Fachhochschule Giessen-Friedberg

Blockseminar 11.-14.Sept.‘01

Java Virtual Machine

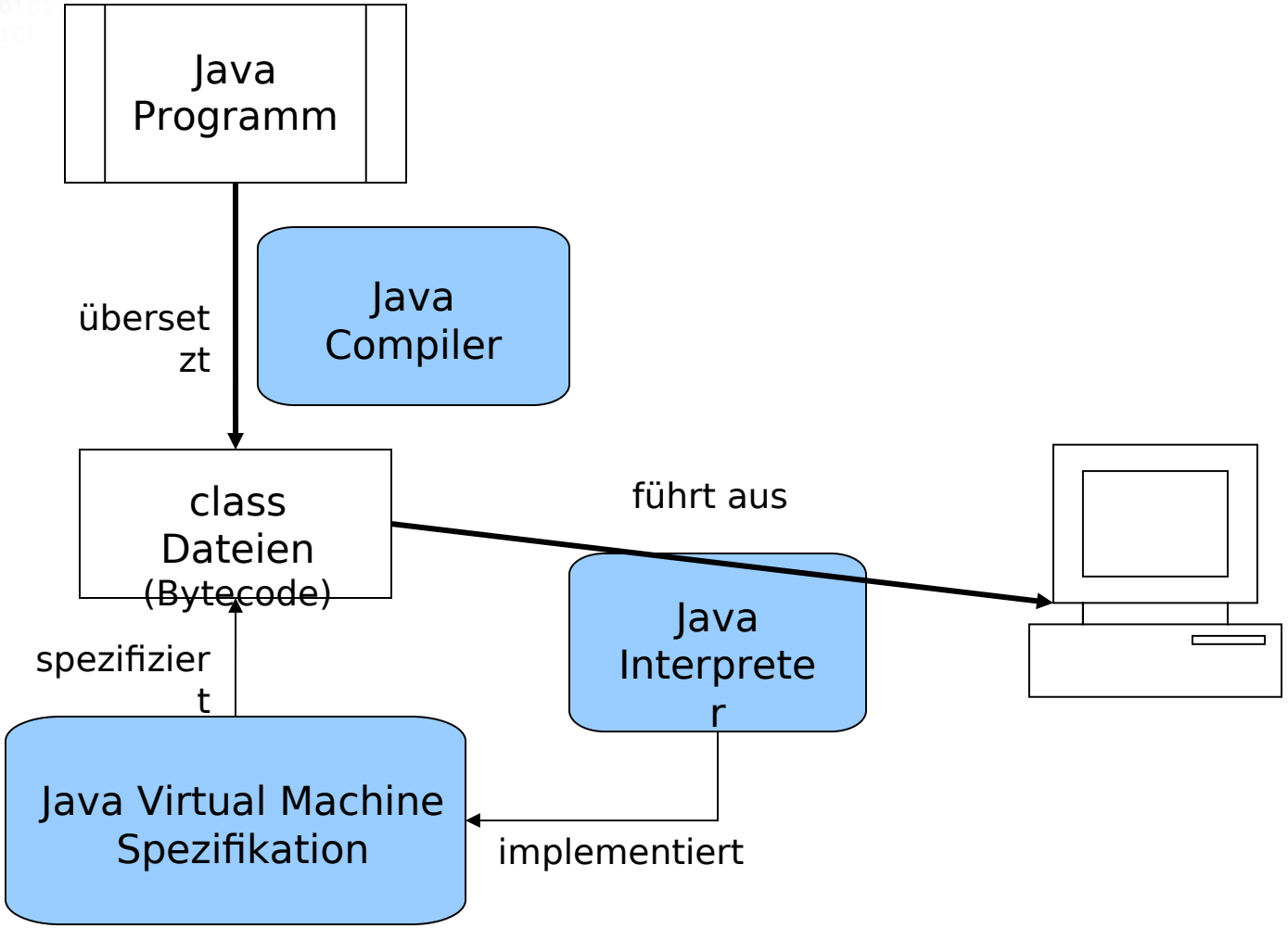
vorgetragen
von

Peter Strzala

Übersicht

- Java ein Technologiekonzept
- Das `class`-Dateiformat
  - Konstantenpool, Felder, Methoden, Attribute, Signaturen
  - Dateibeispiel
- Die Maschinenspezifikation
  - Datentypen
  - Register und lokale Variablen
  - Laufzeitumgebung
  - Stack und Heap
  - Befehlssatz
  - Programm Beispiel

Java ein Technologiekonzept

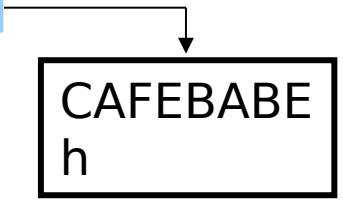


Das `class`-Dateiformat

```
Klassendatei
{
        int                     Erkennungszahl;
        short                   Subversionsnummer;
        short                   Versionsnummer;
        short                   Anzahl_Konstanten;
        Konstantenpooleintrag   Konstantenpool[Anzahl_Konstanten –
1];
        short                   Merkmale;
        short                   Klasse;
        short                   Superklasse;
        short                   Anzahl_Interfaces;
        short                   Interfaces[Anzahl_Interaces];
        short                   Anzahl_Felder;
        Feldeintrag             Felder[Anzahl_Felder];
        short                   Anzahl_Methoden;
        Methodeneintrag         Methoden[Anzahl_Methoden];
        short                   Anzahl_Attribute;
        Attributeintrag         Attribute[Anzahl_Attribute];
}
```

