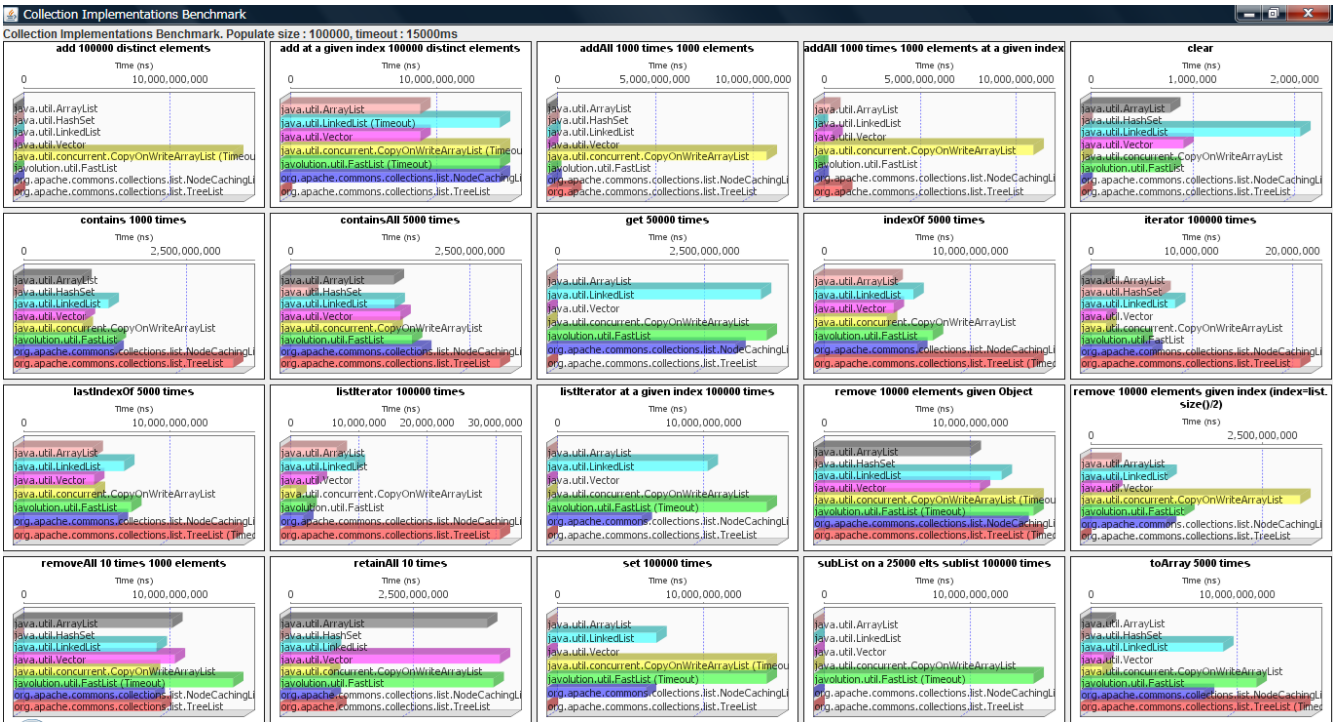
Nachdenkzettel: Collections

1. ArrayList oder LinkedList – wann nehmen Sie was?

ArrayList für eine Liste in dem die Variablen in der Liste nichtmehr vertauscht werden und auch nichts mehr zur liste hinzugefügt oder entfernt wird.
LinkedList für eine Liste in der immer wieder Objekte hinzugefügt oder entfernt wird

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



Das die CopyOnWriteArrayList extrem langsam ist beim hinzufügen von Elementen

3. Wieso ist CopyOnWriteArrayList scheinbar so langsam?
Da diese nur in einem Thread läuft. Die ArrayList zum Beispiel kann multithreading.

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???)

Die meisten Listen sind nicht thread-sicher, da so weniger Rechenpower. Um eine Liste thread-sicher zu machen kann CopyOnWriteArrayList genutzt werden. Hier wird bei jeder Modifikation eine frische Kopie der Liste angefertigt.

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?)

Das Problem ist, wenn man wären dem durchgehen elemente aus der liste entfernt verschiebt sich der curser der einmal über alles geht und es kommt zu einem Fehler.

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

Die remove() Methode entfernt den Eintrag aus der Garbage Collection. Die get() Methode nicht.

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!

Nein es war eine schlechte Investition da bei diesem Codeabschnitt nur ein Core verwendet wird.