

Funktionale und objektorientierte Programmierkonzepte

Übungsblatt 04



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Prof. Karsten Weihe

Wintersemester 23/24

Themen:

Relevante Foliensätze:

Abgabe der Hausübung:

v1.0

Klassen und Interfaces mit FOPBot

bis 01g

24.11.2023 bis 23:50 Uhr

Hausübung 04

<Übungstitel>

Gesamt: 32 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`.

Verbindliche Anforderung: Dokumentieren Ihres Quelltexts

Alle von Ihnen deklarierten Klassen, Interfaces, Enumerationen und Methoden (inklusive Konstruktoren), die nicht `private` sind, *müssen* für diese Hausübung mittels JavaDoc in Englisch oder alternativ Deutsch dokumentiert werden. Für jede korrekte Deklaration ohne Dokumentation verlieren Sie jeweils einen Punkt.

Beachten Sie die Seite *Hausübungen* → *Dokumentieren von Quelltext* im Studierenden-Guide.

Verwenden Sie in Ihrem Quelltext 1:1 die auf diesem Übungsblatt gewählten Identifier! Andernfalls wird die jeweilige Aufgabe nicht automatisiert bewertet.

H1: Move Strategies**?? Punkte**

In dieser Aufgabe erstellen Sie Interfaces und Klassen für Move Strategies. Als *Move Strategy* bezeichnen wir eine Strategie, mittels welcher sich ein Roboter von seiner aktuellen Position aus zu einer gegebenen Position bewegt.

Erstellen Sie zuerst im Package `h04` ein Package `h04.strategy`.

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

Erstellen Sie nun im Package `h04.strategy` ein **public**-Interface `MoveStrategy`.

Deklaren Sie zuletzt im Interface `MoveStrategy` eine rückgabefreie Methode `start`, die einen ersten formalen Parameter vom Typ `Robot` und einen zweiten formalen Parameter vom Typ `Field` hat. Die Methode `start` wird für Objekte von Klassen, die das Interface `MoveStrategy` implementieren, aufgerufen, um einen gegebenen Roboter zu einer gegebenen Position zu bewegen.

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

Erstellen Sie im Package `h04.strategy` ein **public**-Interface `MoveStrategyWithCounter`, welche das Interface `MoveStrategy` erweitert.

Deklaren Sie im Interface `MoveStrategyWithCounter` eine parameterlose Methode `getMoveCount`, welche einen Wert des Typs `int` liefert. Die Methode `getMoveCount` liefert die Anzahl der beim letzten Aufruf der Methode `start` mittels der Methode `move` der Klasse `Robot` durchgeführten Schritte.

Nun implementieren Sie die verschiedenen Move Strategies.

H1.3: Move me by a Teleport!**?? Punkte**

Erstellen Sie im Package `h04.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 Field „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 me by a Walk!**?? Punkte**

Erstellen Sie zuerst im Package `h04.strategy` eine **public**-Klasse `MoveByWalk`, welche das in Aufgabe H1.2 erstellte Interface `MoveStrategyWithCounter` implementiert.

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

Implementieren Sie zuletzt 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 Selection Listeners**?? Punkte**

In dieser Aufgabe erstellen Sie Interfaces und Klassen für Field Selection Listeners. Als *Field Selection Listener* bezeichnen wir ein Element, welches Field Selection Events „beobachtet“. Ein *Field Selection Event* stellt die Auswahl eines Feldes in der Welt dar. Im weiteren Verlauf dieses Übungsblatts werden Sie sich mit *Field Selectors* beschäftigen, wobei ein *Field Selector* der Auswahl eines Feldes in der Welt dient. Das mit der Auswahl eines Feldes ausgelöste Field Selection Events

Erstellen Sie zuerst im Package `h04` ein Package `h04.selection`.

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

Erstellen Sie im Package `h04.selection` ein `public`-Interface `FieldSelectionListener`.

Deklariieren Sie im Interface `FieldSelectionListener` eine rückgabelose Methode `onFieldSelection`, welche einen formalen Parameter des Typs `Field` hat. Die Methode `onFieldSelection` wird von einem Field Selector aufgerufen, wenn mittels dieses Field Selector ein Field ausgewählt wurde, wobei als aktueller Parameter ebendieses Field verwendet wird.

