Technische Universität Braunschweig
Institut für Anwendungssicherheit

Prof. Dr. Martin Johns
Marius Musch
Maximilian von Unwerth

Programmieren I

04. Übungsblatt

Melden Sie sich bitte für die Veranstaltung bei Stud.IP (http://studip.tu-bs.de) an und tragen sich in einer der kleinen Übungen ein. (`https://studip.tu-braunschweig.de/dispatch.php/course/details?sem_id=d08ea3cd66c59967be1218a540bdefa7`)

Verspätete, nicht hochgeladene Abgaben, Abgaben, die per Mail getätigt werden, sowie Plagiate führen zur Bewertung der Aufgabe mit 0 Punkten.

*Insgesamt können Sie für das Lösen der Pflichtaufgabe **15 Punkte** erhalten. Berücksichtigt werden ausschließlich Ergebnisse, die bis zum **10.01.2020 um 23:59** in Ihr Git Repository auf `https://programming.ias.cs.tu-bs.de/` gepusht wurden.*

Ihre Lösung muss in den bereits für dieses Übungsblatt angelegten Ordner (blatt4) im Git Repository gespeichert werden, damit diese gewertet wird.

Nur compilierende Programme gelten als Lösung! Ein Programm das Teillösungen beinhaltet aber nicht compiliert oder aus anderen Gründen nicht ausführbar ist erhält 0 Punkte.

Aufgabe 1 Gegeben sei eine natürliche Zahl $n \in \mathbb{N}, n \geq 1$. Mit $\sigma(n)$ bezeichnen wir die Summe aller positiven Teiler von n. Beispielsweise ist $\sigma(12) = 1+2+3+4+6+12 = 28$. Schreiben Sie ein Java-Programm, welches die Zahl k, mit $k \geq 100$ und $k \leq 999$ findet, für die $\sigma(k)$ maximal ist.

Aufgabe 2 Gegeben ist folgender Algorithmus:

1. Lies den Wert von n ein.
2. Setze i auf 2.
3. Solange $i < 3(n+1)$ ist:
   a. Erhöhe i um 3.
   b. Gib $(2i+1)^2$ aus.

Dieser soll auf drei verschiedene Arten implementiert werden:

- mit einer `while` Schleife
- mit einer `for` Schleife
- mit einer `do-while` Schleife

Sämtliche Programmstücke sollen die gleiche Ausgabe erzeugen.

Aufgabe 3 Diese Aufgabe nimmt Bezug auf die in der Vorlesung vorgestellte Klasse `Konto` und deren Testklasse `KontoTest`. Der Quelltext steht Ihnen im Stud.IP zur Verfügung.
Aufgabe ist es, Klassen zu erstellen welche doe Funktionalität der Klasse `Konto` erweitern. Nutzen Sie hierfür Vererbung.

a) Erstellen Sie eine Klasse `Sparbuch`, die sicherstellt, dass das Attribut `stand` einer Instanz niemals negativ werden kann.

b) Erstellen Sie eine Klasse `Girokonto`, die einen Überziehungsrahmen (Dispokredit) bietet. Geldbewegungen innerhalb des erweiterten Rahmens sind möglich, Auszahlungen darüber hinaus sollen nicht möglich sein. Die Klasse soll über eine Klassenvariable `sollzins` verfügen, die den Zinssatz für die überzogene Summe enthält.

c) Erweitern Sie die Klasse `KontoTest` so, dass sie die neuen Kontotypen und deren Funktionalität testet.

Pflichtaufgabe 10: Interfaces (5 Punkte) Mit Interfaces können Sie Funktionen bereits vorhandener Interfaces auf Ihr Interface übertragen. Als Beispiel erstellen Sie eine Klasse `IntArrayList`, die das Java-Interface `List` erweitert (1 Punkt). Ihre Klasse soll die Funktionalität einer einfachen Liste, die int-Werte speichert bereitstellen. Ihre Klasse sollte ein int-Array besitzen, dass alle Werte speichert. Beim Erstellen einer Liste soll eine Liste mit einem vorerst leeren Array erstellt werden. Beachten Sie, dass ein int-Array beim Erstellen mit Nullen gefüllt ist. Bei dieser Klasse implementieren Sie die folgenden Methoden: (je 0.5 Punkte)

