Hochschule Karlsruhe
University of Applied Sciences

Fakultät für **Informatik**
und Wirtschaftsinformatik

Prof. Dr. Carsten Sinz
Lehrgebiet Software-Entwicklung

Übungsblatt 7 zur Vorlesung Programmieren

Ausgabe: Di 07.07.2023 – Abgabe: Do 21.12.2023

Aufgabe 1 (Javadoc):

Javadoc erlaubt es, auf einfache Weise eine Dokumentation von Java-Programmen zu erzeugen durch Verwendung eines speziellen Kommentarstils. Javadoc-Kommentare können automatisch in HTML-Seiten konvertiert werden (mit dem Tool `javadoc`), in den meisten IDEs können darüber hinaus Javadoc-Kommentare auch als Tooltips (oder auf ähnliche Weise) angezeigt werden. Javadoc-Kommentare werden im Quellcode durch `/**` eingeleitet und mit `*/` beendet.

- Lesen Sie sich die Dokumentation zu Javadoc auf der Seite `https://docs.oracle.com/javase/1.5.0/docs/tooldocs/windows/javadoc.html` durch. Insbesondere der Abschnitt "Javadoc Tags" ist hier von Interesse.
- Wie werden bei generischen Datentypen die Typparameter dokumentiert?
- Welche Angaben sollten gemacht werden bei der Dokumentation von (a) Klassen, (b) Methoden und (c) Attributen?

Aufgabe 2 (Generisches Interface Stack):

Ein Stack (Stapel) ist ein abstrakter (Container-)Datentyp, der die folgenden Operationen unterstützt:

- `void push(E element)`: Ein neues Element vom Typ `E` wird auf den Stapel gelegt.
- `E pop()`: Das oberste Element wird vom Stapel entfernt und der Wert zurückgegeben.
- `E top()`: Der Wert des obersten Elements wird zurückgegeben, der Stapel bleibt unverändert.
- `boolean isEmpty()`: Gibt wahr zurück genau dann, wenn der Stack leer ist.
- `int size()`: Gibt die Anzahl der Elemente zurück, die auf dem Stapel liegen.

Dabei wird mit dem Parameter `E` der Typ der auf dem Stapel gespeicherten Elemente bezeichnet, d.h. der Datentyp soll generisch sein.

- Erstellen Sie ein generisches `interface Stack`, das die obigen Methoden enthält.
- Fügen Sie Javadoc-Kommentare für (a) das Interface und (b) die Methoden des Interface hinzu. Dokumentieren Sie insbesondere bei den Methoden sämtliche Parameter und den Rückgabewert, sowie beim Interface den Typparameter.

Aufgabe 3 (Implementierung generischer Stapel):

Erstellen Sie eine generische Implementierung des Interfaces `Stack` in einer Klasse `ArrayStack<E>`.

Gehen Sie dazu wie folgt vor:

a) Verwenden Sie ein Array zur Speicherung der Elemente. Die Größe des Array soll sich bei Verwendung dynamisch anpassen. Zum Start soll das Array ein Element aufnehmen können.

Wenn das Array vollständig gefüllt ist, soll sich beim nächsten einzufügenden Element seine Größe verdoppeln. Sobald das Array zu weniger als 40% gefüllt ist, soll sich seine Größe halbieren. Verwenden Sie die Felder (Attribute) `capacity` und `size` zur Speicherung der allozierten Arraygröße und der Anzahl aktuell belegter Arrayelemente.

b) Schreiben Sie einen Konstruktor, der einen leeren Stapel erzeugt. Dabei soll auch bereits ein Array mit Platz für ein Element angelegt werden.

c) Implementieren Sie die fünf oben genannten Methoden des Interfaces `Stack`. Überlegen Sie sich ein sinnvolles Verhalten für den Fall, dass die Methoden `pop()` und `top()` auf einem leeren Stapel aufgerufen werden.

d) Testen Sie Ihre Klasse mit zwei verschiedenen Elementtypen Ihrer Wahl (z.B. `Integer` und `String`). Für jeden Elementtyp soll jede der 5 Methoden mindestens einmal aufgerufen werden und der Stapel zu einem Zeitpunkt mindestens 5 Elemente enthalten. Am Ende des Tests soll der Stapel wieder leer sein.

e) Prüfen Sie mit Hilfe eines Debuggers anhand Ihrer Testprogramme, ob sich die dynamische Größe Ihres Arrays wie gefordert anpasst.

Hinweis: Es ist in Java nicht möglich, ein generisches Array `E[]` als Attribut einer Klasse anzulegen. Verwenden Sie stattdessen die Basisklasse `Object` für den Typ der Array-Elemente, d.h. ein `Object[]`-Array. Wenn Sie Elemente aus dem Array auslesen, können Sie diese in den Elementtyp `E` konvertieren. Dabei kann es beim Compilieren zu Warnungen *"uses unchecked or unsafe operation"* kommen. Diese können Sie durch Voranstellen des Attributes `@SuppressWarnings("unchecked")` vor die entsprechenden Methoden unterbinden.

Aufgabe 4 (Implementierung Stack mit Listen):

Erstellen Sie eine weitere Implementierung des Stack-Interfaces, `ListStack<E>`, die anstelle von dynamischen Arrays doppelt verkettete Listen zur internen Repräsentation verwendet. (Sie können hier die Klasse `LinkedList<E>` aus der Java-Standardbibliothek verwenden).

Erweitern Sie Ihre Klasse `ListStack<E>` so, dass sie das Interface `Iterable<E>` aus dem Paket `java.lang` implementiert. Testen Sie die Implementierung mittels einer *for-each*-Schleife, die alle Elemente eines Stacks (vom obersten zum untersten) ausgibt.