H2.2: Robot Mover**?? Punkte**

Erstellen Sie zuerst im Package `h04.moveable` eine `public`-Klasse `RobotMover`, welche das Interface `FieldSelectionListener` implementiert.

Implementieren Sie nun in der Klasse `RobotMover` einen `public`-Konstruktor, welcher einen formalen Parameter vom Typ `MoveStrategy` hat. Der aktuelle Parameter muss innerhalb jeder anderen Methode der Klasse `RobotMover` abrufbar sein.

Implementieren Sie nun 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 zuletzt in der Klasse `RobotMover` die Methode `onFieldSelection` aus dem Interface `FieldSelectionListener` so, dass diese die Methode `start` der dem Konstruktor gegebenen `MoveStrategy` mit *jedem* der Methode `addRobot` gegebenen `Robot` und dem aktuellen Parameter der Methode `onFieldSelection` aufruft.

H2.3: Movable Robot**?? Punkte**

Erstellen Sie zuerst im Package `h04.moveable` eine `public`-Klasse `MoveableRobot`, welche direkt von der Klasse `Robot` abgeleitet ist und das Interface `FieldSelectionListener` implementiert.

Implementieren Sie nun in der Klasse `MoveableRobot` einen `public`-Konstruktor, welcher einen formalen Parameter vom Typ `MoveStrategy` hat und einen beliebigen Konstruktor der Basisklasse aufruft, welcher den Robot an Position (0,0) platziert. Der aktuelle Parameter muss innerhalb jeder anderen Methode der Klasse `MoveableRobot` abrufbar sein.

Implementiere Sie zuletzt in der Klasse `MoveableRobot` die Methode `onFieldSelection` aus dem Interface `Field` so, dass diese die Methode `start` der dem Konstruktor gegebenen `MoveStrategy` mit dem aktuellen Robot aufruft. Wenn die Klasse des dem Konstruktor gegebenen Objekts weiter das Interface `MoveStrategyWithCounter` implementiert, dreht der Roboter nach dem Aufruf von `start` so oft wie dieser Schritte ausgeführt hat.

H3: Field Selectors

?? Punkte

In dieser Aufgabe beschäftigen Sie sich mit *Field Selectors* und *Field Selection Listeners*. Ein *Field Selector* dient der Auswahl eines Feldes in der Welt. Ein *Field Selection Listener* wird bei einem *Field Selector* *registriert* und von diesem über *Field Selections* dieses *Field Selector* *benachrichtigt*.

Erstellen Sie zuerst im Package `h04` ein Package `selection`.

H3.1: Interface für Field Selectors

?? Punkte

Erstellen Sie im Package `h04.selection` ein `public`-Interface `FieldSelector`.

Deklariieren Sie im Interface `FieldSelector` eine rückgabelose Methode `setFieldSelectionListener`, welche einen formalen Parameter des Typs `FieldSelectionListener` hat. Die Methode `setFieldSelectionListener` wird von einem *Field Selection Listener* aufgerufen, um diesen bei dem jeweiligen *Field Selector* zu registrieren.

H3.2: Die Eingabe mit der Maus ...

?? Punkte

Ein *Mouse Field Selector* dient der Auswahl eines Feldes mittels der Maus. Der Nutzer wählt ein Feld aus, indem er das Feld in der Welt mit der Maus anklickt.

Erstellen Sie zuerst eine Klasse `MouseFieldSelector`, die das in der Aufgabe H3.1 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 mit der Tastatur auf ein Field in der *World*) aufgerufen.

Dabei liefert der aktuelle Parameter der Methode `onFieldClick` Informationen über das *Field Click Event*: Die Methode `getField` der Klasse `FieldClickEvent` liefert das Field, 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 nun die Methode `setFieldSelectionListener` aus dem Interface `FieldSelector` so, dass der aktuelle Parameter innerhalb jeder anderen Methode der Klasse `MouseFieldSelector` abrufbar ist.