Das `class`-Dateiformat

```
Klassendatei
{
        int                     Erkennungszahl;
        short                   Subversionsnummer;
        short                   Versionsnummer;
        short                   Anzahl_Konstanten;
        Konstantenpooleintrag   Konstantenpool[Anzahl_Konstanten –
1];
        short                   Merkmale;
        short                   Klasse;
        short                   Superklasse;
        short                   Anzahl_Interfaces;
        short                   Interfaces[Anzahl_Interaces];
        short                   Anzahl_Felder;
        Feldeintrag             Felder[Anzahl_Felder];
        short                   Anzahl_Methoden;
        Methodeneintrag         Methoden[Anzahl_Methoden];
        short                   Anzahl_Attribute;
        Attributeintrag         Attribute[Anzahl_Attribute];
}
```



Das `class`-Dateiformat

```
Klassendatei
{
        int                     Erkennungszahl;
        short                   Subversionsnummer;
        short                   Versionsnummer;
        short                   Anzahl_Konstanten;
        Konstantenpooleintrag   Konstantenpool[Anza
1];
        short                   Merkmale;
        short                   Klasse;
        short                   Superklasse;
        short                   Anzahl_Interfaces;
        short                   Interfaces[Anzahl_Interaces];
        short                   Anzahl_Felder;
        Feldeintrag             Felder[Anzahl_Felder];
        short                   Anzahl_Methoden;
        Methodeneintrag         Methoden[Anzahl_Methoden];
        short                   Anzahl_Attribute;
        Attributeintrag         Attribute[Anzahl_Attribute];
}
```

Versionsnummer des Compilers

Das `class`-Dateiformat

```
Klassendatei
{
        int                     Erkennungszahl;
        short                   Subversionsnummer;
        short                   Versionsnummer;
        short                   Anzahl_Konstanten;
        Konstantenpooleintrag   Konstantenpool[Anzahl_Konstanten -
1];
        short
        short
        short
        short
        short
        short
        Feldeintr
        short
        Methodene
        short
        Attributeintrag         Attribute[Anzahl_Attribute];
}
```

Typbezeichnung	Typ
CONSTANT_Utf8	1
CONSTANT_Integer	3
CONSTANT_Float	4
CONSTANT_Class	7
CONSTANT_NameAndType	12

```
CONSTANT_Utf8_Eintrag
{
        byte   Typ;
        short  Länge;
        byte   Zeichen[Länge];
}
```

Das `class`-Dateiformat

```
Klassendatei
{
    int                     Erkennungszahl;
    short                   Subversionsnummer;
    short                   Versionsnummer;
    short                   Anzahl_Konstanten;
    Konstantenpooleintrag   Konstantenpool[Anzahl_Konstanten -
1];
    short                   Merkmale;
```

Merkmale	Wert	Bedeutung
ACC_PUBLIC	0x0001	Sichtbar für alle
ACC_PRIVATE	0x0002	Nur für die eigene Klasse sichtbar
ACC_PROTECTED	0x0004	Nur für die abgeleitete Klasse sichtbar
ACC_STATIC	0x0008	Variable oder Methode ist statisch
ACC_SYNCHRONIZED	0x0020	Benutzung muss durch Monitor geschützt werden
ACC_NATIVE	0x0100	Enthält kein Java sondern eine andere Sprache
ACC_INTERFACE	0x0200	Ist eine Schnittstelle
ACC_ABSTRACT	0x0400	Es gibt keinen Code

Das `class`-Dateiformat

```
Klassendatei
{
        int                     Erkennungszahl;
        short                   Subversionsnummer;
        short                   Versionsnummer;
        short                   Anzahl_Konstanten;
        Konstantenpooleintrag   Konstantenpool[Anzahl_Konstanten –
1];
        short                   Merkmale;
        short                   Klasse;
        short                   Superklasse;
        short                   Anzahl_Interfaces;
        short                   Interfaces[Anzahl_Interfaces];
        short                   Anzahl_Felder;
        Feldeintrag             Felder[Anzahl_Felder];
        short                   Anzahl_Methoden;
        Methodeneintrag         Methoden[Anzahl_Methoden];
        short                   Anzahl_Attribute;
        Attributeintrag         Attribute[Anzahl_Attribute];
}
```

Index in den Konstantenpool

Das `class`-Dateiformat

```
Klassendatei
{
        int                     Erkennungszahl;
        short                   Subversionsnummer;
        short                   Versionsnummer;
        short                   Anzahl_Konstanten;
        Konstantenpooleintrag   Konstantenpool[Anzahl_Konstanten –
1];
        short                   Merkmale;
        short                   Klasse;
        short                   Superklasse;
        short                   Anzahl_Interfaces;
        short                   Interfaces[Anzahl_Interfaces];
        short
        Feldeintrag
        short
        Methodeneintrag
        short
        Attributeintrag
}
```

