Guava Release 22.0: Release Notes

- 22.0 was released on May 22, 2017.
- 22.0-rc1 was released on May 2, 2017.

(See [[ReleaseHistory]].)

New: Android version

The main Guava artifact (`22.0`) requires Java 8, as it has since 21.0.

New in 22.0 is a version of Guava targeted at Android, differentiated by its version number: `22.0-android`.

While the focus of this version is on Android, it's also the version you should use if you need to target Java 7.

We're using version number rather than artifact ID to differentiate so that Maven can detect both versions being in its dependency tree; having both in the dependency tree should be avoided. The intent is that the Android version always be a strict subset of what's available in the main/Java 8 version, so anything compiled against version N of the Android version will work when run against the Java 8 version. So if you're a library depending on Guava and want to support Android and/or Java 7 users, you should depend on the `-android` version.

So what's different in the Android version?

- Primarily, Java 8-specific changes made in 21.0 are not present. Nothing extends new Java 8 types and APIs for supporting new Java 8 APIs are removed. See the API diffs between the Java 8 and Android versions for specifics.
- While not Java 8, the `MoreFiles` class (added in 21.0) is not present because it requires (and is only useful with) support for the `java.nio.file` APIs, which are not supported on Android.
- Some optimizations have been made to a number of data structures (primarily our `ImmutableCollection` implementations) to make them better for use on Android: lower memory footprint and less garbage generated in their use, for example.

API documentation

Java 8:

- guava
- guava-testlib

Android / Java 7:

- guava
- guava-testlib

Using Guava in your project

	Guava	Guava (Android)	Guava (GWT)
Maven Identifier	com.google.guava:guava:22.0	com.google.guava:guava:22.0-android	com.google.guava:guava-gwt:22.0
Jar	guava-22.0.jar	guava-22.0-android.jar	guava-gwt-22.0.jar
Javadoc	guava-22.0-javadoc.jar	guava-22.0-android-javadoc.jar	guava-gwt-22.0-javadoc.jar

Sources	guava-22.0-sources.jar	guava-22.0-android-sources.jar	guava-gwt-22.0-sources.jar

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

Issues resolved

9+ issues are resolved in this release.

API Changes

- Java 8: Full JDiff Report of changes since release 21.0.
- Android: Full JDiff Report of changes since release 20.0 (the previous release that could be used on Android).

Significant API additions and changes

(For Android, see the changes for [[21.0|Release21]], ignoring those that are Java 8 specific, or the JDiff report linked above.)

common.primitives

New immutable primitive array types! Think of these as `ImmutableList` for primitive types (but not implementing `List` directly).

- `ImmutableIntArray` and `.Builder`
- `ImmutableLongArray` and `.Builder`
- `ImmutableDoubleArray` and `.Builder`

common.base

- `Stopwatch.elapsed()` : returns the elapsed time as a `java.time.Duration` .
- `Throwables.getCauseAs(Throwable, Class<X>)` : a way of getting the cause of an exception and casting it to a specific type without losing the original exception stack trace if the cause is *not* that type.

common.collect

- `Comparators.empties(First|Last)()` : `Comparator` s for `java.util.Optional` s
- `Comparators.(least|greatest)(k, Comparator)` : `Collector` s for collecting the least/greatest `k` elements (according to a `Comparator`) from a `Stream` to a `List` (faster and more memory efficient than sorting and limiting the stream).
- New `Collector` s for `Multiset` s and `Immutable(Sorted)Multiset` s that take a `ToIntFunction` to get counts for elements.
- `RangeMap.putCoalescing(Range<K>, V)` : Inserts the given range and value to the map, merging it with any connected entries with equal values.
- `Streams.forEachPair(Stream, Stream, BiConsumer)` : For-each over pairs of corresponding elements from two streams.
- `Tables.synchronizedTable(Table)`

common.graph

- All `common.graph` methods that previously accepted `Object` for nodes/edges now require the specified node/edge type. This breaks all custom implementations of `Graph` , `Network` and

`ValueGraph`.

- `ValueGraph` no longer extends `Graph`; `Graphs.equivalence()` has been deprecated in favor of the more usual `equals()` methods on each interface (now that `ValueGraph` and `Graph` no longer need to have compatible definitions).

common.hash

- `BloomFilter.approximateElementCount()`: Estimate of the number of distinct elements that have been added to the Bloom filter.
- `Hashing.md5()` and `Hashing.sha1()`: Deprecated. Will continue to exist to allow interoperation with systems requiring them.

common.io

- `CharSink.writeLines(Stream[, String])`: Methods for writing elements of a `Stream` of `CharSequence`s as lines to a `CharSink`.
- `CharSource.lines()`: Returns a `Stream<String>` (which must be closed!) of lines of text from a `CharSource`.
- `CharSource.forEachLine(Consumer)`: For-each over the lines of text in a `CharSource`.
- `Files`: Many methods that are redundant with creating a source or sink for a file and calling a method on it are now deprecated.
- `MoreFiles.equal(Path, Path)`: Returns whether or not two files are regular files containing the same bytes.

common.util.concurrent

- `CheckedFuture`: Deprecated. (See the javadoc for details.)
- `TimeLimiter`: `callWithTimeout` (taking a `boolean interruptible`) deprecated; `callWithTimeout`, `callUninterruptiblyWithTimeout`, `runWithTimeout` and `runUninterruptiblyWithTimeout` added.

Additional changes

Guava's dependencies on some annotation-only libraries are no longer `<optional>true</optional>`. This makes them available at runtime and at compile time for libraries that compile against Guava. This fixes some projects' compile errors (#2721), but it may introduce dependency conflicts into other projects (#2824). If you see dependency conflicts, we believe you should be safe excluding all but the newest version of each of the annotation libraries.