

BRÜCKENKURS PROGRAMMIEREN - FIONA NÜESCH

REPETITION

ARRAYS UND SCHLEIFEN

frei wählbarer Name Anzahl Elemente, die die Sammlung beinhaltet (ganzzahlig)

```
int[] arrayName = new int[10];
```

Datentyp Ein Referenztyp wird über die Anweisung **new** Erzeugt

```
for(int i = 0; i < arrayName.length; i++){  
    arrayName[i] = i*2;  
}
```

VERZWEIGUNGEN UND BEDINGUNGEN

```
1 int a = 10;  
2 int b = 5;  
3 int c = 30;  
4  
5 if( !( a > b && c != a ) || c > a ){  
6     println("komplizierte Bedingung erfüllt");  
7 }else{  
8     println("nicht erfüllt");  
9 }
```

METHODEN

```
1
2 void setup() {
3
4     int result = line(2);    Methodenaufruf
5
6     printWords("my", "name", "is", "fiona");
7
8 }
```

```
12
13 int line(int x) {
14
15     int y = 2 * x + 1;    Methodendefinition
16
17     return y;
18 }
19
20
21 void printWords(String word1, String word2, String word3, String word4) {
22
23     println( word1 + word2 + word3 +word4 );
24
25 }
26
```

Methoden Definition

Rückgabe Typ (Datentyp des Outputs) **Methodenname** **Methodenparameter** (Input mit Typ und Name)

```
int line(int x){  
    int y = 2 * x + 1;  
    return y;  
}
```

Methoden
Code
Block

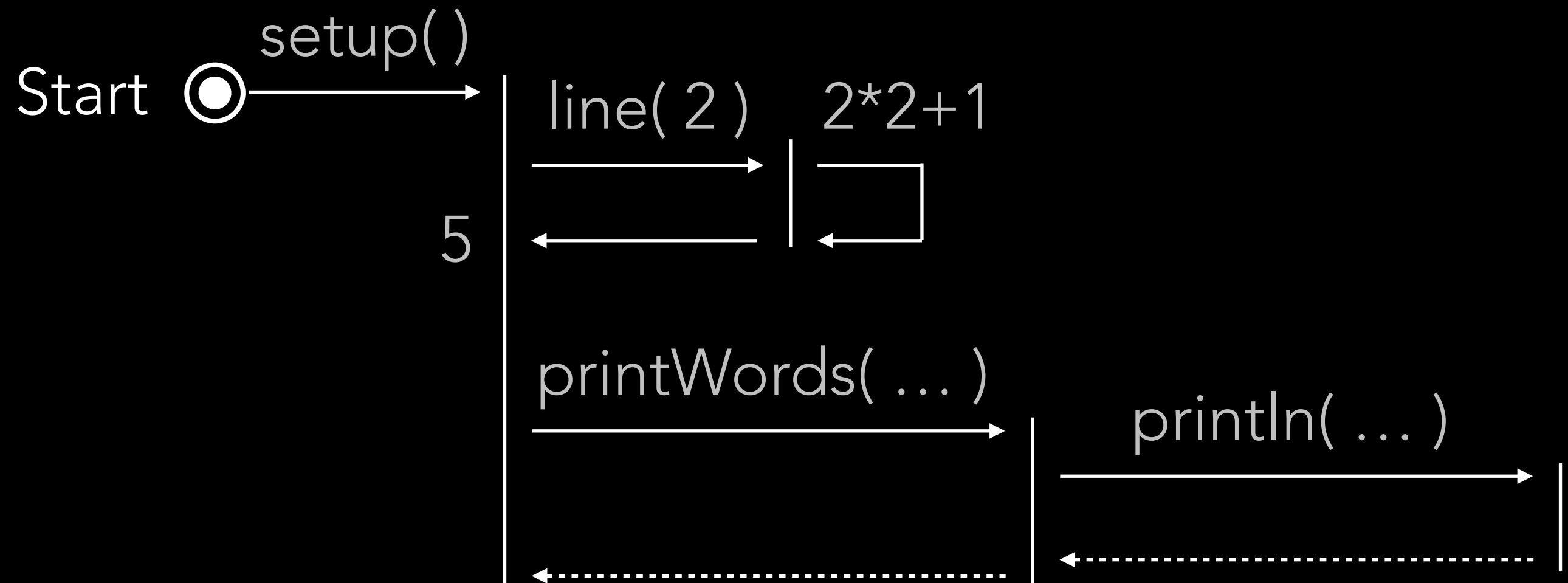
neuer Datentyp: **void**
für keinen
Rückgabewert (keine
return-Anweisung)

return Statement

- Definiert welche Variable zurückgegeben wird.
- Muss dem Typ des Rückgabewerts entsprechen.
- Beendet die Methode.

```
1 void setup(){
2
3
4   int result = line(2);
5
6   printWords("my", "name", "is", "fiona");
7
8 }
```

```
12
13 int line(int x){
14
15   int y = 2 * x + 1;
16
17   return y;
18 }
19
20
21 void printWords(String word1, String word2, String word3, String word4){
22
23   println( word1 + word2 + word3 +word4 );
24
25 }
26
```



```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}
```

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```

```
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

Start

⊙ → `void setup(){
 int myX = 2;
 int result = line(myX);
 printResult("MY", "RESULT", "IS", result);
}`

`int line(int x){
 int y = 2*x+0;
 return y;
}`

`void printResult(String word1, String word2,
String word3, int result){
 println(word1, word2, word3, result);
}`


```
→ void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}  
  
int line(int x){  
    int y = 2*x+0;  
    return y;  
}  
  
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

```
→ void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}  
  
int line(int x){  
    int y = 2*x+0;  
    return y;  
}  
  
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}
```

→

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```

```
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}
```

→

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```

```
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}
```

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```



```
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}
```

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```

5
←

```
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

→
result:5

```
void setup(){
    int myX = 2;
    int result = line(myX);
    printResult("MY", "RESULT", "IS", result);
}

int line(int x){
    int y = 2*x+0;
    return y;
}

void printResult(String word1, String word2,
String word3, int result){
    println(word1, word2, word3, result);
}
```

```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
→    printResult("MY", "RESULT", "IS", result);  
}
```

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```

```
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```



```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}
```

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```

→

```
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}
```

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```

```
void printResult(String word1, String word2,  
String word3, int result){  
→    println(word1, word2, word3, result);  
}
```

```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}
```

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```

```
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

←.....

