Einheit 2

Klassen

Klassen

- Datentypen (können auch selbst definiert werden)
- Fassen mehrere Elemente zu einer Einheit zusammen
- Klassen können unterschiedliche Elemente beinhalten (Arrays nur gleichartige)
- Klassen werden in JAVA **IMMER GROß** geschrieben

2

Klassen

- repräsentieren Dinge der realen Welt auf abstrahiertem Niveau
- Übergeordnetes Schema

```
public class Person {
   String vorname;
   String nachname;
}
```

Ausgehend vom Schema werden Objekte (Instanzen) erzeugt/instanziert

```
Person muster = new Person();
```

3

Objekte

- Ein Objekt ist ein einzelnes Exemplar einer Klasse
  - `muster`
- Objekte entsprechen dem Schema der Klasse
- Repräsentieren einzelne Objekte der realen Welt

4

Objekte

- Instanzierung erfolgt mit Konstruktoren und Schlüsselwort **`new`**
- Beispiel Klasse Person:
  Instanzierung:

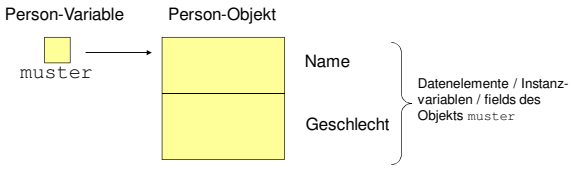
```
Person muster = new Person();
```

- Man sagt auch: der Datentyp der Variable `muster` ist die Klasse Person

5

Objekte

- Objektvariablen (Referenzvariablen) enthalten Zeiger auf Objekte im Speicher



6

Konstruktor

- Dient zum Erzeugen neuer Objekte (Instanzen)
- Heißt wie die Klasse, initialisiert Attribute
- Verschiedene Konstruktoren möglich
  - Unterscheidung durch Art und Anzahl der übergebenen Parameter

```
public Person();
public Person(String vorname, String nachname);
```

- Aufruf mit Operator `new`

```
new Person();
new Person("Max","Muster");
```

7

this

- Mit Schlüsselwort this greift das Objekt auf sich selbst zu
- Ermöglicht:
  - Zugriff auf eigene Attribute
  - Zugriff auf eigene Methoden
  - Bsp. Person:
    - Initialisieren der Attribute im Konstruktor

```
this.vorname = vorname;
this.nachname = nachname;
```

8

Zeiger

- Objektvariablen können mit null initialisiert werden

```
Person z = null;
```

- wird als „kein Objekt“ gelesen und kann abgefragt werden
- Zwei Variablen des gleichen Typs können einander zugewiesen werden

```
Person neu = muster;
```

9

Datenkapselung

- Reduktion der Abhängigkeit zwischen Modulen (Implementierung ist für Aufrufer irrelevant)
- ist über Modifier realisiert
  - `public String name; // von überall zugreifbar`
  - `private String vorname; /* nur innerhalb der Klasse sichtbar */`
  - `int alter; // innerhalb des Packages zugreifbar`

10

Objekte und Instanzvariable

- Instanzvariablen stellen Attribute des Objekts dar
- Je Objekt unterschiedliche Werte
- Machen Objekte unterscheidbar
- Instanzvariablen über Punkt-Notation ansprechbar

```
String name = muster.nachname;
```

11

Methoden

- Klassen stellen Methoden zur Verfügung. Über diese kann das Objekt mit der Umwelt kommunizieren (Botschaften empfangen und senden).
- Methoden sind über Punkt-Notation ansprechbar
- Klassische Methoden:
  - „Getter“ und „Setter“ zum Bearbeiten (Schreiben bzw. Lesen der Instanzvariablen)

```
String name = muster.getName();
muster.setAlter(40);
```

12

Überladung von Methoden

- Gleicher Methodenname in der selben Klasse möglich, wenn Art oder Anzahl der Parameter unterschiedlich sind
- Beim Aufruf wird automatisch die Methode gewählt, die zu den übergebenen Parametern passt

```
muster.setPerson("Karl");
muster.setPerson("Susi","Wimmer");
```

13

Statische und objektbezogene Variablen und Methoden

- Klassen sind selbst Objekte
- Sowohl Klassenobjekt als auch Instanzobjekt haben Variablen und Methoden
- Klassenvariablen sowie Klassenmethoden weisen dem Modifier `static` auf.
- Klassenvariablen
  - weisen nur einen Wert pro Klasse und Virtual machine auf

```
public static int zaehler = 0;
```

14

Zugriff

- Zugriff auf statische Methoden ohne konkrete Objekte möglich (man benötigt für deren Aufruf keine Instanz der Klasse).
- Aufruf funktioniert über den Klassennamen

```
Person.zaehler;
```

- Instanzmethoden können auf statische Variablen/Methoden zugreifen, aber nicht umgekehrt

```
public int getInfo() {
     return Person.zaehler;
}
```

15

Zusammenfassung

	Objektbezogen	Statisch
Deklaration	ohne static	mit static
Existenz	in jedem Objekt	einmal pro Klasse
Ansprechen der Variablen	objekt.variable this.variable	Klasse.variable
Methodenaufruf	objekt.methode(); this.methode();	Klasse.methode();

16