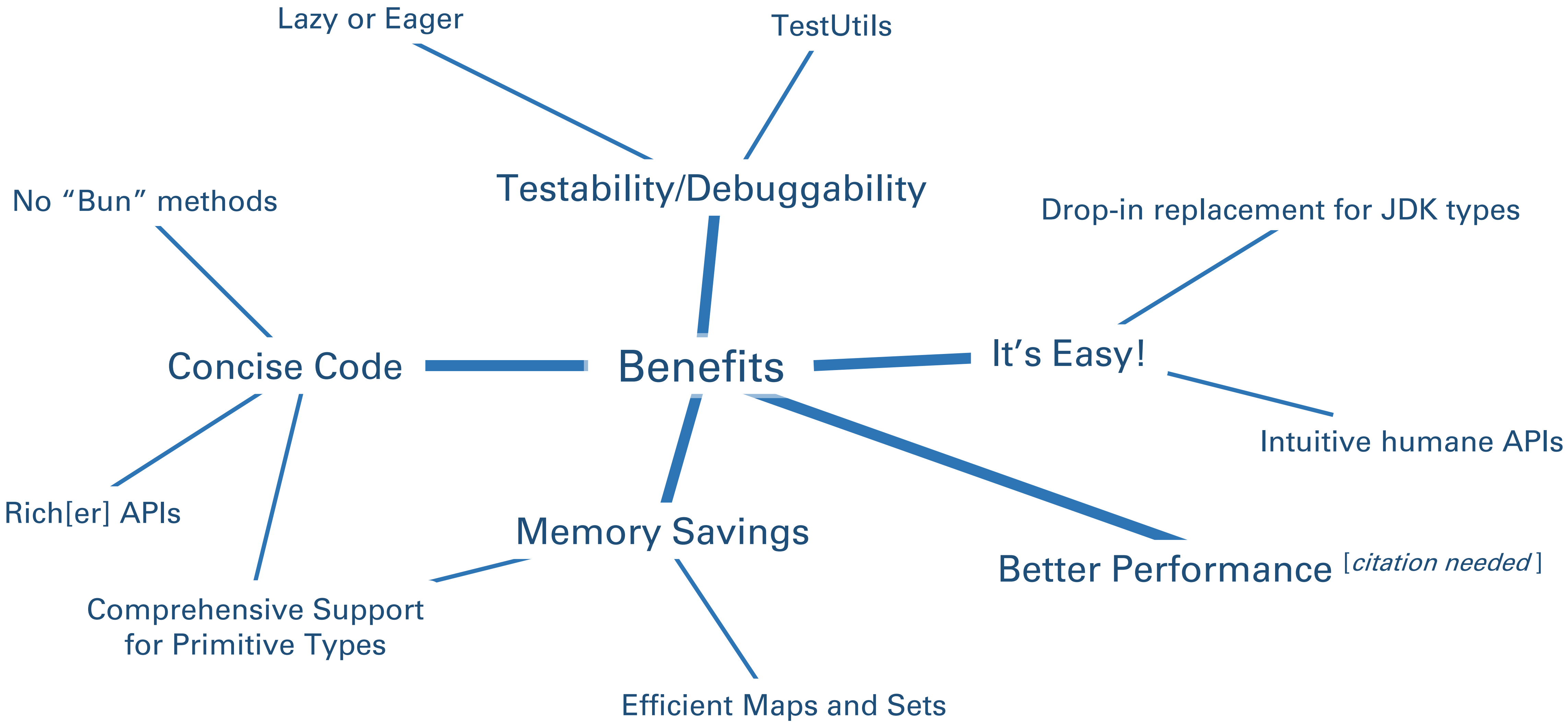
Refactoring to Eclipse Collections

Making Your Java Streams
Leaner, Meaner, and Cleaner

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?



