Wir programmieren ein Spiel mit Java!

Kommentare

```
// Ich bin ein Kommentar. Ich werde nicht ausgeführt.

/*
Ich bin ein langer Kommentar,
der über mehrere Zeilen hinaus geht.
Ich werde auch nicht ausgeführt.
 */
```

Start-Methode

```java
/**
 * Ich bin JavaDoc. Ich erkläre den Sinn von Programmbestandteilen.
 * Die Start-Methode wird beim Start des Programms ausgeführt.
 *
 * @param args besondere Start-Informationen für das Programm.
 *             Werden normalerweise nicht gebraucht.
 */
public static void main(String[] args) {
    // Hier beginnt das Programm. Führe hier Befehle aus.
}
```

Befehle

- Java ist imperativ. Das bedeutet, dass man dem Computer in Java Befehle zum Ausführen gibt.
- Befehle enden immer mit einem Semikolon (;)
- Befehle werden von oben nach unten und von links nach rechts ausgeführt.
  - Ausnahmen sind Methoden, die nur beim Aufruf ausgeführt werden.

Primitive Datentypen

Name	Bedeutung	Werte
byte	Speichereinheit	[-128 bis 127] (2^8)
short	Ganze Zahl	[-32768 bis 32767] (2^16)
int	Ganze Zahl	± 2 Milliarden (2^32)
long	Ganze Zahl	± 9 Trillionen (2^64)
float	Fließkommazahl	ca. 7 Ziffern
double	Fließkommazahl	ca. 16 Ziffern
boolean	Wahrheitswert	true (wahr) oder false (falsch)
char	Einzelnes Zeichen	z.B. 'a', '7', 'N', '-', '#' oder '('

Variablen

```java
// Variablen deklarieren: <Datentyp> <Name>( = <Wert>);
int sterneAmHimmel = 9095;
boolean sterneLeuchten = true;
char zweiterBuchstabe = 'B';
float temperaturInGradCelsius = 20.21f;

// Variablen verwenden: <Name>
System.out.println(sterneAmHimmel); // Ergebnis: 9095

// Variablen verändern: <Name> = <Neuer Wert>;
sterneAmHimmel = sterneAmHimmel + 1;
System.out.println(sterneAmHimmel); // Ergebnis: 9096
```

Objekte

Name	Bedeutung	Werte
String	Zeichenkette	Beliebig viele Zeichen (Text)
Scanner	Eingaben-Leser	Informationen zum Konsole-Lesen
Array	Reihe	Beliebig viele Variablen

… weitere selbst definierte

Objekte sind immer Instanzen von Klassen. Mehr dazu unter „Objektorientierung“

Scanner

```java
// Erstelle einen Scanner. Er soll aus der Konsole (System.in) lesen.
Scanner scanner = new Scanner(System.in);

// Eingaben einlesen
System.out.println(scanner.nextLine());
```

Objektvariablen

```
// Variablen deklarieren: <Datentyp> <Name> = <Wert>;
String nameDesHellstenSterns = "Sirius";
Scanner scanner = new Scanner(System.in);
int[] lieblingszahlen = {2, 13, 42, 1111};
```

Wenn man Objektvariablen keinen Wert zuweist, haben sie den Wert null.
Beim Zugriff auf eine Objektvariable mit dem Wert null, also Zugriff auf eine Instanzmethode oder ein Attribut des (nicht existenten) Objekts, stürzt das Programm mit einer NullPointerException ab.

Typen von Variablen

- Variablen primitiver Datentypen speichern diese (byte, short, int, long, float, double, boolean, char) direkt.
- Objektvariablen speichern Referenzen auf Objekte (z.B. String, Scanner, Random). Objekte können in mehreren Variablen referenziert werden.

Begriffe - Variablen

Begriff	Bedeutung	Syntax
Variablen deklarieren	Variablen im Code erstellen.	<Datentyp> <Name>;
Variablen setzen / Werte zuweisen	Variablen verändern	<Name> = <Neuer Wert>;
Variablen initialisieren	Variablen erstmalig einen Wert zuweisen.	^^

Man kann Deklaration und Initialisierung kombinieren:

<Datentyp> <Name> = <Wert>;

Was sind Methoden?

