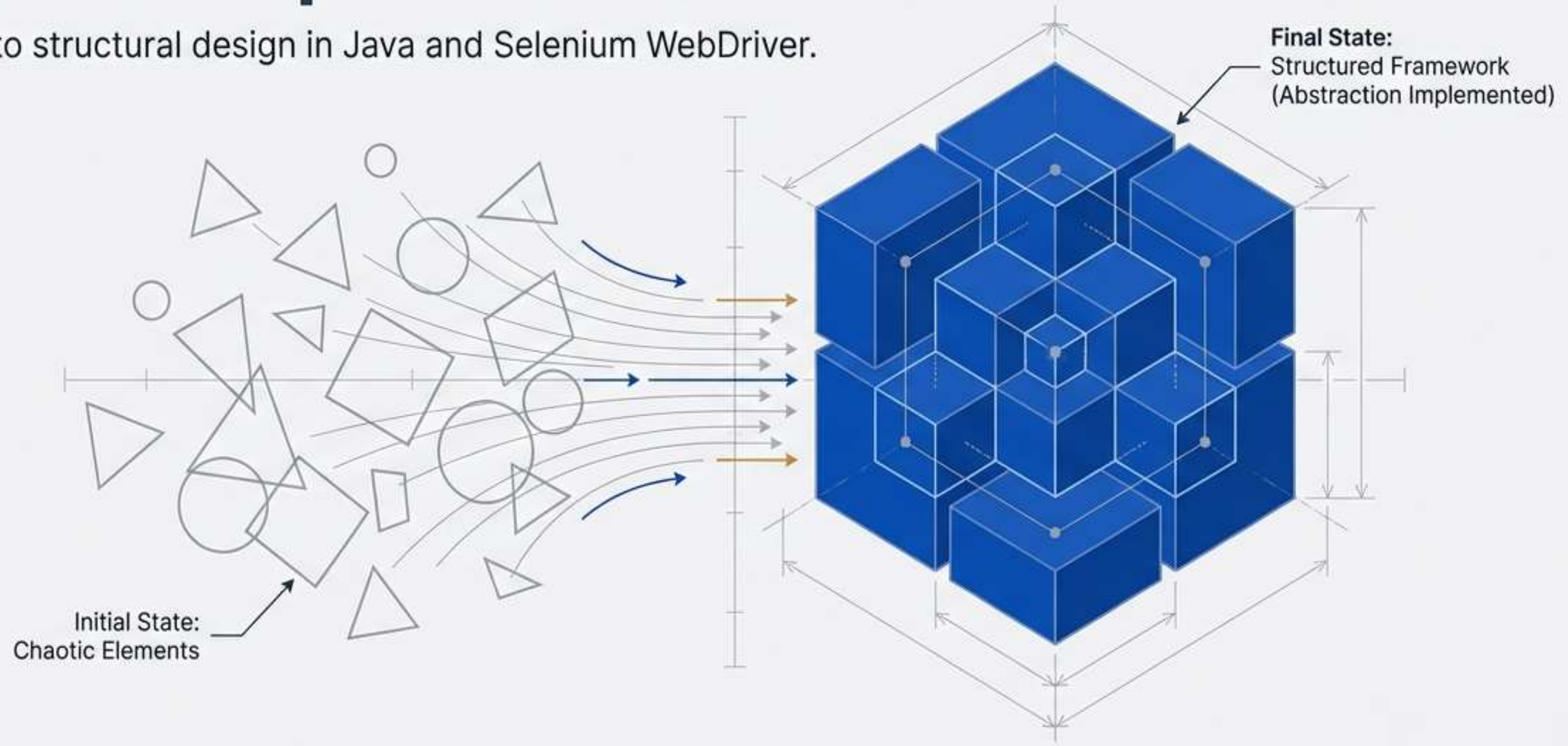
Testleaf
Always Ahead

Unlocking Abstraction: From Concept to Framework

A guide to structural design in Java and Selenium WebDriver.

