Preconditions

Guava provides a number of precondition checking utilities in the `Preconditions` class. We strongly recommend importing these statically.

Each method has three variants:

1. No extra arguments. Any exceptions are thrown without error messages.
2. An extra `Object` argument. Any exception is thrown with the error message `object.toString()`.
3. An extra `String` argument, with an arbitrary number of additional `Object` arguments. This behaves something like printf, but for GWT compatibility and efficiency, it only allows `%s` indicators.
    - Note: `checkNotNull`, `checkArgument` and `checkState` have a large number of overloads taking combinations of primitive and `Object` parameters rather than a varargs array — this allows calls such as those above to avoid both primitive boxing and varags array allocation in the vast majority of cases.

Examples of the third variant:

```
checkArgument(i >= 0, "Argument was %s but expected nonnegative", i);
checkArgument(i < j, "Expected i < j, but %s >= %s", i, j);
```

Signature (not including extra args)	Description	Exception thrown on failure
`checkArgument(boolean)`	Checks that the `boolean` is `true`. Use for validating arguments to methods.	`IllegalArgumentException`
`checkNotNull(T)`	Checks that the value is not null. Returns the value directly, so you can use `checkNotNull(value)` inline.	`NullPointerException`
`checkState(boolean)`	Checks some state of the object, not dependent on the method arguments. For example, an `Iterator` might use this to check that `next` has been called before any call to `remove`.	`IllegalStateException`
`checkElementIndex(int index, int size)`	Checks that `index` is a valid *element* index into a list, string, or array with the specified size. An element index may range from 0 inclusive to `size` **exclusive**. You don't pass the list, string, or array directly; you just pass its size. Returns `index`.	`IndexOutOfBoundsException`
`checkPositionIndex(int index, int size)`	Checks that `index` is a valid *position* index into a list, string, or array with the specified size. A position index may range from 0 inclusive to `size` **inclusive**. You don't pass the list, string, or array directly; you just pass its size. Returns `index`.	`IndexOutOfBoundsException`
`checkPositionIndexes(int`	Checks that `start` are `end` both in the range `[0,`	`IndexOutOfBoundsException`

start, int end, int size)	size] (and that end is at least as large as start). Comes with its own error message.	

We preferred rolling our own preconditions checks over e.g. the comparable utilities from Apache Commons for a few reasons. Briefly:

- After static imports, the Guava methods are clear and unambiguous. `checkNotNull` makes it clear what is being done, and what exception will be thrown.
- `checkNotNull` returns its argument after validation, allowing simple one-liners in constructors: `this.field = checkNotNull(field);` .
- Simple, varargs "printf-style" exception messages. (This advantage is also why we recommend continuing to use `checkNotNull` over `Objects.requireNonNull`)

We recommend that you split up preconditions into distinct lines, which can help you figure out which precondition failed while debugging. Additionally, you should provide helpful error messages, which is easier when each check is on its own line.