How Goldman Sachs works with Java

Deanna Lin 林文

we BUILD

Goldman Sachs

Agenda

1 **Engineering at Goldman Sachs**

2 **Using Java and OpenJDK in Goldman Sachs**

3 **Eclipse Collections**

- **What's Eclipse Collections?**
- **Benefit of Eclipse Collections compared to Streams and JCF**
- **Learning Eclipse Collections by solving problems**

Goldman Sachs

Engineering at Goldman Sachs

Goldman Sachs

The Goldman Sachs Group, Inc. is a leading global investment banking, securities and investment management firm that provides a wide range of financial services to a substantial and diversified client base that includes corporations, financial institutions, governments and individuals. Founded in 1869, the firm is headquartered in New York and maintains offices in all major financial centers around the world.

Goldman Sachs Technology Division

We build solutions to some of the most complex problems, creating transformational technologies and developing systems that drive business and financial markets worldwide.

Engineering at Goldman Sachs

Leading latest open technology (global initiatives)

Open Compute Project

- A project to build effective computation infrastructure at the lowest possible cost, resulting with optimized data center operation.

Open Container Initiative

- A project to define format and runtime of container technology. More than 30 organizations including Docker are participating in the project.

Java Community Process – Executive Committee

- JCP Executive Committee is responsible of important specification process for Java platform. Goldman Sachs was elected to the committee in 2011 for the first time and continue to be since then.

Engineering at Goldman Sachs

Goldman Sachs

Using Java in Goldman Sachs

Goldman Sachs

History of Goldman Sachs and Java

1998	• Began internal Java evaluation
2000	• Java became one of the strategic platforms
2000-	• Continuously working closely with Sun/Oracle to improve Java
2004	• Ready for Lambdas in GS Collections Framework predecessor
2012	• Elected into JCP Executive Committee • GS Collections Framework Open-sourced
2013	• Signed OCA • Started contributing to OpenJDK
2015	• Migrated GS Collections to Eclipse Foundation

Goldman Sachs

Java in GS

- JCP Executive Committee member
- Part of the Expert Groups for the following JSRs:
  - JSR 107: JCACHE – Java Temporary Caching API
  - JSR 335: Lambda Expressions for the Java Programming Language
  - JSR 351: Java Identity API

- Over 3000 Java Developers
- Maintain >130 million lines of Java code
- On average, ~125,000 Java processes running at any given hour every day

Goldman Sachs

Cutting Edge Java in GS

We do push the boundary of Java technology

- Very large heap
  - Some applications use as much as 190G heap
  - https://bugs.openjdk.java.net/browse/JDK-6806226
  - Integer overflow issues with GrowableArray caused issue with Mark Stack

- JVM stressed under high volume conditions or unusual usage pattern
  - Code cache capacity

Open Source internally developed framework

- Eclipse Collections: Collection framework that is richer and more fluent than Stream API

Goldman Sachs

Why We Use OpenJDK

- Source access is important!
  - Timely resolution for production issues
  - Identify potential area for improvement
  - Good for security
- Transparency
  - How is the code being developed?
  - Bug database
- Contribute patches for bugs and enhancements that benefit us and the wider Java community

Goldman Sachs

Typical Usage of OpenJDK

- **Troubleshooting**
  - Resolve production issues
  - Useful during development
- **Research new features**
  - Identify optimization opportunities in the JDK
  - How they can be applied and beneficial to applications

Goldman Sachs

Troubleshooting

Having access to JDK source code is important

- Timely Diagnostics
  - Crash dump analysis
  - Debug builds
- Identify workarounds and potential fixes
  - Create patched builds for developers to verify fixes
    - Including patches not yet available to GA builds

Goldman Sachs

Research

- Understanding new features in hotspot
  - How does it work?
  - Does it benefit our applications?
  - If it does, how could applications exploit it?

- Identify performance bottleneck
  - Potential workaround and features to improve application performance

Case Study – Enhancement for OpenJDK

Improve Array.sort() logic (DualPivotQuicksort) for primitive collections

How

Changed the algorithm to

- reduce number of “run” (a “run” is how long you go through the array with it being in order)
- Stop sorting in the case of descending numbers

Test case

- Additional functional unit tests
- Performance benchmark using JMH

Patch contribution

- Discussed patch at http://openjdk.5641.n7.nabble.com/Patch-to-improve-primitives-Array-sort-td224296.html
- Patch was accepted in JDK 9 development branch
- In JDK-8080945, the enhancement is included in June 2015

Goldman Sachs

Eclipse Collections

Open for contributions!

https://github.com/eclipse/eclipse-collections/blob/master/CONTRIBUTING.md

Goldman Sachs

What is Eclipse Collections?