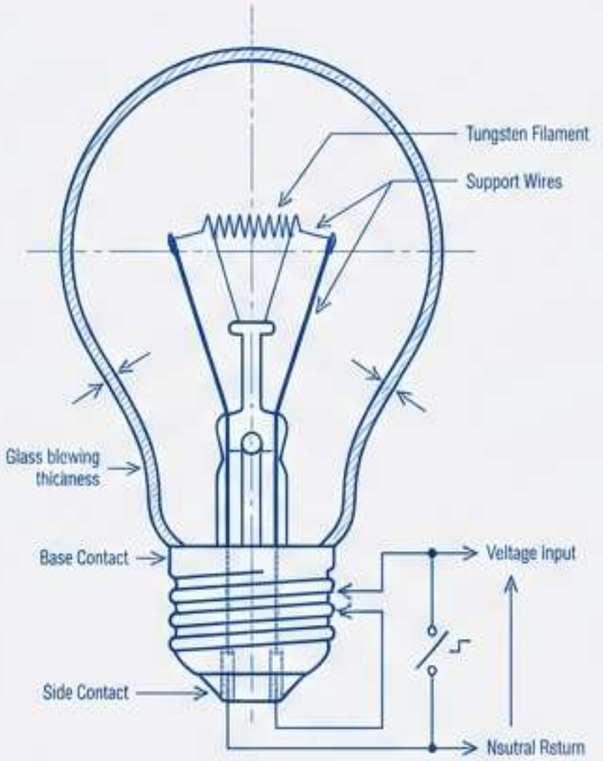


Abstraction is an Experience Choice

Testleaf
Always Ahead

The Interface (What you see)

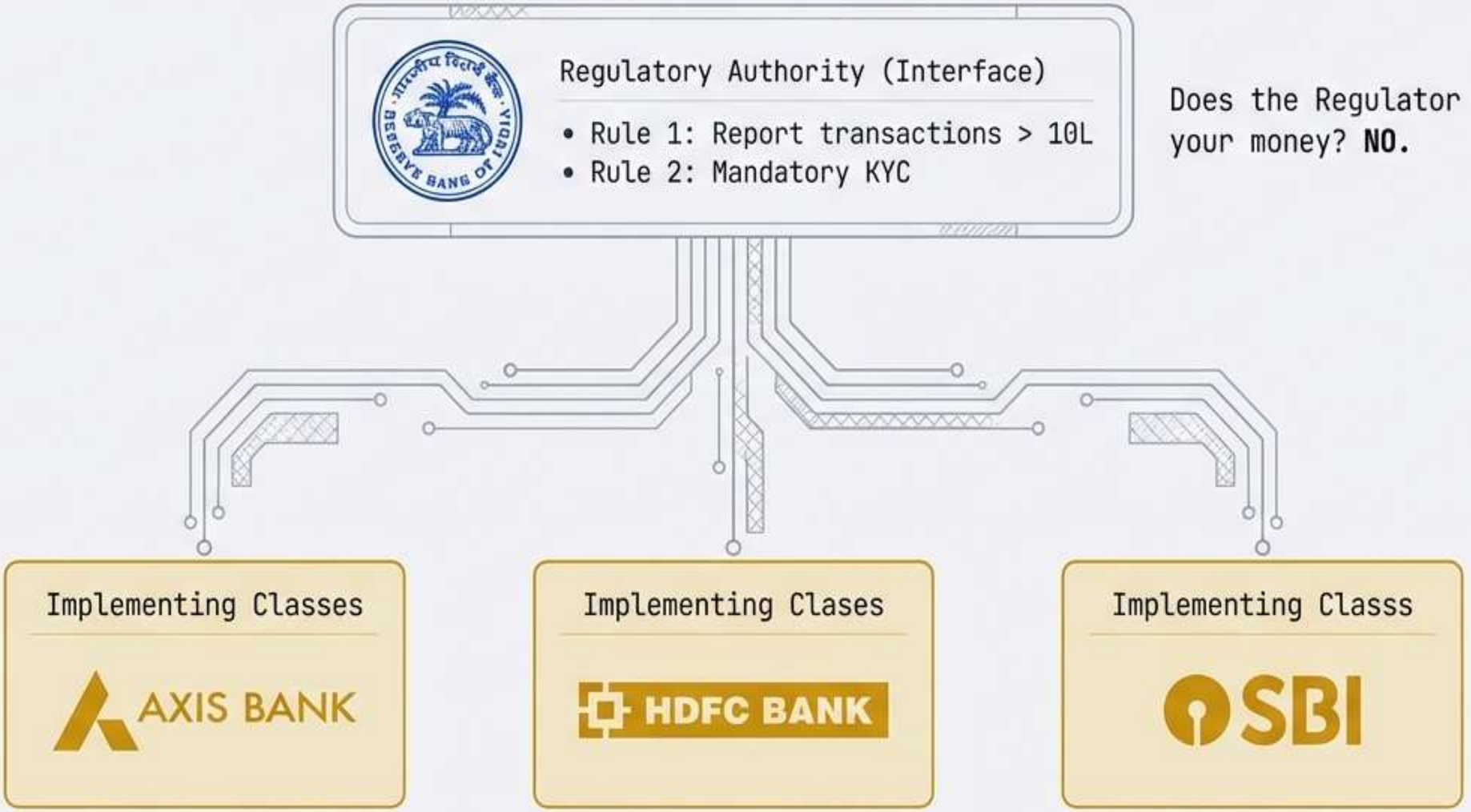
The Implementation (How it works)



