Programmierung 2

Kapitel 1 – Einführung

Vielen Dank an Prof. Erik Buchmann
für die beigesteuerten Folien

Überblick

- Zu meiner Person
- Organisatorisches
- Ziele der Vorlesung
- Erste Schritte mit Objektorientierung
- Die Programmiersprache Java

Dr. Ingo Müller

- Studium Diplom-Informatik FSU Jena
- Promotion Swinburne University of Technology, Melbourne

- Mitbegründer Agent Factory GmbH
- Software Ingenieur Lock Box Pty. Ltd.
- Wissenschaftlicher Mitarbeiter NICTA/CSIRO
- Seit 1.3. an der HfTL

- Interessen
  - Digitale Identität
  - Datensicherheit in der Cloud
  - Sichere Softwareentwicklung

Wearable Authentication

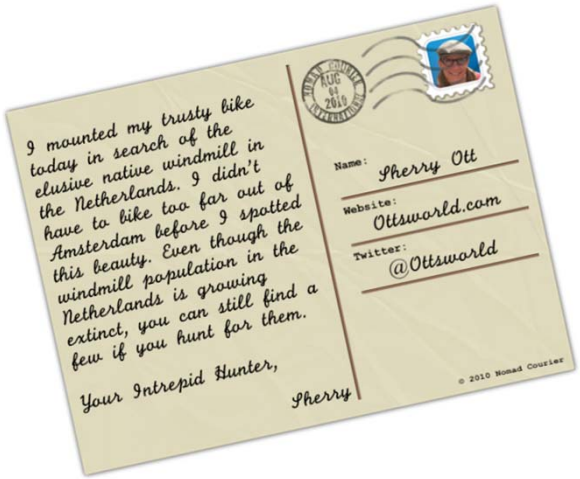
- Authentifizieren ohne Passworte…

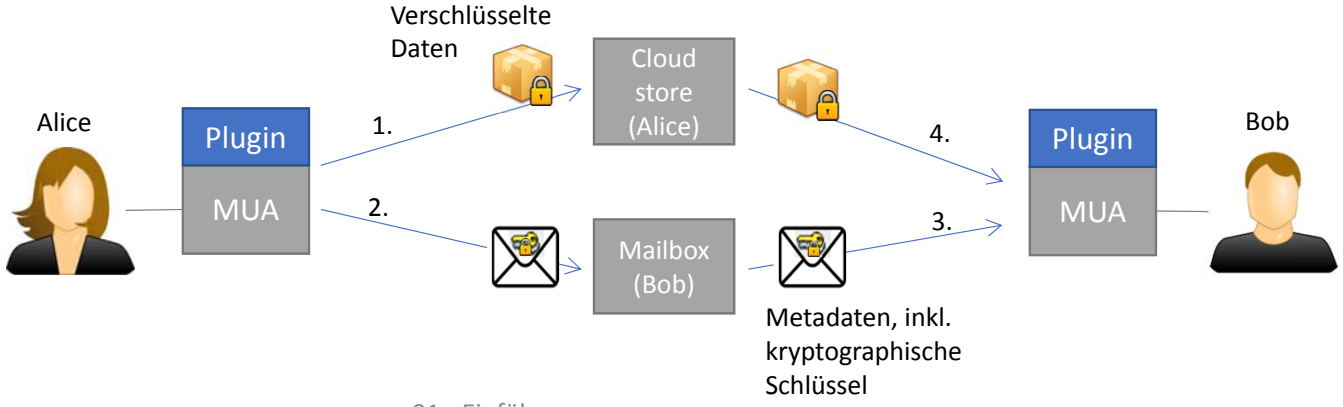
1. Nutzer authentifiziert sich an Smartwatch mit biometrischen Modalitäten (Fingerabdruck, Iris Scan des Auges)

2. Nutzer authentifiziert sich gegenüber dritten Parteien mit der Smartwatch und NFC

End-to-End Secure File Sharing via Email

- Email ist so unsicher wie ein Postkarte
- PGP und S/MIME Lösungen sind kompliziert und verschlüsseln nicht die gesamte Email
- Gibt es eine bessere Lösung die in die aktuelle Infrastruktur integriert werden kann?