Benefits

Testability/Debuggability

Lazy or Eager

TestUtils

Concise Code

No “Bun” methods

Rich[er] APIs

Comprehensive Support for Primitive Types

Memory Savings

Efficient Maps and Sets

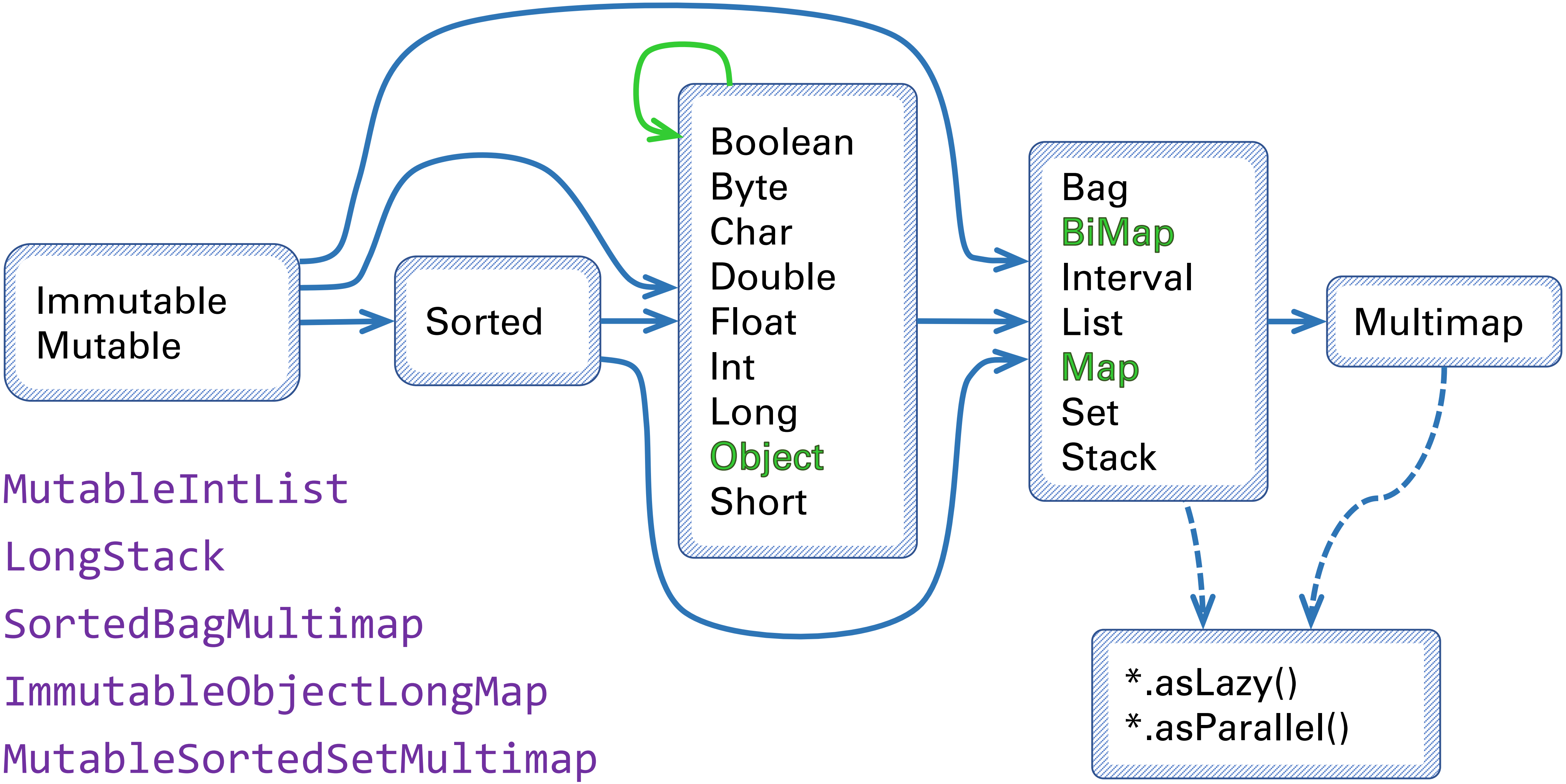
It’s Easy!

