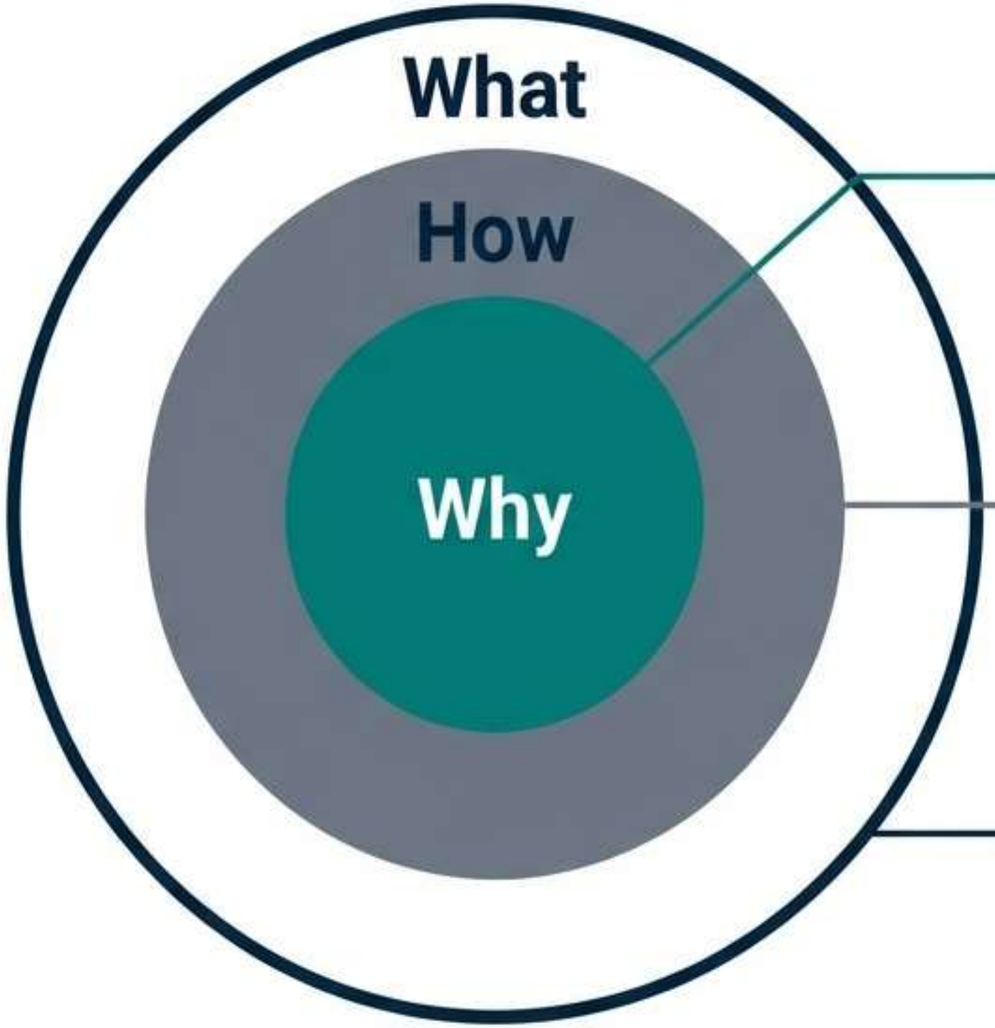
Testleaf
Always Ahead

Java Methods & Objects in Selenium Automation

Mastering Modular Code Structure and Scope Control

The Golden Circle: Understanding Understanding Methods

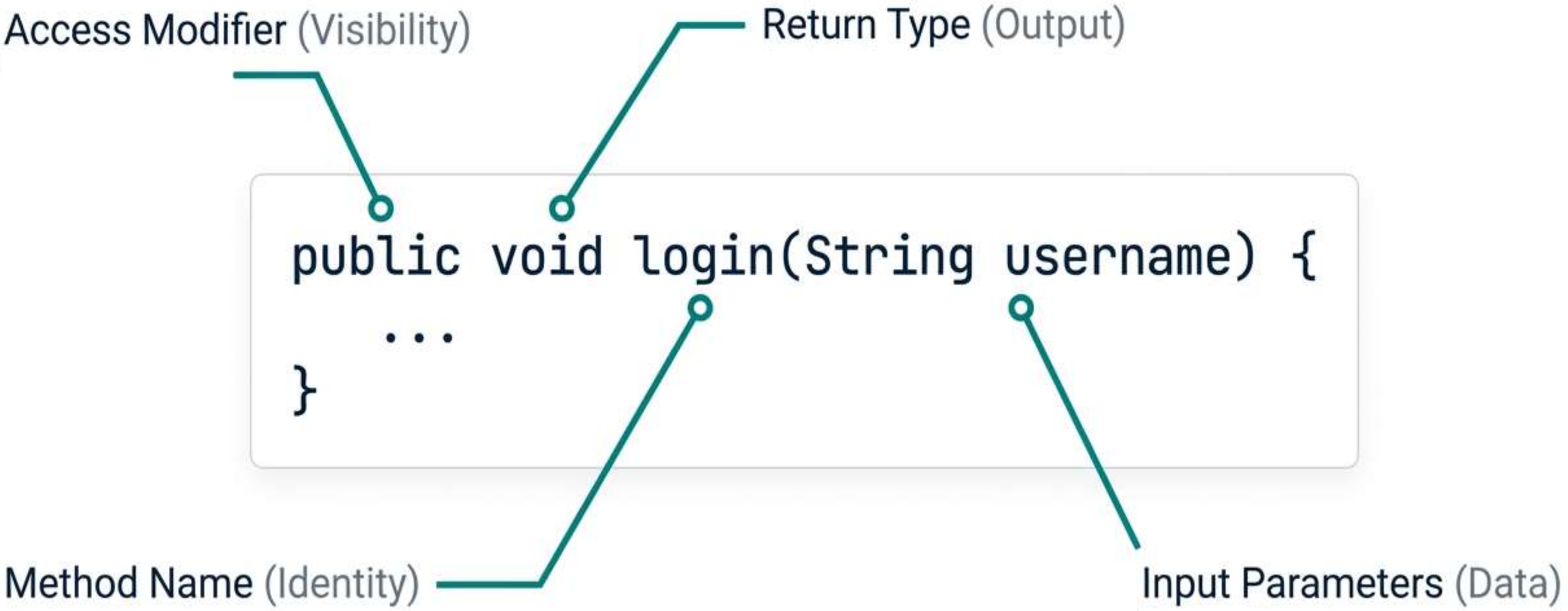