Enthält Indizes auf Interfaceeinträge in den Konstantenpool

```
CONSTANT_InterfaceMethodref_Eintrag
{
    byte Typ;
    short Klassenindex;
    short Name-/Typ-Index;
}
```

Das `class`-Dateiformat

```
Klassendatei
{
    int                     Erkennungszahl;
    short
    short
    short
    Konstantenpooleintrag
1];
    short
    short
    short
    short
    short                   Interfaces[Anzahl_Interaces];
    short                   Anzahl_Felder;
    Feldeintrag             Felder[Anzahl_Felder];
    short                   Anzahl_Methoden;
    Methodeneintrag         Methoden[Anzahl_Methoden];
    short                   Anzahl_Attribute;
    Attributeintrag         Attribute[Anzahl_Attribute];
}
```

```
Feldeintrag
{
    short           Merkmale
    short           Namesindex;
    short           Signatur-Index;
    short           Anzahl_Attribute;
    Attributeintrag Attribute[Anzahl_Attribute]
}
```

Das `class`-Dateiformat

```
Klassendatei
{
    int                     Erkennungszahl;
    short                   Subversionsnummer;
    short                   Versionsnummer;
    short
    Konstantenpooleintrag
1];
    short
    short
    short
    short
    short
    short
    Feldeintrag             Felder[Anzahl_Felder];
    short                   Anzahl_Methoden;
    Methodeneintrag         Methoden[Anzahl_Methoden];
    short                   Anzahl_Attribute;
    Attributeintrag         Attribute[Anzahl_Attribute];
}
```

```
Methodeneintrag
{
    short            Merkmale
    short            Namesindex;
    short            Signatur-Index;
    short            Anzahl_Attribute;
    Attributeintrag  Attribute[Anzahl_Attribute]
}
```

Das `class`-Dateiformat

```
Klassendatei
{
        int
        short
        short
        short
        Konstantenpooleintrag
1];
        short
        short
        short
        short
        short
        short
        Feldeintrag
        short
        Methodeneintrag
        short
        Attributeintrag
}
```

Attribute von Klassen

```
SourceFile-Attribut
{
    short   Attributname-Index;
    int     Attributlänge;
    short   Quelldatei-Index;
}
```

Attribute von Feldern

```
ConstantValue-Attribut
{
    short   Attributname-Index;
    int     Attributlänge;
    short   Konstanten-Index;
}
```

```
Anzahl_Methoden;
Methoden[Anzahl_Methoden];
Anzahl_Attribute;
Attribute[Anzahl_Attribute];
```

Das `class`-Dateiformat

```
Klassendatei
{
        int
        short
        short
        short
        Konstantenpooleintrag
1];
        short
        short
        short
        short
        short
        short
        Feldeintrag
        short                   Anzahl_Methoden;
        Methodeneintrag         Methoden[Anzahl_Methoden];
        short                   Anzahl_Attribute;
        Attributeintrag         Attribute[Anzahl_Attribute];
}
```

Attribute von Methoden

```
Code-Attribut
{
    short               Attributname-Index;
    int                 Attributlänge;
    short               Stackgröße;
    short               Anzahl_lokale_Variablen;
    int                 Code-Länge;
    byte                Code[Code-Länge];
    short               Ausnahmetabelle_Größe;
    {
            short       Anfang_pc;
            short       Ende_pc;
            short       Handler_pc;
            short       Ausnahmentyp;
    }Ausnahmetabelle[Ausnahmetabelle_Größe];
    short               Anzahl_Attribute;
    Attributeintrag     Attribute[Anzahl_Attribute]
}
```

Das `class`-Dateiformat

```
Klassendatei
{
        int
        short
        short
        short
        Konstantenpooleintrag
1];
        short
        short
        short
        short
        short
        short
        Feldeintrag
        short                   Anzahl_Methoden;
        Methodeneintrag         Methoden[Anzahl_Methoden];
        short                   Anzahl_Attribute;
        Attributeintrag         Attribute[Anzahl_Attribute];
}
```

Attribute von Methoden

```
Exceptions-Attribut
{
    short Attributname-Index;
    int   Attributlänge;
    short Anzahl_Ausnahmen;
    short Ausnahmenindextabelle[Anzahl_Ausnahmen];
}
```

Das `class`-Dateiformat

```
Klassendatei
{
        int
        short
        short
        short
        Konstantenpooleintrag
1];
        short
        short
        short
        short
        short
        short
        Feldeintrag
        short
        Methodeneintrag
        short
        Attributeintrag
}
```

Signaturen (Zeichenkette im Utf8-Format)

Buchstabe	Java-Datentyp
Z	Boolean
B	Byte
C	Char
S	Short
I	Int
J	Long
F	Float
D	Double

