Constants: Naming the Values You Keep Retyping

static final fixes a value's name to one place, and a final class with a private constructor tells the compiler that a constants holder is never meant to become an object.

Automation Foundations · ashishautomationlabs.store

A **magic string** is a literal value typed directly into code, with no name to explain it. `30`, `"testdata/logins.csv"`, a URL repeated nine times. Each is a small landmine: change the value and you must find every copy, and the one you miss fails quietly.

The fix is `static final`:

```
final class TestConstants {
    static final int DEFAULT_TIMEOUT_SECONDS = 30;
    static final String DATA_FILE = "testdata/logins.csv";
    private TestConstants() {
    }
}
```

```
Timeout   : 30s
Data file : testdata/logins.csv
Full URL  : https://uat.shopcart.internal
```

Three pieces are doing work. `static` means it belongs to the class, so you write `TestConstants.DEFAULT_TIMEOUT_SECONDS` without creating an object, exactly as Chapter 8 described. `final` means it cannot be reassigned -- try, and the compiler stops you. And the naming convention for constants is capitals with underscores, which is how every Java reader recognizes one on sight.

The two extra touches are worth copying. The class is `final`, so nobody can extend it, and its constructor is `private`, so nobody can instantiate it. A holder of constants and static helpers is not meant to become an object, and those two lines say so in a way the compiler enforces. The same pattern suits utility classes:

```
final class UrlUtil {
    private UrlUtil() {
    }
    static String forEnv(String env) {
        return "https://" + env + ".shopcart.internal";
    }
}
```

One warning worth keeping precise: is `static final` the same as a constant in other languages? Close enough for daily use, with one catch. `final` stops the *variable* being reassigned; it does not freeze the object it points at. A `static final List` cannot be pointed at a different list, but items can still be added to it. For real immutability, use immutable values or an unmodifiable collection.

`static final` solves repetition -- one name instead of nine copies. It does not solve a deeper problem: nothing stops that name's *value* from being any text at all, valid or not.