Technische Universität Braunschweig
Institut für Anwendungssicherheit

Prof. Dr. Martin Johns

Programmieren I

04. Übungsblatt

Aufgabe 20: 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 21: Sie sollen verschiedene Variablen in einem Programm deklarieren. Deklarieren sie eine Variable mit dem passenden Typ die folgenden Wert enthält:

- mit welchem Buchstaben Ihr Nachname beginnt
- wie viele Menschen in Deutschland leben
- wie viele Menschen auf der Erde leben
- ob Sie männlich oder weiblich sind
- wie viele Semester Sie studieren werden
- wie viele Student(innen) an dieser Vorlesung teilnehmen
- ob Sie BaFöG erhalten
- wie groß die Masse der Erde ist
- Ihre y-Nummer

Aufgabe 22: Gegeben ist das folgende Programm:

```
1 public class BrowserTest {
2
3     public static void main(String[] args) {
4         System.out.print("BrowserTest");
5         http://www.google.com;
6         System.out.println("Suchbegriff");
7     }
8
9 }
```

listings/BrowserTest.java

Kompiliert es? Falls ja, was passiert bei seiner Ausführung?

Aufgabe 23: 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 24: 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 25: In dieser Pflichtaufgabe soll das Spiel *Minesweeper* implementiert werden. Minesweeper war eins der standard Spiele, welche mit früheren Versionen von Windows ausgeliefert wurden. Die Windows XP Version von Minsweeper ist in Abbildung 6 zu sehen.

Das Spiel funktioniert folgendermaßen: Auf einem quadratischen Spielfeld mit Länge und Breite n sind x zufällige Felder vermint. Der Spieler kann durch Eingabe von Koordinaten ein Feld auswählen und dort eine von zwei Aktionen ausführen:

Betreten Das gewählte Feld wird aufgedeckt. War dieses vermint, stirbt der Spieler und das Spiel endet in einer Niederlage. Falls nicht, wird in diesem Feld die Anzahl der Minen in den direkt benachbarten Feldern angezeigt.

Entschärfen Wenn das gewählte Feld vermint ist, wird die Mine durch das Entschärfen unschädlich gemacht. Andernfalls hat diese Aktion keinen Effekt.

Ziel des Spiels ist es, alle Minen unbeschadet zu entschärfen. Zur initialen Orientierung sind dafür v nicht verminte Felder aufgedeckt.
Geforderte Features:

- Einlesen von Koordinaten über die Kommandozeile. Diese sollen in einer eigenen Klasse gespeichert werden. Die Gültigkeit der eingelesenen Koordinaten soll hier validiert werden. Zum Einlesen von Koordinaten der Form `H0` können Sie die Scanner Klasse wie in Abbildung 7 verwenden.

- Implementieren einer geeigneten Abstraktion für die verschiedenen Feldtypen (vermintes oder normales Gelände). Ob Sie diese mit einem Interface oder einer abstrakten Klasse und Vererbung umsetzen, steht Ihnen frei. Sie müssen aber innerhalb des Javadoc Kommentars der Basis Klasse/des Interfaces begründen, wieso Sie sich für die gewählte Form der Abstraktion entschieden haben.

- Implementieren der beiden Feldtypen (Landmine und normales Gebiet) basierend auf der im vorherigen Unterpunkt gewählten Abstraktion. Beide Feldtypen *müssen* hier in eigenen Klassen realisiert werden.

- Implementieren einer Klasse für das Spielfeld. Folgende Parameter sollen an den Konstruktor übergeben werden: Die Größe n, die Anzahl der verminten Felder x und die Anzahl der initial aufgedeckten Felder v.
  Basierend auf diesen Parametern soll ein Spielfeld initialisiert werden. Die verminten, sowie die initial aufgedeckten Felder sollen dabei zufällig im Spielfeld verteilt sein.

