



Java lernen – Parameter

HPI-Team

Rückblick

Bisher...

- ...hatten wir Methoden mit verschiedenen Rückgabetypen
- Beispiel: `String sagHallo() { return "Hallo"; }`

Jetzt...

- ...lernen wir wie man Eingabewerte an Methoden übergibt.
- Diese Eingabewerte nennt man Parameter.

Eingabewerte für Methoden

Parameter sind Werte, die einer Methode übergeben werden

Beispiel

```
1 class HalloWelt {  
2     public void sag(String aussage) {  
3         System.out.println(aussage);  
4     }  
5  
6     public void sagNichts() {  
7         System.out.println("Nichts");  
8     }  
9 }
```

Diagram illustrating the components of a method signature:

- Typ** (Type): Points to `String` in `sag(String aussage)`.
- Bezeichner** (Identifier): Points to `aussage` in `sag(String aussage)`.

„aussage“ ist hierbei der Parameter

Die Methode `sagNichts` hat keine Parameter (leere Klammern)

Aufrufe der Methoden

Die Werte für die Parameter werden beim Aufruf der Methode übergeben

Beispiel

```
1 class HalloWelt {  
2     public void sag(String aussage) {  
3         System.out.println(aussage);  
4     }  
5 }
```



Gültiger Aufruf

```
object.sag("Hallo Welt");
```

Aufruf mit einer Variablen

```
String halloOpenHPI = "Hallo openHPI";  
object.sag(halloOpenHPI);
```

Achtung: Für jeden definierten Parameter muss man einen Wert

~~object.sag();~~ Fehler!

Mehrere Parameter

Mehrere Parameter werden durch Kommata getrennt

Beispiel

```
1 class Hallowelt {  
2     public void sag(String aussage, String zweiteAussage) {  
3         System.out.println(aussage);  
4         System.out.println(zweiteAussage);  
5     }  
6 }
```

Gültiger Aufruf

```
object.sag("Hallo Welt", "Hallo openHPI");
```

Java lernen -
Parameter

HPI-Team

Datentypen

Parameter können für jeden Datentyp definiert werden

Beispiel

```
1 class HalloWelt {  
2     public void sag(String aussage, int nummer) {  
3         System.out.println(aussage);  
4         System.out.println(nummer);  
5     }  
6 }
```

Gültiger Aufruf

```
objekt.sag("Hallo Welt", 1);
```

Achtung: Die Reihenfolge und der Datentyp müssen stimmen

```
objekt.sag(1, "Hallo Welt"); Fehler!
```

Java lernen -
Parameter

HPI-Team

Gültigkeitsbereich - Scope

Parameter sind nur innerhalb der Methode gültig

Beispiel

```
1 class HalloWelt{  
2     public void sag(String aussage){  
3         System.out.println(aussage);  
4     }  
5  
6     public void sagLaut(){  
7         System.out.println(aussage);  
8     }  
9 }
```



Fehler!
„aussage“ ist nur in
der Methode sag
definiert

Java lernen -
Parameter

HPI-Team

Gültigkeitsbereich - Scope

Parameter sind nur innerhalb der Methode gültig

Beispiel

```
1 class HalloWelt{  
2     public void sag(String aussage){  
3         System.out.println(aussage);  
4     }  
5  
6     public void sagLaut(){  
7         System.out.println(aussage);  
8     }  
9 }
```

Fehler!
„aussage“ ist nur in
der Methode sag
definiert

Java lernen -
Parameter

HPI-Team



Java lernen – Attribute

HPI-Team

Rückblick

Bisher

- Bisher haben wir Methoden als das Verhalten eines Objekts kennengelernt.
- Beispiel: Der Papagei kann Hallo sagen, fliegen, fressen, ...

Jetzt

- Objekte haben aber in der Regel auch einen Zustand.
- Dieser Zustand wird in sogenannten Attributen gespeichert.
- Beispiel: Der Papagei ist rot, 20 Jahre alt und heißt Paco.

