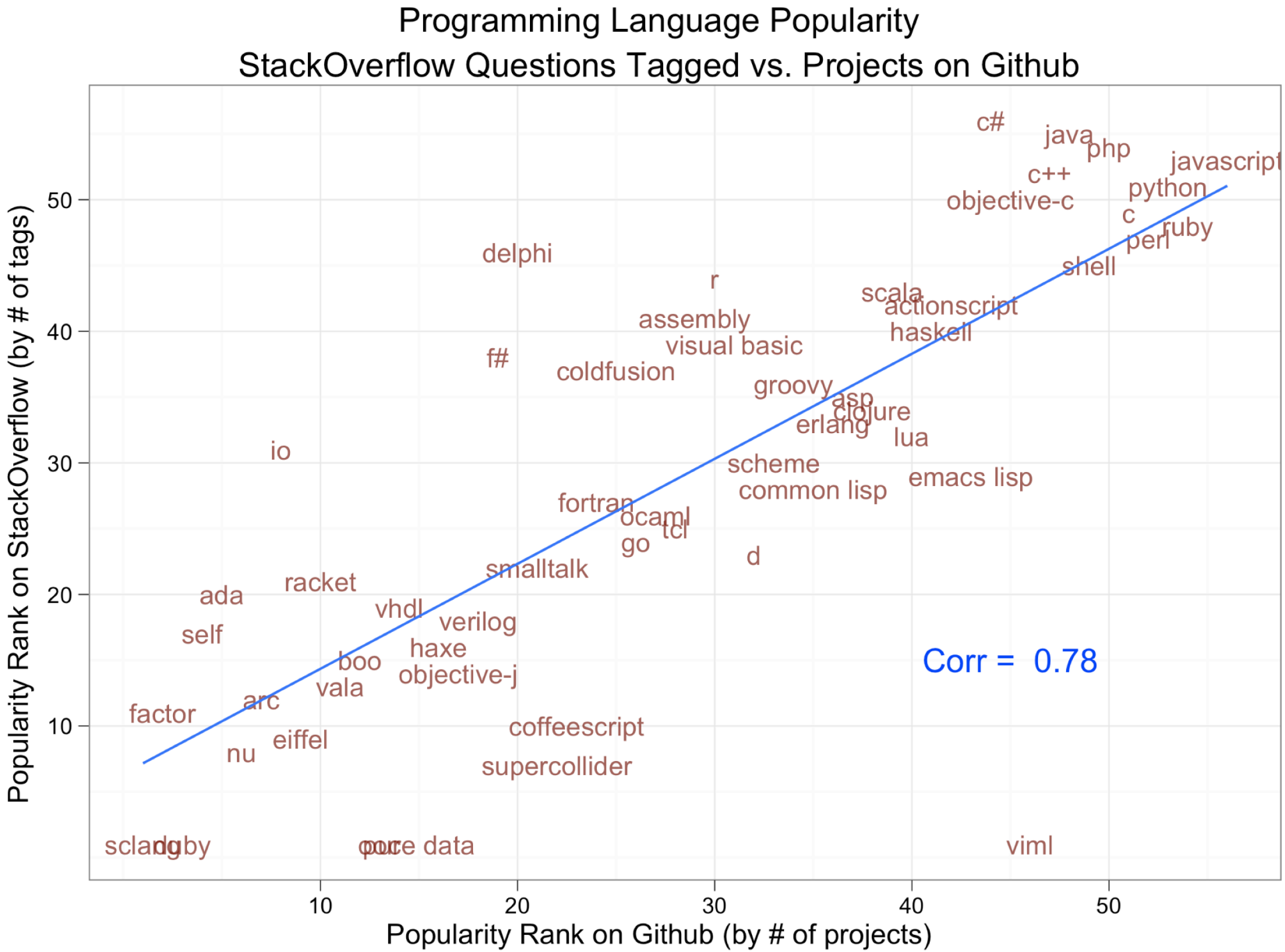
Hello World-Programme

- „Unsere“ Programmiersprache
- Vereinbarungen und Anweisungen
- Hello World
- Einlesen von Strings
- Variable
- Fahrenheit <-> Celsius
- Syntaktischer Zucker

Nutzung von Programmiersprachen



URL: www.dataists.com/2010/12/ranking-the-popularity-of-programming-langauges/
Abgerufen: 9/2017

Die Entscheidung

Wir benutzen Java, weil es

- sehr viel benutzt wird / sehr weit verbreitet ist
- auf verschiedener Hardware / verschiedenen Betriebssystemen „identisch“ läuft
- sehr strenge Prüfungen auf „Korrektheit des Programms“ ***vor*** dessen Ausführung durchführt

Was kann „Korrektheit“ bedeuten?