Kontakt

- Sprechzeiten
  - Mittwochs, 14 bis 16 Uhr
  - Raum L1.18
- Email
  - mueller05@hft-leipzig.de
- Forum in ILIAS

Organisatorisches

Voraussetzungen für dieses Modul

- Programmierkenntnisse in einer imperativen Sprache (C, VB/VBA)
  - Grundelemente: Anweisungen, Verzweigungen, Schleifen, ...
- Kenntnisse in Algorithmen und Datenstrukturen
  - Arrays, Listen, Bäume, Stacks, Heaps, …
  - Sortieren, Finden, …
- Problemlösungsstrategien
  - Entwickeln von Algorithmen
  - Finden weiterführender Informationen (Stackoverflow, Google, Youtube)

Inhalte des Moduls

- Auffrischung imperative Programmierung
  - Variablen, Arrays, Kontrollstrukturen, Algorithmen
- Grundlagen der Objektorientierung und Java
  - Syntax, Klassen, Objekte, Methoden
- Erweiterte Objektorientierte Konzepte
  - Vererbung, Polymorphie, Schnittstellen, Ausnahmebehandlung
- Graphische Nutzerschnittstellen
  - Ereignisbehandlung, Nebenläufigkeit, Collections
- Arbeitstechniken
  - Dokumentation, Eingabevalidierung, Debugging
  - Design Patterns

Lehrmaterialien

- Folien
- Skript (Prof. Dr. Krause)
- Übungsaufgaben im Praktomat
- Literaturverweise zum Selbststudium

- Zugang zu den Lehrmaterialien im ILIAS
  - Direktstudium -> Sommersemester 2018 -> WI17 Programmierung 2
  - Sie können sich für diesen Ordner selbst anmelden

Praktomat

- Anmeldung über http://praktomat.hft-leipzig.de
  - Zugang wurde bereits eingerichtet -> Infolabs Account
- Übungsaufgaben als PDF Datei verfügbar
  - Lösungen als ZIP Archiv hochladen
- Automatische Überprüfung
  - Fehler können bis zur Deadline beliebig oft korrigiert werden
- Detailinformationen in ILIAS
  - Siehe Anleitung_Praktomat.pdf
  - Ansprechpartner: P. Schmidt, schmidt@hft-leipzig.de

Termine

- Übungsblätter werden 3 Wochen vor Abgabetermin online gestellt

KW	Veranstaltung	Abgabe
11	Vorlesung + Übung	
12	Vorlesung + Übung	
13	Vorlesung + Übung	Blatt1
14		
15	Vorlesung + Übung	Blatt2
16	Vorlesung + Übung	Blatt3
17	Vorlesung + Übung	
18	Vorlesung + Übung	Blatt4
19	Vorlesung + Übung	
20		
21	Vorlesung + Übung	Blatt5
22	Vorlesung + Übung	Blatt6
23	Vorlesung + Übung	Blatt7
24		
25	Prüfungszeitraum	
26		
27		

Prüfung

- Zulassungsvoraussetzung
  - 60% der Punkte in den Übungsaufgaben (Praktomat)
- Schriftliche Prüfung
  - 90 Minuten
  - Beidseitig handbeschriebenes A4-Blatt als Gedächtnisstütze
- Themen
  - Inhalte der Vorlesung und Übungen (Skript ist umfangreicher)

Ziel der Vorlesung

Lernziele

- Sie erlernen die Fähigkeit objektorientiert in Java zu programmieren
  - Übungsaufgaben
  - Selbststudium
    (Online-Tutorials wie https://docs.oracle.com/javase/tutorial/java/)

- Sie erlernen qualitativ hochwertige Programme zu schreiben
  - Anwendung grundlegender Arbeitstechniken

Beispiel für qualitativ schlechten Code

```
package t;
public class X
{public int min(MyClass x, MyClass y, MyClass z) {if (x.value <
y.value && x.value < z.value) {return x.value}; if (y.value <
x.value && y.value < z.value) {return y.value}; {return z.value;}}}
```

