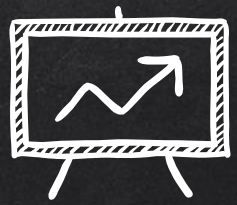


Kotlin Collection API

What Is the Collection API?

- Data manipulation on Sets/Collections of Data
- Implementation of methods such as:
  - Mapping
  - Grouping
  - Filtering
  - Sorting
- Equivalent to Java 8 Streams
- Benefits of Kotlin
  - Simpler Syntax
  - Available on Java 6 + (including Android)

Extract List of Names

```
// Java:
List<String> list = people.stream().map(Person::getName).collect
(Collectors.toList());

// Kotlin:
val list = people.map { it.name }
```

Filter By Gender

```
// Java:
List<Person> males = people.stream()
    .filter( item -> item.getGender() == Person.Gender.MALE )
    .collect(Collectors.toList());

// Kotlin:
val males = people.filter { item.gender == Person.Gender.MALE }
```

Sorting By Multiple Values

```
// Java:
Comparator<Person> byFirstName = (p1, p2) ->
                    p1.getFirstName().compareTo(p2.getFirstName());
Comparator<Person> byLastName = (p1, p2) ->
                    p1.getLastName() .compareTo(p2.getLastName());
List<Person> phoneBook = people.stream()
                    .sorted(byLastName.thenComparing(byFirstName))
                    .collect(Collectors.toList());


// Kotlin:
val phoneBook =people.sortedWith(compareBy({it.lastName}, {it.firstName}))
```

Name of Females in List

```
// Java:
List<String> namesOfFemales = people.stream()
    .filter(p -> p.getGender() == Person.Gender.FEMALE)
    .map(p -> p.getName())
    .collect(Collectors.toList());

// Kotlin:
val namesOfFemales = people.filter { it.gender == Person.Gender.FEMALE
}.map { it.name }
```

Grouping By Gender

```
// Java:
Map<Person.Sex, List<String>> namesByGender =
    people.stream().collect(
      Collectors.groupingBy( Person::getGender,
        Collectors.mapping( Person::getName,  Collectors.toList())));

// Kotlin:
val namesByGender = people.groupBy { it.gender }
                                    .mapValues { it.value.map { it.name } }
```

Partitioning Collections
(Splitting on Boolean)

```
// Java:
Map<Boolean, List<Person>> peopleByGender =
    people.stream()
          .collect(Collectors.partitioningBy(s -> s.getGender() == Person.
Gender.MALE));

// Kotlin:
val peopleByGender = people.partition { it.gender>= Person.Gender.MALE }
```

Find Person With Longest Name

```
// Java:
String longest = people.stream()
    .max(Comparator.comparing(item -> item.name.length()))
    .get();

// Kotlin:
val people = people.maxBy { it.name.length }
```

Kotlin Sequences

- All previous examples generate a new collection instance per operation
- An alternative (especially with large collections) is to use Sequences
- Sequences are lazily evaluated (like Python Generators)
- Typically need to be converted back to Collection (like Java Stream)

Examples of Kotlin Sequences

```
// Find 1st 5 Odd Squares
val squares = sequence(1) {it + 1}.map {it * it}
val oddSquares = squares.filter {it % 2 != 0}
 println(oddSquares.take(5).toList())

// Read Lines from File
val reader:java.io.BufferedReader = ...
val lines = sequence {reader.readLine()}.takeWhile {it != null}
```

... can now process each line without holding whole file in memory

Hands-On Kotlin Collections

Exercise

Apply lessons learnt on these slides to an example project.

Given a list of countries, perform operations that apply sorting, grouping and filtering,

Starting Point

A Project which can be imported into IntelliJ.

The list of Countries will be populated from a JSON File (using Kotson library).

Domain objects & Test Cases are already defined.

To Do

Make the Unit Test cases pass by implementing the missing function bodies.

THANKS!