**Java lernen -
Attribute**

HPI-Team

Attribute als Variablen eines Objekts

Attribute werden für jedes Objekt in deren Klasse angelegt

Beispiel

```
1 public class Papagei {  
2     String name = "Paco";  
3  
4     public void sagDeinenNamen() {  
5         System.out.println("Ich heiße " + name);  
6     }  
7 }
```

„name“ ist hierbei das Attribut

**Java lernen -
Attribute**

HPI-Team

Anlegen von Attributen

Attribute...

- ...werden ganz oben in der Klasse angelegt
- ...werden wie Variablen klein geschrieben
- ...können für jeden Datentyp definiert werden



```
1 public class Papagei {  
    String name = "Paco";  
    int alter = 22;  
4  
5 public void sagDeinenNamen() {  
6     System.out.println("Ich heiße " + name);  
7 }  
8  
9 public void sagDeinAlter() {  
10    System.out.println("Ich bin " + alter + " Jahre alt");  
11 }  
12 }
```

Gültigkeitsbereiche im Vergleich

```

1 public class Papagei {
2
3     public void sagDeinenName(){
4         String name = "Paco";
5         System.out.println(name);
6     }
7
8     public void schreiDeinenNamen(){
9         System.out.println(
10             name.toUpperCase());
11     }
12 }

```

Funktioniert!

Funktioniert nicht, da „name“ nur innerhalb von „sagDeinenNamen“ existiert, d.h. eine lokale Variable in der Methode „sagDeinenNamen“ ist.

```

1 public class Papagei {
2     String name = "Paco";
3
4     public void sagDeinenName(){
5         System.out.println(name);
6     }
7
8     public void schreiDeinenNamen(){
9         System.out.println(
10             name.toUpperCase());
11     }
12 }

```

Funktioniert!

Funktioniert auch, da „name“ ein Attribut ist und innerhalb des gesamten Objekts zur Verfügung steht.

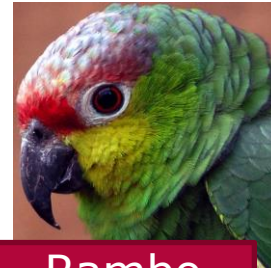
Setzen des Werts von Attributen während der Objekterzeugung

Parameter im Konstruktor:

```
1 public class Papagei {  
2     String name;  
3  
4     public Papagei(String name) {  
5         this.name = name;  
6     }  
7 }
```



Paco



Rambo



Polly

Zugriff über:

```
Papagei paco = new Papagei("Paco");  
Papagei polly = new Papagei("Polly");  
Papagei rambo = new Papagei("Rambo");
```

Java lernen -
Attribute

HPI-Team

Zugriff auf Attribute von außerhalb des Objekts

Definition von „Getter“ und „Setter“-Methoden:



Rambo

```

1 public class Papagei {
2     private String name;
3
4     public Papagei (String name) {
5         this.name = name;
6     }
7
8     public String getName() {
9         return name;
10    }
11
12    public void setName(String name) {
13        this.name = name;
14    }
15 }

```

```

Papagei paco = new Papagei("Paco");
paco.getName();
paco.setName("Rambo");

```

**Java lernen -
Attribute**

HPI-Team

Zugriff auf Attribute von außerhalb des Objekts

Definition von „Getter“ und „Setter“-Methoden:



```

1 public class Papagei {
2     private String name;
3
4     public Papagei (String name) {
5         this.name = name;
6     }
7
8     public String getName() {
9         return name;
10    }
11 }

```

```

Papagei paco = new Papagei("Paco");
paco.getName();
paco.setName("Rambo");

```

Fehler

Umbenennen des Papageis ist außerhalb der Klasse nicht mehr möglich.

Java lernen - Attribute
HPI-Team

Zugriff auf Attribute von außerhalb des Objekts

Über „Getter“ und „Setter“-Methoden kann der Entwickler anderen Objekten Lese- bzw. Schreibrechte an Attributen einräumen.

- Gibt es für ein Attribut nur einen Getter, kann das Attribut von außen nur gelesen aber nicht beschrieben werden.
- Gibt es für ein Attribut nur einen Setter, kann das Attribut von außen nur beschrieben aber nicht gelesen werden.
- Gibt es sowohl einen Getter als auch einen Setter, kann das Attribut von außen gelesen und beschrieben werden.

