Goldman Sachs

ゴールドマン・サックスのJavaへの取り組み

Java Day Tokyo 2015
#jtd62

ゴールドマン・サックス
テクノロジー部　ヴァイス・プレジデント
伊藤博志

we BUILD

Goldman Sachs

Agenda

- ゴールドマン・サックスのエンジニアリング
- ゴールドマン・サックスとJava
- OpenJDKの活用事例
  - OpenJDKを使う理由
  - トラブルシューティング事例
  - リサーチ事例
- GS Collections
  - なぜGS Collectionsなのか
  - Stream APIとの比較
  - デザインコンセプト
  - メモリ使用量の比較
  - GS Collectionsを始めてみよう

Goldman Sachs

ゴールドマン・サックスのエンジニアリング

ゴールドマン・サックスのエンジニアリング

ゴールドマン・サックスとは

投資銀行業務、証券業務および投資運用業務を中心に、企業、金融機関、政府機関、富裕層など多岐にわたるお客様を対象に幅広い金融サービスを提供している世界有数の金融機関です。1869年に創業、ニューヨークを本拠地として、世界の主要な金融市場に拠点を擁しています。

ゴールドマン・サックス　エンジニアリング

複雑な問題へのソリューションの構築、時代を変えるテクノロジーの創出、ビジネスと金融マーケットをグローバルに牽引するシステム開発を行っています。

http://www.goldmansachs.com/japan/what-we-do/engineering/index.html

Goldman Sachs

ゴールドマン・サックスのエンジニアリング

Goldman Sachs

ゴールドマン・サックスとJava

Goldman Sachs

ゴールドマン・サックスとJavaの歴史

1998	•社内でJavaの評価が始まる
2000	•Javaが戦略的プラットフォームのひとつに
2000-	•Sun/Oracleとともに継続的にJavaの改善に貢献
2004	•GS Collectionsの前身となるフレームワークの開発。ラムダ式導入への準備期間。
2012	•JCP Executive Committeeに選任 •GS Collections をオープン・ソースとして公開
2013	•OCA (Oracle Contributor Agreement) の締結 •OpenJDKへの貢献を開始

ゴールドマン・サックスにおけるJava

- JCPのExecutive Committeeメンバー
- 以下のJSRのエキスパートグループに属する
  - JSR 107: JCACHE – Java Temporary Caching API
  - JSR 335: Lambda Expressions for the Java Programming Language
  - JSR 351: Java Identity API

- **3000人**を超えるJavaエンジニア
- **1.3億**行を超えるJavaのソースコード
- 1時間ごとに平均して約**12万5千**ものJavaプロセスが走る

Goldman Sachs

Javaの最前線での取り組み

- Javaの技術的な限界の引き上げ
  - 巨大なヒープ
    - 例：190Gを超えるヒープを持つアプリケーション
      - GrowableArrayでIntegerのオーバーフローが起きることによるMark Stackの問題
      - https://bugs.openjdk.java.net/browse/JDK-6806226
  - 大量の処理や通常外の利用パターンによるJVMの限界への挑戦
    - 例：コードキャッシュのキャパシティ
- 社内の基盤技術のオープンソース化
  - GS Collections : Stream APIを凌駕するフレームワーク

JavaOne 2014への参加

- キーノート登壇
- セッション：GS Collections and Java 8: Functional, Fluent, Friendly, and Fun
- セッション：Banking on OpenJDK: How Goldman Sachs Is Using and Contributing to OpenJDK

Goldman Sachs

OpenJDKの活用事例

Goldman Sachs

OpenJDKを使う理由

- ソースコードへのアクセスの重要性
  - 本番環境で発生する問題への迅速な対応
  - 改善の余地がある機能の見極め
  - セキュリティへの対応
- 透明性の確保
  - コードの開発過程の理解
  - バグデータベースの活用
- バグや機能拡張のパッチを送ることでJavaコミュニティへの還元

Goldman Sachs

OpenJDKの活用法

- トラブルシューティング
  - 本番環境における問題解決
  - 開発時の活用
- 新機能のリサーチ
  - JDKにおいて社内の活用に特化して最適化できる機能の研究
  - 新機能のアプリケーションへの活用法の研究

Goldman Sachs

トラブルシューティング

JDKのソースコードアクセスが重要

- 迅速な問題の分析
  - クラッシュした際のダンプの分析
  - デバッグ用のビルド作成
- 回避策と修正の見極め
  - パッチを当てたビルドを作成し、エンジニアが修正を確認
    - GAビルドにまだ適用されていないパッチも含められる

Goldman Sachs

トラブルシューティング: ケーススタディ

JVMTIエージェントが下記のようなアサーションを投げ、JVMクラッシュ

```
*** java.lang.instrument ASSERTION FAILED ***: "error == JVMTI_ERROR_NONE" at
../../../src/share/instrument/Reentrancy.c line: 161
```

