Klassen

Hallo. Du hast von diesen mysteriösen Klassen in Java schon gehört, aber für dich ist das nichts weiter als Hokuspokus? In den nächsten fünf Minuten wird sich das schnell ändern.

Aber was sind eigentlich Klassen und was haben sie mit Objekten zu tun? Eine Klasse kannst du dir wie einen Bauplan vorstellen. Sie legt alle Eigenschaften eines bestimmten Objekttyps fest. Das kann ein simpler Würfel sein, aber auch etwas so komplexes, wie ein Flugzeug. Du kannst mit einem Bauplan beliebig viele Objekte des gleichen Typs bauen und kannst, ähnlich wie große Hersteller, sogar einige benutzerdefinierte Änderungen zulassen. So kannst du Flugzeuge zum Beispiel in allen Farben und Größen bauen oder deine Würfel verschiedene Augenzahlen anzeigen lassen. Außerdem liegt eine Klasse auch das Verfahren, der mit ihr erzeugten Objekte fest. Das liegt daran, dass deren Methoden in ihr definiert sind. Also ist die Klasse nicht nur der Bauplan, sondern auch der Pilot des Flugzeugs und bei Bedarf auch der Entsorgungsbetrieb. Wie du siehst, sind Klassen essentiell zur effizienten Nutzung von Objekten. Doch wie sind sie aufgebaut, um das alles leisten zu können? Um das zu untersuchen, bedienen wir uns der Unified Modeling Language (kurz UML) die uns erlaubt, die als Quellcode recht komplex wirkenden Strukturen einer Klasse schematisch darzustellen. Oben findest du den Klassennamen. Dieser ist immer ein Substantiv im Singular und wird großgeschrieben. Im Feld darunter legen wir eine Reihe von Attributen fest. Davon gibt es zwei verschiedene Typen: Zum einen die normalen Attribute, die für jedes Objekt initialisiert werden und zum anderen Klassenattribute. Diese existieren nur einmal pro Klasse, egal wie viele Objekte wir erzeugen. Sie gelten also für alle Objekte einer Klasse. Klassenattribute existieren sogar bereits bevor auch nur ein einziges Objekt in einer Klasse erzeugt wurde. Ein Beispiel hierfür wäre ein Bestandszähler, der die Anzahl deiner Flugzeuge festhält, damit du weißt, wie viele du noch bauen muss, um deinen Auftrag erfolgreich abzuschließen. Attribute werden immer klein geschrieben Klassenattribute immer unterstrichen. Dabei achten wir darauf, ob wir ein Attribut public, also öffentlich, oder private, also privat, deklarieren. Das ist wichtig, denn dadurch wird festgelegt, ob auf das Attribut auch aus anderen Klassen heraus direkt zugegriffen werden kann. Ob ein Attribut public oder private ist, erkennt man im UML-Diagramm am vorangestellten Plus beziehungsweise Minus und im Quellcode an nur bei privater Deklarierung vorhandenen private vor dem primitiven Datentyp. Außerdem unterscheiden wir zwischen Kann- und Soll-Attributen. Soll-Attribute müssen für jedes Objekt mit einem gültigen Wert initialisiert werden, sonst ist eine Konstruktion nicht möglich. Kann-Attribute hingegen haben keinen Einfluss darauf, ob ein Flugzeug gebaut werden kann. Gibt der Kunde keine Größe an, wird es dir nicht möglich sein, ein Flugzeug zu bauen. Gibt er allerdings nur nicht an, dass er die Sonderausstattung haben möchte, so baust du für ihn einfach die Standardausführung. So, nun haben wir festgelegt, wie unsere Flugzeuge aussehen sollen, aber damit wir sie bauen können, brauchen wir noch Operationen. Operationen einer Klasse werden genauso wie Attribute in Klassen Operationen und normale Operationen eingeteilt. Die Kennzeichnung ist hier ebenfalls dieselbe. Stellen wir uns also vor, wir haben bereits eine Fabrik, die in der Lage ist in der von uns gewünschten Anzahl Flugzeuge zu produzieren. Jetzt müssen wir noch dafür sorgen, dass unsere Mitarbeiter das auch können.

Dazu lernen wir sie erst einmal ein und zeigen ihnen, wie man ein Flugzeug überhaupt herstellt. Als Sicherheitsmaßnahme sollen sie auch noch lernen, was zu tun ist, falls das Herstellen scheitert. Dazu legen wir einen Konstrukteur an. Er bekommt alle Daten, die er zur Konstruktion benötigt und produziert für uns, sofern unsere Angaben gültig waren, ein nagelneues Flugzeug. Sollten die Eingabe-Parameter ungültig gewesen sein oder ein Fehler bei der Produktion aufgetreten sein, so gibt uns der Konstrukteur eine Exception aus. Aber allein mit der Mitarbeiterschulung ist unsere Arbeit nicht getan. Nun müssen wir das Informationsmanagement unserer Firma in Angriff nehmen. Jede Eigenschaft der zu produzierenden Flugzeuge muss jederzeit von jedem Mitarbeiter erfragbar und im Bedarfsfall anpassbar sein. Dazu erstellen wir für jedes Attribut sogenannte getter- und setter-Methoden. Mit Aufruf der getter-Methode eines Attributes kann ein Mitarbeiter Werte abrufen, die mithilfe der setter-Methode dem Attribut einen neuen Wert zuweisen. Außerdem muss vor Produktionsbeginn überprüft werden können, ob alle Soll-Attribute einen gültigen Wert besitzen. Dafür fügen wir noch extra sogenannte Check-Operationen hinzu.

Bravo, unsere Fabrik funktioniert nun perfekt. Es gibt jedoch einen wichtigen Unterschied zwischen realen und digitalen Objekten, der dir beim Programmieren sehr gelegen kommen wird: Im Gegensatz zu realen Flugzeugen, die einmal produziert unveränderlich sind, können wir unsere virtuellen erzeugten Flugzeuge jederzeit den Wünschen unseres Kunden anpassen. Während der Programmlaufzeit lassen sich nämlich alle Attribute, für die wir das erlaubt haben, innerhalb der gültigen werte ändern. So, nun steht einem Testflug in deinem neuen Flugzeug wohl nichts mehr im Wege. Du solltest nur daran denken vorher noch die Steuerung zu programmieren. Machs gut!