Refactoring to Eclipse Collections
Making Your Java Streams Leaner, Meaner, and Cleaner

Vladimir Zakharov, Managing Director, Goldman Sachs
Introduction

• What is Eclipse Collections?
  • Feature rich, memory efficient Java Collections framework

• History
  • Eclipse Collections started off as an internal collections framework named Caramel at Goldman Sachs in 2004
  • In 2012, it was open sourced to GitHub as a project called GS Collections
  • GS Collections was migrated to the Eclipse Foundation, re-branded as Eclipse Collections in 2015

• Learn Eclipse Collections with Kata
• Eclipse Collections is open for contributions!
Why Refactor to EC?

- No “Bun” methods
- Concise Code
- Comprehensive Support for Primitive Types
- Rich[er] APIs
- Testability/Debuggability
- Memory Savings
- It’s Easy!
- Better Performance [citation needed]
- Drop-in replacement for JDK types
- Intuitive humane APIs
- Lazy or Eager
- TestUtils
- Efficient Maps and Sets
- It's Easy!
Any Types You Need

MutableIntList
LongStack
SortedBagMultimap
ImmutableObjectLongMap
MutableSortedSetMultimap
## Instantiate Them Using Factories

<table>
<thead>
<tr>
<th>Primitive Type</th>
<th>Container Type</th>
<th>Mutability</th>
<th>Multimap</th>
<th>Initialized</th>
<th>Lazy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Bags</td>
<td>.mutable</td>
<td>.bag</td>
<td>.empty()</td>
<td>.asLazy()</td>
</tr>
<tr>
<td>Byte</td>
<td>BiMaps</td>
<td>.immutable</td>
<td>.list</td>
<td>.of(one)</td>
<td></td>
</tr>
<tr>
<td>Char</td>
<td>Lists</td>
<td>.fixedSize</td>
<td>.set</td>
<td>.with(one)</td>
<td></td>
</tr>
<tr>
<td>Double</td>
<td>Maps</td>
<td></td>
<td>.sortedSet</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Float</td>
<td>Multimaps</td>
<td></td>
<td></td>
<td>.of(one,...,ten)</td>
<td></td>
</tr>
<tr>
<td>Int</td>
<td>Sets</td>
<td></td>
<td></td>
<td>.with(one,...,ten)</td>
<td></td>
</tr>
<tr>
<td>Long</td>
<td>SortedBags</td>
<td></td>
<td></td>
<td>.of(... elements)</td>
<td></td>
</tr>
<tr>
<td>Object</td>
<td>SortedMaps</td>
<td></td>
<td></td>
<td>.with(... elements)</td>
<td></td>
</tr>
<tr>
<td>Short</td>
<td>SortedSets</td>
<td></td>
<td></td>
<td>.ofAll(Iterable)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stacks</td>
<td></td>
<td></td>
<td>.withAll(Iterable)</td>
<td></td>
</tr>
</tbody>
</table>

**ImmutableLongStack**

```java
LongStacks.immutable.with(1, 2, 3).asLazy()
```

**LazyLongIterable**

```java
LazyLongIterable
```
Let’s Do It!
Memory Usage (Overhead in KB by Count)
JMH Benchmark Results

- `getMeatAndNonMeatEatersEc`
- `getMeatAndNonMeatEatersJdk`
- `mostPopularFoodItemEc`
- `mostPopularFoodItemJdk`
- `printNumberOfFavoriteFoodItemsToAnimalsEc`
- `printNumberOfFavoriteFoodItemsToAnimalsJdk`
- `uniqueFoodsEcWithTargetCollection`
- `uniqueFoodsEcWithoutTargetCollection`
- `uniqueFoodsJdk`
Thank you! Questions?
GREAT INDIAN DEVELOPER SUMMIT 2019
Conference: April 23-26, Bangalore

Register early and get the best discounts!

www.developersummit.com  @greatindiandev  bit.ly/gidslinkedin