The String Methods You Will Actually Use

The page says Total: Rs 1,299.00. Your expected value is the number 1299. Cleaning a scraped value into something you can assert on is usually a short pipeline of chained calls, not one method.

Automation Foundations · ashishautomationlabs.store

The page says `Total: Rs 1,299.00`. Your expected value is the number `1299`. Between those two facts sits about a third of the work in real test automation. Almost nothing arrives ready to compare. A price is scraped with a currency symbol, a thousands separator, and two decimal places you do not care about. A name comes out of a spreadsheet with a trailing space nobody can see. A status arrives from an API in a different case than your expectation. Every one of these is a text problem, and getting them wrong produces the worst kind of test result: a failure that is not a defect.

Start with the raw material and look at what is available:

```
String scraped = "  Total: Rs 1,299.00  ";

raw          : '  Total: Rs 1,299.00  '
trim()       : 'Total: Rs 1,299.00'
length       : 18
contains Rs? : true
startsWith   : true
indexOf 'Rs' : 7
upper        : TOTAL: RS 1,299.00
```

Note the single quotes in that output. Printing a value inside quote marks is a small habit worth adopting permanently, because it makes invisible whitespace visible. `' DEF-101 '` and `'DEF-101'` look identical in a log without them, and they are not equal.

The everyday set is short. `trim()` removes leading and trailing whitespace, and `strip()` is its modern equivalent that also handles unusual space characters. `length()` counts characters. `contains`, `startsWith`, and `endsWith` answer yes-or-no questions. `indexOf` gives the position of the first match, or `-1` when there is none. `toUpperCase` and `toLowerCase` normalize case -- which, as Chapter 6 showed, is what makes a comparison survive real data. And `isEmpty()` asks whether the length is zero, while `isBlank()` also treats a string of only spaces as empty, which is usually the question you meant -- test data from spreadsheets is full of cells holding a single space, so `isBlank` is the check you usually want.

Cleaning: from scraped text to a number

Put them together and the opening problem solves itself:

```
String cleaned = scraped.trim()
                        .replace("Total:", "")
                        .replace("Rs", "")
                        .replace(",", "")
                        .trim();
double amount = Double.parseDouble(cleaned);
```

```
cleaned : '1299.00'
as number: 1299.0
rupees   : 1299
```

Two things to notice. The calls chain, because each one returns a String that the next one operates on. And the final `trim()` is not redundant -- removing `"Total:"` from the middle leaves a space behind, and the parse would fail without it. Cleaning is usually a small pipeline, not a single call.

The last step converts text to a number, which is where a test suite most often meets reality. `Integer.parseInt` and `Double.parseDouble` are strict: they accept digits and a sign, and reject everything else with the `NumberFormatException` you met in Chapter 9. The comma is exactly why the cleaning had to come first.

Every one of those calls just used quietly returned a brand-new String and left the original alone. That is not incidental -- it is a rule with real consequences, and it explains a bug you've already met.