Programmierung A/B (in Java) – Übung 13

Thema: *Kreuz und quer durch den Vorlesungsinhalt*

Lernziele: *In dieser Übung sollen Sie Ihr über das Semester erworbene Wissen anwenden.*

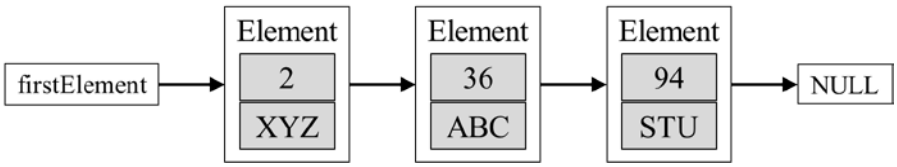
Hinweis: *Die mit * markierten Aufgaben sind nur für Studierende relevant, welche an Teil B der Veranstaltung teilnehmen. Das heißt natürlich nicht, dass man sie nicht trotzdem zur Übung bearbeiten kann ;-)*

Aufgabe 1

Schreiben Sie eine Funktion, die überprüft, ob ein eindimensionales Feld von chars ein Palindrom ist. Palindrome sind Wörter, die sich sowohl von links nach rechts, als auch umgekehrt identisch lesen, wie zum Beispiel HANNAH oder MADAM. Die Funktion soll true zurückgeben, wenn ein Feld ein Palindrom enthält, sonst false.
Lösen Sie diese Aufgabe ohne die Verwendung von Hilfsfunktionen aus anderen Java-Klassen.

Aufgabe 2

Gegeben seien die Klassen *SortedLinkedMap* und *Element* (die Java-Dateien sind in Moodle zu finden). Diese Klassen realisieren eine Map auf Basis einer sortierten verketteten Liste. Hierbei verwendet die Map Integer als Schlüssel und Strings als Werte. Schlüssel-Wert-Paare werden jeweils in Objekten vom Typ Element gespeichert, welche über eine Referenz auf das nachfolgende Element miteinander verkettet sind. Die Reihenfolge der Elemente soll dabei so realisiert sein, dass diese in aufsteigender Reihenfolge ihrer Schlüssel miteinander verbunden sind:



a) Vervollständigen Sie die Methode *find*, die einen Schlüssel als Parameter annimmt und innerhalb der Map nach einem Element mit dem angegebenen Schlüssel sucht. Existiert ein solches Element, so soll dieses Element zurückgegeben werden, andernfalls null.
b) Vervollständigen Sie die Methode *add*, welche einen Schlüssel und einen Wert als Parameter entgegennimmt und dieses Paar als Element an der richtigen Stelle in der Liste einfügt, sodass die Sortierung erhalten bleibt. Für den Fall, dass ein Element mit dem angegeben Schlüssel bereits existiert, soll nur dessen Wert geändert werden. Überlegen Sie sich außerdem, was Sie bei einer leeren Map zu beachten haben.
c) Verändern Sie die Funktionsweise der Methode *size* mit Hilfe einer zusätzlichen privaten Hilfsmethode so, dass die Größe der Map endrekursiv ermittelt wird. In der parameterlosen Methode *size* soll Ihre Hilfsmethode mit den geeigneten initialen Parametern aufgerufen werden.

Aufgabe 3*

Gegeben sei folgende Klassenhierarchie:

```
public abstract class Base { public void base() {...} }

public class SubclassA extends Base { public void subA() {...} }

public class SubclassB extends Base { public void subB() {...} }

public class SubclassB1 extends SubclassB { public void subB1() {...} }

public class SubclassB2 extends SubclassB { public void subB2() {...} }
```

Schreiben Sie eine Klasse *MyList*, welche die generische Klasse *ArrayList* so erweitert, dass *MyList* **ausschließlich** Objekte vom Typ *Base* speichert.

Implementieren Sie nun innerhalb Ihrer Klasse eine generische Methode *addSubclass*, welche ein Objekt eines beliebigen, von *SubclassB* abgeleiteten Typs als Parameter akzeptiert und void zurückgibt. Zuerst soll auf dem Objekt die *base()* Methode der Oberklasse *Base* aufgerufen werden. Nachfolgend soll das Objekt der Liste hinzugefügt werden. Abschließend soll in Abhängigkeit des tatsächlichen Typs des Objekts die in der Klasse implementierte Funktion aufgerufen werden (z.B. ruft ein Objekt vom Typ *SubclassB1* die Methode *subB1()* auf).

Aufgabe 4*

Überführen Sie den Lambda-Ausdruck in Zeile 3 in eine äquivalente Definition einer anonymen Klasse. Wie sieht das zugehörige Interface zu diesem Lambda aus?

```
int a = 5;
int b = 7;
IFunctional ops = (x, y) -> System.out.println(x*y);
ops.execute(a, b);
```