UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Übung zur „Einführung in die Programmierung – Java-Projekt“, WS 21/22

Übungsleiter: Ingrid Schumacher <schumache@itm.uni-luebeck.de>
Klaus-Dieter Schumacher <schumacher@itm.uni-luebeck.de>

Aufgabenblatt 1

Übungsblatt vom: Sonntag, 14.11.2021
Abgabe der Übung: Sonntag, 28.11.2021 um 23:30 Uhr

Vorbemerkungen

Vorgabe: Basis für die Bearbeitung der Projektaufgabe ist das Rahmenprojekt `SchiffeVorgabe`, das Sie bereits bei der Bearbeitung des vorbereitenden Aufgabenblatts 0 aus Moodle geladen haben. Es enthält bereits diverse Klassen, siehe Abbildung 1. Diese Klassen sind bewusst auf mehrere Pakete verteilt, angelehnt an das MVC-Modell[1], das gerne eingesetzt wird, um eine grafische Oberfläche von der eigentlichen Applikation – hier dem Spiel – sauber zu trennen.

Quellcode `SchiffeVorgabe.zip` Bei dem Rahmenprojekt handelt es sich um ein compilierbares, lauffähiges Programm zum Schiffe versenken, das auf wesentliche Implementierungen reduziert wurde. Das Projekt soll von Ihnen in 5 Teilschritten zu einem einfachen Schiffe-Versenken-Spiel ergänzt werden.

Im Rahmenprojekt sind auch Elemente von Java enthalten, die nicht in der Vorlesung behandelt werden und dann natürlich auch nicht geprüft werden. Teile des Rahmenprogramms sollen Sie aber analysieren, um dessen Komplexität zu verstehen und Beispiele für die Programmierung zu bekommen.

[1] `http://de.wikipedia.org/wiki/Model_View_Controller`, MVC bedeutet Model-View-Controler

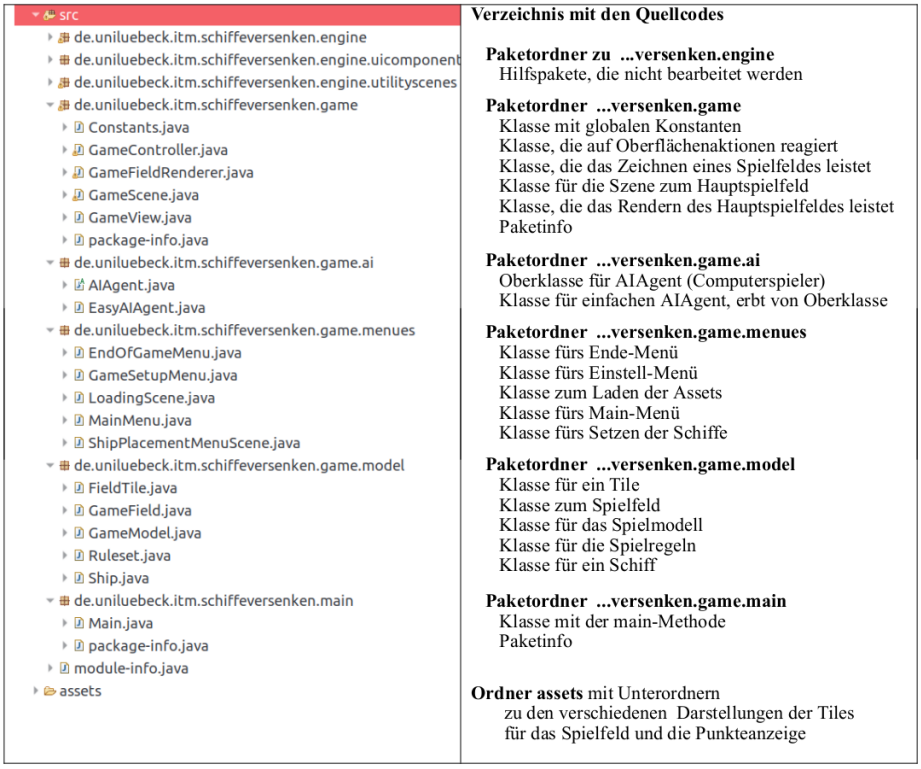

Abbildung 1: Klassen und ihre Bedeutung

Bearbeitungshinweise

- Bevor Sie beginnen zu programmieren, lesen Sie zunächst die gesamte Aufgabenstellung mit allen Teilaufgaben, damit Sie einen vollständigen Überblick über das von Ihnen zu erstellende Teilprojekt erhalten.
- Beachten Sie bei der Bearbeitung der Aufgaben die Informationen in den folgenden Dateien
  - `Projektablauf und Rahmenbedingungen.pdf`
  - `Style-Guide für das Erstellen von Java-Quellcode.pdf`
  - `Arbeiten mit Eclipse.pdf`, falls Sie Eclipse als IDE nutzen
- Bei der Ergänzung des Programms arbeiten Sie zunächst in den Klassen `Ship`, `GameField` und `GameFieldRenderer`, die sich in den Paketen `de.uniluebeck.itm.schiffeversenken.game.model` und `de.uniluebeck.itm.schiffeversenken.game` befinden, siehe auch Abbildung 1.
- Es ist sinnvoll, das Programm nach jeder Änderung auf seine Funktionalität zu testen!