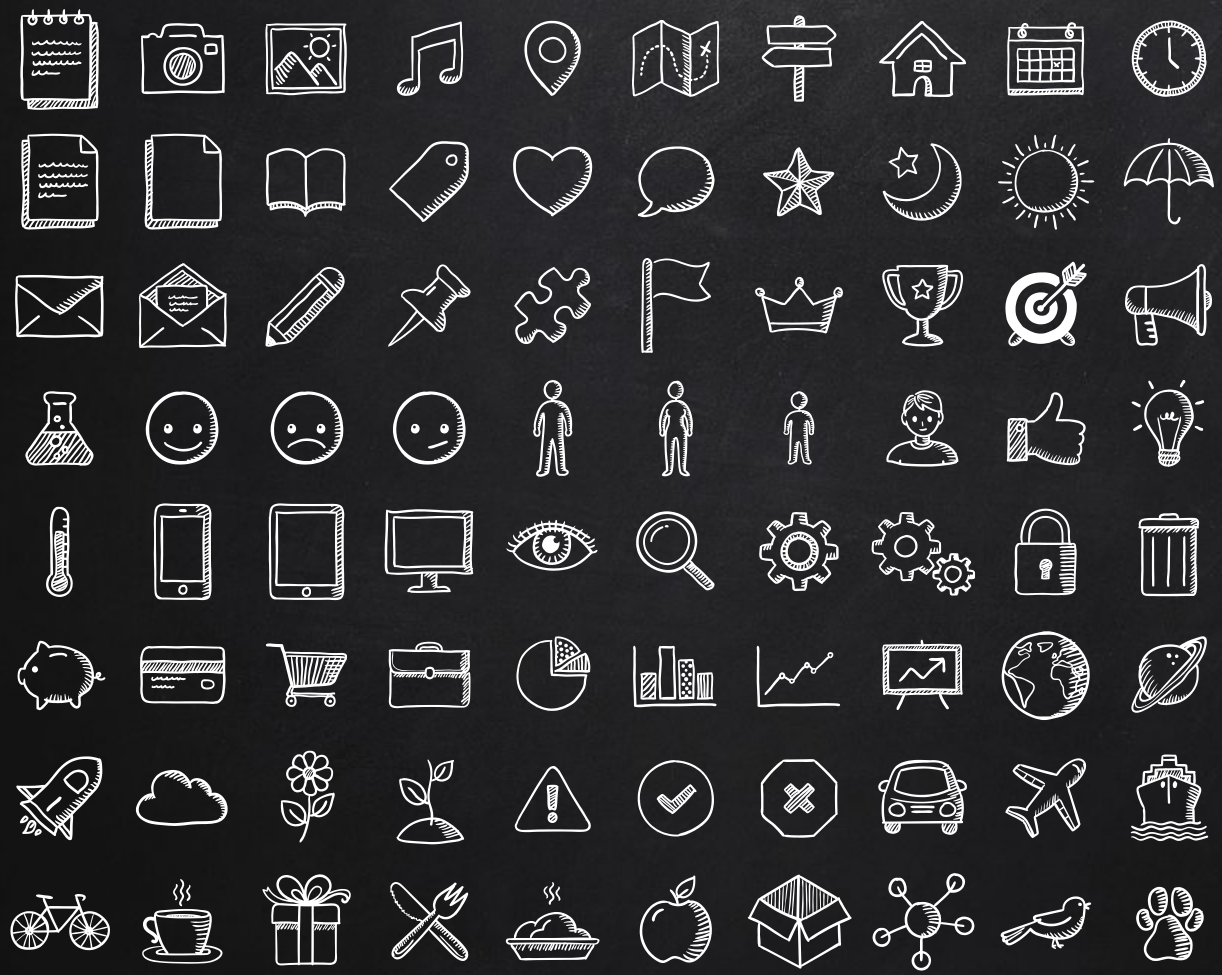
Any questions?

You can find me at
@AndyJBowes
andy.bowes@gmail.com

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash



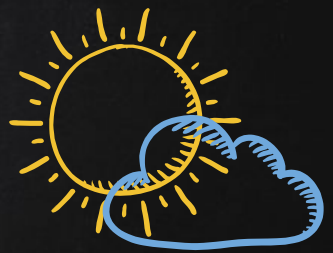
SlidesCarnival icons are editable shapes.

This means that you can:

- Resize them without losing quality.
- Change fill color and opacity.

Isn't that nice? :)

Examples:



Extra graphics