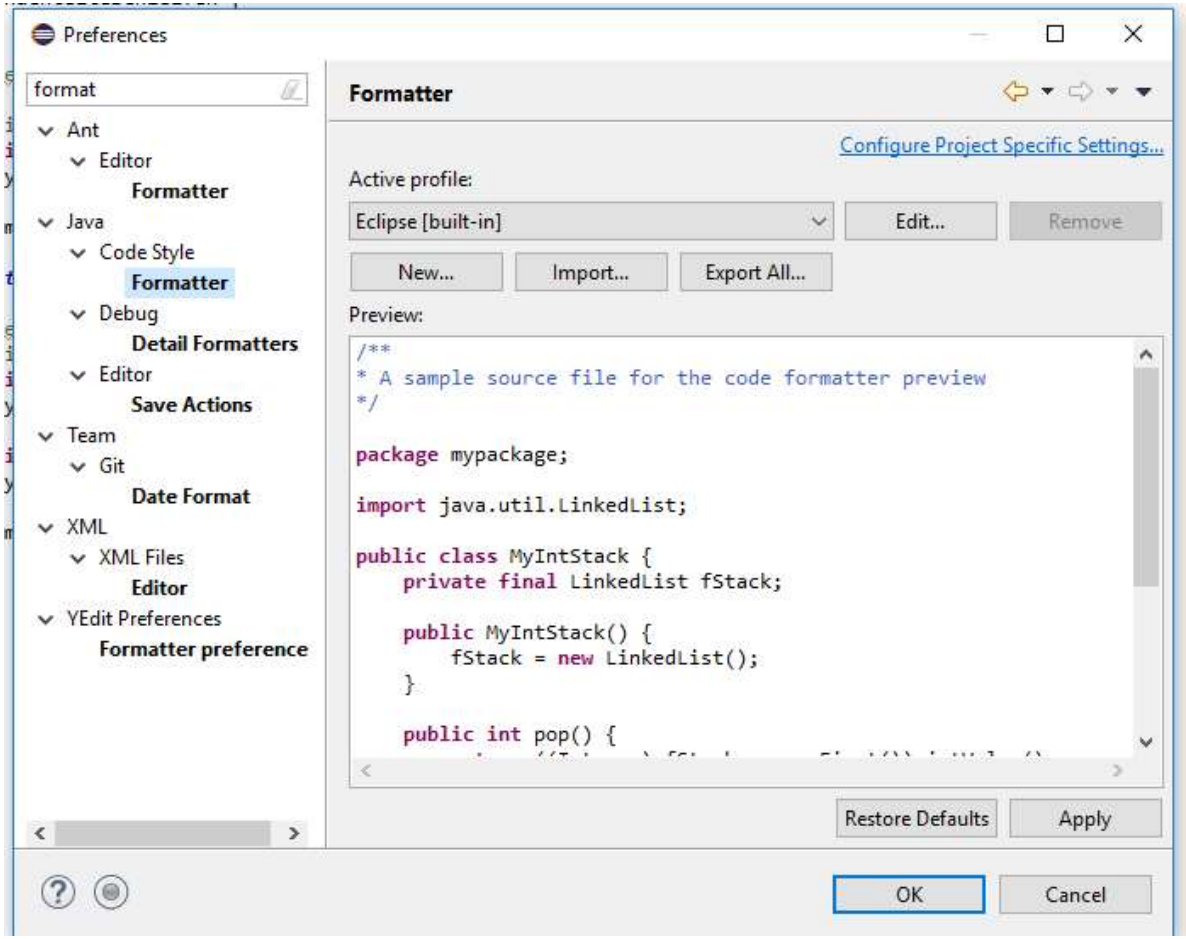
- Seit 2001 freie **I**ntegrated **D**evelopment **E**nvironment
- Für Java aber auch alle möglichen anderen Sprachen
- Wird durch Plug-Ins aufgebaut
- Bis Photon (2018) erschien jedes Jahr im Juni eine neue Major-Version, danach ~3 Service Releases
- Seit 2018-09 nun 3-monatliche „RollingReleases“

