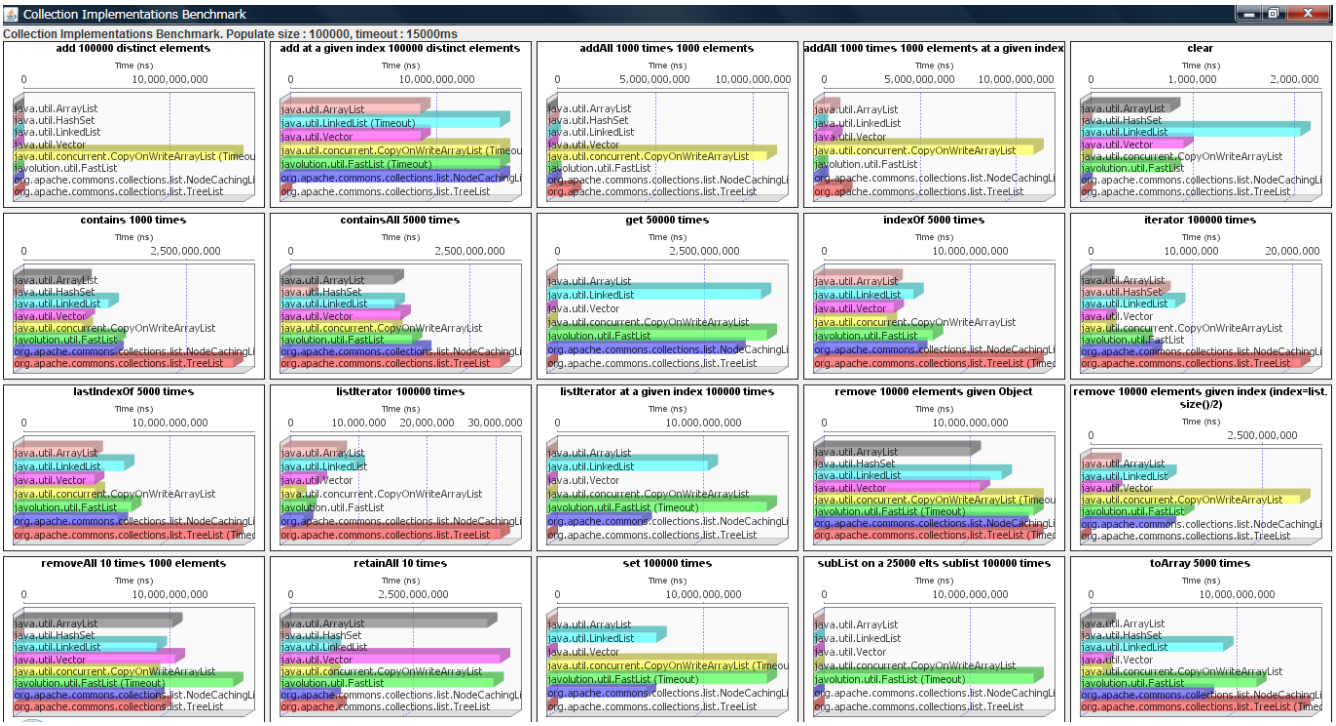
Nachdenkzettel: Collections

1. ArrayList oder LinkedList – wann nehmen Sie was?

If you need to store and retrieve data quickly, you should use an ArrayList.
LinkedList, on the other hand, is the tool to use if you need to alter previously saved data quickly.

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



- These benchmarks show that CopyOnWriteArrayList is extremely sluggish.
- When compared to Lists on contains(), HashSet is rather quick.
- While ArrayList isn't particularly sluggish in any situation, it isn't always the quickest.

3. Wieso ist CopyOnWriteArrayList scheinbar so langsam?

Every time a modification is made, a CopyOnWriteArrayList creates a new copy of the underlying array.
This is owing to its increasing popularity as a simple way to deal with synchronized data processing.

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???)

Collections.synchronizedCollection()

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?)

At first glance, Java's behavior here doesn't seem to make sense.
However, if you delve deeper into the subject, Java behaves as expected.

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

Both methods return the value associated with the specified index.
Remove also removes the entry from the list, whereas get does nothing.
If the object reference isn't stored somewhere else, the Garbage Collector must wipe it from ram.

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!

Only one of the eight cores is used in the above implementation.
So, no, it hasn't been a good investment by conventional measures.