- `toString(...)`: gibt die Elemente der Liste in einer Zeile aus.
- `size(...)`: gibt die Länge der `IntArrayList` zurück. Diese entspricht der Anzahl der Elemente in dem Array der Liste.
- `isEmpty(...)`: Gibt zurück, ob die Liste mindestens ein Element enthält.
- `contains(...)`: Gibt zurück, ob die übergebene Zahl in der Liste mindestens einmal enthalten ist.
- `add(...)`: Fügt eine Zahl zur Liste hinzu. Beachten Sie, dass Sie dass Array vergrößern müssen, um ein neues Element hinzuzufügen.
- `remove(...)`: Entfernt das übergebene Element aus der Liste. Kommt dies mehrfach vor, wird nur der erste gefundene Eintrag entfernt. Beachten Sie, dass Sie Ihr Array wieder verkleinern.
- `clear(...)`: Entfernt alle Einträge aus dem Array.
- `get(...)`: Gibt den Index des ersten gefundenen Eintrags, der dem Parameter entspricht zurück.

Beachten Sie, dass Sie gegebenenfalls einige weitere Methoden, die das Interface `List` vorgibt, implementieren müssen, da die Implementierung des Interfaces dies erfordert. Hier reicht es, wenn Sie die Methoden mit "Dummy"-Funktionalität ausstatten. Es reicht, wenn Sie die Funktionalität der o.g. Methoden implementieren. Recherchieren Sie die Signaturen der zu implementierenden Methoden.

Pflichtaufgabe 11: Widerstand ist zwecklos (10 Punkte) Die Welt wird von einer der größten Gefahren der Galaxis bedroht: **Den Borg!** Zum Glück hat man sich auf die Ankunft der Borg im Alpha-Quadranten vorbereitet und sich einen Bauplan für die Schiffe der Borg besorgt. Diese werden stets nach dem gleichen Verfahren aufgebaut und offenbaren uns somit ihre einzigen Schwachstellen. Nun ist es Ihre Aufgabe, den Bauplan zu verstehen und ein Programm zu entwickeln, welches den Admirälen der Flotte ihre Angriffsziele klarmacht.

Dazu müssen Sie ein Programm implementieren, dass den Bauplan der Raumschiff umsetzt und die Schiffe auf der Kommandozeile ausgibt. Nutzen Sie ein zweidimensionales

Array, um das Raumschiff zu speichern. Jedes Borg-Schiff besteht zu allererst aus einem stabilen Rahmen, der wie folgt aussieht:

```
1 - - - - - - -
2 |           |
3 |           |
4 |           |
5 |           |
6 |           |
7 - - - - - - -
```

Generieren Sie diesen Rahmen in der Größe, die als Kommandozeilenparameter übergeben wird und speichern Sie ihn im Array ab (1 Punkt). Ein einfaches Borg-Schiff der Größe 5, wie oben abgebildet, hat eine Höhe und eine Breite von 5 Feldern. Erstellen Sie eine Methode printShip(), die das gesamte Schiff (das gesamte Array) auf der Kommandozeile ausgibt. (1 Punkt)

Ein Borg-Schiff verfügt nun neben dem stabilen Raumschiffrahmen über eine Hauptversorgungsleitung, die durch das gesamte Schiff verläuft. Mit dieser sieht das zum Beispiel Schiff wie folgt aus:

```
1 - - - - - - -
2 | -   - - | |
3 | |       | |
4 | | - - - - |
5 |           | |
6 | - - - o | |
7 - - - - - - -
```

Die Versorgungsleitung wird rekursiv für jedes Schiff aufgebaut. Erstellen Sie daher eine Methode, die diese Leitung generiert. Beachten Sie, dass Sie auf die verschiedenen Komponenten dieser Methode nur Punkte erhalten, wenn Ihre Methode einen rekursiven Aufruf enthält! Für einen sinnvollen rekursiven Aufruf erhalten Sie außerdem (1 Punkt).