Drop-in replacement for JDK types

Intuitive humane APIs

Better Performance [*citation needed*]

Any Types You Need



MutableIntList

LongStack

SortedBagMultimap

ImmutableObjectLongMap

MutableSortedSetMultimap

Instantiate Them Using Factories

Primitive	Container				
Type	**Type**	**Mutability**	**Multimap**	**Initialized**	**Lazy**
Boolean	Bags	.mutable	.bag	.empty() | .of() | .with()	.asLazy()
Byte	BiMaps	.immutable	.list	.of(one) | .with(one)	
Char	Lists	.fixedSize	.set	...	
Double	Maps		.sortedSet	.of(one,…,ten)	
Float	Multimaps			.with(one,…,ten)	
Int	Sets			.of(... elements)	
Long	SortedBags			.with(... elements)	
Object	SortedMaps			.ofAll(Iterable)	
Short	SortedSets			.withAll(Iterable)	
	Stacks				

$$\underbrace{\overbrace{\texttt{LongStacks.}\textit{\textbf{immutable}}\texttt{.with(1, 2, 3)}}^{\texttt{ImmutableLongStack}}\texttt{.asLazy()}}_{\texttt{LazyLongIterable}}$$

Methods by Category - Highlights

transform

collect[With]
collect[Boolean,Byte,Char,Double,Float,Int,Long,Short]
collectIf
collectKeysAndValues
collectValues
collectWithIndex
collectWithOccurrences
flatCollect

group

groupBy
groupByEach
groupByUniqueKey
sumBy[Double,Float,Int,Long]
sumOf[Double,Float,Int,Long]
aggregateBy
aggregateInPlaceBy

wrap

asLazy
asParallel
asReversed
asSynchronized
asUnmodifiable

find

detect[With]
detect[With]IfNone
detect[With]Optional
max[By]
min[By]

convert

toArray
toBag
toImmutable
toList
toMap
toReversed
toSet
toSortedBag[By]
toSortedList[By]
toSortedMap
toSortedSet
toSortedSet[By]
toStack
toString

filter

select[With]
selectByOccurrences
selectInstancesOf
reject[With]
partition[With]
partitionWhile

test

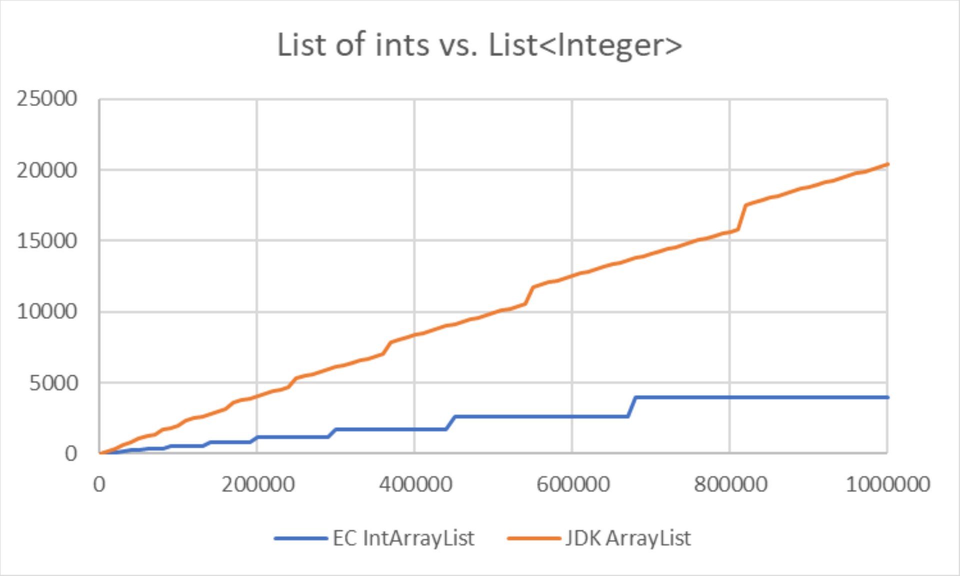
allSatisfy[With]
anySatisfy[With]
noneSatisfy[With]
notEmpty
isEmpty