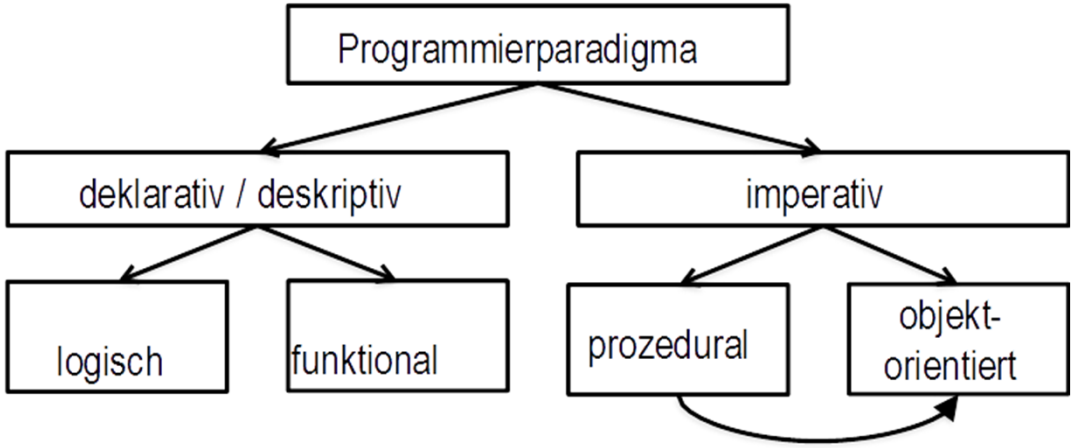
- Keine Kommentare, was der Code tut
- Keine aussagekräftigen Namen
- Unlesbare Formatierung
- Direkter Zugriff auf an anderer Stelle definierte Variablen
- Inkorrekte und „dumme“ Berechnung des Minimums
- Fehlende Eingabevalidierung und Ausnahmebehandlung

Warum Qualität wichtig ist…

- Große Projekte
  - 100.000+ Zeilen Code
  - Viele parallele Entwickler
  - Unterstützung für Zeitzonen, Fremdwährungen, Zeichensätze...
  - Parallele Entwicklungszweige mit gemeinsamer Codebase + Spezialisierungen
  - Erweiterungen und Updates
  - Integration mit anderen Anwendungen und Systemen

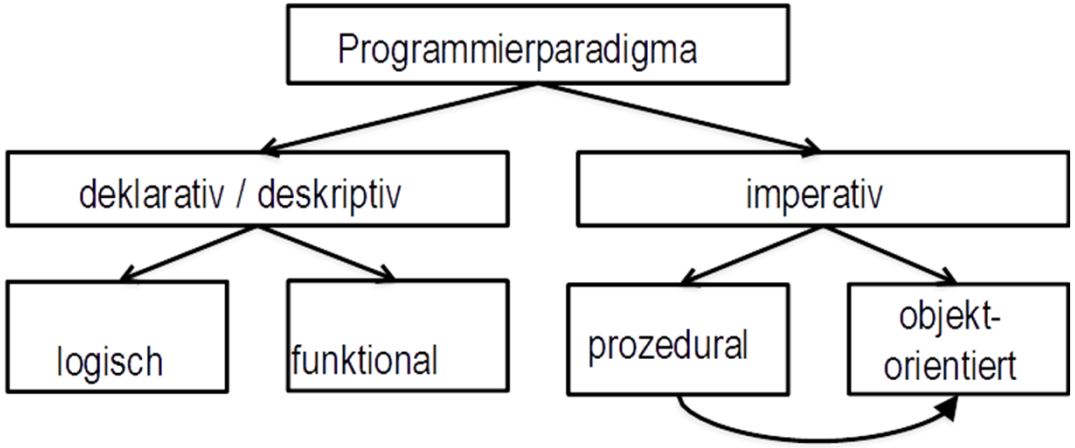
Erste Schritte

Programmierparadigmen



- Deklarativ
  - **WAS** soll mit den Eingabedaten gemacht werden?
  - formale Spezifikation
  - funktionale Sprachen wie HASKELL, logische Sprachen wie Prolog
- Imperativ
  - **WIE** soll etwas berechnet werden?
  - Abfolge von Befehlen, z.B. in C, VBA, Pascal, Java, C# ...

Programmierparadigmen



- Prozedural
  - Funktionen (Prozeduren) zur Aufteilung in Teilaufgaben
  - Programmiersprachen: C, Visual Basic, ...
- Objektorientiert
  - Einheit von Daten und Funktion (Methoden)
  - Programmiersprachen: Java, C++, … *(praktisch alle OO-Sprachen können auch prozedural)*

Prozedural vs. Objektorientiert

- Prozedurale Programmierung
  - Daten und Funktionen werden getrennt betrachtet
  - Entscheidend ist die Reihenfolge der Anweisungen bzw. Prozeduraufrufe

- Objektorientierte Programmierung
  - Klassen kapseln Daten und Funktionen als Einheit
  - Entscheidend ist die Kommunikation der Objekte untereinander

Was ist ein Objekt?

- Ein in sich abgeschlossener „Behälter“ für
  - Zustand (→ Variablen)
  - Verhalten (→ Methoden)
- Beschreibt Gegenstände, Dokumente, Konzepte, etc.
  - Auto, Fahrrad, Fisch
  - Email-Vorgang mit dem Prüfungsamt
  - Eine Sammlung von MP3-Dateien
- Bsp. Email:
  - Zustand: Sender, Empfänger, Inhalt, Zeichensatz, Sprache...
  - Methoden: antworten(), weiterleiten(), löschen(), drucken()...