```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
→ }
```

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```

```
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

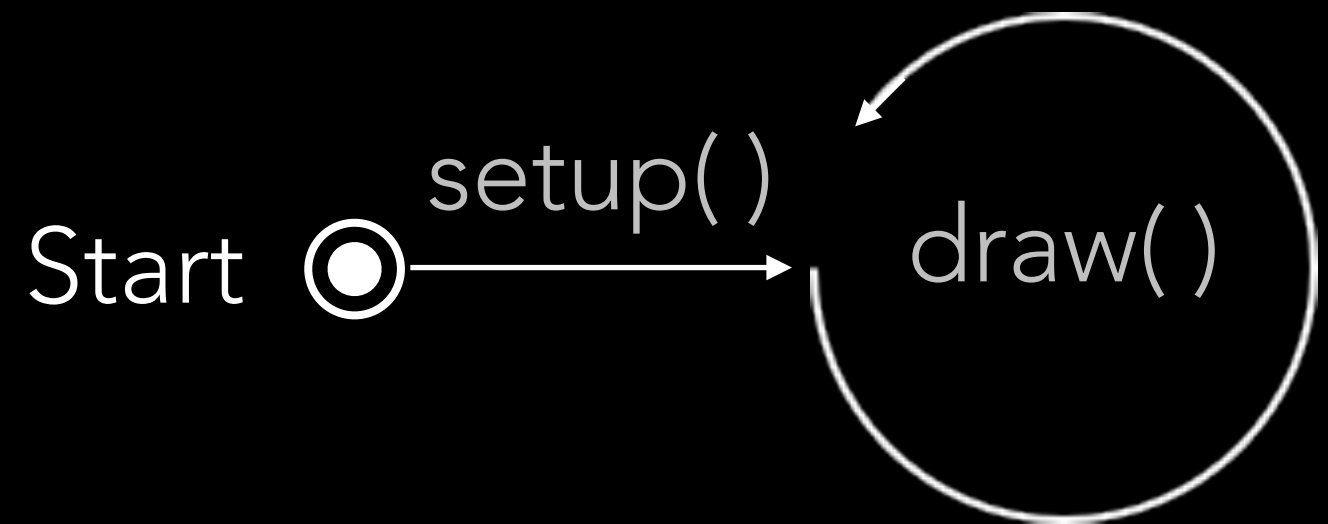
```
void setup(){  
    int myX = 2;  
    int result = line(myX);  
    printResult("MY", "RESULT", "IS", result);  
}
```

```
void draw(){  
    line(random(1, 10));  
}
```

```
int line(int x){  
    int y = 2*x+0;  
    return y;  
}
```

```
void printResult(String word1, String word2,  
String word3, int result){  
    println(word1, word2, word3, result);  
}
```

