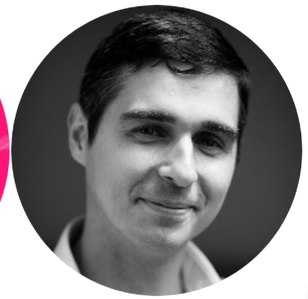
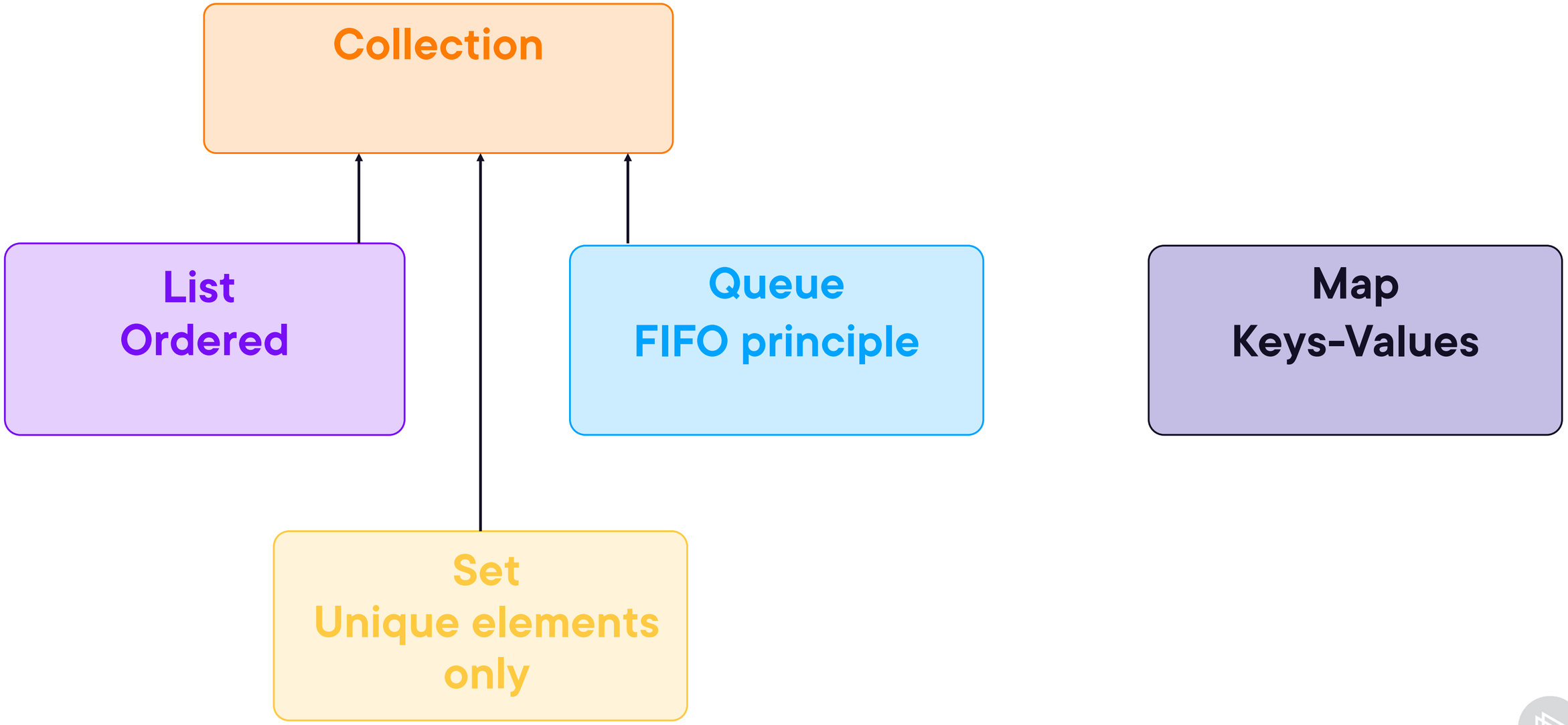
Working with Arrays and Collections



Andrejs Doronins
Software Developer in Test


Collection
List
Ordered
Queue
FIFO principle
Map
Keys-Values
Set
Unique elements
only

```
class ArrayList {
    get();
    add();
    remove();
}
```