Methoden können zwei Dinge tun:

(1) Befehle ausführen

(2) Ergebnisse liefern

Methoden verkapseln Programmcode.

```
// Verwendung einer Methode:
// [Objekt.]<Name>(<Parameter>);
```

```java
// 1 = Befehle ausführen, 2 = Ergebnisse liefern

// println (1) gibt "Hallo" in der Konsole aus.
System.out.println("Hallo");

// length (2) liefert die Länge eines Strings.
int buchstabenInLangemWort = "Heizölrückstoßabdämpfung".length();

// contains (2): Ist "affe" ein Teil von "Giraffe"? Ja.
boolean affeInGiraffe = "Giraffe".contains("affe");

// nextLine (1&2): Wartet auf Eingabe und liefert die Eingabe.
String next = scanner.nextLine();

// equals (2): hat "Feuer" die gleichen Buchstaben wie "Wasser"? Nein.
boolean feuerIstWasser = "Feuer".equals("Wasser");
```

Warum eigene Methoden schreiben?

- Wenn wir eine Methode einmal geschrieben haben, können wir sie immer wieder verwenden. Daher müssen wir nie ähnlichen Programmcode doppelt schreiben.
- Methoden haben immer einen Namen. Über diese kann man einfacher verstehen, was der Zweck eines Teils des Programmcodes ist.
- Wir können Methoden unabhängig von anderem Programmcode schreiben. So können wir ein großes Spiel in viele kleine Teilprobleme zerlegen.

Wie schreiben wir Methoden?

Dafür müssen wir über diese Dinge nachdenken:

```
// Struktur einer Methode:
// <Modifikatoren> <Rückgabetyp> <Name>(<Parameter>) {
//     <Befehle>
// }
```

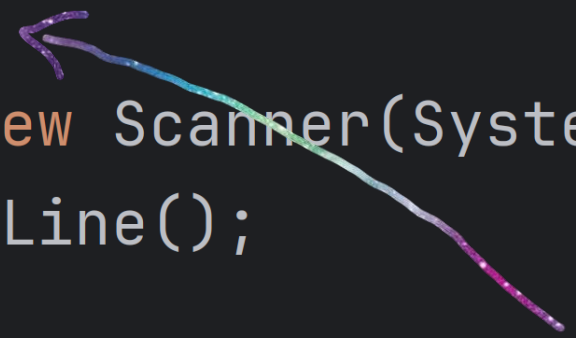
Ergebnis der Methode festlegen

- Mit dem „return“-Schlüssenwort bestimmen wir, welches Ergebnis unsere eigene Methode liefern soll.
- Das Ergebnis ist rechts von dem „return“-Schlüsselwort
- Methoden mit dem Rückgabetypen „void“ liefern kein Ergebnis. Sie führen nur Befehle aus.
- Nach Verwendung des „return“-Schlüsselworts ist die Methode vorbei. Es werden keine Befehle mehr ausgeführt.

```java
private static void gibAus(String text) {
    System.out.println(text);
}

private static String frag(String text) {
    gibAus(text + "?");
    Scanner scanner = new Scanner(System.in);
    return scanner.nextLine();
}

private static void begrüßen() {
    String name = frag("Wie heißt du?");
    System.out.println("Hallo " + name + "!");
}
```



Doppeltes Fragezeichen

Methoden und Funktionen

- In Java bedeuten die Begriffe das gleiche.
- In einigen anderen Sprachen:
  - Methoden führen Befehle aus, die den Zustand des Programms.
  - Funktionen dienen ausschließlich dazu, Ergebnisse zu liefern.