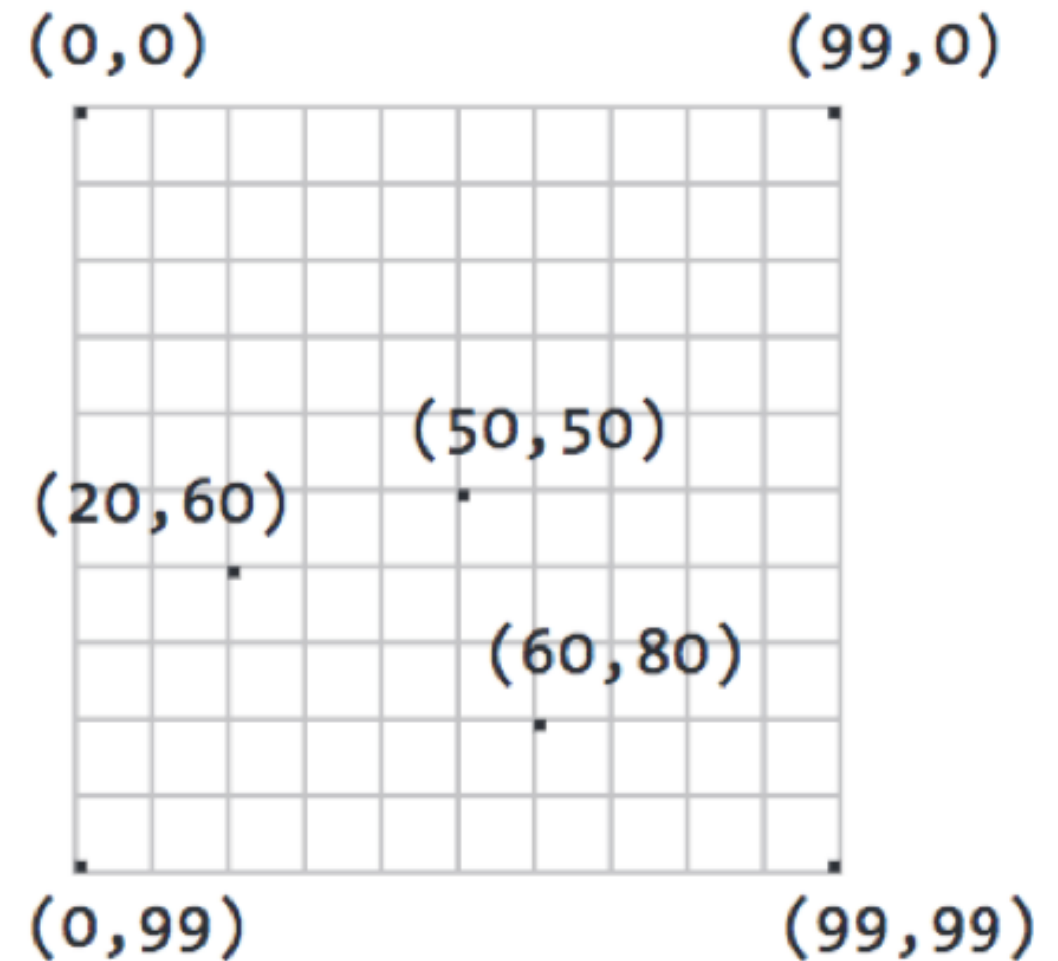
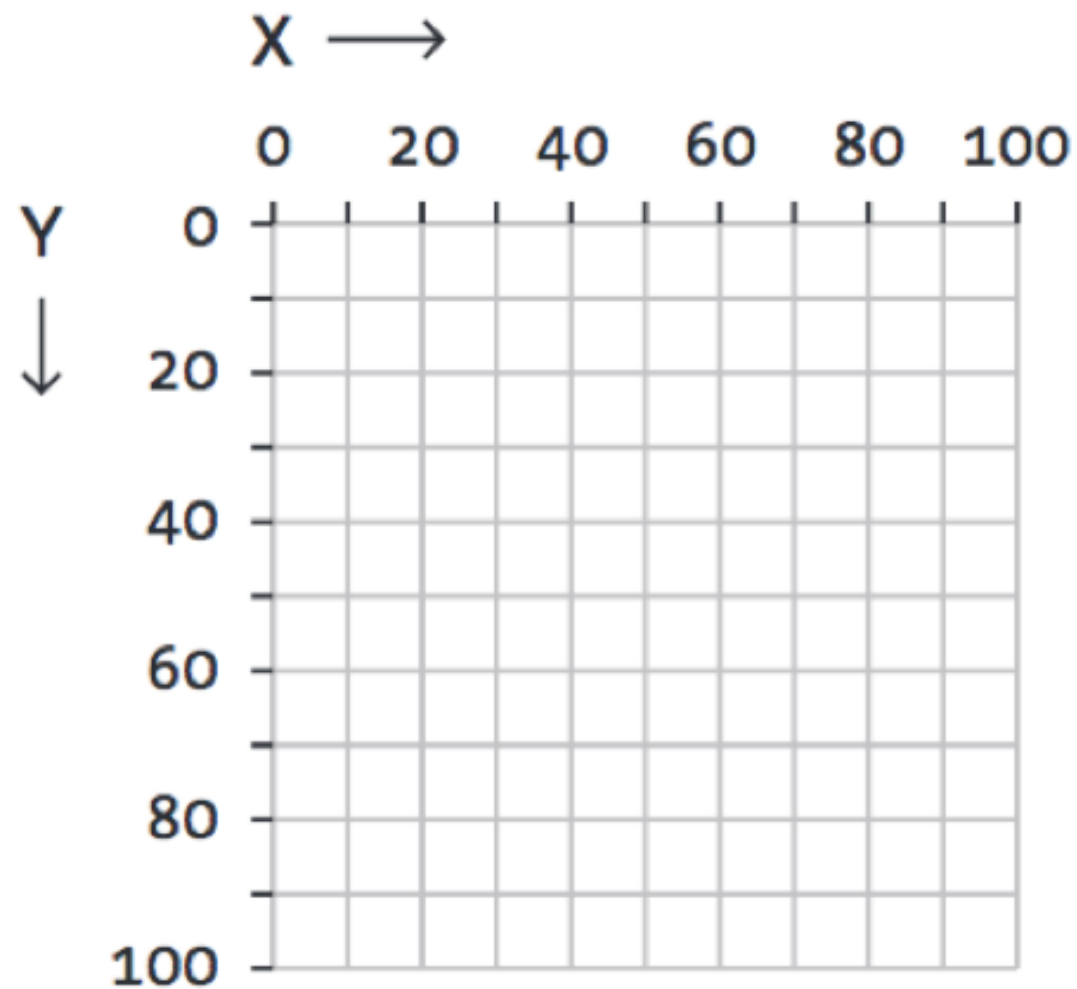
PROCESSING



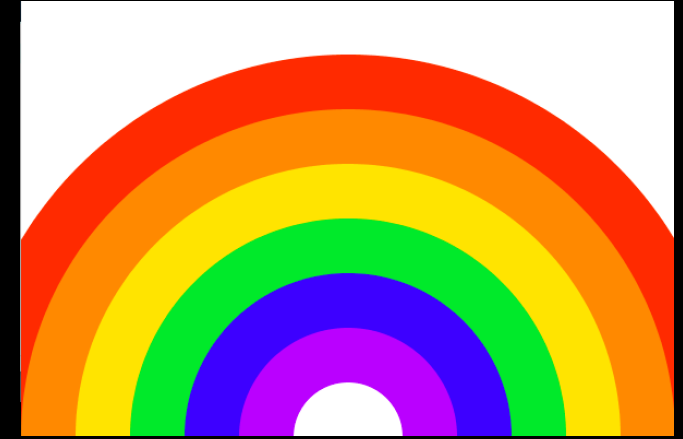
sketch_180613a

```
1 void setup(){
2   size(300, 300);
3 }
4
5 void draw(){
6   background(0);
7   text("Hello World!", 100, 100);
8   fill(0, 102, 153);
9 }
10
```