1. Das Programm enthält keine Tippfehler
2. Das Programm enthält keine Bedeutungsfehler; Gegenbeispiele:

   3 > Hallo
   x / 0
3. Das Programm „tut seine Aufgabe richtig“; Beispiele
   - Spam-E-Mails automatisch erkennen und löschen
   - Alle Kontakte mit dem Suchnamen „Meier“ anzeigen
4. Das Programm „tut das Richtige“ (d.h. das, was der Anwender möchte); Beispiele
   - Spam-E-Mails automatisch erkennen und löschen
     <-> Spam-E-Mails besonders markieren, aber nicht löschen
   - Alle Kontakte mit dem Suchnamen „Meier“ anzeigen
     <-> nur den ersten Kontakt mit Namen „Meier“ anzeigen

Korrektheit 2/3

- Zu 1: keine Tippfehler
  - Fachbegriff: ***Syntax***-Fehler
  - Engl. syntax = die Schreibweise, der Satzbau
  - Das leisten sehr viele Programmiersprachen
    - → Positiv-Beispiele: Java, C#, C++, ...
    - → Negativ-Beispiele: Javascript, ...
- Zu 2: keine Bedeutungsfehler
  - Fachbegriff: ***Semantik***-Fehler
  - Engl. semantics = Bedeutungslehre
  - Programmiersprachen verhalten sich sehr unterschiedlich
    - → Quasi keine Prüfungen: Maschinensprache, Assembler
    - → Wenige Prüfungen: C, C++
    - → Sehr streng: Java, C#
  - Konsequenz weniger Prüfungen
    - → Viele Semantik-Fehler werden erst bei der Ausführung eines Programms entdeckt
    - → Die Behebung von Fehlern ist teuer, schädlich fürs Image, ...

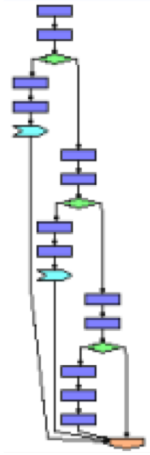
Korrektheit 3/3

- Zu 3: tut seine Aufgabe richtig
  - Solche Fehler sind festzustellen durch
    - → (Aufwändige) Programmanalyse
    - → Tests
  - Beides ist aufwändig, daher dauern gründliche Überprüfungen lange und sind teuer

- Zu 4: tut das Richtige
  - Solche Fehler sind nur durch sehr sorgfältige / umfangreiche Tests bzw. normale Ausführung durch den Anwender/Kunden zu entdecken

