

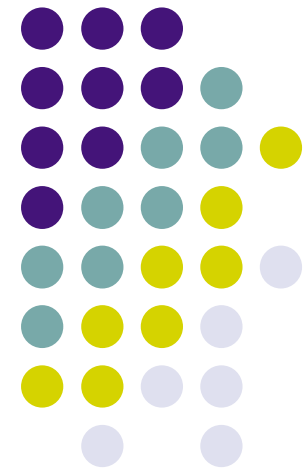
# Einführung in Java

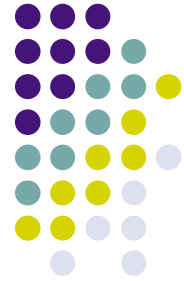
---

PING e.V. Weiterbildung

Andreas Rossbacher

24. März 2005





# Gliederung

1. Was ist Java / Geschichte von Java
2. Prinzip der Plattformunabhängigkeit
3. Wie kommt man vom Quellcode zum Programm
4. Das Prinzip der Objektorientierung
5. Umsetzung in Java
6. Klassische Sprachkonzepte in Java
7. Ein kleines Beispielprogramm

# Was ist Java / Die Geschichte von Java

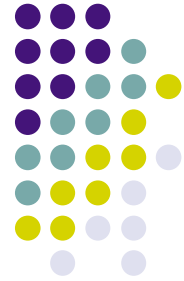


- Java ist eine moderne objektorientierte Programmiersprache
- Java-Programme sind plattformunabhängig
- Java ist streng typenorientiert
- Java ist „von der Hardware gelöst“
  - Es gibt keine Zeiger (Pointer)
  - Kein Destruktor (Garbage collection)

# Was ist Java / Die Geschichte von Java



- 1991 – 1995 von Sun entwickelt (Java 1.0)
- Ziel: Plattformunabhängige Betriebssystemumgebung
- Erster Einsatz in sog. „Applets“ in Webseiten
- Heute eher Web- und Mobile-Applikationen
- Seit 1995 immer wieder erweitert worden (API sowie Sprache)
- Aktuell ist Version 1.5.0 (Java 5) von 2004



# Gliederung

1. Was ist Java / Geschichte von Java
2. **Prinzip der Plattformunabhängigkeit**
3. Wie kommt man vom Quellcode zum Programm
4. Das Prinzip der Objektorientierung
5. Umsetzung in Java
6. Klassische Sprachkonzepte in Java
7. Ein kleines Beispielprogramm

# Prinzip der Plattformunabhängigkeit



- Java-Programme werden kompiliert und dann ausgeführt.
- Dabei folgen sie dem Ansatz: „Compile once run everywhere“
- Java-Programmen steht eine Auswahl an plattformunabhängigen Systemfunktionen zur Verfügung (API – Application Programming Interface)
- Einmal kompilierte Java-Programme laufen auf allen Plattformen mit gleicher oder größerer Java-Version und gleicher oder besserer API-Ausstattung

# Prinzip der Plattformunabhängigkeit



- Java-Umgebung für jede unterstützte Plattform
  1. JRE: Java Runtime Environment (Ausführen)
  2. JDK: Java Development Kit (JRE + Kompilieren)
- Zwei zentrale Komponenten:
  1. Java-Compiler: **javac**
  2. Java-Runtime: **java**