Objekte: **L**java/lang/Object**;**

Arrays: [#Datentyp

Void: **V**

Beispiel:

```
StringBuffer append( char[] str,
                     int offset,
                     int len);
```

enstricht

([CII)Ljava/lang/StringBuffer;

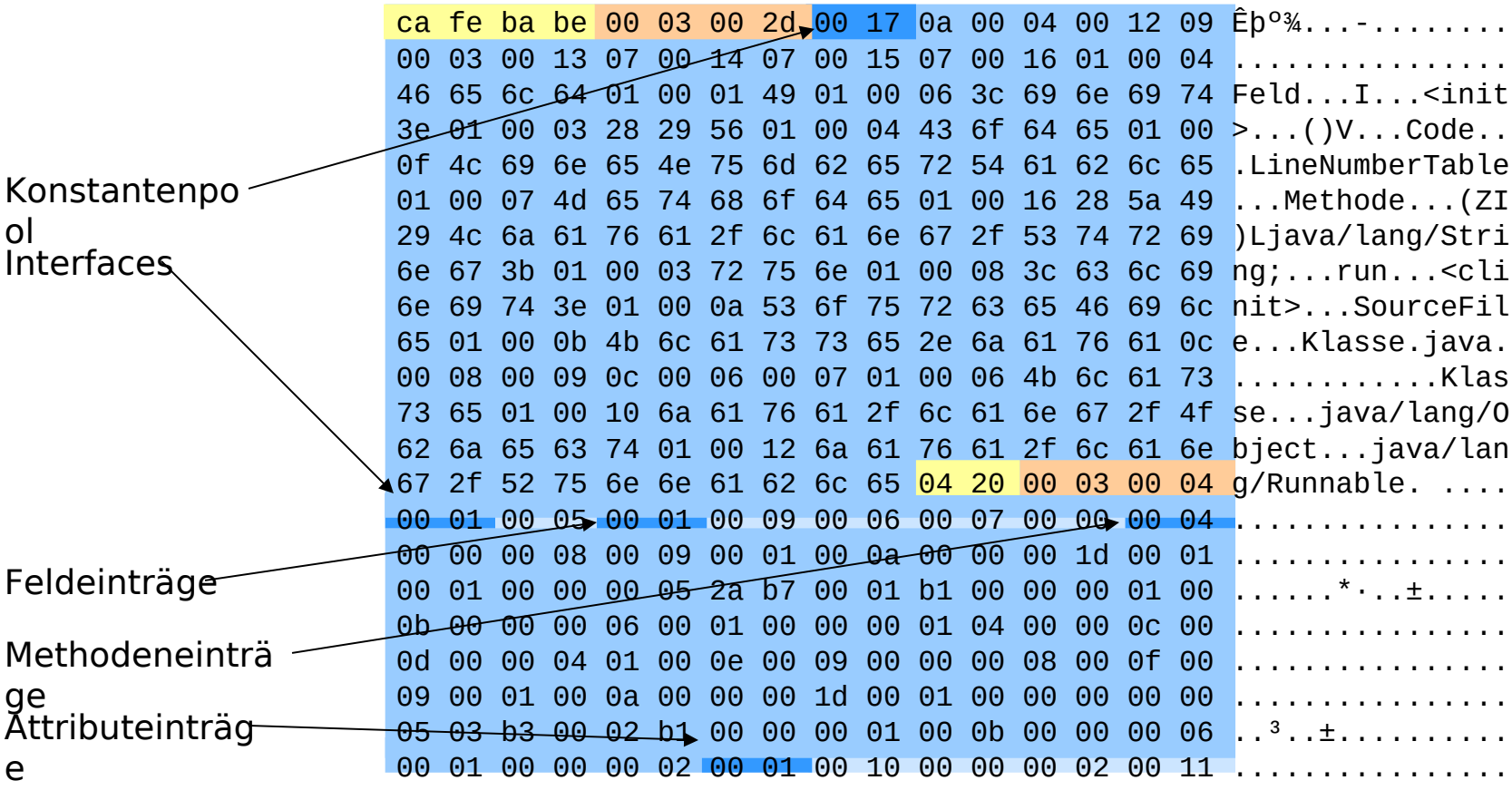
Dateibeispiel

```
abstract class Klasse implements Runnable {
    public static int Feld = 0;
    abstract String Methode(boolean b, int i);
}
```

```
ca fe ba be 00 03 00 2d 00 17 0a 00 04 00 12 09 Êþº¾...-........
00 03 00 13 07 00 14 07 00 15 07 00 16 01 00 04 ................
46 65 6c 64 01 00 01 49 01 00 06 3c 69 6e 69 74 Feld...I...<init
3e 01 00 03 28 29 56 01 00 04 43 6f 64 65 01 00 >...()V...Code..
0f 4c 69 6e 65 4e 75 6d 62 65 72 54 61 62 6c 65 .LineNumberTable
01 00 07 4d 65 74 68 6f 64 65 01 00 16 28 5a 49 ...Methode...(ZI
29 4c 6a 61 76 61 2f 6c 61 6e 67 2f 53 74 72 69 )Ljava/lang/Stri
6e 67 3b 01 00 03 72 75 6e 01 00 08 3c 63 6c 69 ng;...run...<cli
6e 69 74 3e 01 00 0a 53 6f 75 72 63 65 46 69 6c nit>...SourceFil
65 01 00 0b 4b 6c 61 73 73 65 2e 6a 61 76 61 0c e...Klasse.java.
00 08 00 09 0c 00 06 00 07 01 00 06 4b 6c 61 73 ............Klas
73 65 01 00 10 6a 61 76 61 2f 6c 61 6e 67 2f 4f se...java/lang/O
62 6a 65 63 74 01 00 12 6a 61 76 61 2f 6c 61 6e bject...java/lan
67 2f 52 75 6e 6e 61 62 6c 65 04 20 00 03 00 04 g/Runnable. ....
00 01 00 05 00 01 00 09 00 06 00 07 00 00 00 04 ................
00 00 00 08 00 09 00 01 00 0a 00 00 00 1d 00 01 ................
00 01 00 00 00 05 2a b7 00 01 b1 00 00 00 01 00 ......*·..±.....
0b 00 00 00 06 00 01 00 00 00 01 04 00 00 0c 00 ................
0d 00 00 04 01 00 0e 00 09 00 00 00 08 00 0f 00 ................
09 00 01 00 0a 00 00 00 1d 00 01 00 00 00 00 00 ................
05 03 b3 00 02 b1 00 00 00 01 00 0b 00 00 00 06 ..³..±..........
00 01 00 00 00 02 00 01 00 10 00 00 00 02 00 11 ................
```

