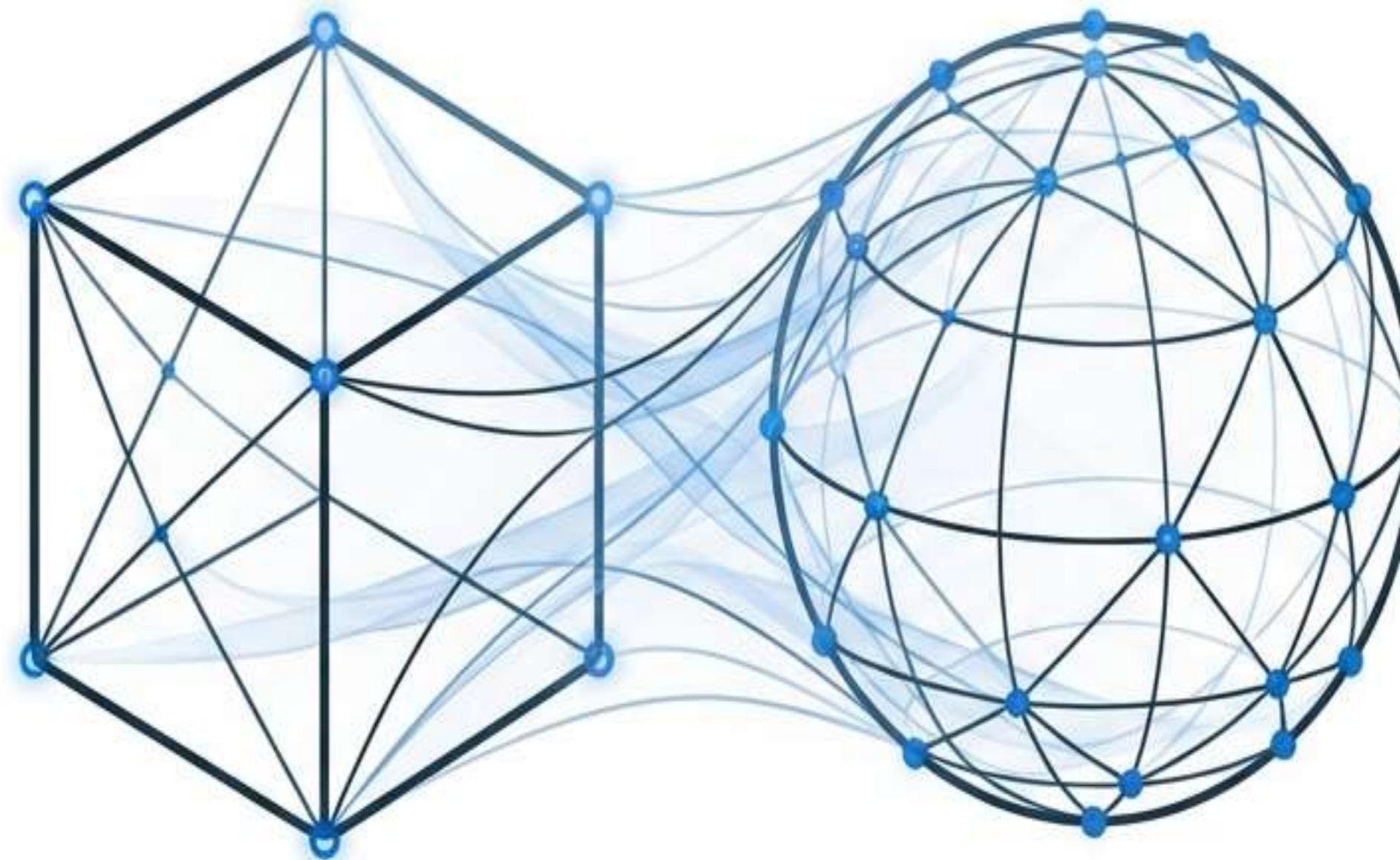


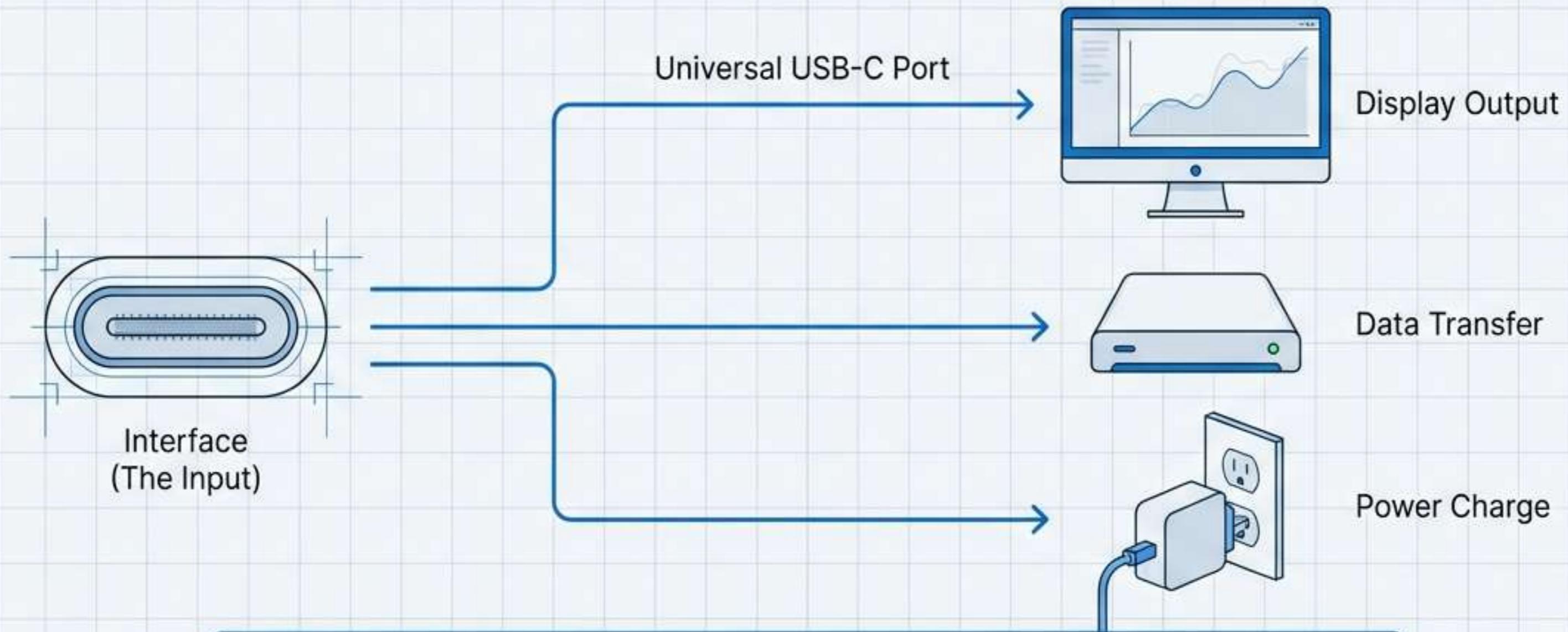
Mastering Polymorphism in Java & Selenium

Principles, Patterns, and Modern Implementation Strategies



The Concept: One Interface, Many Forms

Poly (Many) +Morphs (Forms). The ability of a message or object to be displayed in more than one form.



We perform a single action (plug in), but the result changes based on the object connected. This is the heart of polymorphism.

