Guava Release 21.0: Release Notes

- 21.0 was released on January 12, 2017.
- 21.0-rc2 was released on January 3, 2017.
- 21.0-rc1 was released on December 19, 2016.

(See [[ReleaseHistory]].)

API documentation:

- guava
- guava-testlib

Known issues

- If you see errors like "cannot access com.google.errorprone.annotations.CanIgnoreReturnValue," you can work around them by adding a local dependency on `error_prone_annotations`. (This problem is fixed in Guava 22, which makes that dependency present for users.)

Using Guava in your project

	Guava	Guava (GWT)
Maven Identifier	com.google.guava:guava:21.0	com.google.guava:guava-gwt:21.0
Jar	guava-21.0.jar	guava-gwt-21.0.jar
Javadoc	guava-21.0-javadoc.jar	guava-gwt-21.0-javadoc.jar
Sources	guava-21.0-sources.jar	guava-gwt-21.0-sources.jar

See [[UseGuavaInYourBuild]] for help integrating Guava into your build environment.

Java 8!

Important: Guava 21.0 *requires* Java 8. If you need Java 7 or Android compatibility, use Guava 20.0 for now. Guava 22.0 and on will introduce a Java 7/Android compatible backport of Guava that includes all of the latest changes that don't require Java 8. (As for Java 6, we don't plan to support it past 20.0.)

Issues resolved

5+ issues are resolved in this release.

API Changes

Full JDiff Report of changes since release 20.0.

Significant API additions and changes

common.base

- `Function`, `Predicate` and `Supplier`: changed to extend the new `java.util.function` interfaces with the same names.

- `Optional` : added `toJavaUtil` and `fromJavaUtil` methods for easy conversion between Guava's `Optional` and `java.util.Optional` .
- `Objects` : removed deprecated `firstNonNull` and `toStringHelper` methods (both found on `MoreObjects` since Guava 18.0).

common.cache

New default methods on `ConcurrentMap` that were added in Java 8 are now implemented and safe to use for `Cache.asMap()` views.

common.collect

Many APIs in collect now have better implementations of many of the `default` methods added to `Collection` and `Map` types in Java 8.

New classes

- `Comparators` : With the addition of many useful methods to the `Comparator` type in Java 8, `Ordering` now provides little benefit. `Comparators` is a new location for methods on `Ordering` that still don't have a good equivalent in the JDK.
- `Streams` : Utility class for working with `java.util.stream.Stream` . Includes methods for creating streams (such as `stream(Iterable)` , `stream(Optional)` and `concat(Stream...)`) and methods that do things with streams (such as `findLast(Stream)`).
- `MoreCollectors` : Factories for `java.util.stream.Collector` objects; note that `Collector` s for Guava's collection types are generally found on those types themselves rather than here.
- `Interners.InternerBuilder` : Builder for `Interner` instances, with options similar to those found on `MapMaker` . Created with `Interners.newBuilder()` .

Removed classes

- `MapConstraint` and `MapConstraints` : deprecated since 19.0.

Changes

- `FluentIterable` : added `stream()` method.
- `ForwardingBlockingDeque` : deprecated; moved to `util.concurrent` .
- `Immutable*` types: added methods to all named `toImmutable[Type]()` (e.g. `ImmutableList.toImmutableList()`) which return a `Collector` for collecting a `Stream` into an immutable collection/map object. As with most methods that create `Collector` s, these are generally intended to be used as static imports.
- `Multimap` : added `forEach(BiConsumer)` method.
- `Multimaps` : added `toMultimap` and `flatteningToMultimap` methods returning `Collector` objects that collect to a `Multimap` .
- `Multiset` : added `forEachEntry(ObjIntConsumer)` method.
- `Maps` : added `toImmutableEnumMap` methods returning `Collector` objects that collect to an `ImmutableMap` with `enum` keys.
- `Sets` : added `toImmutableEnumSet` method returning a `Collector` that collects to an `ImmutableSet` of `enum` s.
- `Tables` : added `toTable` methods returning `Collector` objects that collect to a `Table` .
- `RangeSet` : added default `addAll(Iterable<Range>)` , `removeAll(Iterable<Range>)` and `enclosesAll(Iterable<Range>)` methods.

- `ImmutableRangeSet` : added `copyOf(Iterable<Range>)` , `unionOf(Iterable<Range>)` , `union(RangeSet)` , `intersection(RangeSet)` and `difference(RangeSet)` methods.
- `TreeRangeSet` : added `create(Iterable<Range>)` method.
- A number of rarely-used methods on concrete implementations of Guava collection types that aren't present on the interface they implement have been deprecated. These include: `ArrayListMultimap.trimToSize()` , `TreeMultimap.keyComparator()` , and `TreeBasedTable.row/columnComparator()` .

common.io

- `MoreFiles` : New class which contains methods similar to those in `Files` , but for use with `java.nio.file.Path` objects.
- This includes `deleteRecursively` and `deleteDirectoryContents` methods which are secure against race conditions that Java previously had no way of dealing with provided that the `FileSystem` supports `SecureDirectoryStream` (modern Linux versions do; Windows [NTFS at least] does not). For security, these will throw an exception if `SecureDirectoryStream` is not supported unless `RecursiveDeleteOption.ALLOW_INSECURE` is passed when calling the method.

common.primitives

- Most classes: added `constrainToRange([type] value, [type] min, [type] max)` methods which constrain the given value to the closed range defined by the `min` and `max` values. They return the value itself if it's within the range, the `min` if it's below the range and the `max` if it's above the range.

common.util.concurrent

- `ForwardingBlockingDeque` : added; moved from `common.collect` because `BlockingDeque` is a concurrent type rather than a standard collection (it's defined in `java.util.concurrent`).
- `AtomicLongMap` : added a number of methods such as `accumulateAndGet(K, LongBinaryOperator)` that take advantage of new Java functional types.
- `Monitor` : added `newGuard(BooleanSupplier)` .
- `MoreExecutors` : removed `sameThreadExecutor()` ; deprecated since 18.0 in favor of `directExecutor()` (preferable) or `newDirectExecutorService()` .