Dateibeispiel

```
abstract class Klasse implements Runnable {
    public static int Feld = 0;
    abstract String Methode(boolean b, int i);
}
```




```
ca fe ba be 00 03 00 2d 00 17 0a 00 04 00 12 09  Êþº¾...-........
00 03 00 13 07 00 14 07 00 15 07 00 16 01 00 04  ................
46 65 6c 64 01 00 01 49 01 00 06 3c 69 6e 69 74  Feld...I...<init
3e 01 00 03 28 29 56 01 00 04 43 6f 64 65 01 00  >...()V...Code..
0f 4c 69 6e 65 4e 75 6d 62 65 72 54 61 62 6c 65  .LineNumberTable
01 00 07 4d 65 74 68 6f 64 65 01 00 16 28 5a 49  ...Methode...(ZI
29 4c 6a 61 76 61 2f 6c 61 6e 67 2f 53 74 72 69  )Ljava/lang/Stri
6e 67 3b 01 00 03 72 75 6e 01 00 08 3c 63 6c 69  ng;...run...<cli
6e 69 74 3e 01 00 0a 53 6f 75 72 63 65 46 69 6c  nit>...SourceFil
65 01 00 0b 4b 6c 61 73 73 65 2e 6a 61 76 61 0c  e...Klasse.java.
00 08 00 09 0c 00 06 00 07 01 00 06 4b 6c 61 73  ............Klas
73 65 01 00 10 6a 61 76 61 2f 6c 61 6e 67 2f 4f  se...java/lang/O
62 6a 65 63 74 01 00 12 6a 61 76 61 2f 6c 61 6e  bject...java/lan
67 2f 52 75 6e 6e 61 62 6c 65 04 20 00 03 00 04  g/Runnable. ....
00 01 00 05 00 01 00 09 00 06 00 07 00 00 00 04  ................
00 00 00 08 00 09 00 01 00 0a 00 00 00 1d 00 01  ................
00 01 00 00 00 05 2a b7 00 01 b1 00 00 00 01 00  ......*·..±.....
0b 00 00 00 06 00 01 00 00 00 01 04 00 00 0c 00  ................
0d 00 00 04 01 00 0e 00 09 00 00 00 08 00 0f 00  ................
09 00 01 00 0a 00 00 00 1d 00 01 00 00 00 00 00  ................
05 03 b3 00 02 b1 00 00 00 01 00 0b 00 00 00 06  ..³..±..........
00 01 00 00 00 02 00 01 00 10 00 00 00 02 00 11  ................
```

Dateibeispiel

```
          ...
          00 17                                     Anzahl Konstanten  (23)
Konstantenpool
     3.   07 00 14                                  CONSTANT_Class     (20)
     12.  01 00 07 4d 65 74 68 6f 64 65             CONSTANT_Utf8      (7)   „Methode“
     13.  01 00 16 ...                              CONSTANT_Utf8      (22)  „(ZI)Ljava/lang/String;“
     16.  01 00 0a 53 6f 75 72 63 65 46 69 6c 65    CONSTANT_Utf8      (10)  „SourceFile“
     17.  01 00 0b 4b 6c 61 73 73 65 2e 6a 61 76 61 CONSTANT_Utf8      (11)  „Klasse.class“
     20.  01 00 06 4b 6c 61 73 73 65                CONSTANT_Utf8      (6)   „Klasse“
          ...
          04 20 (ACC_ABSTRACT, ACC_SYNCHRONIZED)    Merkmal
          00 03                                     Klasse             (3)
          ...
          00 04                                     Anzahl_Methoden    (4)
Methodeneinträge
          ...
     2.   04 00 (ACC_ABSTRACT)                      Merkmale
          00 0c                                     NamensIndex        (12)
          00 0d                                     Signatur-Index     (13)
          00 00                                     Anzahl_Attribute   (0)
          ...
          00 01                                     Anzahl_Attribute   (1)
Attributeinträge
     1.   00 10                                     Attributname-Index (16)
          00 00 00 02                               Attributlänge      (2)
          00 11                                     Quelldatei-Index   (17)
```

