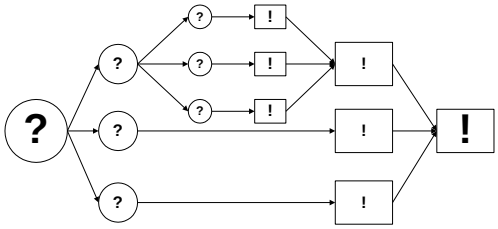
gds2 Gottlieb-Daimler-Schule 2
Technisches Schulzentrum Sindelfingen
mit Abteilung Akademie für Datenverarbeitung

Inhaltsverzeichnis

1 Allgemeines

Unter Modularisierung versteht man die **Aufteilung eines Programms in mehrere Einzelteile** (=Module). Es sei erinnert an das Prinzip der Strukturierten Programmierung, bei der ein Problem in Teilprobleme und die Beziehung zwischen diesen Teilproblemen (Schnittstellen) zerlegt wird.



Jedes Modul
- löst ein Teilproblem des Gesamtproblems
- wird durch ein eigenes Struktogramm beschrieben
- wird einmal erstellt und kann beliebig oft benutzt werden: Wiederverwendbarkeit
- wird in Java durch eine "Klassenmethode" umgesetzt.

Die Regeln zur Bildung eines Moduls wurden im Informationsblatt "Kontrollstrukturen im Struktogramm" beim Thema "Aufruf" dargestellt. Die dort aufgerufenen Struktogramme werden in Form von sogenannten Klassenmethoden (statischen Methoden) in Java programmiert.

2 Realisierung in Java

2.1 Klassenmethoden (statische Methoden)

2.1.1 Beschreibung

Klassenmethoden werden in anderen Programmiersprachen auch als "Unterprogramme" oder "Funktionen" bezeichnet. Diese Begriffe gibt es in Java nicht.

Jede Klassenmethode
- hat einen *Namen*, über den sie von verschiedenen Stellen des Programms aus aufgerufen werden kann
- kann einen oder mehrere Parameter haben, die das Verhalten der Methode zur Laufzeit beeinflussen können.
- kann einen *Rückgabewert* haben, der an den Aufrufer zurückgegeben wird
- Die Kombination aus Name, Parametern und Rückgabewert bezeichnet man als Schnittstelle oder *Signatur* der Klassenmethode.
- wird *in einer Klasse* programmiert.
  Das kann diejenige sein, in der auch die Methode main programmiert ist oder auch eine andere (siehe unten).

2.1.2 Erstellung

Für die Erstellung von Klassenmethoden gilt folgendes allgemeine Schema:

```
1   public class ...
    {
      ...

2     <Zugriffsattribut> static <Ergebnistyp> methodenname (Parameterliste)
3     {
4       // Deklaration von lokalen Variablen, z.B. für den Rückgabewert
5       <Ergebnistyp> Rückgabewert;
6       // Code der Methode <Methodenname>

7       return Rückgabewert;
8     }
      ...
    }
```

1	Methoden werden immer *innerhalb* einer Klasse definiert.
2	<Zugriffsattribut> meistens public **static** *muss* bei Klassenmethoden zusätzlich angegeben werden (deswegen auch statische Methoden genannt) <Ergebnistyp> der Datentyp des Wertes, der als Rückgabewert verwendet wird methodenname () wird üblicherweise klein geschrieben. Beginnt häufig mit einem Verb, das zum Ausdruck bringt, welche Operation in der Methode ausgeführt wird. Dem Methodennamen folgen immer runde Klammern, (). <Parameterliste> Liste der Parameter, durch Komma getrennt, kann auch leer sein. Jeder Parameter wird wie eine lokale Variable betrachtet, besteht also aus Datentyp und Variablenname. Die Wertzuweisung an diese Variable erfolgt beim Aufruf der Klassenmethode.
3,8	Die Variablendeklarationen und Anweisungen der Methode sind in geschweifte Klammern eingeschlossen.
4,5	In jeder Methode können eigene Variablen deklariert werden. Diese werden als "lokale Variablen" bezeichnet und sind nur innerhalb dieser Methode bekannt.
6	Die Anweisungen bilden den Code der Methode. Anweisungen dürfen nur innerhalb von Methoden stehen.
7	Mit dem Aufruf des Schlüsselworts return wird die Methode verlassen und die Programmausführung hinter dem Methodenaufruf fortgesetzt. Nach return folgt ein Ausdruck, dessen Ergebnistyp dem Ergebnistyp der Methode entsprechen muss (siehe 2). *Eine Methode sollte nur eine return Anweisung enthalten.* Fehlt return, so wird die Methode nach der letzten Anweisung verlassen und die Programmausführung ebenfalls hinter dem Methodenaufruf fortgesetzt. Der Ergebnistyp ist dann void

