

<b>NACHNAME:</b>	<b>SEMESTER:</b> <input type="checkbox"/> M5 <input type="checkbox"/> M6 <input type="checkbox"/> M3 <input type="checkbox"/> M7
<b>VORNAME:</b>	<b>VERTIEFUNG:</b> <input type="checkbox"/> FV <input type="checkbox"/> IM

**Aufgabe 1: (14 Punkte)**

**WICHTIG:** Achten Sie in **allen** Aufgabenteilen darauf, dass keine illegalen Objekte entstehen. javadoc-Kommentare sind hingegen **nicht** verlangt.

Die Klasse `Monom` beschreibt eine Funktion der Form

$$f(x) = a \cdot x^n \quad (a \in \mathbb{R} \setminus \{0\}, n \in \mathbb{N}_0)$$

Eine Implementierung beginne so:

```
public final class Monom {
    private final double mKoeffizient;
    private final int mHochzahl;
}
```

a) Ein elementarer Test prüft einen der Konstruktoren von `Monom`:

```
@Test
public void testeParabel() {
    final int grad = 2;
    Monom p = new Monom(1., grad);
    assertEquals(grad, p.grad());
}
```

Geben Sie eine mögliche Implementierung des im Test verwendeten Konstruktors an. Andere im Test benötigte Methoden werden als bereits vorhanden angenommen.

b) Ein anderer Test prüft die Methode `ableitung`:

```
@Test
public void testeAbleitungsGrad() {
    final int startGrad = 86;
    Monom m = new Monom(1., startGrad);
    assertEquals(startGrad, m.grad());
    for (int grad = startGrad; grad > 0; grad--) {
        m = m.ableitung();
        assertEquals(grad - 1, m.grad());
    }
}
```

Wie könnte die Implementierung der Methode `ableitung` aussehen?


c) Sinnvollerweise sollte nicht nur der Grad der Ableitung getestet werden, sondern die ganze Funktion, etwa so:

```
@Test
public void testeAbleitungsfunktion() {
    final int grad = 5;
    Monom p = new Monom(1., grad);
    Monom ps = new Monom(grad, grad - 1);
    assertEquals(ps, p.ableitung());
}
```

Was muss in der Klasse `Monom` getan sein, damit dieser Test funktioniert?


**Aufgabe 2: (10 Punkte)**

Die Aufgabe bezieht sich auf die in **Aufgabe 1** definierte Klasse `Monom`:

a) In der Klasse `Monom` werde versucht, die Deklaration von `mHochzahl` um eine explizite Initialisierung zu ergänzen:

```
public final class Monom {  
    private final double mKoeffizient;  
    private final int mHochzahl = 0;  
}
```

Was passiert dann im Konstruktor `Monom(double, int)` und weshalb?


b) In einem JUnit-Test werde versucht, das `Monom m` mittels

```
System.out.println("" + m.mKoeffizient  
    + "x^" + m.mHochzahl);
```

auszudrucken. Was passiert und weshalb?


c) In einer Methode der Klasse `Monom` werde versucht, auf die Daten eines anderen Objekts der Klasse zuzugreifen:

```
public Monom mal(Monom anderes) {  
    double a = mKoeffizient * anderes.mKoeffizient;  
    int n = mHochzahl * anderes.mHochzahl;  
    return new Monom(a, n);  
}
```

Was passiert und weshalb?


**Aufgabe 3: (16 Punkte)**

Eine Klasse StringTango enthalte folgende Methode:

```
public static List<String> tuWasMit(List<String> texte) {
    List<String> erg = new ArrayList<String>();
    for (String text : texte) {
        if (!text.equals("Cha")) erg.add(text);
    }
    if (texte.size() - erg.size() > 2) {
        texte = erg;
    } else if (erg.size() < 2) {
        erg = texte;
    }
    texte.add("tanzen");
    return erg;
}
```

An anderer Stelle werde diese Methode aufgerufen:

```
List<String> parameter = ...
List<String> ergebnis = tuWasMit(parameter);
```

Prüfen Sie welches Ergebnis (ergebnis) bei dem jeweils angegebenen Argument (parameter) erzielt wird und wie sich gegebenenfalls das Argument dabei verändert.

**WICHTIG:** Zur syntaktischen Vereinfachung werden als Strings nur einzelne Wörter verwendet und die Listen als Sätze dargestellt. In der ersten auszufüllenden Zeile besteht die Liste also aus den beiden Strings „Niemals“ und „Tango“.

parameter vor Aufruf	ergebnis	parameter nach Aufruf
Niemals Tango		
Niemals Cha Cha		
Ich mag Cha Cha		
Ich mag Cha Cha Cha		