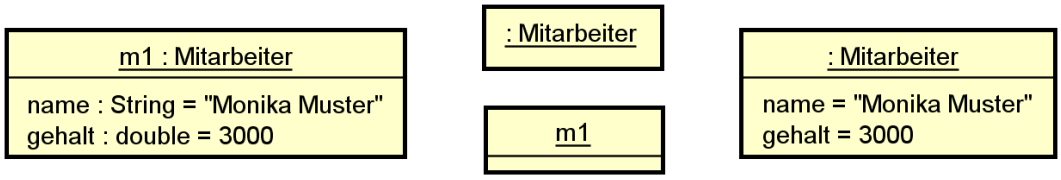
UML Objektdiagramm

Ein Objektdiagramm ist ein **Strukturdiagramm**. Es sieht auf den ersten Blick dem Klassendiagramm sehr ähnlich. Folgendes Bild zeigt verschiedene Möglichkeiten, ein Objekt namens `m1` der Klasse `Mitarbeiter` darzustellen:

Darstellungsarten von Objekten



Es gibt grundsätzlich drei Möglichkeiten, ein Objekt zu bezeichnen:

Objektbezeichner

`objektname:Klasse`	vergibt einen Namen für das Objekt und zeigt Klasse an
`:Klasse`	anonymes Objekt ohne Angabe von Objektnamen
`objektname`	wenn der Objektname ausreicht und der Name der Klasse aus dem Kontext klar ist

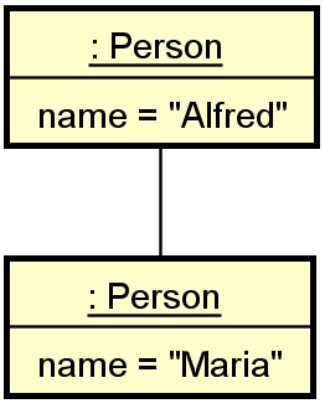
Objektname, Datentypen, Sichtbarkeiten

Wichtig ist, dass der Bezeichner **unterstrichen** ist. Dies unterscheidet das Objekt- vom Klassendiagramm. Der Objektname dient der Unterscheidung zwischen mehreren Objekten. Bei Attributen kann optional der Datentyp angegeben werden. Dies macht dann Sinn, wenn er nicht anhand des Werts ersichtlich ist. Wenn es für die Aufgabenstellung relevant ist, können auch Sichtbarkeiten angegeben werden. Methoden werden nicht ins Diagramm aufgenommen, da sie für alle Objekte gleich sind. Sie können im Klassendiagramm nachgesehen werden.

Beziehungen und Assoziationen

Der Objektzustand wird nicht nur durch die Attributwerte bestimmt, sondern auch durch die aufgebauten **Objektbeziehungen**. Während Attributwerte (Objekt) den konkreten Zustand eines Attributs (Klasse) angibt, geben Beziehungen (Objekt) den Zustand einer Assoziation (Klasse) an.

Objektbeziehung zweier Objekte

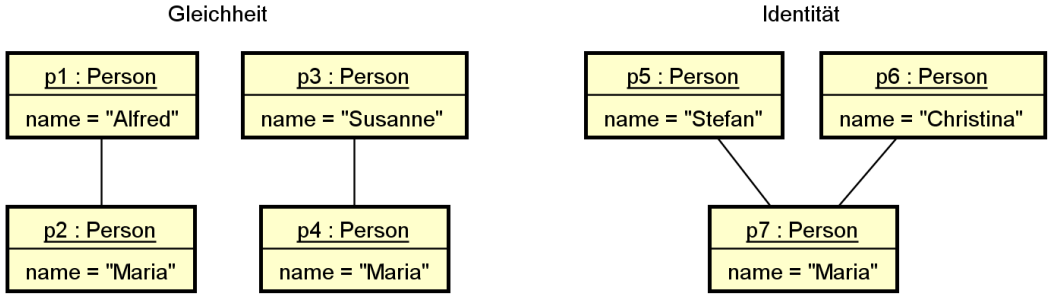


In diesem Fall existiert eine Objektbeziehung zwischen einer Person mit dem Namen `Alfred` und einer Person mit dem Namen `Maria`. Dies bedeutet, dass die Klasse `Person` eine Assoziation mit sich selbst besitzen muss – eine sogenannte reflexive Assoziation.

Objektidentität

Ein Objekt besitzt immer eine Identität. Die Identität unterscheidet das Objekt von allen anderen Objekten, auch wenn die Attributwerte gleich sind. Folgendes Bild veranschaulicht den Unterschied zwischen **Gleichheit** und **Identität**:

Gleichheit vs. Identität



Links sieht man, dass Alfred und Susanne jeweils ein Kind haben. Beide Kinder heißen „Maria“. Die Objekte `p2` und `p4` sind daher *gleich*, weil alle Attribute denselben Wert besitzen. Sie sind jedoch **nicht** *identisch*, da es sich um zwei unterschiedliche Objekte handelt. Rechts sieht man, dass Stefan und Christina zusammen ein Kind namens Maria haben. Sie haben daher zwei Beziehungen zum *identischen* Objekt `p7`.

equals() und == bei Strings

Für uns zeigt sich der Unterschied zwischen Gleichheit und Identität beispielsweise beim Vergleich von `String`-Objekten. Da Strings Objekte sind, besitzen sie eine eigene Identität. Möchten wir überprüfen, ob zwei Strings dieselben Zeichen beinhalten, müssen wir daher ihre Gleichheit über die `equals()`-Methode überprüfen. Ein Vergleich mithilfe von „==“ überprüft nämlich die Identität, welche uns bei Strings normalerweise nicht interessiert. Manchmal

funktioniert der Vergleich über „==“ bei Strings trotzdem und liefert dasselbe Ergebnis wie `equals()`. Das liegt daran, dass in Java manche `String`s in einem `String`-Pool gecached werden – nämlich jene, die als String-Literale im Sourcecode vorhanden sind (also z.B. `String s = "Hallo"`). Werden `String`s hingegen über `new` erzeugt, werden sie nicht gecached und es wird tatsächlich ein neues Objekt erzeugt.

Primitive Datentypen und Identität

Weshalb funktioniert dann der Vergleich mit „==“ bei Datentypen wie `int` und `double`?

Das liegt daran, dass sie **primitive Datentypen** sind. Primitive Datentypen sind keine Objekte und besitzen daher auch keine Identität. Ihre Identität bzw. Gleichheit wird daher ausschließlich über ihren Wert bestimmt. Strings sind im Gegensatz dazu Objekte und besitzen daher eine Identität.

Objektdiagramme als Snapshots

Ein Objektdiagramm stellt daher einen Zustand des Systems zu einem bestimmten Zeitpunkt dar. Die Attributwerte und Beziehungen ändern sich während der Ausführung des Programms. Ein Objektdiagramm ist sozusagen ein **Snapshot** des Systems zu einer bestimmten Zeit.

Habe ich es verstanden?

- Ich kann UML Objektdiagramme lesen, sinnvoll erstellen und den Unterschied zu Klassendiagrammen erklären
- Ich kann Objekte inkl. Zustand und Beziehungen in Objektdiagrammen grafisch darstellen
- Ich kann die unterschiedlichen Bezeichnungen von Objekten erklären
- Ich kann den Unterschied zwischen Gleichheit und Identität von Objekten erklären und anhand eines Beispiels darstellen
- Ich kenne die Feinheiten beim Vergleich von Strings und kenne die Funktionsweise des String-Pools