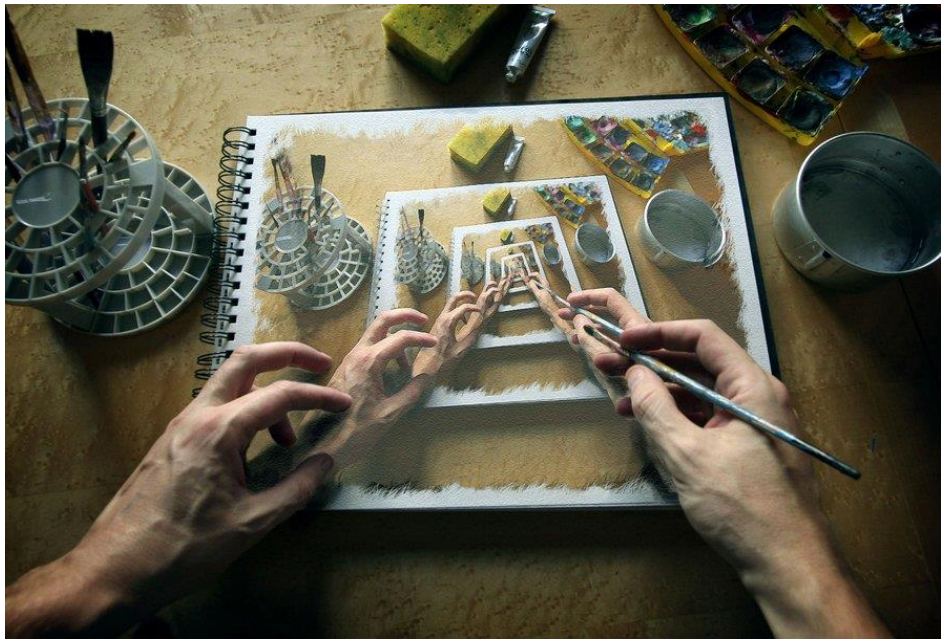
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Programmiervorkurs
Wintersemester 20/21

Tag 6

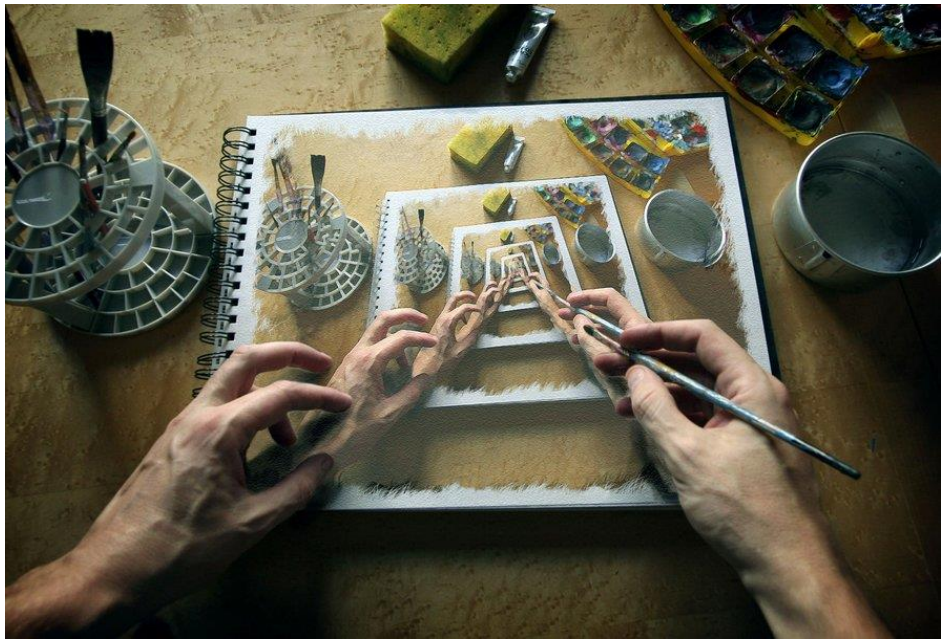


http://de.forwallpaper.com/wallpaper/artist-art-recursion-the-picture-is-dip-brush-177932.html