条件

- デーモンスレッド
- シャットダウン中にクラスローディング

テストケース

- デーモンスレッドにクラスローディングをさせ、JVMを終了させる

原因究明

- ソース： src/share/instrument/Reentrancy.c
- JDK-6572160において、シャットダウン時にコンディションチェックが足されるというバグ修正を発見
- このチェックがいくつかのユーティリティ関数に存在していないことが判明

パッチのコミット

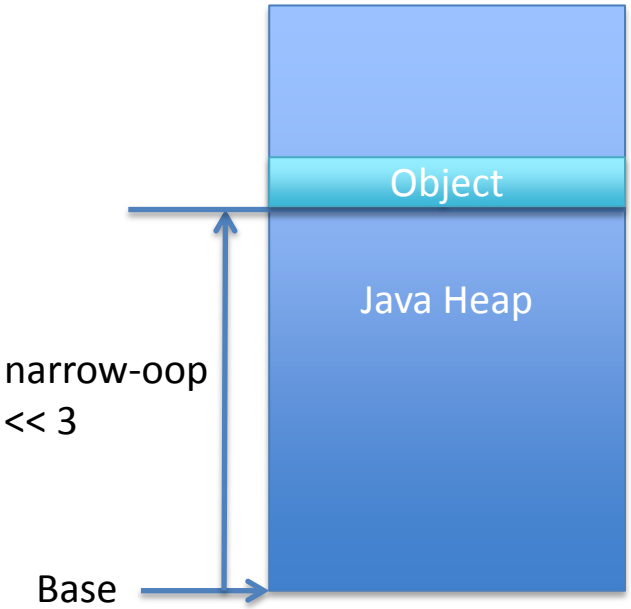
- http://mail.openjdk.java.net/pipermail/serviceability-dev/2014-January/013991.htmlにてパッチのディスカッション
- JDK 8の開発ブランチにてアクセプトされる
- 後に当社のリクエストによりJDK 7にバックポートされる

Goldman Sachs

リサーチ

- Hotspotの新機能の理解
  - 実装の理解
  - 社内のアプリケーションに適用する利点があるか
  - 利点があるとしたらどう適用していくか
- パフォーマンスボトルネックの発見
  - 回避策や、アプリケーションのパフォーマンスを改善する機能の研究

リサーチ: ケーススタディ



事例 : 圧縮OOP

- オブジェクトの参照を64-bit (8 bytes)の代わりに32-bit (4 bytes)でストアする
- もともと32Gのヒープサイズに限られていた (8-byte alignment)
- プライベートビルドでテストした結果、16-byte alignment (<<4) でも社内のアプリケーションに有利に働く知見が得られた。
- JVMのデフォルトになる前に開発者に推奨

Goldman Sachs

GS Collections

Goldman Sachs

GS Collectionsとは?

- ゴールドマン・サックスの開発したオープンソースコレクションフレームワーク
  - 2004年より開発を開始
  - GitHubにて公開（Apache 2.0 License）
    - github.com/goldmansachs/gs-collections
- GS Collections Kata
  - 2007年に開発されたトレーニング教材
  - 1,500人以上のGS Javaエンジニアが受講
  - GitHubにて公開（Apache 2.0 License）
    - github.com/goldmansachs/gs-collections-kata

Goldman Sachs

Stream APIがある今
なぜGS Collectionsなのか？

Goldman Sachs

GS Collectionsの豊富な機能

- コレクションの即時実行イテレーションパターン
- より豊富なイテレーションパターン
  - zip(), chunk(), partition(), makeString(), tap() 他多数
- コレクションプロトコルにおける共変の戻り値型
- 新しいコレクション型
  - Bag, SortedBag, BiMap, Multimap
- メモリ効率のよいSetとMapの実装
- プリミティブ型のコンテナ
- イミュータブルなコンテナ

Stream API: ハンバーガーみたい

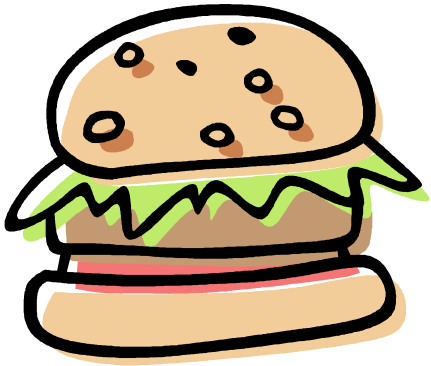
```
List<Person> people = Arrays.asList(
                    new Person("Bob", 17),
                    new Person("Dave", 23),
                    new Person("Joe", 32));

List<Person> adults = people.stream()
    .filter(person -> person.getAge() > 20)
    .collect(Collectors.<Person>toList());
```

