

Programmieren III (Java)

2. Praktikum: Generics

Wintersemester 2023

Christopher Auer



Lernziele

- ▶ Arbeiten mit Generics
- ▶ Eigene generische Typen deklarieren
- ▶ Type-Bounds
- ▶ Wildcard-Operator mit Type-Bounds

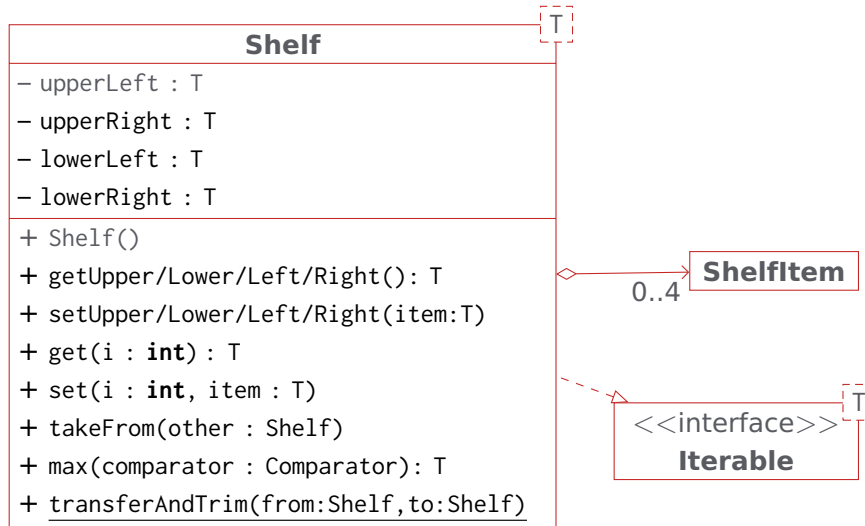
Hinweise

- ▶ Die Praktika sind eine *ausgezeichnete Prüfungsvorbereitung*; aber nur, wenn Sie sie *eigenständig* bearbeiten oder es zumindest *versuchen*. Nachvollziehen der Lösungsvorschläge *reicht nicht* aus.
- ▶ Bearbeiten Sie die Aufgaben *vor* dem Praktikumstermin.
- ▶ Im Praktikum können Sie *Ihre Lösung* zeigen und *Fragen* stellen.
- ▶ Je nach *zeitlichem Verlauf*, wird während des Praktikumstermins der *Lösungsvorschlag besprochen*.
- ▶ Der *Lösungsvorschlag* wird online gestellt, nachdem *alle Gruppen* das Praktikum durchlaufen haben.

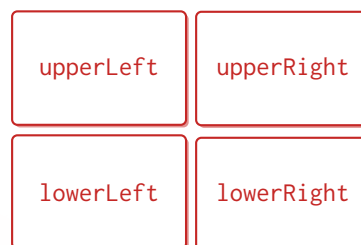


Aufgabe 1: Ein generisches Regal

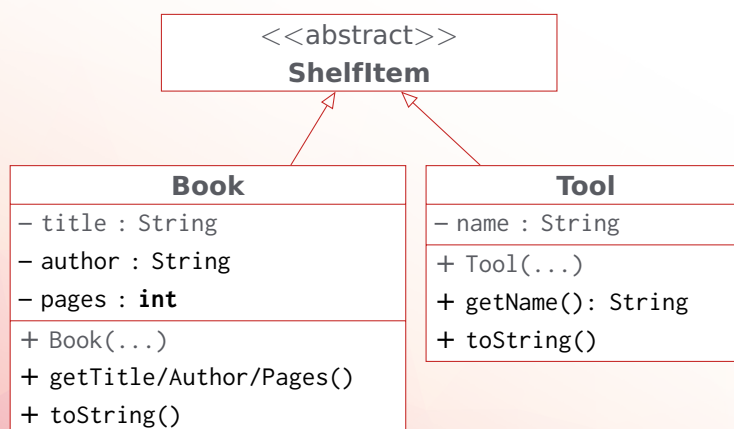
Gegeben sei folgendes UML-Diagramm für ein „*generisches Regal*“:



Die *generische Klasse* Shelf modelliert ein *Regal mit vier Fächern*. Die Attribute entsprechen dabei *Positionen im Regal*:



Fächer können dabei *leer* sein (`== null`), wobei der *Konstruktor* ein *leeres Regal* erstellt. Shelf kann Objekte vom Typ ShelfItem *oder abgeleitet* aufnehmen, wobei ShelfItem wie folgt definiert ist:



Von der *abstrakten Klasse* ShelfItem leiten zwei Klassen ab:

- Book für *Bücher* mit den *unveränderlichen Attributen* Titel, Autor und Seitenzahl.

- ▶ Tool für *Werkzeug* mit *unveränderlichem* Namen.

ShelfItem und Grundversion von Shelf

- ▶ *Deklarieren* Sie ShelfItem, Book und Tool mit den oben gezeigten *Methoden* und *Konstruktoren* zum *Initialisieren* noch *ohne Implementierung*.
- ▶ Erstellen Sie *JUnit-Testfälle* für Book und Tool (die Tests werden natürlich zunächst *scheitern*).
- ▶ Implementieren Sie die *Methoden* und prüfen Sie die *Korrektheit* mit Ihrer *JUnit-Tests*.
- ▶ *Deklarieren* Sie die *generische Klasse* Shelf die ShelfItem oder *abgeleitete Klassen* aufnehmen kann.
- ▶ *Implementieren* Sie die *Getter und Setter* der Fächer und den *Konstruktor*.
- ▶ Erstellen Sie einen *JUnit-Test* für die *bisherige Implementierung* von Shelf.