GS Collections

- Open source Java collections framework developed in Goldman Sachs
- Inspired by Smalltalk Collections Framework
- In development since **2004**
- Hosted on GitHub w/ Apache 2.0 License in **January 2012**

Eclipse Collections 7.0

- Project @ Eclipse Foundation https://www.eclipse.org/collections/
- Feature set is the same as GS Collections 7.0
- Public released on Jan 20th!
- Packages renamed **com.gs -> org.eclipse**

eclipse FOUNDATION MEMBER

Goldman Sachs

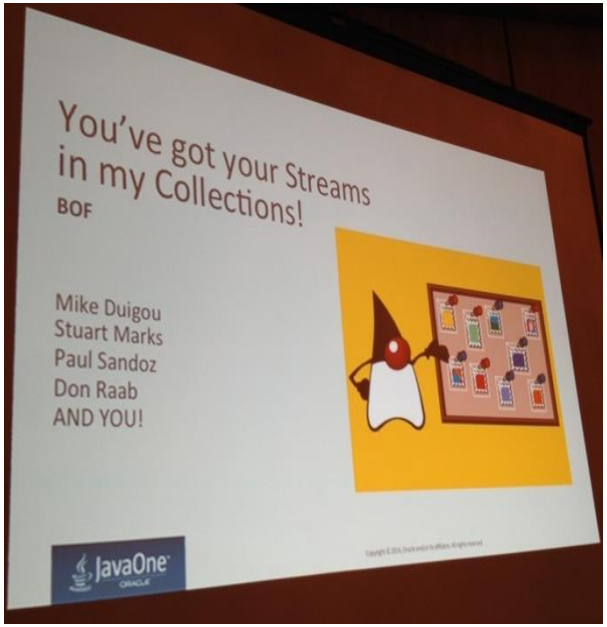
JavaOne2015 – Eclipse Collections by Example

Eclipse Collections: Rich set of features

Java 8 Stream API
- Functional APIs
- Lazy only
- Single use
- Serial & Parallel
- Primitive streams (3 types)
- Extensible Collectors

