Throwables

Guava's `Throwables` utility can frequently simplify dealing with exceptions.

Propagation

Sometimes, when you catch an exception, you want to throw it back up to the next try/catch block. This is frequently the case for `RuntimeException` or `Error` instances, which do not require try/catch blocks, but can be caught by try/catch blocks when you don't mean them to.

Guava provides several utilities to simplify propagating exceptions. For example:

```
try {
  someMethodThatCouldThrowAnything();
} catch (IKnowWhatToDoWithThisException e) {
  handle(e);
} catch (Throwable t) {
  Throwables.throwIfInstanceOf(t, IOException.class);
  Throwables.throwIfInstanceOf(t, SQLException.class);
  Throwables.throwIfUnchecked(t);
  throw new RuntimeException(t);
}
```

Here are quick summaries of the propagation methods provided by Guava:

Signature	Explanation
`void propagateIfPossible(Throwable, Class<X extends Throwable>) throws X`	Throws `throwable` as-is only if it is a `RuntimeException`, an `Error`, or an `X`.
`void throwIfInstanceOf(Throwable, Class<X extends Exception>) throws X`	Propagates the throwable as-is, if and only if it is an instance of `X`.
`void throwIfUnchecked(Throwable)`	Throws `throwable` as-is only if it is a `RuntimeException` or an `Error`.

NOTE: We deprecated `Throwables.propagate(Throwable)` in v20.0. Read about why.

Causal Chain

Guava makes it somewhat simpler to study the causal chain of an exception, providing three useful methods whose signatures are self-explanatory:

- `Throwable getRootCause(Throwable)`

- `List<Throwable> getCausalChain(Throwable)`
- `String getStackTraceAsString(Throwable)`