→ Wichtiges Konzept der Rechteverwaltung innerhalb eines Programms

**Java lernen -
Attribute**

HPI-Team

Zugriff auf Attribute von außerhalb des Objekts

Sichtbarkeit

- Um den oben beschriebenen Mechanismus effizient einsetzen zu können fehlt uns noch das Konzept der „Sichtbarkeit“ von Attributen und Methoden eines Objekts.
- **public, private, protected**
- Mit diesem Konzept werden wir uns in der nächsten Woche genauer beschäftigen.
- Bis dahin begnügen wir uns mit der vereinfachten Faustregel, dass Methoden zumeist **public**, Attribute dagegen als **private** deklariert werden.

Java lernen -
Attribute

HPI-Team

Zugriff auf Attribute von außerhalb des Objekts

Beispiel:

```
1 public class Papagei {  
2     private String name;  
3  
4     public Papagei (String name) {  
5         this.name = name;  
6     }  
7  
8     public String getName() {  
9         return name;  
10    }  
11 }
```

**Java lernen -
Attribute**

HPI-Team

Zugriff auf Attribute von außerhalb des Objekts

Beispiel:

```
1 public class Papagei {  
2     private String name;  
3  
4     public Papagei (String name) {  
5         this.name = name;  
6     }  
7  
8     public String getName() {  
9         return name;  
10    }  
11 }
```

**Java lernen -
Attribute**

HPI-Team



Java lernen – Bedingungen und logische Operatoren

HPI-Team

Bisher

- Geradliniger Ablauf
- Variablen für
 - Ausgabe
 - Berechnungen



Jetzt

- Ablauf mit Verzweigungen
 - Entscheidung treffen
 - Abhängig von Werten



**Java lernen -
Bedingungen &
logische
Operatoren**

HPI-Team

Wie trifft man Entscheidungen?

Bedingungen/ Vorschriften

- Bestimmen Verhalten
- Im Idealfall
 - Bedingung erfüllt (**wahr**)
 - Oder nicht erfüllt (**falsch**)

Grundlage einfacher
Entscheidungen:

Wahrheitswert (boolean)



**Java lernen -
Bedingungen &
logische
Operatoren**

HPI-Team

Einfache Bedingungen

Wenn () dann () sonst ()

```
1 public class Papagei {  
2     public void sprich(boolean hungrig){  
3         if(hungrig){  
4             System.out.println("Ich bin hungrig");  
5         } else {  
6             System.out.println("Ich bin nicht hungrig");  
7         }  
8     }  
9 }
```

false



Verneinung – Der ! Operator

Nun soll der Papagei nur dann sprechen wenn er **nicht** hungrig ist.

```
1 [...]
2 ▾  if(!hungrig){
3     System.out.println("Ich bin nicht hungrig");
4 }
5 [...]
```

! = nicht

Java lernen -
Bedingungen &
logische
Operatoren

HPI-Team

Verknüpfen von Bedingungen – UND und ODER

Berücksichtigung mehrerer Faktoren

&& (Und)	True	False
True	True	False
False	False	False

 (Oder)	True	False
True	True	True
False	True	False

Bindungsstärke

&& bindet stärker als || (Klammerung!)

Java lernen -
Bedingungen &
logische
Operatoren

HPI-Team

Verknüpfen von Bedingungen – UND und ODER

Beispiele

 `true && true == true;`
 `false && true == false;`
 `!false && true == true;`

 `false || true == true;`
 `!false || false == true;`

**Java lernen -
Bedingungen &
logische
Operatoren**

HPI-Team

Vergleichen

Bevor man sich entscheidet sind oft Vergleiche notwendig

Primitive Datentypen (Zahlen, Wahrheitswerte):

```
1 == 2 // ergibt false
3 == 3 // ergibt true
```

Strings müssen über **equals** verglichen werden:

```
"Hallo Welt".equals("Hallo") // ergibt false
"Hallo".equals("Hallo") // ergibt true
```

Primitive Datentypen vergleicht man mit "=="