Java

Agenda

- Workshops zur Wiederholung
- Tutorium
  - Referenzdatentypen
  - Rekursionen
- Abschlussaufgabe



http://de.forwallpaper.com/wallpaper/artist-art-recursion-the-picture-is-dip-brush-177932.html

Ergebnisse des Workshops

- String.toCharArray(“Text“) wandelt einen String in ein CharArray.
- Integer.parseInt(String text) wandelt einen String in einen Integer:
  - Ist dies nicht möglich wird eine Exception (Fehlermeldung) ausgeworfen.
  - Ähnliche parse-Methoden gibt es auch für andere Datentypen wie Double oder Short.
  - Wichtig: Um die parse-Methode aufzurufen benötigt man die Klasse (hier Integer) und nicht den primitiven Datentyp (z.B. int)!
- Es gibt diverse Methoden, die für alle Klassen/Objekte automatisch existieren, auch wenn man diese nicht explizit implementiert:
  - toString() liefert eine String-Repräsentation des Objekts
  - equals() dient dem Vergleich zweier Objekte
  - Überschreibt man diese Methoden, kann man selbst festlegen, wie z.B. der String aussehen soll oder anhand welcher Attribute 2 Objekte verglichen werden sollen.

Beispiel

Wie lauten die Ausgaben der Java-Programme?

```
public class WhatDoThis1 {

  public static void main(String[] args) {
    // Guthaben initial setzen und ausgeben
    int guthaben;
    guthaben = 100;
    System.out.println("Guthaben: " +
guthaben);
    // Guthaben irgend einer Methode
übergeben
    secretMethod(guthaben);
    // Guthaben ausgeben
    System.out.println("Guthaben: " +
guthaben);
  }

  public static void secretMethod(int y) {
    y = y - 10;
  }
}
```

Hinweis: Keine Rückgabe!

```
Guthaben: 100
Guthaben: 100
```

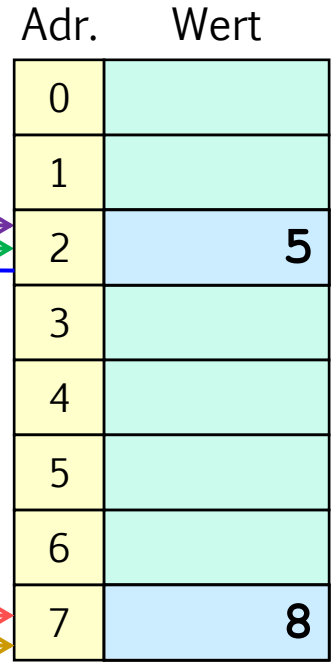
Warum ?

```
public class WhatDoThis2 {

  public static void main(String[] args) {
    // Guthaben initial setzen und ausgeben
    int[] guthaben = new int[1];
    guthaben[0] = 100;
    System.out.println("Guthaben: " +
guthaben[0]);
    // Guthaben irgend einer Methode übergeben
    secretMethod(guthaben);
    // Guthaben ausgeben
    System.out.println("Guthaben: " +
guthaben[0]);
  }

  public static void secretMethod(int[] guthaben)
{
    guthaben[0] = guthaben[0] - 10;
  }
}
```

```
Guthaben: 100
Guthaben: 90
```

