

Funktionale und objektorientierte Programmierkonzepte Übungsblatt 04



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Prof. Karsten Weihe

Wintersemester 23/24

Themen:

Relevante Foliensätze:

Abgabe der Hausübung:

v1.0

<Themen>

bis 01g

XX.XX.202X bis 23:50 Uhr

Hausübung 04

<Übungstitel>

Gesamt: 31 Punkte

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/h04` und ggf. `src/test/java/h04`.

Einleitung

H1: Move Strategies**?? Punkte**

In dieser Aufgabe beschäftigen Sie sich mit *Move Strategies* für Roboter, welche im weiteren Verlauf dieses Übungsblatts Verwendung finden. Als *Move Strategy* bezeichnen wir eine Strategie, mit welcher sich ein Roboter von seiner aktuellen Position aus zu einer gegebenen Position bewegt.

Erstellen Sie zunächst im Package `h03` ein Package `strategy`.

H1.1: Interface für Move Strategies**?? Punkte**

Erstellen Sie im Package `strategy` ein `public`-Interface `MoveStrategy`.

Deklarien Sie im Interface `MoveStrategy` eine rückgabelose Methode `start`, welche einen ersten formalen Parameter vom Typ `Robot` und einen zweiten formalen Parameter vom Typ `Field` hat. Die Methode `start` wird in Implementationen des Interface `MoveStrategy` aufgerufen, um einen gegebenen Roboter mittels der jeweiligen *Move Strategy* zu einer gegebenen Position zu bewegen.

H1.2: Interface für Move Strategies mit Counters**?? Punkte**

Erstellen Sie im Package `strategy` ein `public`-Interface `MoveStrategyWithCounter`, welche das Interface `MoveStrategy` erweitert.

Das Interface `MoveCounter` deklariert eine parameterlose Methode `getMoveCount`, welche einen Wert des Typs `int` liefert. Hierbei handelt es sich um die Anzahl der beim letzten Aufruf der Methode `start` mittels der Methode `move` der Klasse `Robot` durchgeführten Schritte.

H1.3: Move by Teleport**?? Punkte**

Nun implementieren Sie die erste *Move Strategy* *Move by Teleport*.

Erstellen Sie hierzu im Package `strategy` eine `public`-Klasse `MoveByTeleport`, welche das in H1.1 erstellte Interface `MoveStrategy` implementiert.

Implementieren Sie die Methode `start` in der Klasse `MoveByTeleport` so, dass der gegebene Roboter auf das gegebene Feld „teleportiert“ wird, indem die Position des Roboters mittels der Methoden `setX` und `setY` oder `setField` der Klasse `Robot` gesetzt wird.

Verbindliche Anforderung:

Die Methode `move` der Klasse `Robot` darf *nicht* verwendet werden.

H1.4: *Move by Walk*

?? Punkte

Nun implementieren Sie die zweite Move Strategy *Move by Walk*.

Erstellen Sie hierzu im Package `strategy` eine `public`-Klasse `MoveByWalk`, welche das in Aufgabe H1.2 erstellte Interface `MoveStrategyWithCounter` implementiert.

Implementieren Sie die Methode `start` aus dem Interface `MoveStrategy` in der Klasse `MoveByWalk` so, dass der gegebene Roboter auf das gegebene Feld *bewegt* wird, indem die Methoden `move` und `turnLeft` aus der Klasse `Robot` aufgerufen werden.

Weiter implementieren Sie die Methode `getMoveCount` aus dem Interface `MoveStrategyWithCounter` in der Klasse `MoveByWalk` folgendermaßen: Wenn `start` mindestens einmal aufgerufen wurde, liefert die Methode `getMoveCount` die Anzahl an Aufrufen der Methode `move` aus der Klasse `Robot` im *letzten* Aufruf von der Methode `start`. Andernfalls liefert die Methode `getMoveCount` den Wert `-1`.