Objektdatentypen vergleicht man mittels ihrer "equals" Methode

Java lernen -
Bedingungen &
logische
Operatoren

HPI-Team

Die Welt ist nicht 1/0

Nicht immer können wir Entscheidungen auf zwei Varianten reduzieren:



```
1 public class Papagei {  
2  
3     public void sprich() {  
4         if (istHungrig()) {  
5             System.out.println("Hunger!");  
6         } else if (istMuede()) {  
7             System.out.println("Gute Nacht.");  
8         } else {  
9             System.out.println("Hallo Welt!");  
10        }  
11    }  
12 }
```

Java lernen -
Bedingungen &
logische
Operatoren

HPI-Team

Die Welt ist nicht 1/0

Nicht immer können wir Entscheidungen auf zwei Varianten reduzieren:

```
1 ▼ public class Papagei {  
2  
3 ▼     public void sprich() {  
4 ▼         if (istHungrig()) {  
5             System.out.println("Hunger!");  
6 ▼         } else if (istMuede()) {  
7             System.out.println("Gute Nacht.");  
8 ▼         } else {  
9             System.out.println("Hallo Welt!");  
10        }  
11    }  
12 }
```

**Java lernen -
Bedingungen &
logische
Operatoren**

HPI-Team



Java lernen – Schleifen

HPI-Team

Bisher

Bisher kann unser Papagei einen Satz sagen:

```
1 public class Papagei {  
2     public void sprich(){  
3         String satz = "Hallo Welt";  
4         System.out.println(satz);  
5     }  
6 }
```

Ausgabe

Hallo Welt

**Java lernen -
Schleifen**

HPI-Team

Wiederholung

Es gibt Situationen, in denen man eine Aktion mehrfach ausführen möchte.
Zum Beispiel gibt es Papageien, die Sätze wiederholen:

```
1 public class Papagei {  
2     public void sprich(){  
3         String satz = "Hallo Welt";  
4         System.out.println(satz);  
5         System.out.println(satz);  
6     }  
7 }
```

Ausgabe

```
Hallo Welt  
Hallo Welt
```

**Java lernen -
Schleifen**

HPI-Team

Noch mehr Wiederholung

Was ist, wenn man etwas 10 mal wiederholen möchte?

```
1 public class Papagei {  
2     public void sprich(){  
3         String satz = "Hallo Welt";  
4         System.out.println(satz);  
5         System.out.println(satz);  
6         [...]  
7     }  
8 }
```

Zu viel Schreibaufwand!

**Java lernen -
Schleifen**

HPI-Team

Die while Schleife

Wiederholen von Anweisungen bis eine Bedingung nicht mehr zutrifft

Beispiel: Zehnmaliges Wiederholen eines Satzes

```
1 public class Papagei{  
2     void sprich(){  
3         int i = 0;  
4         while(i < 10){  
5             i++;  
6             System.out.println("Hallo Welt");  
7         }  
8     }  
9 }
```

Java lernen -
Schleifen

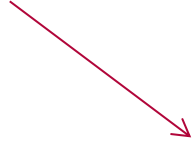
HPI-Team

Aufbau der while-Schleife

```
while (i < 10) {i++;...}
```



Die Bedingung:
entscheidet wann
die Schleife
beendet wird



**Wird solange ausgeführt wie die Bedingung
wahr ist. Nach jedem Schleifendurchlauf
wird die Bedingung erneut überprüft.**

Aufbau der while-Schleife

Achtung: Gefahr von Endlosschleifen

```
1  [...]
2  int i = 0;
3  while (i<10){
4      i--;
5      System.out.println(i);
6  }
7  [...]
```

Die Variable i wird nie größer als 10!!

Java lernen -
Schleifen

HPI-Team

Eigentlicher Einsatz der while Schleife

- Bisher: Bekannte Anzahl von Wiederholungen mit einer Zählvariablen
- Anweisungen in der while Schleife werden wiederholt, solange eine beliebig aufgebaute Bedingung wahr ist
- Anzahl der Schleifendurchläufe ist dabei unbekannt

```
3      [...]  
4  while (alex.istHungrig()){  
5      alex.issFutter();  
6  }  
7  [...]
```

