

Lösungsvorschläge zum 8. Übungsblatt „Programmierung“ im Wintersemester 2016/17

Abgabe bis 12. Dezember 2016, 07:59 Uhr über das Abgabesystem

Aufgabe 1: Fehlersuche in Quelltexten

5 Punkte

Finden Sie alle Fehler in den nachfolgenden Funktionen. Gehen Sie davon aus, dass die Code-Stücke alleine in einer sonst leeren Klasse eingebettet sind. Beschreiben Sie kurz und genau was den Fehler verursacht und wie dieser zu beheben ist.

a) Gibt alle Werte eines Arrays aus.

```
public static void printArray(int[] array) {  
    for (int i = 0 ; i <= array.length; i++) {  
        System.out.print(array[i] + " ");  
    }  
}
```

Die Abbruch-Bedingung geht über den maximalen Array-Index hinaus. `i < array.length` ist hier richtig.

b) Vertauscht die Werte eines Arrays an den gegebenen Stellen x und y.

```
public static void swap(int[] array, int x, int y) {  
    int tmp = array[x];  
    array[x] = array[y];  
    array[y] = array[x];  
}
```

Es wurde falsch vertauscht (letzter Ausdruck), richtig:

```
public static void swap(int[] array, int x, int y) {  
    int tmp = array[x];  
    array[x] = array[y];  
    array[y] = tmp;  
}
```

Hinweise:

- Gesucht sind semantische Fehler (keine syntaktischen)
- Versuchen Sie die Fehler ohne Testen auf einem Computer zu finden

Bepunktung:

- 2 Punkte: 1 Punkt für die Erkennung und genaue Beschreibung der falschen Stelle bzw. des falschen Ausdrucks und 1 Punkt für die Korrektur.
- 3 Punkte: 1,5 Punkt für die Erkennung und genaue Beschreibung der falschen Stelle bzw. des falschen Ausdrucks und 1,5 Punkt für die Korrektur.

Aufgabe 2: Strings parsen

15 Punkte

Implementieren Sie eine Methode `parseInt` (Signatur siehe nachstehend) die, im gültigen Falle, eine Zeichenkette einer maximal neunstelligen Dezimalzahl in einen `int` umwandelt. Falls die Zeichenkette ungültige Zeichen enthält, soll eine Fehlermeldung auf der Konsole ausgegeben werden.

```
public static int parseInt(String str);
```

Überprüfen Sie ihre Implementierung bezüglich korrekter und fehlerhafter Eingaben, indem Sie in Ihrer `main`-Methode entsprechende Tests hinzufügen.

Hinweis: Die Verwendung von `Integer.parseInt()` oder andere Methoden, die die Aufgaben des Parsens übernehmen, führt zu 0 Punkten.

Bepunktung:

- 10 Punkte für Implementierung: 3 Punkte Iteration über String, 2 Punkte negative Zahl erkennen, 2 Punkte für "not a number", 3 Punkte für korrekte Umrechnung (inkl. Behandlung von negativen Zahlen)
- 5 Punkte für Testen: 1 Punkt je abgedecktem Fehlerszenario bzw. richtiger Eingabe.

Aufgabe 3: Cäsar-Verschlüsselung

20 Punkte

Die Cäsar-Verschlüsselung ist ein einfaches symmetrisches Verschlüsselungsverfahren, das auf der monographischen und monoalphabetischen Substitution basiert. Bei der Verschlüsselung wird jeder Buchstabe des Klartexts auf einen Geheimtextbuchstaben abgebildet. Diese Abbildung ergibt sich, indem man die Zeichen eines geordneten Alphabets um eine bestimmte Anzahl zyklisch nach rechts verschiebt (rotiert). Die Anzahl der verschobenen Zeichen bildet den Schlüssel, der für die gesamte Verschlüsselung unverändert bleibt.¹

Schreiben Sie eine Klasse `Caesar` mit einem Konstruktor und zwei Methoden `encrypt` und `decrypt`. Der Konstruktor soll einen Parameter `offset` entgegen nehmen, der die Anzahl der Zeichen, um die verschoben wird, angibt. Der Offset darf dabei nur Werte zwischen -26 und 26 annehmen. Wenn ein anderer Wert übergeben wird, soll der Offset auf 0 gesetzt werden.

Die Methode `encrypt` soll einen String (`klartext`) entgegen nehmen, die Cäsar-Verschlüsselung mit dem im Konstruktor übergebenen Offset anwenden und den verschlüsselten Text zurückgeben.

Die Methode `decrypt` soll einen String (`geheimtext`) entgegen nehmen, den Text entschlüsseln und den ursprünglichen Text zurückgeben.

Beispiel mit Offset 7:

`encrypt`: Für jeden Großbuchstaben muss der siebte Nachfolger im Alphabet bestimmt werden.

