Christian Reisenauer

JAVA Weihnachtsübung - Theorie

Was ist das Ziel der Objektorientierung?

Objektorientierung hat das Ziel, die „reale Welt“ in Programmcode darzustellen. Dadurch ermöglicht Objektorientierung sehr praxisnahes und effizientes Programmieren (als Entwickler kann man Klassen entsprechend dem „realen“ Objekt designen) und verbessert zugleich noch die Lesbarkeit und Wartbarkeit von Software, da nicht eine Datei mit tausenden Zeilen Code entsteht, sondern der Programmcode auf viele Klassen aufgeteilt wird, wobei jede Klasse entweder eine einzige Aufgabe hat (z. B. Main-Klasse „Application“) oder ein Objekt beschreibt (z. B. Klasse „Auto“).

Gibt es OOP nur in Java?

Nein. Mittlerweile gibt es OOP in sehr vielen Programmiersprachen, z. B. PHP, Python und sogar JavaScript!

Was ist der Unterschied zwischen Objekt und Klasse?

Eine Klasse ist der „Bauplan“ für ein Objekt. Eine Klasse „beschreibt“ quasi ein Objekt, legt dessen Attribute und Methoden fest => es beschreibt das „Aussehen“ und das „Verhalten“ des Objektes. Aus einer (nicht abstrakten) Klasse kann man dann beliebig Objekte instanziieren (= erzeugen), diese werden auf dem Heap-Speicher abgelegt. Jedes Objekt kann – innerhalb der Vorgaben durch die Klasse – verschiedene Attribute haben.

Wie erzeuge ich eine neue Instanz? (Welches Schlüsselwort gibt es dafür)

Eine Instanz eines Objektes wird in Java wie folgt erzeugt:

Dingsbums o = new Dingsbums(…);

Es wird eine Objektvariable mit dem Datentyp „Objekt“ (der Datentyp des Objekts bzw. der Klasse) angelegt. Auf der Seite der Zuweisung (rechts vom „=“) wird mittels dem Schlüsselwort new ein Objekt instanziiert und Speicherplatz auf dem Heap reserviert. Dann wird der Konstruktor des Objekts aufgerufen, der das Objekt tatsächlich erfolgt Dingsbums(…) – einem Konstruktor können natürlich Parameter übergeben werden.

Was bedeutet das Schlüsselwort static und wo kann es überall verwendet werden?

„static“ bedeutet in Java, dass für die Verwendung einer Variable/Methode kein Objekt instanziiert werden muss, d. h. die Methode kann mittels „Klasse.methodenName()“ aufgerufen werden!

Nicht-statische Methoden benötigen immer ein Objekt, dass sie aufruft (z. B. „auto.setColor(blue);“)

Statische Methoden werden üblicherweise für Hilfsklassen verwendet (Kalkulationen, Timestamps, …), da es sehr aufwändig ist, Instanzen dafür extra zu erzeugen. Zudem ist es rechenintensiver, für alles Objekte zu erstellen (Heap!).

Wozu dient die Vererbung?

Vererbung ermöglicht es in der Objektorientierung, dass man Klassen weiter Spezifizieren kann. Dabei gibt es eine Elternklasse und eine oder mehrere Kindklassen. Die Kindklassen erben alle Attribute und Methoden von der Elternklasse (außer jene mit dem Modifikator „private“) und können um weitere, spezifische Attribute und Methoden erweitert werden.

Simples Beispiel:
Klasse Fahrzeug, Kindklassen PKW und Motorrad => beide haben eine Leistung (in PS), aber ein Auto hat vier Reifen, während ein Motorrad nur zwei davon hat!

Kann in Java von mehreren Klassen geerbt werden? Wenn ja wie?

Theoretisch gibt es in Java keine Mehrfachvererbung! Ein ähnlicher Effekt kann mit Interfaces erzielt werden. Interfaces geben Methoden vor, die eine Klasse, die ein IF benutzt, zwingend implementieren muss.

Welche Vererbungshierarchien kennst du? (Ein Bild reicht aus)

In Java gibt es bei der Vererbung immer eine Superklasse (Elternklasse) und mindestens eine oder mehrere Subklassen (Kindklassen), die von der Elternklasse erben. Vererbt werden Instanzvariablen und -methoden, die nicht auf „private“ gesetzt sind. Mehrfachvererbung ist grundsätzlich nicht möglich (Ausnahme Interfaces).

Eine Elternklasse kann beliebig viele Kindklassen haben, eine Kindklasse aber nur eine Elternklasse.

Was bedeutet Casten und wie ist die Syntax in Java dafür?

Unter „Casten“ versteht man eine Typumwandlung, d. h. die Umwandlung eines Datentyps in Java.
Es gibt explizites und implizites Casten.

Beispiel für exp. Casting:
int summe = (int) zahlEins + (int) zahlZwei;

Was ist der Unterschied zwischen explizitem und impliziertem Typecasting?

Explizites Casting wird durch den Entwickler explizit ausgedrückt (siehe Beispiel oberhalb).

Implizites Casting ist ein durch den Compiler automatisch vorgenommenes Casting.

Erkläre die folgenden Schlüsselwörter: super, this

„super“: Ist ein Schlüsselwort aus der Vererbung. Es ist eine Referenz auf die Instanzvariable der unmittelbaren Elternklasse. Mittels „super.yourMethod()“ kann eine Instanzmethode der Elterninstanz aufgerufen werden, mittels „super()“ wird der Konstruktor der Elternklasse aufgerufen.

„this“: Dieses Schlüsselwort bedeutet immer eine Referenz auf das aktuelle Objekt. Es findet z. B. Anwendung bei der Zuweisung von Variablen im Konstruktor. „this.wert“ bedeutet, dass die Variable „wert“ von der aktuellen Instanz referenziert wird.

Für was dient der instanceof Operator? Gib ein sinnvolles Beispiel.

Der instanceof-Operator dient dazu, bei Instanzvariablen (Objektvariablen) zu überprüfen, ob diese Variable eine gültige Instanz einer bestimmten Klasse ist. Er gibt entweder TRUE oder FALSE zurück.

Beispiel:

if(meinAuto instanceof PKW)=> True

if(meinAuto instanceof Motorrad) => False