Heinrich Heine Universität Düsseldorf

Programmierung – WS 2025/26
Dr. Markus Brenneis
Abgabefrist für Feedback:
10.11.25, 10 Uhr

4. Übungsblatt (Lösungsvorschlag[1])

Aufgabe 1: Selbsttests zu Arrays nach VL 6

Wenn Arrays noch neu für Sie sind, benutzen Sie den Selbsttest zu Arrays auf der Kursseite, um zu lernen, wie Probleme mithilfe von Arrays und Schleifen gelöst werden können.

Auch in den Übungsstunden werden Sie den Umgang mit Arrays üben.

Aufgabe 2: Lotto nach VL 6

Beim Lotto 6 aus 49 gibt es 49 Kugeln, die mit den Zahlen 1 bis 49 beschriftet sind. Von diesen Kugeln werden bei einer Lottoziehung 6 verschiedene Kugeln zufällig ausgewählt. Es gibt aber auch andere Lottosysteme, z. B. 6 aus 45.

Schreiben Sie ein Programm `Lotto` mit einer Methode `static int[] ziehung(int n, int m)`, die eine Lottoziehung simuliert. Dabei sollen n **verschiedene**, zufällige natürliche Zahlen größer 0 und kleiner oder gleich m bestimmt werden. Gehen Sie davon aus, dass $n \geq 0$ und $n \leq m$ gilt.

Der Aufruf `ziehung(6, 49)`, um Lotto 6 aus 49 zu simulieren, könnten also z. B. ein Array mit den sechs Zahlen 5, 23, 49, 34, 36, 1 liefern.

Aufgabe 3: Referenzen nach VL 6

In dieser Aufgabe wollen wir uns nochmal ansehen, was es zu beachten gibt, wenn Objekt-Typen und primitive Typen als Parameter einer Methode benutzt werden.

Schauen Sie sich den vorgegebenen Code in der Datei `References.java` an und beantworten Sie folgende Fragen:

- Welche Ausgaben machen die fünf `println`-Aufrufe?
- Begründen Sie jeweils an den drei gekennzeichneten Stellen, wie es zu den jeweiligen Ausgaben durch die vorangegangen `println`-Befehle kommt. Beziehen Sie sich dabei darauf, was Sie über Heap und Stack und die Übergabe von Parametern gelernt haben.

[1] Bei den meisten Programmieraufgaben gibt es mehr als einen funktierenden Lösungsweg. Diskutieren Sie gerne untereinander Ihre Lösungsansätze und lerne Sie damit verschiedene Lösungsstrategien und verschiedene Anwendungsmöglichkeiten der Java-Funktionalitäten kennen. Ihre Abgabe müssen Sie aber final selbst formulieren/eintippen.

Schreiben Sie Ihre Antworten zu den Fragen als Kommentare in den Quellcode und geben Sie Ihren kommentierten Code ab. Als Teil Ihrer Begründung können Sie auch Zeichnungen[2] mit abgeben.

Lösungsvorschlag

- siehe `References.java`
- Bepunktung:
  1. 3 Punkte für die richtigen Ausgaben, für jede falsche/fehlende Angabe 1 Punkt Abzug (keine negativen Teilpunkte)
  2. je einen Punkt pro sinnvoller Begründung (darf knapper ausfallen, muss aber schlüssig sein; insgesamt 3 Punkte möglich)
- Lernziele bei dieser Aufgabe:
  - erklären, welche Daten im Heap/Stack gespeichert werden (Vorlesung *Stack*)
  - erklären, wo Objekte gespeichert sind (Vorlesung *Objekte*)
  - die Semantik von = erklären (Vorlesung *Stack*)
  - Methoden schreiben, die übergebene Objekte (nicht) verändern (Vorlesung *Stack*)

[2] separate Dateien im JPEG-, PNG- oder SVG-Format

Aufgabe 4: Kniffel nach VL 6

Kniffel ist ein Würfelspiel mit fünf Würfeln (D6, also Augenzahlen von 1 bis 6), bei dem in verschiedenen Kategorien abhängig von den gewürfelten Zahlen verschiedene Punktzahlen erreicht werden können. Es gibt insgesamt 13 Kategorien mit verschiedenen Regeln, wie aus den gewürfelten Augenzahlen eine Punktzahl berechnet wird.

Wir wollen nun das vorgegebene Java-Programm `Kniffel` vervollständigen. Es enthält Methoden, die für jede dieser Kategorien für eine gegebene Würfelkonstellation die erreichten Punkte berechnen sollen. Dabei werden die fünf gewürfelten Augenzahlen als aufsteigend sortiertes[3] Integer-Array an die Methoden übergeben. Diese Methoden sollen Sie vervollständigen.

