PM2 Aufgabenblatt A05

Youssef Benlemlih, Jonas Krukenberg 05.05.2020

Java Denksportaufgaben

1. SuspiciousSort

Das Programm gibt `UNORDERED` aus. Erwarten würden wir eher die Ausgabe `DESCENDING`.

Die Ausgabe kommt dadurch zustande, weil `cmp.compare(int i1, int i2)` als Sortiermethode für `Array.sort` verwendet wird und da Array `arr` zufällige Integerzahlen zwischen -2^{31} und 2^{31} beinhaltet.

Durch dieses maximale Intervall kann es in der compare-Methode bei `i2 - i1` zu einem Integer Overflow kommen, wenn z.B. i2 = -2^{31} und i1 = 2^{31} ist. Dadurch kommt ein falsches Ergebnis zustande und die Zahlen werden falsch einsortiert.

Bei `i1=-2070774536` und `i2=2098106171` ergibt sich `(i2-i1)=-126086589`, das Ergebnis sollte aber positiv sein.

Um das richtigzumachen, können wir entweder die Methode `Integer.compare` benutzen oder bei sort den Comparator komplet weglassen. Wir delegieren die `compare` Methode an `Integer.compare(i2,i1)`, die zu keinem Overflow kommt, da sie keine Differenz berechnet.

```
// so
Comparator<Integer> cmp = new Comparator<Integer>() {
    @Override
    public int compare(Integer i1, Integer i2) {
        return Integer.compare(i2,i1);
    }
};
Arrays.sort(arr, cmp);
 // oder so
Arrays.sort(arr);
```

2. Pair<T>

- In der main-Methode wird `p` nicht parametrisiert, also Angabe des generischen Typs mit `Pair` deklariert. `p` ist dann ein RawType. RawTypes sind dazu da um backward compatibility mit den Java Versionnen die keine Gernerics unterschtützen zu gewährleisten.

Alternativ kann man den Typ `Object` benutzen, wovon andere Typen erben. Ohne Typeingabe kommt der Kompiler nicht damit klar.

In der Klasse `Pair`, wird in der Methode `stringList()`, die Methode `String.valueOf` verwendet.

Man muss also bei der Initialisierung von `p` den Typ deklarieren

```
// anstatt:
// Pair p = new Pair<Object>(23, "skidoo");
Pair<Object> p = new Pair<Object>(23, "skidoo");
```

3. LinkedList<E>

Das Problem hier liegt beim Typ-Parameter E, der sowohl für `LinkedList` als auch für `Node` verwendet wird. Dadurch ist `E` bereits bei der Initialisierung einer `LinkedList` gebunden, die innere Klase `Node` hat aber wieder einen ungebundenen Typ-Parameter E, wodurch `Node` und `LinkedList` unterschiedliche Typen haben können.

Gelöst wird das Problem, indem ein neuer Typnameg ewählt wird, was folgende innere Klasse zur Folge hat:

```
private class Node<T> {
    T value;
    Node<T> next;// Node constructor links the node as a new head

    @SuppressWarnings("unchecked")
    Node(T value) {
        this.value = value;
        this.next = (Node<T>) head;
        head = (Node<E>) this;
    }
}
```

Wir müssen dann im Konstruktor einmal `head` auf einen `Node<T>` und einmal `this` auf ein `Node<E>` casten, damit die korrekten Datentypen angesprochen werden.
`@Suppresswarnings("unchecked")` können wir hier verwenden, da durch `add(E e)` sichergestellt wird, dass `Node` und `LinkedList` letztendlich vom gleichen Typ sind.

Generische Klasse

1. Die Klasse MyDeque implementiert das Interface Deeque, dafür wir intern die Klasse MyEntry benutzt, was ein Element der List representiert.

Die Klasse `MyEntry`

Jedes Element der Liste ist mit einem Objekt der Klasse `MyEntry` representiert. Es hat einen (nicht veränderbaren) Wert `value` und einen Verweis auf das nächste Element in der List `next`. Damit zwei Elemente gleich sind, reicht es nicht dass sie den gleichen Wert darstellen, sondern die ganze Kette der nächsten Elemente soll gleich sein. Die Methoden `getHead()` die das letze und `getBeforeHead()`, die das vorletzte Element in der Liste liefern werden in der Klasse MyDeque nutzlich gemacht.

Die Klasse `MyDeque`

Die Klasse `MyDeque` hat zwei Konstruktoren:

- Einmal den leeren Konstruktor, um eine leere Liste zu erstellen.
- Einmal `public MyDeque(E... initialElements)`, der die den `Deque` mit Elemented bevölkert (populate)
  - Beispiel: `Deque<Integer> deque = new MyDeque(1,2)`

Intern, hat die Klasse `MyDeque` einen Verweis auf das erste Element `bottom` nur, das dann für den Zugriff auf anederen benutzt wird.

Zwei `MyDeque` sind genau gleich wenn sie die gleichen Elemente haben.

Quellen
Oracles Dokumentation zu Raw type