The Missing File That Fails Loudly

A missing test-data file is a setup failure, not an empty list. A suite that reports zero rows and prints green has checked nothing -- and that's worse than a suite that fails.

Automation Foundations · ashishautomationlabs.store

Data flows both ways. Writing a results file uses the same pattern as reading one:

```
try (PrintWriter writer = new PrintWriter(Files.newBufferedWriter(out))) {
    writer.println("test,status,ms");
    writer.println("LoginTest,PASS,240");
    writer.println("CartTest,FAIL,512");
}

Written to: testdata/results.csv
test,status,ms
LoginTest,PASS,240
CartTest,FAIL,512
```

A CSV of results can be opened in a spreadsheet by anyone, which makes it a genuinely useful thing for a suite to produce alongside its framework report.

When the file is not there

This is the case that decides whether your suite is trustworthy. Handle it explicitly:

```
try (BufferedReader reader = Files.newBufferedReader(file)) {
    System.out.println(name + " -> first line: " + reader.readLine());
} catch (NoSuchFileException e) {
    System.out.println(name + " -> NOT FOUND (looked in " + Path.of("").toAbsolutePath(
) + ")");
} catch (IOException e) {
    System.out.println(name + " -> read failed: " + e.getMessage());
}

testdata/logins.csv -> first line: username,password,expected
testdata/missing.csv -> NOT FOUND (looked in /home/claude/java-hero/code)
```

Notice what the failure message includes: the directory the program actually looked in. That single detail resolves the most common file bug in test automation, where a relative path works on your machine and fails in the pipeline because the working directory differs. The multi-catch is Chapter 9's specific-before-general rule, with `NoSuchFileException` handled separately because it has a clear, actionable message.

A data-driven suite reads its CSV, and the loader returns an empty list when the file is missing so the run "does not crash." That sounds defensive. It's not -- it's silent. Zero rows means zero tests run, and a suite that runs zero tests reports green. A team can ship on a green run that tested

nothing, and the log that would have shown it goes unread, because nobody reads logs on a green build. If the data file is missing, the setup is broken -- throw, and let the run fail with the path in the message.

The rule for a test framework: **a missing data file is a setup failure, and setup failures must stop the run loudly.** They must never be turned into an empty list. Fail fast and name the file and the working directory. The one result worse than a red suite is a green suite that checked nothing.

Files and properties cover most test data. Two formats you will certainly meet in real work need a library Java doesn't ship with -- and they're worth a word about what's genuinely known here versus assumed.