Praktikum Programmiertechnik (Technische Informatik)
SS 2016, Hochschule für Angewandte Wissenschaften (HAW), Hamburg
Prof. Dr. Philipp Jenke, Prof. Dr. Axel Schmolitzky, Norbert Kasperczyk-Borgmann

Aufgabenblatt 7: Nullstellen

In diesem Aufgabenblatt entwickeln Sie Funktionalität zum Finden von Nullstellen mit dem Newton-Verfahren. Wir betrachten eindimensionale Funktionen

y = f(x) mit x, y aus den reellen Zahlen (`double`)

Funktionen werden durch ein Interface `Funktion` repräsentiert, das zwei Methoden hat:
f(x) und f'(x), also Berechnung des Funktionswertes für eine Stelle x und Berechnung der Ableitung an einer Stelle x. Nullstellen der Funktion werden mit dem Newton-Verfahren berechnet. Für einen Startwert x_0 ergibt sich die Schätzung der Nullstelle über die iterative Vorschrift

$$x_{n+1} = x_n - f(x_n) / f'(x_n).$$

Aufgabe 7.1: Newton-Verfahren

Schwerpunkte: Umsetzung eines numerischen Verfahrens
Aufgabe: Schreiben das Interface `Funktion`, mindestens eine Klasse quadratische Funktion ($f(x) = ax^2 + bx + c$), die Funktion implementiert, und eine Klasse `Nullstellen`, die das Newton-Verfahren implementiert. Darin gib es die Methode

```
double findeNullstelle(double startWert),
```

die für einen gegebenen Startwert das Newton-Verfahren anwendet. Die Klasse hat außerdem einen Schwellwert **ε**. Damit wird geprüft, ob das Verfahren bereits eine Nullstelle gefunden hat (nämlich wenn f(x) < **ε**). Testen Sie die Funktionalität in einer eigenen JUnit-Testklasse.
Beispiel: $f(x) = x^2-1$, $x_0 = 2$, $e = 10^{-5}$, x = 1

Aufgabe 7.2: Ausnahmebehandlung

Schwerpunkte: Exceptions implementieren und verwenden
Aufgabe: Bei der Anwendung des Newton-Verfahrens können (unter anderem) zwei Probleme auftreten: Division und 0 und keine Konvergenz. Diese beiden Probleme sollen durch eine eigene Exceptionklasse `NullstellenException` gemeldet werden. Instanzen der Klasse können je genau einen der beiden Fehler repräsentieren (verwenden Sie einen Aufzählungstyp). Erweitern Sie die Methode `findeNullstelle()` aus der vorherigen Aufgabe. Sie soll nun eine Exception werfen, wenn durch 0 geteilt wird oder wenn eine maximale Anzahl von Iterationen berechnet wurde, ohne dass die Berechnung konvergierte. Erweitern Sie die Testklasse, sodass Sie sicherstellen, dass das Exception-System in beiden Fehlerfällen korrekt arbeitet.
Beispiele:

- Division durch 0 für f(x) = 1, $x_0 = 2$, $e = 10^{-5}$, `maxAnzahlIterationen = 10`
- Division durch 0 für $f(x) = x^2+1$, $x_0 = 2$, $e = 10^{-5}$, `maxAnzahlIterationen = 10`

Aufgabe 7.3: Mehrere Nullstellen

Schwerpunkte: Collections-Framework, `equals()`, `hashCode()`
Aufgabe: Nun soll die Klasse `Nullstellen` auch mehrere Nullstellen selbständig finden, ohne dass man einen Startwert vorgibt. Schreiben Sie also eine Methode

```
List<Double> findeNullstellenRandomisiert(int min, int max, int anzahlVersuche)
```

Diese sucht nach Nullstellen im Interval [min,max] mit einem randomisierten Verfahren. Zunächst wird im Intervall eine Zufallszahl bestimmt. Diese wird als Startwert für das Standard-Verfahren verwendet. Wenn das Verfahren mit dem Startwert konvergiert, wird sich das Ergebnis in einer Menge gemerkt. Dieses Vorgehen wird `anzahlVersuche` oft wiederholt. Am Ende wird die Menge in eine Liste konvertiert, aufsteigend sortiert und zurückgegegeben.
Leider funktioniert das Verfahren so nicht, es werden `anzahlVersuche` Lösungen gefunden, wobei die meisten Lösungen sehr ähnlich sind. Grund ist, dass beim Vergleich von Doubles (Einfügen in die Menge) nicht mit einer **ε** -Genauigkeit gearbeitet wird. Um dies zu beheben, müssen Sie eine eigene Klasse `DoubleWert` schreiben, die einen `double`-Wert repräsentiert. Diese muss die Methoden `equals()` und `hashCode()` implementieren. `equals()` soll dann wahr liefern, wenn die verglichenen double-Werte numerisch gleich sind (Unterschied < e). Verwenden Sie dann in der Methode `findeNullstellenRandomisiert()` intern den Datentyp `DoubleWert`.
Beispiel: Beispiel: $f(x) = x^2-1$, $x_0 = 2$, $e = 10^{-5}$, anzahlVersuche = 100, [min,max] = [-5,5] ⟶ Nullstelle: {-1,1}