<epam> Excellence in Software Engineering

Guava

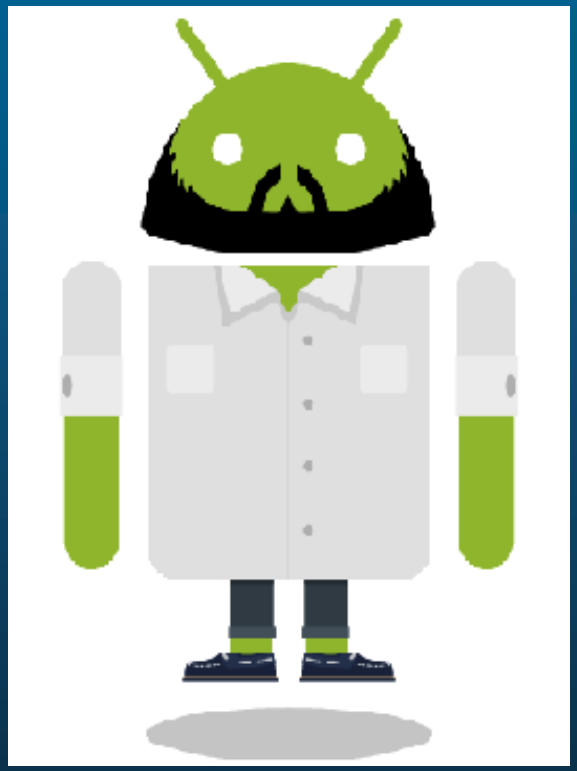
As a tool for every day

Hello!

My name is **Izzet**

I’m@epam.com for 5+ years

In touch with coding, agile, devOps…



Agenda

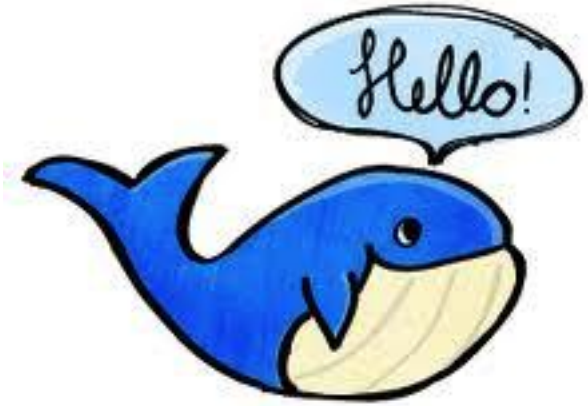
- Introduction
- Collections
- Functional elements
- IO Streams
- Preconditions
- Cache
- String
- Objects

What is Guava stands for?

INTRODUCTION

Introduction

- Guava is a modern core library for Java >=1.5
- Built and used at Google
- Open source



Guava collection enhancements

COLLECTIONS

Collections/Lists

```
List<String> list = new ArrayList<String>();
list.add(“a”)
…
list.add(“e”)
```

```
List<String> list = Lists.newArrayList("a", "b", "c", "d", "e");
```

Collections/Lists

Lists.newArrayListWithExpectedSize(size)

Lists.newCopyOnWriteArrayList()

Lists.newLinkedList()

Lists.partition(list, size)

Lists.reverse(list)

....

Collections/Maps

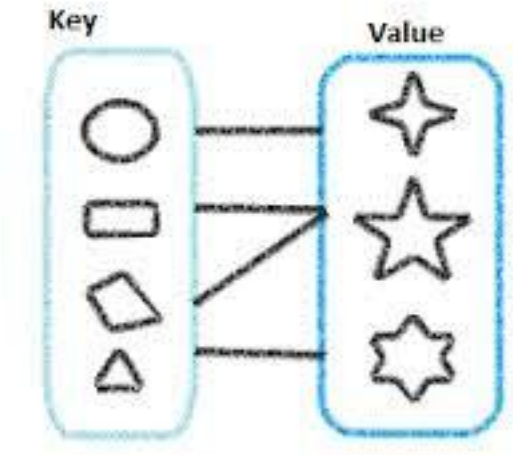
Maps.newHashMap()

Maps.newLinkedHashMap()

Maps.newTreeMap()

Maps.newEnumMap(type)

….