The Concept: We display the utility (Light: ON/OFF) while hiding the complexity (Voltage, Filaments, Circuitry). JetBrains Mono.

The Key Takeaway: You interact with the switch, not the wiring.

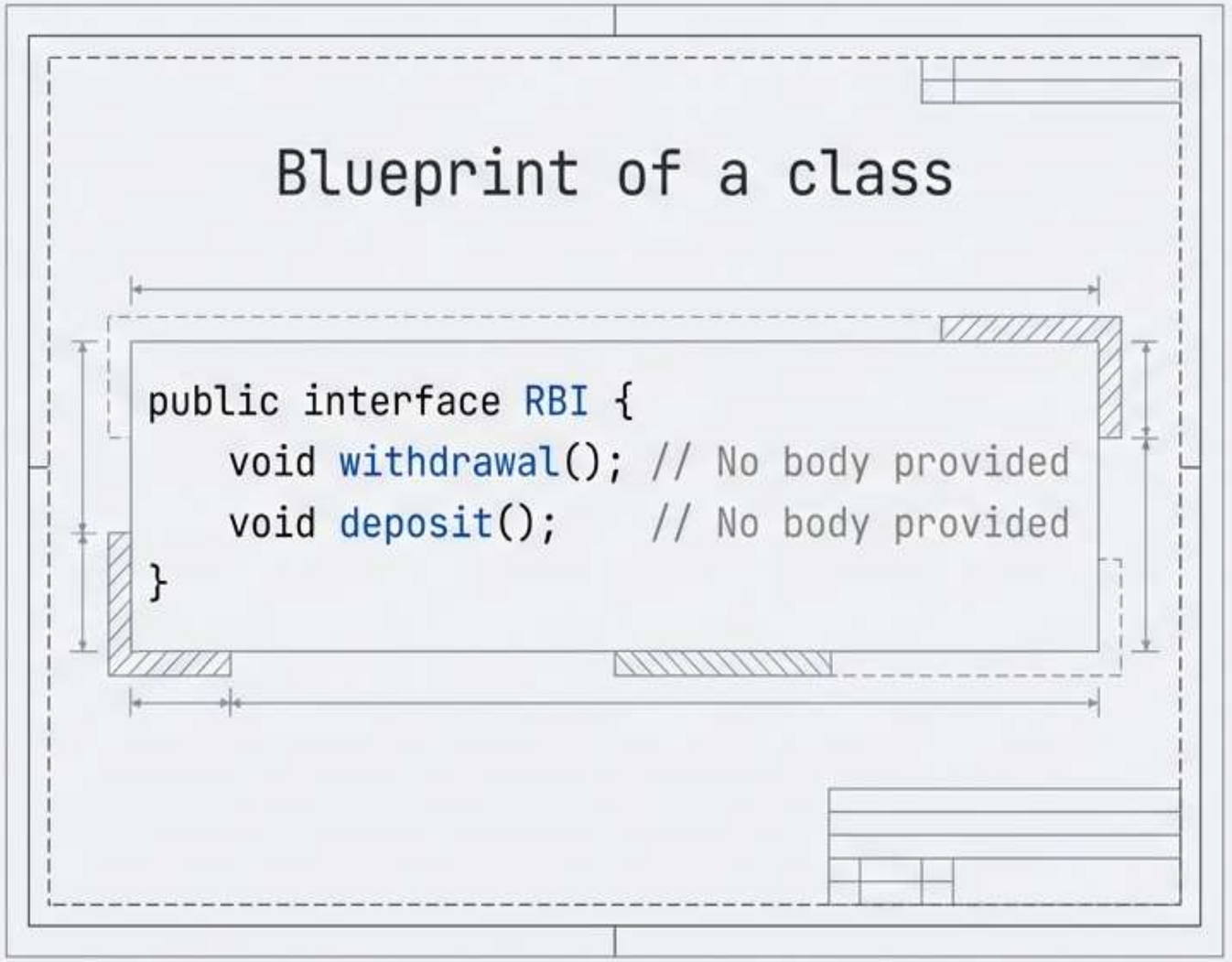
The Regulator and The Executor



Do the Banks execute the rules? **YES.**

Testleaf
Always Ahead

The Interface: The Digital Blueprint



The Definition:

An Interface is a strict contract. It defines *what* must be done, but never *how*.

The Classical View (Java 1.7):

100% Abstract methods. Pure design.