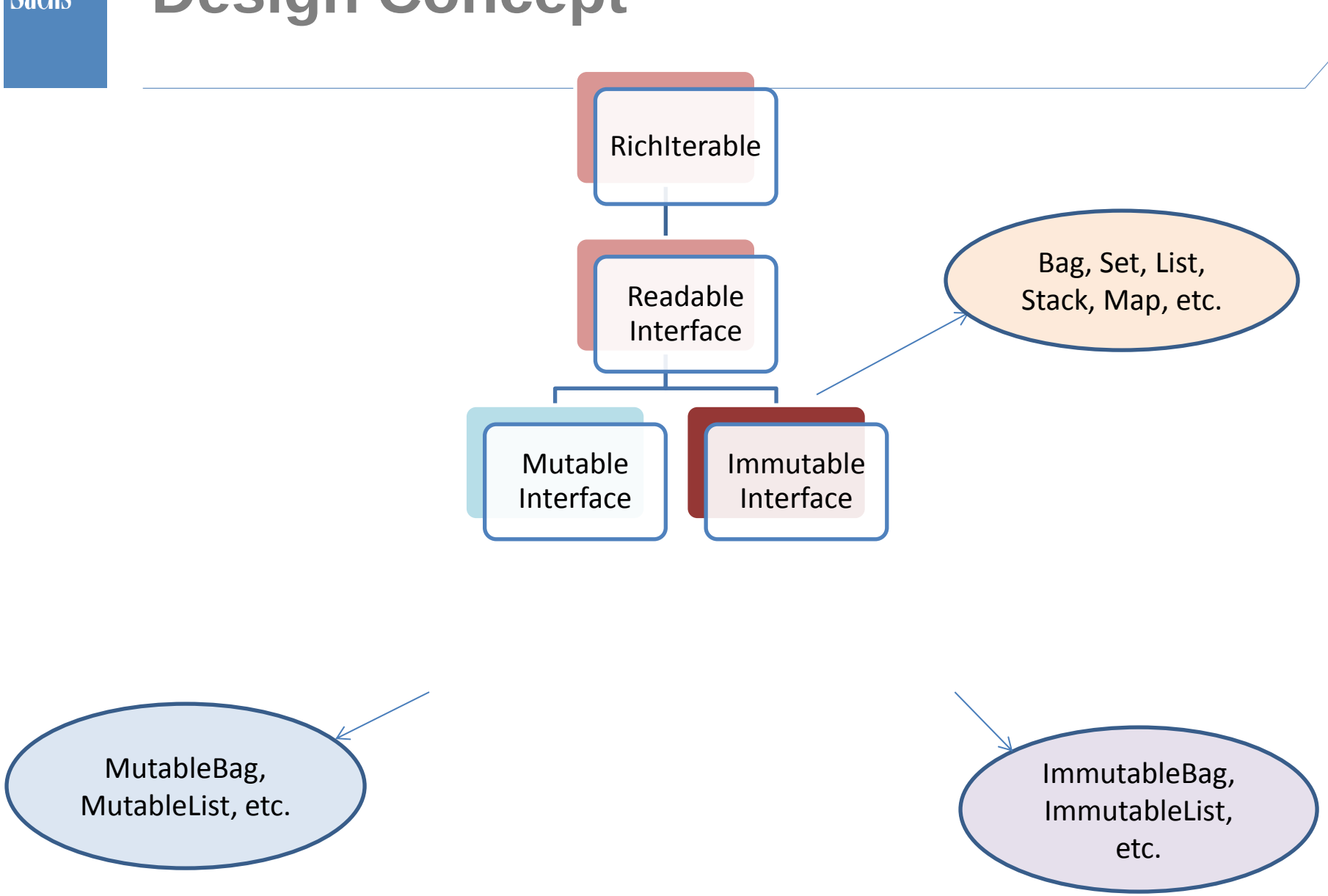
Eclipse Collections 7.0

- Eager & Lazy, Serial & Parallel
- Memory efficient containers
- Primitive containers (all 8)
- Immutable containers
- More container types
- More iteration patterns
- "With" method patterns
- "target" method patterns
- Covariant return types
- Java 5+ compatible



