Übungen zur Einführung in die Programmiersprache Java

Universität Regensburg
NWF II - Physik
Dominik Köppl

Wintersemester 2011/12
Blatt 6

---

23 Division durch Null

Tritt in Java eine Division durch Null auf, so wird die Exception `ArithmeticException` ausgeworfen.

- Erstellen Sie eine eigene Exception `MyDivisionError`
- Schreiben Sie eine Funktion
  `double divide(int a, int b) throws MyDivisionError;`, welche $\frac{a}{b}$ berechnen soll.
- Wird eine `ArithmeticException` ausgeworfen, so soll diese abgefangen und eine Exception vom Typ `MyDivisionError` ausgeworfen werden.

24 Trigonometrische Funktionen

Man kann Sinus und Cosinus als Taylorreihe bei 0 entwickeln lassen:

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} \ , \ \cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

Taschenrechner und Computer nutzen diese Reihenformel, um Sinus und Cosinus berechnen zu können. Nun können wir nicht unendlich viele Summanden aufsummeren, sodass wir die Reihe irgendwann abbrechen.

- Zunächst setzen wir

$$\sin_n(x) = \sum_{k=0}^{n} (-1)^k \frac{x^{2k+1}}{(2k+1)!} \ \forall n \in \mathbb{N}.$$

- Nun gilt es, die Folge $\{\sin_n(x)\}_{n\in\mathbb{N}}$ zu berechnen.
- Wir wählen ein $\epsilon \in \mathbb{R}$ klein genug, s.d. wir die Berechnung abbrechen, sobald $|\sin_n(x) - \sin_{n-1}(x)| < \epsilon$.
- Schließlich geben wir $\sin_n(x)$ als Abschätzung für den Sinus zurück.
- Das Verfahren funktioniert analog für den Cosinus.

Implementieren Sie die Funktionen `double sinus(double x, double epsilon);` und `double cosinus(double x, double epsilon);`, die nach obigen Abschätzungsverfahren Sinus und Cosinus berechnen.

25 Nullstellensuche

Gegeben sei eine stetige Funktion $f : I \to \mathbb{R}, I := [a, b] \subset \mathbb{R}$ Intervall mit $f(a) < 0 < f(b)$. Nun muss sie als stetige Funktion mindestens eine Nullstelle zwischen a und b haben. Wir können eine Nullstelle approximieren, indem wir schrittweise folgenden Algorithmus befolgen:

repeat
  $m \leftarrow \frac{a+b}{2}$
  if $f(m) < 0$ **then**
    $a \leftarrow m$
  else
    $b \leftarrow m$
  end if
until $|f(m)| < \epsilon$

Dieses Verfahren heißt Intervallhalbierungsverfahren, da es stets das zu durchsuchende Intervall in zwei Hälften teilt.

(a) Schreiben Sie nun ein Interface `HatFunktion`, das

- die Methode `double f(double);`
- die Getter `double getA()` sowie `double getB()` (für die Intervallgrenze $[a, b]$)

beinhaltet.

(b) Schreiben Sie eine Funktion `double nullstelle(HatFunktion fun, double epsilon);`, welches mit dem Intervallhalbierungsverfahren die Methode `fun.f()` nach eventuellen Nullstellen durchsucht.

(c) Sie haben doch bestimmt eine Lieblingsfunktion, von der Sie die Nullstellen bereits kennen? Testen Sie das Intervallhalbierungsverfahren mit dieser Funktion, indem Sie in der Main-Routine eine Instanz einer anonymen Klasse der `nullstelle`-Funktion übergeben.