Die Maschinenspezifikation

- Datentypen
- Register und lokale Variablen
- Laufzeitumgebung
- Stack und Heap
- Befehlssatz
- Programm Beispiel

Die Maschinenspezifikation
Datentypen

Typ	Bytes	Minimum	Maximum	Representation
byte	1	-128	127	2-Komplement
short	2	-32768	32767	2-Komplement
int	4	-2147483648	2147483647	2-Komplement
long	8	-9223372036854775808	-9223372036854775807	2-Komplement
float	4	$+-1,40239846^{-45}$	$+-3,40282347^{38}$	IEEE-754
double	8	$+-4,9406564584124654^{-324}$	$+-1,79769313486231570^{300}$	IEEE-754
char	2	\u0000	\uFFFF	Unicode
object	4			unbestimmt
returnAdresse	4			unbestimmt
boolean				int

- Die Endianess VM-interner Daten ist nicht spezifiziert.

Die Maschinenspezifikation
Register und lokale Variablen

Register

- es gibt keine
- VM-intern gibt es 4 Register

pc	ProgramCounter
vars	zeigt auf Speicherbereich der lokalen Variablen
optop	zeigt auf obersten Stackwert
frame	zeigt auf aktuelle Laufzeitumgebung

Lokale Variablen

- abgelegt in 32 Bit breiten aufeinander folgenden Speicherbereichen
- long, double belegen 64 Bit
- von 0 an fortlaufend durchnumeriert

Die Maschinenspezifikation
Laufzeitumgebung

Enthält

- Verweise auf Symboltabelle des Java Interpreters der aktuellen Klasse und Methode
- Register der aufrufenden Methode
- Liste von Exception Handlern (markierter Codebereich, Exception Code)

Unterstützt

- Dynamisches (spätes) Binden
- Rücksprung aus Methoden
- Weiterreichen von Ausnahmezuständen und Fehlern
- Implementationsabhängige Erweiterungen (z.B. Debugging)

Die Maschinenspezifikation
Stack und Heap

Stack

- Ist Operandenstack
- 32 Bit breit (long, double 2x32 Bit)
- Endianess ist nicht festgelegt

Heap

- Enthält die mit „new“ angelegten Objekte
- Speicherverwaltung ist nicht spezifiziert
- Objekte können nicht gezielt freigegeben werden (Freispeicherverwaltung)
- Nachteil des Garbage Collectors
  - mehr Zeitaufwand
  - schlechtes Echtzeitverhalten

Die Maschinenspezifikation
Befehlssatz

Allgemein

- Opcodes werden durch ein Byte dargestellt
- Operanden im Big-Endian-Format
- Kein Alignment für 16 Bit große Operanden
- Anzahl und Größe der Operanden aus Opcode ersichtlich

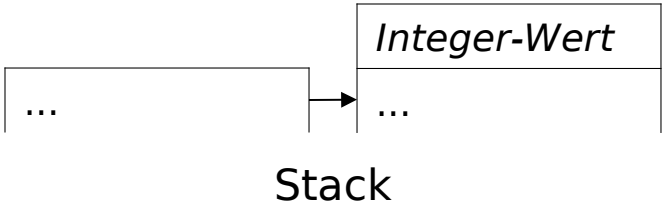
Besondere Opcodes

- Opcodes für Stackmanipulation
- Opcodes zur Array-Behandlung
- Opcodes zur Unterstützung von Parallelverarbeitung