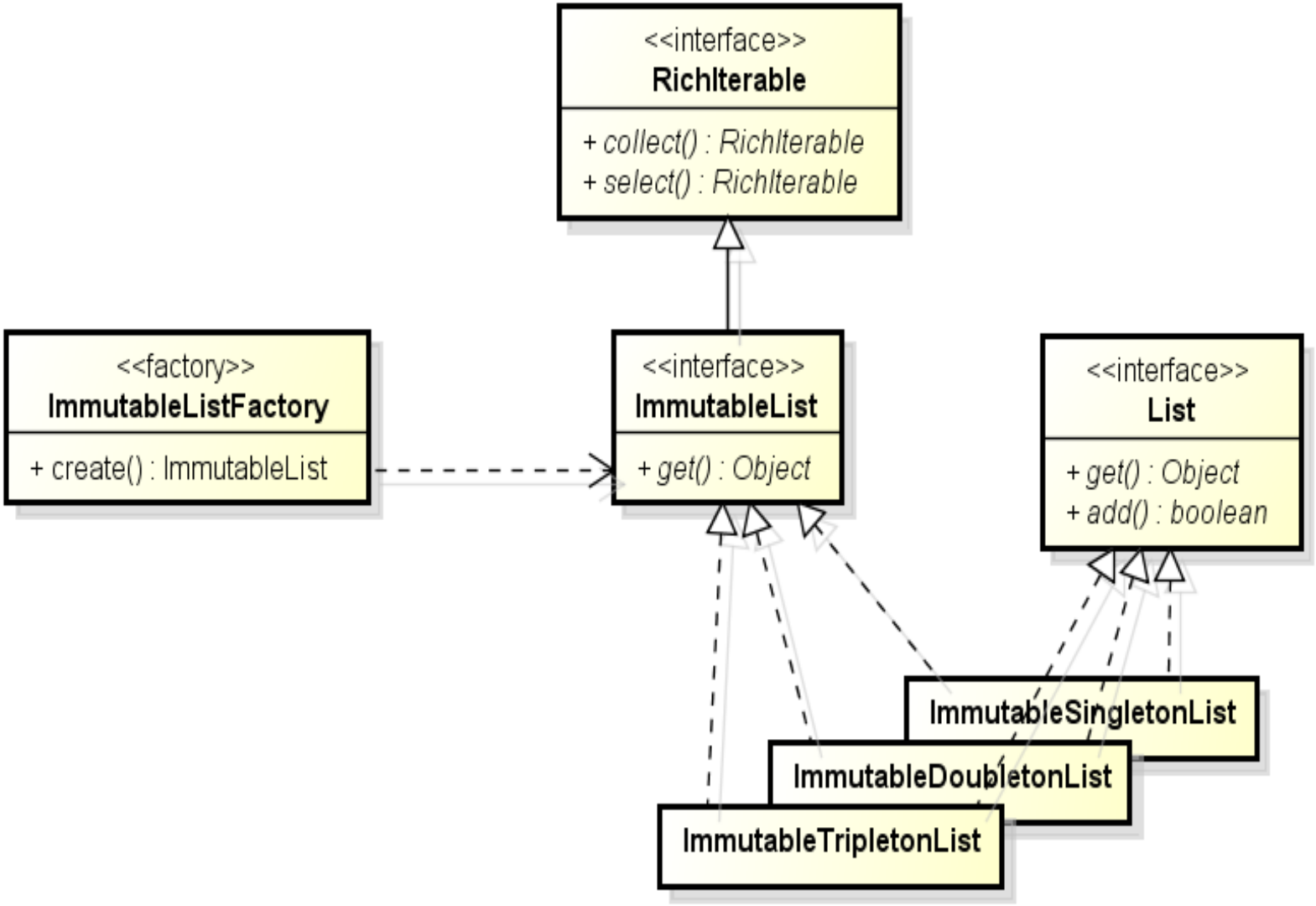
Goldman Sachs

Design Concept



Goldman Sachs

Inheritance diagram - Immutability



Goldman Sachs

Collection Framework Comparisons

Features	Eclipse Collections	Java 8	Guava	Trove	Scala
Rich API	✔	✔	✔		✔
Interfaces	Readable, Mutable, Immutable, FixedSize, Lazy	Mutable, Stream	Mutable, Fluent	Mutable	Readable, Mutable, Immutable, Lazy
Optimized Set & Map	✔ (+Bag)			✔	
Immutable Collections	✔		✔		✔
Primitive Collections	✔ (+Bag, +Immutable)			✔	
Multimaps	✔ (+Bag, +SortedBag)		✔ (+Linked)		(Multimap trait)
Bags (Multisets)	✔		✔		
BiMaps	✔		✔		
Iteration Styles	Eager/Lazy, Serial/Parallel	Lazy, Serial/Parallel	Lazy, Serial	Eager, Serial	Eager/Lazy, Serial/Parallel (Lazy Only)

Goldman Sachs

Benefit of Eclipse Collections compared to Streams and JCF

Goldman Sachs

Write Concise Code

Stream API

```
boolean result = this.people.stream().anyMatch(person -> person.hasPet(PetType.CAT));

long result = this.people.stream().filter(person -> person.hasPet(PetType.CAT)).count();

List<Person> peopleWithCats = this.people.stream().filter(person -> person.hasPet(PetType.CAT))
                                    .collect(Collectors.toList());
```

Eclipse Collections

```
boolean result = this.people.anySatisfy(person -> person.hasPet(PetType.CAT));

int result = this.people.count(person -> person.hasPet(PetType.CAT));

MutableList<Person> peopleWithCats = this.people.select(person -> person.hasPet(PetType.CAT));
```

Goldman Sachs

Leverage method references

Stream API

```
boolean result = this.people.stream().anyMatch(person -> person.hasPet(PetType.CAT));

long result = this.people.stream().filter(person -> person.hasPet(PetType.CAT)).count();

List<Person> peopleWithCats = this.people.stream().filter(person -> person.hasPet(PetType.CAT))
                                    .collect(Collectors.toList());
```

Eclipse Collections

```
boolean result = this.people.anySatisfyWith(Person::hasPet, PetType.CAT);

int result = this.people.countWith(Person::hasPet, PetType.CAT);

MutableList<Person> peopleWithCats =this.people.selectWith(Person::hasPet, PetType.CAT);
```

Goldman Sachs

Code with Better Readability

Stream API

```
List<Person> peopleWithCats = this.people.stream().filter(person -> person.hasPet(PetType.CAT))
                                  .collect(Collectors.toList());

List<Person> peopleWithoutCats = this.people.stream().filter(person -> !person.hasPet(PetType.CAT))
                                  .collect(Collectors.toList());

Map<Boolean, List<Person>> peopleWithcatsAndNoCats = this.people.stream().collect(
                                    Collectors.partitioningBy(person -> person.hasPet(PetType.CAT)));
peopleWithcatsAndNoCats.get(true);
peopleWithcatsAndNoCats.get(false);
```

Eclipse Collections

```
MutableList<Person> peopleWithCats = this.people.select(person -> person.hasPet(PetType.CAT));

MutableList<Person> peopleWithoutCats = this.people.reject(person -> person.hasPet(PetType.CAT));

PartitionMutableList<Person> peopleWithcatsAndNoCats = this.people.partition(
                                  person -> person.hasPet(PetType.CAT));
peopleWithcatsAndNoCats.getSelected();
peopleWithcatsAndNoCats.getRejected();
```

Goldman Sachs

Reuse Code

Stream API

```
Stream<Integer> stream = Lists.mutable.with(1, 2, 3).stream();
Assert.assertEquals(1, stream.mapToInt(i -> i).min().getAsInt());
Assert.assertEquals(3, stream.mapToInt(i -> i).max().getAsInt());
```