The Landscape of Polymorphism

Polymorphism

Compile-time / Early Binding

Method Overloading

Readability & Convenience

Run-time / Late Binding

Method Overriding

Architectural Flexibility

Static Polymorphism: Method Overloading

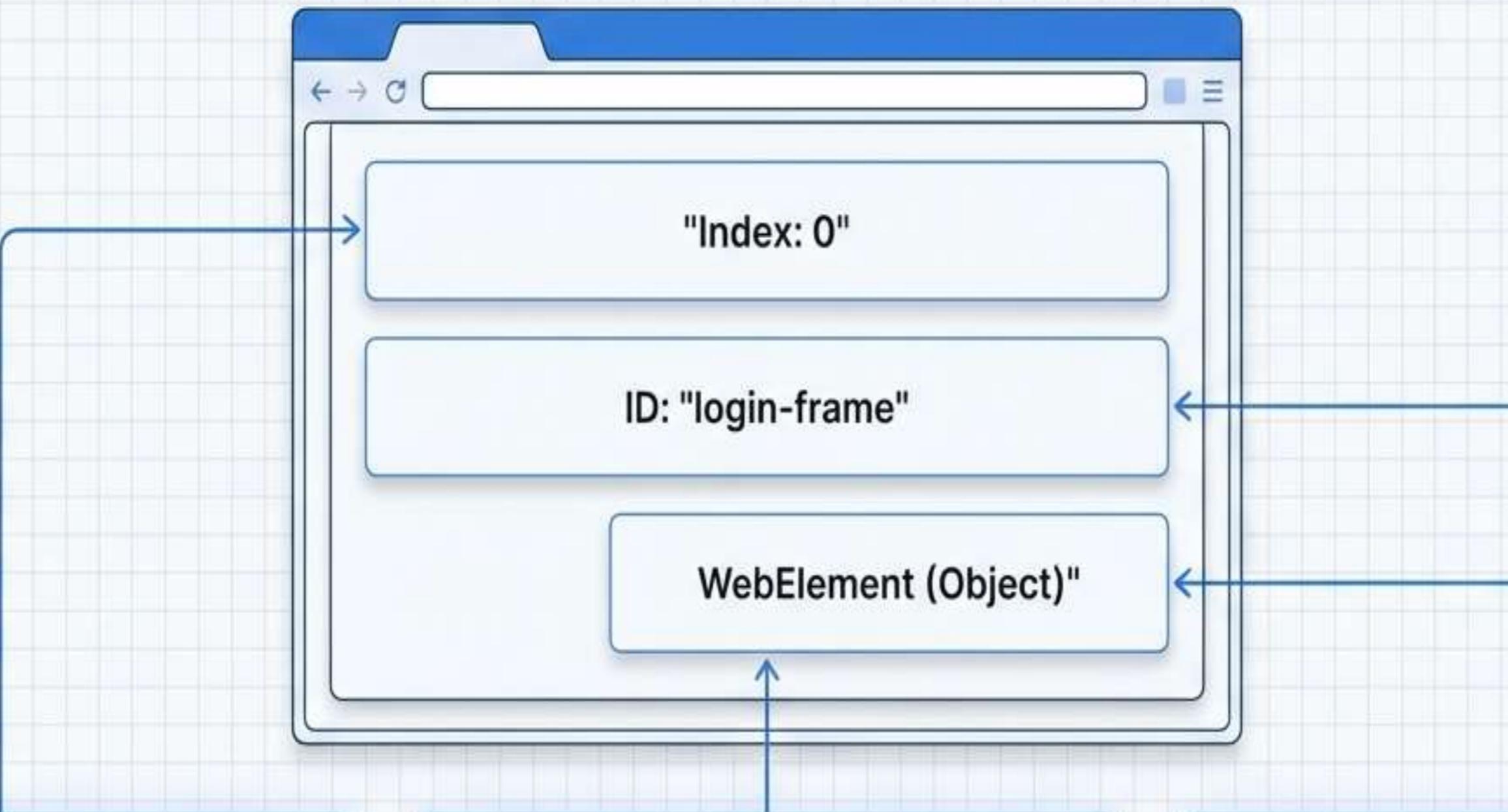
Core Rule: Same class, same method name, different signature.

- Condition A: Different number of arguments.
- Condition B: Different type of arguments.
- Purpose: Simplicity. Prevents API clutter (e.g., avoiding 'clickWithWait', 'clickWithJS').

