Five Types That Matter

Java has eight primitives. Automation work mostly needs five -- plus String for text. Four questions choose the type; int is the everyday default when every answer is no.

Automation Foundations · ashishautomationlabs.store

Java has exactly eight built-in **primitive** types -- simple, raw values with no machinery attached. Here is the complete set, sorted honestly for an automation career:

Type | Stores | You will use it...

`int` | whole numbers (~±2.1 billion) | Constantly -- counts, indexes, IDs

`long` | bigger whole numbers (literal needs `L`) | Sometimes -- timestamps, big IDs, row counts

`double` | decimals (_approximate_ -- you have seen it) | Often -- rates, measurements

`boolean` | `true` / `false` only | Constantly -- every pass/fail

`char` | one character in single quotes: `'A'` | Rarely alone; underlies `String`

`byte`, `short` | small whole numbers | Almost never -- recognize and move on

`float` | smaller decimal than `double` | Almost never -- `double` won

And there is one more type that behaves as if it belongs on that list, but does not: `String`, for text. The difference between `String` and the primitives is the deepest idea in the next lessons. For today: you use it constantly for environments, messages, IDs-as-text, and locators.

You do not have to memorize all eight. Knowing which five matter is more useful than reciting the full list.

A status report from your day job

```
public class TypesTour {
    public static void main(String[] args) {
        int testCasesPlanned = 480;
        long recordsInTestDb = 9_400_000_000L;
        double passRate = 97.5;
        boolean buildIsGreen = true;
        char severity = 'A';
        String environment = "UAT-2";
        System.out.println("Planned test cases : " + testCasesPlanned);
        System.out.println("Records in test DB : " + recordsInTestDb);
        System.out.println("Pass rate : " + passRate + "%");
        System.out.println("Build green? : " + buildIsGreen);
        System.out.println("Severity band : " + severity);
```

```
        System.out.println("Environment : " + environment);
    }
}
```

```
Planned test cases : 480
Records in test DB : 9400000000
Pass rate : 97.5%
Build green? : true
Severity band : A
Environment : UAT-2
```

Three details worth noting:

- Underscores in `9_400_000_000L` are legal visual separators -- Java ignores them;

humans thank you.

- The `L` at the end is not visual: 9.4 billion is too large for an `int` literal, so

you must mark it as `long`.

- Quotes matter. `char` takes **single** quotes for one character. `String` takes

double quotes for text of any length. So `'A'` and `"A"` are different types -- a difference the compiler enforces, and interviewers enjoy.

Choosing a type: four questions

Ask in order. The first "yes" wins:

- Is it text, or one character? -> `String` / `char`
- Is it a yes/no verdict? -> `boolean`
- Does it need decimals? -> `double`
- Can it exceed ~2.1 billion? -> `long`

When every answer is "no," `int` **is the everyday default.**

That decision tree is not memorization. It is four quick questions -- the same shape as a triage checklist for test data.

You can name the five types that earn daily use, spot `String` as text-not-primitive, and choose with four questions. The next lesson boundary-tests these types the way you boundary-test any spec -- including two traps that silently corrupt real reports.