Zugriff von Shelf über Index

Für einen leichteren *Zugriff* auf die *Fächer* über einen *Index* gibt es die Methoden

```
public T get(int index)
public void set(int index, T item)
```

Index ist dabei ein Wert $0 \leq \text{index} < 4$ und *entspricht* den Fächern wie folgt:

0	1
2	3

- ▶ *Deklarieren* Sie die *Methodenköpfe* von *set* und *get* und schreiben Sie *JUnit-Tests* dafür
- ▶ *Implementieren* und *testen* Sie *set* und *get*

Shelf iterierbar machen

- ▶ *Machen* Sie Shelf *iterierbar* (\square Iterable), wobei die Inhalte des Regals in der Reihenfolge des *Index durchlaufen* werden (auch *leere Fächer* werden dabei vom \square Iterator geliefert). Sie können den Iterator als *innere Klasse* von Shelf implementieren.
- ▶ Schreiben Sie *JUnit-Tests* um *Ihre Implementierung* von \square Iterable und \square Iterator zu testen
 - ▶ Überlegen Sie sich wie Sie \square Iterator.next() und \square Iterator.hasNext() *testen können*.
 - ▶ Was muss passieren wenn next *aufgerufen wird* obwohl hasNext()==false? *Testen* Sie auch diesen Fall.

ShelfMain-Hauptprogramm

- ▶ *Erstellen* Sie eine Klasse ShelfMain mit einer *main-Methode*.
- ▶ *Implementieren* Sie die *statische Methode* printShelf in ShelfMain, die den *Inhalt* eines Regals mit Hilfe des vorher implementierten *Iterators* ausgibt (*Hinweis*: verwenden Sie den Wildcard-Operator für printShelf).

- **Erstellen** Sie in main ein *Buchregal* Shelf bookShelf mit Typparameter Book und dem *Inhalt*:

Index	Autor	Titel	Seiten
0	Ulllenbloom	Java ist auch eine Insel	1246
1	Schirach	Schuld	208
2		leer	
3	Börnstädt	Bibi und Tina	34

Geben Sie den *Inhalt* mit printShelf aus

- **Erstellen** Sie in main ein *Werkzeugregal*: Shelf toolShelf mit Typparameter Tool und dem *Inhalt*:

Schraubenzieher	
	Säge

Auch das Werkzeugregal *geben Sie mit printShelf aus*.


Die Methode takeFrom

Die Methode Shelf.takeFrom(other) *nimmt* die Gegenstände in other und *platziert* sie in dem Regal auf dem takeFrom *aufgerufen wurde*.

- **Deklarieren** Sie takeFrom noch *ohne Implementierung*! Verwenden Sie den *Wildcard-Operator* mit *Type-Bounds* für den Typparameter von other.
- Schreiben Sie nun *JUnit-Tests* für takeFrom.
- **Implementieren und testen** Sie takeFrom.
- **Erweitern** Sie main wie folgt (jeweils mit *Ausgaben*)
 - **Instanziieren** Sie Shelf<Book> newBookShelf und *befüllen* Sie den Inhalt mit Hilfe der Methode takeFrom mit dem Inhalt aus bookShelf (*oben*).
 - **Instanziieren** Sie Shelf<ShelfItem> generalShelf und *befüllen* Sie den Inhalt mit Hilfe der Methode takeFrom mit dem Inhalt aus newBookShelf.
- **Meckert** der Compiler? Prüfen Sie die *Type-Bounds* von takeFrom!

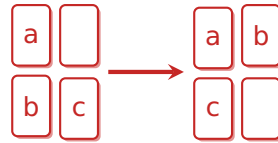
Die Methode max

Die Methode max(comparator) findet das *maximale Elemente* im Regal bezüglich des übergebenen  Comparator:

- **Deklarieren** Sie max noch *ohne Implementierung*
- Überlegen Sie sich, wie Sie max sinnvoll in einem *JUnit-Test* testen können. Implementieren Sie dann den Test.
- **Implementieren** Sie max und *testen* Sie Ihre Implementierung.
- **Erweitern** Sie main wie folgt:
 - **Stellen** Sie den ursprünglichen *Inhalt* von bookShelf wieder her (s. *Tabelle oben*).
 - **Rufen Sie** max mit einem  Comparator auf, so dass max das Buch mit den *meisten Seiten* liefert.
 - **Geben** Sie das *Ergebnis* mit printShelf aus.

Die Methode transferAndTrim

Die *statische Methode* transferAndTrim(from, to) übernimmt zwei Shelves und *durchläuft* die Fächer von from und *transfериert* den Inhalt *nicht-leerer* Fächer in das *nächste freie* Fach von to. Fächer von to werden dabei *vorher geleert*. *Beispiel*:



- ▶ *Deklarieren* Sie transferAndTrim noch *ohne Implementierung*! Welche *Type-Bounds* müssen die Parameter haben *um möglichst allgemein zu sein*?
- ▶ Schreiben Sie *JUnit-Tests* für transferAndTrim.
- ▶ *Implementieren* und *testen* Sie transferAndTrim.
- ▶ *Rufen* Sie transferAndTrim in main auf folgenden Shelf-Instanzen auf (jeweils mit *printShelf-Ausgabe* nach dem Transfer):
 - ▶ bookShelf → newBookShelf
 - ▶ newBookShelf → generalShelf
- ▶ *Meckert* der Compiler wieder? Prüfen Sie die *Type-Bounds* von transferAndTrim!