Bun

Meat

Bun



ちょっと冗長？

Goldman Sachs

GS Collectionsで書くと

```
MutableList<Person> people = Lists.mutable.with(
                        new Person("Bob", 17),
                        new Person("Dave", 23),
                        new Person("Joe", 32));

MutableList<String> adults = people
```

Meat

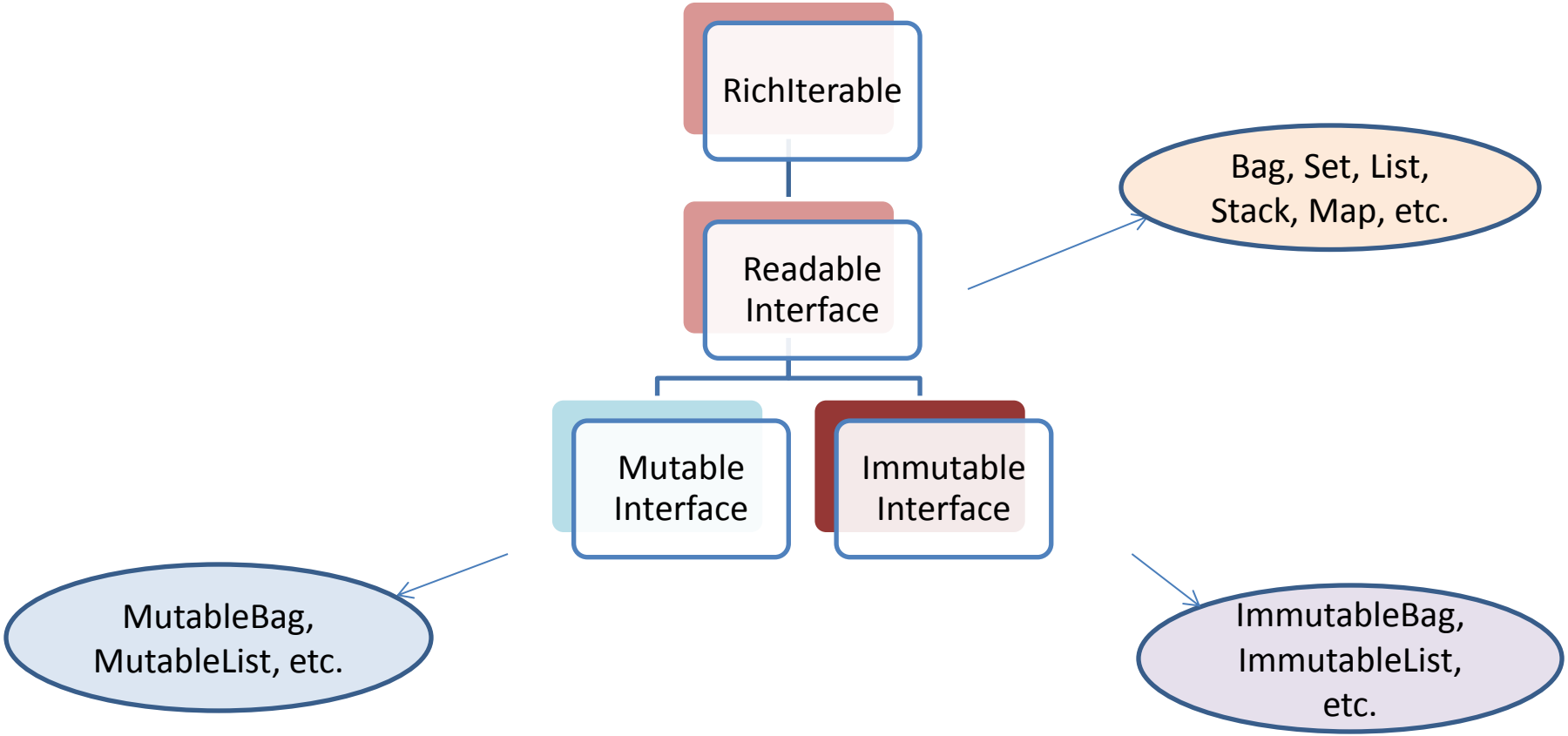
```
    .select(person -> person.getAge() > 20);
```