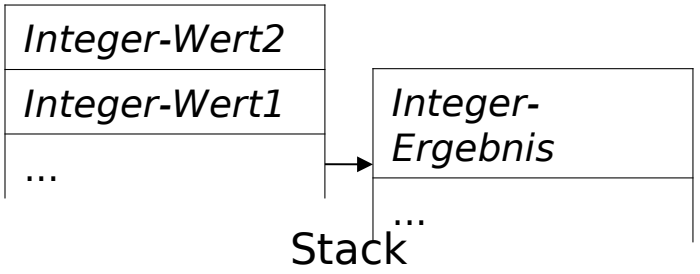
Die Maschinenspezifikation
Befehlssatz

Beispiel

sipush *byte1 byte2*



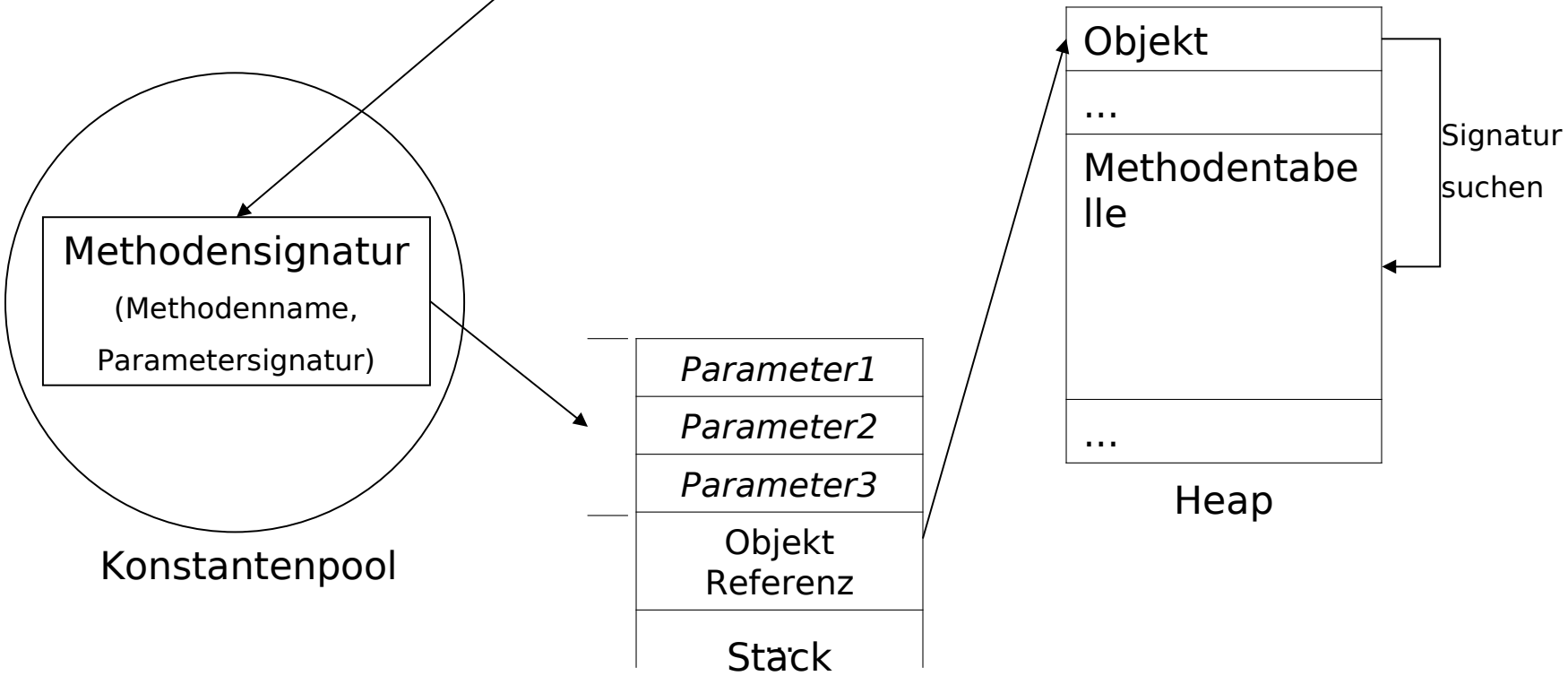
iadd



Die Maschinenspezifikation
Befehlssatz

Methodenaufrufe

invokevirtual *index-byte1 index-byte2*



Die Maschinenspezifikation
Programm Beispiel 1

```
class Main {

    private static int add(int a, int b){
          return a + b;
    }

    public static void main(String args[]){
          int x = 2;
          int y = add(1 + x, 3);
    }

}
```

Die Maschinenspezifikation
Programm Beispiel 1

```
Method void main(java.lang.String[])
   0 iconst_2
   1 istore_1
   2 iconst_1
   3 iload_1
   4 iadd
   5 iconst_3
   6 invokestatic #2 <Method int add(int, int)>
   9 istore_2
  10 return
```

0	Array-Ref.
1	???
2	???

lokale Variablen

...

Stack

Die Maschinenspezifikation
Programm Beispiel 1

```
Method void main(java.lang.String[])
    0 iconst_2
    1 istore_1
    2 iconst_1
    3 iload_1
    4 iadd
    5 iconst_3
    6 invokestatic #2 <Method int add(int, int)>
    9 istore_2
   10 return
```

0	Array-Ref.
1	???
2	???

lokale Variablen

2
...

Stack

Die Maschinenspezifikation
Programm Beispiel 1

```
Method void main(java.lang.String[])
   0 iconst_2
   1 istore_1
   2 iconst_1
   3 iload_1
   4 iadd
   5 iconst_3
   6 invokestatic #2 <Method int add(int, int)>
   9 istore_2
  10 return
```

0	Array-Ref.
1	**2**
2	???

lokale Variablen

...

Stack

Die Maschinenspezifikation
Programm Beispiel 1

```
Method void main(java.lang.String[])
   0 iconst_2
   1 istore_1
   2 iconst_1
   3 iload_1
   4 iadd
   5 iconst_3
   6 invokestatic #2 <Method int add(int, int)>
   9 istore_2
  10 return
```

0	Array-Ref.
1	2
2	???

lokale Variablen

1
...

