

# 1.2 Erste Schritte mit Java

- Vorerst nur Entwicklung im REPL-Interface
  - Erste Schritte
  - Variablen
    - Deklaration
    - Zuweisung

# Programmentwicklung

## 1. Klassische Entwicklungswerkzeuge

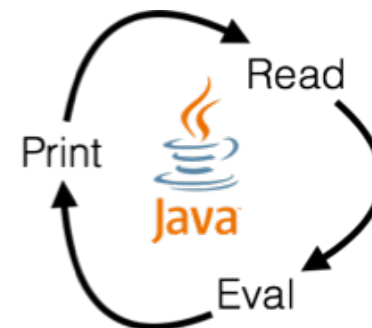
- Texteditor
- Übersetzer

## 2. Heute: Integrierte Entwicklungsumgebungen

- Anlegen von Projekten
- Zusätzlich werden Werkzeuge angeboten, um
  - Programme zu testen
  - Fehler zu finden
  - Programmentwicklung im Team

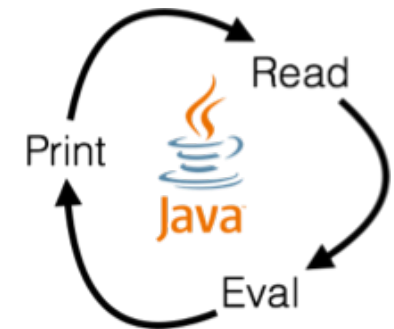


## 3. REPL-Interface (seit Java 9)



# REPL-Schnittstelle

- Neue Möglichkeit um Java-Befehle direkt auszuführen
  - REPL = Read–Eval–Print Loop
- Vorgehensweise (Windows)
  - Kommandozeile cmd.exe aufrufen
  - Befehl jshell eingeben



```
Eingabeaufforderung - jshell
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\bhseeger>jshell
| Welcome to JShell -- Version 9
| For an introduction type: /help intro

jshell> ■
```

# Erste Befehle mit REPL

- Im Wesentlichen funktioniert alles wie bei einem Taschenrechner.
  - Die Ausgabe eines Ergebnisses ist leider etwas komplizierter.
- Beispiele
  1. `System.out.print("Hallo Welt!")`  
Gibt den Text zwischen den beiden Anführungszeichen aus.
  2. `System.out.print(7)`  
Gibt die ganze Zahl 7 aus.
  3. `System.out.print(3.14)`  
Gibt die Gleitpunktzahl 3.14 aus.
  4. `System.out.print('a')`  
Gibt das Zeichen a aus.

# Erste Berechnungen

- Ähnlich zum Taschenrechner können im REPL Berechnungsschritte angegeben und ausgeführt werden.
- Beispiele
  1. `System.out.print("Hallo" + " Hallo")`  
Verknüpfung von zwei Texten und Ausgabe des Ergebnis.
  2. `System.out.print(7 + 8)`  
Addiert  $7 + 8$  und gibt die ganze Zahl 15 aus.
  3. `System.out.print(3.14*2*2)`  
Multipliziert die 3 Zahlen und gibt das Ergebnis aus.

# Speichern von Zwischenergebnissen (1)

- Analog zum Taschenrechner kann man sich Werte im Zwischenspeicher merken.
  - Zwischenspeicher im REPL ist nahezu beliebig groß!
- Beispiele
  1. "Hallo Welt!"