Verbindliche Anforderung:

Die Methoden `setX`, `setY` und `setField` der Klasse `Robot` dürfen *nicht* verwendet werden.

H2: Field Selectors und Field Selection Listeners**?? Punkte**

H2.1: Interface für Field Selection Listeners**?? Punkte**

Erstellen Sie im Package `selection` ein `public`-Interface `FieldSelectionListener`.

Deklarieren Sie im Interface `FieldSelectionListener` eine rückgabefreie Methode `onFieldSelection`, welche einen formalen Parameter des Typs `Field` besitzt.

H2.2: Interface für Field Selectors**?? Punkte**

Erstellen Sie im Package `selection` ein `public`-Interface `FieldSelector`.

Deklarieren Sie im Interface `FieldSelector` eine rückgabefreie Methode `setFieldSelectionListener`, welche einen formalen Parameter des Typs `FieldSelectionListener` besitzt.

H2.3: Die Eingabe mit der Maus**?? Punkte**

Erstellen Sie eine Klasse `MouseFieldSelector`, welche das in der Aufgabe H2.2 erstellte Interface `FieldSelector` sowie das im Package `fopbot.FieldClickListener` gegebene Interface `FieldClickListener` implementiert.

Anmerkung:

Im Interface `FieldClickListener` ist als einzige Methode die Methode `onKeyPress` deklariert, welche einen formalen Parameter des Typs `FieldClickEvent` besitzt.

Wenn ein Objekt einer Klasse, welche das Interface `FieldClickListener` implementiert, beim zentralen *Input Handler* registriert ist, wird die Methode `onFieldClick` bei jedem *Field Click Event* (Klick auf ein Feld *World*) aufgerufen.

Dabei liefert der aktuelle Parameter der Methode `onFieldClick` Informationen über das *Field Click Event*: Die Methode `getField` der Klasse `FieldClickEvent` liefert den das Feld, welches angeklickt wurde.

Ein Objekt `object` einer Klasse, welche das Interface `FieldClickListener` implementiert, kann dem zentralen *Input Handler* folgendermaßen hinzugefügt werden:

```
1 InputHandler.addFieldClickListener(object);
```

Implementieren Sie die Methode `setFieldSelectionListener` aus dem Interface `FieldSelector` so, dass diese den gegebenen *Field Selection Listener* (Objekt des Typs `FieldSelectionListener`) einem von Ihnen in der Klasse `MouseFieldSelector` deklarierten Attribut zuweist.

Implementieren Sie die Methode `onFieldClick` aus dem Interface `FieldClickListener` folgendermaßen: Wenn die Methode `onFieldClick` *nicht* zum ersten Mal aufgerufen wird *und* das *Clicked Field* *dasselbe* wie beim letzten Aufruf ist, wird der *Field Click* als *Field Selection* gewertet.

Implementieren Sie den Konstruktor der Klasse `MouseFieldSelector` so, dass dieser das aktuelle Objekt dem zentralen *Input Handler* als *Field Click Listener* hinzufügt, indem Sie folgende Anweisung nutzen:

H2.4:

?? Punkte

Erstellen Sie im Package `selection` eine Klasse `KeyFieldSelector`, welche das in der Aufgabe H2.2 erstellte Interface `FieldSelector` sowie das im Package `fopbot` gegebene Interface `KeyPressListener` implementiert.

Anmerkung:

Im Interface `KeyPressListener` ist als einzige Methode die Methode `onKeyPress` deklariert, welche einen formalen Parameter des Typs `KeyPressEvent` besitzt.

Wenn ein Objekt einer Klasse, welche das Interface `KeyPressListener` implementiert, beim zentralen *Input Handler* registriert ist, wird die Methode `onKeyPress` bei jedem *Key Press Event* (Druck einer Taste auf der Tastatur innerhalb einer *World*) aufgerufen.

