Wirtschaftsinformatik
Hochschule Ravensburg-Weingarten

Klausur

Objektorientierte Programmierung Praktikum

Studiengang

Wirtschaftsinformatik und E-Business (Bachelor)
WI PLUS

Prof. Dr. Thomas Bayer

Sommersemester 2019

(Bitte in Druckbuchstaben ausfüllen!)

Matrikel-Nr:	Datum: **08.07.2019**
Sitzplatz:	Semester: **SS 2019**
Hilfsmittel: **keine**	Bearbeitungszeit: **90 Minuten**

(Bitte nicht ausfüllen!)

	Unterschrift Prüfer:					Note:				
Aufgabe	CL	put	get	rem	cont	forE	size			Gesamt
max. Punkte	**15**	**25**	**15**	**15**	**10**	**15**	**5**			**100**
Punkte										

Ausgangssituation (Map)

Es soll die Datenstruktur Map auf Basis von Arrays implementiert werden. Eine Map kann man sich wie eine Tabelle vorstellen, in der zu jedem Schlüssel (key) ein Wert (value) vorhanden ist (ohne Reihenfolge). Bsp.:

Key	A	XY	Z
Value	Objektorientierte	Programmierung	Testat

Mit der Operationen put(k, v) wird zum Key k der Wert v aufgenommen. Mit get(k) wird der zum Schlüssel k gespeicherte Wert zurück gegeben. Eine Reihenfolge ist nicht vorhanden! Bsp.:

- `put(„A“, „Objektorientierte“); put(„XY“, „Programmierung“), put(„Z“, „Testat“)` führt zur obigen Tabelle
- `get(„XY“)` liefert „Programmierung“, `get(„XXX“)` liefert null, da zu dem Key „XXX“ kein Wert in der Map gespeichert ist

Inferface

```
public interface ISimpleMap<K, V> {

    // speichert zu dem Schlüssel k den Wert v, sofern beide nicht null sind
    // (ist mind. einer null, wird null zurück gegeben) .
    // Falls zu k bereits ein Wert w vorhanden ist, wird dieser durch v
    // ersetzt. Der alte Wert w wird zurückgegeben.
    // Falls zu k noch kein Wert gespeichert war, wird zu k der Wert v
    // aufgenommen und null zurückgegeben
    V put(K k, V v);

    // Liefert den zum Schlüssel k gespeicherten Wert. Liefert null,
    // falls k == null oder falls zu k kein Wert gespeichert ist
    V get(K k);

    // Entfernt den zum Schlüssel k gespeicherten Wert v und
    // liefert v zurück. Liefert null, falls k == null oder falls
    // zu k kein Wert gespeichert war
    V remove(K k);

    // Liefert true, falls der Wert v in der Map (zu irgendeinem Schlüssel)
    // gespeichert ist
    boolean contains(V v);

    // wendet die Methode consumer.accept(v) auf alle in der
    // Map gespeicherten Werte an
    void forEach(Consumer<? super V> consumer);

    int size(); // Anzahl an in der Map gespeicherten Werte
```

Aufgaben

- Implementieren Sie das Interface `ISimpleMap` auf Basis von Arrays (verwenden Sie keine Klassen aus `java.util`! – keine `ArrayList`, `LinkedList`, `HashMap`, etc.!!!!!)
- Verwenden Sie zwei Arrays, eines für die Schlüssel, eines für die Werte.
- Im Konstruktor muss die Größe der Arrays übergeben werden!

Vorbereitung

1. Starten Sie Eclipse mit dem Workspace im Verzeichnis C:\InsightFiles
2. Erstellen Sie das Projekt 2019-SS-Testat-##### (##### = Ihre Matrikelnummer)
3. Importieren Sie das jar-File und JUnit5 (Projekt->Properties)
4. Erstellen Sie das Paket solution#####
5. Erstellen Sie die Klasse SimpleMapTest#####, die von ASimpleMapTest erbt
6. Bringen Sie die Unit-Tests zum Laufen

Vorgehen

- Implementieren Sie das Interface ISimpleMap mit der Klasse SimpleArrayMap##### (##### = Ihre Matrikelnummer) – Nur die entsprechenden Methodenrümpfe!
- Passen Sie Ihre Testklasse so an, dass eine Instanz Ihrer Interfaceimplementierung übergeben wird
    a. Methode getInstance() – Rufen Sie den Konstruktor so auf, dass die beiden Arrays genau 5 Objekte aufnehmen können
    b. 24 Tests, 0 Errors, 10 Failures
- Programmieren Sie die Methoden aus
- Exportieren Sie Ihr Projekt als Archiv in das Verzeichnis C:\InsightFiles
- Vor Abgabe melden Sie sich!

Hinweise

- Ein Array der Größe n, mit Typ T wird wie folgt instanziiert: `(T[]) new Object[n];`
- Verwenden Sie ein Array für die Keys und eines für die Values (achten Sie auf die Reihenfolge in beiden Arrays)
- Führen Sie das Attribut size mit, das die Anzahl an gespeicherten Keys bzw. Values enthält
- Mit `a.length` wird die Größe eines Arrays a ermittelt, nicht die Anzahl an enthaltenen Objekten!
- Die Methode `contains(…)` bezieht sich auf die Werte, nicht die Schlüssel

Hinweise zur Benotung (100 Punkte)

- **Nachschlagen im eigenen Workspace (aus den Übungen): 0 Punkte**
- **(Versuchte) Manipulation der im jar-File enthaltenen Klassen/Interfaces: max. 25 Punkte**
- **Kopieren der im jar-File enthaltenen Klassen/Interfaces in ein eigenes Paket: 0 Punkte**
- **Programm nicht lauffähig (Syntaxfehler): max. 25 Punkte**
- **Verwendung von ArrayList, LinkedList, …, HashMap, etc. aus java.util: max. 25 Punkte !**
- Sowohl Unit-Tests als auch der Quelltext werden für die Benotung herangezogen
- Unit-Tests dienen nur zur Orientierung!