うまみの凝縮したステーキ！！

簡潔に書けます

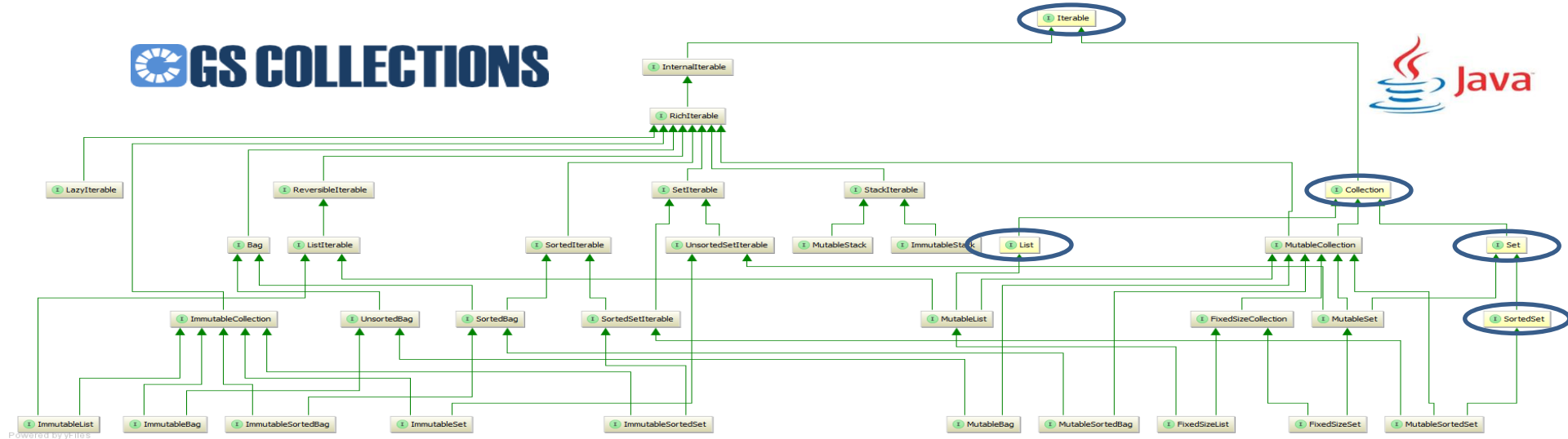
Goldman Sachs

デザインコンセプト



Goldman Sachs

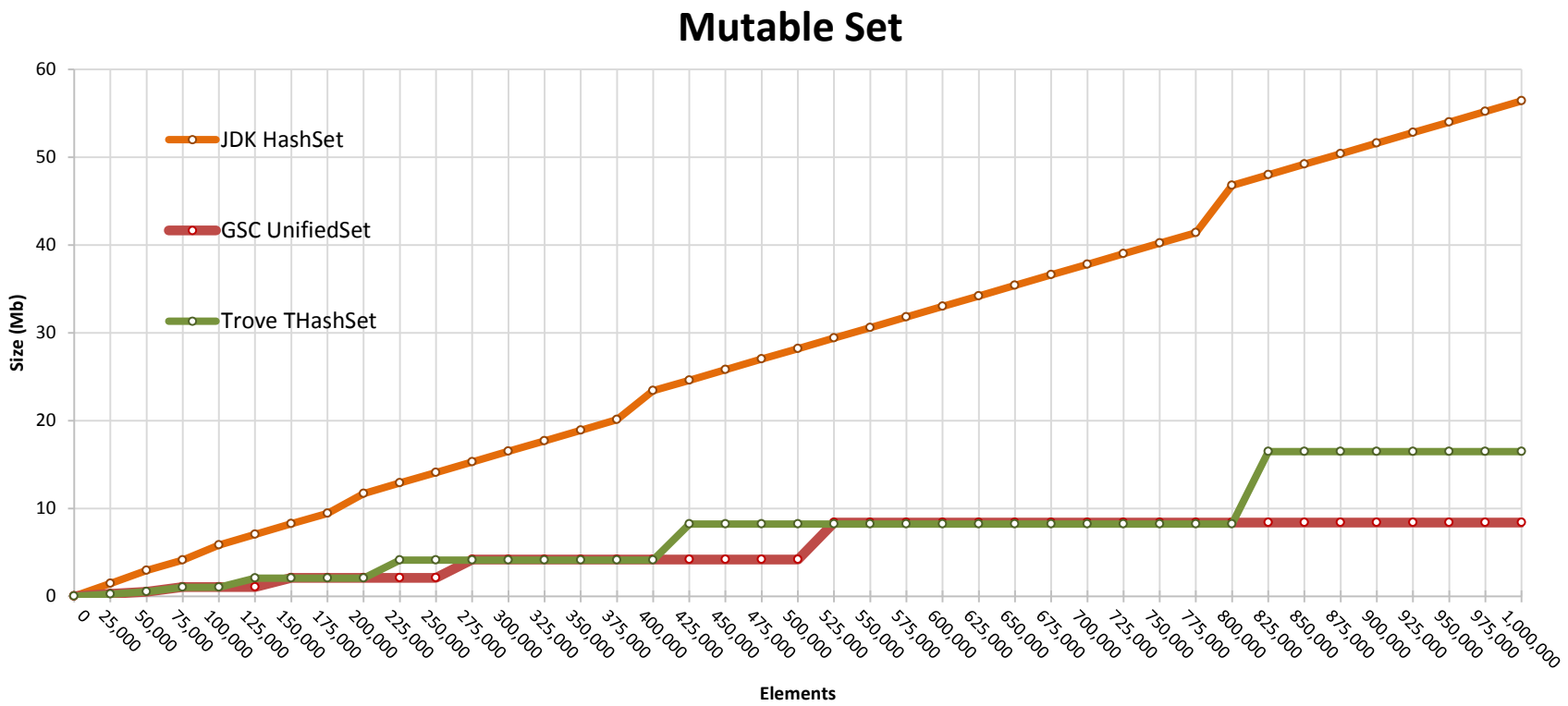
Collectionの階層



GS CollectionsはJDK Collections のインターフェースを拡張し、
30ものインターフェースを追加しています。

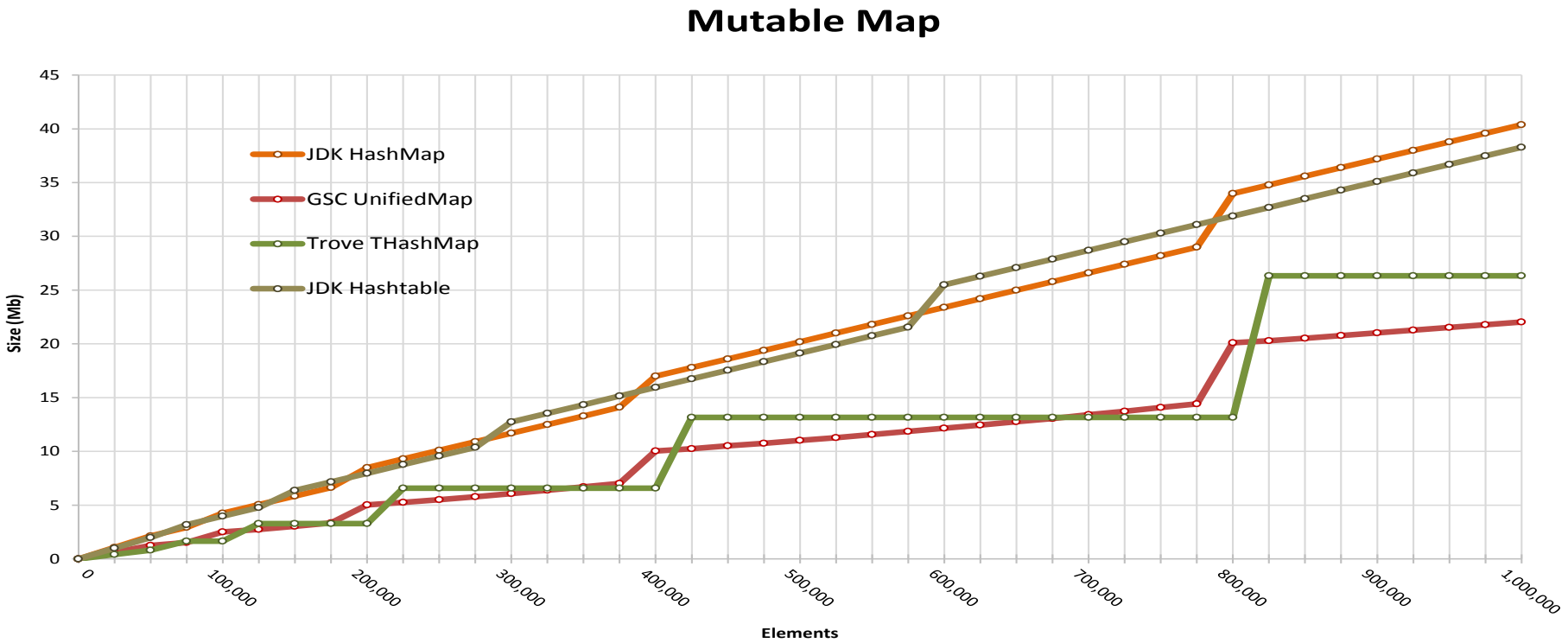
Goldman Sachs

UnifiedSetでメモリ削減



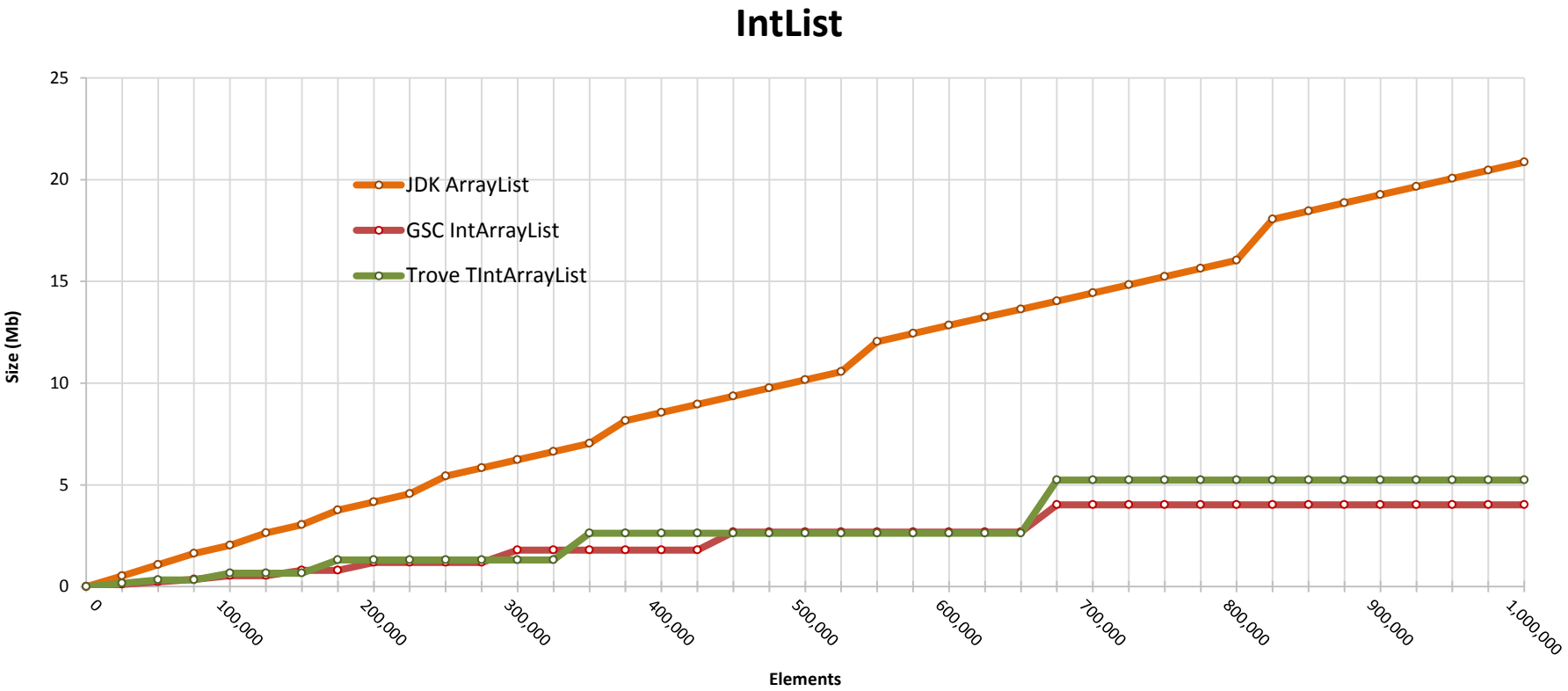
Goldman Sachs

UnifiedMapでメモリ削減



Goldman Sachs

プリミティブ型のCollectionでメモリ削減



Goldman Sachs

コレクションフレームワークの比較

Features	GS 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)

GS Collectionsを始めてみよう

Maven

```
<dependency>
    <groupId>com.goldmansachs</groupId>
    <artifactId>gs-collections-api</artifactId>
    <version>6.1.0</version>
</dependency>

<dependency>
    <groupId>com.goldmansachs</groupId>
    <artifactId>gs-collections</artifactId>
    <version>6.1.0</version>
</dependency>

<dependency>
    <groupId>com.goldmansachs</groupId>
    <artifactId>gs-collections-testutils</artifactId>
    <version>6.1.0</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>com.goldmansachs</groupId>
    <artifactId>gs-collections-forkjoin</artifactId>
    <version>6.1.0</version>
</dependency>
```

お気に入りのビルド・依存性管理ツールで

Ivy

```
<dependency org="com.goldmansachs" name="gs-collections-api" rev="6.1.0" />
<dependency org="com.goldmansachs" name="gs-collections" rev="6.1.0" />
<dependency org="com.goldmansachs" name="gs-collections-testutils" rev="6.1.0" />
<dependency org="com.goldmansachs" name="gs-collections-forkjoin" rev="6.1.0"/>
```