Aufgaben

Preferences:

- SaveActions einstellen
- Formatter zeigen
- Templates zeigen
- Standard shortcuts durchgehen
- Neues Java-Projekt erstellen
- Package anlegen
- Java Klasse mit Main Methode anlegen

Formatter


Preferences
format
Ant
Editor
Formatter
Java
Code Style
Formatter
Debug
Detail Formatters
Editor
Save Actions
Team
Git
Date Format
XML
XML Files
Editor
YEdit Preferences
Formatter preference
Formatter
Configure Project Specific Settings...
Active profile:
Eclipse [built-in]
Edit...
Remove
New...
Import...
Export All...
Preview:
/**
* A sample source file for the code formatter preview
*/
package mypackage;
import java.util.LinkedList;
public class MyIntStack {
private final LinkedList fStack;
public MyIntStack() {
fStack = new LinkedList();
}
public int pop() {
Restore Defaults
Apply
OK
Cancel

Agenda Samstag

Vorstellung und Überblick

Eclipse

Einstieg „FizzBuzz“

Unit-Testing mit Junit und Mockito

„FizzBuzz“ objektorientiert 1

Git

„FizzBuzz“ objektorientiert 2

„FizzBuzz“ objektorientiert 3

Maven

Pakete

- Geben Klassen einen Namensraum, haben Einfluss auf deren Sichtbarkeit
- Bieten eine Möglichkeit, Code fachlich und technisch zu modularisieren
- Einhaltung von Modulgrenzen und Beziehungsvorgaben aber aktuell nur durch externe Tools/Bibliotheken sicherstellbar
- Java 9 hat ein Modulsysten, womit sich das verbessern lässt

(lokale) Variablen

- Gültig in ihrem Scope
- Über-schatten ggf. Felder des gleichen Namens

Primitive Typen, Objekte, Wrapper & Autoboxing

- Java ist anders als z.B. Smalltalk oder Scala nicht komplett objekt-orientiert
- Man hat vmtl. aus Performancegründen die einfachen Datentypen von C übernommen: int, long, float, double, char
- Alle anderen Datentypen **sind** Objekte (später dazu mehr)
- Um die Brücke zwischen beiden Welten zu schaffen wurden in Java 5 die Wrappertypen eingeführt: Integer, Long, etc.
- Das automatische „hin- und herschalten“ zwischen beiden nennt man Autoboxing

Kontrollstrukturen

- Verzweigungen
  - If, Else
  - Switch – Case
- Schleifen
  - For, (foreach später bei Collections)
  - While, Do – while

FizzBuzz

Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”. Separate with comma and print everything in one line. Don’t add a comma at the end.

```
1, 2, Fizz, 4, Buzz, Fizz, 7, […], 98, Fizz, Buzz
```

Agenda Samstag

Vorstellung und Überblick

Eclipse

Einstieg „FizzBuzz“

Unit-Testing mit Junit und Mockito

„FizzBuzz“ objektorientiert 1

Git

„FizzBuzz“ objektorientiert 2

„FizzBuzz“ objektorientiert 3

Maven

Testen

- Testen ist notwendig. Manuelles testen ist auch notwendig, für alles aber viel zu teuer. Manuell sollte nur black-box, explorativ getestet werden
- Häufigster Test sollte der white-box unit-test sein, der vom Entwickler zur Implementierung mit geschrieben wird
- Aber Achtung: artet leicht in Code-Coveragitis aus. Konzentration auf Geschäftslogik!

TODO:

- **Eclipse Marketplace: MoreUnit**
- **Test schreiben mit Junit und Mockito**

Agenda Samstag

Vorstellung und Überblick

Eclipse

Einstieg „FizzBuzz“

Unit-Testing mit Junit und Mockito

„FizzBuzz“ objektorientiert 1

Git

„FizzBuzz“ objektorientiert 2

„FizzBuzz“ objektorientiert 3

Maven

FizzBuzz objektorientiert

Eigentlich handelt es sich um ein Kinderspiel. Mehrere Kinder sitzen zusammen (z.B. Klassenraum) und sagen der Reihe nach die nächste Zahl beziehungsweise das entsprechende Wort.

