Name: Matrikelnummer:

Bachelorprüfung: Programmiermethoden und Techniken

WS14/15

Erlaubte Hilfsmittel: keine

Lösung ist auf den Klausurbögen anzufertigen. (eventuell Rückseiten nehmen)

Bitte legen Sie den Studentenausweis auf den Tisch.

Bearbeitungszeit: 90 Minuten

Unterschrift

Benotung

Aufgabe:	1	2	3	4	5		Gesamt	Note
Punkte:	20	20	16	24	20		100	
erreicht:								

Aufgabe 1 (20 Punkte)
(Iteratoren in Java)
In dieser Aufgabe sollen Sie zwei Klassen schreiben, die die Schnittstelle `Iterator` implementieren.

a) (8 Punkte)
Schreiben Sie eine Klasse `CountDown`. Die Klasse soll die Schnittstelle `Iterator<Integer>` implementieren.

Die Klasse soll im Konstruktor eine positive Zahl erhalten.

Bei der Methode `next` soll die im Konstruktor übergebene Zahl zurückgegeben werden und für den nächsten Aufruf um eins verringert werden.

Der Iterator endet die Iteration, wenn die 0 erreicht ist.

Beispiel: Das Objekt `new CountDown(5)` iteriert über die Folge $5, 4, 3, 2, 1, 0$.

b) (8 Punkte)
Schreiben eine Klasse `DoublingIterator<A>`, die `Iterator<A>` implementiert.

Sie soll ein Objekt des Interfaces `Iterator<A>` im Konstruktor übergeben bekommen.

Ein Objekt der Klasse `DoublingIterator<A>` soll über doppelt so viele Elemente iterieren, wie der im Konstruktor übergebene Iterator. Hierzu soll jedes Element des übergebene Iterators zweimal hintereinander von der Methode `next` zurück gegeben werden.

Iteriert also der ursprüngliche Iterator `it` über die Elemente: $x_1, x_2, \ldots x_n$ dann soll der der `new DoublingIterator<A>(it)` über die Folge $x_1, x_1, x_2, x_2, \ldots x_n, x_n$ iterieren.

c) (4 Punkte)
Schreiben Sie unter verwendung der Klassen aus a) und b) eine Klasse `DoubleCountDown`, einen `Iterator<Integer>` realisiert, der für eine Startzahl einen Countdown durchführt, allerdings jede Zahl zweimal hintereinander zurück gibt.

Beispiel: Das Objekt `new DoubleCountDown(5)` iteriert über die Folge $5, 5, 4, 4, 3, 3, 2, 2, 1, 1, 0, 0$.

Aufgabe 2 (20 Punkte)
(XML)

a) (8 Punkte)
Schreiben Sie mit dem DOM API (Paket `org.w3c.dom`) folgende Methode:
`public static void collectTagNames(Node node, Set<String> result);`
Sie soll alle Tagnamen von Elementknoten des übergebenen XML-Baums in die Ergebnismenge einfügen.

b) (8 Punkte)
Gegeben sei folgendes Interface

```
1 interface Property<A>{
2   boolean test(A o);
3 }
```

Listing 1: Property.java

Schreiben Sie mit dem DOM API folgende Methode:

`static int countNodesWithProperty(Node n, Property<Node> p)`
Es soll berechnet werden, für wie viele Knoten des übergebenen XML-Baums der Methode `test` des `Property`-Objekts true ergibt.

c) (4 Punkte)
Schreiben Sie eine Methode:
`static int countNodeWith10ChildNodes(Node n)`
Sie soll berechnen, wie viele Knoten in dem übergebenen XML-Baum genau 10 Kindknoten haben.

Benutzen Sie bei der Implementierung einen Lambda-Ausdruck und die Methode `countNodesWithProperty`.

Aufgabe 3 (16 Punkte)
(Strings in C)
Gegeben sei die folgende Struktur, die String-Objekte realisiert:

```
1 typedef struct{
2   unsigned int length;
3   char* data;
4 } String;
```

Listing 2: String.c

Schreiben Sie folgende Funktionen in C:

a) `unsigned int countCharsWithPorperty(String this, bool test(char))`

Es soll die Anzahl der Zeichen im String, für die die übergebene Funktion `true` ergibt, berechnet werden.

b) `unsigned int countCapitalLetters(String this)`
Es soll berechnet werdem, wieviel Großbuchstaben in dem String enthalten sind. Hierzu soll eine Hilsfunktion geschrieben werden und dann die Funktion `countCharsWithPorperty` mit einem Funktionszeiger aufgerufen werden.

Aufgabe 4 (24 Punkte)

Gegeben sei folgende Struktur, die eine Liste von ganzen Zahlen ausdrückt.

```
1 typedef struct {
2   int* array;
3   unsigned int capacity;
4   unsigned int size;
5 } AL;
```

Listing 3: AL.h

Entwerfen Sie in dieser Aufgabe eine kleine C-Bibliothek für Array-Listen. Das Attribut `array` dient dabei als Datenspeicher und hat die maximale Kapazität, die im Attribute `capacity` gespeichert ist. Das Attribute `size` gibt an, wie viele Elemente in der Liste gespeichert sind.

a) Schreiben Sie eine Konstruktorfunktion,
`AL* newAL();`
mit der ein Zeiger auf ein `AL`-Objekt erzeugt wird. Das erzeugte Heap-Objekt soll eine leere Liste repräsentieren. Die Kapazität des Arrays soll initial 10 sein.

b) Schreiben Sie eine Destruktorfunktion,
`void deleteAL(AL* this);`
mit der ein AL-Objekt vollständig aus dem Heap gelöscht wird.

c) Schreiben Sie eine Funktion `void add(AL* this, int elem)`, die in einer Liste ein Element am Ende einfügt. Berücksichtigen Sie den Fall, in dem die Kapazität des Arrays nicht mehr ausreicht.

d) Schreiben Sie eine Funktion `void forEach(AL* this, void consumer(int))`, die die übergebene Funktion `consumer` auf jedes Listenelement anwendet.

Aufgabe 5 (20 Punkte)

Erklären Sie in kurzen Worten.

a) Was muss ein C Programmierer beachten, wenn er mit dem Heap arbeitet im Gegensatz zur Arbeit mit dem Stack?

b) Nennen sie die XPath-Achsen, die nur Knoten bezeichnen, die im Dokment vollständig vor dem Ausgangsknoten liegen.

c) Was ist der Unterschied zwischen diesen beiden C-Funktionen?

```c
void f1(int x){
  x = 2*x;
  printf("%d\n",x);
}
void f2(int* x){
  *x = 2 * *x;
  printf("%d\n",*x);
}
```

Listing 4: A4c.c

d) In der Schnittstelle `java.util.stream.Stream<T>>` gibt es folgende Methode:
`void forEach(Consumer<? super T> action)`.
Erklären Sie mit einfachen Worten, was diese Methode mit den Elementen des Streams macht. Was für einen Vorteil kann der Aufruf dieser Methode unter Umständen gegenüber der Verwendung einer Schleife?

e) Was wird in Java 8 als default-Methoden bezeichnet?