Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Angewandte und Numerische Mathematik

Programmieren: Einstieg in die Informatik mit Java

WS 2006/2007

Dr. G. Bohlender
Dipl.–Math. techn. M. Richter

20.11.2006

Aufgabenblatt 4

Bearbeitungszeitraum: 23.11.2006 – 06.12.2006

Aufgabe 10: *Primfaktorzerlegung*

Betrachten Sie noch einmal das in Aufagabe 6 entwickelte Programm `Primzahltest`. Erstellen Sie ein neues Java-Programm mit dem Namen `Primfaktorzerlegung`, das eine ganze Zahl vom Typ `long` in ihre Primfaktoren zerlegt. Gehen Sie wie folgt vor:

(a) Erstellen Sie eine statische Methode mit dem Namen `findePrimfaktor`, die zu einer gegebenen natürlichen Zahl n den kleinsten Primfaktor, der in n enthalten ist, als `long`-Wert zurück gibt. Übergeben Sie die Zahl n als Argument vom Typ `long` an die Methode. Die Deklaration der Methode sollte also wie folgt aussehen:

```
static long findePrimfaktor(long n) {
  /* Anweisungen */
}
```

Verwenden Sie innerhalb der Methode die `do-while`-Schleife, die Sie für das Programm `Primzahltest` entwickelt haben. Wenn diese Schleife abbricht, steht der kleinste Primfaktor von n in der Variable `teiler`. Ist dieser Wert gleich `1`, so ist n selbst eine Primzahl und zerfällt daher nicht weiter in Faktoren.

(b) Lesen Sie in der `main`-Methode eine `long`-Variable `N` von der Konsole ein. Bestimmen Sie mit der Methode `findePrimfaktor` einen Primfaktor der eingelesenen Zahl. Ist das Ergebnis p von `findePrimfaktor` gleich `1`, so geben Sie die Zeile „N `ist eine Primzahl`" auf dem Bildschirm aus. Falls p ungleich `1` ist, so geben Sie in einer `while`-Schleife p aus. Dividieren Sie anschließend den Wert in der Variable `N` durch p, und bestimmen Sie einen neuen Primfaktor p dieser Zahl. Die Schleife soll durchlaufen werden, solange p ungleich `1` ist. Vergessen Sie nicht, als letzten Primfaktor noch den Wert in der Variable `N` auszugeben.

Aufgabe 11 (Pflichtaufgabe): *Numerische Integration*

In vielen naturwissenschaftlichen Modellen spielen Integrale von Funktionen eine wichtige Rolle. In der Kinematik, beispielsweise, ist die momentane Geschwindigkeit $v(t)$ eines Körpers gleich dem Zeitintegral $\int_0^t a(\tau)\, d\tau$ der Beschleunigung a über dem Zeitintervall $[0, t]$.

Aus dem Schulunterricht ist bereits bekannt, dass nicht alle Funktionen f eine Stammfunktion F besitzen, mit der das Integral

$$\int_a^b f(x)\, dx \tag{1}$$

durch $F(b) - F(a)$ berechnet werden könnte. Deshalb kann ein solches Integral nur numerisch bestimmt werden. Dazu werden sogenannte *Quadraturformeln* verwendet. Eine Quadraturformel nähert das Integral durch eine Summe der Form

$$\frac{b-a}{2} \sum_{j=1}^{q} w_j f(\xi_j) \tag{2}$$

an. Dabei bezeichnet man die Stellen $\xi_1, \ldots, \xi_q \in [a, b]$ als *Stützstellen* und die Werte $w_1, \ldots, w_q$ als *Gewichte* der Quadraturformel. Die Zahl q heißt die *Ordnung* der Quadraturformel.
Schreiben Sie ein Java-Programm mit dem Namen `Quadratur`, welches die Integrale der Funktionen

$$f_1(x) = x + 1, \qquad f_2(x) = x^3 + x^2 - 2x, \qquad f_3(x) = x^5 - 5x^4 + x$$

über beliebigen Intervallen $[a, b]$ durch Quadraturformeln annähert. Gehen Sie im Einzelnen wie folgt vor:

(a) Erstellen Sie zunächst eine statische Methode namens `evalFunction`, die für einen Funktionsindex $k \in \{1, 2, 3\}$ und für eine Stelle $x \in \mathbb{R}$ den Funktionswert $f_k(x)$ als `double`-Wert zurück gibt. Der Funktionsindex k soll als `int`-Argument, die Stelle x als `double`-Argument an die Methode übergeben werden, d.h. die Methode soll wie folgt deklariert werden

```
static double evalFunction(int k, double x) {
  /* Anweisungen */
}
```

(b) Schreiben Sie eine statische Methode namens `evalQuadratureRule`, welche zu einem gegebenen Funktionsindex $k \in \{1, 2, 3\}$ und zu einem gegebenen $q \in \{1, 2, 3\}$ das Integral

$$\int_{-1}^{1} f_k(z)\, dz \tag{3}$$

der Funktion f_k über dem sogenannten *Referenzintervall* $[-1, 1]$ durch eine Quadraturformel der Ordnung q annähert. Quadraturformeln für Integrale über dem Referenzintervall sind von der Form

$$\sum_{j=1}^{q} w_j f_k(\zeta_j) \tag{4}$$

mit den *Referenzstützstellen* $\zeta_1, \ldots, \zeta_q$. Übergeben Sie sowohl den Funktionsindex k als auch die Ordnung q als `int`-Argumente an die Methode `evalQuadratureRule`. Das Ergebnis der Quadraturformel (4) soll von der Methode als `double`-Wert zurück gegeben werden. Verwenden Sie die Referenzstützstellen und Gewichte der sogenannten *Gauß-Legendre-Quadratur*:

Ordnung	Referenzstützstellen	Gewichte
$q = 1$	$\zeta_1 = 0$	$w_1 = 2$
$q = 2$	$\zeta_1 = -\sqrt{\frac{1}{3}},\ \zeta_2 = \sqrt{\frac{1}{3}}$	$w_1 = 1,\ w_2 = 1$
$q = 3$	$\zeta_1 = -\sqrt{\frac{3}{5}},\ \zeta_2 = 0,\ \zeta_3 = \sqrt{\frac{3}{5}}$	$w_1 = \frac{5}{9}, w_2 = \frac{8}{9}, w_3 = \frac{5}{9}$

Lesen Sie in der `main`-Methode Werte für k und q von der Konsole ein und geben Sie das Ergebnis von `evalQuadratureRule` auf dem Bildschirm aus.

Hinweis für Interessierte: Die Referenzstützstellen und Gewichte der Gauß-Legendre-Quadratur sind so gewählt, dass die Quadraturformeln für Polynome vom Grad 1 (für $q = 1$), vom Grad 3 (für $q = 2$) bzw. vom Grad 5 (für $q = 3$) den exakten Integralwert berechnen.