Collections/Maps

Maps.difference(leftMap, rightMap)

Maps.filterEntries(unfilteredMap, entryPredicate)

Maps.filterKeys(unfilteredMap, keyPredicate)

Maps.filterValues(unfilteredMap, valuePredicate)

Collections/Multimap

```
HashMap<String, List<String>> oldSchoolMultimap = …;
ArrayList<String> emptyList = newArrayList();
oldSchoolMultimap.put(key, emptyList);
for (String dataItem : dataList) {
        oldSchoolMultimap.get(key).add(dataItem);
}

ArrayListMultimap<String, String> arrayListMultimap = ArrayListMultimap.create();
for (String dataItem : dataList) {
        arrayListMultimap.put(key, dataItem);
}
assertArrayEquals(oldSchoolMultimap.get(key).toArray(),
arrayListMultimap.get(key).toArray());
```

Collections/Multimaps impls

- HashMultimap //does not store duplicate key-value pairs
- ImmutableListMultimap
- ImmutableMultimap
- LinkedHashMultimap //does not allow duplicate key-value entries and that returns collections whose iterators follow the ordering in which the data was added to the multimap
- LinkedListMultimap
- TreeMultimap

….

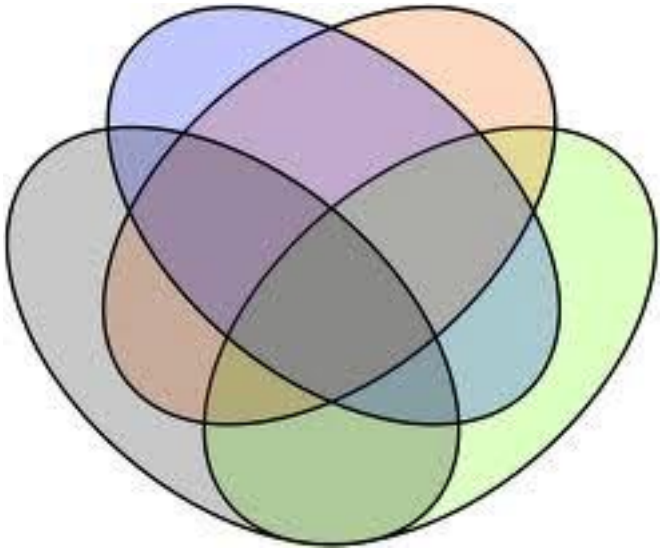
Collections/Sets

Sets.newCopyOnWriteArraySet()

Sets.newLinkedHashSet()

Sets.newTreeSet()

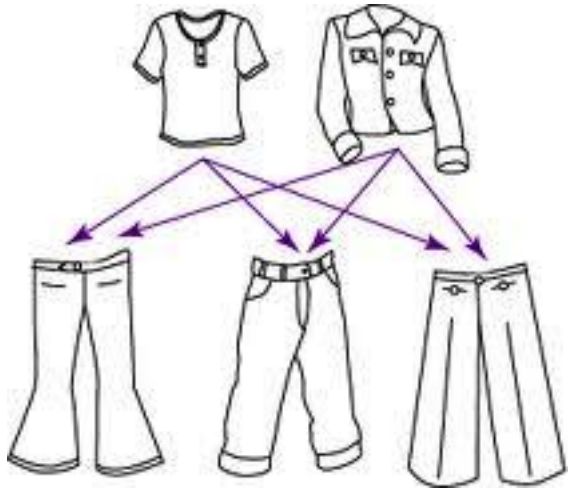
Sets.newSetFromMap(map)



Collections/Sets/Cartesian Product

```
Sets.cartesianProduct(ImmutableList.of(
        ImmutableSet.of(1, 2),
        ImmutableSet.of("A", "B", "C")))
```

```
ImmutableList.of(1, "A")
ImmutableList.of(1, "B")
ImmutableList.of(1, "C")
ImmutableList.of(2, "A")
ImmutableList.of(2, "B")
ImmutableList.of(2, "C")
```