Das Koordinatensystem



Grafik



```
1 void setup(){
2   size(600, 400);
3   background(255);
4   noStroke();
5
6   fill(255, 45, 0);
7   ellipse(300, height, 700, 700);
8   fill(255, 137, 7);
9   ellipse(300, height, 600, 600);
10  fill(255, 226, 3);
11  ellipse(300, height, 500, 500);
12  fill(11, 232, 60);
13  ellipse(300, height, 400, 400);
14  fill(62, 12, 255);
15  ellipse(300, height, 300, 300);
16  fill(184, 0, 255);
17  ellipse(300, height, 200, 200);
18  fill(255);
19  ellipse(300, height, 100, 100);
20 }
```

Display Window Dimension 600 x 400 px

Display Window Hintergrundfarbe : weiss

alle folgenden Figuren ohne Outline

fülle folgende Figur mit rot

zeichne einen Kreis

Mittelpunktkoordinaten: (300, height)

und Länge, Breite mit 700 px

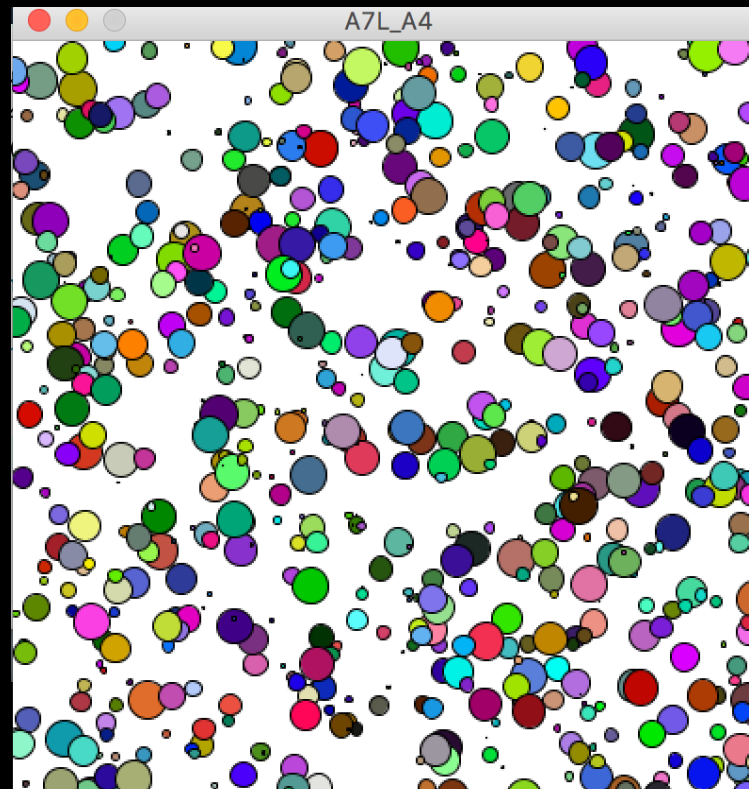
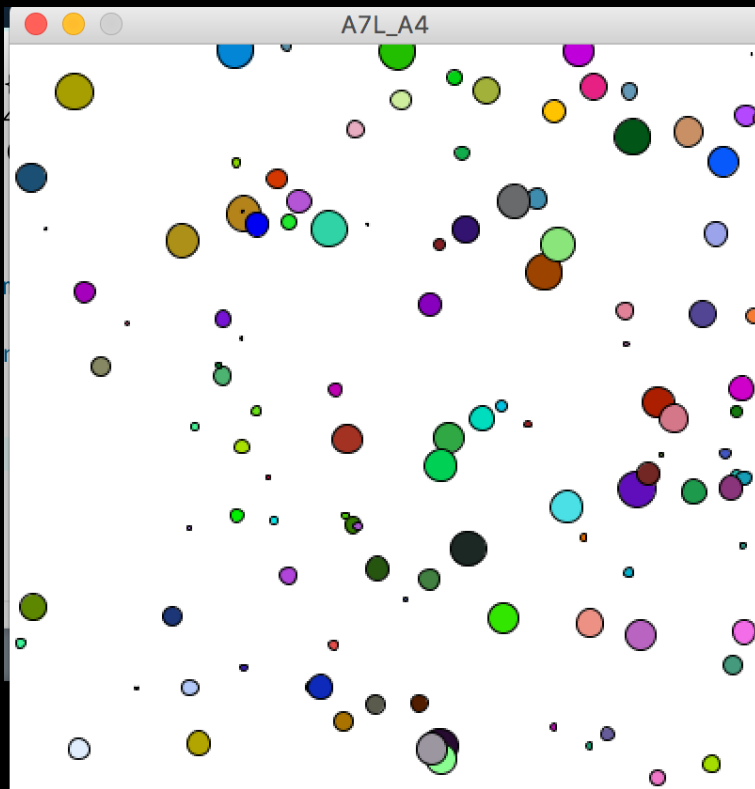
Grafik über die Zeit

```
1 void setup(){
2   size(400, 400);
3   background(255);
4 }
5
6
7 void draw(){
8   fill(random(255), random(255), random(255));
9   float diam = random(20);
10  ellipse(random(width), random(height), diam, diam);
11 }
```

füllt die folgende Figur für jeden draw Durchgang mit einer zufälligen Farbe

generiert für jeden draw Durchgang eine zufällige Zahl, die als Kreisdurchmesser verwendet wird

zeichnet für jeden draw Durchgang einen Kreis an zufälliger Stelle



Animation

```
1 int x = 0;
2 int y = 0;
3
4 void setup() {
5     size(500, 500);
6 }
7
8 void draw() {
9     background(255);
10    rect(x, y, 50, 50);
11    x += 5;
12    y += 5;
13 }
```