Logically groups statements into reusable functions.

Utilising selection, iteration, and jump statements.

Class members that perform specific actions.

The Anatomy of a Method



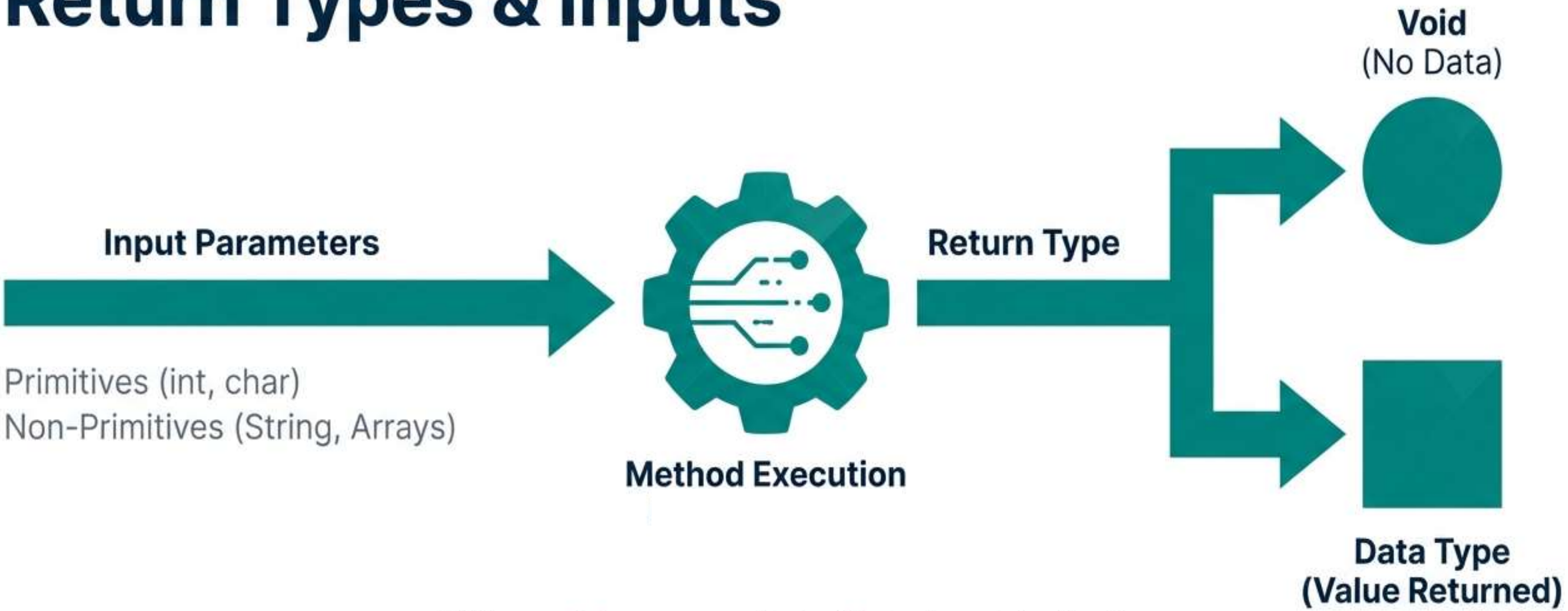
Controlling Visibility: Access Modifiers