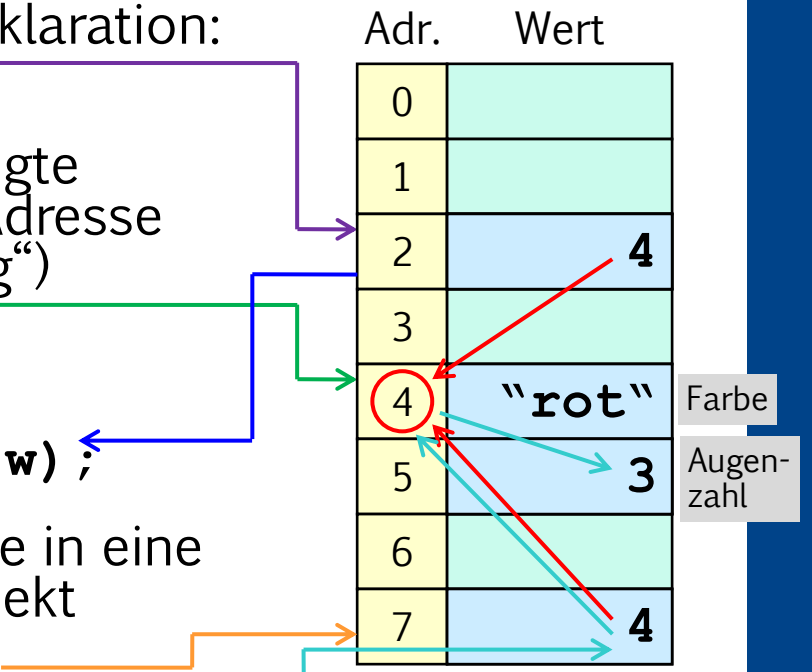
Primitive Datentypen

- Primitive Datentypen:
  `boolean, char, byte, short, int, long, float, double`
  ≈ alle klein geschriebenen Datentypen, keine Arrays
- Reservierung eines Bereiches im Arbeitsspeicher bei Deklaration:
  `int x;`
- Belegung mit Werten bei Wertzuweisung:
  `x = 5;`
- Bei Variablenübergabe wird der Wert der Zelle (=5) übergeben
  `secretMethod(x);`
- Die nächste Methode speichert sich den Wert in eine eigene Variable
  `public void secretMethod(int y) {` …
- Änderungen, welche die aufgerufene Methode an "ihrer" Variable vornimmt, haben keinen Einfluss auf das "Original"
  `y = 8;`



Referenzdatentypen

- Referenzdatentypen:
  `String, Integer, Arrays, Objekte,` …
  ≈ alle groß geschriebenen Datentypen
- Reservierung eines Bereiches im Arbeitsspeicher bei Deklaration:
  `Wuerfel w;`
- Wenn das Objekt instanziiert wird, wird der dafür benötigte Speicher (oftmals größerer Bereich) reserviert und die Adresse dieses Speichers in der Variable abgelegt (“Verknüpfung“)
  `w = new Wuerfel(“rot“);`
- Bei Variablenübergabe wird der Wert der Zelle (=Speicheradresse 4) übergeben **`secretMethod(w);`**
- Die nächste Methode speichert sich die Speicheradresse in eine eigene Variable, die aber immer noch auf das selbe Objekt referenziert
  `public void secretMethod(Wuerfel v) {` …
- Änderungen, welche die aufgerufene Methode an “ihrem“ Objekt vornimmt, betreffen auch das ursprüngliche Objekt, da die es den selben Bereich im Speicher referenziert: **`v.wuerfeln();`**



Wiederholung Beispiel

Wie lauten die Ausgaben der Java-Programme?

```
public class WhatDoThis1 {

  public static void main(String[] args) {
    // Guthaben initial setzen und ausgeben
    int guthaben;
    guthaben = 100;
    System.out.println("Guthaben: " + guthaben);
    // Guthaben irgend einer Methode übergeben
    secretMethod(guthaben);
    // Guthaben ausgeben
    System.out.println("Guthaben: " + guthaben);
  }

  public static void secretMethod(int y) {
    y = y - 10;
  }
}
```

Primitiver Datentyp

Wert wird als Kopie übergeben

Operation auf kopiertem Wert

```
Guthaben: 100
Guthaben: 100
```

```
public class WhatDoThis2 {

  public static void main(String[] args) {
    // Guthaben initial setzen und ausgeben
    int[] guthaben = new int[1];
    guthaben[0] = 100;
    System.out.println("Guthaben: " + guthaben[0]);
    // Guthaben irgend einer Methode übergeben
    secretMethod(guthaben);
    // Guthaben ausgeben
    System.out.println("Guthaben: " + guthaben[0]);
  }

  public static void secretMethod(int[] guthaben) {
    guthaben[0] = guthaben[0] - 10;
  }
}
```

