30 Jahre Java

Ein Blick zurück und in die Zukunft

Michael Wiedeking · Java Forum Stuttgart · 10. Juli 2025

1968

The Art of Computer Programming

- 1968 Volume 1 – Fundamental algorithms
- 1969 Volume 2 – Seminumerical algorithms
- 1973 Volume 3– Sorting and searching
- 2011 Volume 4A – Combinatorial algorithms
- 2022 Volume 4B – Combinatorial algorithms
- XXXX Volume 4C, 4D, ... – Combinatorial algorithms
- XXXX Volume 5 – Syntactic algorithms
- XXXX Volume 6 – The Theory of context-free languages
- XXXX Volume 7 – Compiler techniques

1971

Pascal

- 1971 Pascal wird veröffentlicht
- 1973 Pascal-P System (p-Code)
- 1975 Pascal-S
- 1977 UCSD p-System
- 1980 Business Operating System (BOS)

- 1957 Fortran — **For**mula **Tran**slation
- 1958 Algol — **Algo**rithmic **L**anguage
- 1959 Cobol — **Co**mmon **B**usiness **O**riented **L**anguage
- 1965 Simula — **Simu**lation **La**nguage

- 1957 Fortran — **For**mula **Tran**slation
- 1958 Algol — **Algo**rithmic **L**anguage
- 1959 Cobol — **Co**mmon **B**usiness **O**riented **L**anguage
- 1965 Simula — **Simu**lation **La**nguage

The major cause of the software crisis is] that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

Edsger Dijkstra: The Humble Programmer, 1972

1973

C

- 1969 Arbeit an C beginnt
- 1973 C „fertig“ (für PDP-11)
- 1978 The C Programming Language
- 1989 C standardisiert

Prozessor	Registerbreite	Jahr
DEC PDP-8	12 Bit	1965
DEC PDP-11	16 / 18 Bit	1970
Intel 4004	4 Bit	1971
CDC 6600	60 Bit	1964
GE-600 Serie	36 Bit	1965
Digital VAX	32 / 36 / 64 Bit	1977
Symbolics 3600	36 Bit	1983
MIT Whirlwind	16 / 20 Bit	1951

1977

TeX

- 1977 The Art of Computer Programming, 2. Auflage
- 1978 Knuths Sabbatical
- 1978 Erste TeX-Version für PDP-10

  Device Independent File Format (DVI)
- 1987 Literate Programming mit WEB
- 1989 TeX „fertig“

80er

80er

- Token Ring
- Ethernet
- The Network is the Computer (Sun)

80er

- 1983 Simple Mail Transfer Protocol (SMTP)
- 1984 X Window System (X11)
- 1985 C++
- 1985 IEEE Standard for Floating-Point Arithmetic (IEEE 754)
- 1986 Standard Generalized Markup Language (SGML)
- 1989 Intel 80486
- 1989 World Wide Web

90er

- 1991 The Green Project (Oak)
- 1993 Common Gateway Interface (CGI)
- 1993 NCSA Mosaic
- 1994 Apache HTTP Server
- 1994 Netscape
- 1994 Design Patterns: Elements of Reusable Object-Oriented Software

23. Mai 1995

Designziele

- einfach, objektorientiert, verteilt und vertraut
- robust und sicher
- architekturneutral und portabel
- leistungsfähig
- interpretierbar, parallelisierbar und dynamisch

Zahlen

- byte 8 Bit
- short 16 Bit
- int 32 Bit
- long 64 Bit
- vorzeichenbehaftet
- 2er-Komplement

- float 32 Bit
- double 64 Bit
- Angelehnt an IEEE 754

Shift

assert 0x8000_0000 >>> 31 == 0x0000_0001

assert 0x8000_0000 >> 31 == 0xFFFF_FFFF

assert 0x8000_0000 >> 32 == ???