// throws

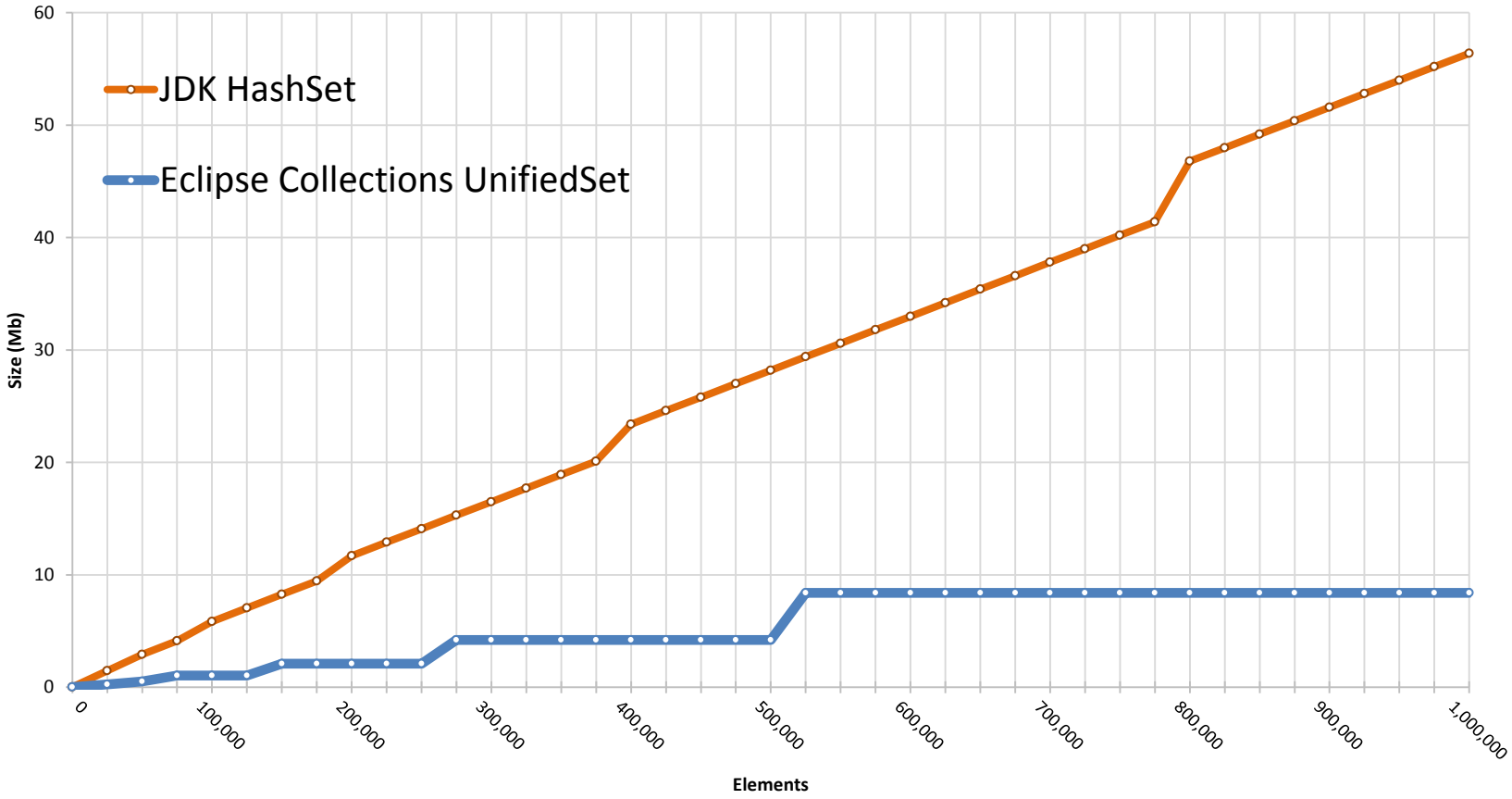
```
java.lang.IllegalStateException: stream has already been operated upon or closed
            at java.util.stream.AbstractPipeline.<init>(AbstractPipeline.java:203)
            at java.util.stream.IntPipeline.<init>(IntPipeline.java:91)
            at java.util.stream.IntPipeline$StatelessOp.<init>(IntPipeline.java:592)
            at java.util.stream.ReferencePipeline$4.<init>(ReferencePipeline.java:204)
            at java.util.stream.ReferencePipeline.mapToInt(ReferencePipeline.java:203)
```

Eclipse Collections

```
LazyIterable<Integer> lazy = Lists.mutable.with(1, 2, 3).asLazy();
Assert.assertEquals(1, lazy.collectInt(i -> i).min());
Assert.assertEquals(3, lazy.collectInt(i -> i).max());
```