Referenzdatentyp

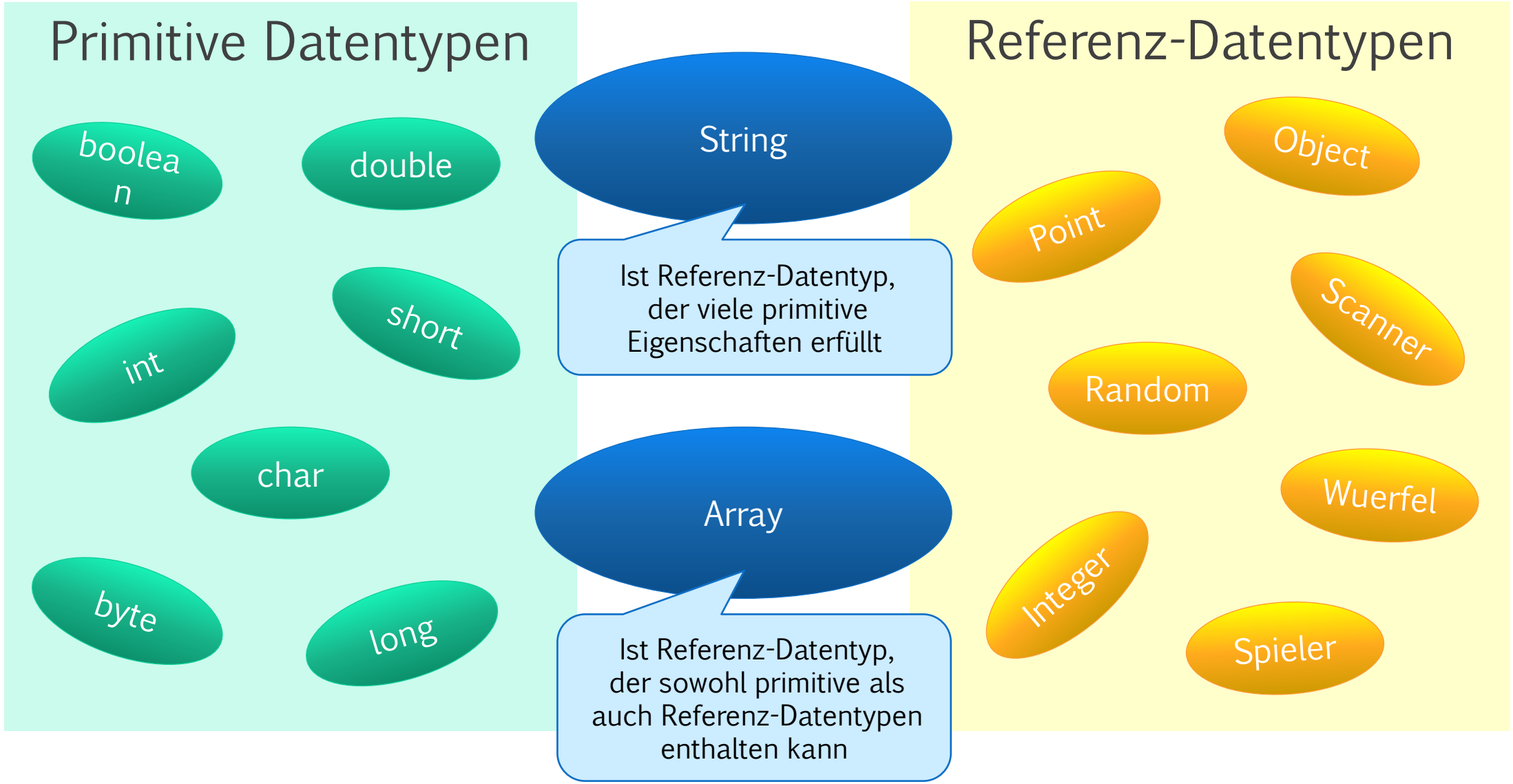
Speicheradresse wird übergeben

Operation auf ursprünglichem Objekt

```
Guthaben: 100
Guthaben: 90
```

Hinweis: Falls die Operation der Methode **`secretMethod()`** auf die aufrufende Methdode **`main()`** übernommen werden soll, "**`return y;`**" verwenden.