Shift

assert 0x8000_0000 >>> 31 == 0x0000_0001

assert 0x8000_0000 >> 31 == 0xFFFF_FFFF

assert 0x8000_0000 >> 32 == 0x8000_0000 >> (32 % 32) == 0x8000_0000

assert 0x8000_0000 >> 33 == 0x8000_0000 >> (33 % 32) == 0x4000_0000

write once,
run everywhere

write once,
debug everywhere

```
/**
 * Dies ist ein Kommentar, der das nächste Element beschreibt.
 */
class Example {


}
```

1995

Java 1.0/1.0.2

- java.lang
- java.applet
- java.awt
- java.io
- java.net
- java.util

```c
#define PORT 8080

int main() {

    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};

    const char *hello = "Hello from server";

    server_fd = socket(AF_INET, SOCK_STREAM, 0);

    if (server_fd == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;  // 0.0.0.0
    address.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    if (listen(server_fd, 3) < 0) {
        perror("listen");
        exit(EXIT_FAILURE);
    }

    new_socket = accept(
        server_fd,
        (struct sockaddr *) &address,
        (socklen_t *) &addrlen
    );

    if (new_socket < 0) {
        perror("accept");
        exit(EXIT_FAILURE);
    }

    read(new_socket, buffer, sizeof(buffer));
    printf("Client sagt: %s\n", buffer);
    send(new_socket, hello, strlen(hello), 0);
    printf("Hallo-Nachricht gesendet\n");

    close(new_socket);
    close(server_fd);

    return 0;
}
```

```
public class Server {

    static final int port = 8080;

    public static void main(String[] args) {

        try (ServerSocket serverSocket = new ServerSocket(port)) {

            Socket clientSocket = serverSocket.accept();
            BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
            String message = in.readLine();
            System.out.println("Client sagt: " + message);

            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
            out.println("Hello from server");
            System.out.println("Hallo-Nachricht gesendet");
            clientSocket.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```c
#define PORT 8080

int main() {

    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};

    const char *hello = "Hello from server";

    server_fd = socket(AF_INET, SOCK_STREAM, 0);

    if (server_fd == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;  // 0.0.0.0
    address.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    if (listen(server_fd, 3) < 0) {
        perror("listen");
        exit(EXIT_FAILURE);
    }

    new_socket = accept(
        server_fd,
        (struct sockaddr *) &address,
        (socklen_t *) &addrlen
    );
    …
```

```java
static final int port = 8080;

public static void main(String[] args) {

    try (var serverSocket = new ServerSocket(port)) {

        new_socket = serverSocket.accept()

        …

    }
```

```
File file = new File("./test.txt")

InputStream fin = new FileInputStream(file);

Reader in = new InputStreamReader(in, StandardCharsets.UTF_8);

in = new BufferedReader(in, 8192);
```

```
File file = new File("./test.txt")

InputStream fin = new FileInputStream(file);

Reader in = new InputStreamReader(in, StandardCharsets.UTF_8);

in = new BufferedReader(in, 8192);


Reader in = Files.newBufferedReader(file.toPath(), StandardCharsets.UTF_8)
```

1997

Java 1.1

- java.lang
- java.applet
- java.awt
- java.io
- java.net
- java.util

- package java.awt.datatransfer
- package java.awt.event
- package java.awt.image
- package java.beans
- package java.rmi
- package java.rmi.dgc
- package java.rmi.registry
- package java.rmi.server

- package java.security
- package java.security.acl
- package java.security.interfaces
- package java.sql
- package java.text
- package java.util
- package java.util.zip
- package java.lang.reflect
- package java.math

```
class Point {

    int x;
    int y;

    int getX() {
        return x;
    }

    int setX(int x) {
        this.x = x;
    }

    int getY {
        return y;
    }

}
```

```
class Point {

    …

    int getX() {
        return …;
    }

    int setX(int x) {
        …;
    }

    int getY {
        return …;
    }

}
```

```
class Point {

    …

    int getX() {
        return …;
    }

    int setX(int x) {
        …;
    }

    int getY {
        return …;
    }

}
```