Collections/Sets methods

Sets.difference(set1, set2)

Sets.filter(unfilteredSet, predicate)

Sets.intersection(set1, set2)

Sets.union(set1, set2)

Collections/Multiset

```
Map<String, Integer> wordCounter = newHashMap();
for (String word : words) {
  Integer count = wordCounter.get(word);
  if (count == null) wordCounter.put(word, 1);
  else wordCounter.put(word, count + 1);
}
```

```
Multiset<String> wordsMultiset;
wordsMultiset = HashMultiset.create();
wordsMultiset.addAll(words);
wordsMultiset.count(word)
```

Collections/Multisets impls

- ConcurrentHashMultiset //supports concurrent modifications
- HashMultiset //backed by an HaspMap
- ImmutableMultiset
- LinkedHashMultiset
- TreeMultiset
- EnumMultiset //backed by an EnumMap

….

Collections/Ranges

Range is a portion of a particular domain

Range requires that the upper endpoint may not be less than the lower endpoint

(a..b) Ranges.**open**(C, C)

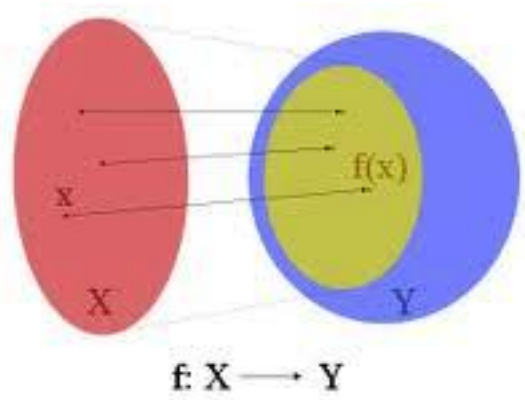
[a..b] Ranges.**closed**(C, C)

[a..b) Ranges.**closedOpen**(C, C)

(a..b] Ranges.**openClosed**(C, C)

(a..+∞) Ranges.**greaterThan**(C)

….



Collections/Ranges operations

```
assertTrue(Ranges.closed(1, 3).contains(2)); // query

assertFalse(Ranges.closed(4, 4).isEmpty());


assertTrue(Ranges.closed(3, 5).isConnected(Ranges.open(5, 10))); //
connection between ranges


Range<Integer> intersection = Ranges.closed(3,
5).intersection(Ranges.open(5, 10)); // intersection

assertEquals(Ranges.openClosed(5, 5), intersection);


….
```

Collections/Even More

Table/Tables

BiMap

Constraints

Queues

RangeSet

RangeMap

Guava functional elements

FUNCTIONAL ELEMENTS

Functional elements/Predicate

Determines a true or false value for a given input. Useful with filters on iterables.

```
interface Predicate<T> {
        boolean apply(T input);
}
```

Functional elements/Predicate

```
List<Integer> inputList = newArrayList(1, 5, 6, 7, 8, 9);
Predicate<Integer> lessThen=new Predicate<Integer>() {
        public boolean apply(final Integer input) {
                return input > 5;
        }
};

assertFalse(Iterables.all(inputList, lessThen));
```

Functional elements/Function

Determines an output value based on an input value. Useful with transformers

```
interface Function<F, T> {
        T apply(F input);
}
```