Stack

Die Maschinenspezifikation
Programm Beispiel 1

```
Method void main(java.lang.String[])
   0 iconst_2
   1 istore_1
   2 iconst_1
   3 iload_1
   4 iadd
   5 iconst_3
   6 invokestatic #2 <Method int add(int, int)>
   9 istore_2
  10 return
```

0	Array-Ref.
1	2
2	???

lokale Variablen

2
1
...

Stack

Die Maschinenspezifikation
Programm Beispiel 1

```
Method void main(java.lang.String[])
   0 iconst_2
   1 istore_1
   2 iconst_1
   3 iload_1
   4 iadd
   5 iconst_3
   6 invokestatic #2 <Method int add(int, int)>
   9 istore_2
  10 return
```

0	Array-Ref.
1	2
2	???

lokale Variablen

3
...

Stack

Die Maschinenspezifikation
Programm Beispiel 1

```
Method void main(java.lang.String[])
   0 iconst_2
   1 istore_1
   2 iconst_1
   3 iload_1
   4 iadd
   5 iconst_3
   6 invokestatic #2 <Method int add(int, int)>
   9 istore_2
  10 return
```

0	Array-Ref.
1	2
2	???

lokale Variablen

3
3
...

Stack

Die Maschinenspezifikation
Programm Beispiel 1

```
Method void main(java.lang.String[])
   0 iconst_2
   1 istore_1
   2 iconst_1
   3 iload_1
   4 iadd
   5 iconst_3
   6 invokestatic #2 <Method int add(int, int)>
   9 istore_2
  10 return
```

0	Array-Ref.
1	2
2	???

lokale Variablen

6
...

Stack

Die Maschinenspezifikation
Programm Beispiel 1

```
Method void main(java.lang.String[])
   0 iconst_2
   1 istore_1
   2 iconst_1
   3 iload_1
   4 iadd
   5 iconst_3
   6 invokestatic #2 <Method int add(int, int)>
   9 istore_2
  10 return
```

0	Array-Ref.
1	2
2	**6**

lokale Variablen

...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
class Klasse {
    public String meinName(){ return "Klasse"; }
}

class AbgeleiteteKlasse extends Klasse {
    public String meinName(){ return "AbgeleiteteKlasse"; }
}


class Main {
    public static void main(String args[]){
          Klasse klasse = new AbgeleiteteKlasse();
          System.out.println("Ich bin die " + klasse.meinName());
    }
}
```

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
   0 new #2 <Class AbgeleiteteKlasse>
   3 dup
   4 invokespecial #3 <Method AbgeleiteteKlasse()>
   7 astore_1
   8 getstatic #4 <Field java.io.PrintStream out>
  11 new #5 <Class java.lang.StringBuffer>
  14 dup
  15 invokespecial #6 <Method java.lang.StringBuffer()>
  18 ldc #7 <String "Ich bin die ">
  20 invokevirtual #8 <Method\
         java.lang.StringBuffer append(java.lang.String)>
  23 aload_1
  24 invokevirtual #9 <Method java.lang.String meinName()>
     ...
  36 return
```

0	Array-Ref.
1	???

lokale Variablen

...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	???

lokale Variablen

OR Ab.Klasse
...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	???

lokale Variablen

OR Ab.Klasse
OR Ab.Klasse
...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	???

lokale Variablen

OR Ab.Klasse
...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	**OR Ab.Klasse**

lokale Variablen

...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	OR Ab.Klasse

lokale Variablen

OR PrintStream
...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	OR Ab.Klasse

lokale Variablen

OR StringBuffer
OR PrintStream(out)
...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	OR Ab.Klasse

lokale Variablen

OR StringBuffer
OR StringBuffer
OR PrintStream(out)
...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	OR Ab.Klasse

lokale Variablen

OR StringBuffer
OR PrintStream(out)
...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	OR Ab.Klasse

lokale Variablen

OR String
OR StringBuffer
OR PrintStream(out)
...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	OR Ab.Klasse

lokale Variablen

OR StringBuffer
OR PrintStream(out)
...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	OR Ab.Klasse

lokale Variablen

OR Ab.Klasse
OR StringBuffer
OR PrintStream(out)
...

Stack

Die Maschinenspezifikation
Programm Beispiel 2

```
Method void main(java.lang.String[])
    0 new #2 <Class AbgeleiteteKlasse>
    3 dup
    4 invokespecial #3 <Method AbgeleiteteKlasse()>
    7 astore_1
    8 getstatic #4 <Field java.io.PrintStream out>
   11 new #5 <Class java.lang.StringBuffer>
   14 dup
   15 invokespecial #6 <Method java.lang.StringBuffer()>
   18 ldc #7 <String "Ich bin die ">
   20 invokevirtual #8 <Method\
          java.lang.StringBuffer append(java.lang.String)>
   23 aload_1
   24 invokevirtual #9 <Method java.lang.String meinName()>
      ...
   36 return
```

0	Array-Ref.
1	OR Ab.Klasse

lokale Variablen

OR String
OR StringBuffer
OR PrintStream(out)
...

Stack