Name des  
Zwischenspeichers

```
Eingabeaufforderung - jshell

C:\Users\bhseeger>jshell
| Welcome to JShell -- Version 9
| For an introduction type: /help intro

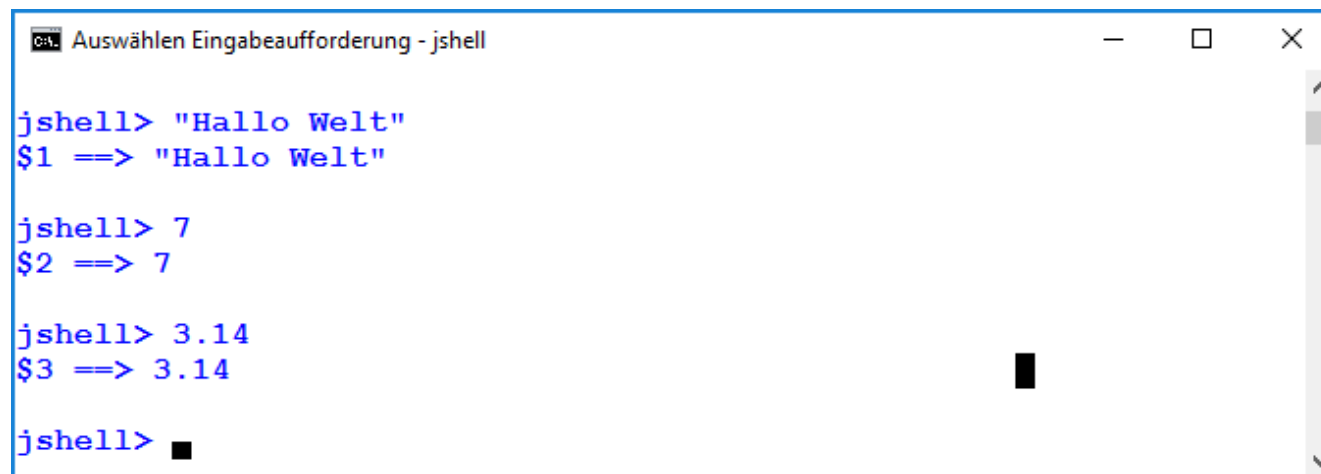
jshell> "Hallo Welt!"
$1 => "Hallo Welt!"

jshell> System.out.print($1)
Hallo Welt!
jshell> ■
```

Wiederverwendung des Werts  
aus dem Zwischenspeicher.

# Speichern von Zwischenergebnissen (2)

- Der Namen der Zwischenspeicher sind eindeutig.
  - Ein Name wird nur einmal vergeben.



```
ct: Auswählen Eingabeaufforderung - jshell

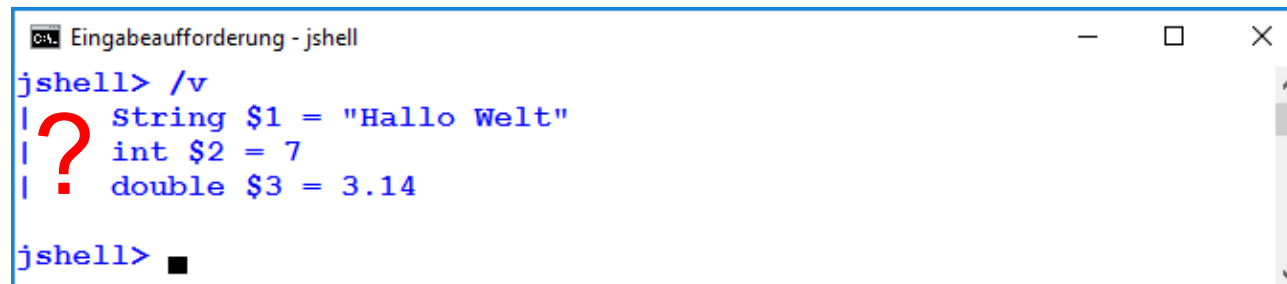
jshell> "Hallo Welt"
$1 ==> "Hallo Welt"

jshell> 7
$2 ==> 7

jshell> 3.14
$3 ==> 3.14

jshell> █
```

- Die Liste aller gespeicherten Werte kann man durch den Befehl **/vars** oder kurz **/v** bekommen.



```
ct: Eingabeaufforderung - jshell

jshell> /v
| String $1 = "Hallo Welt"
| ? int $2 = 7
| ! double $3 = 3.14

jshell> █
```

# Datentypen

- Jeder Zwischenspeicher merkt sich noch zusätzlich aus welcher Wertemenge ein Wert ist.
  - "Hallo Welt!" hat den **Typ String** und repräsentiert die Menge aller Zeichenketten.
  - 7 hat den **Typ int** und repräsentiert die Menge der ganzen Zahlen in dem Bereich -2147483648 bis 2147483647
  - 3.14 ist vom **Typ double** und repräsentiert die Menge der Gleitpunktzahlen
- Zusätzlich zu der Wertmenge besitzt ein Datentyp

