Konstruktoren in TD 6.3

Einleitung

Team Developer 6.3 bietet nunmehr auch die Möglichkeit, beim Design und der Verwendung von Klassen, Konstruktoren und Destruktoren zu hinterlegen. In diesem Papier soll dargelegt werden, welche Möglichkeiten diese neue Funktionalität dem Programmierer schafft.

Konstruktoren bieten die Möglichkeit, Eigenschaften eines Objekts bei seiner Instanziierung automatisch zu setzen. Mithilfe von Konstruktoren kann der Programmierer einer Klasse, spezifische Variablen des Objekts, abweichend vom Standard, automatisch vorbelegen. Damit wird eine mögliche Fehlerquelle bei der Programmierung eliminiert, da der Programmierer bisher beim Eintreten eines Ereignisses (z.B. SAM_Create) diese Vorbelegung vornehmen musste.

Anhand einer kleinen Beispielanwendung soll gezeigt werden, welche Optionen dem Programmierer mit Team Developer 6.3 im Bereich der Klassenprogrammierung durch Verwendung von Konstruktoren zusätzlich zur Verfügung stehen.

Beispiel

In diesem Beispiel wurden die Funktionsklassen **Fahrzeug**, **PKW** und **Motorrad** definiert, wobei die Klassen **PKW** und **Motorrad** von der Klasse **Fahrzeug** abgeleitet wurden. In der Klasse **Fahrzeug** sind die Variablen **Name**, **Hersteller** und **Anzahl Räder** definiert.

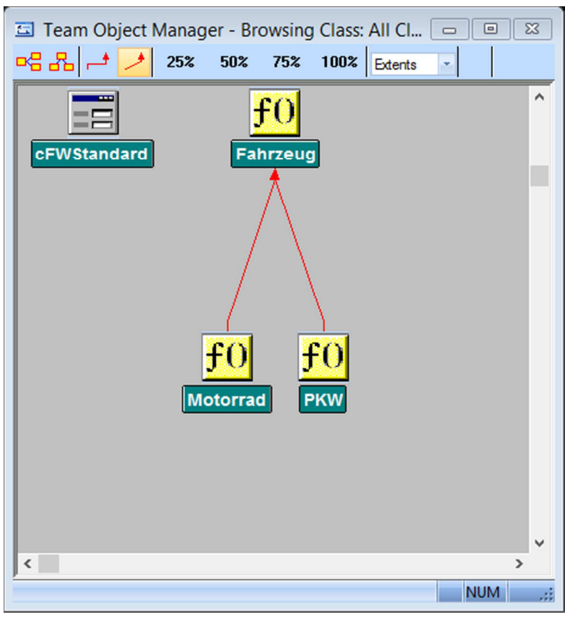

Abbildung 1: Visualisierung der Klassenhierarchie des Beispiels

Wenn nun eine Instanz der Klasse **Motorrad** angelegt wird, soll automatisch die Variable **Anzahl Räder** mit der Zahl **2** vorbelegt werden, während die Initialisierung der gleichen Variable bei einer Instanz der Klasse **PKW** mit **4** erfolgen soll. Mit diesen Klassen soll es möglich sein, PKWs und Motorräder, aber auch einen „Sonderfall“ anzulegen und zu visualisieren.