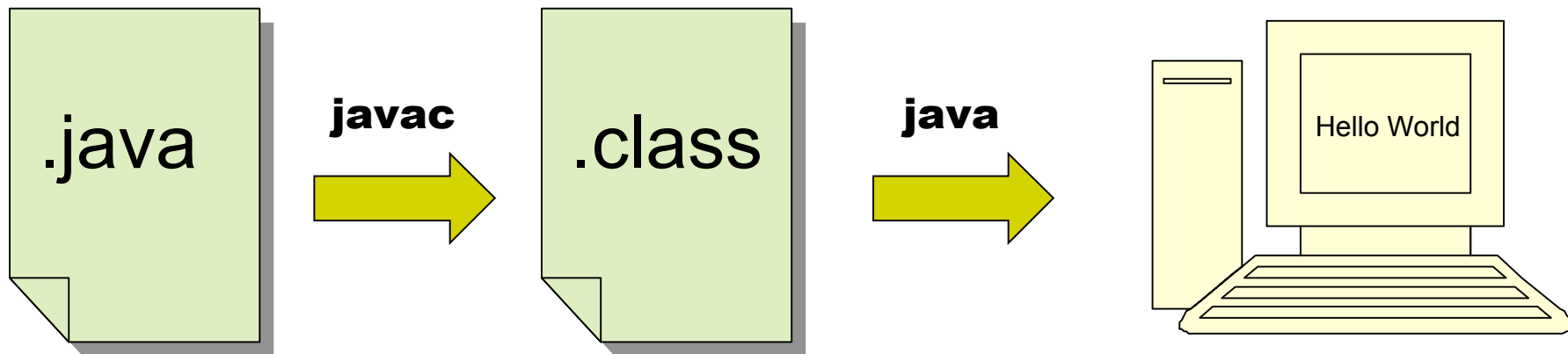
# Gliederung

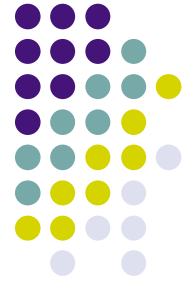
1. Was ist Java / Geschichte von Java
2. Prinzip der Plattformunabhängigkeit
3. **Wie kommt man vom Quellcode zum Programm**
4. Das Prinzip der Objektorientierung
5. Umsetzung in Java
6. Klassische Sprachkonzepte in Java
7. Ein kleines Beispielprogramm

# Wie kommt man vom Quellcode zum Programm?



- Java-Quellcode wird durch den Java-Compiler (**javac**) in eine Java-Klassendatei überführt.





# Gliederung

1. Was ist Java / Geschichte von Java
2. Prinzip der Plattformunabhängigkeit
3. Wie kommt man vom Quellcode zum Programm
4. **Das Prinzip der Objektorientierung**
5. Umsetzung in Java
6. Klassische Sprachkonzepte in Java
7. Ein kleines Beispielprogramm

