Funktionale und objektorientierte Programmierkonzepte
Übungsblatt 02

TECHNISCHE UNIVERSITÄT DARMSTADT

Entwurf

Achtung: Dieses Dokument ist ein Entwurf und ist noch nicht zur Bearbeitung/Abgabe freigegeben. Es kann zu Änderungen kommen, die für die Abgabe relevant sind. Es ist möglich, dass sich **alle** Aufgaben noch grundlegend ändern. Es gibt keine Garantie, dass die Aufgaben auch in der endgültigen Version überhaupt noch vorkommen und es wird keine Rücksicht auf bereits abgegebene Lösungen genommen, die nicht die Vorgaben der endgültigen Version erfüllen.

Hausübung 02

Gesamt: 32 Punkte

Cleaning Convoy

Beachten Sie die Seite *Verbindliche Anforderungen für alle Abgaben* im Moodle-Kurs.

Verstöße gegen verbindliche Anforderungen führen zu Punktabzügen und können die korrekte Bewertung Ihrer Abgabe beeinflussen. Sofern vorhanden, müssen die in der Vorlage mit `TODO` markierten `crash`-Aufrufe entfernt werden. Andernfalls wird die jeweilige Aufgabe nicht bewertet.

Die für diese Hausübung relevanten Verzeichnisse sind `src/main/java/h02` und ggf. `src/test/java/h02`.

Verbindliche Anforderungen für die gesamte Hausübung:

- Die vorgegeben Roboter-Klassen sind wie vorgeschrieben zu verwenden, insbesondere ist die Verwendung der Superklasse `Robot` untersagt.

Hinweise (für die gesamte Hausübung)**:**

- In der `main`-Methode der Klasse `Main` finden sie ein Beispiel zur Verwendung der `ControlCenter`-Klasse.
- Sie können die `main`-Methode beliebig verändern, um Ihren Code zu testen. Sie müssen diese vor der Abgabe nicht zurücksetzen.

Einleitung

Nachdem Sie in der letzten Hausübung selber einen Roboter steuern mussten, um eine Welt aufzuräumen, soll dies in dieser Hausübung vollautomatisch über eine Kolonne an Robotern geschehen, die eine Welt erst auf darin vorhandene Münzen untersucht und diese anschließend automatisch einsammelt.

Hierzu sind zwei Arten an Robotern vorgegeben, die Sie verwenden müssen:

- `ScanRobot`:
  Diese Roboter dienen zur Erkennung von Münzen, sind jedoch nicht in der Lage, diese aufzusammeln. Die Methode `pickCoin` kann somit nicht verwendet werden.

- `CleanRobot`:
  Diese Roboter können Münzen aufsammeln, jedoch nicht feststellen, ob sich auf dem aktuellen Feld eine Münzen befindet, somit sind hier die Methoden `isOnACoin` und `isNextToACoin` deaktiviert. Da der Versuch des Aufsammelns ohne Münze auf dem Feld zu einem Crash führt, müssen diese Roboter also genau angesteuert werden, damit dies vermieden wird.

Diese Roboter können Sie bis auf die nicht vorhandene Funktionalität identisch zu normalen Robotern vom Typ `Robot` verwenden.

> **Ausblick:**
>
> Dies wird durch Vererbung umgesetzt. Mit diesem Grundkonzept der objektorientierten Programmierung werden Sie sich in Hausübung 03 befassen. Für unsere Zwecke ist ein tieferes Verständnis nicht notwendig, für Interessierte ist der relevante Foliensatz Kapitel 01f der FOP.

Die Kontrolleinheit wird von der `ControlCenter`-Klasse repräsentiert, in der Sie im Verlauf der Hausübung Methoden für das Aufräumen der Welt implementieren werden. Diese werden dann in der Methode `cleanWorld` verwendet.

Zur Speicherung der Roboter werden Arrays verwendet, um die Verwaltung der Roboter dynamisch an die Größe der Welt anpassen zu können und einfacher zu gestalten.

