THM TECHNISCHE HOCHSCHULE MITTELHESSEN | CAMPUS GIESSEN | MNI Mathematik, Naturwissenschaften und Informatik

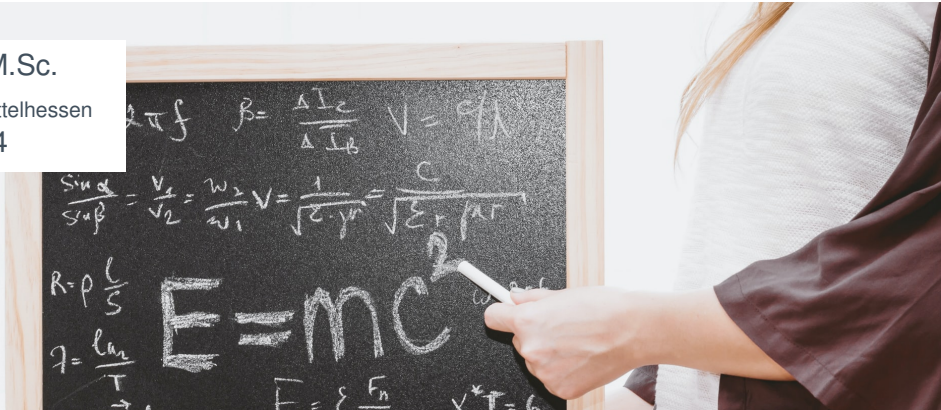
Brückenkurs Programmieren

Tag 4: Objektorientierung

Jakob Czekansky, M.Sc.

Technische Hochschule Mittelhessen

05. September 2024



Rückblick - Tag 1: Variablen und Verzweigungen

Variablen

```
int i = 10;
float f = 3.5;
boolean b = true;
char c = 'x';
String blabla = "Blabla";
```

If-Abfrage

```
if (<Bedingung>) {
  <Anweisungsblock1>
} else {
  <Anweisungsblock2>
}
```

Logische Operatoren

> >= == !=

&& || !

Arithmetische Operatoren

\+ - * / %

Rückblick - Tag 2: Schleifen

Beispiel: For-Schleife

```
for (int i=0;i<end;i++) {
  ellipse(i*10,20,10,10);
}
```

Syntax: While-Schleife

```
while(<Bedingung>) {
  <Anweisungsblock>
}
```

Rückblick - Tag 2: Animationen und Arrays

setup und draw

```
void setup() {
  //einmal am Anfang
}
void draw() {
  //alle 15 ms
}
```

Mausposition

```
void draw() {
  ellipse(mouseX,20,10,10);
}
```

Arrays

```
int[] numbers = new int[10];
numbers[0] = 20;
numbers[3+2] = numbers[0];
int val = numbers[5];
```

Rückblick - Tag 3: Funktionen

Funktionsdefinition

```
int add(int a, int b) {
  return a + b;
}
```

Funktionsaufruf

```
int x = add(10,3);
```

Rückblick - Tag 3: Events

keyPressed

```
int counter = 0;
void keyPressed() {
  if(key == 'a') {
    counter++;
  } else if (key == 'x') {
    exit(0);
  }
}
```

mousePressed

```
void mousePressed() {
  if(mouseButton == LEFT){
    background(255);
  }
  else{
    background(0);
  }
}
```

Inhalt

Inhalt

Wert- vs Referenzsemantik

Erinnerung

Java unterscheidet zwischen primitiven Typen (z.B. `int`) und Referenztypen (Arrays und Klassen).

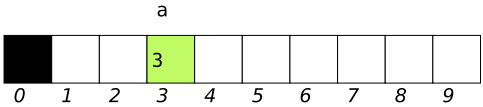
Frage: Produzieren die beiden folgenden Codestücke die gleiche Ausgabe?

```
int a = 3;
int b;
b = a;
b = 4;
println(a);
```

```
int[] a = new int[2];
a[0] = 3;
int[] b;
b = a;
b[0] = 4;
println(a[0]);
```

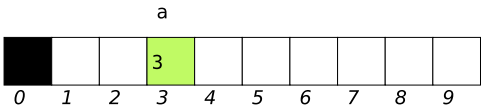
Wertsemantik bei primitiven Typen

```
int a = 3;
```