# Das Prinzip der Objektorientierung



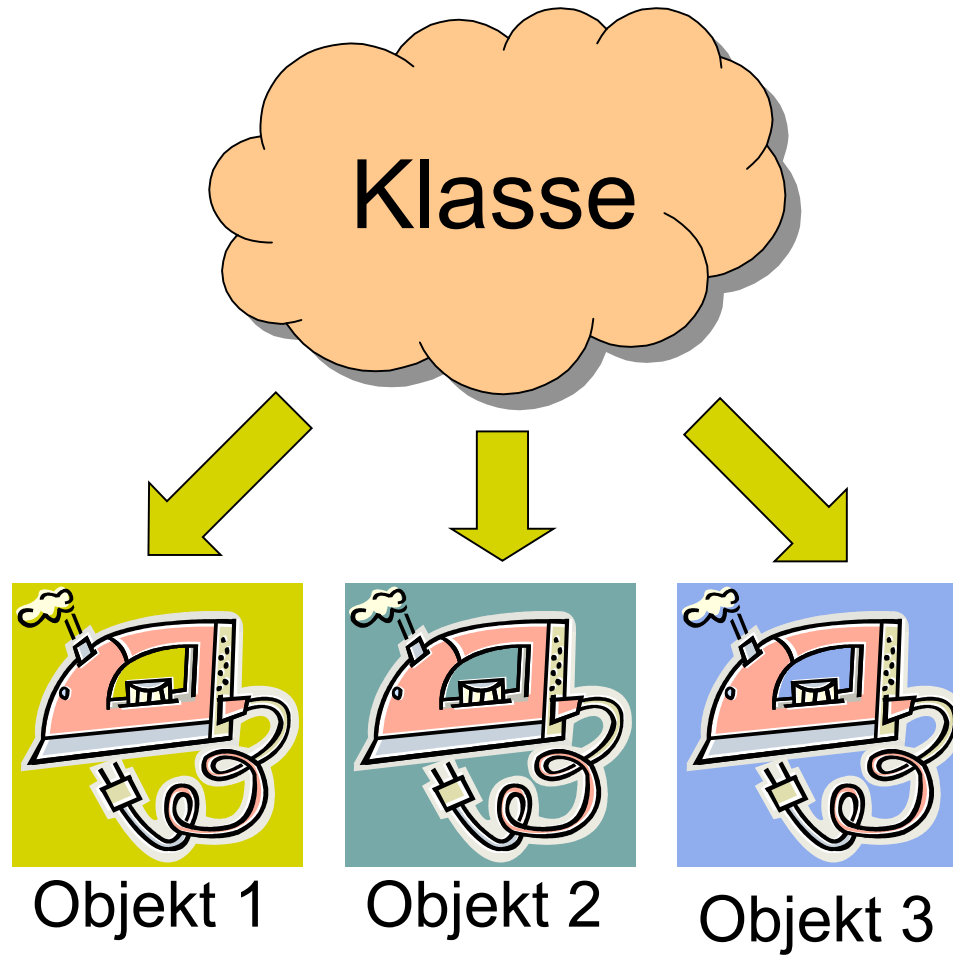
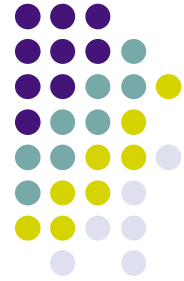
- Klassische Programmiersprachen (BASIC etc.) sind iterativ
- Moderne Programmiersprachen (C++, C#, Ruby) sind objektorientiert
- Programmstruktur besteht nicht direkt aus Abläufen sondern aus Objekten mit Eigenschaften und Methoden
- Programm besteht aus Interaktionen zwischen Objekten

# Das Prinzip der Objektorientierung



- Programmierung von Klassen
- Klassen sind Schablonen aus denen später Objekte erzeugt werden können
- Klassen haben Eigenschaften und Methoden
- Diese geben sie an die Objekte weiter die aus dieser Klasse erstellt werden
- Die Erstellung eines Objektes aus einer Klasse nennt sich instanziiieren