Goldman Sachs

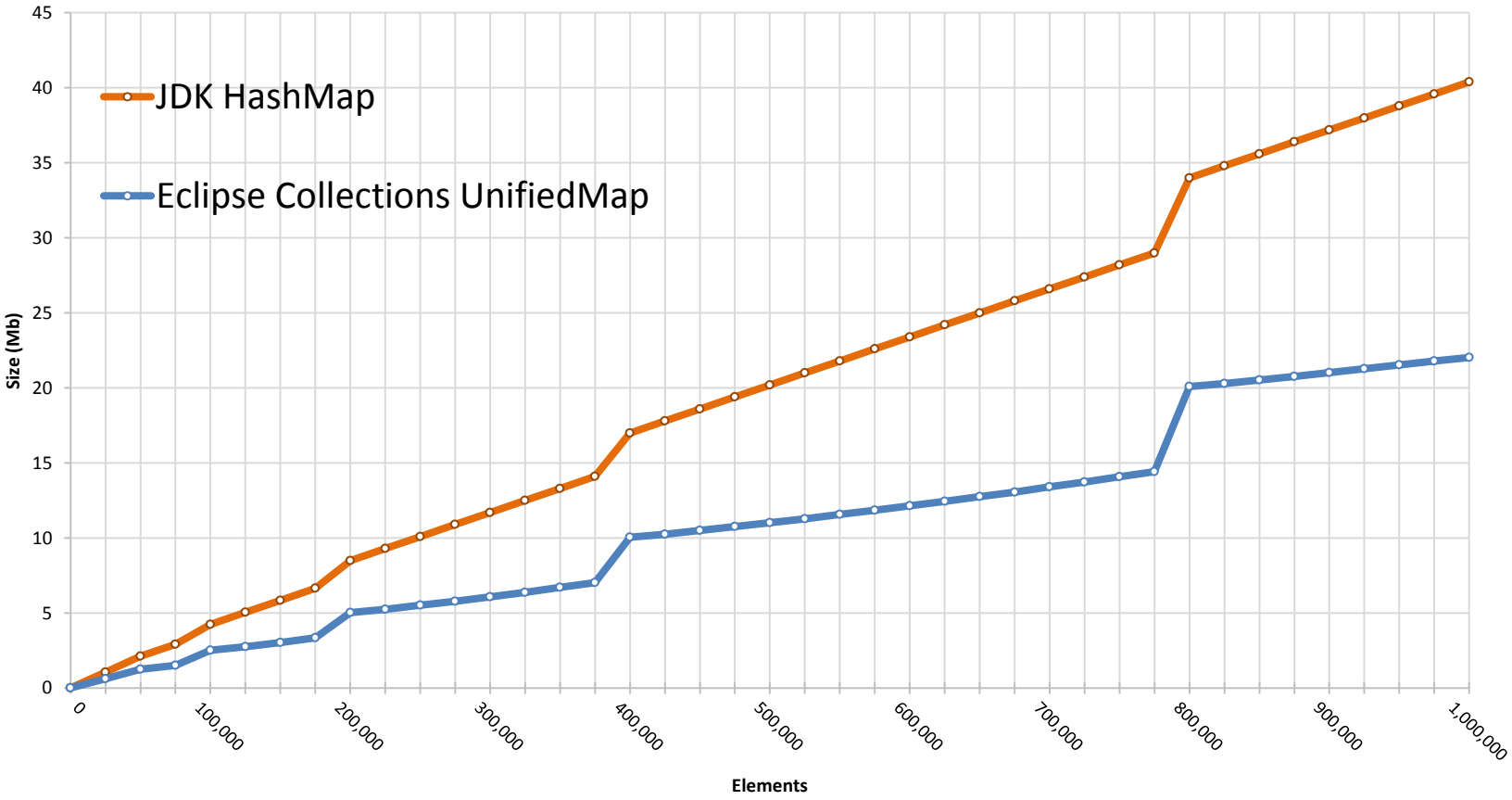
Memory Efficient (HashSet vs UnifiedSet)



Data collected by https://github.com/goldmansachs/gs-collections/tree/master/jmh-tests