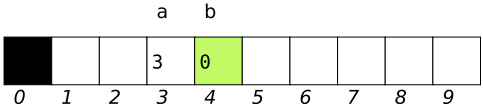


Wertsemantik bei primitiven Typen

```
int a = 3;
```

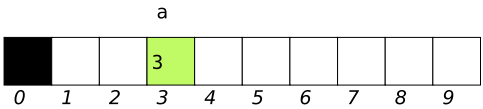



```
int b;
```

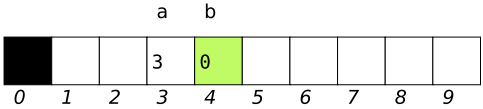


Wertsemantik bei primitiven Typen

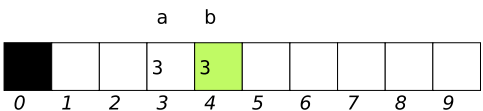
```
int a = 3;
```




```
int b;
```

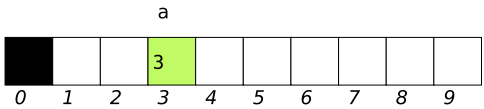



```
b = a;
```

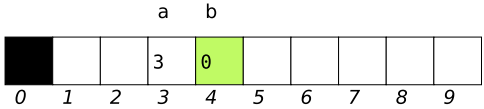


Wertsemantik bei primitiven Typen

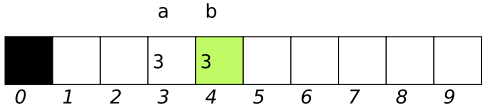
```
int a = 3;
```



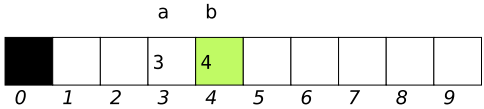

```
int b;
```




```
b = a;
```

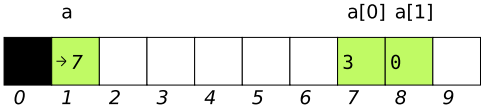



```
b = 4;
```



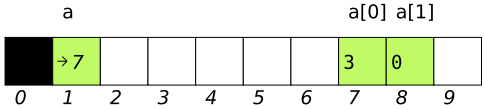
Referenzsemantik bei Referenztypen

```
int[] a = new int[2];
a[0] = 3;
```

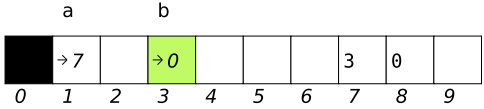


Referenzsemantik bei Referenztypen

```
int[] a = new int[2];
a[0] = 3;
```

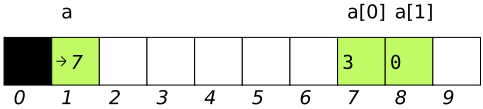



```
int[] b;
```

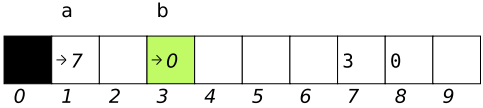


Referenzsemantik bei Referenztypen

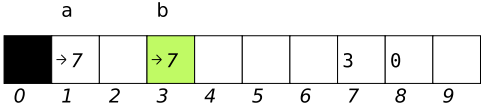
```
int[] a = new int[2];
a[0] = 3;
```




```
int[] b;
```

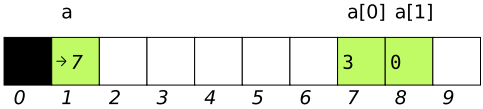



```
b = a;
```

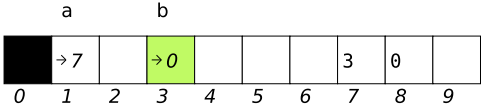


Referenzsemantik bei Referenztypen

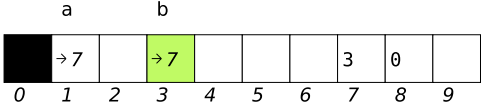
```
int[] a = new int[2];
a[0] = 3;
```



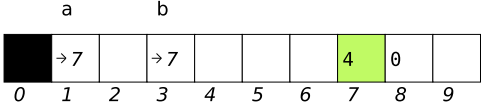

```
int[] b;
```




```
b = a;
```




```
b[0] = 4;
```



Wert- vs Referenzsemantik: Zusammenfassung

Der Inhalt von Variablen unterscheidet sich je nach dem, welchen Typ die Variable hat.

