Übung 10: Schrittweise Verfeinerung

1. Schiffe Versenken

Punkte: 24

Lernziel: In dieser Aufgabe sollen Sie das Prinzip der *schrittweisen Verfeinerung* vertiefen und bei der Programmierung des Spiels *Schiffe versenken* anwenden.

Implementieren Sie eine (vereinfachte) Variante des Spiels *Schiffe versenken* auf der Konsole. Zunächst wird ein 2D-Spielfeld generiert, auf der eine fixe Anzahl an Schiffen zufällig platziert wird. Anschließend versucht der Spieler die Positionen der Schiffsteile auf dem Spielfeld zu erraten, um sie so zu "versenken". Das Spiel endet, sobald alle Schiffsteile "versenkt" wurden.

Ein Spielablauf soll folgendermaßen aussehen:

- Wenn das Spiel gestartet wird, wird vom Benutzer eingelesen, wie viele Schiffe platziert werden sollen (maximal 4).
- Es wird ein leeres Feld mit der Größe 4x4 erzeugt.
- Anschließend werden darauf zufällig Schiffe mit der Größe 1x2 platziert (Höhe 1, Breite 2). Auf bereits besetzten Feldern kann kein weiteres Schiff platziert werden, d.h. Schiffe können sich nicht "überlappen", sie dürfen sich jedoch gegenseitig berühren ("direkt nebeneinander sein").
- Das Spielfeld wird auf der Konsole ausgegeben. Die Reihen und Spalten sind jeweils von 0 bis 3 nummeriert. Vor dem ersten Raten sind alle Zellen verdeckt und sollen mit Stern (`'*'`) dargestellt werden.
- Der Spieler beginnt nun, mittels Reihen- und Spaltenangabe eine Zelle des Spielfelds aufzudecken:
  - Stellen Sie zunächst sicher, dass es sich um gültige Koordinatenwerte handelt (0, 1, 2 oder 3).
  - Bei ungültigen Koordinatenwerten oder bei Eingabe von Koordinaten eines bereits aufgedeckten Feldes soll eine Meldung ausgegeben werden und anschließend neue Koordinaten abgefragt werden.
  - Ansonsten wird das Feld aufgedeckt.
  - Falls durch das Aufdecken ein Teil eines Schiffes ("Schiffsteil") getroffen wurde, soll eine entsprechende Meldung ausgegeben werden.
  - Unabhängig von einem Treffer soll immer die verbliebene Anzahl an Schiffsteilen ausgegeben werden. Anfangs gibt es $2 * n$ Schiffsteile, wobei n die Anzahl an platzierten Schiffen ist.
  - Abschließend wird das aktualisierte Spielfeld auf der Konsole ausgegeben, wobei nicht aufgedeckte Zellen mit dem Zeichen `'*'`, aufgedecktes Wasser mit dem Zeichen `'o'` und aufgedeckte Schiffsteile mit dem Zeichen `'X'` dargestellt werden.
- Das Spiel endet, wenn alle Schiffsteile getroffen wurden. Der Spieler soll auf der Konsole eine Meldung darüber erhalten.

Wenden Sie bei der Problemlösung die *schrittweise Verfeinerung* an. Starten Sie nicht direkt mit der Implementierung, sondern überlegen Sie zuerst, wie die Hauptmethode des Spiels aussehen muss und welche Datenstrukturen und Funktionen Sie benötigen, um die Spielzüge durchzuführen. Zerlegen Sie das Hauptproblem wiederholt in Teilprobleme, bis diese klein genug sind und einfach (beispielsweise in einer einzelnen Methode) gelöst werden können. Beschreiben Sie Ihre Vorgehensweise sowie die einzelnen Programmbestandteile ähnlich wie in den Folien der Vorlesung und geben Sie diese Beschreibung zusätzlich zu Ihrer Implementierung ab.

Hinweise

- Bei der Planung der Klassen- und Datenstrukturen können Sie sich am `MineSweeper`-Beispiel orientieren, bei welchem ebenfalls Zellen auf einem Feld aufgedeckt werden müssen.
- Um zufällige ganze Zahlen im Bereich $[0, n-1]$ zu generieren, können Sie `Math.random()` verwenden, das Ergebnis skalieren und schließlich zu einem `int` casten:
  `(int) (Math.random() * n)`
- Sie dürfen die Methode `readNumber` aus dem `MineSweeper`-Beispiel für das Einlesen von Zahlen übernehmen und ggf. anpassen.