H1: Erstellen der Roboter-Arrays

Zuerst müssen die Roboter in der Welt platziert werden. Damit später noch auf sie zugegriffen werden kann, sollen ihre Referenzen in einem Array gespeichert werden. Es wird sowohl ein Array vom Typ `CleanRobot` als auch eines vom Typ `ScanRobot` erstellt werden.

Verbindliche Anforderungen (für beide Teilaufgaben)**:**

- Die Größe der Arrays muss minimal sein.
- Die Arrays dürfen keine `null`-Einträge enthalten.
- Die entsprechende x/y-Koordinate, je nach Typ der Roboter, soll sich mit steigendem Index im Array erhöhen.

Hinweis:

Auf die Abmessungen der Welt können sie über die Methoden `getWidth` und `getHeight` der Klasse `World` zugreifen.

H1.1: ScanRobots — 3 Punkte

Implementieren Sie die Methode `initScanRobots`. Diese Methode hat keine Parameter und soll ein Array vom Typ `ScanRobot` zurückliefern. Die Roboter sollen dabei folgendermaßen initialisiert werden:

- Die Roboter sollen die unterste Zeile der Welt bis auf das erste Feld füllen.
- Auf keinem Feld darf sich mehr als ein Roboter befinden.
- Die Roboter sollen alle nach oben ausgerichtet sein.
- Die Roboter besitzen zu Beginn keine Münzen.

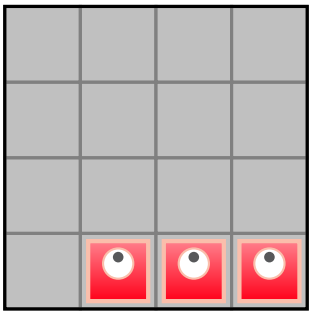

Abbildung 1: Beispiel für die Platzierung der ScanRobots bei einer Welt der Größe 4×3.

H1.2: CleanRobots 3 Punkte

Implementieren Sie die Methode `initCleaningRobots`. Diese Methode hat keine Parameter und soll ein Array vom Typ `CleanRobot` zurückliefern. Die Roboter sollen dabei folgendermaßen initialisiert werden:

- Die Roboter sollen die erste Spalte der Welt bis auf das erste Feld füllen.
- Auf keinem Feld darf sich mehr als ein Roboter befinden.
- Die Roboter sollen alle nach rechts ausgerichtet sein.
- Die Roboter besitzen zu Beginn keine Münzen.

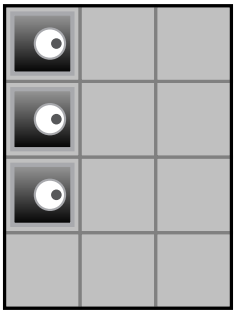

Abbildung 2: Beispiel für die Platzierung der ScanRobots bei einer Welt der Größe 3×4.

H2: Platzieren der Münzen 3 Punkte

In dieser Aufgabe geht es um das Platzieren der Münzen in der Welt, welche später eingesammelt werden sollen.

Hierfür soll die `void`-Methode `placeCoinsInWorld` in der Klasse `Main` implementiert werden. Diese hat den formalen Parameter `coins` vom Typ „Array von Array von `int`“. Dieser gibt an, wo und wie viele Münzen platziert werden sollen, indem der Eintrag `coins[y][x]` angibt, wie viele Münzen an der Position (y, x) in der Welt zu platzieren sind.

Hinweis:

Münzen können in der Welt mithilfe der Methode `putCoins` der Klasse `World` platziert werden. Der erste und zweite Parameter sind hierbei die x- und y-Koordinaten und der dritte Parameter ist die Anzahl der zu platzierenden Münzen. Die Verwendung von `putCoins` wäre normalerweise verboten (siehe verbindliche Anforderungen für alle Abgaben), ist für diese Teil-Aufgabe jedoch erlaubt.

Unbewertete Verständnisfrage:

Warum ist es bei einem 2-Dimensionalen Array sinnvoll, dass die y-Koordinate zuerst angegeben wird?

H3: Flip and rotate

In dieser Aufgabe geht es um verschiedene Arten der Rotation, die auf ein Array an Robotern durchzuführen sind.