Testleaf
Always Ahead

	Same Class	Same Package	Different Package	Different Project
Public	✓	✓	✓	✓
Protected	✓	✓	✓ Subclass only	✗
Default	✓	✓	✗	✗
Private	✓	✗	✗	✗

Testleaf
Always Ahead

Data Exchange: Return Types & Inputs

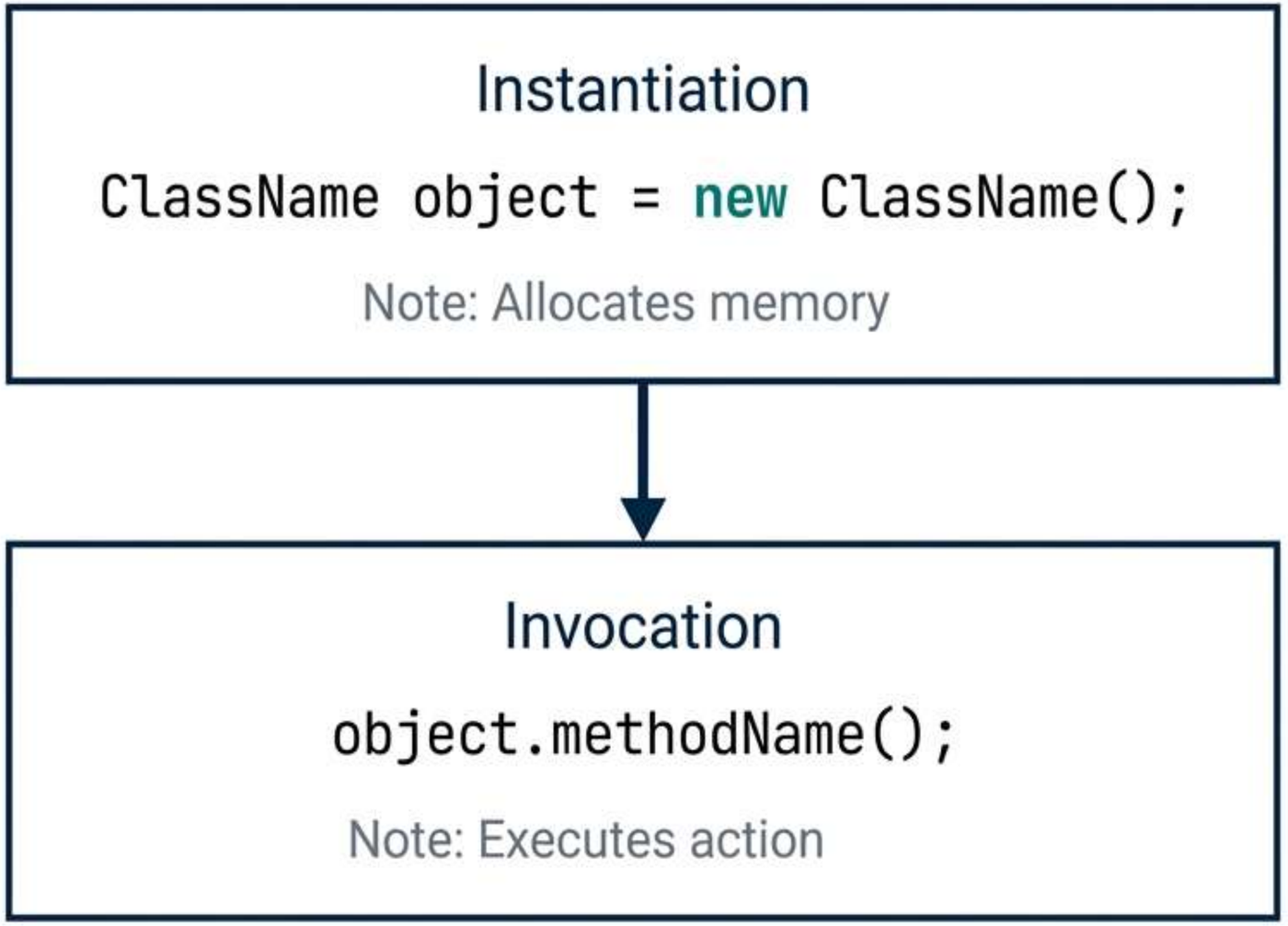



```
public void processData(int input) { …}
```

Return Type (Output)

Input Parameters (Data)

Execution Flow: Calling a Method



Applying Methods in Selenium

```java
// LoginPage.java
public class LoginPage {

    private WebDriver driver;
    private By usernameField = By.id("username");

    public void enterUsername(String uName) {
        driver.findElement(usernameField).sendKeys(uName);
    }

    // Other methods...
}
```

Public methods in Page Objects allow tests to drive browser interactions.

Optimising Method Design

Testleaf
Always Ahead

Best Practices

- **Naming Conventions**
  Meaningful Name
- **Single Responsibility**
- **Parameter Usage**
- **Scope Management**

Code Examples

use camelCase (e.g., `clickSubmit`)

One method = One specific action

Keep arguments minimal

Use `private` for internal helpers