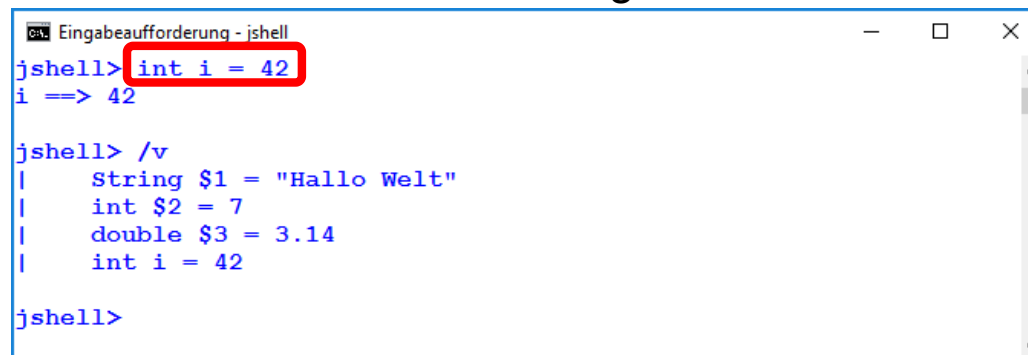
## Operationen

- Datentyp int
  - + , - , \* , / und %
- Datentyp String
  - +



# Variablen

- Statt nun einfach REPL zu überlassen, wie Speicherplätze benannt werden, kann man dies auch selbst tun.
- Man spricht dann von einer **Variable**.
  - Bei der **Deklaration der Variable** muss immer der **Datentyp** angegeben werden.
    - Jede Variable darf nur einmal deklariert werden.
  - Zudem ist es empfehlenswert der Variablen einen Wert zuzuweisen.
    - Hierzu benutzt man den **Zuweisungsoperator** (Symbol =). Dies bedeutet, dass die Variable den Wert rechts von = zugewiesen bekommt.



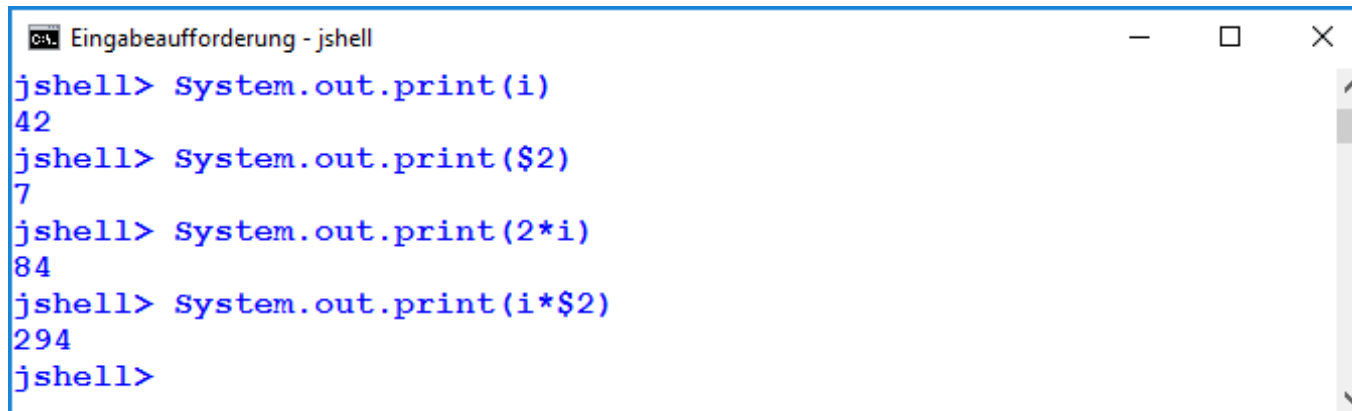
```
Eingabeaufforderung - jshell
jshell> int i = 42
i ==> 42

jshell> /v
|   String $1 = "Hallo Welt"
|   int $2 = 7
|   double $3 = 3.14
|   int i = 42

jshell>
```

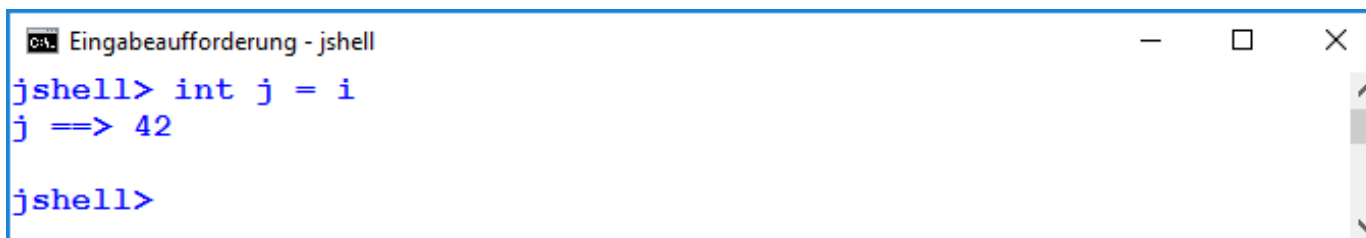
# Verwendung von Variablen

- Variablen können immer dort stehen, wo auch der Wert der Variablen stehen darf.



```
Eingabeaufforderung - jshell
jshell> System.out.print(i)
42
jshell> System.out.print($2)
7
jshell> System.out.print(2*i)
84
jshell> System.out.print(i*$2)
294
jshell>
```

- Man kann einer Variablen den Wert einer anderen Variablen zuweisen.