In den Kommentaren oberhalb der Methoden ist jeweils kurz zusammengefasst, nach welchen Regeln die Punktzahl berechnet wird. Wenn Sie mit Kniffel nicht vertraut sind, können Sie die Bepunkuntgsregeln ausführlicher unter https://de.wikipedia.org/wiki/Kniffel#Oberer_Block nachlesen.

Wir haben in der Vorgabe auch eine main-Methode zum Testen vorgegeben.[4]

Beispiel mit den Augenzahlen 11123**:**

```
Einser: 3
Zweier: 2
Dreier: 3
Vierer: 0
Fünfer: 0
Sechser: 0
Dreierpasch: 8
Viererpasch: 0
Full House: 0
Kleine Straße: 0
Große Straße: 0
Kniffel: 0
Chance: 8
```

Beispiel mit den Augenzahlen 23456**:**

```
Einser: 0
Zweier: 2
Dreier: 3
Vierer: 4
Fünfer: 5
Sechser: 6
Dreierpasch: 0
Viererpasch: 0
Full House: 0
Kleine Straße: 30
Große Straße: 40
Kniffel: 0
Chance: 20
```

Tipps:

- Ein guter Anfang sind die Methoden für 1er bis 6er, Kniffel und Chance.
- Überlegen Sie, ob Sie beim Code für die ersten sechs Kategorien (1er bis 6er) geschickt vermeiden können, ähnlich aussehenden Code zu wiederholen.

[3] Sie müssen sich in dieser Aufgabe nicht selbst um das Sortieren kümmern – das passiert automatisch durch unseren vorgegebenen Code. Wie man Zahlen sortiert, schauen wir uns später in der Vorlesung an.

[4] Die Zahlen in der main-Methode dürfen Sie zu Testzwecken verändern, die Signaturen der vorgegeben Methoden dürfen Sie aber *nicht* verändern; weitere Methoden dürfen aber ergänzt werden. Die automatischen Tests prüfen, ob die vorgegebenen Methoden korrekt funktionieren.

Aufgabe 5: Debugging: Längstes Plateau nach VL 6

Wir haben ChatGPT gebeten, die folgende Aufgabe zu lösen. Sie finden den von ChatGPT geschriebenen Code in den vorgegebenen Dateien zu diesem Übungsblatt im Ilias. Leider funktioniert der Code nicht ganz richtig.

Korrigieren Sie den Code. Verändern Sie nur die Teile des Programms, die falsch sind; schreiben Sie den Code nicht komplett neu.

Außerdem gibt es einen Code-Block, der unnötig ist. Welcher?

> Ein Forschungsteam untersucht die Höhenprofile von Bergen. Das Team interessiert sich vor allem für Gebiete mit geringen relativen Höhenunterschieden, sogenannte Plateaus.
>
> Schreiben Sie eine Methode `static void longestPlateau(int[])`. Der Methode werden beliebig viele ganze Zahlen (positiv und negativ) übergeben. Die Methode soll dann sowohl die Position, als auch die Länge des längsten Plateaus innerhalb dieser Werte bestimmen und auf der Standardausgabe ausgeben.
>
> Als Plateau wird eine zusammenhängende Folge gleicher Werte bezeichnet, wobei die Werte direkt vor und direkt nach dieser Folge kleiner sind. Direkt am Rand kann es also kein Plateau geben.
>
> Damit ein Plateau existieren kann, müssen mindestens drei Zahlen übergeben werden. Falls weniger Werte übergeben werden, soll eine aussagekräftige Fehlermeldung ausgeben werden, die `ERROR` enthält.
>
> Ausgegeben werden soll zuerst der Index und dann, durch ein Leerzeichen getrennt, die Länge des ermittelten Plateaus. Sollten mehrere längste Plateaus gleicher Länge existieren, sollen Index und Länge des ersten Plateaus ausgegeben werden. Sollte kein Plateau existieren (z. B. wenn eine Folge aufsteigend sortierter Zahlen übergeben wird), soll `Kein Plateau` ausgegeben werden.
>
> **Beispiel-Aufrufe der Methode:**

```
> longestPlateau(new int[]{3, 5})
ERROR: Bitte mindestens 3 Zahlen übergeben!
> longestPlateau(new int[]{2, 3, 3, 3, 1, 2, 2, 2, 1})
1 3
> longestPlateau(new int[]{1, 2, 3, 4, 3, 2, 1})
3 1
> longestPlateau(new int[]{1, 2, 3})
Kein Plateau
```