Comparing try-with-resources to the traditional try clause

try-with-resources examples	traditional try clause

`// First Example:`
`try (FileReader reader = new FileReader(filename)) {`
`        // do Something`
`}`
 	
`// Second Example:`
`try (FileReader reader = new FileReader(filename);`
`    FileWriter writer = new FileWriter("New" + filename))`
`    // do Something`
`}`
	`FileReader reader = null;`
	`try {`
	`    reader = new FileReader(filename);`
	`} finally {`
	`    if (reader != null) {`
	`        try {`
	`            reader.close();`
	`        } catch (IOException e) {`
	`            // do Something`
	`        }`
	`    }`
	`}`

The try-with-resources takes a **colon delimited list** of resource variables.

The resources in this list must implement the AutoCloseable or the Closeable interface.

The try-with-resources statement can be used without a finally block, because all resources are automatically closed when this type of try block completes, or if it gets an exception.

Throwable

This chart shows you which classes are Checked, in blue, and Unchecked, in Reddish Brown.

The `Error` class indicates serious problems, that a reasonable application shouldn't try to catch or recover from.

I've already mentioned there are two types of Exceptions, those that subclass from RuntimeException, and those that don't.

This hierarchy becomes important for the second variation of a catch clause.