Dabei liefert der aktuelle Parameter der Methode `onKeyPress` Informationen über das *Key Press Event*: Die Methode `getKey` der Klasse `KeyPressEvent` liefert den *Key* (die gedrückte Taste auf der Tastatur in Form einer Konstanten der Enumeration `Key`). Die Methode `getWorld` der Klasse `KeyPressEvent` liefert die *World*, innerhalb welcher das *Key Press Event* ausgelöst wurde.

Ein Objekt `object` einer Klasse, welche das Interface `KeyPressListener` implementiert, kann dem zentralen *Input Handler* folgendermaßen hinzugefügt werden:

```
1 InputHandler.addKeyPressListener(object);
```

Wir bezeichnen ein Feld als *markiert*, wenn die Farbe dieses Feldes auf `Color.RED` gesetzt wurde und als *nicht markiert*, wenn die Farbe dieses Feldes nicht oder auf `null` ist. Weiter bezeichnen wir ein nicht markiertes Feld als *entmarkiert*, wenn dieses Feld zuvor markiert war. Sie können davon ausgehen, dass maximal ein Feld gleichzeitig markiert ist.

Implementieren Sie die Methode `onKeyPress` folgendermaßen:

Wenn die Methode `onKeyPress` zum ersten Mal aufgerufen wird, wird das Feld an Position (0, 0) markiert. Sie können davon ausgehen, dass in diesem Fall kein anderes Feld markiert ist.

Wenn ein Feld markiert ist *und* der *Key* des *Key Press Event* gleich der Konstanten `UP`, `LEFT`, `DOWN` oder `RIGHT` der Enumeration `Key` ist, wird das bisher markierte Feld entmarkiert und das *erste* Feld oberhalb, links, unterhalb bzw. rechts von dem zuvor markierten Feld markiert. Sollte sich an der jeweiligen Stelle kein weiteres Feld befinden, wird stattdessen das *letzte* Feld in der jeweiligen entgegengesetzten Richtung markiert.

Wenn ein Feld markiert ist *und* der *Key* des *Key Press Event* gleich der Konstanten `SPACE` der Enumeration `Key` ist, wird die Methode `onFieldSelection` des *Field Selection Listener* mit dem markierten Feld als aktuellen Parameter aufgerufen.

Hinweis:

Die Farbe eines Feldes kann mittels eines Aufrufs der Methode `setColor` der Klasse `Field` gesetzt werden, wobei als aktueller Parameter die jeweilige Farbe in Form eines Objekts der Klasse `Color` oder `null` verwendet wird.

H3: Mover**?? Punkte**

H3.1: Robot Mover**?? Punkte**

Erstellen Sie im Package mover eine **public**-Klasse RobotMover, welche das Interface FieldSelectionListener implementiert.

Zuerst implementieren Sie in der Klasse RobotMover einen **public**-Konstruktor, welcher einen formalen Parameter moveStrategy vom Typ MoveStrategy hat. Der aktuelle Parameter für moveStrategy muss innerhalb jeder anderen Methode der Klasse RobotMover abrufbar sein.

Implementieren Sie in der Klasse RobotMover eine rückgabelose **public**-Methode addRobot, welche einen formalen Parameter vom Typ Robot hat. Jeder aktuelle Parameter muss innerhalb jeder anderen Methode der Klasse RobotMover abrufbar sein.

Implementieren Sie in der Klasse RobotMover die Methode onFieldSelection aus dem Interface FieldSelectionListener so, dass diese die Methode move der dem Konstruktor gegebenen MoveStrategy mit *jedem* der Methode addRobot gegebenen Robot und dem aktuellen Parameter der Methode onFieldSelection aufruft. Wenn die Klasse des dem Konstruktor gegebenen Objekts weiter das Interface MoveCounter implementiert, legt jeder Roboter nach dem Aufruf von move so viele Münzen wie dieser Schritte durchgeführt hat. Andernfalls legt jeder Roboter nach dem Aufruf von move eine Münze.

H3.2: Movable Robot**?? Punkte**
