Funktionale und objektorientierte Programmierkonzepte
Übungsblatt 09

Prof. Karsten Weihe

Wintersemester 23/24	v1.0
Themen:	<Themen>
Relevante Foliensätze:	<1>
Abgabe der Hausübung:	XX.XX.202X bis 23:50 Uhr

Hausübung 09
<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/h09` und ggf. `src/test/java/h09`.

Verbindliche Anforderung:

- Der Typ `X` dient als Platzhalter für einen *beliebigen* Typen und existiert in der Vorlage nicht. Sofern nicht anders angegeben, *muss* Ihre Umsetzung für *jeden* beliebigen Typen funktionieren.
- Bei der Instanziierung eines Typparameters *muss* der Typ möglichst genau angegeben werden.

Einleitung

H1 ?? Punkte

Im Package `h09.stack` finden Sie die Klasse `StackOfObject`. Ein Objekt der Klasse `StackOfObject` stellt einen Stack (deutsch: „Stapel“) – eine dynamische[1] Datenstruktur, welche mit einem Stapel vergleichbar ist – dar. Das Modifizieren eines Stack ist auf das *Ablegen* eines Objekts auf den Stapel und das *Entfernen* des obersten Objekts von dem Stapel beschränkt. Die Klasse `StackOfObject` implementiert für beide Operationen folgende Objektmethoden: Die rückgabelose Objektmethode `push` legt den aktualen Parameter auf dem Stapel ab, die parameterlose Objektmethode `pop` entfernt das oberste Objekt von dem Stapel und liefert dieses Objekt.

H1.1: ?? Punkte

Überführen Sie die Klasse `StackOfObjects` in eine generische Klasse. Die Klasse `StackOfObjects` erhält zuerst einen unbeschränkten Typparameter `T`. Der formale Parameter der Methode `push` wird so ausgetauscht, dass als aktualer Parameter Objekte vom Typ `T` und Subtypen verwendet werden können. Der Rückgabetyp der Methode `get` wird so ausgetauscht, dass die Methode ein Objekt vom Typ `T` (inklusive Subtypen) liefern kann. Innerhalb der Methode `get` wird zuletzt für das gelieferte Objekt ein geeigneter Cast durchgeführt. Der Rückgabetyp der Methode `pop` wird so ausgetauscht, dass die Methode Objekte vom Typ `T` (inklusive Subtypen) liefern kann.

Überführen Sie die Klassenmethode `of` der Klasse `StackOfObjects` in eine generische Methode. Die Methode `of` soll für einen aktualen Parameter vom Typ „Array von `X`“ ein Objekt der Klasse `StackOfObjects` vom Typ `X` liefern. Passen Sie die innerhalb der Methode verwendeten Typen entsprechend an.

Unbewertete Verständnisfrage:

Analysieren Sie die Funktionsweise der Klasse `StackOfObjects`. Die Klasse `StackOfObjects` hat ein Objektattribut `objects` vom statischen Typ „Array von `Object`“, welches die Objekte des Stack enthält. Warum kann der statische Typ dieses Attributs nicht durch „Array von `T`“ ausgetauscht werden?

[1] In diesem Kontext bedeutet *dynamisch*, dass neue Objekte hinzugefügt und bestehende Objekte entfernt werden können.

H2 ?? Punkte

H2.1: ?? Punkte

In der Klasse `Functions` finden Sie eine Klassenkonstante `IS_NULL_PREDICATE` vom statischen Typ `Predicate`. Das von `IS_NULL_PREDICATE` dargestellte Predicate liefert für ein Objekt jeder beliebigen Klasse genau dann `true`, wenn der aktuale Parameter dieses Predicate `null` ist.

Überführen Sie die den statischen, nicht-instanziierten Typ der Klassenkonstante `IS_NULL_PREDICATE` in einen instanziierten Typ.

H2.2: ?? Punkte

In der Klasse `Functions` finden Sie die nicht-generischen Klassenmethoden `isInArea` und `hasMinimumNumberOfSeats`.

Die Methode `isInArea` hat einen formalen Parameter vom Typ `char` und den nicht-instanziierten Rückgabetyp `Predicate`. Diese liefert für den aktualen Parameter – im Folgenden auch *Standort-Präfix* genannt – ein Predicate, welches genau dann `true` liefert, wenn das erste Symbol des gegebenen Raumes gleich dem Standort-Präfix ist.

Überführen Sie beide Methoden in generische Methoden mit instanziierten Rückgabetypen.

H2.3: ?? Punkte

In der Klasse `Functions` finden Sie die nicht-generische Klassenmethode `isInAreaAndHasMinimumNumberOfSeats`. Diese Methode hat zwei formale Parameter: Der erste entspricht dem formalen Parameter der Methode `isInArea`, der ist vom Typ `int`. Die Methode `isInAreaAndHasMinimumNumberOfSeats` ruft mit dem ersten aktualen Parameter von der Methode `isInArea` ein Predicate ab und initialisiert mit dem zweiten aktualen Parameter – im Folgen auch *Mindestanzahl an Plätzen* genannt – ein Predicate, welches genau dann `true` liefert, wenn die Anzahl der Plätze des gegebenen Raumes größer oder gleich der Mindestanzahl an Plätzen ist. Die Methode `isInAreaAndHasMinimumNumberOfSeats` verknüpft beide Predicates logisch zu einem Predicate (mittels *AND*) und liefert dieses Predicate.