2.1.3 Aufruf von Klassenmethoden

Klassenmethoden werden immer mit ihrem Namen und – sofern vorhanden - den Übergabeparametern aufgerufen. Dem Methodennamen wird der Name der Klasse, in der die Klassenmethode programmiert ist, vorangestellt und mit einem Punkt (.) abgetrennt:

```
Klassenname.methodenname (Parameter);
```

Der "Aufrufer" einer Klassenmethode ist im Allgemeinen eine andere Methode, z.B. die Methode main[1]. Befindet sich die Aufrufermethode in derselben Klasse wie die aufgerufene Klassenmethode, so kann der vorangestellte Klassenname auch weggelassen werden.
Nach Rückkehr aus der Klassenmethode wird der Programmlauf an der Stelle des Aufrufs fortgesetzt.

Für den Aufruf von Klassenmethoden gilt folgendes allgemeine Schema:

```
1   public class Aufruferklasse
    {
2     public static void main (String [] arg)
      {
        /*
         * Anweisungen in main, z. B. Deklaration von
         * rueckgabewert und Parameter
         * ...
         * Aufruf einer Klassenmethode mit Verarbeitung des Rückgabewerts:
         */

3       rueckgabewert = Klassenname.methodenname (Parameter);
        /*
         * hier Fortsetzung des Programmlaufs nach Rückkehr aus <methodenname>
         */
        System.out.println("Ergebnis :" +rueckgabewert);
4        /*
         * Aufruf einer Klassenmethode ohne Verarbeitung des Rückgabewerts:
         */
        Klassenname.methodenname2 (Parameter);
5     }
    }
```

1	Klasse, in der die aufrufende Methode programmiert ist. Es kann sich dabei um eine andere Klasse handeln als die, in der die aufgerufene Klassenmethode programmiert ist.
2	Methode, von der aus die Klassenmethode aufgerufen wird, hier main.
3	Aufruf der Klassenmethode <methodenname> aus der Klasse <Klassenname>. Der Rückgabewert, den <methodenname> über return liefert, wird der Variablen rueckgabewert zugewiesen.
4	Hier wird das Programm nach Rückkehr aus <methodenname> fortgesetzt. rueckgabewert kann jetzt weiterverabeitet werden, z.B. durch Bildschirmausgabe
5	Liefert <methodenname> keinen Rückgabewert (Ergebnistyp void), dann fehlt die Zuweisung von rueckgabewert.

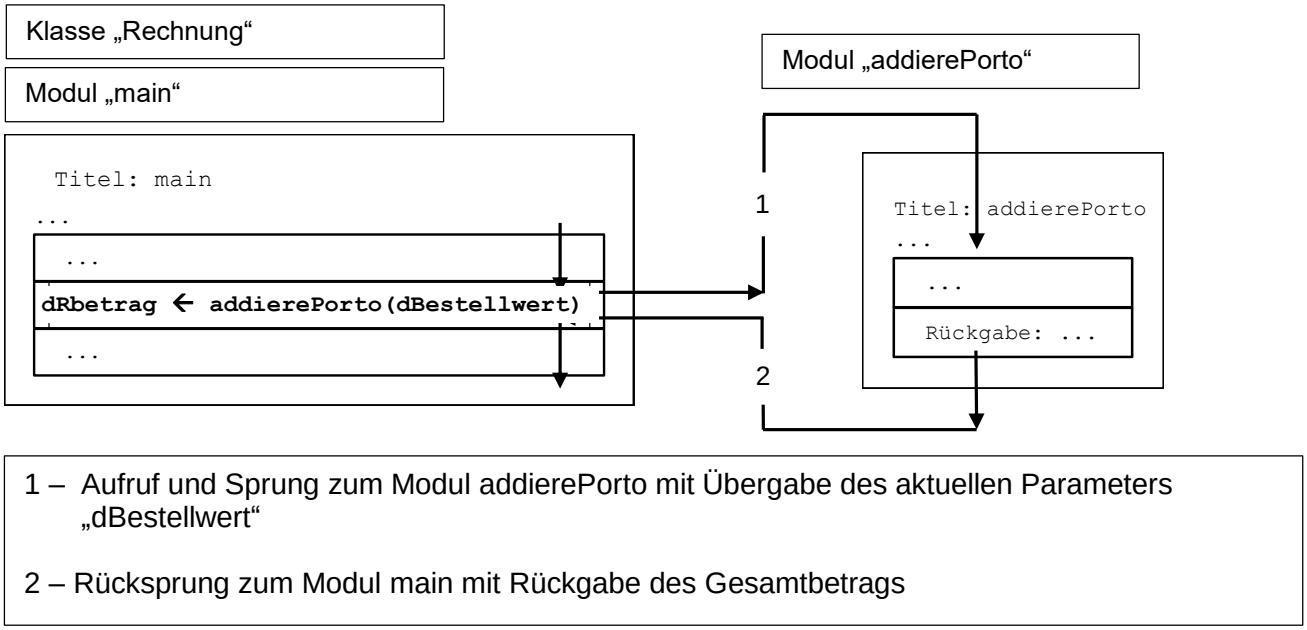
[1] Ruft eine Methode sich selbst auf, so spricht man von einem rekursiven Aufruf, siehe dazu: rekursive Algorithmen in der Literatur.