Klassen

- Sind „Vorlage“ für Objekte, die mit **new** aus einer Klasse erzeugt werden
- Enthalten Daten (fields) und die darauf ausführbarenden Funktionen (methods)
- Daten haben dann wiederum Datentypen, die weitere Klassen sind
- Methoden dienen der Interaktionen zwischen den Objekten

Felder und Methoden

- Felder halten den Zustand eines Objekts
- Felder haben einen Datentyp, einen Bezeichner und eine Sichtbarkeit
- Methoden dienen der Interaktionen zwischen Objekten
- Sie arbeiten auf den Feldern des Objekts
- Konstruktoren als spezielle Methoden vorstellen

Klassenfelder/-methoden: static

- Static fields: Gültig für alle Objekte der Klasse (globale Variable, shared state!!!)
- Static methods: Zum Zugriff auf static fields, Zugriff über Klasse

Call by reference/ call by value, immutability

- Was übergibt man bei einem Methodenaufruf
  - Einen Alias des Übergabeparameters (call by reference)
  - Eine Kopie des Übergabeparameters (call by value)?
- In Java immer eine Kopie (call by value). ABER: das ist nur bei primitiven Datentype intuitiv, da bei Objekten eine Kopie des Zugriffs auf den Objektinhalt übergeben wird (Adresse, Zeiger, …)
- Das bedeutet, dass man zwar den Übergabewert nicht ändern kann, wohl aber dessen „Inhalt“ (genauer Begriff hier: call by sharing)

FizzBuzz objektorientiert

Erste Näherung.