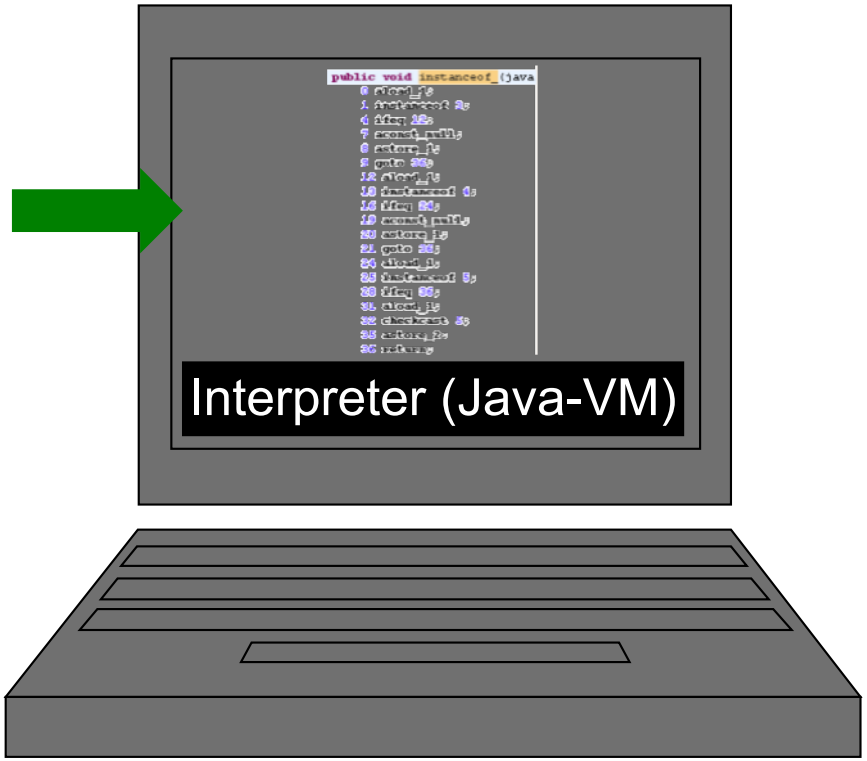
Ausführung von Java-Programmen 2/2

Java-Programm (höhere Programmiersprache)



Compiler

Java-Bytecode



Solche Programmiersprachen müssen (alle) sehr genau definiert sein; es dürfen keinerlei Missverständnisse möglich sein

Java-Programme bestehen aus zwei Teilen

(Java-)Programme bestehen aus zwei Teilen

- Vereinbarungen
- Anweisungen

Anweisungen enthalten wiederum Ausdrücke

Warum benötigen wir „Anweisungen“?

→ Um dem Computer mitzuteilen, was er zu tun hat

Beispiel „Pizzarezept“

Pizza und Pizzaiola

(4 Personen)

Zutaten: Pizzateig

400g	Mehl
1/2 Würfel	frische Hefe
0,2 l	lauwarmes Wasser
2 El.	Olivenöl
	Salz
	Butter für das Backblech

Für den Belag

Salami, Schinken, Thunfisch, Zwiebelringe
Champignons, gelbe und rote Paprika dünsten,
Mozarella, Gouda oder Edamer reiben
Oliven, Pfeffer, Oregano, Basilikum, Majoran

Zutaten: Pizzaiola

1 Dose	geschälte Tomaten
1	große Zwiebel
2	Knoblauchzehen
3 El.	Olivenöl
2 El.	Tomatenmark
2 Tl.	Oregano (getrocknet)
1 Tl.	Basilikum (getrocknet)
1	Lorbeerblatt
1 Tl.	Zucker
1 Tl.	Salz
	gemahlener Pfeffer
	Chilipaste

Vereinbarungen

Zubereitung: Pizzaiola

Zwiebel und *Knoblauch* schälen, sehr fein hacken.
In einem Topf das *Öl* erhitzen. *Zwiebeln* und *Knoblauch* darin bei mittlerer Hitze unter Rühren in etwa 8 Minuten weich und glasig dünsten.
Die *Tomaten* mit einer Gabel zerdrücken und mitsamt dem Saft dazugeben. *Tomatenmark*, *Oregano*, *Basilikum*, *Lorbeerblatt*, *Zucker*, *Salz*, *Chilipaste* und etwas *Pfeffer* in den Topf geben. Die Soße aufkochen lassen. Die Hitze reduzieren und die Soße unter gelegentlichem Rühren bei mittlerer Hitze etwa 30 Minuten leicht köcheln lassen.
Das *Lorbeerblatt* herausnehmen. Nochmal mit *Salz* und *Pfeffer* abschmecken.

Zubereitung: Pizzateig

Das *Mehl* in eine Schüssel geben. In die Mitte eine Vertiefung drücken. Dahinein die *Hefe* bröckeln. Mit etwas lauwarmem *Wasser* und etwas *Mehl* zu Brei rühren. An einem warmen Ort gehen lassen, bis die *Hefe* Blasen wirft.
Öl und *Salz* dazugeben, mit den Knethaken des Handrührgeräts vermischen. Nach und nach das restliche *Wasser* zufügen. Den *Teig* mit einem Tuch abdecken, nochmals 1 Stunde gehen lassen. Auf einem *gefetteten* Backblech ausrollen, mit der *Soße* bestreichen und mit den *Zutaten* belegen.

Anweisungen (benutzen die Vereinbarungen)

Zutaten: Pizzateig

400g	Mehl
½ Würfel	Hefe
200ml	lauwarmes Wasser
2EL	Olivenöl
	Salz

Anweisungen benutzen die Vereinbarungen

Lauwarm!

Wie viel?

Wie viel?

Wie viel?

Lauwarm!

