P1 - Serie 7

Lukas Batschelet (16-499-733)

Stoff

Bis zu Kapitel 11:
Fokus: Methoden Planen und das Collection Framework

Allgemeine Informationen zur Abgabe

Die Abgabe erfolgt online auf ILIAS.
Lösungen zu den Theorieaufgaben muss als *.pdf Datei abgegeben werden. Andere Formate werden nicht akzeptiert.
Arbeit in Zweiergruppen: Geben Sie jeweils nur ein Exemplar der Lösung pro Gruppe ab. Geben Sie in jeder Quellcode-Datei die Namen und Matrikelnummern beider Gruppenmitglieder in den ersten beiden Zeilen als Kommentar an.
Vorbesprechung: 01.12.2023
Abgabe: 08.12.2023 13:00

Theorieaufgaben

(1) Methodenköpfe

Welche der folgenden Methodenköpfe repräsentieren tatsächlich unterschiedliche Signaturen:

1. `public String describe(String name, int count)`
   `public String describe(int count, String name)`
2. `public int count()`
   `public void count()`
3. `public int howMany(int compareValue)`
   `public int howMany(int upper)`
4. `public boolean greater(int val1)`
   `public boolean greater(double val1)`
5. `public boolean greater(int val1)`
   `public boolean greater(int val1, int val2)`
6. `public void say()`
   `private void say()`

Lösung:

1. unterschiedliche Signatur
2. nicht unterschiedlich da nur der Rückgabetyp unterschiedlich ist
3. nicht unterschiedlich da nur der Bezeichner des Parameters anders ist.
4. unterschiedliche Signatur
5. unterschiedliche Signatur
6. nicht unterschiedliche Signatur, da nur die Sichtbarkeit unterschiedlich ist.

(2) BizzBuzz

Geben Sie einen Algorithmus in Pseudo-Code an, der für die ganzen Zahlen von 1 bis 100 überprüft, ob diese durch 3 und/oder durch 5 teilbar sind. Falls die Zahl weder durch 3 noch durch 5 teilbar ist, soll die Zahl ausgegeben werden. Ist die aktuelle Zahl durch 3 teilbar, soll statt der Zahl “Bizz” ausgegeben werden. Ist die aktuelle Zahl durch 5 teilbar, soll statt der Zahl “Buzz” ausgegeben werden. Ist die Zahl sowohl durch 3 als auch durch 5 teilbar, soll “BizzBuzz” ausgegeben werden.

Mögliche Lösung:

pseudo

```
FÜR jede Zahl i von 1 bis 100:
    WENN i durch 3 teilbar ist UND i durch 5 teilbar ist:
        Gib "BizzBuzz" aus
    SONST, WENN i nur durch 3 teilbar ist:
        Gib "Bizz" aus
    SONST, WENN i nur durch 5 teilbar ist:
        Gib "Buzz" aus
    SONST:
        Gib die Zahl i aus
```

(3) Addition zweier Binärzahlen

Wir betrachten das Problem der Addition von zwei n-Bit Binärzahlen, die in zwei Arrays A und B der Länge n gespeichert sind. Die Summe der beiden Zahlen soll in einem Array C der Länge n + 1 gespeichert werden. Geben Sie zur Lösung dieses Problems einen Algorithmus in Pseudo-Code an.
Beispiel: Mit A = 101011 und B = 111110 soll C = 1101001 sein.

Mögliche Lösung:

```
DEFINIERE Algorithmus AddiereBinär(A, B)
    SEI n die Länge von A
    INITIALISIERE Array C mit Länge n + 1 auf Null

    SEI Übertrag = 0
    FÜR i = n − 1 BIS 0 (RÜCKWÄRTS):
        SEI Summe = A[i] + B[i] + Übertrag
        WENN Summe >= 2 DANN
            C[i + 1] = Summe − 2
            Übertrag = 1
        SONST
            C[i + 1] = Summe
            Übertrag = 0

    C[0] = Übertrag
    GIB C ZURÜCK
```

(4) Pseudocode übersetzten

Übersetzen Sie folgenden Pseudocode in eine statische Methode. Hinweis: Schreiben Sie eine Hilfsmethode für das Tauschen zweier Elemente in einer Liste.

