technische universität
dortmund

Bearbeitet von:
Sebastian Hauer
Sergej Karsten

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Iman Kamehkhosh, Marcel Preuß, Henning Timm,
Alexander Lochmann, Bartho Rudak, Thomas Schmitz

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2014/15

Übungsblatt 7

Bearbeitungszeit:
08.12.14 -
12.12.14

Aufgabe 7.1 – Methoden

Bearbeitungszeit: 25 Minuten

Sie kennen aus den bisherigen Übungsblättern bereits die sogenannte `main`-Methode, in welcher der Programmfluss beginnt und die fester Bestandteil jedes ausführbaren Java-Programmes ist. Wir werden uns im folgenden Gedanken darüber machen, wie wir Funktionalitäten eines größeren Programmes in Unterprogramme — sog. statische Methoden — aufteilen können.

Erstellen Sie eine neue Klasse mit dem Namen „MeanCalculator“ mit folgenden öffentlichen und statischen Methoden:

- Einer **main**-Methode
- Einer Methode mit dem Namen **calculateMean**, welche zwei Integer Parameter `sum` und `elements` besitzt und einen Double zurückgibt
- Einer Methode mit dem Namen **readInteger**, welche keinen Parameter besitzt und einen Integer zurückgibt
- Einer Methode **printNice**, welche einen Double als Parameter besitzt und nichts zurückgibt

Die Methode **calculateMean** soll den Mittelwert einer Summe von Zahlen aus der übergebenen Summe und der übergebenen Anzahl an Elementen der Summe berechnen und zurückgeben. Notieren Sie sich kurz die Berechnungsformel:

Die Methode **readInteger** soll einen Scanner auf dem Standard-Eingabe-Stream erstellen. Geben Sie anschließend eine geeignete Eingabeaufforderung aus und lesen Sie einen einzelnen Integer ein und geben Sie diesen zurück.

Die Methode **printNice** soll den übergebenen Double als Teil einer von Ihnen verfassten Nachricht an den Benutzer ausgeben.

Implementieren Sie anschließend in der **main**-Methode eine `for`-Schleife, die wie im vorletzten Aufgabenzettel terminiert, wenn der Benutzer zuletzt eine `0` eingibt. Rufen Sie innerhalb der Schleife die Methode **readInteger** auf, um einen Integer einzulesen und auf eine Summe zu summieren. Rufen Sie nach verlassen der Schleife die Methode **calculateMean** auf, um den Mittelwert aller eingegebenen Zahlen zu berechnen und geben Sie diesen mithilfe der Methode **printNice** aus.

Welche der hier implementierten Methoden berechnen semantisch nicht das, was ihr Name behauptet? Kann man die Funktionalität der Methoden auch anders implementieren? Kann man sie ganz weg lassen?

Aufgaben zur Rekursion

Bearbeiten Sie die folgenden drei Aufgaben über Rekursion, indem Sie eine Klasse „Recursion“ anlegen und die entsprechenden Methoden in ihr implementieren. Schreiben Sie zum testen dieser Methoden in dieser Klasse eine **main**-Methode und rufen Sie die von Ihnen geschriebenen Methoden in dieser mit Beispielwerten auf und geben Sie das Ergebnis auf der Konsole aus.

Aufgabe 7.2 – Rekursion: Fakultät

Bearbeitungszeit: 10 Minuten

Schreiben Sie in Ihrer Klasse „Recursion“ eine Methode, die folgende Rekursionsformel für die Fakultätsfunktion umsetzt:

$$n! := \begin{cases} 1 & \text{für } n = 0 \\ n \cdot (n-1)! & \text{für } n \geq 1 \end{cases}$$

Aufgabe 7.3 – Rekursion: Fibonacci

Bearbeitungszeit: 10 Minuten

Schreiben Sie in Ihrer Klasse „Recursion“ eine Methode, die durch folgende Rekursionsformel die n-te Fibonacci Zahl zurückgibt.

$$\text{fib}(n) := \begin{cases} 1 & \text{für } n = 1 \\ 1 & \text{für } n = 2 \\ \text{fib}(n-1) + \text{fib}(n-2) & \text{sonst} \end{cases}$$

Aufgabe 7.4 – Rekursion: Potenz

Bearbeitungszeit: 10 Minuten

Schreiben Sie in Ihrer Klasse „Recursion“ eine Methode, die durch folgende Rekursionsformel die n-te Potenz der Zahl b berechnet.

$$b^n := \begin{cases} 1 & \text{für } n = 0 \\ b \cdot b^{n-1} & \text{für } n > 0 \end{cases}$$

Aufgabe 7.5 – Nützliche Programme: Selbstgemacht

Bearbeitungszeit: 60 Minuten

Erstellen Sie eine neue Klasse mit dem Namen „Calculator“ und implementieren Sie folgende Methoden:

- Eine, welche zwei Integer-Parameter entgegen nimmt und überprüft, ob der erste Parameter durch den zweiten teilbar ist.
- Implementieren Sie eine weitere Methode, die überprüft, ob eine gegebene Zahl gerade oder ungerade ist.
- Implementieren Sie eine Methode, die das Maximum zweier als Parameter übergebener Zahlen zurück gibt.
- Implementieren Sie eine Methode, die das Minimum zweier als Parameter übergebener Zahlen zurück gibt.
- Implementieren Sie eine Methode, die den größten gemeinsamen Teiler zweier als Parameter übergebener Zahlen berechnet und zurück gibt.
- Implementieren Sie eine Methode, die das kleinste gemeinsame Vielfache zweier als Parameter übergebener Zahlen berechnet und zurück gibt.

Verwenden Sie bei der Implementierung der einzelnen Methoden bei Bedarf Schleifen und bei der Initialisierung der Zählvariablen dieser Schleifen gegebenenfalls die Methoden für die Maximum- und Minimumsuche.

Geben Sie ihren Methoden und deren Parametern entsprechende Namen.

Aufgabe 7.6 – Wiederverwendung

Bearbeitungszeit: 15 Minuten

Einer der Hauptgründe, Programme in Klassen und Pakete aufzuteilen, ist es bereits geschriebene Programme wiederverwenden zu können. Wenn Sie eine neue Klasse in einem Paket anlegen, in dem bereits eine Klasse existiert, können Sie die statischen Methoden, welche Sie in der anderen Klasse geschrieben haben, in der neuen Klasse aufrufen und nutzen. Ansonsten müssen Sie die entsprechende Klasse importieren, indem Sie die Anweisung `import paket.Klasse` verwenden.

Sie können statische Methoden anderer Klassen aufrufen, indem Sie die Programmanweisung **Klasse.methode(Parameter)** verwenden.

Schreiben Sie das Programm des Selbsttests von letzter Woche in Eclipse und verwenden Sie dabei, wenn möglich, die Methoden ihrer Klasse „Calculator“.