Objekt-Orientierung

- **Definition Objektorientierung**: Ansatz der Software-Entwicklung der Daten und dazugehörige Funktionen in Klassen von (Realwelt-)Objekten kapselt
- Anwendungsfelder
  - Objekt-orientierter Entwurf, z.B. Datenmodellierung mit UML
  - Objekt-orientierte Datenbanken
  - Objekt-orientierte Programmierung
- Grundkonzepte
  - Objekte (Instanzen) und Klassen
  - Kapselung
  - Vererbung
  - Wiederverwendung

Die Programmiersprache Java

Geschichte von Java

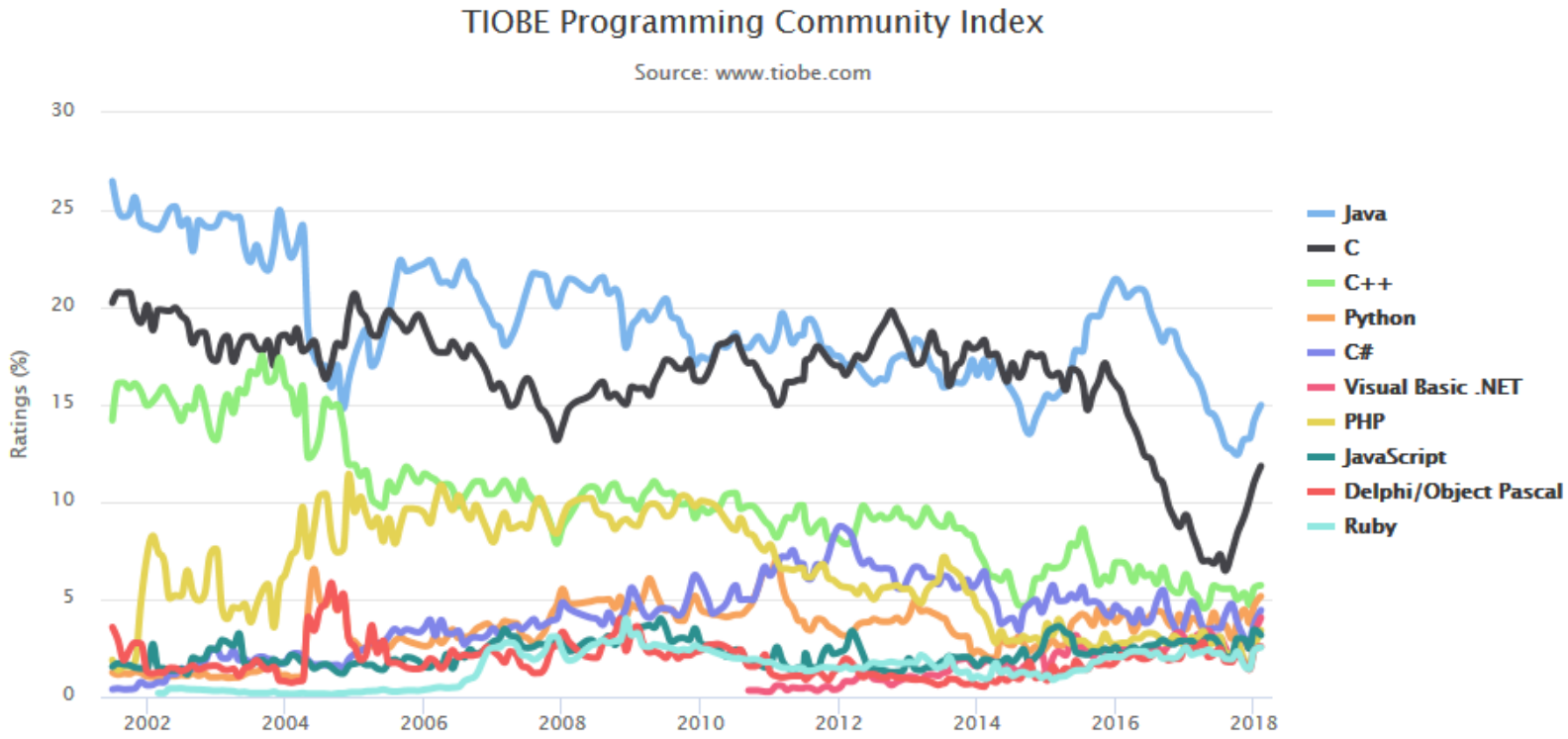


- 1991
  - J. Gosling, P. Naughton, M. Sheridan (Green Team)
  - Entwicklung von Java als Alternative zu C/C++
  - Eingebettete Systeme, Set-top Boxes, PDAs
- 1994
  - Ausrichtung auf das WWW und verteilte Anwendungen
- 1996
  - Erste stabile Java Version (JDK 1.0) veröffentlicht
- 2017
  - Java9 (JDK 9) veröffentlicht