Modern Trend (Java 8+):

Note: Interfaces now support `default` and `static` methods to allow backward compatibility, but the core principle of 'Contract Design' remains unchanged.

Signing the Contract: The implements Keyword

```
public class HDFC implements RBI {

    @Override
    public void withdrawal() {
        // HDFC Specific Logic
        System.out.println("Limit: 20k");
    }
}
```

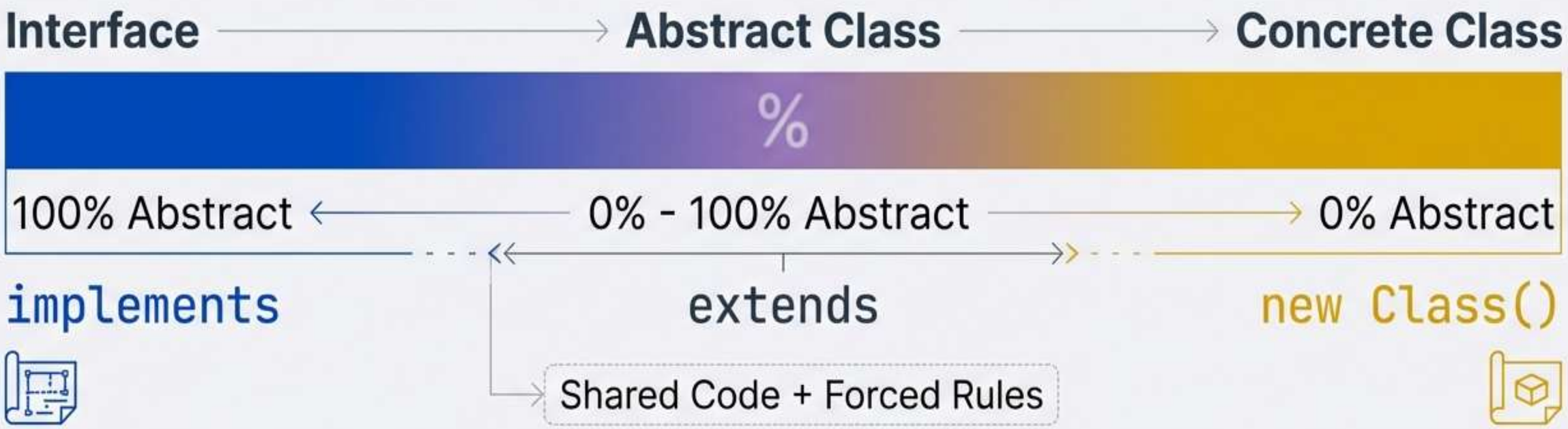
The Obligations

- [x] Implement `withdrawal()`
- [x] Implement `deposit()`
- [x] Adhere to method signature