Operation Complexity

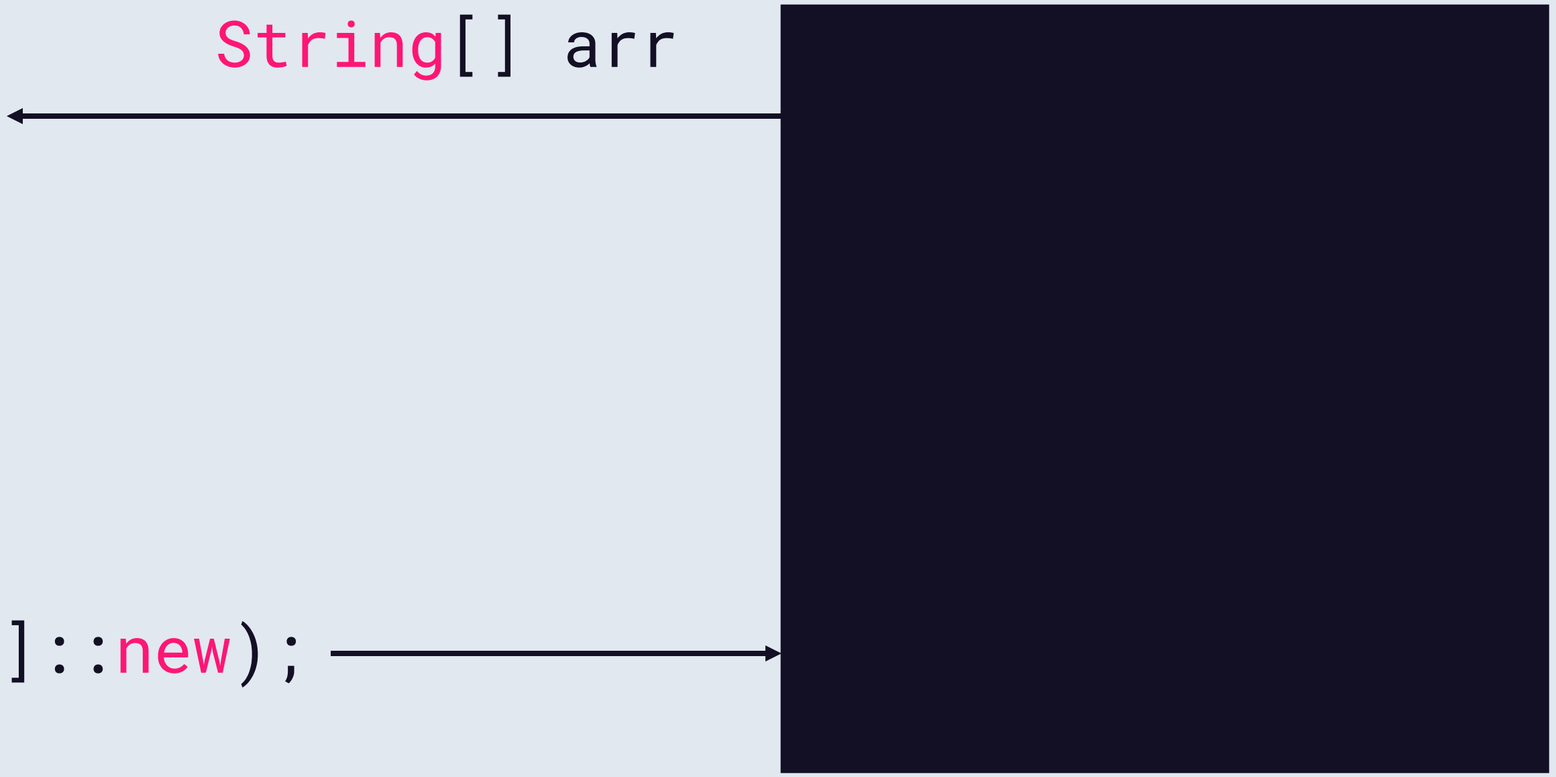
	ArrayList	LinkedList
get(int index)	O(1)	O(n)
add(E element)	O(1)	O(1)
Etc.	Etc.	Etc.

