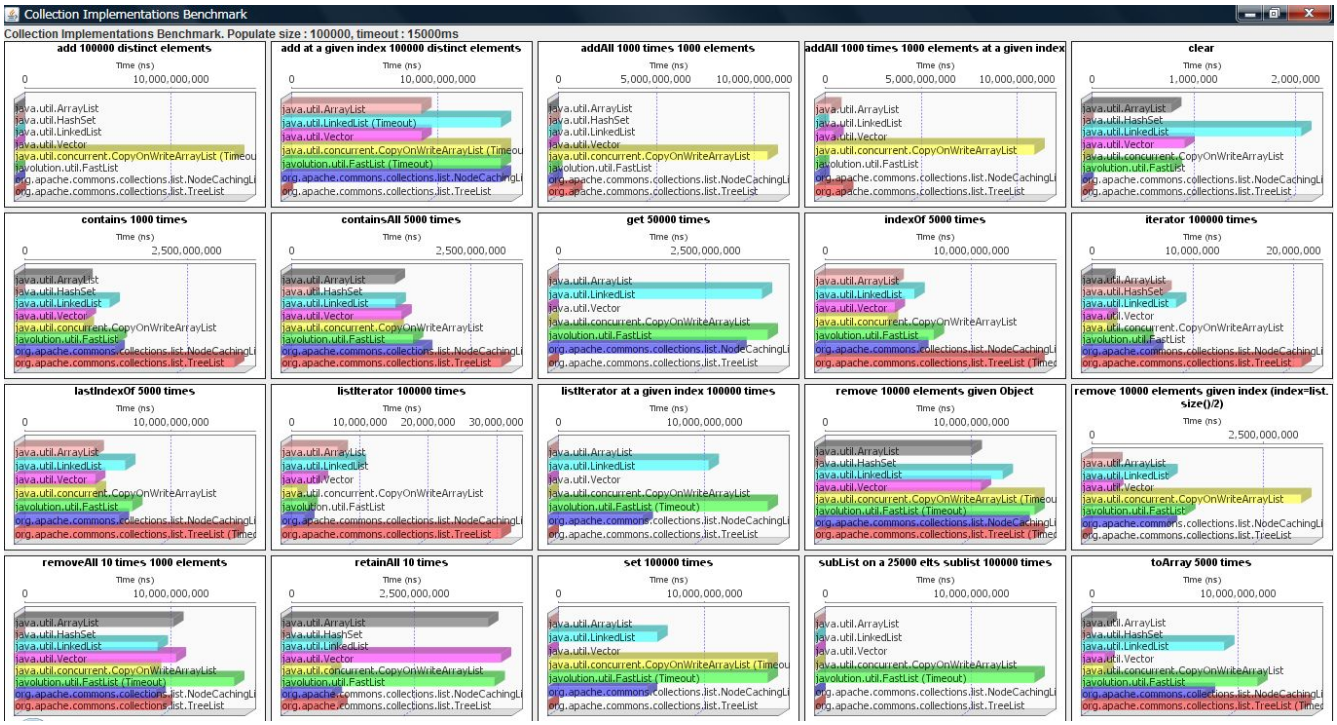
Nachdenkzettel: Collections
Jens Schlegel,Firaz Ilhan, Maximilian Dolbaum

1. ArrayList oder LinkedList – wann nehmen Sie was?

- LinkedList wenn Einträge an bestimmte Stellen hinzugefügt werden soll. Einfaches einfügen und löschen in der Mitte der List
- ArrayLists, wenn noch nicht bekannt wie groß ein Array werden soll. Ist besser, wenn man Elemente an das Ende der Liste speichern möchte

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




Man sollte die passende Collection für das Problem auswählen, da sich die Performance sehr unterscheiden kann.

| | Vorteile |
|---|---|
| Linkedlist | set(), RetainAll(),addAll() |
| ArrayList | add(), contains(), indexOf(), |

3. Wieso ist CopyOnWriteArrayList (COWAL) scheinbar so langsam?

- COWAL makes a copy of the Array when appending a new element.

Weil bei einer Veränderungen jedes Mal das Arrays kopiert werden muss

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???)

Hier sollte man CopyOnWriteArrayList benutzen

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();
   }
}
```

itr.remove(); statt list.remove();

Macht das Verhalten von Java hier Sinn?
ja, da das Iterator Interface auch eine remove() hat

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

Mit einem Cursor kann man vor- und zurück

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

- get() kopiert den Wert
- remove() ist weniger optimal für den GC, da die Referenz entfernt wird und eine neue List ohne das entfernte Element erstellt wird

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….
}
```

Nein, da hier nur ein thread verwendet werden kann

War der Laptop eine gute Investition?

Unterscheidung zwischen

- Sequential Streams(single thread)
- parallele streams (teilt Aufgabe in mehreren threads)

Für die Mutigen: mal nach map/reduce googeln!