- Lauffähige Zwischenstände sollten als Sicherheitskopien angelegt werden, damit bei späteren Fehlern damit weitergearbeitet werden kann. Da es in IDEs nicht möglich ist, mehrere Projekte gleichen Namens zu verwalten, werden die Kopien mit einer laufenden Nummer ergänzt: `<Gruppennummer>-SchiffeA1<.x>`. So entstehen Projektnamen wie `<Gruppennummer>-SchiffeA1.1` oder `<Gruppennummer>-SchiffeA1.2`, je nachdem wie viele Sicherheitskopien während der Bearbeitung des Aufgabenzettels von Ihnen angelegt wurden.
- Nur korrekt benannte, als zip-Archiv-Datei exportierte und mit openjdk 11 erstellte Projekte werden bewertet.
- Debug-Textausgaben, z. B. mit `System.out.println()` sind vor der Abgabe zu entfernen oder auszukommentieren.
- In Moodle finden Sie pro Aufgabenblatt eine Datei `Fragen.txt`. Kopieren Sie diese in den src-Ordner des aktuellen Projektstands. Sie enthält Fragen zu den Aufgabenteilen. Diese sind von Ihnen zusätzlich zu beantworten.

Lehrziele

- Sie verwenden für Ihr Projekt einen korrekten Projektnamen.
- Sie ergänzen die Klasse `Ship`.
- Sie erstellen eine Klasse `MatrixTest` als vorbereitende Übung zur Arbeit mit einer 2D-Matrix.
- Sie erstellen das Spielfeld als 2D-Matrix.
- Sie zeichnen ein Raster in das Spielfeld.
- Sie beantworten die Fragen in der Datei `Fragen.txt`.
- Sie beachten den Java-Style-Guide.
- Sie haben die Dokumentation des Projektes mit Javadoc korrekt erstellt.
- Sie exportieren Ihre Lösung als `zip`-Datei mit dem Namen `<Gruppennummer>-SchiffeA1<.x>.zip`.
- Sie stellen Ihre zip-Datei in Moodle ein und `bestätigen die Erklärung zur Eigenständigkeit`.

Aufgabe 1.1: Klasse Ship vervollständigen

Öffnen Sie die Klasse `Ship` im Paket ...`schiffeversenken.game.model`.

Als Eigenschaften hat ein Schiff eine Länge, d.h. es erstreckt sich über eine Anzahl Kästchen im Spielfeld, und eine Ausrichtung, die vertikal oder horizontal sein kann. Zusätzlich muss die Anzahl der getroffenen Schiffsfelder festgehalten werden. Sind alle Schiffsfelder getroffen, so gilt das Schiff als gesunken.

a) Legen Sie dazu in der Klasse `Ship` folgende drei privaten Variablen für die Länge des Schiffes, für die Anzahl der Treffer und die Ausrichtung des Schiffes an. `True` soll für die vertikale, `false` für die horizontale Ausrichtung stehen.

   - `length` vom Typ `int`,
   - `hits` vom Typ `int`,
   - `orientation` vom Typ `boolean`.

b) Initialisieren Sie im Konstruktor, dem Länge und Ausrichtung übergeben werden, die drei Instanzvariablen.

c) Implementieren Sie dann die Methode `public boolean isSunken()`, die es erlaubt abzufragen, ob das Schiff bereits gesunken ist.

d) In der weiteren Methode `public void hit()` sollen die Treffer hochgezählt werden, jedoch nur, wenn das Schiff noch nicht gesunken ist. (siehe Frage in `Fragen.txt`)

e) Ergänzen Sie die Methode `public boolean isUp()` als get()-Methode zur privaten Variablen `orientation`.