Compiler Error

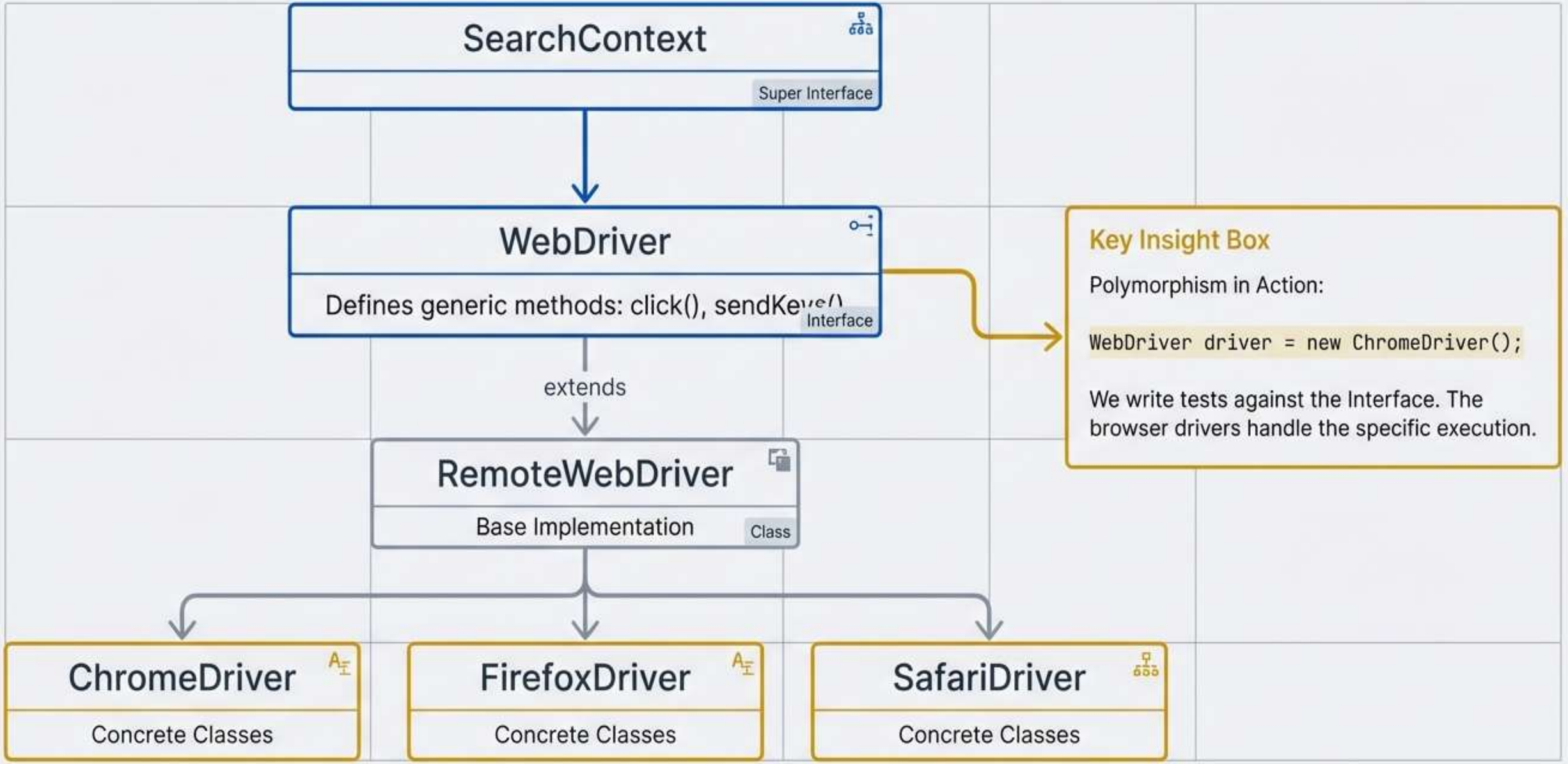
If any method from the Interface is missing, the code will not build. The contract is binding.

Finding the Middle Ground: Abstract Classes



Sometimes you need to enforce a blueprint *and* provide common tools. An Abstract Class allows you to hide implementation details (like an interface) while sharing common logic (like a class).

Abstraction in Automation: The WebDriver Architecture



The Interview Prep Kit

The Concept

What: Hiding internal details to expose utility.

Why: Reduces complexity and coupling.

How: Interfaces & Abstract Classes.

Java Ecosystem

- ∅ Interface: `List`, `Set`, `Map`
- ⬚ Abstract Class: `AbstractList`, `AbstractCollection`

Project Frameworks

- ∅ Interface: `Browser`, `Element`
- ⬚ Abstract Class: `Reporter/BaseClass`
- ▣ Concrete: `ProjectSpecificMethods`

Selenium Ecosystem

- ∅ Interface: `WebDriver`, `WebElement`, `Alert`
- ⬚ Abstract Class: `By`
- ▣ Concrete Class: `RemoteWebDriver`