University of Zurich[UZH]

s.e.a.l. software evolution & architecture lab

Programmierung für Mathematik HS11

Übung 7

1 Aufgabe: Integer Arrays

1.1 Lernziele

1. Arrays repetieren.
2. Verzweigungen und Schleifen repetieren.

1.2 Aufgabenstellung

Versuchen Sie folgende Methoden zu implementieren, wobei die Instanzvariable jeweils nicht verändert werden soll. (Verwenden Sie keine Methoden der Klassen aus der Java-API.)

a)

Schreiben Sie eine Klasse `IntegerArray`, die als Instanzvariable ein Array vom Basistyp `int` besitzt, welches direkt bei der Initialisierung gesetzt werden kann.

b)

Schreiben Sie eine Methode, welche das Array zurückgibt. Es soll aber nicht direkt eine Referenz auf die Instanzvariable zurück gegeben werden, sondern nur eine Referenz auf eine Kopie der Instanzvariable.

c)

Schreiben Sie eine Methode, welche das sortierte Array zurückgibt.

d)

Schreiben Sie eine Methode

```
public int[] replace(int oldNumber, int newNumber)
```

Diese Methode soll `oldNumber` mit `newNumber` ersetzen, sofern `newNumber` im Array vorkommt.

e)

Schreiben Sie eine Methode, welche die Absolutbeträge aller Zahlen im Array berechnet und wieder als Array zurückgibt.

f)

Schreiben Sie eine Methode, welche das Array in umgekehrter Reihenfolge zurückgibt.

g)

Schreiben Sie eine Methode, welche die i-te Position des Arrays löscht (i soll vom Benutzer eingegeben werden können), d.h. es muss ein Array zurückgegeben werden, welches um Eins kürzer ist. Falls der Benutzer eine ungültige Position eingibt, soll das ganze Array zurückgegeben werden.

2 Aufgabe: Rekursion

2.1 Lernziele

1. Mathematische Rekursion verstehen.
2. Rekursionen in Java implementieren können.

2.2 Aufgabenstellung

a) Fibonacci

Schreiben Sie eine Methode, welche das n-te Glied der Fibonacci Folge berechnet.
Hinweis: die Fibonacci Folge ist durch folgende Rekursion gegeben:

$$a_0 := 1, \quad a_1 := 1, \quad a_n := a_{n-1} + a_{n-2} \quad \forall n \in \mathbb{N}_{\geq 2}$$

b) Intervallschachtelung

Betrachten Sie folgende Intervallschachtelung und überlegen Sie sich, was damit berechnet wird (dabei soll $x \in \mathbb{R}_{>1}$ beliebig sein).

$$[a_n, b_n] \subset \mathbb{R} \text{ sei Intervall, wobei:}$$

$$[a_0, b_0] := [1, x] \quad [a_{n+1}, b_{n+1}] := \begin{cases} [\frac{a_n+b_n}{2}, b_n] & \text{falls } \left(\frac{a_n+b_n}{2}\right)^2 \leq x \\ [a_n, \frac{a_n+b_n}{2}] & \text{falls } \left(\frac{a_n+b_n}{2}\right)^2 > x \end{cases}$$

Implementieren Sie diese Rekursion, wobei x vom Benutzer eingegeben werden soll. Überlegen Sie sich eine geeignete Abbruchbedingung. Der Rückgabewert soll der Mittelpunkt des zuletzt berechneten Intervalls sein.

3 Aufgabe: Das Newton Verfahren

3.1 Lernziele

1. Rekursive Methoden schreiben können.
2. Bereits implementierte Klassen sinnvoll erweitern können.

3.2 Aufgabenstellung

Wir wollen mithilfe von Rekursiven Methoden und unserer Klasse `Polynomial` aus Übung 4 das Newton Verfahren zur Bestimmung von Nullstellen implementieren.

Das Neton Verfahren ermöglicht es, mittels des Funktionswertes einer Funktion und deren Ableitung sich sukzessive einer Nullstelle zu nähern, sofern man sich bereits in der Nähe einer Nullstelle befindet. Wir nutzen dazu folgende rekursive Formel:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

a) Die Methode `calculateRoot`

Fügen Sie der Klasse `Polynomial` eine weitere Methode `calculateRoot` hinzu. Dieser Methode soll eine Variable vom Typ `double` übergeben werden, welche den Startwert x_0 der Rekursion festlegt. Die Methode soll nun mittels Newton Verfahren solange rekursiv weiterlaufen, bis sie eine Stelle x_n gefunden hat, deren Funktionswert kleiner als eine Konstante `EPSILON` ist, welche zuvor auf den Wert 10^{-6} festgelegt wurde. Die so gefundene Nullstelle soll als `return`-Wert zurück gegeben werden. Um eine Division durch 0 zu vermeiden, soll die Methode eine Fehlermeldung auf der Konsole ausgeben, falls die Ableitung kleiner als `EPSILON` ist.

b) Verbesserte Methode `calculateRoot`

Wir wollen nun eine Verbesserungen an `calculateRoot` vornehmen.

Wir möchten, dass die Methode nach 100 durchläufen abbricht, falls noch keine Nullstelle gefunden wurde, da wir ansonsten durch einen ungünstig gewählten Startwert in eine endlose Rekursion gerate könnten. Definieren Sie dazu eine neue Konstante `MAX_ITERATIONS`, welche auf 100 gesetzt wird. Passen sie nun die Klasse `Polynomial` so an, dass eine Fehlermeldung ausgegeben wird, falls nach 100 durchläufen noch keine Nullstelle gefunden wurde.

4 Aufgabe: Interface

4.1 Lernziele

1. Sie können eine gemeinsame Schnittstelle schreiben und andere Klassen diese implementieren lassen.
2. Sie verstehen anhand einem ersten Beispiel die Vorteile einer gemeinsamen Schnittstelle.

4.2 Aufgabenstellung

Modellieren Sie die folgenden geomentrischen Figuren jeweils in einer eigenen Klasse. Die Implementierung soll dabei nur so detailliert sein, dass für jede Figur das Volumen abgefragt werden kann:

- Quader
- Gerader Kreiszylinder
- Kugel

Die notwendigen Grössen sollen direkt beim Erzeugen des jeweiligen Objektes gesetzt werden. Falls Ihnen die einzelnen Voluminaberechnungen nicht geläufig sind, hilft diese Auflistung weiter.

Erweitern Sie schliesslich die Klassen dahingehend, dass es möglich ist, unterschiedliche Instanzen in einem geeigneten Array abzulegen und in einer Schleife alle Volumina auszugeben. Schreiben Sie einen `TestDriver` welche die Funktionalität Ihres Programmes testet, wobei die verschiedenen Instanzen in einem `Array` abgespeichert werden sollen.