Aufgabe 1.2: Vorbereitende Übung zum 2D-Array

Im Projekt werden die Spielfelder von Player und Computer als 2D-Arrays implementiert. Um Ihnen die Einarbeitung in das Projekt zu erleichtern, haben wir eine Übung zum Arbeiten mit den 2D-Arrays vorgeschaltet.
Im Projekt werden die 2D-Arrays in der äußeren Schleife in x-Richtung über die Spalten, in der inneren in y-Richtung über die Zeilen durchlaufen. Bitte verfahren Sie in der folgenden Übung ebenso:

a) Erstellen Sie im Paket ...`schiffeversenken.main` zunächst eine neue Klasse mit dem Namen `MatrixTest` und einer main() Methode.

b) Deklarieren Sie zwei Variablen vom Typ `int` für die Anzahl an Spalten und Zeilen und erzeugen Sie damit ein 2D-int-Array namens `field`. Initialisieren Sie die Anzahl an Spalten und Zeilen jeweils mit 10.

c) Legen Sie eine Methode zum Initialisieren der Matrix an. Zum Initialisieren verwenden Sie fortlaufende Zahlen, beginnend bei Null. Da Sie spaltenweise vorgehen, muss die Matrix wie folgt aussehen:

00 10 20 30 40 50 60 70 80 90
01 11 21 31 41 51 61 71 81 91
02 12 22 32 42 52 62 72 82 92
03 13 23 33 43 53 63 73 83 93
04 14 24 34 44 54 64 74 84 94
05 15 25 35 45 55 65 75 85 95
06 16 26 36 46 56 66 76 86 96
07 17 27 37 47 57 67 77 87 97
08 18 28 38 48 58 68 78 88 98
09 19 29 39 49 59 69 79 89 99

d) Erstellen Sie eine Methode zur Ausgabe der Matrix.
Da dies mit `System.out.println()` zeilenweise auf der Konsole erfolgen soll, müssen Sie die Matrix vor der Ausgabe transponieren, also Zeilen und Spalten vertauschen. Sie können dazu mit einer zweiten Matrix arbeiten, die die transponierte Form aufweist.
Sobald die Ausgabe korrekt ist, ändern Sie die Methode bitte so ab, dass die Werte alle zweistellig ausgegeben werden, damit die Matrix auch "schön" aussieht.
Hinweis: Testen Sie auch die nicht transponierte Ausgabe und überlegen sich, warum diese so ausfällt. (siehe Frage in `Fragen.txt`)

e) Implementieren Sie eine Methode, die die Zahlen auf der Hauptdiagonalen auf einen zu übergebenden Wert abändert.

f) Berechnen Sie in einer weiteren Methode die Spaltensummen und geben diese auf der Konsole aus.

g) Eine Methode soll die Zahlenwerte der ersten Zeile auf 0 setzen.

h) Erzeugen Sie in der main()-Methode ein Objekt der Klasse und rufen nacheinander die implementierten Methoden auf.

i) Überprüfen Sie die Korrektheit Ihrer Konsolen-Ausgabe.

Aufgabe 1.3: Spielfelder erstellen

In dieser Aufgabe sollen Sie die Klasse `GameField` im Paket `...schiffeversenken.game.model` vervollständigen, damit die Spielfelder in ihrer vorgesehenen Größe (siehe Übergabeparameter) erzeugt werden.
Ein Spielfeld ist dann ein zweidimensionales Array aus FieldTiles, d.h. jedes Kästchen wird durch ein FieldTile-Objekt repräsentiert.

a) Sehen Sie sich dazu zunächst die Klasse `FieldTile` mit der darin befindlichen Aufzählung (enum) an.

   - Ein FieldTile (ein Kästchen in der Matrix) hat einen Status und einen Verweis auf ein zugehöriges Schiff, sobald dies auf ihm platziert wurde.
   - Beim Instanziieren des Feldes bekommen alle Tiles zunächst den Status "Wasser"(`FieldTileState.STATE_WATER`) und keinen Verweis auf ein Schiff, d.h. dem Verweis wird `null` zugewiesen.
   - Über die zugehörigen get()- und set()-Methoden können Sie später den Status und den Verweis bearbeiten.
   - Die Methode `bombard()` wird aufgerufen, wenn ein FieldTile getroffen wird, liefert dann true oder false, je nach Status des getroffenen FieldTiles.

b) Wechseln Sie in die Klasse `GameField` und analysieren den vorhandenen Code. Es gibt drei Variablen `size`, `field` und `ships`. `Size` vom Typ `Vec2` (siehe ↪Programming manual, Übersicht Engine) liefert die Anzahl der Spalten (x) und Zeilen (y) für das Erzeugen des 2D-FieldTile-Arrays namens `field`. In `ships` sollen die Schiffe verwaltet werden. Der Konstruktor der Klasse erwartet die Größe (size) des zu erzeugenden Spielfelds.

c) Damit das Rahmenprogramm lauffähig wurde, haben wir im Konstruktor ein einzelnes Kästchen als "Matrix" erzeugt. Ersetzen Sie die vorhandenen drei Zeilen mit Ihrem Code. Gehen Sie dazu wie folgt beim Implementieren des Konstruktors vor:

   - Initialisieren Sie die Instanzvariable `size`.
   - Erzeugen Sie das 2D-FieldTile-Array der Größe `size`.
   - Initialisieren Sie jedes Kästchen des 2D-Arrays jeweils mit einem neuen FieldTile-Objekt.
     Denken Sie daran, in der äußeren Schleife über die Spalten zu iterieren.
   - Abschließend ist die Variable `ships` mit einer neuen, leeren LinkedList zu initialisieren: `this.ships = new LinkedList<>()`.

d) Beantworten Sie die Fragen zu 1.3 in `Fragen.txt`.