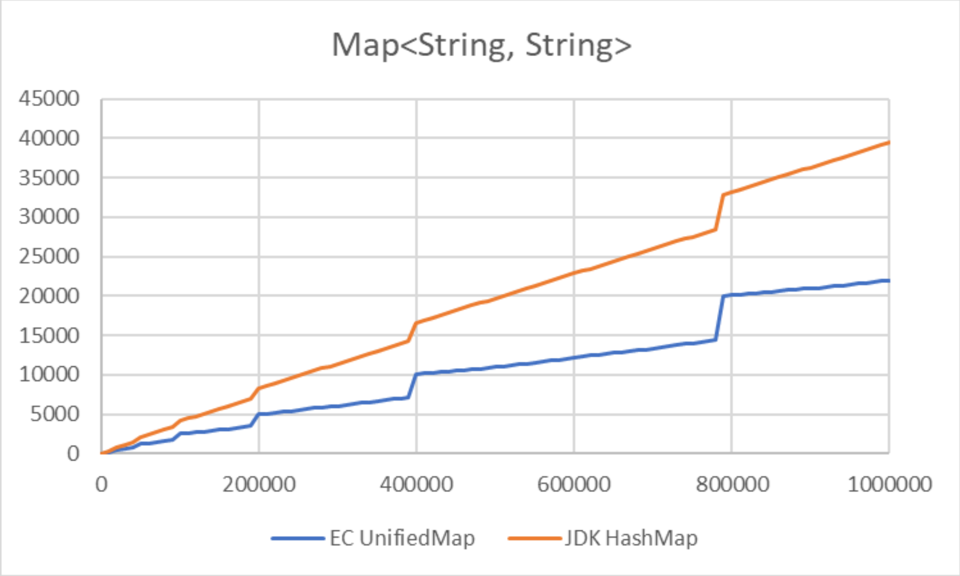
Methods – Lots More…

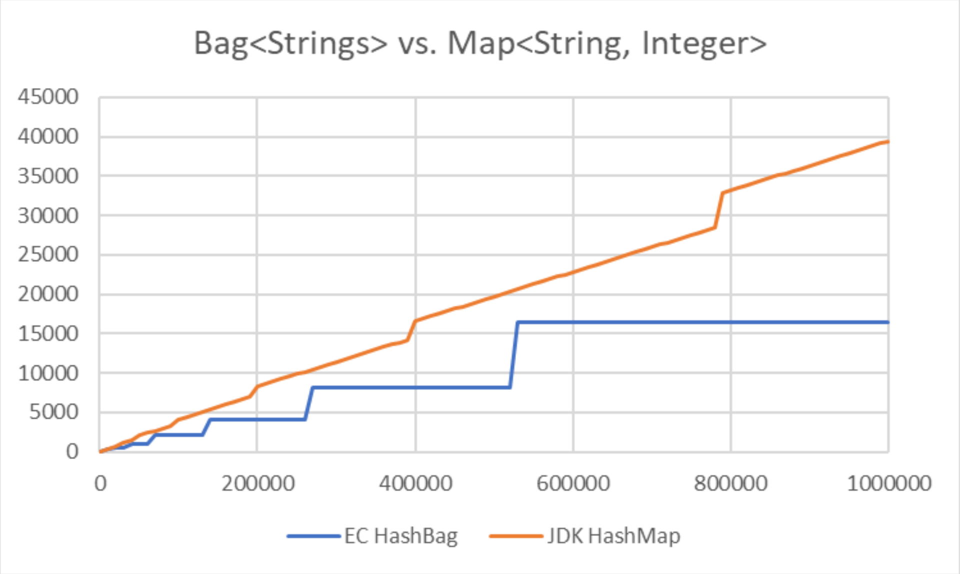
correspond
collectfloat
detectwithifnone
headset
casttomap
headmap
compute
getclass
collectwith
aslazy
collectshort
issubsetof
notifyall
newwithallkeyvalueargument
partition
appendstring
indexof
tosortedset
parallelstream
sortthisbyint
binarysearch
tosortedbagby
oddsfromto
tomapofitemtocount
collectboolean
submap
tosortedlistby
fromtoby
sumofdouble
replace
removeall
newwithall
difference
peek
tostring
inverse
tostack
foreachwith
anysatisfywith
summarizefloat
reversethi
zipwithindex
select
bottomoccurrence
foreachkey
firstkey
containskey
countwith
tointarray
newwithallkeyvalue
symmetricdifference
detectwithoptional
notempty
goforward
containsall
keyset
injectinto
topoccurrence
powerset
tailmap
aggregateby
value
get
union
zeroto
newempty
foreachwithindex
collect
takewhile
flip
dropwhile
toimmutable
newwithoutkey
sumbylong
sublist
clear
toarray
last
zero
remove
keysview
selectinstancesof
size
newwithkeyvalue
detectwith
selectbyoccurrence
toreversed
entryset
retainall
tailset
chunk
withkeyvalue
nonesatisfy
aggregateinplaceby
isempty
countby
maxby
max
put
oneto
pop
fromto
addall
tolist
notify
removeif
putpair
product
groupby
getfirst
reject
zip
flatcollect
tosortedlist
sumoflong
newwith
tobag
push
subset
withall
tomap
merge
zerotoby
count
wait
iterator
run
equal
intersect
hashcode
sort