# Das Prinzip der Objektorientierung



# Das Prinzip der Objektorientierung



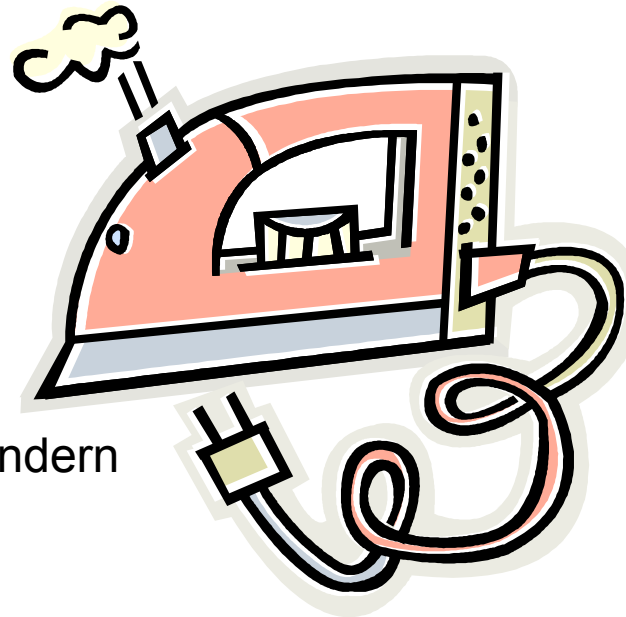
**Bügeleisen** ↔ **Dampf-Bügeleisen**

## Eigenschaften

- Temperatur
- Farbe

## Methoden

- Temperatur verändern



## Eigenschaften

- alle Bügeleisen Eigenschaften
- Dampfstufe

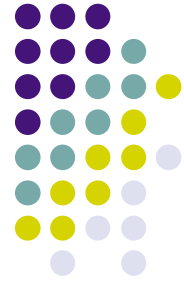
## Methoden

- alle Bügeleisen Methoden
- Dampfstufe verstellen

# Das Prinzip der Objektorientierung

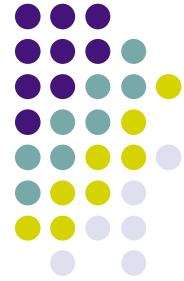


- Klassen können Eigenschaften vererben
- Diese erbende Klasse hat dann eigenen Eigenschaften und Methoden und die Geerbten
- Eine erbende Klasse ist immer spezieller als die von der sie erbt



# Gliederung

1. Was ist Java / Geschichte von Java
2. Prinzip der Plattformunabhängigkeit
3. Wie kommt man vom Quellcode zum Programm
4. Das Prinzip der Objektorientierung
5. **Umsetzung in Java**
6. Klassische Sprachkonzepte in Java
7. Ein kleines Beispielprogramm



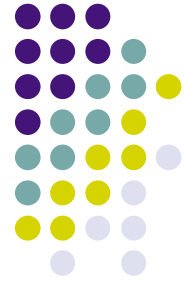
# Umsetzung in Java

## 1. Klasse erstellen:

```
public class Buegeleisen  
{  
}
```

## 2. Eigenschaften einfügen

```
public class Buegeleisen  
{  
    // Farbe des Bügeleisens  
    private String farbe;  
    // Temperatur des Bügeleisens  
    private int temperatur;  
}
```

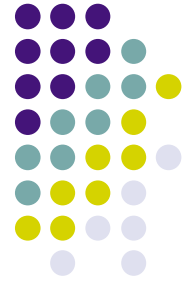


# Umsetzung in Java

## 3. Konstruktor einfügen

```
public class Buegeleisen
{
    // Farbe des Bügeleisens
    private String farbe;
    // Temperatur des Bügeleisens
    private int temperatur;

    public Buegeleisen(String _farbe)
    {
        farbe = _farbe;
        temperatur = 0;
    }
}
```



# Umsetzung in Java

## 4. Methoden einfügen

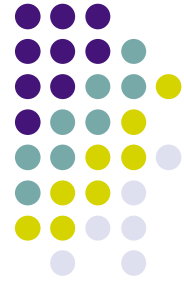
```
public class Buegeleisen
{
    // Farbe des Bügeleisens
    private String farbe;
    // Temperatur des Bügeleisens
    private int temperatur;

    public Buegeleisen(String _farbe)
    {
        farbe = _farbe;
        temperatur = 0;
    }
}
```

```
public void setTemp(int _temperatur)
{
    temperatur = _temperatur;
}

public int getTemp()
{
    return (temperatur);
}

public String getFarbe()
{
    return (farbe);
}
}
```

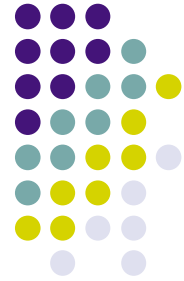


# Umsetzung in Java

## 5. Dampfbügeleisen erbt von Bügeleisen

```
public class Dampfbuegeleisen extends Buegeleisen
{
    private byte dampfstufe;

    public Dampfbuegeleisen(String _farbe)
    {
        super(_farbe);
        dampfstufe = 0;
    }
}
```



# Umsetzung in Java

## 6. Dampfbügeleisen hat zwei zusätzliche Methoden

```
public class Dampfbuegeleisen extends Buegeleisen
{
    private byte dampfstufe;

    public Dampfbuegeleisen(String _farbe)
    {
        super(_farbe);
        dampfstufe = 0;
    }
}
```