Begriffe - Methoden

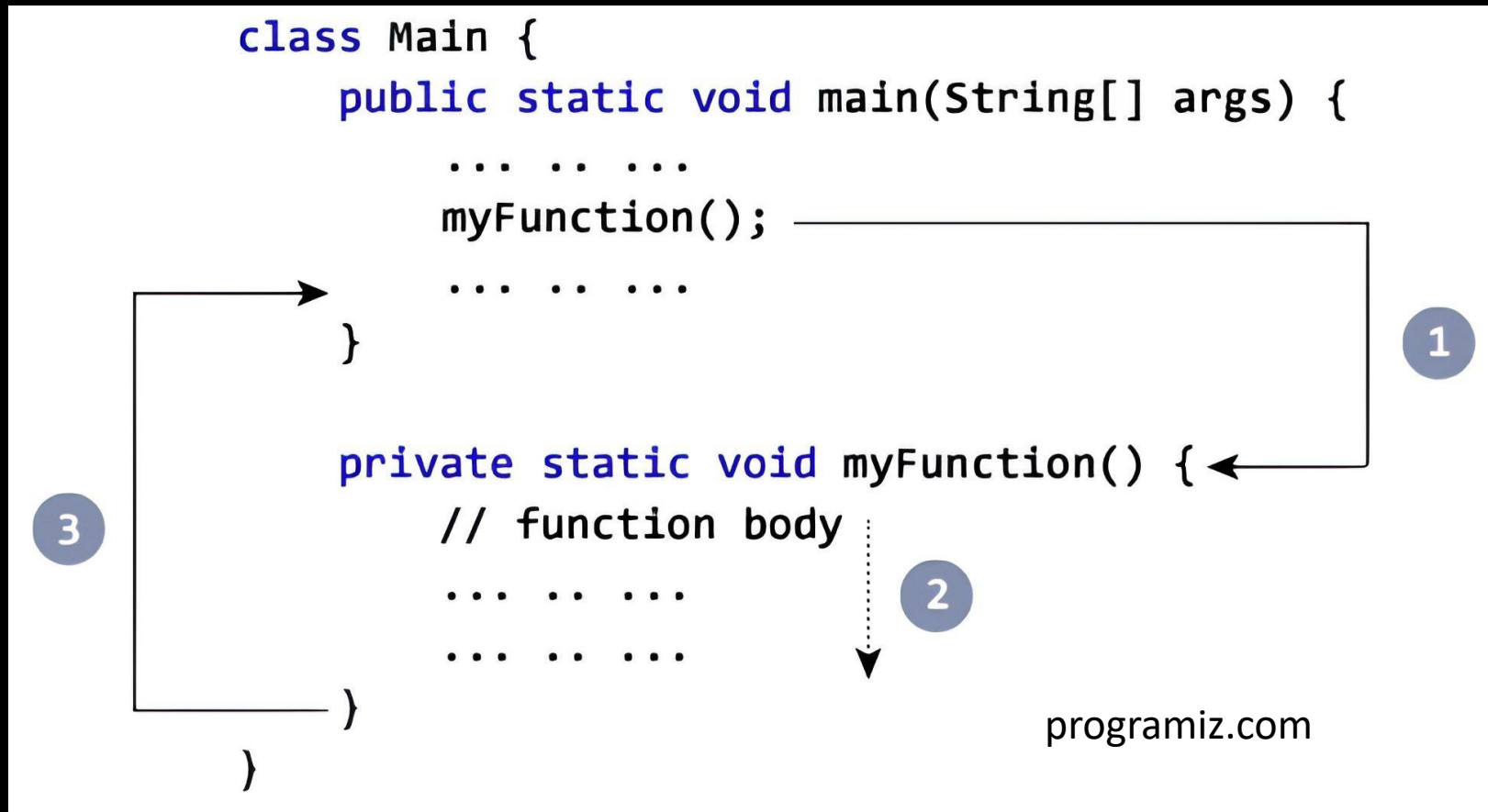
Begriff	Bedeutung
Methode deklarieren	Methoden selbst schreiben. Wir legen die Befehle fest, die die Methode ausführen soll.
Methode aufrufen / ausführen	Methoden verwenden. In den Klammern legen wir hier die Parameter für die Methode fest.
Methoden überschreiben	In abgeleiteten Klassen: Methoden neu deklarieren. (siehe Vererbung)

Arten von Methoden

Art	Bedeutung	Beispiel
Instanzmethode	Methode, die auf einem Objekt ausgeführt wird	Methode „nextLine“ der Klasse „Scanner“
Statische Methode	Methode, die immer, unabhängig von Objekten, verwendet werden kann	Methode „begrüßen“

Nicht verwechseln: Deklaration und Aufruf

- Bei der Deklaration wird die Methode definiert. Dabei entscheiden wir, was die Methode tut.
- Beim Aufruf einer Methode wird sie nur verwendet.
- Bild: Ablauf eines Methodenaufrufes



Arten von Variablen

Art	Bedeutung	Beispiel
Lokale Variable	Variable innerhalb eines Blocks. (Methoden sind auch ein Block)	Um die kleinste Zahl aus einem Array zu finden, verwenden wir eine Schleife. Wir speichern dabei immer die aktuell kleinste Zahl.
Attribut / Instanzvariable	Variable innerhalb einer Klasse. Sie wird für jedes Objekt einmal gespeichert.	Die Klasse „Baum“ könnte das Attribut „Höhe“ haben. Jeder Baum hat eine andere Höhe.
Statische Variable / Klassenvariable	Variable innerhalb einer Klasse. Sie speichert, unabhängig vom Projekt, nur einen Wert.	Die Zahl π ist unabhängig vom Objekt immer gleich.
Parameter / Argument	Variable, die einer Methode übergeben wird.	Die Methode „frag“ bekommt den Parameter „text“, um den Fragetext zu kennen.