Eigenschaften von Java

- Einfach
  - Reduzierter Sprachumfang im Vergleich zu C, C++
- Objektorientiert
- Robust
  - Starke Typisierung, Ausnahmebehandlung
  - Automatische Speicherverwaltung, Verzicht auf Pointer
- Plattformunabhängig und portierbar („write once run anywhere“)
  - Java Programme laufen auf verschiedenen Hardwareplattformen
- Weitere
  - Parallelisierbar (Multithreading)
  - Integrierte Netzwerkprogrammierung (z.B. RMI oder Webservices)
  - Vielzahl an Zusatzbibliotheken, z.B. Datenbankzugriff, GUI

Praxisrelevanz



Anwendungsgebiete…

ATMs
- Parking Meters
- POS Systems
- Lottery/Gaming Systems
- Multi Function Printers
- Intelligent Power Module
- Netbooks

Routers & Switches
- Storage Appliances
- Network Management Systems
- Medical Imaging Systems
- Radar Systems
- Industrial PCs
- Factory Automation Systems
- Geo-Imaging Devices

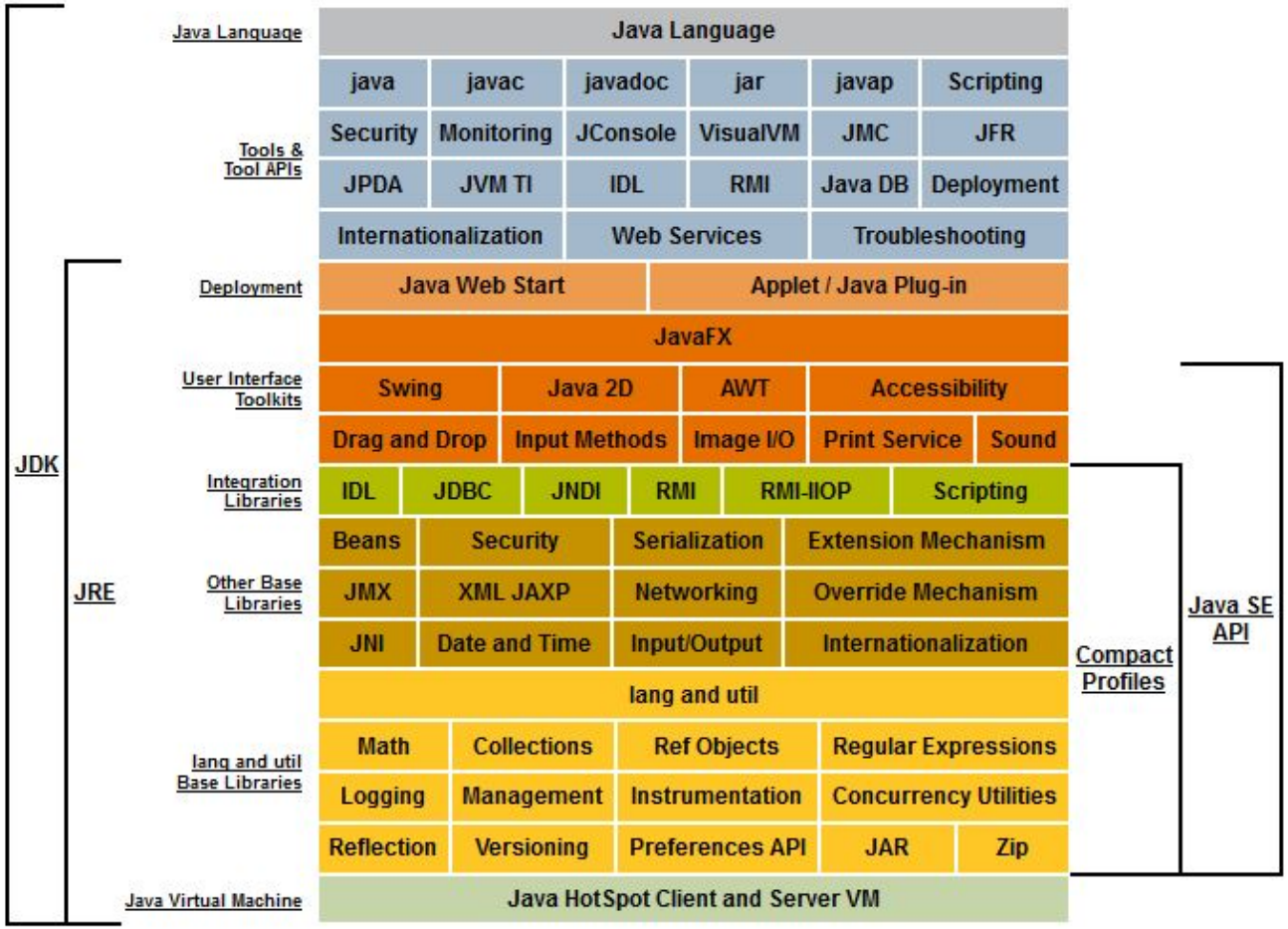
Smart Meters
- RFID Readers
- Video Conferencing Systems
- In-Flight Entertainment Systems
- Video Streaming Systems
- Electronic Voting Systems
- Voice Messaging Systems
- Security Systems