tap
first
withoutkey
flipuniquevalue
detect
foreach
drop
rejectwith
stream
asparallel
lastkey
newwithoutall
collectif
without
anysatisfy
maxoptional
casttolist
clone
spliterator
minby
collectvalue
comparator
sortthisby
withoutall
contain
take
set
collectint
toset
sumbyint
sortthi
collectdouble
factorial
forceput
selectwith
min
summarizeint
minoptional
collectwithindex
countbywith
withallkeyvalueargument
distinct
getifabsent
allsatisfy
sumbyfloat
casttoset
reduceinplace
withallkeyvalue
concatenate
add
asunmodifiable
listiterator
sumbydouble
reduce
detectifnone
getonly
nonesatisfywith
groupbyeach
partitionwith
addallto
allsatisfywith
tosortedbag
makestring
assynchronized
partitionwhile
distinctby
sizedistinct
minbyoptional
addoccurrence
maxbyoptional
groupbyuniquekey
getlast
newwithout
lastindexof
tosortedsetby
intersectinto
toreverselist
differenceinto
collectkeysandvalue
setoccurrence
asreversed
selectandrejectwith
containsalliterable
tosortedmap
newwithoutallkey
withoutallkey
getifabsentputwith
getifabsentput
peekat
updatevaluewith
valuesview
sortthisbybyte
getifabsentputwithkey
summarizelong
shufflethi
detectoptional
putall
reverseforeach
sumofint
removealliterable
casttosortedset
putifabsent
injectintowith
collectwithoccurrence
sumoffloat
onetoby
summarizedouble
replaceall
containsvalue
removekey
unioninto
sortthisbyboolean

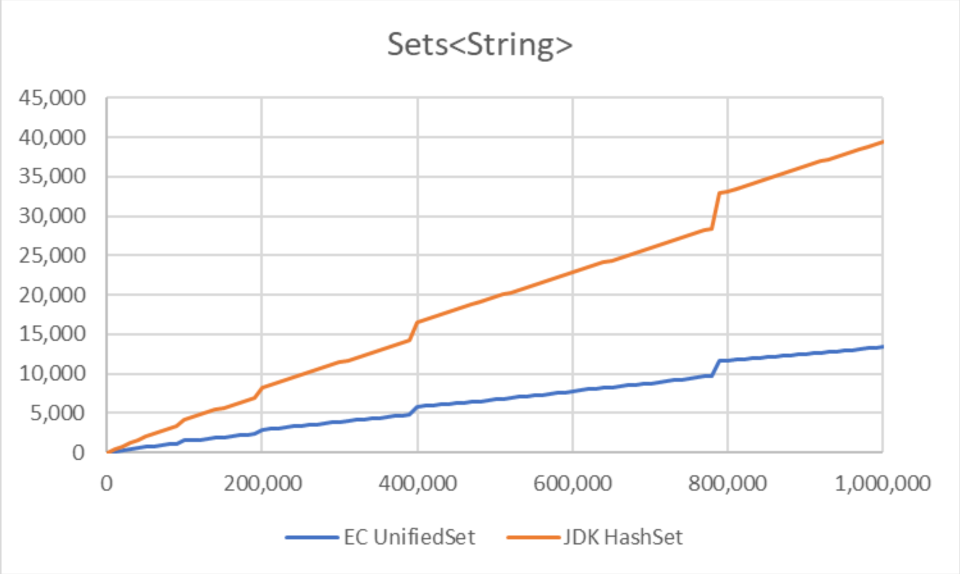
Let’s Do It!