While vs. Do-While

while Schleife

Bedingung prüfen

Schleifenkörper ausführen

Bedingung prüfen

Schleifenkörper ausführen

do-while Schleife

Schleifenkörper ausführen

Bedingung prüfen

Schleifenkörper ausführen

Bedingung prüfen

```
while(i < 10){  
    i++;  
    System.out.println("Hallo Welt");  
}
```

```
do{  
    i++;  
    System.out.println("Hallo Welt");  
} while(i < 10);
```

**Java lernen -
Schleifen**

HPI-Team

Die for-Schleife

Beispiel

```
1 public class Papagei {  
2     public void sprich(){  
3         String satz = "Hallo Welt";  
4         for(int i = 0;i<10;i++){  
5             System.out.println(satz);  
6         }  
7     }  
8 }
```

**Java lernen -
Schleifen**

HPI-Team

Aufbau der for-Schleife

**Anlegen der Zählvariable:
ein Integer**

Abbruchbedingung:

**Wenn dies false ergibt wird
die Schleife beendet**

```
for (int i = 0; i < 10; i++) {  
    System.out...  
}
```

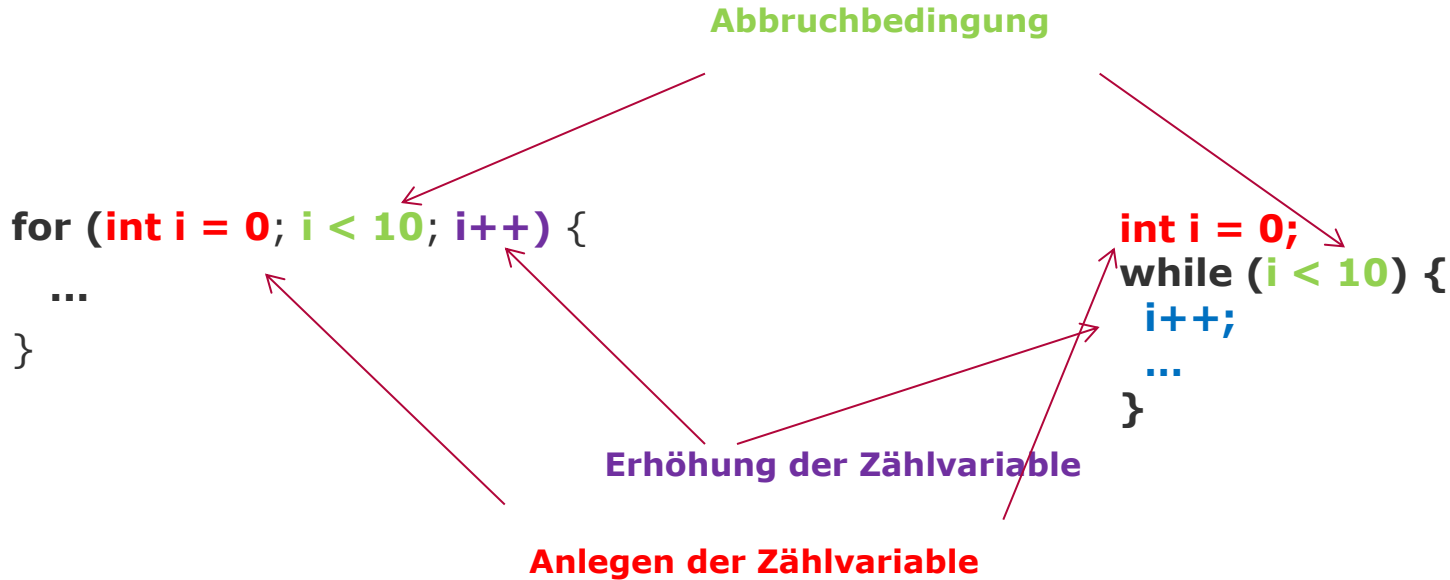
**Wird für jedes Mal
ausgeführt, bei dem die
Bedingung true wird**

**Erhöhung der
Zählvariable, wird nach
jedem Durchlauf
ausgeführt**

**Java lernen -
Schleifen**

HPI-Team

Aufbau der for-Schleife



Aufbau der for-Schleife

```
for (int i = 0; i < 10; i++) {  
    ...  
}
```

```
int i = 0;  
while (i < 10) {  
    i++;  
    ...  
}
```

Java lernen -
Schleifen

HPI-Team



Java lernen – Arrays

HPI-Team

Was ist ein Array?