…Anwendungsgebiete

- Desktop
  - Matlab
  - Entwicklungswerkzeuge, z.B. Eclipse
  - File Sharing Software, z.B. LimeWire
- Mobile
  - Android (Laufzeitumgebung, Anwendungsentwicklung)
- Server
  - Web Servers und Integration mit Backend Systemen

- Populär durch Open Source Lizenzen
  - Spring Framework, Tomcat Web Server, Hibernate, Glassfish

Java SE 9


Java Language
Java Language
java
javac
javadoc
jar
javap
Scripting
Tools & Tool APIs
Security
Monitoring
JConsole
VisualVM
JMC
JFR
JPDA
JVM TI
IDL
RMI
Java DB
Deployment
Internationalization
Web Services
Troubleshooting
Deployment
Java Web Start
Applet / Java Plug-in
JavaFX
User Interface Toolkits
Swing
Java 2D
AWT
Accessibility
Drag and Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC
JNDI
RMI
RMI-IIOP
Scripting
Other Base Libraries
Beans
Security
Serialization
Extension Mechanism
JMX
XML JAXP
Networking
Override Mechanism
JNI
Date and Time
Input/Output
Internationalization
lang and util
lang and util Base Libraries
Math
Collections
Ref Objects
Regular Expressions
Logging
Management
Instrumentation
Concurrency Utilities
Reflection
Versioning
Preferences API
JAR
Zip
Java Virtual Machine
Java HotSpot Client and Server VM
JDK
JRE
Compact Profiles
Java SE API

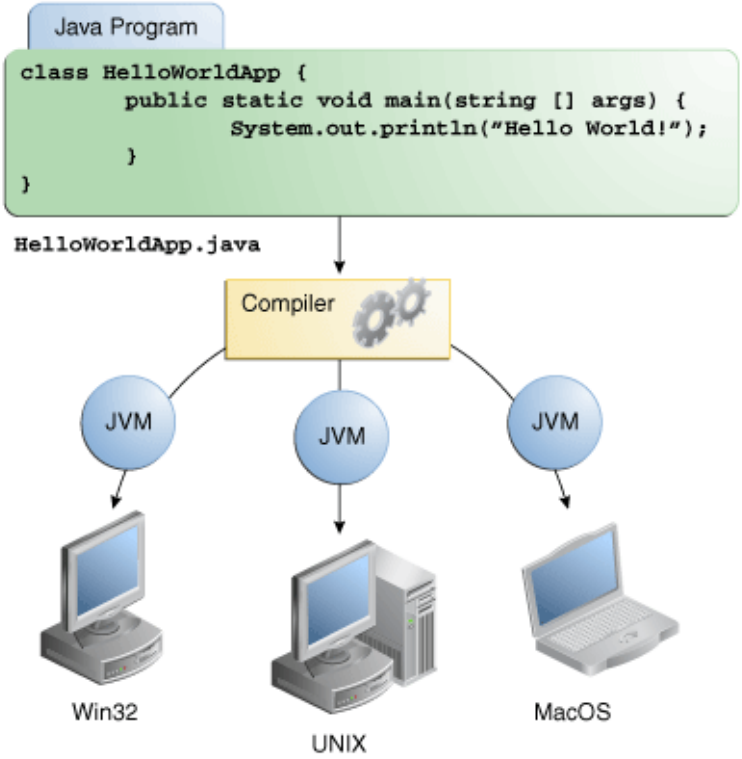
Java Laufzeitumgebungen (JRE)

- Ausführung von Java-Anwendungen
  - Java Virtual Machine (JVM) + Programmierschnittstellen (APIs)
- Varianten
  - **Java SE** (Standard Edition)
    - APIs für Entwicklung von Desktop-Anwendungen
    - Grundlage für Java EE und ME
  - Java EE (Enterprise Edition)
    - Java SE + APIs für mehrschichtige Unternehmens- und Web-Anwendungen
  - Java ME (Micro Edition)
    - für Embedded Products (u.a. Smartphone, Waschmaschinen, ...)