Memory Usage (Overhead in KB, Count)

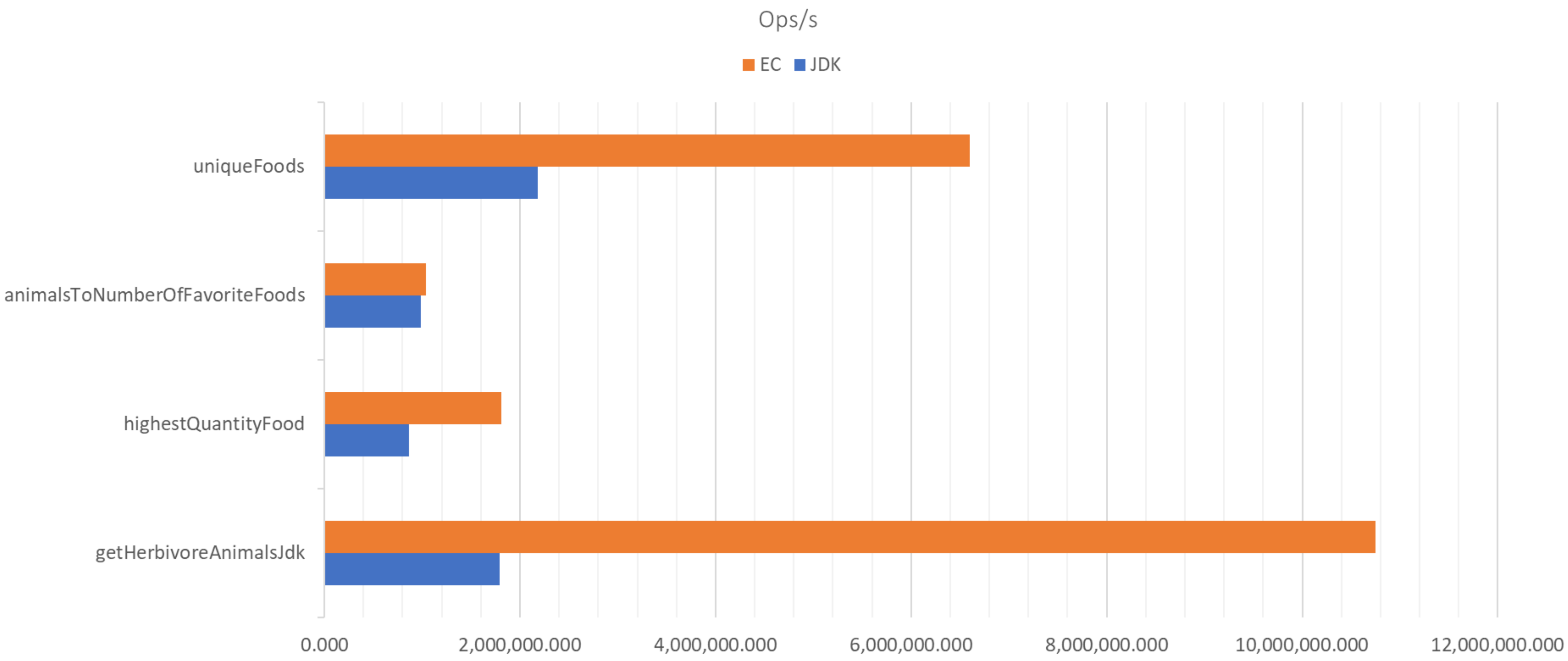








JMH Benchmark Results


Ops/s
🟧 EC 🟦 JDK
uniqueFoods
animalsToNumberOfFavoriteFoods
highestQuantityFood
getHerbivoreAnimalsJdk