H3.1: Invertierung des Roboter-Arrays 4 Punkte

Implementieren Sie die Methode `void`-Methode `reverseRobots` der Klasse `ControlCenter`. Diese hat den formalen Parameter `robots` vom Typ Array von `Robot`. Die Invertierung erreichen Sie, indem Sie zuerst in `robots` den Eintrag an der Position 0 mit dem Eintrag an der Position `robots.length-1` vertauschen. Danach soll der Eintrag an Position 1 mit dem Eintrag `robots.length-2` vertauscht werden, dies setzen Sie fort, bis das gesamte Array invertiert wurde.

Verbindliche Anforderungen:

- Es darf kein neues Array erstellt werden.
- Es dürfen nur die bereits vorhandenen Roboter verwendet werden.
- Die Roboter selbst werden **nicht** bewegt, es werden nur die Referenzen im Array vertauscht.

Hinweise:

- Es ist sinnvoll, sich mindestens eine Variable zum Zwischenspeichern eines Roboters zu erstellen.
- Sie können davon ausgehen, dass das Array nicht `null` ist.

H3.2: Rotation der Roboter 3 Punkte

Implementieren Sie die Methode `void`-Methode `rotateRobots`.

Die Methode soll die Roboter im übergebenen Array so umdrehen, so dass alle Roboter im Array nach Aufruf der Methode in die entgegengesetzte Richtung blicken. Die Roboter sollen, beginnend mit dem niedrigsten Index, rotiert werden. Anschließend soll die Methode `checkForDamage` mit dem soeben rotierten Roboter als Parameter aufgerufen werden. Diese Methode ist bereits implementiert und schaltet den Roboter eventuell ab, um eine Überprüfung auf Abnutzung zu simulieren, wobei ein kaputter Roboter natürlich nicht weiterarbeiten kann.

H3.3: Verschleiß behandeln 3 Punkte

Nachdem die Methode `rotateRobots` aufgerufen wurde, sind eventuell Roboter im ihr übergebenen Array ausgeschaltet, diese sollen nun in der Methode `replaceBrokenRobots` durch neue Roboter ersetzt werden.

Da die Roboter im Array entweder `CleanRobots` oder `ScanRobots` sein können, müssen Sie davon abhängig neue Roboter des jeweiligen Typs erstellen. Hierfür steht Ihnen optional die Methode `isScanRobotArray` zur Verfügung, die `true` genau dann zurückliefert, wenn das Array im aktualen Parameter vom Typ „Array von `ScanRobot`“ ist.

Ausblick:

`isScanRobotArray` funktioniert mithilfe des `instanceof`-Operators. Hiermit werden Sie sich noch in Kapitel 03b der FOP näher beschäftigen.

Die neuen Roboter im Array `robots` sollen also vom selben Typ wie die zu ersetzenden Roboter sein, da das Programm sonst nicht funktionieren würde. Auch soll jeder neue Roboter, der einen ausgeschalteten Roboter ersetzt, alle Attribute übernehmen.

Verbindliche Anforderung:

Die alten Roboter sollen durch neue Roboter ersetzt werden, die dieselbe Position und Ausrichtung haben. Die alten Roboter sollen dabei nicht verändert werden.

Hinweise:

- Ob ein Roboter ausgeschaltet ist, kann mit der Methode `isTurnedOff` überprüft werden.
- Sie können davon ausgehen, dass die Methode entweder ein Array an `CleanRobot`s oder an `ScanRobot`s erhält.
- Sie können davon ausgehen, dass kein Roboter im übergebenen Array `null` ist.

H3.4: spinning Robots — 1 Punkt

Implementieren Sie die Methode `spinRobots`. Diese setzt sich nur aus den vorherigen Methoden zusammen.

1. Das im aktualen Parameter übergebene Robot-Array soll zuerst mithilfe von `invertRobots` invertiert werden.
2. Anschließend sollen die Roboter mithilfe von `rotateRobots` rotiert werden.
3. Zuletzt sollen durch `replaceBrokenRobots` alle ausgeschalteten Roboter durch neue ersetzt werden.

H4: Kolonne marsch!