- Implementieren der Aktionen, also das Aufdecken von nicht verminten Feldern (inklusive Ermittlung der Anzahl der Minen in den benachbarten Feldern), dem Entschärfen von Minen und dem Betreten eines verminten Feldes. Die Auswirkungen dieser Aktionen sollen sich im Zustand der unterliegenden Felder widerspiegeln.

- Implementieren der Anzeige des Spielfelds für den regulären Spielfluss, etwa wie in Abbildung 9 gezeigt. Desweiteren die Ausgabe des Spielfelds mit allen aufgedeckten Feldern, ein Beispiel hierfür ist in Abbildung 8 gegeben.
- Implementieren einer Klasse die den Spielverlauf kontrolliert und den Spieler durch das Spiel führt. Diese soll mit passenden Ausgaben auf die auszuführenden Aktionen hinweisen (wie in Abbildung 9), die Bedingungen für Sieg und Niederlage testen und das Spiel gegebenenfalls beenden. Um das Testen zu vereinfachen, ist es sinnvoll beim Start des Spiels einmal das komplett aufgedeckte Spielfeld anzuzeigen.
- Zeichnen eines UML Klassendiagramms zu dem gewählten Aufbau (Klassen, Interfaces, abtrakte Klassen). Hierbei sollen auch die Verbindungen (Vererbung, Implementierung eines Interfaces, Assoziation zwischen Klassen) der einzelnen Bestandteile deutlich werden.

Für die Parameter (n, x und v) können Sie folgende Werte innerhalb der main Methode als default vorgeben:

- Spielfeldgröße n: 10
- Anzahl Minen x: 23
- Initial aufgedeckte Felder v: 5

Die Parameter via Benutzereingabe oder Kommandozeilenparametern einzulesen ist *nicht* gefordert.

Für die Umsetzung von jedem der obigen Teilaspekte erhalten Sie jeweils einen Punkt. Haben Sie mindestens einen der Teilaspekte (außer der UML Aufgabe) korrekt bearbeitet, können Sie einen weiteren Punkt für Einhaltung der Programmierrichtlinien erhalten. Diese sind:

- Erstellen eines separaten Ordners, in dem der Quellcode abgelegt wird.
- Formatierung des Programms gemäß der Formatierungsrichtlinien.
- Javadoc Kommentare für jede Klasse, jedes Interface und jede public Methode.
- Keine überflüssigen Dateien (hier primär `.class` und `.jar` Datein, aber auch Sonstiges, das nicht unter Quelltext oder Dokumentation fällt) im Git.

Für das Lösen dieser Pflichtaufgabe können Sie maximal 9 Punkte erhalten.
Berücksichtigt werden ausschließlich Ergebnisse, die bis zum 16.01.2020 um 23:59 in das Git Repository Ihrer Gruppe gepusht wurden.
Um Ihre Lösung bewertet zu bekommen, müssen Sie diese in der Woche vom 20.01.2020 - 24.01.2020 dem Tutor/der Tutorin in Ihrer Übungsgruppe vorstellen.
Desweiteren muss die abgebene Lösung compilieren und ein (teil-)funktionstüchtiges

Programm darstellen um Teilpunkte zu bekommen.
Abgaben müssen in dem aufgabenspezifischen Order erfolgen: $pflichtaufgabe04$. Code der nicht in diesem Order (oder Unterordner) liegt wird nicht bewertet.