Algorithm 1 shuffle(Liste list mit ganzen Zahlen)

```
n = Größe von list − 1
for i = n, n−1, ..., 2, 1 do
    r = Zufallszahl zwischen 0 und i (i inklusive)
    tausche die Elemente in der Liste an Position i und r
end for
gib die gemischte Liste list zurück
```

Mögliche Lösung:

```java
public static void shuffle(List<Integer> list) {
        int n = list.size() - 1;
        Random random = new Random();

        for (int i = n; i > 0; i--) {
                int r = random.nextInt(i + 1);
                swap(list, i, r);
        }
}

// Hilfsmethode
private static void swap(List<Integer> list, int i, int j) {
        Integer temp = list.get(i);
        list.set(i, list.get(j));
        list.set(j, temp);
}
```

(5) Welche Ausgabe erzeugt folgendes Programm:

```java
public class Parameter {
     public static void main(String[] args) {
        Language s1 = new Language("Java");
        Language s2 = new Language("Python");
        int i = 12345;
        pass(s1, s2, i);
        System.out.println(s1);
        System.out.println(s2);
        System.out.println(i);
    }
    private static void pass(Language s1, Language s2, int i) {
        s1 = new Language("Ruby");
        s1.increaseVersion();
        s2.increaseVersion();
        i = 54321;
    }
}
//-------------------------------------------------------
public class Language {
    private String name;
    private double version;
    public Language(String name) {
        this.name = name;
        this.version = 1.0;
    }
    public String toString() {
        return this.name + " " + this.version;
    }
    public void increaseVersion() {
        this.version += 0.1;
    }
}
```

Lösung:

```
Ruby 1.1
Python 1.1
12345
```

(6) Stack nach Operationen

Wie sieht der Stack s aus, nachdem folgende Operationen durchgeführt worden sind:

```
s.push(5);
s.push(21);
s.pop();
s.push(72);
s.push(37);
s.push(15);
s.pop();
```

Lösung:

```
37 <-- Top
72
5  <-- Bottom
```

(7) Queue nach Operationen

Wie sieht eine Queue q aus, nachdem folgende Operationen durchgeführt worden sind:

```
q.offer(5);
q.offer(21);
q.poll();
q.offer(72);
q.offer(37);
q.offer(15);
q.poll();
```

Lösung:

```
15 <-- last in
37
72 <-- first out
```

(8) Methode `set` in `OwnArrayList` schreiben

Betrachten Sie die Klasse OwnArrayList aus dem Skript. Schreiben Sie eine Methode `set(int index, Object object)`, welche das Element an Position index in der aktuellen Liste mit dem Element object überschreibt. Im Erfolgsfall geben Sie true zurück – falls der Parameter index zu groß ist (größer als die aktuelle Liste), geben Sie false zurück.

Mögliche Lösung:

```
public boolean set(int index, Object object){
        if (index >= this.size || index < 0)
                return false;
        listElements[index] = object;
        return true;
        }
}
```

(9) Methode `size` in `OwnLinkedList` schreiben

Betrachten Sie die Klassen Node und OwnLinkedList aus dem Skript. Schreiben Sie eine Methode `size()`, welche die Größe der Liste zurückgibt.

```
public int size(){
        if (this.startNode == null)
                return 0;
        Node<E> current = this.startNode;
        int count = 1;
        while (current.getNext() != null){
                current = current.getNext();
                count++;
        }
        return count;
}
```

(10) Methoden `get` und `set` in der Schnittstelle Collection

Weshalb werden die Methoden `get` und `set` nicht in der Schnittstelle `Collection` vorgegeben?

Lösung:

Die Methoden `get` und `set` sind speziell für Sammlungen des Typs `List` definiert, die eine geordnete und indizierte Struktur aufweisen. Dies ermöglicht es, auf Elemente an einem bestimmten Index zuzugreifen oder sie zu ändern. Im Gegensatz dazu sind die Sammlungstypen `Queue` und `Set`, die ebenfalls die `Collection`-Schnittstelle implementieren, von einer anderen Natur. Bei `Sets`, die eine Gruppe einzigartiger, nicht indizierter Elemente darstellen, und bei `Queues`, die auf dem Prinzip des ersten Hinein, ersten Heraus basieren, wären die Methoden `get` und `set` konzeptionell unpassend. Daher sind diese Methoden nicht in der allgemeinen `Collection`-Schnittstelle enthalten, sondern bleiben spezifisch für Listen, wo sie aufgrund der Natur der Datenstruktur sinnvoll und anwendbar sind.