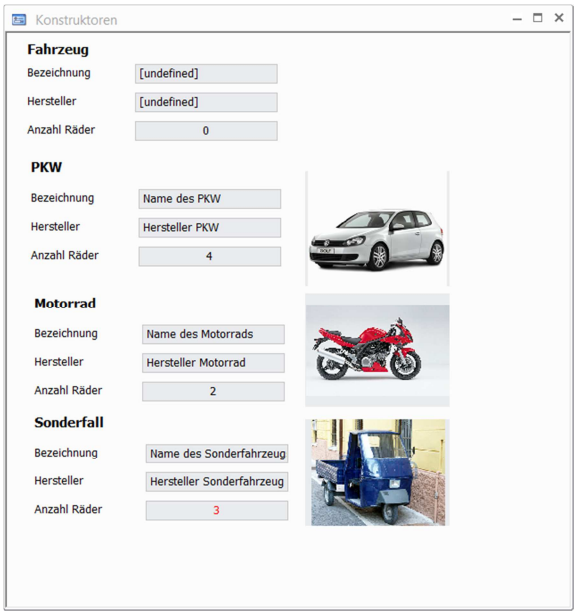

Abbildung 2: Oberfläche der Beispielanwendung

In diesem Beispiel wird eine Oberfläche gezeigt, in der eine Instanz der Klasse **Fahrzeug**, eine Instanz der Klasse **PKW** und eine Instanz der Klasse **Motorrad** visualisiert wird. Als Letztes werden die Variablen eines „Sonderfalls“ angezeigt: Es handelt sich um eine Instanz der Klasse **PKW** mit einer alternativen Initialisierung.

Der Konstruktor bei der Klasse Fahrzeug

```
Constructor: default
    Description:
    Parameters
    Local variables
    Actions
        Set Name = "[undefined]"
        Set Hersteller = "[undefined]"
```

Source Code 1: Konstruktor bei der Klasse Fahrzeug

Bei der Klasse **Fahrzeug** wird ein Konstruktor definiert, über den die Variablen **Name** und **Hersteller** auf einen vordefinierten Wert gesetzt werden. Der Name des Konstruktors ist unerheblich.

Wird eine Instanz der Klasse angelegt, dann bewirkt der Konstruktor, dass die Variablen **Name** und **Hersteller** mit dem Text **[undefined]** vorbelegt sind. Um es anders zu formulieren: Es braucht kein Ereignis in der Oberfläche der Anwendung einzutreten, damit die genannten Variablen entsprechend initialisiert werden – diese Vorbelegung erfolgt durch den Konstruktor.

Konstruktor bei der Klasse Motorrad

Die Klasse **Motorrad** ist von der Klasse **Fahrzeug** abgeleitet. Bei der Klasse **Motorrad** ist ein eigener Konstruktor definiert, bei dem eine für die Klasse **Motorrad** spezifische Initialisierung vorgenommen wird.

```
Constructor: default
    Description:
    Parameters
    Local variables
    Actions
        Set AnzahlRäder = 2
        Set Name = "Name des Motorrads"
        Set Hersteller = "Hersteller Motorrad"
```

Source Code 2: Konstruktor bei der Klasse Motorrad

> *Da Team Developer 6.3 die Vererbung auch bei Konstruktoren unterstützt, wird de facto zuerst die Initialisierung über den Konstruktor der Klasse **Fahrzeug** vorgenommen und danach die Initialisierung über den Konstruktor der Klasse **Motorrad**.*

In diesem Beispiel werden die Textvariablen der Klasse für das Motorrad entsprechend vorbelegt und auch die Anzahl Räder dieses Fahrzeugs auf den Wert **2** gesetzt.

Konstruktoren bei der Klasse PKW

Bei der Klasse **PKW** wurden *zwei* Konstruktoren definiert: **standard** und **variante**. Während der erste Konstruktor keine Parameter hat, arbeitet der zweite Konstruktor mit zwei Parametern.

```
Constructor: default
    Description:
    Parameters
    Local variables
    Actions
        Set AnzahlRäder = 4
        Set Name = "Name des PKW"
        Set Hersteller = "Hersteller PKW"
Constructor: variante
    Description:
    Parameters
        Number: p_nAnzahl
        String: p_sSonderfahrzeug
    Local variables
    Actions
        Set AnzahlRäder = p_nAnzahl
        Set Name = "Name des Sonderfahrzeugs"
        Set Hersteller = p_sSonderfahrzeug
```

Source Code 3: Konstruktoren der Klasse PKW

Mit dem zweiten Konstruktor **variante** soll ein Sonderfall unterstützt werden, bei dem ein PKW nicht **2** sondern eine andere Anzahl von Rädern hat.