Gradle

```
dependencies {
    compile group: 'com.goldmansachs', name: 'gs-collections-api', version: '6.1.0'
    compile group: 'com.goldmansachs', name: 'gs-collections', version: '6.1.0'
    testCompile group: 'com.goldmansachs', name: 'gs-collections-testutils', version: '6.1.0'
    compile group: 'com.goldmansachs', name: 'gs-collections-forkjoin', version: '6.1.0'
}
```

Goldman Sachs

GS Collectionsを始めてみよう

様々なコンテナの初期化の例

List, Set, Bag, Map, Multimap他、com.gs.collections.imple.factory下に多数存在

```
MutableSet<String> mutableSet = Sets.mutable.of("One", "One", "Two", "Three");

MutableBag<String> mutableBag = Bags.mutable.of("One", "One", "Two", "Three");

MutableMap<String, String> mutableMap = Maps.mutable.of("key1", "value1", "key2", "value2", "key3", "value3");

Multimap<String, String> multimapWithList = Multimaps.mutable.list.with("key1", "value1-1", "key1", "value1-2", "key2", "value2-1");
```

それぞれMutable, Immutableのファクトリが存在

```
MutableList<String> mutableList = Lists.mutable.of("One", "Two", "Three");       //Equivalent of FastList.newList("One", "Two", "Three")

MutableList<String> mutableListWithBlank = Lists.mutable.with();  //You have a choice to use of() or with(), whichever you like.

ImmutableList<String> immutableList= Lists.immutable.of("One", "Two", "Three");
```

List, Set, Map等のコンテナにはメモリ効率のよいFixedSize用のファクトリも存在

```
FixedSizeList<String> fixedSizeList= Lists.fixedSize.of("One", "Two");     //Memory efficient if we know the size beforehand
```

Goldman Sachs

GS Collectionsを始めてみよう

即時実行のイテレーションパターン

```
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);
```