Vom Quelltext zur Ausführung

- **Kompilieren**: Quelltext wird in (maschinenunabhängigen) Bytecode übersetzt

- **Starten**: Laufzeitumgebung JVM übersetzt und optimiert Bytecode für Plattform
  - Etwas längere Startzeit
  - Plattformunabhängige Entwicklung – einmal übersetzen, überall starten wo eine VM zur Verfügung steht
  - Extraaufwand für Updates (z.B. Sicherheitsupdates)



Entwicklung von Java Programmen

- JRE ist Laufzeitumgebung für kompilierte Java-Anwendungen
- Java-Programmierung benötigt Java Development Kit (**JDK**)
  - JDK = JRE + Programmiertools, u.a.
    - javac: Java Compiler,
    - jar: Java Archive Files
- integrierte Entwicklungsumgebung (IDE) sehr hilfreich
  - u.a. **Eclipse**, IntelliJ, Netbeans, ...
  - vollständige Integration des Entwicklungsprozesses
    - Programm editieren, kompilieren, ausführen, debuggen etc.
  - Intelligenter Editor, u.a. mit Auto-Vervollständigung und Fehlermeldungen

Manuelle Programmerstellung

1. Erstellen des Programm-Quellcodes (Editor)
2. Abspeichern in einer Datei mit Namen der Klasse und Endung ".java„

```
public class MyClass {
    public static void main(String[] args) {
        System.out.println("Hallo Welt!");
    }
}
```

3. Compilieren des Quellcodes (Compiler **javac**)
   - Beispiel: javac MyClass.java
   - Solange der Compiler Fehlermeldungen liefert wiederhole mit korrigiertem Quellcode
4. Compiler erzeugt Java-Bytecode
   - Klassenname mit File-Extension "class"; Beispiel: MyClass.class
5. Ausführung des Programms (d.h. Bytecodes) mittels Interpreter **java**
   - Beispiel: java MyClass

Grundgerüst

```
package hftl;

/**
 * Diese Klasse ist ein Hallo-Welt
 * @author buchmann
 */
public class HelloWorld {
    /**
     * Dies ist die Startmethode eines Java-Programms
     * @param args Array mit Argumenten von der Kommandozeile
     */
    public static void main(String[] args) {
        // Gebe "Hallo Welt!" auf der Konsole aus
        System.out.println("Hallo Welt!");

        // Programmende
    }
}
```

Package
(für heute: Unterverzeichnis)

Klassenname
(für heute: Dateiname ohne Endung)

Übergabeparameter

Imperatives Programm

Einstiegspunkt in das Programm

Visual Basic

- Variablen

```
Dim i, j, k As Integer
' Integer ... elem. Datentyp
```

- Felder

```
Dim zahlen(6) As Integer
' 7 Elem. (Index: 0..6)

i = zahlen(2)
' Zugriff auf 3.El. (Index 2)

Dim zahlen(2 To 6) As Integer
' 5 Elem. (Index: 2..6)
```

Java

```
int i, j, k;
// int ... elem. Datentyp
// Integer ... abstr. Datentyp (Klasse)
```

```
int zahlen[] = new int[7];
// 7 Elem. (Index: 0..6)

i = zahlen[2];
// Zugriff auf 3.El. (Index 2)

// geht in Java nicht!
```

Visual Basic

- Prozedur (kein Rückgabewert!)

```
Sub meldeMsg(ByVal msg As String)
    MsgBox(msg)
End Sub
```

- Funktion (mit Rückgabewert)

```
Function sqr(ByVal z As Double) As Double
    Return z*z
End Function
```

```
Function sqr(ByVal z As Double) As Double
    sqr = z*z
    ' damit wird der Rueckgabewert gesetzt,
    ' vor dem Verlassen kann weiterer Code folgen
End Function
```

Java

```
void meldeMsg(String msg) {
  System.out.println(msg);
}
```

```
double sqr(double sqr){
  return z*z;
}
```

