TU Ilmenau, Fakultät IA
FG Telematik/Rechnernetze
Prof. Dr.-Ing. G. Schäfer
http://www.tu-ilmenau.de/telematik/pua

TECHNISCHE UNIVERSITÄT
ILMENAU

Programmierung und Algorithmen WS 23/24
Übungsblatt 3

Die Lösungen der Aufgaben sind bis zum 05.11.23, 23:59 Uhr abzugeben.
Die Besprechung der Aufgaben erfolgt in KW 45.

Aufgabe 1 (Übergabe von Parametern in Java) **2 + 2 + 0 Punkte**

Gegeben sei das unten stehende Java-Programm.

(a) Wie lautet die Ausgabe des Programms wenn es mit dem Kommandozeilenparameter „`Hello?`“ aufgerufen wird? Begründen Sie kurz.

(b) Wie müssen Sie den Methodenaufruf in Zeile 7 ändern, so dass der zweite Teil (d.h. nach dem Komma) der Ausgabe aus Zeile 8 stets der erste Kommandozeilenparameter ist? Beachten Sie, dass die Methode `fun` nach wie vor aufgerufen werden soll (Zeile also nicht einfach löschen)! Änderungen in anderen Zeilen sind nicht erlaubt.
Wichtig: Nach Ihrer Modifikation soll der Inhalt von `args` noch immer die Funktion `fun` erreichen.

(c) **Zusatzaufgabe:** Bei zweidimensionalen Arrays (z.B. `int[][] matrix`) müssen weitere Vorkehrungen getroffen werden. Welche und warum?

```
public class Parameter {
  public static void main(String[] args) {
    if(args.length == 0) {
      return;
    }
    int a = 42;
    fun(a, args); // <-- Zeile 7
    System.out.println(a + ", " + args[0]);
    String[] array = new String[] { "Swapped!" };
    swap(array, args);
    System.out.println(args[0]);
  }

  public static void fun(int a, String[] array) {
    a = array.length;
    array[0] = "Hello!";
  }

  public static void swap(String[] a, String[] b) {
    String[] tmp = a;
    a = b;
    b = tmp;
  }
}
```

Hinweis: Sie können sich zunächst z.B. hier[1] in die Thematik einlesen (ohne den Teil zu „RMI“).

[1] http://www.javadude.com/articles/passbyvalue.htm

Aufgabe 2 (Perfekte Zahlen in Java) **4 Punkte**

Eine Zahl $n \in \mathbb{N}$ heißt perfekt oder vollkommen, wenn die Summe ihrer Teiler gleich n ist. Dabei werden nur die Teiler echt kleiner als n betrachtet. Beispielsweise ist `6 = 1 + 2 + 3` eine perfekte Zahl. Schreiben Sie ein Java-Programm, das alle perfekten Zahlen bis 5000 bestimmt. Geben Sie diese auch in Ihrer Lösung an!

Aufgabe 3 (Perfekte Zahlen applikativ) **5 Punkte**

Formulieren Sie einen *applikativen Algorithmus* `perfect(n)` welcher bestimmt, ob die Eingabe `n` $\in \mathbb{N}$ eine perfekte Zahl ist. Halten Sie sich an die Definition eines applikativen Algorithmus aus der Vorlesung!

Hinweise:

- Sie werden eine Hilfsfunktion benötigen.
- Java-Methoden sind keine applikativen Algorithmen. Halten Sie sich an die Definition eines applikativen Algorithmus aus der Vorlesung und beachten Sie die dortige Syntax! Wenn Sie hier Java-Code abgeben wird Ihre Lösung nicht gewertet.

Aufgabe 4 (Zelluläre Automaten) **6 + 2 Punkte**

Conway's Game of Life spielt auf einem zweidimensionalen Feld, dessen Einträge (Zellen) entweder mit `0` oder `1` belegt sind. Zellen mit `1` *leben*, alle anderen sind *tot*. Zellen die sich senkrecht, waagerecht oder schräg berühren, gelten als benachbart. Durch eine gedachte Anordnung auf einem Torus (der Körper in Form eines "Donut") berühren sich auch Zellen, die an gegenüberliegenden Rändern liegen. Rundenweise wird nun für jede Zelle des Feldes ihr künftiger Zustand entschieden:

- Eine tote Zelle mit *genau* 3 lebenden Nachbarn beginnt zu leben.
- Eine lebende Zelle mit *weniger als* 2 oder *mehr als* 3 lebenden Nachbarn stirbt.
- Alle restlichen Zellen behalten ihren aktuellen Zustand.

(a) Schreiben Sie ein Java-Programm, das ausgehend von einem initialisierten Feld (`int[][]`) beliebiger Größe rundenweise die Regeln des *Game of Life* anwendet.

Hinweise:

- Achten Sie darauf, dass Sie die neuen Zustände der Zellen nicht direkt in das Feld der aktuellen Runde schreiben können, da Sie sonst noch benötigte Informationen überschreiben. Stattdessen müssen die Zustände der kommenden Runde zunächst in einem separaten Feld zwischengespeichert werden.
- Sollten sie die modulo-Operation einsetzen wollen, beachten Sie, dass diese von Java für negative Werte nicht mathematisch korrekt implementiert wird. So gilt zwar $-1 \mod 11 \equiv 10$, allerdings rechnet Java `-1 % 11 = -1`.

(b) Visualisieren Sie Ihr Spielfeld auf der Konsole.

Hinweis: Sie finden in der WebIDE ein Testsspielfeld `conway54.txt` und Code, der dieses Spielfeld einliest. Sollten Sie die Aufgabe außerhalb der WebIDE bearbeiten wollen, beachten Sie, dass zur Nutzung von `FileUtils` die PuA-Bibliothek eingebunden werden muss.

Aufgabe 5 (Fibonacci-Zahlen effizient) **4 Punkte**

In der Vorlesung wurde Ihnen ein applikativer Algorithmus zur Berechnung der Fibonacci-Zahlen vorgestellt (Kapitel 4, Folie 16). Leider ist dieser Algorithmus nicht besonders effizient, da viele Zwischenergebnisse mehrfach berechnet werden. Beispielsweise werden beim Aufruf von `fib(5)` die beiden rekursiven Aufrufe `fib(3)` und `fib(4)` getätigt. In beiden dieser Fälle wird unabhängig voneinander (unter anderem) `fib(2)` als Zwischenergebnis berechnet.
Folgende Java-Methode umgeht dieses Problem, indem die komplette Fibonacci-Folge, beginnend bei f_0, bis zur gewünschten Fibonacci-Zahl in einer Schleife berechnet wird:

```
public static int fib(int n) {
    // die Variablen e und f enthalten immer zwei aufeinanderfolgende
    // Fibonacci-Zahlen, am Anfang die ersten beiden
    int e = 1;
    int f = 1;
    for(int i = 0; i < n; ++i) {
        int tmp = f;
        f = e + f; // neues f ergibt sich aus der nächsten Fibonacci-Zahl
        e = tmp; // neues e ergibt sich aus dem alten f
    }
    return e; // wenn i = n gilt (siehe for-Schleife), ist e das Ergebnis
}
```

Formulieren Sie nun einen *applikativen Algorithmus*, der dieselbe Idee wie die Java-Methode verfolgt. Denken Sie daran, dass Sie Schleifen in einem applikativen Algorithmus nicht direkt verwenden können.

Hinweise:

- Definieren Sie eine Hilfsfunktion `h(n, i, e, f)`.
- Wie in Aufgabe 3 kann auch hier Java-Code nicht gewertet werden.