Sichtbarkeitsmodifikatoren

Können für Attribute und Methoden verwendet werden:

Schlüsselwort	**Bedeutung**
private	Kann nur innerhalb der Klasse verwendet werden. So können wir sie absichern.
<kein Modifikator>	Kann im gleichen Ordner (package) verwendet werden.
protected	Kann im gleichen Ordner (package) und abgeleiteten Klassen (siehe Vererbung) verwendet werden.
public	Kann überall verwendet werden

final-Modifikator

- Bei Variablen: Sorgt dafür, dass man keinen neuen Wert zuweisen können
  - Achtung: Referenzierte Objekte können ihren Zustand ändern, es können nur keine neuen Objekte zugewiesen werden.
- Bei Methoden: Sorgt dafür, dass die Methode nicht überschrieben werden kann (siehe Vererbung)
- Bei Klassen: Sorgt dafür, dass die Klasse keine

Eingabe von Symbolen

Name	Kombination	Symbole
Semikolon, Doppelpunkt	Shift + Komma (,) / Punkt (.)	; :
Anführungsstrich(e)	Shift + Hashtag (#) / 2	' "
Runde Klammern	Shift + 8/9	()
Eckige Klammern	Alt Gr + 8/9	[]
Geschweifte Klammern	Alt Gr + 7/0	{}
Ausrufe/Fragezeichen	Shift + 1/ß	! ?
Und, Gleich	Shift + 6/0	& =
Senkrechter Strich	Alt Gr + <	|
Größer	Shift + <	>
Unterstrich	Shift + -	_

Tastatur-Shortcuts

Name	**Kombination**
Element auswählen	Strg + W
Alles auswählen	Strg + A
Kopieren	Strg + C
Einfügen	Strg + V
Löschen und Kopieren	Strg + X
Duplizieren	Strg + D
Rückgängig	Strg + Z
Rückrückgängig	Strg + Y
Suchen	Strg + F
Suchen und Ersetzen	Strg + R

IntelliJ-Shortcuts

Name	**Kombination**
Nach oben / unten scrollen	Strg + ↑↓
Mauszeiger nach links / rechts bewegen	Strg + ←→
Zu Definition / Verwendung springen	Strg + B
Methoden überschreiben	Strg + O
Schnell durch das Projekt navigieren	2x Shift
Programm starten	Shift + F10
Code formatieren (schön einrücken)	Strg + Alt + L
Zurück zum vorherigen Mauszeiger	Strg + Alt + ←→
Zu Mauszeiger springen	Strg + M

Operatoren und Ausdrücke

Name	Syntax	Ergebnis
plus	a + b	Zahl
minus	a – b	Zahl
mal	a * b	Zahl
geteilt	a / b	Zahl
kleiner als	a < b	boolean
größer als	a > b	boolean
und (prüfen, ob zwei Dinge gelten)	a && b	boolean
oder (prüfen, ob mindestens eins von beiden gilt)	a || b	boolean
gleich (prüfen, ob zwei Dinge dieselben sind)	a == b	boolean

String

- Ein String ist eine Zeichenkette.
- Er kann durch Anführungsstriche erstellt werden: "Hallo"
- Um einen beliebigen Buchstaben auszulesen, gibt es die Methode charAt(int index), die Indizes starten bei 0!
  - Achtung: Wenn der Index negativ oder größer/gleich der Länge des Strings ist, stürzt das Programm mit einer StringIndexOutOfBoundsException ab.
- Um die Länge eines Strings zu erfahren, verwende length().
- Um Strings untereinander oder mit anderen Werten zu kombinieren, verwende das Plus-Zeichen: stringA + variableB

System

- Die System-Klasse bietet nützliche Attribute und Methoden:
  - Über System.out kann mit der Methode println(String s) etwas in der Konsole ausgegeben werden.
  - Über System.in können mit Hilfe des Scanners Eingaben aus der Konsole eingelesen werden.
  - System.currentTimeMillis() liefert die Anzahl an Millisekunden (also 1000stel Sekunden) seit dem 1. Januar 1970.