Übersicht Datentypen



Rückblick zu Variablen

Variablen können ...

... Werte zugewiesen werden

```
int x = 5;
```

... an Methoden übergeben werden

```
myMethod(x);
```

... innerhalb der aufgerufenen Methoden verwendet werden

```
public void myMethod(int x) {
    System.out.println("X ist " + x);
}
```

Aufgabe: Referenzen

GRUNDLAGE

- Erstelle eine Klasse Zahl mit dem Attribut “int wert“
- In einer main Methode: Erzeuge ein Objekt der Klasse Zahl mit Wert 1. Erstelle außerdem ein int i mit Wert 1.
- Kopiere das Objekt und den int in jeweils eine neue Variable bzw. Objekt.
- Verändere den Wert der originalen Variable und des Objekts.
- Gib den Wert der neuen Variable und des Objekts aus.

ZUSATZ

- Füge der Klasse “Zahl“ eine Methode hinzu, die eine Kopie von dem Objekt erzeugt, dass sie aufruft.
- Diese Kopie soll jedoch so sein, dass man das Original verändern kann, “ohne die Kopie zu ändern“.
- Überprüfe deine Implementierung in einer main Methode.

Motivation Rekursion

Beispiel: Erkennen von Palindromen

(Palindrom = Wort, das von vorne nach hinten gelesen das selbe ergibt wie von hinten nach vorne, Bsp. “OTTO“)

Ansatz:

Wir prüfen jeweils den ersten und den letzten Buchstaben auf Identität, entfernen diese bei Übereinstimmung und prüfen erneut den ersten und letzten Buchstaben auf Identität, so lange, bis alle Buchstaben entfernt sind.

Palindrom

Skizze der Idee:

```
String wort = "anna"
if (wort.ersterBuchstabe == wort.letzterBuchstabe) {
  wort.loescheErstenBuchstaben();          // "anna" => "nna"
  wort.loescheLetztenBuchstaben();         //  "nna" => "nn"
  if (wort.laenge <= 1) {
    Ausgabe "Palindrom"
  } else {
if (wort.ersterBuchstabe == wort.letzterBuchstabe) {
      wort.loescheErstenBuchstaben();      // "nn" => "n"
      word.loescheLetztenBuchstaben();     //  "n" => ""
      if (wort.laenge <= 1) {
        Ausgabe "Palindrom"
      } else {
        ...
      }
} else {
      Ausgabe "kein Palindrom"
    }
  }
} else {
  Ausgabe "kein Palindrom"
}
```

Praktikabler Ansatz

Aber: Wie programmieren wir das?

Verfeinerung der Idee: Rekursion

```
String wort = "anna"
```

```
pruefePalindrom(String wort) {
 if (wort.laenge <= 1) {
   Ausgabe "Palindrom"
 } else {
  if (wort.ersterBuchstabe == wort.letzterBuchstabe) {
    wort.loescheErstenBuchstaben();
    wort.loescheLetztenBuchstaben();
    pruefePalindrom(wort);
  } else {
    Ausgabe "kein Palindrom"
  }
}
```

Die Methode ruft sich immer wieder selbst auf

Bis die Abbruchbedingung zu Beginn der Methode erfüllt ist

Abbruchbedingung **muss** bei rekursiver Programmierung enthalten sein, sonst Endlosschleife!

Übungen aus der Aufgabensammlung

Name	Seite	Schwierigkeitsgrad (1-10)	Freiheitsgrad (0-3)	Thema
Pascalsches Dreieck	40	8	2	Schleifen, Rekursion
Rekursive Funktion	37	3	0	Rekursion
Newton-Verfahren	29	7	3	Recherche, Rekursion
…				
Der Rest der Aufgabensammlung				

15