3 Konstruktoren

Ein Konstruktor ist eine Methode, welche eine Instanz der zugehörigen Klasse erzeugt. Diese Instanz nennen wir Objekt – den Vorgang „Instanziieren“. Insofern hilft folgender Vergleich bei der grundsätzlichen Unterscheidung:

Die *Klasse* ist der *Bauplan*
Das *Objekt* ist das *Haus*

Bei der Erzeugung von Objekten sind nun zwei Dinge wesentlich. Erstens das offensichtliche – das Objekt muss aus der Klasse instanziiert werden. Zweitens müssen die internen Eigenschaften vorbereitet werden. Beispielsweise wollen wir, dass bei der Erzeugung eines Kreis-Objektes die Farbe und der Radius definiert sein müssen. Wir können dem Computer nicht sagen „erstelle einen Kreis mit *irgendeiner* Farbe und *irgendeinem* Radius“. Diese Informationen müssen also über die Parameterliste bereitgestellt werden. Als Faustregel gilt:

„Alle für den sinnvollen Betrieb des Objektes notwendigen Informationen sollten dem Konstruktor zugeführt werden.“

Insofern ist der Konstruktor für unser Kreisobjekt eine Methode, welche die Parameter Farbe und Radius enthält. Ergänzen Sie den folgenden Code:

```
public class Kreis extends Geo2DObjekt {
  double radius;
}
```

Um nun ein Objekt der Klasse Kreis zu erzeugen, müssen wir das Schlüsselwort „`new`“ vor den Konstruktor setzen:

```
Kreis myKreis = new Kreis(0xff00aa, 10.0);
```

Nun gibt es aber noch einen wichtigen Punkt: Wenn der Konstruktor festlegt, welche Informationen zwingend notwendig sind und somit über die Parameter zugeführt werden müssen, so gilt dies auch für die Elternklassen. Die Klasse Geometrieobjekt müsste also die Übergabe des Farbwertes erzwingen.

Ergänzen wir also die Klasse Geometrieobjekt um einen Konstruktor:

```
public class Geometrieobjekt {
  int farbe;
  public Geometrieobjekt(int farbeInit) {
    farbe = farbeInit;
  }
}
```

Wenn nun gilt, dass alle Kindklassen von Geometrieobjekt die Eigenschaften hiervon erben, so müssen sie auch die Notwendigkeit des Farbparameters im Geometrieobjekt Konstruktor erben! Dies zwingt uns also, dass alle direkten Erben den Konstruktor der Elternklasse, den „Superkonstruktor", aufrufen müssen:

```
public class Geo2DObjekt extends Geometrieobjekt {
  public Geo2DObjekt(int farbeInit) {
    super(farbeInit);
  }
}
```

In Java muss dieser „`super()`" Aufruf immer an erster Stelle des Kindkonstruktors stehen – allerdings nur dann, wenn der Elternkonstruktor mindestens einen Parameter aufweist. Ergänzen Sie nun mit einer anderen Farbe ihren Konstruktor der Klasse Kreis um den Superkonstruktor Aufruf.

Das Schlüsselwort „`super`" ist auch für den Fall, dass eine Variable in der Elternklasse gleich heißt wie in der Kindklasse, wichtig. Wenn zusätzlich noch eine lokale (Parameter-) Variable gleichen Namens existiert, müssen wir diese unterscheiden können. Im folgenden Code gibt es „`farbe`" dreimal:

```
public class ZweiFarbenObjekt extends Geometrieobjekt {
  int farbe;
  public ZweiFarbenObjekt(int farbe, int farbe2) {
    super.farbe = farbe;
    this.farbe = farbe2;
  }
}
```

- Lokale Variable – ohne Zusatz
- Instanzvariable – `this.` (auch Elternvariablen, wenn keine Namensgleichheit mit eigenen)
- Elternvariable – `super.`

Die Frage ist nun – benötigt man immer einen Konstruktor. Hier gibt es zwei Antworten. Zur Erzeugung eines Objektes benötigt man immer einen Konstruktor. Also aus „Nutzersicht" einer Klasse ist die Antwort „ja". Beim Schreiben einer Klasse müssen wir nur dann einen Konstruktor erstellen, wenn wir interne Vorbereitungen für die Nutzung des Objektes durchführen müssen. Ist dies nicht der Fall, so kann man auf das Schreiben eines Konstruktors verzichten. Java würde dann implizit einen Konstruktor vorsehen, der bei der Objekterzeugung (siehe „Nutzersicht") aufrufbar ist. Diesen Konstruktor nennen wir „Standardkonstruktor".

Für die Definition eines Konstruktors in Java gelten nun die folgenden Punkte:

- Alle essenziellen Eigenschaften sollten über die Parameterliste übergeben werden
- Alle von diesen Eigenschaften abhängigen Eigenschaften müssen ermittelt werden
- Der Konstruktor gibt immer ein Objekt der zugehörigen Klasse zurück
- Der Konstruktor heißt in Java genauso, wie die Klasse

Während die ersten Punkte für alle objektorientierten Sprachen gelten, ist der letzte Punkt für Sprachen wie JavaScript, PHP oder Python anders. So heißt der Konstruktor in JavaScript bspw. `constructor()`.