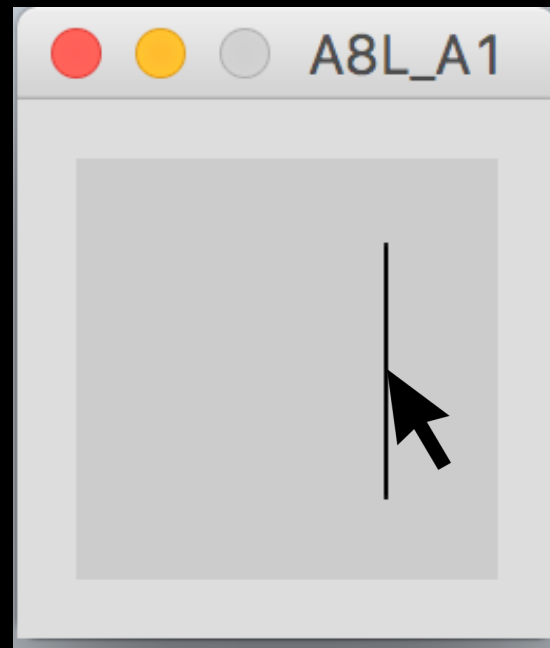
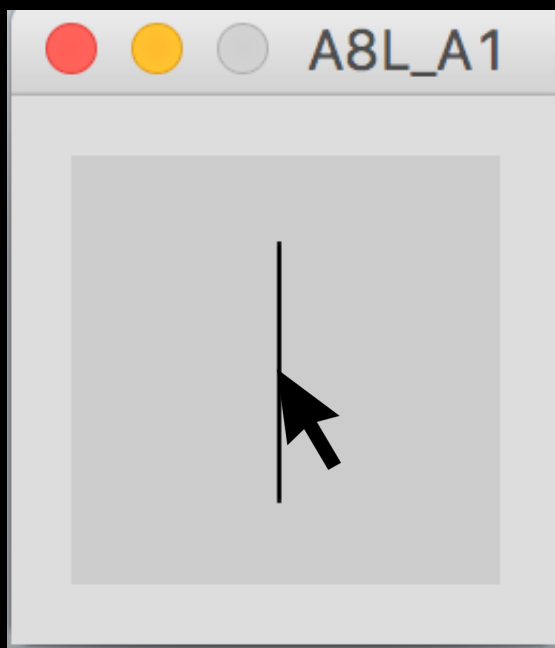


erhöht für jeden draw Durchgang x, y Wert um 5, so dass das Rechteck beim nächsten Durchgang um 5 px nach unten und nach rechts verschoben gezeichnet wird. Da der Hintergrund auch immer neu gezeichnet wird, entsteht eine Animation.

User Input - Maus

mouseX und **mouseY** sind globale Variablen von processing die jeweils die aktuellen Mauskoordinaten enthalten

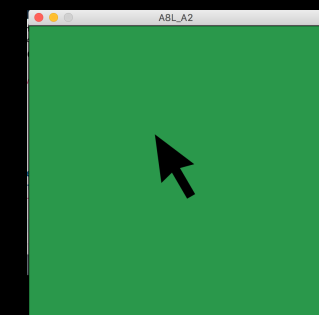
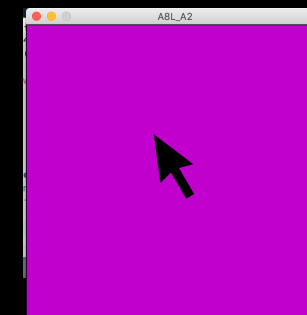
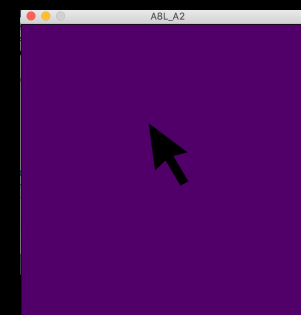
```
1 void draw() {  
2   background(204);  
3   line(mouseX, 20, mouseX, 80);  
4 }
```



void mousePressed wird von processing aufgerufen, wenn der Benutzer mit der Maus klickt.

mousePressed wird nur aufgerufen wenn draw() vorhanden ist. Auch wenn wir da nichts machen

```
1 void setup(){  
2   size(400, 400);  
3   background(0);  
4   fill(0);  
5   rect(0,0, width, height);  
6 }  
7  
8 void draw(){  
9 }  
10  
11  
12 void mousePressed(){  
13   fill(random(255), random(255), random(255));  
14   rect(0,0,width,height);  
15 }
```