Point

int x; // read, write

int y; // read

```
class Renderer {
    byte[ ][ ] image;
    void render(w : Windower) {
      for (int i = 0; i < 1024; i++) {
          for (int j = 0; j < 1024; j++) {
              image[i][j] = w.window(image[i][j]);
          }
       }
}
```

```
class Renderer {
    byte[ ][ ] image;
    void render(w : Windower) {
        image = this.image;
        for (int i = 0; i < 1024; i++) {
            for (int j = 0; j < 1024; j++) {
                image[i][j] = w.window(image[i][j]);
            }
        }
}
```

```
class Renderer {
    byte[ ] image;
    void render(w : Windower) {
        image = this.image;
        int N = 1024 * 1024;
        for (int i = 0; i < N; i++) {
            image[i] = w.window(image[i]);
        }
}
```

```
class Renderer {
    byte[ ] image;
    void render(w : Windower) {
        image = this.image;
       for (int i = 0; true; i++) {
              image[i] = w.window(image[i]);
       }
}
```

```
class Renderer {
    byte[ ] image;
    void render(w : Windower) {
        image = this.image;
        int N = 1024 * 1024;
        try for (int i = 0; true; i++) {
                image[i] = w.window(image[i]);
       } catch (ArrayIndexOutOufBoundsException e) {}
}
```

```
class Renderer {
    byte[ ] image;
    void render(w : Windower) {
        image = this.image;
       try {
            for (int i = 0; true; i++) {
               image[i] = w.window(image[i]);
            }
       } catch (ArrayIndexOutOufBoundsException e) {}
}
```

- 1996 JavaOS

- 1997 Java WorkShop
- 1997 VisualAge for Java
- 1997 JBuilder

- 1999 picoJava

1998

- java.security
- java.security.acl
- java.security.cert
- java.security.interfaces
- java.security.spec
- javax.swing
- javax.swing.border
- javax.swing.colorchooser
- javax.swing.event
- javax.swing.filechooser
- javax.swing.plaf
- javax.swing.plaf.basic
- javax.swing.plaf.metal
- javax.swing.plaf.multi
- javax.swing.table
- javax.swing.text
- javax.swing.text.html
- javax.swing.text.html.parser
- javax.swing.text.rtf
- javax.swing.tree
- javax.swing.undo
- org.omg.CORBA
- org.omg.CORBA.DynAnyPackage
- org.omg.CORBA.ORBPackage
- org.omg.CORBA.portable
- org.omg.CORBA.TypeCodePackage
- org.omg.CosNaming
- org.omg.CosNaming.NamingContextPackage

1999

Dieser Kurs richtet sich an Manager und Entscheider, die vielleicht Java in Internet- oder Intranet-Applikationen einsetzen wollen. Es werden dabei nicht nur die Einsatzmöglichkeiten von Java besprochen, sondern auch an konkreten Beispielen aus Forschung und Industrie gezeigt, wie mit Java effizienter gearbeitet werden kann…

- Was ist Java?
- Wie kann Java eingesetzt werden?
- Wo wird Java bereits eingesetzt?
- Kostenreduzierung durch Einsatz von Java
- Java und Datenbanken
- Java und Netzwerkterminals
- Vorteile von Java gegenüber CGI-Programmierung
- Wie sicher ist Java?
- Softwareentwicklung mit Java
- Wie schnell kann Java erlernt werden?
- Zukunftsaussichten und Entwicklungsprognosen

HotSpot Just-In-Time-Compiler

- Inlining
- Loop-Unrolling
- Dead Code Elimination
- Peephole-Optimierung
- …

2000

Jahr	Version	Pakete	Klassen, Enums, Interfaces
1995	JDK 1.0	8	212
1997	JDK 1.1	23	504
1998	JDK 1.2	59	1520
2000	JDK 1.3	76	1842
2002	JDK 1.4	135	2991
2004	J2SE 5.0	166	3279
2006	JSE 6.0	203	3793
2011	JSE 7.0	209	4024
2014	JSE 8.0	217	4240
2017	JSE 9.0	315	6005
2018	JSE 10.0	314	6002
2018	JSE 11.0	235	4410

2004

Java 5

- Enums
- Generics

2005

Graal VM

- JVM in Java
- Truffle Bibliothek

```
try (Context context = Context.create()) {
    Value function = context.eval("python", "lambda x: x + 1");
    int x = function.execute(41).asInt();
    assert x == 42;
}
```

2011

Java 7

- Invoke Dynamic

2014

Java 8

- Lambda-Ausdrücke

2017

Java 9

- Module

20xx

List<int>

Vielen Dank!