Aufgabe 1.4: Raster zeichnen

a) Wechseln Sie in die Klasse `GameFieldRenderer` im Paket `...schiffeversenken.game`. Die Methode `renderGameField()` ist für das Zeichnen des Spielfeldes zuständig. In ihr werden für alle FieldTiles (Kästchen) des Spielfeldes deren Position und deren Status ermittelt und dann gezeichnet.

b) Analysieren Sie die bisherige Implementierung der Methode und beantworten Sie die dazu gestellten Fragen in der Datei `Fragen.txt`.

c) Zeichnen Sie dann an der entsprechenden TODO-Anweisung weiße Linien um das Spielfeld herum. Nutzen Sie zum Einstellen der Farbe `c.setColor(1, 1, 1);`. Für das Zeichnen der Linien verwenden Sie bitte die Methode `drawLine()` mit `Vec2`-Parametern aus der Klasse `Canvas` (siehe ↪Programming manual, Übersicht Engine).
Tipp: Informationen zum Positionieren des Rahmens erhalten Sie aus den Übergabeparametern und den Variablen `width` und `height`.

d) Zeichnen Sie als nächstes schwarze Linien als Trennung zwischen den Feldelementen. Für die schwarze Farbe verwenden Sie `c.setColor(0, 0, 0);`

Aufgabe 1.5: Java-Dokumentation

a) Die Kopfdokumentation der von Ihnen veränderten Klassen ist gemäß dem Beispiel im Java Style-Guide mit `modified by` anzupassen.

b) Die von uns noch nicht kommentierten Methoden sind von Ihnen einheitlich auf deutsch oder englisch mit Javadoc - Kommentaren zu versehen.

c) Zeilenkommentare sind geeignet einzufügen, um die eigene Implementation später besser nachvollziehen zu können.

d) Beantworten Sie die Fragen zu 1.5 in `Fragen.txt`.

e) Erstellen Sie für den abzugebenden Projektstand die Javadoc-Dateien:

- Markieren Sie den Projektnamen.
- Wählen Sie unter `Project` den Menüpunkt `Generate Javadoc ...` aus.
- Es erscheint das Unterfenster aus Abbildung 2 mit dem ausgewählten Projekt.
- Unter `Destination` wird der Ordner `doc` angegeben, in dem die doc-Dateien angelegt werden sollen. Dies soll ein Unterordner des Projektes sein, z. B.: `/home/student/workspace/<Gruppennummer>-SchiffeA1.1/doc`
- Beenden Sie die Auswahl mit dem Button `Finish`.
- Öffnen Sie im Ordner `doc` des Projektes die Datei `allpackages-index.html`. Prüfen Sie, ob die Dokumentation für alle Klassen in Ihrem Projekt erzeugt wurde.

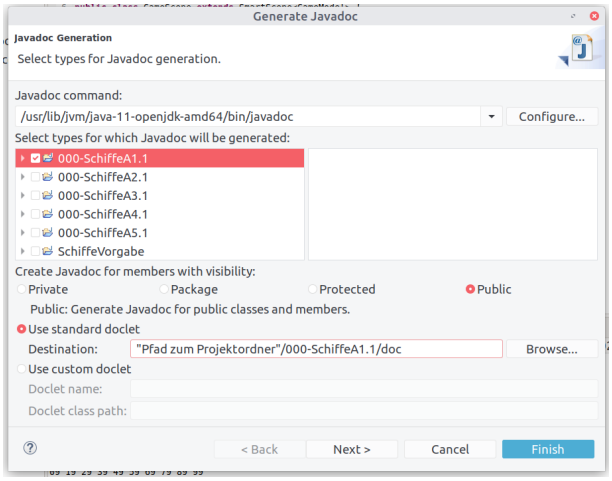

Abbildung 2: Angabe des doc-Ordners für die html-Dateien

Aufgabe 1.6: Packen und Hochladen

Exportieren Sie Ihr Projekt als zip-Datei <Gruppennummer>-SchiffeA1<.x>.zip und laden diese ins Moodle hoch. Sie **bestätigen die Erklärung zur Eigenständigkeit**. Zur laufenden Nummer einer Sicherheitskopie <.x> siehe Abschnitt **Bearbeitungshinweise**.

Die Abgabe ist bis Sonntag, 28.11.2021 um 23:30 Uhr durchzuführen.