Agenda Samstag

Vorstellung und Überblick

Eclipse

Einstieg „FizzBuzz“

Unit-Testing mit Junit und Mockito

„FizzBuzz“ objektorientiert 1

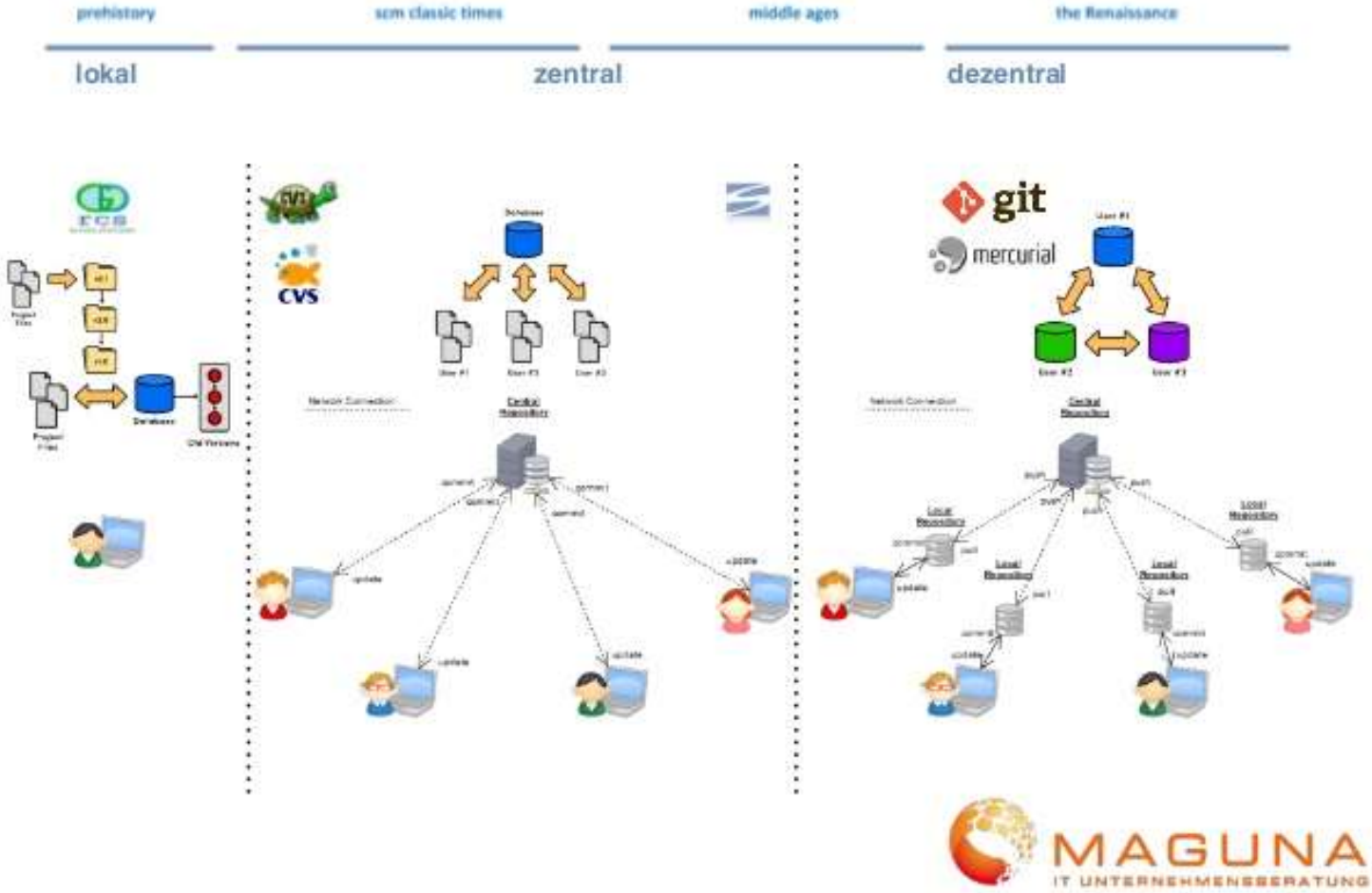
Git

„FizzBuzz“ objektorientiert 2

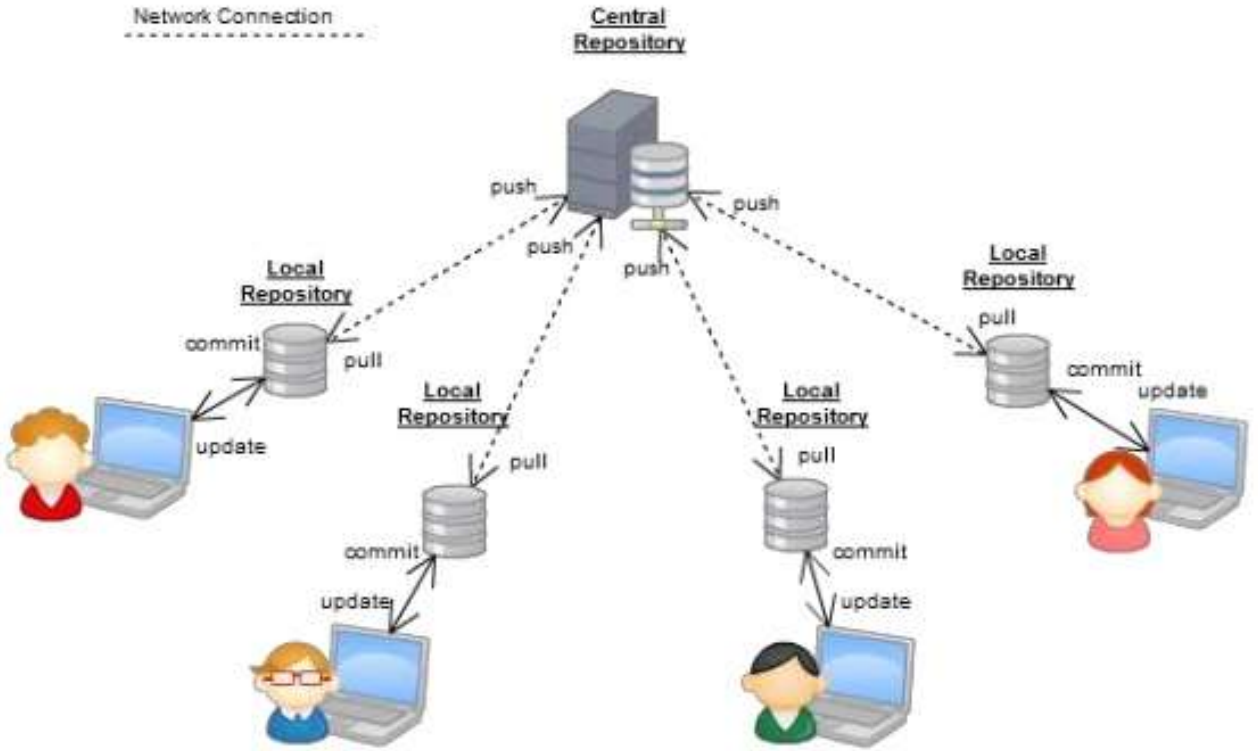
„FizzBuzz“ objektorientiert 3

Maven

Geschichte von Versionsverwaltungen



Verteilte Versionsverwaltung



Git

- Vom „Macher“ von Linux: Linus Torvalds
- Jeder hat das komplette Repo samt History lokal verfügbar
- Sehr effiziente Speicherung der Daten
- Branch und Merge integraler Bestandteil des Werkzeugs
- Wichtigste Befehle:
  - **Clone:** (Remote-)Projekt auschecken
  - **Status**: Unterschied zwischen lokal und remote feststellen
  - **Fetch**: Änderungen von remote holen ohne Verarbeitung
  - **Pull/Rebase**: Aktualisieren des lokalen repos von remote
  - **Add**: markieren von Änderungen für das commit
  - **Commit**: Einchecken ins lokale repo
  - **Push**: Hochladen ins remote repo

Agenda Samstag

Vorstellung und Überblick

Eclipse

Einstieg „FizzBuzz“

Unit-Testing mit Junit und Mockito

„FizzBuzz“ objektorientiert 1

Git

„FizzBuzz“ objektorientiert 2

„FizzBuzz“ objektorientiert 3

Maven

Arrays

- Felder von Werten gleichen Datentyps
  - Z.B. String[] args
- Ist ein Objekt
- Größe muss am Anfang festgelegt werden
  - Z.B. new String[8];