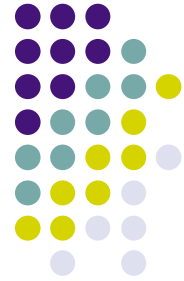


# Umsetzung in Java

## 6. Dampfbügeleisen hat zwei zusätzl. Methoden

```
public void setDampfstufe(byte _dampfstufe)
{
    dampfstufe = _dampfstufe;
}

public byte getDampfstufe()
{
    return(dampfstufe);
}
}
```



# Gliederung

1. Was ist Java / Geschichte von Java
2. Prinzip der Plattformunabhängigkeit
3. Wie kommt man vom Quellcode zum Programm
4. Das Prinzip der Objektorientierung
5. Umsetzung in Java
6. **Klassische Sprachkonzepte in Java**
7. Ein kleines Beispielprogramm

# Klassische Sprachkonzepte in Java



Typ	Inhalt	Standard	Größe	Min	Max
boolean	true oder false	false	1 Bit	-	-
char	Unicode-Zeichen	\u0000	16 Bit	\u0000	\uffff
byte	Integer + Vorzeichen	0	8 Bit	-128	127
short	Integer + Vorzeichen	0	16 Bit	-32768	32767
int	Integer + Vorzeichen	0	32 Bit	-2147483648	2147483647
long	Integer + Vorzeichen	0	64 Bit	- 9223372036854 77808	922337203685477 5807
float	Fließkommazahl	0.0	32 Bit	+ -1.4023984eE- 45	+ -3.40282347E+38
double	Fließkommazahl	0.0	64 Bit	+ - 4.940656458412 46544E-324	+ - 1.79769313486231 570E+308

# Klassische Sprachkonzepte in Java



- if/else:

```
if (1 > zahl)
{
    zahl++;
}
else
{
    zahl--;
}
```

- `zahl++` wird nur ausgeführt, wenn die Anweisung in den runden Klammern wahr ist.
- In diesem Fall gibt es ein `else`, also `zahl--`, wenn die Anweisung in der runden Klammer falsch ist

# Klassische Sprachkonzepte in Java



- do/while:

```
while (1 > zahl)
{
    zahl++;
}
```

```
do {
    zahl--;
}
while (1 > zahl);
```

- Im ersten Fall findet eine Prüfung der Bedingung vor dem (ersten) Durchlauf statt.
- Im zweiten Fall findet eine Prüfung der Bedingung erst nach dem (ersten) Durchlauf statt.

# Klassische Sprachkonzepte in Java



- Switch:

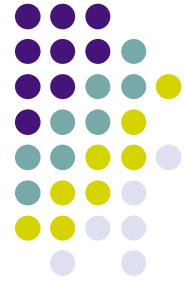
```
switch (zahl)
{
  1: methode1(); break;
  2: methode2();
  ...
}
```

- `zahl` darf nur vom Typ *byte*, *char*, *short*, *int* oder *long* sein
- Nach einem `break` wird der switch-Block abgebrochen

- For-Schleife

```
for (int i=0;i<y;i++)
{
  y--;
}
```

- `y--` wird so lange ausgeführt, bis `i` grösser als `y` ist



# Gliederung

1. Was ist Java / Geschichte von Java
2. Prinzip der Plattformunabhängigkeit
3. Wie kommt man vom Quellcode zum Programm
4. Das Prinzip der Objektorientierung
5. Umsetzung in Java
6. Klassische Sprachkonzepte in Java
7. **Ein kleines Beispielprogramm**

# Ein kleines Beispielprogramm



- Wir nehmen unser Beispielprogramm „Bügeleisen“ und erweitern es ein wenig.
- Folgendes ist hierfür noch wichtig:
  - `System.out.println(„text“);` gibt Text auf der Konsole aus
  - `public static void main (String[] args) {}` ist die Methode in der das Programm „startet“



# Quellen

- [http://de.wikipedia.org/wiki/Java\\_%28Programmiersprache%29](http://de.wikipedia.org/wiki/Java_%28Programmiersprache%29)
- Java in a nutshell 2. Auflage – David Flanagan – O´REILLY 1998