Goldman Sachs

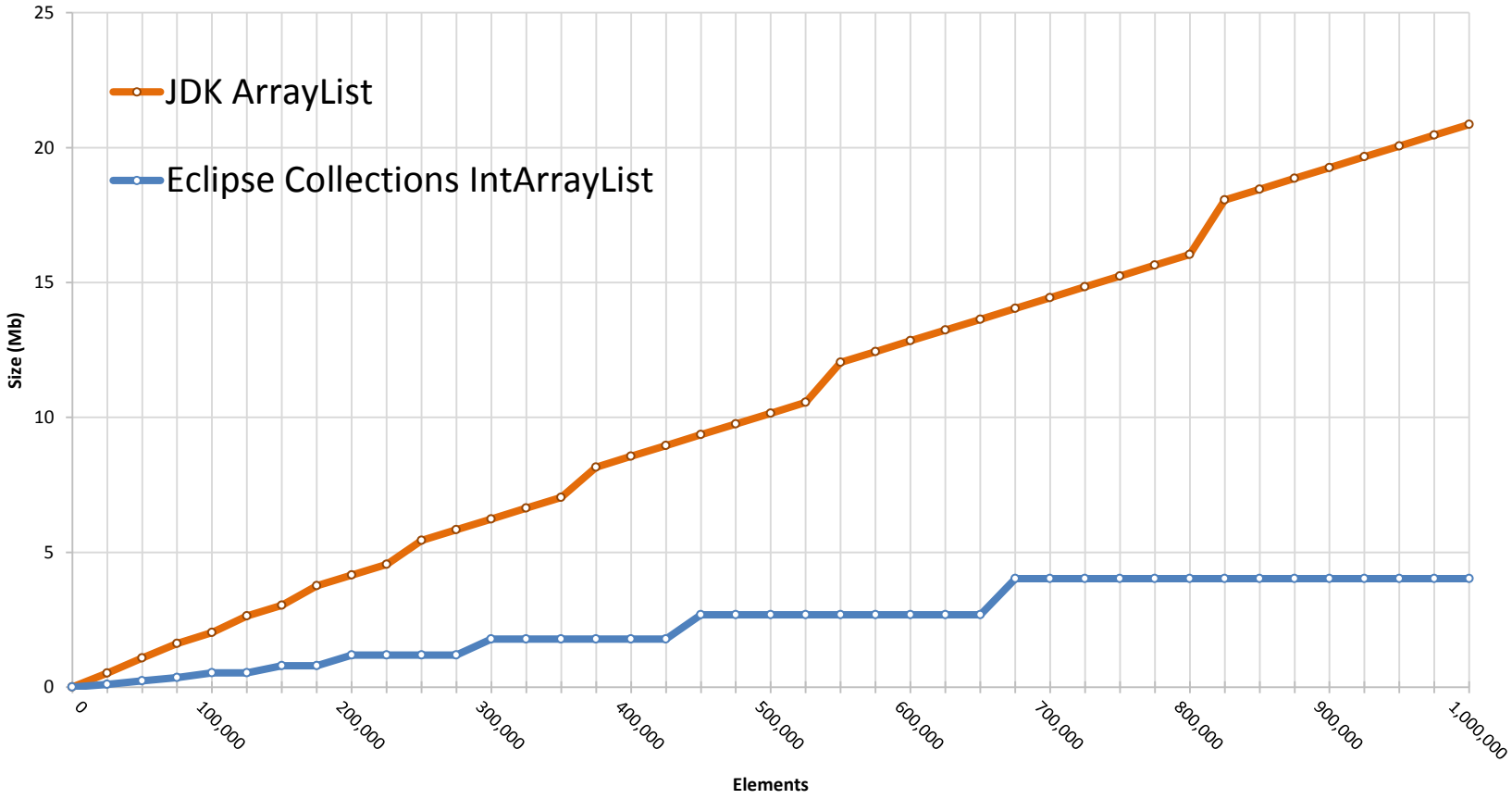
Memory Efficient (HashMap vs UnifiedMap)



Data collected by https://github.com/goldmansachs/gs-collections/tree/master/jmh-tests

Goldman Sachs

Memory Efficient (primitive collections)



Data collected by https://github.com/goldmansachs/gs-collections/tree/master/jmh-tests

Goldman Sachs

Concise Exception Handling

Java Collections

```
Appendable appendable = new StringBuilder();
List<String> jdkList = Arrays.asList("one");
   // jdkList.forEach(appendable::append);
   // Compile Error: incompatible thrown types java.io.IOException in method reference
jdkList.forEach(each -> {
    try
    {
        appendable.append(each);
    }
    catch (IOException e)
    {
        throw new RuntimeException(e);
    }
});
Assert.assertEquals("one", appendable.toString());
```

Eclipse Collections

```
ImmutableList<String> eclipseList = Lists.immutable.with("two");
eclipseList.each(Procedures.throwing(appendable::append));
Assert.assertEquals("onetwo", appendable.toString());
```

Goldman Sachs

Appendix

Goldman Sachs

OpenJDK Enhanced Arrays.sort() JMH Results

Test Sample

An array of int with size 10,000,000, 32 pair flips, a run of zeroes in the middle, and 32 pair flips at the end

Benchmark	(listType)	Mode	Cnt	Score	Error	Units
Sort(JDK8)	pairFlipZeroPairFlip	thrpt	10	4.886	± 0.031	ops/s
Sort(OpenJDK9)	pairFlipZeroPairFlip	thrpt	10	14.793	± 0.217	ops/s

J
tl
y

Name	Description
Score	The mean for the throughput benchmarking mode
Error	Estimate the "noisyness" of the measurements taken by the microbenchmark
Thrpt	Mode.Throughput. Calculate number of operations in a time unit.
Ops/s	Operations per second

Goldman Sachs

Eclipse Collections: Patterns

Eager iteration pattern

```
Interval list = Interval.fromTo(0, 100);

list.detect(each -> each > 50);

list.select(each -> each > 50);

list.reject(each -> each > 50);

list.anySatisfy(each -> each > 50);

list.allSatisfy(each -> each > 50);

list.collect(Object::toString);

list.injectInto(3, Integer::sum);
```