- Schneller wahlfreier Zugriff auf Einzelelemente
  - Z.B. args[0], args[1]

Collections

- List (Liste)
  - Hat eine Ordnung auf den enthaltenen Element
    - Index basierter Zugriff möglich
  - Doppelte möglich
- Set (Menge)
  - Ungeordnet
    - Kein Index
  - Keine doppelten
- Map (Abbildung, Key-Value-Paar)
  - Keys bilden ein Set → Eindeutigkeit der Schlüssel

Wir brauchen mehr Kinder

Enum

- Aufzählungstypen
  - Z.B. Wochentag (Montag, Dienstag,…,Sonntag)
- Neu in Java 5, haben Konstantenklassen/-interfaces abgelöst
- Jede Ausprägung ist ein Singleton (vergleichbar also mit ==)

Enum Word

Agenda Samstag

Vorstellung und Überblick

Eclipse

Einstieg „FizzBuzz“

Unit-Testing mit Junit und Mockito

Git

„FizzBuzz“ objektorientiert 2

„FizzBuzz“ objektorientiert 3

Maven

Interfaces

- Beschreiben einen Typ, oder auch eine Rolle bzw. eine Schnittstelle
- Erhält Methodensignaturen
- Kann ausimplementierte static bzw. default Methoden enthalten
- Kann Konstanten enthalten
- Klassen können Interfaces implementieren

Vererbung

- Typ-Vererbung: eine **IST-EIN** Beziehung (implements Interface) [subtyping]
- Implementierungs-Vererbung: eine **IST-EIN** Beziehung und erben von öffentlichen und protected Feldern und Methoden (extends Superclass) [subclassing]
- In Java ist Einfach-Subclassing und Mehrfach-Subtyping möglich