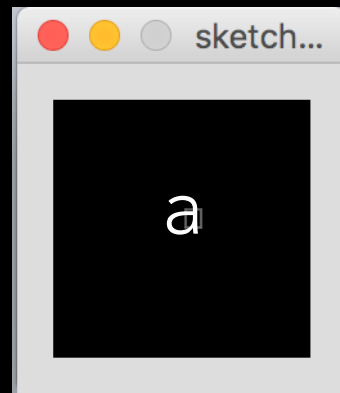


User Input - Keyboard

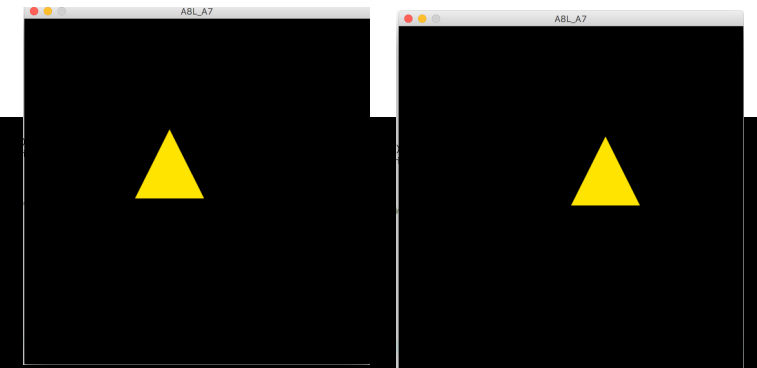
void keyPressed wird von processing aufgerufen, wenn der Benutzer eine Taste klickt.
wird nur aufgerufen wenn draw() vorhanden ist.

key und **keyCode** sind globale Variablen von processing die jeweils die aktuelle Tasteneingabe enthalten

```
1 void setup(){
2   background(0);
3 }
4
5 void draw(){
6 }
7
8 void keyPressed(){
9   background(0);
10  text(key, 50, 50);
11 }
```



```
1 int x;
2 int y;
3 int size = 50;
4
5 void setup(){
6   size(500, 500);
7   x = width/2;
8   y = height/2;
9
10  noStroke();
11 }
12
13 void draw(){
14   background(0);
15   fill(255, 226, 3);
16   triangle( x, y-size, x+size, y+size, x-size, y+size );
17 }
18
19 void keyPressed(){
20   if(keyCode == DOWN){
21     y += 10;
22   }else if(keyCode == UP){
23     y -= 10;
24   }else if(keyCode == LEFT){
25     x -= 10;
26   }else if(keyCode == RIGHT){
27     x += 10;
28   }
29 }
```