Überführen Sie die Methode `isInAreaAndHasMinimumNumberOfSeats` in eine generische Methode mit einem instanziierten Rückgabetyp.

H2.4: ?? Punkte

Im Package `h09.function` finden Sie ein Interface `StackPredicate`, welches das Interface `Predicate` erweitert.

Überführen Sie das Interface `StackPredicate` in ein generisches Interface und instanziieren Sie den Typen `Predicate`, welchen das Interface `StackPredicate` erweitert.

Eine Predicate, welches mittels eines Objekts der Klasse `StackPredicate` dargestellt wird, erhält als Eingabe ein Objekt der Klasse `StackOfObjects`. Ein solcher Stack of Objects enthält nur Objekte von dem Typ, mit dem der Typparameter des Interface `StackPredicate` instanziiert wurde.

H2.5: ?? Punkte

In der Klasse `Functions` finden Sie die nicht-generische Klassenmethode `toRoomTypeOrNull`. Diese hat einen formalen Parameter vom Typ `Class` und den Rückgabetyp `Function`. Die Methode `toRoomTypeOrNull` liefert für aktualen Parameter – ein Objekt der Klasse `Class`, welches den *Zieltyp* dargestellt – eine Funktion, welche als Eingabe ein Objekt vom Typ `Room` erhält. Wenn der Typ des eingegebenen Objekts vom Zieltyp ist, castet die Funktion das eingegebene Objekt zu einem Objekt des Zieltyps und liefert dieses Objekt. Andernfalls liefert die Funktion `null`.

Überführen Sie die Methode `toRoomTypeOrNull` in eine generische Methode mit einem instanziierten Typ des formalen Parameters und Rückgabetyp.

> **Anmerkung:**
>
> Ein Objekt der generischen Klasse `Class` stellt einen *beliebigen* Typen dar. Bei diesem Typen muss es sich *nicht* um eine Klasse handeln, auch wenn der Name „`Class`“ dies vermuten lässt. Der Typparameter der Klasse `Class` wird mit dem dargestellten Typen instanziiert. Für einen gegebenen Typ `Type` kann das dazugehörige Objekt der Klasse `Class` mittels `Type.class` abgerufen werden.

H3 ?? Punkte

H3.1: ?? Punkte

Überführen Sie die rückgabelose Klassenmethode `filter` in eine generische Klassenmethode mit einem Typparameter `T`.

Instanziieren Sie die Typparameter der formalen Parameter so, dass (1) der erste aktuale Parameter ein beliebiger Stack sein kann, aus welchem Objekte des Typs `T` gelesen werden können, (2) der zweite aktuale Parameter ein beliebiger Stack sein kann, in welchen Objekte des Typs `T` geschrieben werden können und (3) der dritte aktuale Parameter ein beliebiger Filter sein kann, welcher auf Objekte des Typs `T` angewendet werden kann. Passen Sie die innerhalb der Methode verwendeten Typen entsprechend an.

H3.2: ?? Punkte

Überführen Sie die rückgabelose Klassenmethode `map` in eine generische Klassenmethode mit zwei Typparametern `O` und `I`.

Instanziieren Sie die Typparameter der formalen Parameter so, dass (1) der erste aktuale Parameter ein beliebiger Stack sein kann, aus welchem Objekte des Typs `O` gelesen werden können, (2) der zweite aktuale Parameter ein beliebiger Stack sein kann, in welchen Objekte des Typs `I` geschrieben werden können und (3) der dritte aktuale Parameter eine beliebige Funktion sein kann, welche ein Objekt des Typs `O` auf ein Objekt des Typs `I` abbilden kann. Passen Sie die innerhalb der Methode verwendeten Typen entsprechend an.

H4: Testen mittels JUnit ?? Punkte

TODO Snippet, dass Tests in Verzeichnis test erstellt werden müssen.

> **Hinweis:**
>
> TODO Erinnerung entfernt TODO eventuell schon oben erwähnen?

H4.1: *Die großen 5, definiert von Stadtmitte und Lichtwiese.* ?? Punkte

Die fünf größten Hörsäle an der TU Darmstadt sind – *aufsteigend* sortiert nach Namen – L402/1, S101/A1, S105/122, S206/030 und S311/08, wobei der kleinste dieser fünf Hörsäle S105/122 mit 372 Plätzen ist.

Prüfen Sie, ob die Filter-Funktion aus der Aufgabe H3.1 zum Filtern der größten fünf Hörsäle an der TU Darmstadt funktioniert, indem Sie die Methode `filter` mit dem von der Klassenmethode `Rooms.stackOfLectureHalls()` gelieferten Stack und einem geeigneten Predicate aufrufen. Prüfen Sie zuerst mittels der Methode `assertEquals`, ob der von der Methode `filter` gelieferte Stack 5 Elemente enthält. Prüfen Sie dann mittels der Methode `assertEquals` für *jeden* der fünf größten Hörsäle, ob sich dieser an der erwarteten Position im Stack befindet, indem Sie den Hörsaal an der erwarteten Position abrufen und den Namen dieses Hörsals mit dem erwarteten Namen vergleichen.

H4.2: Test von map ?? Punkte

TODO keine map

H5: Weitere Tests ?? Punkte