Overwriting vs. overloading und Object

- Subklassen können Methoden ihrer Superklassen überschreiben (toString())
- Alle Java-Klassen erben implizit von Objekt
- Besondere Relevanz neben toString() haben die Methoden equals() und hashcode(), die immer gemeinsam überschrieben werden müssen
- Dagegen meint overloading, dass es Methoden gleichen Namens mit unterschiedlicher Formalparameterliste gibt

Annotation @Override

- Mit Annotation werden Metadaten und Steuerungsinformationen hinterlegt (seit Java 5)
- Vermutlich eines der wichtigsten und mächtigsten Features, die jemals Java hinzugefügt worden sind
- Die @Override Annotation ist eher harmlos. Damit weißt man eben darauf hin, dass hier eine Method überschrieben werden soll
- Falls mit der Signatur irgendetwas nicht passt, kann der Compiler meckern

Polymorphie

- Meint Vielgestaltigkeit
- Nutzt Typ-Vererbung und „late binding“ aus
- Late Binding: erst zur Laufzeit wird die konkrete Implementierung einer Methode ermittelt

Delegation

- Implementierungs-Vererbung gibt es in Java nur zusammen mit Typ-Vererbung – das hat durchaus Nachteile
- Möchte man nur Code an einer Stelle bündeln, setzt man besser auf Delegation: eine **HAT-EIN** Beziehung

Smart and Dumb kid

Agenda Samstag

Vorstellung und Überblick

Eclipse

Einstieg „FizzBuzz“

Unit-Testing mit Junit und Mockito

Git

„FizzBuzz“ objektorientiert 2

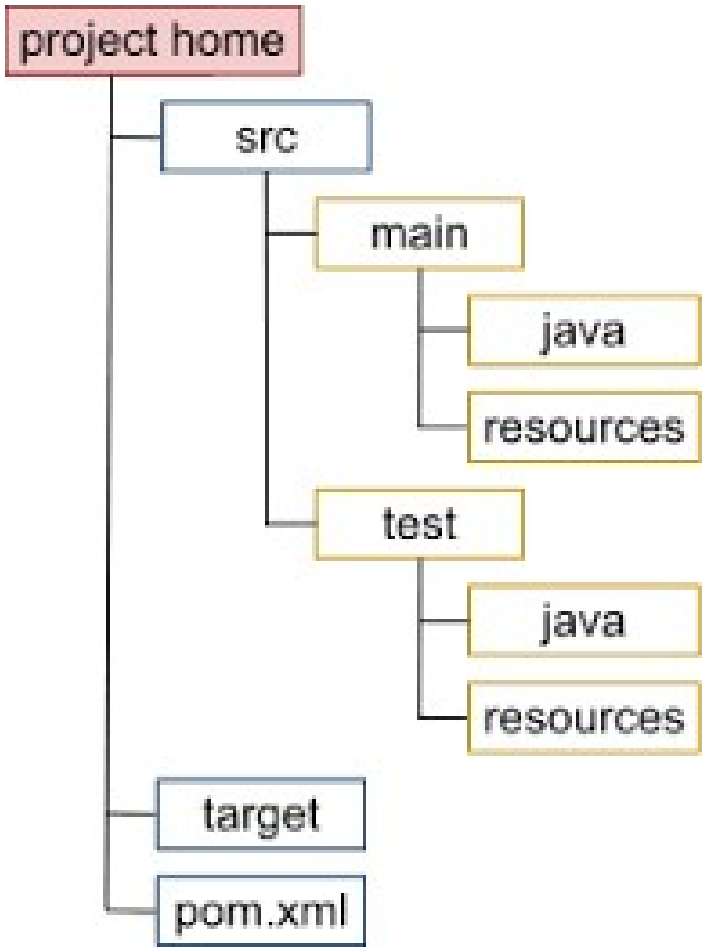
„FizzBuzz“ objektorientiert 3

Maven

Infos über Maven

- Seit 2001
- Build-System
- Dependency-Management
- Alternative: gradle

Folders


project home
src
main
java
resources
test
java
resources
target
pom.xml

Lifecycle

Clean (eigener Befehl)

Validate
Compile
Test
Package
Verify
Install
Deploy