Übung 12 Repetition

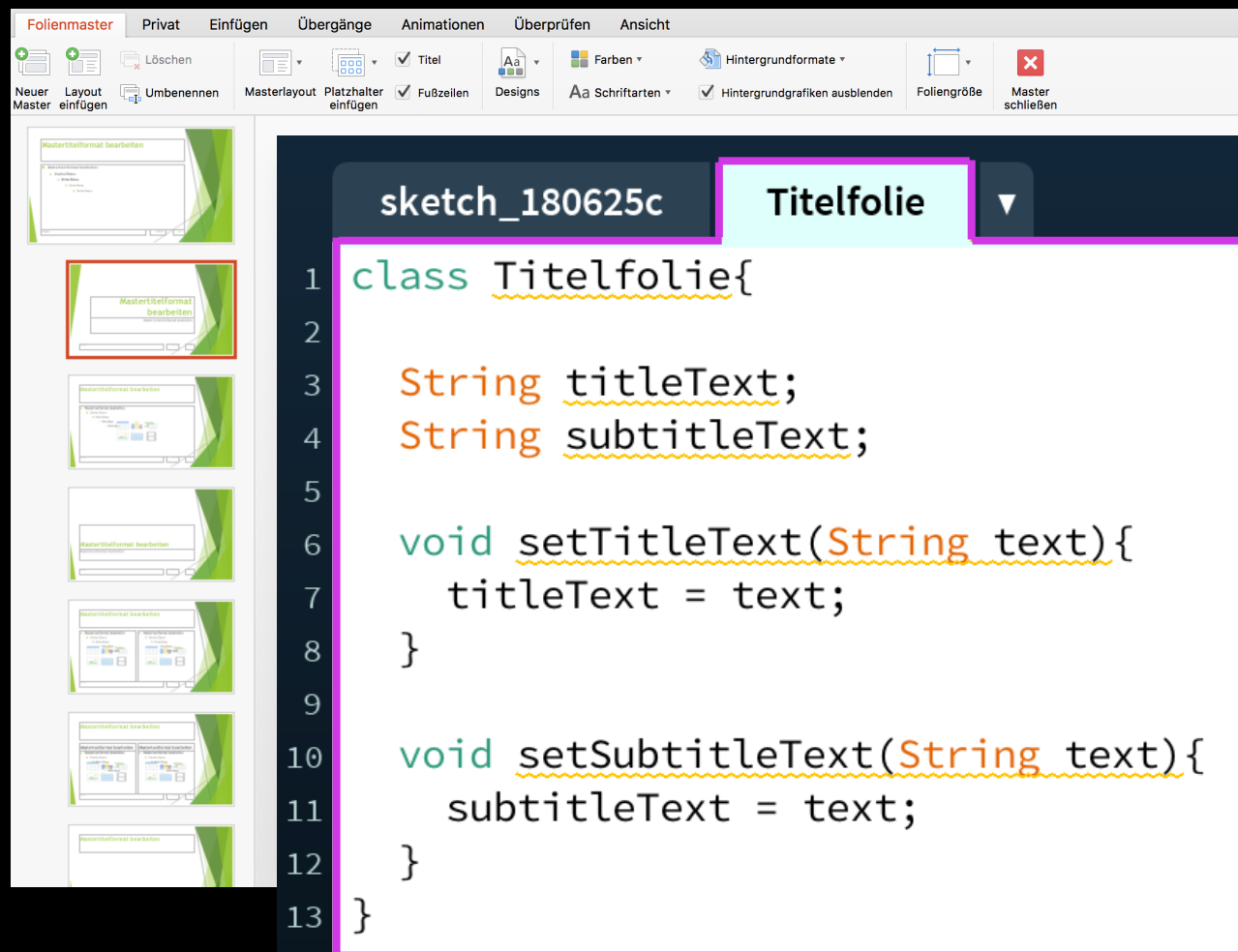
„Repetition.“

–CHRISTIAN BAUER

KLASSEN

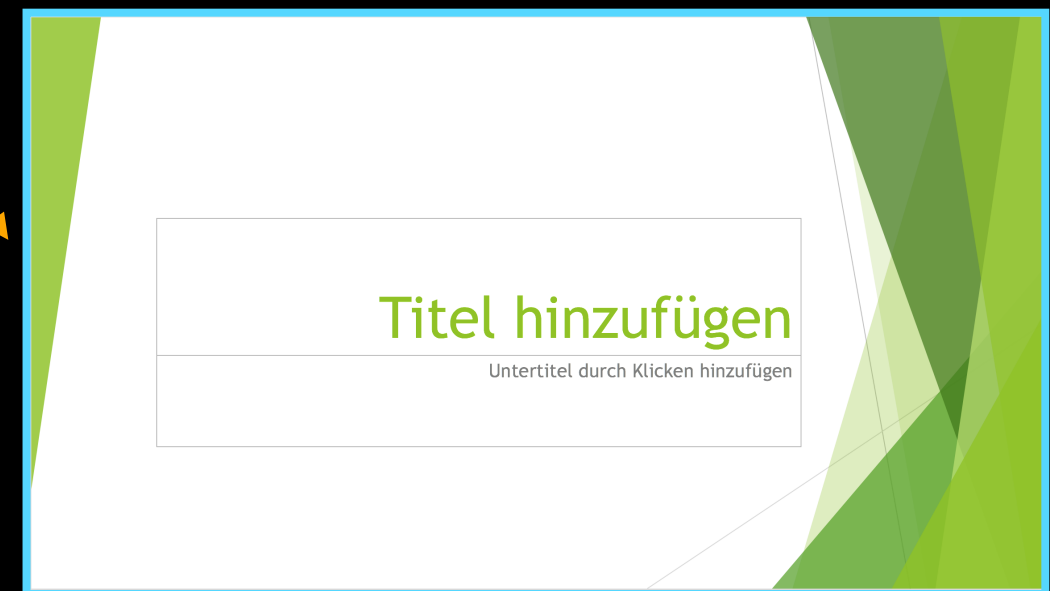
ein Template

aus dem wir ein Objekt instanziiieren können



```
void setup(){
  Titelfolie folie1 = new Titelfolie();
}
```

Objekt folie1



```
class KLASSENNAMEN {
```

Attribute (class member): in Klasse gültige globale Variablen, speichern Zustand

Methoden: definieren Funktionalitäten

```
}
```

Klassen und Objekte verwenden

sketch_180625c

Titelfolie



```
1 class Titelfolie{  
2  
3     String titleText;  
4     String subtitleText;  
5  
6     void setTitleText(String text){  
7         titleText = text;  
8     }  
9  
10    void setSubtitleText(String text){  
11        subtitleText = text;  
12    }  
13 }
```

```
; grammieren");
```

new Titelfolie()



```
1 void setup(){
2   Titelfolie folie1 = new Titelfolie();
3   folie1.setTitleText("Brückenkurs Programmieren");
4   folie1.subtitleText("Fiona Nüesch");
5 }
```

neues Objekt vom Typ Titelfolie

new Titelfolie()



```
powerpoint  Titelfolie  ▼
1 void setup(){
2   Titelfolie folie1 = new Titelfolie();
3   folie1.setTitleText("Brückenkurs Programmieren");
4   folie1.subtitleText("Fiona Nüesch");
5 }
```



benutze auf diesem
Objekt die
Methode setTitleText()
um den Zustand zu
ändern

setTitleText(„Brück...“)↓

```
3   String titleText;
4   String subtitleText;
5
6   void setTitleText(String text){
7       titleText = text;
8   }
9
10  void setSubtitleText(String text){
11      subtitleText = text;
12  }
```



setTitleText(„Brück...“) ↓



subtitleText(„Fiona...“) ↓

```
1  
2  
3 String titleText;  
4 String subtitleText;  
5  
6 void setTitleText(String text){  
7     titleText = text;  
8 }  
9  
10 → void setSubtitleText(String text){  
11     subtitleText = text;  
12 }
```



```
powerpoint Titelfolie ▼  
1 void setup() {  
2     Titelfolie folie1 = new Titelfolie();  
3     folie1.setTitleText("Brückenkurs Programmieren");  
4 → folie1.subtitleText("Fiona Nüesch");  
5 }
```

benutze auf diesem
Objekt die
Methode subtitleText()
um den Zustand zu
ändern

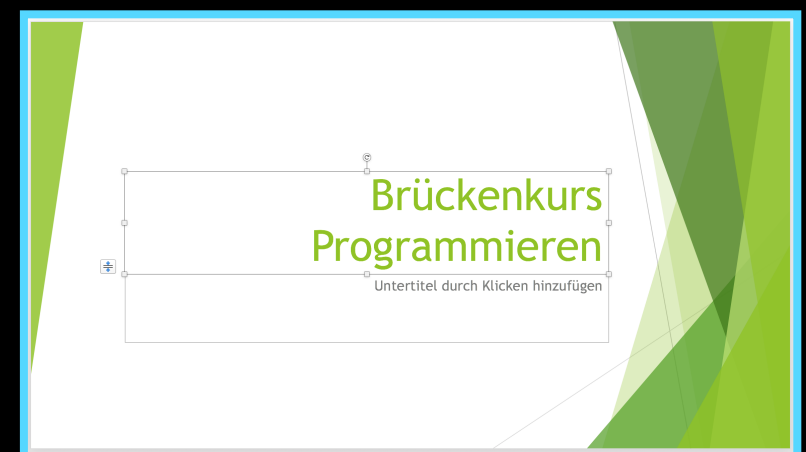
new Titelfolie()



```
powerpoint  Titelfolie ▼
1 void setup(){
2   Titelfolie folie1 = new Titelfolie();
3   folie1.setTitleText("Brückenkurs Programmieren");
4   folie1.subtitleText("Fiona Nüesch");
5 }
```