Overview

- Adding
- Removing
- Replacing
- Sorting
- Finding

Converting Array to List

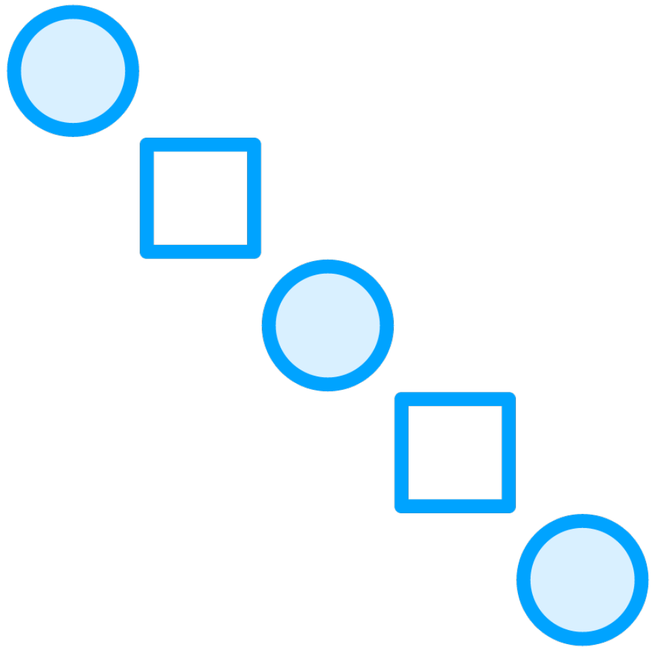
```
List.of(arr)

// add, remove, etc.

list.toArray(String[]::new);
```



	List.of()	Arrays.asList()	new ArrayList()
add() / remove()	No	No	Yes
set()	No	Yes	Yes
Allows nulls	No	Yes	Yes

```java
Integer[] array = {1,2,3};
List<Integer> list = Arrays.asList(array);
array[1] = 5;
System.out.println(list); // [1, 5, 3]
```

```java
Integer[] array = {1,2,3};
List<Integer> list = List.of(array);
array[1] = 5;
System.out.println(list); // [1, 2, 3]
```



Lists (and sets):

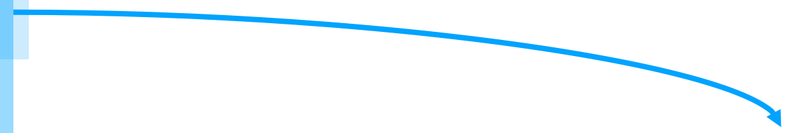
- Adding
- Removing
- Checking if empty
- Apply the Stream API

Lists vs. Maps

Element	“Java”	“C#”	“Python”
Index	0	1	2

Key	Value
1	“Java”
2	“C#”
3	“Python”

Key	Value
1	Java
2	C#
3	C#
4	JavaScript
5	JavaScript



```
frequency(map.values(), value)
```

Key	Value
1	Java
2	C#
3	C#
4	JavaScript
5	JavaScript

1

```
frequency(map.values(), value)
```

Key	Value
1	Java
2	C#
3	C#
4	JavaScript
5	JavaScript

```
frequency(map.values(), value)
```

true

2 < 1

```
int addFive(int input) {
    return input + 5;
}

Function<Integer, Integer> addFive =
                              (input) -> input + 5;
```

```
String concat(int i1, String i2) {
        return String.valueOf(i1).concat(i2);
    }

BiFunction<Integer, String, String> concat =
            (i1, i2) -> String.valueOf(i1).concat(i2);
```

Map: Key,Value

```
BiFunction<Integer, String, String>
```

new value

Map Implementations

	HashMap	TreeMap	LinkedHashMap
Order	No	Natural order	As inserted

1,2,3,4

a,b,c,d

Summary

Working with Arrays and Collections

Functional Interfaces, e.g. BiFunction

Stream API

Collections Libraries

Apache

Guava

Apache

```
Collection<String> result = CollectionUtils.intersection(
                            List.of("a", "b", "c"),
                            List.of("b", "c", "d", "e"));
System.out.println(result); // [b, c]


Collection<String> result2 = CollectionUtils.collate(
                            List.of("a", "b", "c"),
                            List.of("b","c","d","e"));
System.out.println(result2); // [a, b, b, c, c, d, e]
```

Guava

```
List<String> reversed = Lists.reverse(
                                List.of("e", "z", "f"));

System.out.println(reversed); // [f, z, e]
```

Up Next:

Writing Succinct I/O Code