Beispielausgabe:

```
How many ships (size 1x2) do you want to place? (1-4) 6
Number must be in [1, 4]
How many ships (size 1x2) do you want to place? (1-4) 2
  0 1 2 3
0|* * * *|0
1|* * * *|1
2|* * * *|2
3|* * * *|3
  0 1 2 3

Please make your next guess:
Row: 1
Column: 2
You sunk a ship part!
There are 3 ship parts left.

  0 1 2 3
0|* * * *|0
1|* * X *|1
2|* * * *|2
3|* * * *|3
  0 1 2 3

Please make your next guess:
Row: 1
Column: 3
You sunk a ship part!
There are 2 ship parts left.

  0 1 2 3
0|* * * *|0
1|* * X X|1
2|* * * *|2
3|* * * *|3
  0 1 2 3

Please make your next guess:
Row: 1
Column: 1
There are 2 ship parts left.

  0 1 2 3
0|* * * *|0
1|* o X X|1
2|* * * *|2
3|* * * *|3
  0 1 2 3

Please make your next guess:
Row: 1
Column: 1
Selected cell is already visible!

Please make your next guess:
Row: 3
Column: 1
There are 2 ship parts left.

  0 1 2 3
0|* * * *|0
```

```
1|* o X X|1
2|* * * *|2
3|* o * *|3
  0 1 2 3

Please make your next guess:
Row: 9
Number must be in [0, 3]
Row: 2
Column: 1
There are 2 ship parts left.

  0 1 2 3
0|* * * *|0
1|* o X X|1
2|* o * *|2
3|* o * *|3
  0 1 2 3

Please make your next guess:
Row: 0
Column: 1
You sunk a ship part!
There is 1 ship part left.

  0 1 2 3
0|* X * *|0
1|* o X X|1
2|* o * *|2
3|* o * *|3
  0 1 2 3

Please make your next guess:
Row: 0
Column: 0
There is 1 ship part left.

  0 1 2 3
0|o X * *|0
1|* o X X|1
2|* o * *|2
3|* o * *|3
  0 1 2 3

Please make your next guess:
Row: 0
Column: 2
You sunk a ship part!
There are 0 ship parts left.

  0 1 2 3
0|o X X *|0
1|* o X X|1
2|* o * *|2
3|* o * *|3
  0 1 2 3

Congratulations, you won!
```

Zusätzliche Anweisungen

- Verwenden Sie zur Implementierung der Ein- und Ausgabe für alle Programme nur die Funktionen der beiden bereitgestellten Klassen `In` und `Out`, wie in der Übung gezeigt. Die entsprechenden Java-Dateien samt HTML-Dokumentation sind in Moodle unter *InOut.zip* zu finden.
- Implementieren Sie formatierte Ausgaben, indem Sie `String.format` in Kombination mit `Out.print` und `Out.println` verwenden, wie in der Übung gezeigt. Verwenden Sie als Referenz den Foliensatz `StringFormat.pdf` im Moodle.
- Sie dürfen, falls nötig, `Math`-Funktionen der Java-Bibliothek nutzen.
- Verwenden Sie für Variablen passende Datentypen Ihrer Wahl.
- Alle Namen (Klassennamen, Variablennamen, Methodennamen, etc.) sind in Englisch zu wählen.
- Formatierung, Namenswahl, etc. fließen in die Bewertung mit ein.
- Vermeiden Sie Codeduplikation.

Abzugeben

Geben Sie *eine .zip Datei* mit dem Namen kxxxxxxxx_UExx.zip ab (Beispiel: k01234567_UE10.zip). Darin muss enthalten sein:

- Der *Source Code* der aktuellen Übung (alle `.java` Dateien, keine `.class` Dateien).
- Ein Testprotokoll eines kompletten Spiellaufs (`Testprotokoll.txt` oder `Testprotokoll.pdf`). Testen Sie sowohl gültige als auch ungültige Eingaben.