Functional elements/Function

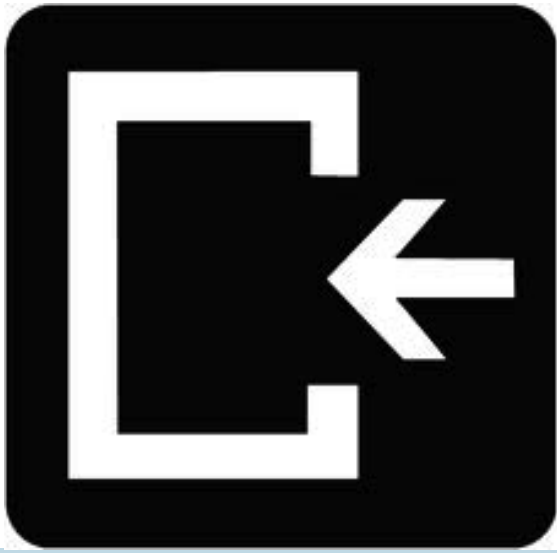
```
Function<Integer, String> toString = new Function<Integer, String>() {
        public String apply(final Integer value) {
                return value.toString();
        }
};

Iterable<String> strings;
strings = Iterables.transform(newArrayList(1, 2, 3, 4), toString);

assertEquals("1", Iterables.getFirst(strings, null));
```

Guava IO enhancements

IO Streams

IO Streams/InputSupplier

```
interface InputSupplier<T> {

  // T is typically an InputStream, a Reader, etc.

   T getInput() throws IOException;

}

// Guava opens/closes streams for you
```



IO Streams/OutputSupplier

```
interface OutputSupplier<T> {

  // T is typically an OutputStream, a Writer, etc.

   T getOutput() throws IOException;

}
```

IO Streams/CharStreams

```
InputSupplier<StringReader> stringReadeSupplier;

stringReadeSupplier =
CharStreams.newReaderSupplier(”a\nb\nc");


List<String> stringLines =
CharStreams.readLines(stringReadeSupplier);


assertEquals(”a", stringLines.get(0));

assertEquals(”b", stringLines.get(1));
```

IO Streams/ByteStreams

```
String hello = "hello";
InputStream is = new ByteArrayInputStream(hello.getBytes());

assertEquals(hello, new String(ByteStreams.toByteArray(is)));
```

IO Streams/Files

```
File file = …;
List<String> lines = null;
try {

  lines = Files.readLines(file, Charsets.UTF_8);

} catch (IOException e) {
  //
}
```

IO Streams/Files

```
BufferedReader in = null;

StringBuilder sb = new StringBuilder();

try {

    in = new BufferedReader(new FileReader(file));

    String str;

    while ((str = in.readLine()) != null) sb.append(str).append('\n');

} catch (IOException e) {

    // do something here

} finally{

    if(in!=null)in.close();

}

sb.toString();
```

String contents = Files.toString(file, Charset.UTF_8)

IO Streams/Files more

- CharStreams.join(InputSupplier...)
- CharStreams.toString(InputSupplier)
- ByteStreams.copy(InputSupplier, OutputSupplier)
- ByteStreams.write(byte[], OutputSupplier)
- Files.copy(File, File) Files.touch(File)
- Files.write(CharSequence, File, Charset)

….

Guava methods preconditions

PRECONDITIONS

Preconditions

```
// IllegalArgumentException

Preconditions.checkArgument(age>0, "Invalid age");


// NPE

Preconditions.checkNotNull(name,  "name must have a value");


// IllegalStateException

Preconditions.checkState(field>0);
```

Guava cache enhancements

CACHE

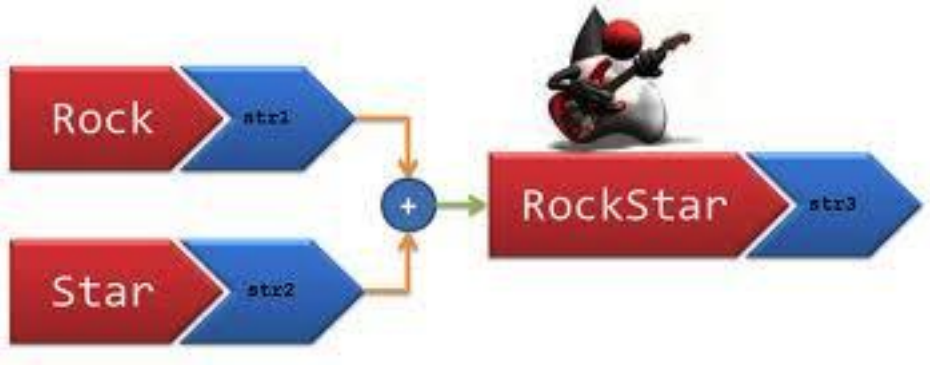
Cache

Supersede features over ConcurrentMap like:

- Different loading capabilities (simple, bulk, atomic)
- Eviction of entries (size, time, reference based; explicit)
- Removal listeners
- Refreshing of entries
- Cleanup
- Statistics
- View asMap

Cache/Eviction

- Size-based – evicts less used entries if size reached, ability to set weights.
- Time-based – evicts entries after some time of access or write
- Reference-based – entries are garbage-collected by weak keys or values, soft values
- Explicit eviction
- Removal listeners



Guava String enhancements

STRING

String/Splitter & Joiner

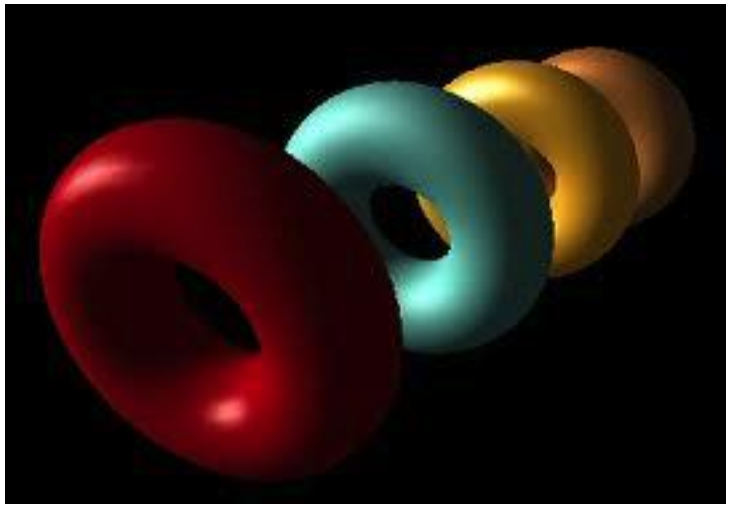
```
String text = "a,b,,c, d, e ";          Iterable<String> split =
Splitter.on(",”)
.omitEmptyStrings()
.trimResults().split(text);

String joined = Joiner.on(".").join(split);
assertEquals("a.b.c.d.e", joined);
```

String/Strings

```
Strings.commonPrefix(“ab”, “ac”) // a

Strings.commonSuffix(“aa”, “ba”) // a

Strings.isNullOrEmpty(“”) // true

Strings.repeat(“a”, 2) // aa
```

String/CharMatcher

```
CharMatcher.WHITESPACE

CharMatcher.ASCII

CharMatcher.is('x’)

CharMatcher.isNot('_’)

CharMatcher.oneOf("aeiou").negate()

CharMatcher.inRange('a’,'z').or(inRange('A', 'Z'))

….

String seriesId = CharMatcher.DIGIT.or(CharMatcher.is('-'))

.retainFrom(“11-1AB”); // 11-1
```



Guava Objects enhancements

OBJECTS

Objects/equals

```
public boolean equals(Object obj) {
 if (obj instanceof Item) {
   Item that= (Item) obj;
   return Objects.equal(this.name, that.name);
 }
 return false;
}
```

Objects/hashCode

```
public int hashCode() {
        return Objects.hashCode(name, email);
}
```

Objects/toString

```
public String toString() {
        return Objects.toStringHelper(this)
                .add(”name", name)
                .add(”email", email)
        .toString();
}
```

Conclusion

Pros

- Well designed and consistent
- Is in active development
- Functional support when Java 8 is not in prod
- Less code less errors

Cons

- Is not a single tool for everything
- Need to introduce new dependency
- Need to spend some time to learn
- If your environment requires Java <= 1.4

Questions

Thank you!

Guava as a tool for every day

Prepared by: izzet_mustafayev@epam.com
Blog: http://webdizz.name/
Twitter: @webdizz

Get more knowledge:
http://goo.gl/g3b57