Impliziter und expliziter Aufruf von Konstruktoren

In Team Developer wurde in der Vergangenheit eine Instanz einer Klasse schon allein durch ihre Deklaration angelegt. Im Form Window **frm1** werden in diesem Beispiel vier Instanzvariablen angelegt.

```
Fahrzeug: oFahrzeug
PKW: oPKW
PKW: oSonderfall
Motorrad: oMotorrad
```

Source Code 4: Instanzen der Klassen als Window Variablen (von frm1)

Bei der Instanzvariablen **oFahrzeug**, **oPKW** und **oMotorrad** ist ein Konstruktor ohne Parameter hinterlegt. Dieser Konstruktor wird automatisch ausgeführt, wenn die Instanz angelegt wird. Alternativ könnte auch der folgende Code (beispielsweise beim Ereignis **SAM_Create**) ausgeführt werden.

```
Set oFahrzeug = new Fahrzeug
Set oMotorrad = new Motorrad
Set oPKW = new PKW
```

Source Code 5: explizite Instanziierung der Klassen

Bei der Instanz **oSonderfall** (der Klasse **PKW**) muss der Konstruktor mit den Parametern explizit aufgerufen werden.

```
Set oSonderfall = new PKW (3, "Hersteller Sonderfahrzeug")
```

Source Code 6: Instanziierung der Klasse PKW für einen „Sonderfall“

In diesem Fall wird die Initialisierung der Instanz nicht über den Standardkonstruktor (d.h. über einen Konstruktor ohne Parameter), sondern über einen speziellen Konstruktor vorgenommen: die Parameter des speziellen Konstruktors werden in Klammern durch Kommata getrennt eingegeben, um den spezifischen Konstruktor aufzurufen.

Destruktoren

Neben der Definition eines oder mehrere Konstruktoren innerhalb eines Klasse, besteht auch die Möglichkeit einen Destruktor bei einer Klasse ohne Parameter zu hinterlegen. Der Destruktor wird dann aufgerufen und ausgeführt, wenn vom Betriebssystem die notwendige *garbage collection* ausgeführt wird. Dieser Zeitpunkt ist in der Regel für den Anwendungsprogrammierer nicht voraussagbar. Es sollten daher in einem Destruktor keine Funktionen aufgerufen werden, die sich auf „andere Komponenten“ (Datenbankzugriff, Automatisierung, etc.) beziehen, da nicht vorauszusehen ist, ob diese Komponenten zum Zeitpunkt der *garbage collection* überhaupt noch vorhanden sind.

Zusammenfassung

Konstruktoren, insbesondere wegen ihrer „Vielgestaltigkeit“, bieten einen eleganten Weg, um gewünschte und notwendige Initialisierungen von Klasseninstanzen vorzunehmen. Durch Hinterlegung von Konstruktoren bei Klassen, wird die Verwendung von Instanzen dieser Klassen robuster und weniger fehleranfällig.

Definiert werden können:

- Standardkonstruktoren ohne Parameter
- Spezifische Konstruktoren mit unterschiedlichen Parametern

Standardkonstruktoren werden ausgeführt, wenn eine Instanz der Klasse deklariert wird oder explizit eine neue Instanz mit dem Operator **new** angelegt wird. Bei der Initialisierung über spezifische

Konstruktoren (mit Parametern) müssen diese Instanzen programmtechnisch mit **new** angelegt und die Parameter durch Kommata getrennt in Klammern eingegeben werden.

MD Consulting & Informationsdienste GmbH

www.md-consulting.de

Michaelisstraße 13 a
99084 Erfurt
03 61 / 5 65 93-0

Berghamer Straße 14
85435 Erding
0 81 22 / 97 40-0

info@md-consulting.de