- Das Mehl in eine Schüssel geben. In die Mitte eine Vertiefung drücken. Dahinein die Hefe bröckeln. Mit etwas Wasser und etwas Mehl zu Brei rühren. An einem warmen Ort gehen lassen, bis die Hefe Blasen wirft.
- Öl und Salz dazugeben, mit den Knethaken des Handrührgeräts vermischen. Nach und nach das restliche Wasser zufügen.
- Den Teig mit einem Tuch abdecken, nochmals 1 Stunde gehen lassen.

Die Fehlerarten am Beispiel

Zutaten: Pizzateig

	Mehl
½ Würfel	Hefe
200ml	lauwarmes Wasser
	Salz

Semantik: Mengenangabe fehlt in der Vereinbarung

Syntax: Tippfehler

Semantik: Menge passt nicht zur Vereinbarung

Syntax: Grammatik

- Das Mehl in eine Schüsel geben. In die Mitte eine Vertiefung drücken. Dahinein den Hefewürfel bröckeln. Mit etwas Wasser und etwas Mehl zu Brei rühren. An eine warmen Ort gehen lassen, bis er doppelt so groß ist.
- Öl und Salz dazugeben, mit den Knethaken des Handrührgeräts vermischen. Nach und nach das restliche Wasser zufügen.
- Den Teig mit einem Tuch abdecken, nochmals 1 Stunde gehen lassen.

Semantik: Vereinbarung fehlt

tut es richtig: Blasen sind entscheidend

tut das Richtige: der Kunde wollte Lasagne (keine Pizza)!