- primitiver Typ: Variable enthält Wert
  - bei Zuweisung wird der Wert direkt kopiert
- Referenztyp: Variable enthält Referenz bzw. Zeiger
  - verweist auf Speicherstelle an der die Daten liegen
  - bei Zuweisung wird nur die Referenz kopiert
  - die Daten bleiben die selben

Wert- vs Referenzsemantik: Übungsaufgabe

Gemeinsam an der Tafel: Was steht im Speicher, wenn der folgende Code ausgeführt wurde?

```
float[] ypos;
float[] xpos;

ypos = new float[]{100, 150, 200}
xpos = ypos;

float x = xpos[1];
x *= 2;
ypos[1] += 1;
```

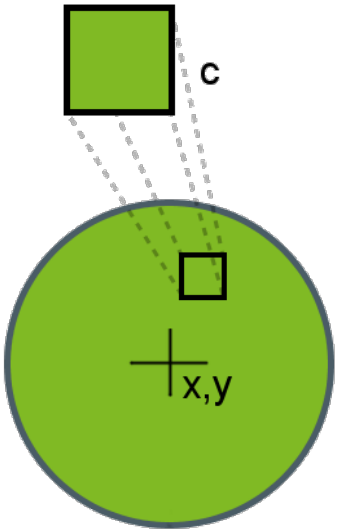
Inhalt

Klassen als eigener Datentyp

Definition

Eine Klasse definiert einen eigenen Datentyp, der Variablen und Funktionen zu einer einheitlichen Struktur zusammenfasst.

```
class Ball {
  float x,y;
  color c;
  Ball(float x, float y, color c) {
    this.x = x;
    this.y = y;
    this.c = c;
  }
}
```



Konstruktor

Definition

Ein Konstruktor dient der Initialisierung von Instanzen einer Klasse. Er bestimmt, wie die Variablen eines Objektes dieser Klasse direkt nach der Erstellung belegt sind.

```
class Ball {
  float x,y;
  color c;
  Ball(float x, float y, color c) {
    this.x = x;
    this.y = y;
    this.c = c;
  }

}
```

Erzeugen von Objekten

Definition

Ein Objekt ist eine Instanz einer Klasse. Jedes Objekt hat seine eigenen Variablenwerte, die durch den Konstruktoraufruf mit dem Schlüsselwort `new` gesetzt werden.

```
Ball b1 = new Ball(100,100,color(255,0,0));
Ball b2 = new Ball(150,200,#80ba24);
```

Inhalt

Objektmethoden

Bisher:

- Objekte sind reine Datenbehälter
- Klasse `Ball` gibt keine Informationen zur Darstellung
- `ellipse`-Befehl muss jedesmal neu aufgerufen werden

Erinnerung: Klasse

Eine Klasse definiert einen eigenen Datentyp, der Variablen und Funktionen zu einer einheitlichen Struktur zusammenfasst.

Objektmethoden: Definition

Definition

Eine Methode ist eine Funktion, die an ein Objekt gebunden ist. Sie kann direkt auf Objektvariablen zugreifen und so das Verhalten von Objekten einer Klasse definieren.

```
class Ball {
  float x,y;
  color c;
  Ball(float x, float y, color c) { ... }

  void display() {
    fill(c);
    ellipse(x,y,50,50);
  }


}
```

Objektmethoden: Aufruf

Aufruf von Objektmethoden:

- genau wie bei normalen Funktionen
- aber immer gebunden an ein Objekt
- Aufrufschema: `<objektvariable>.<methode>(<argument1>, <argument2>, ...)`

```
Ball b2 = new Ball(150,200,#80ba24);

void setup() {
  b2.display();
}
```

Zusammenfassung

Klasse als eigener Datentyp

```
class Ball {
  float x, y;
  color c;
  Ball(float x, float y, color c) {
    this.x = x;
    this.y = y;
    this.c = c;
  }
}
```

Objekte und Methodenaufruf

```
Ball b = new Ball(100,200);
b.display();
```

Klasse mit Methoden

```
class Ball {
  float x, y;
  color c;
  Ball(float x, float y, color c) {
    this.x = x;
    this.y = y;
    this.c = c;
  }
  void display() {
    fill(this.c);
    ellipse(x,y,50,50);
  }
  void moveX(int x){
    this.x = this.x + x;
  }
}
```