Fragen zu „Objektorientierte Programmierung mit MATLAB“
(c) Ulrich Stein

3. Objektorientierung

Welche Blöcke gibt es in der classdef-Beschreibung?
In MATLAB kann die classdef-Datei folgende Blöcke enthalten:
properties, *methods*, *events* und *enumeration*.

Welcher Block enthält die Eigenschaften eines Objekts?
Der Block *properties* enthält die Eigenschaften eines Objekts.

Was versteht man unter Datenkapselung?
Datenkapselung beschränkt den Zugriff auf die Eigenschaften eines Objekts.

Was sind Methoden einer Klasse?
Methoden enthalten die Beschreibung der Operationen, die mit den Objekten einer Klasse durchgeführt werden können. Formal sind Methoden in MATLAB aufgebaut wie Funktionen.

Mit welcher Operation kann man auf Eigenschaften und Methoden zugreifen?
Und zum Aufruf von Eigenschaften oder Methoden einer Klasse benötigt man ein Objekt dieser Klasse. Der Zugriff erfolgt dann normalerweise über den Punkt-Operator.

Was ist die Aufgabe des Konstruktors?
Konstruktoren dienen dazu, Objekte vom Typ der zugehörenden Klasse zu erzeugen. Bei dieser Operation werden meist auch schon die Daten der Klasse initialisiert.

Was versteht man unter der Überladung einer Funktion?
Das Überladen einer Funktion ist Teil der Polymorphie. In der Klassendefinition kann man Methoden anlegen, die denselben Namen tragen wie Methoden anderer Klassen, oder dieselben Operatoren, wie „+“, verwenden. Polymorphie gehört zu den Stärken der OOP. So können die unterschiedlichen Objekte je nach Datentyp auf ähnliche Nachrichten individuell reagieren.

Was ist der Sinn einer Vererbung?
Durch die Vererbung kann man gemeinsame Eigenschaften und Methoden verwandter Klassen in eine Basis-Klasse auslagern. Von der Basisklasse werden die anderen Klassen abgeleitet. Dabei erben diese abgeleiteten Klassen die Eigenschaften und Methoden der Basisklasse. Die Unterschiede legt man erst in der Definition der abgeleiteten Klasse fest.

Wie unterscheidet sich die Überschreibung einer Methode von einer Überladung?
In abgeleiteten Klassen kann man Methoden der Basis-Klasse **überschreiben**. Damit kann es zu einem Objekt mehrere Methoden mit derselben Signatur (Name, Anzahl und Typ der Eingangsparameter) geben.
Beim **Überladen** von Methoden verwendet man dagegen nur den Namen von Methoden nicht verwandter Klassen. Hier unterscheiden sich die Signaturen der Methoden.

Was ist eine abstrakte Klasse?

Abstrakte Klassen dienen dazu, gemeinsame Eigenschaften abgeleiteter Klassen zu verwalten. Hierbei werden die Methoden der Klassen bereits durch die Angabe ihrer Signatur deklariert. Die Implementierung der einzelnen Methoden erfolgt jedoch erst in den abgeleiteten Klassen.

Wie überlädt man Operatoren?

Zu jedem zugelassenen mathematischen Operator hat MATLAB einen Funktionsnamen definiert, der bei der zugehörenden Operation aufgerufen wird. Zum Überladen dieser Operatoren erzeugt man in der Klasse eine Methode mit diesem Namen und der zugehörigen Signatur.

Was ist der Unterschied zwischen Value- und Handle-Klassen?

MATLAB kennt zwei Arten von Klassen: Value-Klassen und Handle-Klassen. Value-Klassen enthalten hauptsächlich Datenwerte (values) und sind für abstrakte Objekte gedacht, beispielsweise für mathematische Berechnungen. Handle-Klassen verweisen dagegen auf reale Objekte, die eindeutig identifiziert werden können. Das unterschiedliche Verhalten der beiden Klassen wird besonders bei Kopier- oder Übergabeoperationen sichtbar.

Wie definiert man Ereignisse?

Mögliche Ereignisse werden in einer Handle-Klasse im Abschnitt *events* spezifiziert. Die Operation *notify* verschickt eine Meldung, wenn das Ereignis eingetreten ist. Listener-Objekte bekommen diese Meldung mitgeteilt und können darauf mittels einer Callback-Funktion reagieren.

Was ist ein Destruktor?

Destruktoren sind Methoden von Handle-Klassen, die aufgerufen werden, wenn ein Objekt dieser Klasse gelöscht wird. In MATLAB tragen die Destruktor-Methoden den Namen *delete*.

Wie erzeugt man konstante Eigenschaften?

Konstante Eigenschaften werden in der classdef-Spezifikation durch das Attribut *Constant* festgelegt.

Was ist das Besondere an statischen Methoden?

Statische Methoden können auch ohne ein explizites Objekt der Klasse aufgerufen werden.

Wie definiert man Aufzählungen?

Aufzählungen werden in einer Handle-Klasse im Abschnitt *enumeration* spezifiziert.

Wozu dienen Pakete und Namensbereiche?

Pakete definieren einen Namensbereich (*namespace*) für ihre Mitglieder. Wenn die Mitglieder eines Teams jeweils unterschiedliche Pakete entwickeln, können dadurch Namenskollisionen verhindert werden. Die Eindeutigkeit der Namen ist nur innerhalb eines Pakets notwendig.

Wie erzeugt man ein leeres Objekt?

Ein leeres Objekte erzeugt man mit der Methode *empty*.

Wie läuft eine Exception ab?

Exceptions sind Ausnahme-Regelungen, die automatisch dafür sorgen, dass die aktuelle Funktion und alle darüber liegenden beendet werden. Im Fehlerfall wirft man eine Exception (*throw*), die durch Programm-Code fliegt. Eine andere Funktion fängt die Exception auf (*catch*) und reagiert auf die Fehlermeldung – oder auch nicht.

Was bewirkt die Funktion error?

Die Funktion *error* erzeugt eine einfach aufgebaute Exception, der man einen Mitteilungstext und optional eine Fehler-ID mitgeben kann.

Wie ist ein Identifier zu einer Exception aufgebaut?

Die optionale Variable *msgId* ist ein frei wählbarer Message-Identifier. Er besteht aus zwei Wörtern, die durch einen Doppelpunkt getrennt sind, also beispielsweise „Fahrer:argCheck“.

Wie fängt man eine Exception?

Zum Fangen einer Exception muss der problematische Befehl in einen try-Block gepackt werden. Tritt kein Fehler auf, macht das Programm hinter dem *end* weiter. Wird vom Fahrer-Aufruf jedoch eine Exception geworfen, landet man hinter dem *catch* und kann dort auf den Fehler reagieren.

Wie wirft man eine Exception?

Selbst definierte Exceptions werden durch den Aufruf der Funktion *try* geworfen.

Welche Information liefert ein MException-Objekt?

Ein MException-Objekt liefert Daten zu *identifier*, *message*, *cause* und *stack*.

Was bewirkt die Funktion assert?

Die Funktion *assert* erzeugt eine einfach aufgebaute Exception, die automatisch dann aufgerufen wird, wenn eine Vorbedingung (*expression*) verletzt ist.