

## Präsenzaufgaben 8

25.11.2021

Die Lösung der Aufgaben wird am Ende der Übung von Ihnen vorgestellt.

### Ableitung von Funktionen

Schreiben Sie ein Interface `Funktion` mit den Methoden

```
public double getY(double d);  
public Funktion getAbleitung();
```

Die erste Methode gibt den Funktionswert  $f(x)$  zurück. Die zweite Methode gibt ein neues Funktionsobjekt zurück, das der Ableitung  $f'(x)$  entspricht.

Schreiben Sie zunächst drei Klassen, die das Interface implementieren:

- Die Klasse `Null` stellt die konstante Funktion  $f(x)=0$  dar.
- Die Klasse `AXhochN` stellt die Funktion  $f(x)=a \cdot x^n$  dar.
- Die Klasse `Summe` hat zwei Funktionsobjekte als Attribute und stellt die Summe beider Funktionen dar.

All diese Klassen erhalten zusätzlich eine `toString`-Methode. Eine Testklasse (auf der Veranstaltungsseite) berechnet von der Funktion  $f(x)=3x^2+7x$  die ersten drei Ableitungen und sollte folgendes Ergebnis ausgeben:

```
0. Ableitung: 3.0x^2.0+7.0x^1.0  
f(2) = 26.0  
1. Ableitung: 6.0x^1.0+7.0x^0.0  
f(2) = 19.0  
2. Ableitung: 6.0x^0.0+0.0x^-1.0  
f(2) = 6.0  
3. Ableitung: 0.0x^-1.0+-0.0x^-2.0  
f(2) = 0.0
```

### Optimierung der Darstellung

Die Ausgabe aus dem vorigen Teil ist zwar korrekt, aber schöner wäre es, wenn z.B. bei der 3. Ableitung einfach  $f'''(x)=0$  ausgegeben würde. Optimieren Sie dazu Ihr Programm:

Kopieren Sie die Klassen und bringen Sie nun die folgenden Änderungen an:

- Optimieren Sie die `toString`-Methode aus `AXhochN`, indem Sie die Ausgabe in den Sonderfällen  $a=1$ ,  $n=0$  und  $n=1$  optimieren.
- Für den Sonderfall  $a=0$  soll sich das Objekt „automatisch“ in ein `Null`-Objekt verwandeln. Wir benutzen dazu eine Variante des Entwurfsmusters **Factory** (Fabrik).
- Verhindern Sie, dass Objekte der Klasse `AXhochN` mit `new` erzeugt werden. Setzen Sie dazu den Konstruktor auf `private`.
- Um neue `AXhochN`-Objekte zu erzeugen, fügen Sie der Klasse eine Funktion (statisch!)

```
public static Funktion get(double a, double n)  
hinzu. Diese Funktion gibt ein Null-Objekt zurück, falls  $a=0$  ist, und ein neues AXhochN-Objekt, falls  $a \neq 0$  ist. Da die get-Funktion Teil der AXhochN-Klasse ist, können Sie den privaten Konstruktor von AXhochN benutzen.
```

- Ersetzen Sie jetzt alle Aufrufe des Konstruktors durch den entsprechenden `get`-Aufruf.
- Verfahren Sie genauso bei der Klasse `Summe`. Wenn einer der beiden Summanden ein `Null`-Objekt ist, wird der andere Summand zurückgegeben, ansonsten ein `Summe`-Objekt.

Als weitere Optimierung können Sie für die Klasse `Null` ein weiteres Entwurfsmuster einfügen: **Singleton**. Da alle `Null`-Objekte gleich sind, genügt es, ein einziges `Null`-Objekt zu erzeugen und immer nur dieses eine `Null`-Objekt zu verwenden:

- Setzen Sie auch hier den Konstruktor auf `private`.
- Führen Sie eine private Konstante ein:  

```
private static final Null N = new Null();
```
- Schreiben Sie auch hier eine `get`-Funktion. Diese gibt jetzt einfach die Konstante zurück.

Die Ausgabe sollte nun wie folgt aussehen:

0. Ableitung:  $3.0x^2.0+7.0x$

$f(2) = 26.0$

1. Ableitung:  $6.0x+7.0$

$f(2) = 19.0$

2. Ableitung:  $6.0$

$f(2) = 6.0$

3. Ableitung:  $0$

$f(2) = 0.0$