`encrypt("TEST") = "ALZA"`

`decrypt`: Für jeden Großbuchstaben muss der siebte Vorgänger im Alphabet bestimmt werden.

`decrypt("ALZA") = "TEST"`

Hinweise:

- In der Vorlage finden Sie einen JUnit-Test für diese Aufgabe.
- Die Verschlüsselung soll nur auf Großbuchstaben angewendet werden. Alle anderen Zeichen bleiben erhalten.
- Den Buchstaben an Stelle x des Strings s , erhalten Sie durch den Aufruf `s.charAt(x)`.
- Einen Buchstaben (vom Typ `char`) können Sie an einen beliebigen String anhängen, indem Sie folgendes aufrufen:
`myString += letter;`
Achten Sie aber darauf, dass der String vorher initialisiert wurde:
`String myString = "";`

Bepunktung:

- Überprüfung des Wertebereichs im Konstruktor (2 Punkte)

¹Quelle: <https://de.wikipedia.org/w/index.php?title=Cäsar-Verschlüsselung&oldid=135936579>

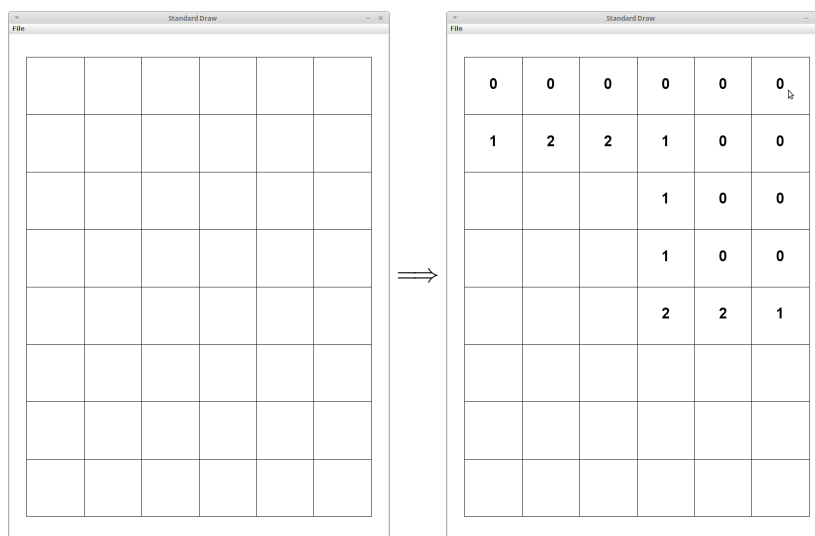
- Überprüfung des Wertebereichs in encrypt (2 Punkte)
- Verschiebung des Zeichens in encrypt (3 Punkte)
- Negatives Ergebnis korrekt behandeln in encrypt (3 Punkte)
- Geheimtext erzeugen in encrypt (1 Punkt)
- Überprüfung des Wertebereichs in decrypt (2 Punkte)
- Verschiebung des Zeichens in decrypt (3 Punkte)
- Negatives Ergebnis korrekt behandeln in decrypt (3 Punkte)
- Geheimtext erzeugen in decrypt (1 Punkt)

Aufgabe 4: Minesweeper, Teil 4

20 Punkte

Erweitern Sie die Lösung von “Minesweeper, Teil 3” zu einem vollständigen Minesweeper-Clone. Dieses Mal sollen alle Nachbarn aufgedeckt werden, wenn sich in der Nachbarschaft eines aufgedeckten Feldes keine Minen befinden. Implementieren Sie dazu die Methode `uncoverNeighbors`, die die Anzahl an aufgedeckten Feldern zurückgibt. Diese Anzahl wird dazu benötigt, um zu erkennen, ob der Spieler alle möglichen Felder aufgedeckt und somit gewonnen hat (siehe dazu Musterlösung “Minesweeper, Teil 3”). Beachten Sie, dass Ihre Implementierung sich selbst wieder aufruft (Rekursion), falls in der Nachbarschaft wieder ein Feld ohne Mine in der Nachbarschaft existiert.

Beispiel: Nachstehend sehen Sie exemplarisch, welche Felder aufgedeckt werden müssen, wenn der Spieler auf Feld (5,7) klickt.



Hinweis:

- Implementieren Sie die Methode `uncoverNeighbors` rekursiv. Eine iterative Lösung mit einer Schleife ergibt 0 Punkte.

Bepunktung:

- 10 Punkte: Korrekte Rückgabe der Anzahl aufgedeckter Felder (Abzüge je nach Fehler)
- 10 Punkte: Alle geforderten Felder werden aufgedeckt und nicht zu viele (Abzüge je nach Fehler)