Implementieren Sie nun die Methode `onFieldClick` aus dem Interface `FieldClickListener` so, dass wenn die Methode `onFieldClick` *nicht* zum ersten Mal aufgerufen wird *und* das Field *dasselbe* wie beim letzten Aufruf der Methode `onFieldClick` ist, das *Field Click Event* als *Field Selection* gewertet wird und damit die Methode `onFieldSelection` des registrierten Field Selection Listener mit dem Field des Field Click Event aufgerufen wird.

Implementieren Sie zuletzt in der Klasse `MouseFieldSelector` einen parameterlosen `public`-Konstruktor, welcher das aktuelle Objekt dem zentralen *Input Handler* als Field Click Listener hinzufügt.

H3.3: ... und der Tastatur

?? Punkte

Ein *Keyboard Field Selector* dient der Auswahl eines Feldes mittels der Tastatur. Der Nutzer wählt ein Feld aus, indem er mittels der Tasten W, A, S und D zu dem jeweiligen Feld navigiert und die Auswahl mittels der Leertaste bestätigt.

Erstellen Sie zuerst im Package `h04.selection` eine `public`-Klasse `KeyboardFieldSelector`, welche das in der Aufgabe H3.1 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 Field als *markiert*, wenn die Farbe dieses Field auf `Color . RED` gesetzt wurde und als *nicht markiert*, wenn die Farbe dieses Field nicht *oder* auf `null` ist. Weiter bezeichnen wir ein nicht markiertes Field als *entmarkiert*, wenn dieses Field zuvor markiert war. Sie können davon ausgehen, dass maximal ein Field gleichzeitig markiert ist.

Implementieren Sie nun die Methode `setFieldSelectionListener` aus dem Interface `FieldSelector` analog zur Klasse `MouseFieldSelector` so, dass der aktuelle Parameter innerhalb jeder anderen Methode der Klasse `MouseFieldSelector` abrufbar ist.

Implementieren Sie nun die Methode `onKeyPress` in der Klasse `KeyboardFieldSelector` folgendermaßen: Wenn die Methode `onKeyPress` zum ersten Mal aufgerufen wird, wird das Field an Position (0, 0) markiert. Sie können davon ausgehen, dass in diesem Fall kein anderes Field markiert ist. Wenn ein Field 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 Field entmarkiert und das *erste* Field oberhalb, links, unterhalb bzw. rechts von dem zuvor markierten Field markiert. Sollte sich an der jeweiligen Stelle kein weiteres Field befinden, wird stattdessen das *letzte* Field in der jeweiligen entgegengesetzten Richtung markiert. Wenn ein Field 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 Field als aktuellen Parameter aufgerufen.

Implementieren Sie zuletzt in der Klasse `KeyboardFieldSelector` einen parameterlosen `public`-Konstruktor, welcher das aktuelle Objekt dem zentralen Input Handler als Key Press Listener hinzufügt.

Hinweis:

Die Farbe eines Field 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.

H4: It's better together!**?? Punkte**

Nachdem Sie Move Strategies, Field Selection Listeners und Field Selectors implementiert haben, verknüpfen Sie diese Komponenten miteinander.

Implementieren Sie in Klasse `Main` die Methode `main01` folgendermaßen:

Zuerst wird ein Objekt der Klasse `MouseFieldSelector` und ein Objekt der Klasse `MoveByWalk` konstruiert. Danach wird ein Objekt der Klasse `RobotMover` konstruiert, wobei als aktueller Parameter das konstruierte Objekt der Klasse `MoveByWalk` verwendet wird. Zuletzt werden drei Objekte der Klasse `Robot` mit *beliebigen* Parametern konstruiert, welche dem Objekt der Klasse `RobotMover` mittels der Methode `addRobot` hinzugefügt werden.

Implementieren Sie in Klasse `Main` die Methode `main02` folgendermaßen:

Zuerst wird ein Objekt der Klasse `KeyboardFieldSelector` und ein Objekt der Klasse `MoveByTeleport` konstruiert. Danach wird ein Objekt der Klasse `MoveableRobot` konstruiert, wobei als aktueller Parameter das konstruierte Objekt der Klasse `MoveMyTeleport` verwendet wird.

Testen Sie Ihre Implementation, indem Sie wechselweise in der `main`-Methode die Methoden `main01` und `main02` aufrufen.