HOCHSCHULE DER WIRTSCHAFT FÜR MANAGEMENT
UNIVERSITY OF APPLIED MANAGEMENT STUDIES

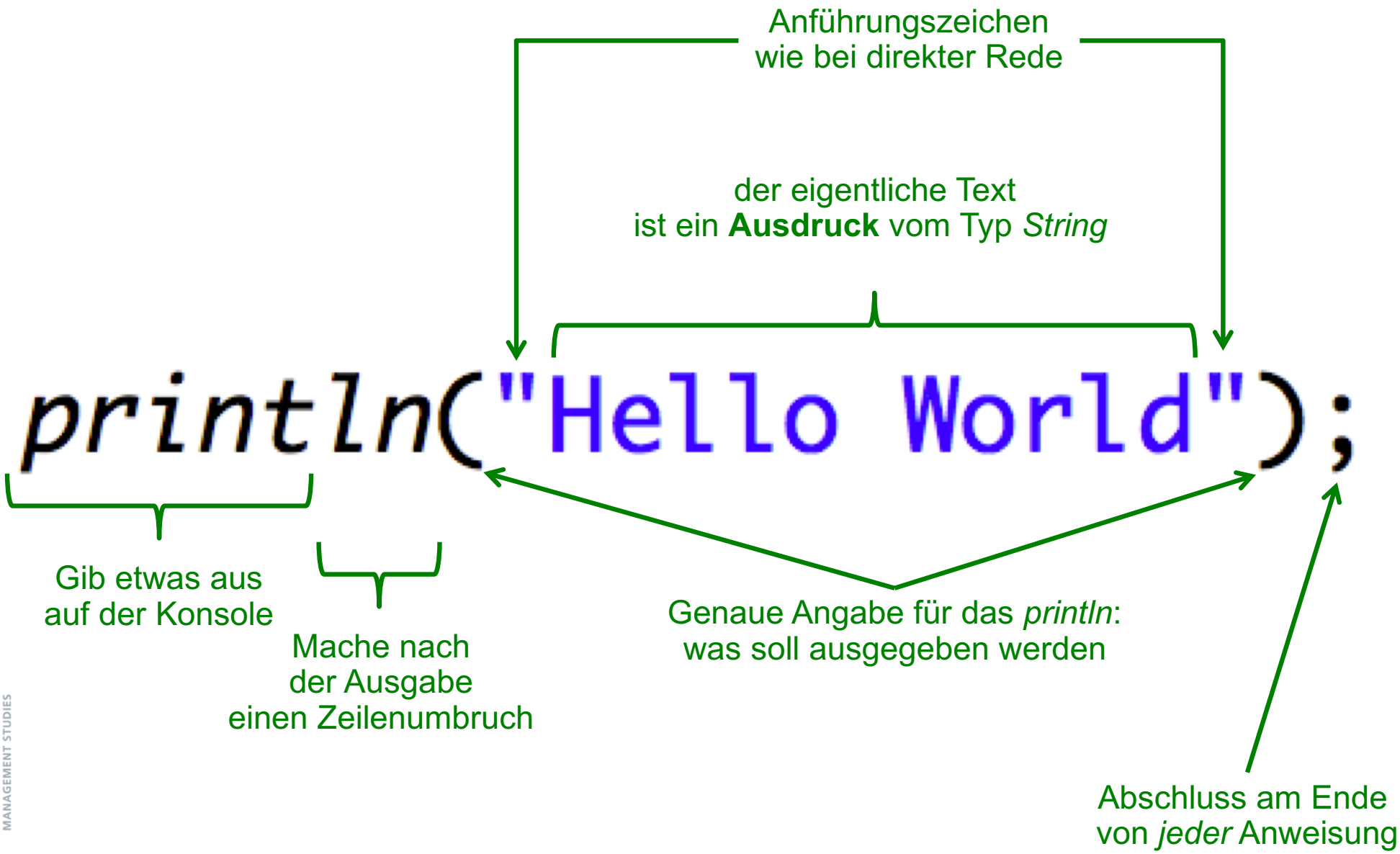
Wir werden lernen,

- welche Anweisungen es in Java gibt,
- welche Vereinbarungen es in Java gibt und
- wie man sie einsetzt

Wir machen es uns einfacher:

- Wir behandeln nur einen (notwendigen) Teil von Java
  -> nicht Programmieren-Können ist Ziel der Vorlesung
- Wir benutzen eine Hilfsbibliothek für schwierige Dinge

Das erste (traditionelle) Java-Programm: eine einzelne Anweisung



HOCHSCHULE DER WIRTSCHAFT FÜR MANAGEMENT
UNIVERSITY OF APPLIED MANAGEMENT STUDIES

Ein anderer „Ein-Zeiler“

```
println("Guten Tag!");
```

Außer dem eigentlichen Text hat sich nichts verändert

Alle „Formalitäten“ (d.h. die Syntax) bleiben gleich

Einlesen (und Ausgeben) eines Textes

Das Plus-Zeichen verbindet den linken *String* mit dem rechten *String*

Einlesen einer (Text-)Zeile von der Konsole (Abschluss mit <return>). *readString()* liefert einen *String*

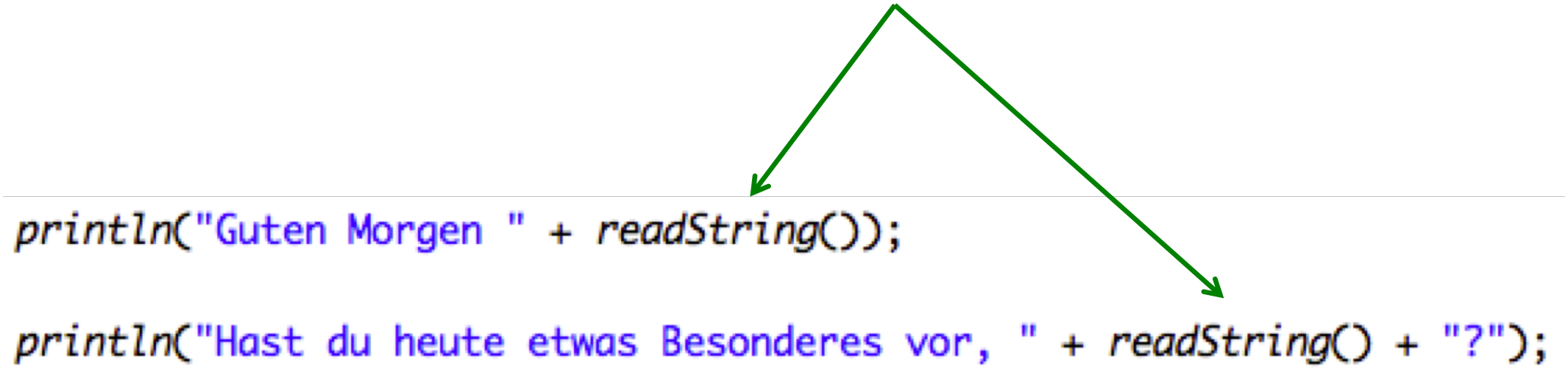
Das kennen wir alles...

...schon

Für *readString* sind keine weiteren Angaben nötig; die Klammern müssen trotzdem stehen!

