

Übungsblatt 3: Rekursion auf Datenstrukturen

Hinweis: Diese Aufgaben schließen die Lücke zwischen „einfacher Mathe-Rekursion“ und „komplexer Grafik“. Hier müssen Arrays und Strings verarbeitet werden.

Übungsaufgabe 1: Rekursion mit Strings (Palindrom)

Thema: String-Verarbeitung, Helper-Methode

Ein Palindrom ist ein Wort, das vorwärts und rückwärts gleich gelesen wird (z.B. „OTTO“, „RENTNER“, „A“).

Schreiben Sie eine rekursive Methode:

```
public boolean istPalindrom(String s)
```

- **Die Strategie:** Vergleichen Sie das erste und das letzte Zeichen des Strings.
- Wenn sie gleich sind, prüfen Sie rekursiv den Rest des Wortes (ohne das erste und letzte Zeichen).
- Wenn sie ungleich sind, ist es kein Palindrom (**false**).
- **Abbruch:** Wenn der String Länge 0 oder 1 hat, ist er ein Palindrom (**true**).

Tip: Nutzen Sie `s.charAt(0)`, `s.length()` und `s.substring(...)`.

Übungsaufgabe 2: Rekursion auf Arrays

Thema: Index-Verwaltung (Wrapper-Methode)

Schreiben Sie eine Methode, die die **Summe** aller Zahlen in einem Integer-Array berechnet, **ohne eine Schleife** zu verwenden.

Vorgegeben ist die Signatur:

```
public int summe(int[] zahlen)
```

Aufgabenstellung:

1. Da die Methode nur das Array als Parameter hat, fehlt uns der Index-Zähler.
 2. Schreiben Sie eine private Hilfsmethode `summeRekursiv(int[] arr, int index)`.
 3. **Logik:** Die Summe ab Index `i` ist der Wert an `arr[i]` plus die Summe vom Rest (Index `i+1`).
 4. **Abbruch:** Wenn der Index gleich der Array-Länge ist, geben Sie 0 zurück.
-

Übungsaufgabe 3: Die „Böse“ Fibonacci-Zahl

Thema: Verständnis von Rekursionsbäumen

Die Fibonacci-Folge ist definiert als:

$$fib(n) = fib(n - 1) + fib(n - 2)$$

mit $fib(0) = 0$ und $fib(1) = 1$.

Implementieren Sie die Methode `int fib(int n)`.

Verständnisfrage (Theorie): Warum ist diese naive rekursive Implementierung für große Zahlen (z.B. $n = 50$) extrem langsam, während eine Schleife (iterativ) blitzschnell wäre?

(Hinweis: Zeichnen Sie den Baum für $fib(5)$. Wie oft wird $fib(1)$ berechnet?)

Übungsaufgabe 4: Grafische Rekursion (Concentric Circles)

Thema: Graphics, Parameter-Variation

Erstellen Sie ein Panel, das in `paintComponent` eine Art „Zielscheibe“ rekursiv zeichnet.

Methode: `void drawCircles(Graphics g, int x, int y, int radius)`

1. Zeichnen Sie einen Kreis (`drawOval`) am Mittelpunkt (x, y) mit dem angegebenen Radius.
*Achtung: drawOval erwartet die linke obere Ecke, nicht die Mitte! Sie müssen umrechnen: $x - radius, y - radius, radius*2, \dots$*
2. **Abbruch:** Wenn `radius < 2`, hören Sie auf.
3. **Rekursion:** Rufen Sie die Methode erneut auf, aber verringern Sie den Radius um 10 Pixel.
4. **Zusatz (für Profis):** Ändern Sie bei jedem Aufruf die Farbe zufällig, um einen psychedelischen Effekt zu erzielen.

Tipps für die Prüfung

- **Helper-Pattern (Wrapper):**
Wenn die Aufgabe eine Methode `func(Daten)` vorgibt, du aber einen Zähler brauchst →
Schreibe `private func(Daten, Zähler)` und rufe sie auf.
- **Strings zerlegen (für Aufgabe 1):**
`String rest = s.substring(1, s.length() - 1);`
Dies schneidet den ersten und letzten Buchstaben ab.
- **Array-Rekursion (für Aufgabe 2):**
Denke immer: „Mein Ergebnis = Mein aktueller Wert + Das Ergebnis vom Rest“.
`return array[i] + helper(array, i+1);`