Writing Data to a File

There are a lot of reasons why you might want to write data to a file.

These include:

- **Storing user data.**
- **Logging application events to a log file.**
- **Storing configuration data.**
- **Exporting Data for Exchange of Information.**
- **Supporting Offline Usage in a File Cache.**
- **Generating file products.**

Is Writing to a File so different than reading to it?

Some of the concepts of writing to a file are naturally similar, to those of reading from a file.

You'll use similar named classes, but instead of InputStream, you'll work with an OutputStream, for example.

There's a FileWriter class, rather than a FileReader class, and so on.

Understanding buffered data becomes more important, as well as managing multiple writes, to a single file from different threads.

There are different ways to open a file for writing.

Default Open Options

All available options are found on an enum in the java.nio.file package, called StandardOpenOption.

The default options for Files.write methods are shown in this table.

Option	**Description**
CREATE	This creates a new file if it does not exist.
TRUNCATE_EXISTING	If the file already exists, and it's opened for WRITE access, then its length is truncated to 0.
WRITE	The file is opened for write access.