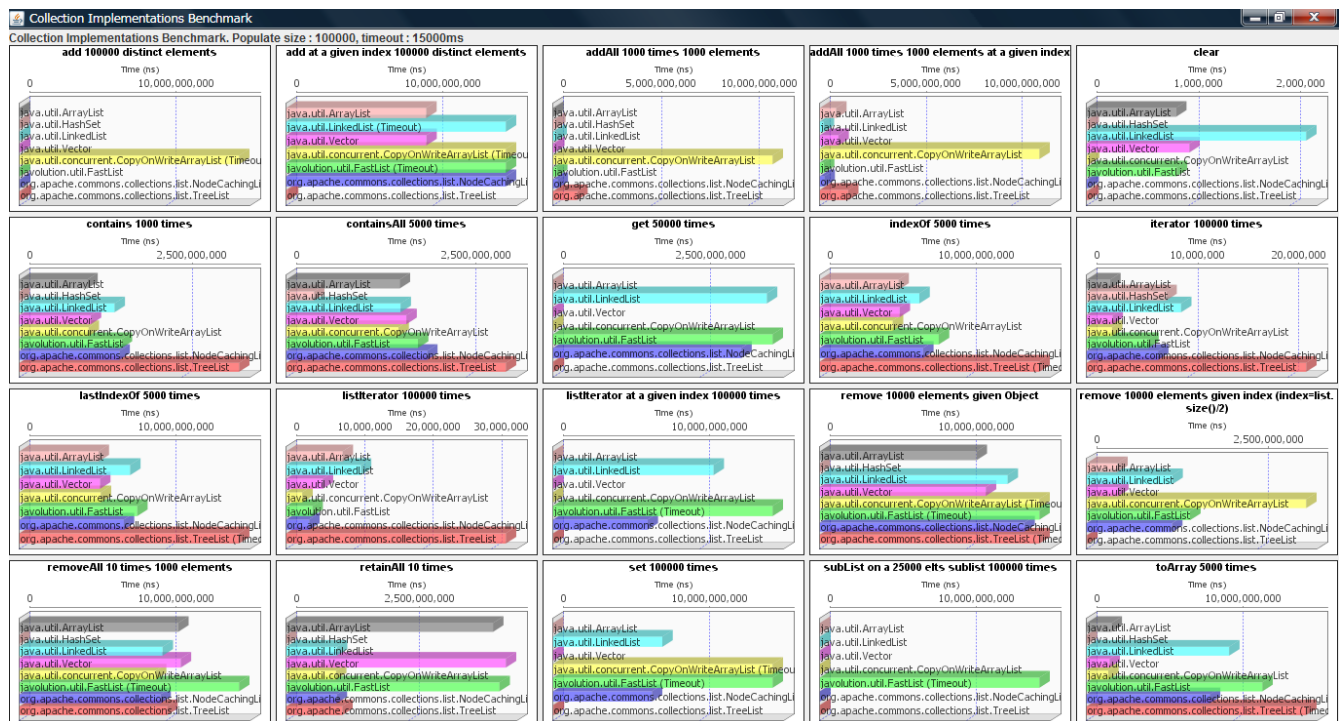


# Nachdenkzettel: Collections

## 1. ArrayList oder LinkedList – wann nehmen Sie was?

## 2. Interpretieren Sie die Benchmarkdaten von: <http://java.dzone.com/articles/java-collection-performance>. Fällt etwas auf?



## 3. Wieso ist CopyOnWriteArrayList scheinbar so langsam?

4. Wie erzeugen Sie eine thread-safe Collection (die sicher bei Nebenläufigkeit ist) (WAS?? die ArrayLists, Linkedlists, Maps etc. sind NICHT sicher bei multithreading??? Wer macht denn so einen Mist???)

5. Achtung Falle!

```
List<Integer> list = new ArrayList<Integer>;
```

```
Iterator<Integer> itr = list.iterator();
while(itr.hasNext()) {
    int i = itr.next();
    if (i > 5) { // filter all ints bigger than 5
        list.remove();
    }
}
```

Falls es nicht klickt: einfach ausprobieren...

Macht das Verhalten von Java hier Sinn?

Gibt es etwas ähnliches bei Datenbanken? (Stichwort: Cursor. Ist der ähnlich zu Iterator?)

6. Nochmal Achtung Falle: What is the difference between get() and remove() with respect to Garbage Collection?

7. Ihr neuer Laptop hat jetzt 8 cores! Ihr Code für die Verarbeitung der Elemente einer Collection sieht so aus:

```
Iterator<Integer> itr = list.iterator();
while(itr.hasNext()) {
    int i = itr.next();
    //do something with i...
}
```

War der Laptop eine gute Investition?

Für die Mutigen: mal nach map/reduce googeln!