遅延実行のイテレーションパターン

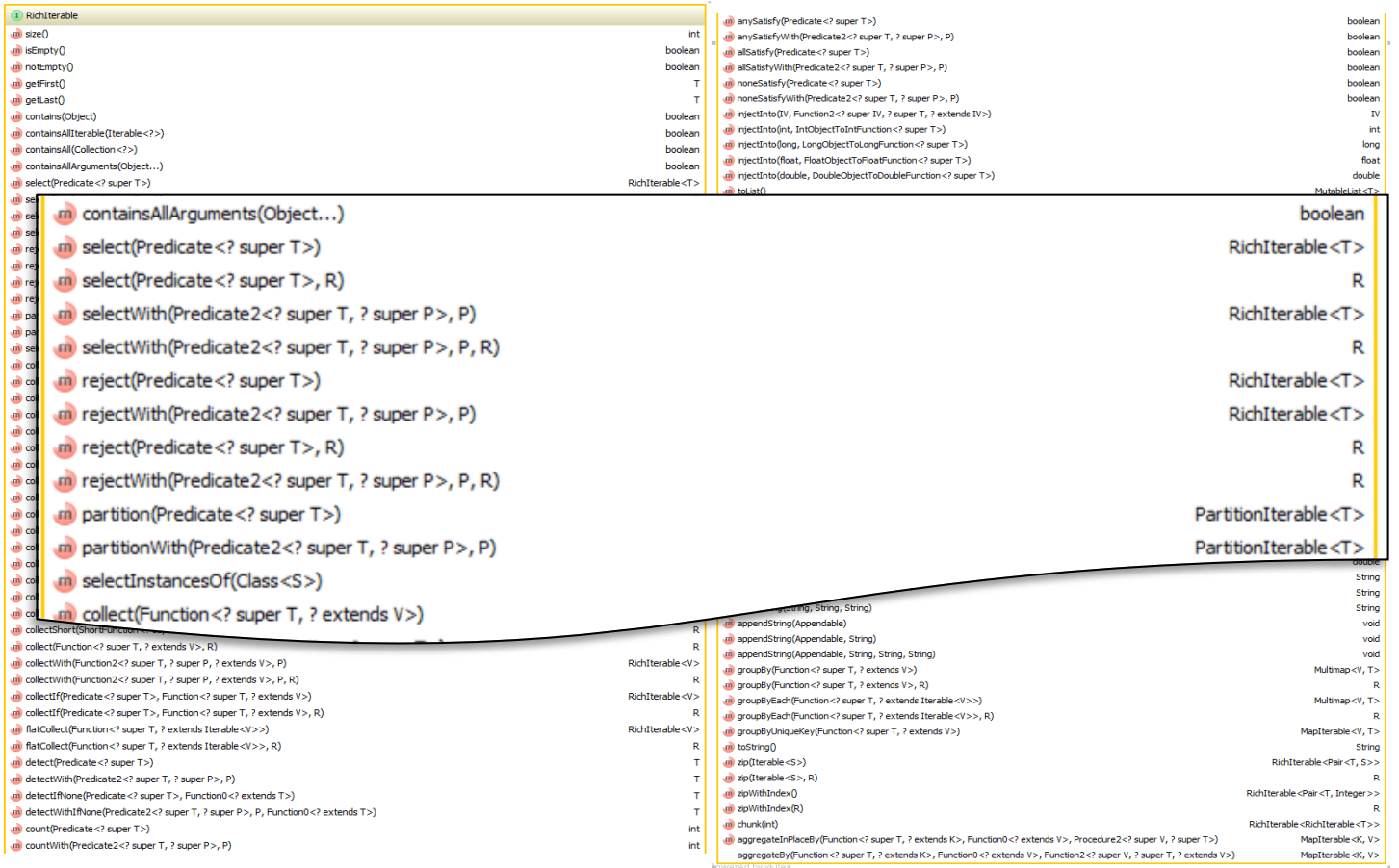
```
list.asLazy()
    .select(each -> each > 95)
    .collect(Object::toString)
    .makeString("[", ",", "]");


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

GS Collectionsを始めてみよう

GS Collectionsにはここでは紹介しきれない様々な機能が数多くあります！

まずはRichIterable上に存在する様々なAPIを堪能し、JavaDoc、パッケージやインターフェースを探索してみてください。



付録

Goldman Sachs

セッション - Parallel-lazy Performance: Java 8 vs Scala vs GS Collections

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

QCon NY発表後に注目を浴びページビュートップに

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

出典：http://www.infoq.com/news/2014/12/qcon-ny-2015 より引用 （オレンジの囲みは発表者による）

セッション - GS Collections and Java 8: Functional, Fluent, Friendly, and Fun

http://www.goldmansachs.com/gs-collections/presentations/2014-09-29_JavaOne_GSC.pptx

JavaOne 2014で好評を得たセッション

JJUG CCC 2015 Spring にて日本語での再演決定！！

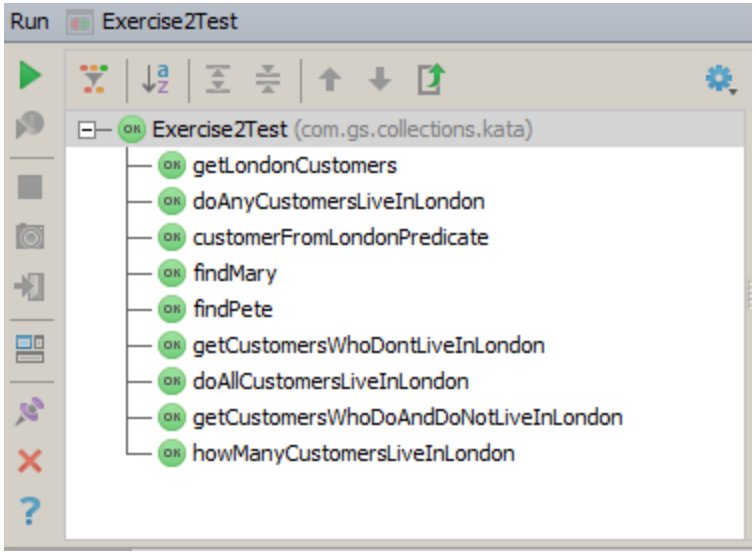
Room F:
15:00 – 15:50

出典：http://www.java-users.jp/?page_id=1663 より引用

研修資料 - GS Collections Kata

https://github.com/goldmansachs/gs-collections-kata

- ユニットテストをひとつずつパスしていくTDD型トレーニングマテリアル
- ゴールドマン・サックスの研修にも使用
- GS Collectionsの使用法を学習するのに最適



Goldman Sachs

セッション - 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
- Scalaのコレクション実装のパフォーマンス分析
- GS Collectionsで得られた知見から、Scalaのコレクション実装のパフォーマンス改善案を提言

Goldman Sachs

Resources

- GS Collections on GitHub

  https://github.com/goldmansachs/gs-collections

  https://github.com/goldmansachs/gs-collections/wiki

  https://github.com/goldmansachs/gs-collections-kata

- GS Collections Memory Benchmark

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

- 実例で学ぶGS Collections （InfoQ 日本語翻訳記事）

  http://www.infoq.com/jp/articles/GS-Collections-by-Example-1

  http://www.infoq.com/jp/articles/GS-Collections-by-Example-2

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Goldman Sachs

we

BUILD

Learn more at GS.com/Engineering

© 2015 Goldman Sachs. This presentation should not be relied upon or considered investment advice. Goldman Sachs does not warrant or guarantee to anyone the accuracy, completeness or efficacy of this presentation, and recipients should not rely on it except at their own risk. This presentation may not be forwarded or disclosed except with this disclaimer intact.