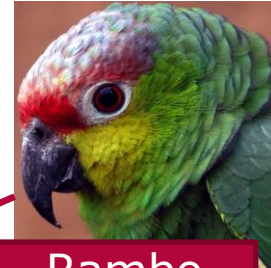
Array

- Container/Feld
- kann mehrere Objekte des gleichen Typs aufnehmen
- Arrays sind statisch (feste Größe)

Beispiel: Papageienhaus im Zoo



Paco



Rambo



Polly

Java lernen -
Arrays

HPI-Team

45

Anlegen eines Arrays

Anlegen

```
Typ[] variablenName = new Typ[anzahl];  
Typ variablenName[] = new Typ[anzahl];
```

Beispiel

```
1 public class Zoo {  
2     public static void main(String[] args) {  
3         Papagei[] papageienHaus = new Papagei[2];  
4     }  
5 }
```



**Java lernen -
Arrays**

HPI-Team

Zugriff auf Arrays

Array ist ein Container

- Inhalt durchnummeriert (ganzzahlig) = „Index“

Beispiel

- Array, dass Elemente vom Typ String enthält
 - `String[] papageienNamen = new String[4];`
- Erstes Element: Index = 0 (Achtung: Nullbasierter Index)
- Letztes Element: Index = 3

Index	0	1	2	3
Name	"Polly"	"Paco"	"Kikki"	"Rambo"

Zugriff auf Arrays

Zugriff

- `variablenName[index];`

Beispiel

- Array, dass Elemente vom Typ String enthält

Index	0	1	2	3
Name	"Polly"	"Paco"	"Kikki"	"Rambo"

`System.out.println(papageienNamen[0])` Polly

`System.out.println(papageienNamen[1])` Paco

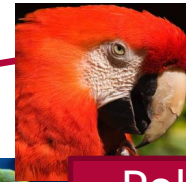
Initialisierung eines Arrays

Initialisierung

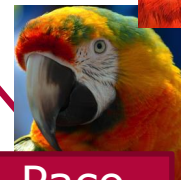
- Bevor Werte zugewiesen werden, muss das Array angelegt werden
- `variablenName[index] = wert;`

Variante 1

```
1 public class Zoo {  
2     public static void main(String args[]){  
3         Papagei[] papageienHaus = new Papagei[2];  
4         papageienHaus[0] = new Papagei("Polly");  
5         papageienHaus[1] = new Papagei("Paco");  
6     }  
7 }
```



Polly



Paco

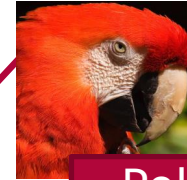
Anlegen & Initialisieren

Deklaration und Initialisierung in einem Schritt

- Alternative Schreibweise
- `Typ[] variablenName = {wert1, wert2, ...};`

Beispiel

```
1 public class Zoo {  
2     public static void main(String args[]){  
3         Papagei polly = new Papagei("Polly");  
4         Papagei paco = new Papagei("Paco");  
5         Papagei[] papageienHaus = {polly, paco};  
6     }  
7 }
```



Polly



Paco



Java lernen -
Arrays

HPI-Team

Operationen auf Arrays

Effektives Verwalten

- Bekanntes Konzept: Schleifen
- Besonders geeignet: for-Schleife
- Attribut length gibt die Größe des Arrays zurück

Beispiel

```
1 public class MeineKlasse {  
2     public static void main(String[] args) {  
3         int[] array = new int[10];  
4         for(int i=0; i < array.length; i++) {  
5             array[i] = i;  
6         }  
7     }  
8 }
```

Operationen auf Arrays

Effektives Verwalten

- Bekanntes Konzept: Schleifen
- Besonders geeignet: for-Schleife
- Attribut length gibt die Größe des Arrays zurück

Beispiel

```
1 public class MeineKlasse {  
2     public static void main(String[] args) {  
3         int[] array = new int[10];  
4         for(int i=0; i < array.length; i++) {  
5             array[i] = i;  
6         }  
7     }  
8 }
```

Java lernen -
Arrays

HPI-Team