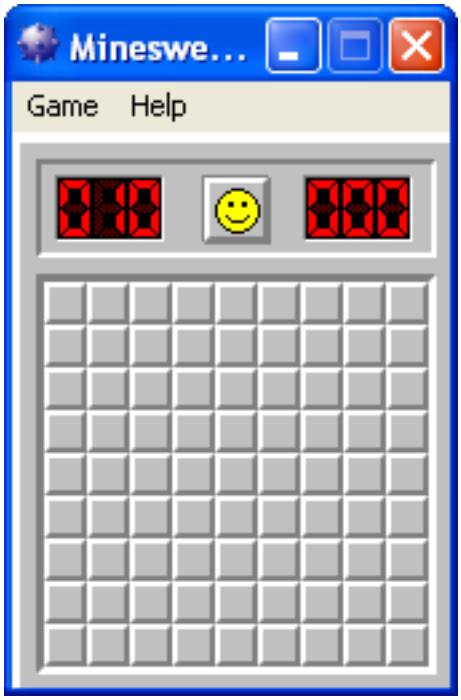

Abbildung 6: Minesweeper von Windows XP (Bild von `https://en.wikipedia.org/wiki/Microsoft_Minesweeper`).

```java
// Setup
Scanner input = new Scanner(System.in);
// Setting the delimiter as "" means each entered letter is returned separately.
input.useDelimiter("");

// Reading a coordinate pair
char column = input.next().charAt(0);
int row = input.nextInt();
// As the return to confirm the input is not among the delimiters,
// we have to explicitly clear it after retrieving the coordinates
input.next();
```

Abbildung 7: Verwendung der Scanner Klasse zum Einlesen von Koordinaten der Form *H0*.

```
 |A|B|C|D|E|F|G|H|I|J
0|X|1|0|0|0|0|0|2|X|X
1|2|3|1|1|1|1|1|3|X|4
2|X|3|X|2|1|X|1|2|X|2
3|2|4|X|2|2|3|4|3|3|2
4|X|3|3|2|3|X|X|X|2|X
5|2|X|2|X|3|X|4|2|2|1
6|1|1|2|1|2|1|2|1|1|0
7|1|1|1|1|1|2|2|2|X|1|0
8|2|X|2|1|X|X|3|1|1|0
9|2|X|2|1|3|X|2|0|0|0
```

Abbildung 8: Ausgabe des Spielfelds mit allen Feldern aufgedeckt.

```
 |A|B|C|D|E|F|G|H|I|J
0| | | | | | | | | |
1| | | | | | | | | |
2| | | | | | | | | |
3| | | | |2|3| | | |
4| | | | | | | | | |
5| | | | | | | | | |1
6| | | | | | | | | |
7| | | | | | | | | |
8| | | | | | | | | |1|
9| | | | | | | | | |

Enter next coordinate (e.g., "H4"): C0
Enter action (1 for 'move onto', 2 for 'defuse'): 1
Moving onto 'C0'
 |A|B|C|D|E|F|G|H|I|J
0| | |0| | | | | | |
1| | | | | | | | | |
2| | | | | | | | | |
3| | | | |2|3| | | |
4| | | | | | | | | |
5| | | | | | | | | |1
6| | | | | | | | | |
7| | | | | | | | | |
8| | | | | | | | | |1|
9| | | | | | | | | |
```

Abbildung 9: Beispiel für die Auswahl von Aktionen.

```
 |A|B|C|D|E|F|G|H|I|J
0| | |0| | | | | | |
1| | | | | | | | | |
2| | | | | | | | | |
3| | | | |2|3| | | |
4| | | | | | | | | |
5| | | | | | | | | |1
6| | | | | | | | | |
7| | | | | | | | | |
8| | | | | | | | | |1|
9| | | | | | | | | |

Enter next coordinate (e.g., "H4"): A0
Enter action (1 for 'move onto', 2 for 'defuse'): 2
Trying to defuse 'A0'
Defused a landmine!
 |A|B|C|D|E|F|G|H|I|J
0|!| |0| | | | | | |
1| | | | | | | | | |
2| | | | | | | | | |
3| | | | |2|3| | | |
4| | | | | | | | | |
5| | | | | | | | | |1
6| | | | | | | | | |
7| | | | | | | | | |
8| | | | | | | | | |1|
9| | | | | | | | | |
```

Abbildung 10: Beispiel für das Entschärfen einer Mine.