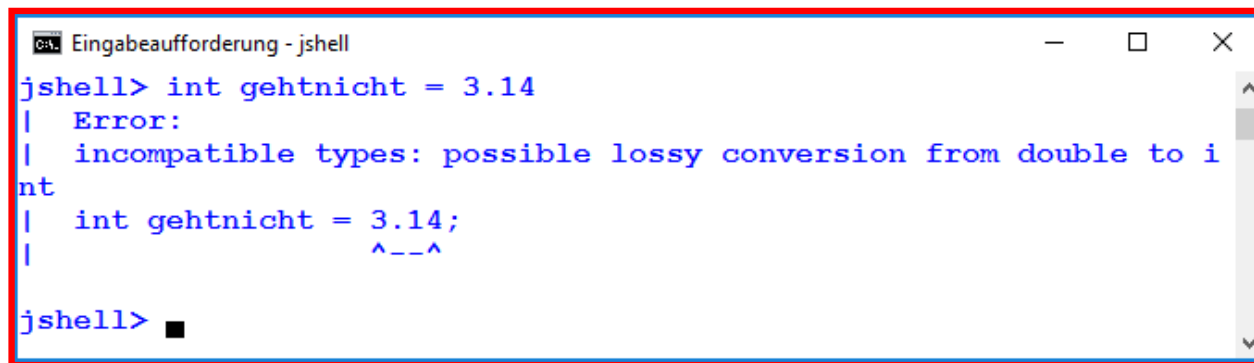


```
Eingabeaufforderung - jshell
jshell> int j = i
j ==> 42
jshell>
```

- Der Wert von i wird aus dem Speicher gelesen und dann in der Variable j gespeichert.

# Dies ist jedoch zu beachten!

- Einer Variablen  $v$  kann nur ein Wert zugewiesen werden, wenn der Datentyp von  $v$  auch den Wert enthält.

A screenshot of a Java REPL window titled "Eingabeaufforderung - jshell". The window shows the following text:

```
jshell> int gehtnicht = 3.14
|   Error:
|   incompatible types: possible lossy conversion from double to i
nt
|   int gehtnicht = 3.14;
|                       ^__^
jshell> █
```

The error message indicates that the variable 'gehtnicht' is declared as an 'int' but is being assigned a 'double' value '3.14', which is a lossy conversion.

- REPL liefert dann eine Fehlermeldung (Error) mit einer Begründung.
  - Der Datentyp `int` enthält nur ganze Werte. Eine Zuweisung von einer Gleitpunktzahl ist nicht möglich.



# Überschreiben von Variablen

- Prinzipiell ist es möglich, bereits deklarierten Variablen einen neuen Wert zuzuweisen.
  - Hierfür verwendet man wieder den Zuweisungsoperator.
  - Der alte Wert ist dann nicht mehr verfügbar.

```
Eingabeaufforderung - jshell

jshell> /v
|   String $1 = "Hallo Welt!"
|   int $2 = 7
|   int i = 42

jshell> i = 23
i ==> 23

jshell> $1 = "Hallo Uni Marburg!"
$1 ==> "Hallo Uni Marburg!"

jshell> /v
|   String $1 = "Hallo Uni Marburg!"
|   int $2 = 7
|   int i = 23

jshell> █
```

## ... und übrigens

- Verlassen von REPL mit
  - `/exit`
- Abspeichern der bisherigen Befehle mit
  - `/save` meineREPLDatei
- Einlesen der Befehle aus der Datei „meineREPLDatei“ beim Starten von REPL.
  - `jshell` meineREPLDatei

# Zusammenfassung

- Erste Schritte mit Java unter Verwendung von REPL
- Wichtige Konzepte
  - Variablen
  - Datentypen
    - Wertemengen
    - Operationen
  - Zuweisungsoperator