Lazy iteration pattern

```
list.asLazy()
    .select(each -> each > 95)
    .collect(Object::toString)
    .makeString("[", ",", "]");


=> [96,97,98,99,100]
```

Goldman Sachs

Parallel-lazy Performance:
Java 8 vs Scala vs Eclipse Collections

http://www.infoq.com/presentations/java-streams-scala-parallel-collections

The session page got the top page view after QCon New York

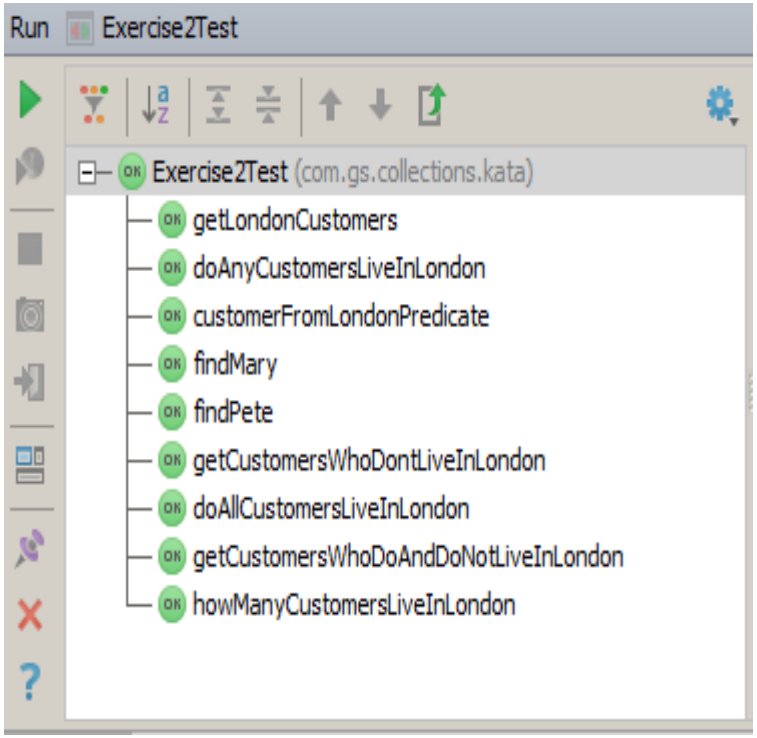
Here are the top 10 presentations from QCon New York 2014, based on page views:

- Parallel-lazy Performance: Java 8 vs Scala vs GS Collections - Craig Motlin
- Evolving Java - Brian Goetz
- Caml Trading - Experiences with OCaml on Wall Street - Yaron Minsky
- End to End Reactive Programming at Netflix - Jafar Husain, Matthew Podwysocki
- Spring 4 on Java 8 - Juergen Hoeller
- 10 Reasons Why Developers Hate Your API - John Musser
- What's the Best Way to Improve Software Architectures? - Eric Evans,Michael Feathers,Duncan DeVore,Leo Gorodinski
- How Facebook Scales Big Data Systems - Jeff Johnson
- A Day in the Life of a Functional Data Scientist - Richard Minerich
- Whither Web programming - Gilad Bracha

Training - Eclipse Collections Kata

https://github.com/eclipse/eclipse-collections-kata

- A training material by passing failed unit test one by one in TDD way
- It is also used in Goldman Sachs trainings
- Useful to learn the usage patterns of Eclipse Collections



Goldman Sachs

Scala Collections Performance

http://www.goldmansachs.com/gs-collections/presentations/2015-03-17%20Scala%20Days%202015%20-%20Scala%20Collections%20Performance.pptx

- Scala Days San Francisco 2015
- Performance analysis for Scala collections and other implementations
- Suggestions for Scala collection implementation to improve its performance, based on the knowledge we acquire from Eclipse Collections experience

Goldman Sachs

Eclipse Collections accepts contributions

How To Contribute: https://github.com/eclipse/eclipse-collections/blob/master/CONTRIBUTING.md

- Summary: Sign Eclipse Foundation CLA and submit pull-requests
- Developer mailing list: https://dev.eclipse.org/mailman/listinfo/collections-dev
- Reporting Issues: https://github.com/eclipse/eclipse-collections/issues

Goldman Sachs

Resources

- Eclipse Collections on GitHub

  http://www.infoq.com/cn/news/2016/01/GS-Collections-Eclipse-Foundn

- Eclipse Collections Memory Benchmark

  http://www.goldmansachs.com/gs-collections/presentations/GSC_Memory_Tests.pdf

- Learn-by-Example: Eclipse Collections

  http://www.infoq.com/cn/articles/gs-collections-examples-tutorial-part1
  http://www.infoq.com/cn/articles/gs-collections-examples-tutorial-part2

  Eclipse Collections Kata https://github.com/eclipse/eclipse-collections-kata

we
BUILD

Learn more at GS.com/Engineering