2.1.4 Beispiel

Das Beispiel Portoberechnung entspricht dem Beispiel aus dem Informationsblatt "03_Struktogramme", Kapitel "Aufruf von Modulen". Der dort beschriebene Zusammenhang wird in Java in Form einer Klassenmethode programmiert. Die zugrundeliegenden Struktogramme können dort nachgelesen werden.

Der Benutzer ruft das Programm "Rechnung" auf. Hier wird der Bestellwert erfasst und anschließend an die Klassenmethode "addierePorto" übergeben. Nach deren Durchlauf wird der errechnete "Rechnungsbetrag" zurückgeliefert und der Variablen "dRBetrag" im Programm "Rechnung" zugewiesen.

Schematischer Ablauf:



1 – Aufruf und Sprung zum Modul addierePorto mit Übergabe des aktuellen Parameters „dBestellwert“

2 – Rücksprung zum Modul main mit Rückgabe des Gesamtbetrags

```
1   public class Rechnung
    {
      public static void main (String [] arg)
      {
2       double dBestellwert;
3       double dRbetrag;

        dBestellwert = Eingabe.getDouble("Bitte den Bestellwert eingeben:");
4       dRbetrag = Rechnung.addierePorto(dBestellwert);

        System.out.println ("Der Rechnungsbetrag ist "+ dRbetrag);
5     } // Ende der Methode main

      /*****************************************************************
      * hier beginnt der Code der Klassenmethode.
      * Achtung: ausserhalb von main, aber innerhalb der Klasse Rechnung
      */
6     public static double addierePorto(double dBestellwert)
      {
        double dBetrag = 0.0;
        if (dBestellwert < 100)
        {
          dBetrag = dBestellwert + 5;
        }
        else
        {
          dBetrag = dBestellwert + 2;
        }
7       return dBetrag;
8     }
    }
```

1	Klasse, in der die aufrufende Methode programmiert ist. In diesem Beispiel sind main und addierePorto in derselben Klasse programmiert.
2	Deklaration der lokalen Variablen dBestellwert.
3	Deklaration einer lokalen Variablen, der der Rückgabewert von addierePorto zugewiesen wird.
4	Aufruf der Klassenmethode addierePorto aus der Klasse Rechnung. Übergabe von dBestellwert als Parameter, Zuweisung des Rückgabewerts an dRbetrag.
5	Ende der Methode main
6	Programmcode der Klassenmethode addierePorto
7	Rückgabe des Wertes von dBetrag; mit der Return Anweisung geht der Kontrollfluss zurück an die aufrufende Methode; eine Methode sollte im Allgemeinen nur eine Return Anweisung enthalten.
8	Ende der Definition der Klassenmethode addierePorto

2.2 Zusammenfassung von Klassenmethoden in eigenen Klassen

In der Regel stellen Klassenmethoden allgemeingültige Funktionalitäten bereit, die jederzeit von anderen Methoden benützt werden können.

Mehrere Klassenmethoden, die ein gemeinsames Thema behandeln, können in einer Klasse zusammengefasst werden. Die Klasse dient in diesem Fall als Container für eine Sammlung von Klassenmethoden.

Beispiele:

- Die (im jdk vorhandene) Klasse java.lang.Math enthält zahlreiche Klassenmethoden zu arithmetischen Berechnungen
- Die (selbst erstellte) Klasse Eingabe enthält zahlreiche Klassenmethoden zur Eingabe von einfachen Datentypen von Tastatur

3 Regeln für die Erstellung Klassenmethode

Die Entscheidung, ob die Erstellung einer Klassenmethode sinnvoll ist, hängt von den gleichen Kriterien ab wie die Bildung eines Struktogramms für einen Aufruf:

Kann mindestens eine der folgenden Fragen mit ja beantwortet werden, kann die Bildung einer Klassenmethode sinnvoll sein.

- Gibt es eine sinnvolle (Teil-)aufgabe?
- Kann das Modul die (Teil-)Aufgabe vollständig lösen?
- Ist die Schnittstelle überschaubar?
  - Können oder müssen zur Lösung Parameter übergeben werden?
  - Ist die Anzahl der zu übergebenden Parameter "gering"?
  - Kann das Modul ein Ergebnis liefern?
  - Wird das Modul dadurch flexibler einsetzbar?
- Hat das Modul Wiederverwendungscharakter?
- Wird die Gesamtaufgabe dadurch überschaubarer?