Visual Basic

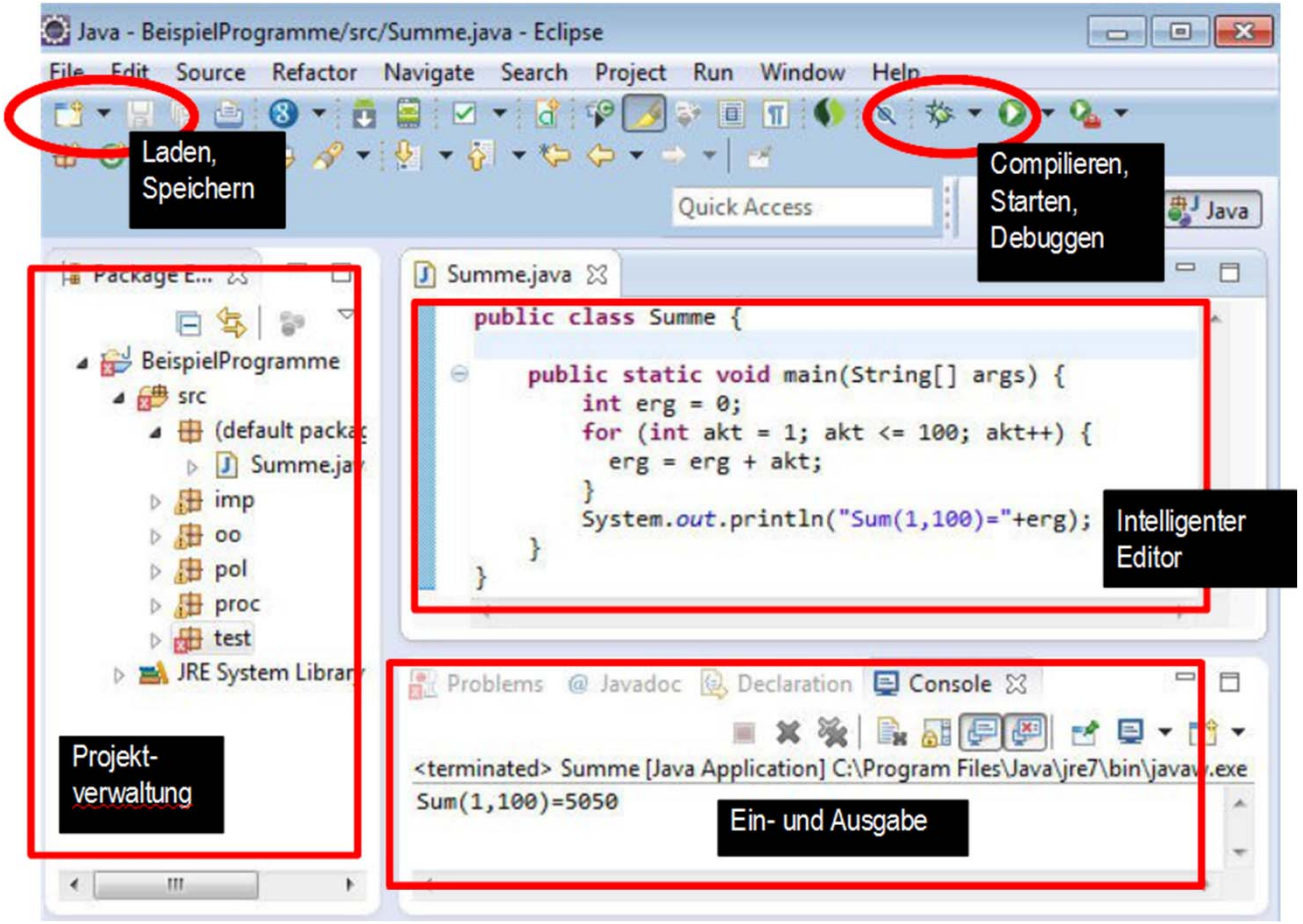
- Einfaches Programm

```
Module Program
  Sub Main(args As String())
    Console.WriteLine("Hello World!")
  End Sub
End Module
```

Java

```
public class Program {
  public static void main( String[] args) {
    System.out.println("Hello World!");
  }
}
```

Werkzeugunterstützung mit Eclipse



Kontrollfragen

- Was ist der Unterschied zwischen gutem und schlechtem Code?
- Was ist eine imperative Programmiersprache?
- Was ist Objektorientierung?
- Erklären Sie, weshalb eine starke Typisierung und ein Verzicht auf Pointer bei der Programmierung hilfreich sind.
- Wie ist der Ablauf vom Quelltext bis zur Ausführung für ein Java-Programm?