Nachdenkzettel Collections

Aufgabe 1

Summary: `ArrayList` with `ArrayDeque` are preferable in *many* more use-cases than `LinkedList`. If you're not sure — just start with `ArrayList`.

In ArrayList accessing an element takes constant time [O(1)] and adding an element takes O(n) time [worst case]. In LinkedList adding an element takes O(n) time and accessing also takes O(n) time. LinkedList uses more memory than ArrayList but it's faster then ArrayList when editing data

---

- Usecase Arraylist: Storing and accessing data
- Usecase LinkedList: Better for manipulating data

Aufgabe 2

https://dzone.com/articles/java-collection-performance

- `CopyOnWriteArray` is significantly slow on data modification
- Also the TreeList requires a lot of time in 9 examples

Aufgabe 3

Because CopyOfWriteArrayList will make a copy of an ArrayList

---

1. CopyOnWriteArrayList(): Creates an empty list.
2. CopyOnWriteArrayList(Collection obj): Creates a list containing the elements of the specified collection, in the order, they are returned by the collection’s iterator.
3. CopyOnWriteArrayList(Object[] obj);: Creates a list holding a copy of the given array.

Aufgabe 4

Use `Collections.synchronizedList()`

Aufgabe 5

If you remove elements from the list storage the iterator points to, there might be a Nullpointer exception

You should use the iterator to remove elements from the list (line 6) (→ `itr.remove()` instead of `list.remove()`)

Aufgabe 6

The remove() method removes an entry from the garbage collector. The get() method doesn't.

Aufgabe 7

No, it wasn't a good Investment because the compiler only uses one core and not 8. (No multi - threads are used)