setTitleText(„Brück...“) ↓



subtitleText(„Fiona...“) ↓



```
1 void setup() {  
2     int i = 1;  
3     int j = 2;  
4     i = i+j;  
5     j = i*2;  
6 }
```

```
1 void setup(){
2     Titelfolie folie1 = new Titelfolie();
3     folie1.setTitleText("Brückenkurs Programmieren");
4     folie1.subtitleText("Fiona Nüesch");
5
6     Titelfolie folie2 = new Titelfolie();
7     folie2.setTitleText("Babedi bubedi");
8     folie2.subtitleText("WM ohne Italien");
9 }
10
```



Objekt folie1



Objekt folie2

KLASSEN KONSTRUKTOREN

- spezielle Methode die verwendet wird um ein **Objekt zu erzeugen/instanziieren**
- wenn wir keinen selber schreiben wird immer der default constructor verwendet
- ist dazu da, den Grundzustand des Objektes zu setzen

Default Constructor

```
Titelfolie title1 = new Titelfolie();
```

Name der Klasse, keine Parameter

```

1 class Titelfolie{
2
3     // Klassen Attribute
4     String titleText;
5     String subtitleText;
6
7     // Konstruktor
8     Titelfolie(String title, String subtitle){
9         this.titleText = title;
10        this.subtitleText = subtitle;
11    }
12
13    Titelfolie(String title){
14        this.titleText = title;
15    }
16
17    // Klassen Funktionalitäten
18    void setTitleText(String text){

```

- wird wie eine Methode aber ohne Rückgabe definiert.
- Der Name entspricht immer dem Klassennamen.
- Pro Klasse können mehrere verschiedene definiert werden
- -> verschieden heisst sie müssen sich bei den Parametern unterscheiden

```
folie1 = new Titelfolie("Babedi Bubedi", "WM ohne Italien");
```

wird wie eine Methode benutzt, nur das die Rückgabe immer ein neues Objekt vom genannten Typ ist.

Mit einer Klasse definieren wir einen neuen Datentyp

```
Titelfolie title1 = new Titelfolie();
```

```
Titelfolie folie1 = new Titelfolie("Babedi Bubedi", "WM ohne Italien")
```

Das heisst wir können einen Array erstellen

```
Titelfolie[] folien = new Titelfolie[10];
```

Datentyp

```
folien[0] = new Titelfolie("Title", "Sub");
```

Konstruktor
Aufruf -> neues
Objekt

Und generell wie Datentypen benutzen die wir schon kennen

```
powerpoint_example | Presentation | Titelfolie ▼
1 class Presentation{
2   Titelfolie title;
3
4   Presentation(String title, String subtitle){
5     this.title = new Titelfolie(title, subtitle);
6   }
7
8   Presentation(Titelfolie title){
9     this.title = title;
10  }
11
12  Titelfolie getTitle(){
13    return title;
14  }
15 }
```

Konstruktor
Aufruf -> ein
Objekt wird
erstellt

Übung 11 Klassen benutzen

„Oft schiessen trifft das Ziel.“

– SPRICHWORT



Was wir bis jetzt haben...

- Öffne den Sketch und füge da einen gelben Kreis hinzu, den man über die Tastatur bewegen kann.
- Füge 1 rotes Quadrat hinzu über die man mit dem Kreis nicht drüber fahren kann.



- Erstelle eine Klasse Pacman die deine Pacman Figur repräsentieren soll.
- Welche Attribute brauchen wir für diese Repräsentation?
- mit den Methoden: `moveLeft()`, `moveRight()`, `moveUp()` und `moveDown()` die die Figur in diese Richtungen bewegen sollen

ARRAYLIST - „FLEXIBLER ARRAY“

- Nachteil Array: die Anzahl Elemente sind fix
- Lösung: Datenstruktur `ArrayList<Typ>`
- Klasse von Java, die es ermöglicht eine Sammlung von Elementen flexibler zu verwalten

Datentyp der Elemente
der Sammlung

Variablenname

```
// Deklaration  
ArrayList<Pacman> pacList;  
// Erstellung  
pacList = new ArrayList();
```

```
// Elemente hinzufügen  
Pacman one = new Pacman(0,0,1);  
pacList.add(one);  
pacList.add(new Pacman(2,3,2));  
// Zugriff auf Elemente  
Pacman another = pacList.get(1);
```

```
// Erstellung  
pacList = new ArrayList();
```

neues ArrayList Objekt, mit Funktionalität eine Sammlung vom Typ Pacman zu speichern aber noch leer.

pacList



```
// Elemente hinzufügen  
Pacman one = new Pacman(0,0,1);  
pacList.add(one);  
pacList.add(new Pacman(2,3,2));  
// Zugriff auf Elemente  
Pacman another = pacList.get(1);
```

pacList

Pacman Objekt one

x: 0

y: 0

size: 1

pacList

Pacman Objekt one

x: 0

y: 0

size: 1

Pacman Objekt

x: 2

y: 3

size: 2