Das zweite Java-Programm: Variable 1/2

Sehr unschön:
Der Name wird zwei Mal abgefragt und muss zwei Mal eingegeben werden



```
println("Guten Morgen " + readString());

println("Hast du heute etwas Besonderes vor, " + readString() + "?");
```

Besser:
Wir merken uns die Eingabe!

Das zweite Java-Programm: Variable 2/2

Das einfache Gleichheitszeichen bewirkt keinen Vergleich,
sondern **eine Zuweisung** (wie in vielen Programmiersprachen):
Der eingelesene *String* wird einer **Variablen** namens *firstName* zugewiesen

Nach der Zuweisung ist der Text beliebig oft
unter dem Variablennamen verfügbar

```
firstName = readString();

println("Guten Morgen " + firstName);

println("Hast du heute etwas Besonderes vor, " + firstName + "?");
```

Aber Java ist streng:
Alle **Namen** müssen **vor ihrer Verwendung** (in Anweisungen)
vereinbart werden!

Das zweite Java-Programm: Variable 2/2

Variablen-Vereinbarung
(immer mit Angabe eines Datentyps)

Bei der Zuweisung prüft Java,
ob der **Typ des Ausdrucks** (*readString()* liefert einen *String*)
zum vereinbarten **Typ der Variablen** (*firstName*) passt

```
String firstName;

firstName = readString();

println("Guten Morgen " + firstName);

println("Hast du heute etwas Besonderes vor, " + firstName + "?");
```

Aber Java ist streng:
Alle **Namen** müssen **vor ihrer Verwendung** (in Anweisungen)
vereinbart werden!

3. Java-Programm: Temperaturumrechnung Grad Fahrenheit -> Grad Celsius

$$°C = (°F - 32) * 5/9$$

Eingabeaufforderung ohne Zeilenumbruch (*print* statt *println*)

Die Variable *gradFahrenheit* muss mit dem korrekten Datentyp *int* vereinbart werden bevor sie verwendet werden kann

```
print("Geben Sie die Temperatur in Grad Fahrenheit ein: ");
int gradFahrenheit;
gradFahrenheit = readInt();

print("Das entspricht in Grad Celsius: ");
println(gradFahrenheit - 32 * 5 / 9);
```

Lesen einer **ganzen Zahl** von der Konsole; der zugehörige Datentyp heißt ***int*** nach dem engl. integer = ganzzahlig

Ausgabe eines Ausdrucks: Der Wert (vom Typ *int*) wird komplett berechnet und dann ausgegeben

3. Java-Programm: Temperaturumrechnung Grad Fahrenheit -> Grad Celsius

$$°C = (°F - 32) * 5/9$$

Eingabeaufforderung ohne Zeilenumbruch (*print* statt *println*)

Die Variable *gradFahrenheit* muss mit dem korrekten Datentyp *int* vereinbart werden bevor sie verwendet werden kann

```
print("Geben Sie die Temperatur in Grad Fahrenheit ein: ");
int gradFahrenheit;
gradFahrenheit = readInt();

print("Das entspricht in Grad Celsius: ");
println((gradFahrenheit - 32) * 5 / 9);
```

Lesen einer **ganzen Zahl** von der Konsole; der zugehörige Datentyp heißt ***int*** nach dem engl. integer = ganzzahlig

Version mit korrekter Klammerung

Ausgabe eines Ausdrucks: Der Wert (vom Typ *int*) wird komplett berechnet und dann ausgegeben

Temperaturumrechnung
Grad Celsius -> Grad Fahrenheit

$$°F = (°C * 9/5) + 32$$

```
print("Geben Sie die Temperatur in Grad Celsius ein: ");
int gradCelsius;
gradCelsius = readInt();

print("Das entspricht in Grad Fahrenheit: ");
println(gradCelsius * 1.8 + 32);
```

Java verwendet einen **Dezimalpunkt** für Gleitkommazahlen anstelle eines Dezimal***kommas***

Das vollständige Programm inklusive syntaktischem Zucker

```
package helloWorld;

import static pr.MakeItSimple.*;

public class FahrenheitToCelsius {

    public static void main(String[] args) {

        print("Geben Sie die Temperatur in Grad Celsius ein: ");
        int gradCelsius;
        gradCelsius = readInt();

        print("Das entspricht in Grad Fahrenheit: ");
        println(gradCelsius * 1.8 + 32);

    }

}
```

Name der Klasse / des Programms

F R A G E N