Zuletzt wollen wir uns mit der Hauptaufgabe der Robotern beschäftigen, dem Scannen und Aufräumen der Welt. Hierfür werden nun die Methoden zur Bewegung umgesetzt.

H4.1: Die heimkehrenden Helden — 2 Punkte

Zuerst benötigen wir die `void`-Methode `returnRobots`, welche den formalen Parameter `robots` vom Typ Array von `Robot` besitzt. Sie soll die Roboter in diesem Array einzeln bis an das Ende der Welt bewegen.

Beginnen Sie also mit dem in `robots` an Index 0 befindlichen Roboter und lassen Sie diesen laufen, bis der nächste Aufruf von `move` diesen auf ein Feld außerhalb der Welt bewegen soll. Nun soll mit dem Roboter an Index 1 dasselbe geschehen, danach mit dem an Index 2. Setzen Sie dies fort, bis alle Roboter im Array an das Ende der Welt bewegt wurden.

H4.2: Scannen der Welt — 5 Punkte

Implementieren Sie die Methode `scanWorld`, welche ein Objekt vom Typ „Array von Array von `boolean`“ zurückgibt. Diese besitzt einen Parameter vom Typ „Array von `ScanRobot`“ namens `robots` und soll die Welt „einscannen“. Dafür erstellen Sie zunächst ein Array an `boolean`-Arrays. Anschließend soll das soeben erstellte Array, im folgenden `coinPositions` genannt, folgendermaßen bearbeitet werden:

1. Zuerst soll jeder Eintrag `false` sein.
2. Für jede Koordinate (x, y) in der Welt, an der sich mindestens eine Münze befindet, soll anschließend der Eintrag `coinPositions[y][x]` auf `true` gesetzt werden.

Gehen Sie hierbei folgendermaßen vor:

1. Die Roboter sollen in einer Reihe in der Welt vorrücken. Beginnend mit dem Roboter am niedrigsten Index wird jeder Roboter ein Feld nach vorne bewegt, bis das Ende der Welt erreicht wird. Betritt hierbei ein Roboter ein Feld, auf dem sich eine Münze befindet, soll diese Tatsache sofort in `coinPositions` eingetragen werden.
2. Hat jeder Roboter das Ende der Welt erreicht, so soll die Methode `spinRobots` mit dem Roboter-Array als aktualem Parameter aufgerufen werden.
3. Wenn alle Roboter umgedreht sind, rufen sie als nächstes die Methode `returnRobots` mit `robots` als aktualem Parameter auf, damit die Roboter in ihre Startpositionen zurückkehren.
4. Zuletzt müssen die Roboter noch einmal mithilfe von `spinRobots` gedreht werden, damit sie für die nächste Iteration der Hauptschleife bereit sind.

H4.3: Bewegung der Putzkolonne — 5 Punkte

Implementieren Sie die Methode `moveCleanRobots`, diese besitzt den Parameter `robots` vom Typ „Array von `CleanRobot`“ und den Parameter `coinPositions` vom Typ „Array von Array von `boolean`“. Ziel dieser Methode besteht darin, mithilfe der Einträge in `coinPositions` Münzen in der Welt zu sammeln. Gehen Sie hierbei folgendermaßen vor:

1. Die Roboter sollen in einer Reihe in der Welt vorrücken. Beginnend mit dem Roboter am niedrigsten Index wird jeder Roboter ein Feld nach vorne bewegt, bis das Ende der Welt erreicht wird. Betritt hierbei ein Roboter ein Feld, auf dem sich eine Münze befindet, soll dieser diese sofort aufsammeln.

2. Hat jeder Roboter das Ende der Welt erreicht, so soll die Methode `spinRobots` mit `robots` als aktualem Parameter aufgerufen werden.

3. Nun sind Roboter umgedreht, rufen Sie nun also die Methode `returnRobots` auf, damit die Roboter in ihre Startpositionen zurückkehren.

4. Zuletzt müssen die Roboter noch einmal mithilfe von `spinRobots` gedreht werden, damit sie für die nächste Iteration der Hauptschleife bereit sind.