Die Hauptversorgungsleitung beginnt immer an Koordinate (1,1), also oben links innerhalb des äußeren Rahmens. Daraufhin verläuft Sie **über 2 Felder** zufällig in eine Richtung. Erstellen Sie eine Methode `fieldValid(int x, int y)`, die prüft, ob die übergebene Koordinate ein Feld im Array ist, wo eine Leitung platziert werden kann. Leitungen verlaufen grundsätzlich nur innerhalb des Rahmens und auch nur dort, wo noch keine Leitung platziert wurde (1 Punkt). Wohin eine Leitung verläuft ist vom Zufall abhängig. Erstellen Sie daher einen Zufallsgenerator, der den Leitungen vorgibt, welche Richtungen Sie in welcher Reihenfolge ausprobieren (1 Punkt). Kann eine Leitung an zum Beispiel der Koordinate (3,1) platziert werden, nachdem ausgewürfelt wurde, dass die Leitung zuerst nach unten verlaufen soll, könnte das Schiff wie folgt aussehen (Hier im Beispiel ein Schiff der Größe 7):

```
 - - - - - - -
 | |         |
 | |         |
 | |         |
 |           |
 |           |
 - - - - - - -
```

Achten Sie darauf, dass eine Leitung immer über 2 Felder verlegt wird, nie nur über eines und Sie somit auch stets die Zwischenfelder füllen müssen. Im Beispiel also auch Koordinate (2,1). Verläuft die Leitung nach oben oder unten soll sie als |, verläuft sie nach links oder rechts als **-** visualisiert werden. An Abzweigungen ist es egal, ob sie diese mit horizontalen oder vertikalen Balken visualisieren. (1 Punkt) Ist die erste Leitung gelegt, macht die Methode einen rekursiven Aufruf mit den Koordinaten des aktuellen Leitungsendes (1 Punkt). So kann diese jetzt zum Beispiel nach rechts weiter verlaufen. Dies würde wie im Beispiel folgendes Bild ergeben:

```
 - - - - - - -
 | |         |
 | |         |
 | | - -     |
 |           |
 |           |
 - - - - - - -
```

An einer bestimmten Stelle kann es sein, dass ihr Algorithmus an einer Stelle angelangt, an welcher es nun keine Möglichkeit mehr gibt eine neue Leitung zu platzieren. Dadurch, dass Sie Ihren Algorithmus rekursiv aufgebaut haben, ist es diesem nun möglich an einer Stelle, an der er wie im Beispiel unten zuerst nach oben abgebogen ist nun nochmal nach unten weiterzugehen (1 Punkt). So würde er, nachdem er in der obersten Zeile keinen Platz mehr für eine neue Leitung findet, in der hintersten Zeile noch den Weg nach unten entdecken und nutzen .

```
 - - - - - - -
 | -   - - | |
 | |       | |
 | | - - - - |
 |           |
 |           |
 - - - - - - -
```

Sorgen Sie außerdem dafür, dass Ihr Algorithmus, nachdem er alle Richtungen für die entsprechenden Felder geprüft hat terminiert (1 Punkt).

Die Borg bauen außerdem als jedes siebtes Verbindungselement einen Signalverstärker **o** in ihre Leitungen ein, die die große Schwachstelle ihrer Schiffe sind (1 Punkt).

Hier im Beispiel sind die Schritte einmal durchnummeriert, um das Konzept einfacher sichtbar zu machen.

```
- - - - - - -
| -   - e | |
| a       d |
| | b - c - |
|         f |
| - h - o | |
- - - - - - -
```

Wir beschreiben im folgenden kurz die einzelnen Schritte beim Aufbau der Leitung:

a) Nach unten

b) Nach rechts

c) Weiter nach rechts

d) Nach oben

e) Nach links

f) Es geht nicht weiter nach links, aber auch in keine andere Richtung, gehe daher so viele Schritte zurück, bis es noch irgendwo in eine andere Richtung ging. Gehe hier weiter nach unten

g) Gehe nach links, füge als Zwischenelement keine einfach Leitung, sondern einen Signalverstärker **o** ein.

h) Gehe weiter nach links

i) Es geht in keine andere Richtung mehr weiter. Gehe den Rekursionsbaum weiter nach oben und suche nach noch vorhandenen Möglichkeiten. Bis zum Wurzelelement sind keine weiteren Möglichkeiten mehr vorhanden. Der Aufbau der Leitung ist also beendet.

Das Endergebnis kann sich sehen lassen. Da Sie nun wissen, wie die Borg-Schiffe abhängig von Ihrer Größe aussehen können, sind die Admiräle vorbereitet und können ihre Angriffe auf die Signalverstärker der Raumschiffe konzentrieren. Gute Arbeit!