ElementUtils.java

```
public class ElementUtils {  
  
    // Standard click  
    public void click() {  
        element.click();  
    }  
  
    // Overloaded: Wait then click  
    public void click(long timeOut) {  
        wait(timeOut);  
        element.click();  
    }  
  
    // Overloaded: JavaScript click  
    public void click(String jsScript) {  
        jsExecutor.executeScript(jsScript);  
    }  
}
```

Selenium in Practice: Overloading frame()



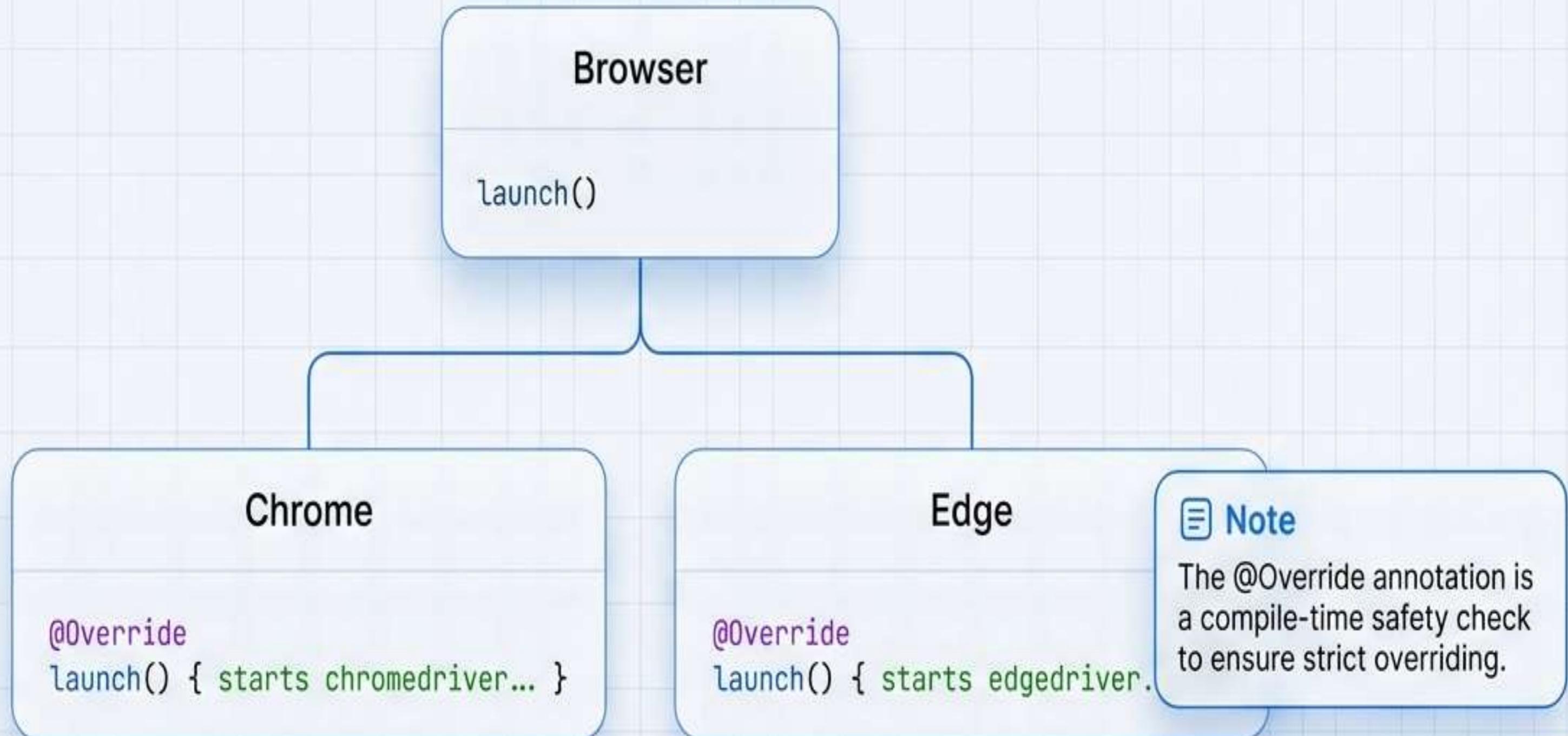
```
driver.switchTo().frame(int  
index);
```

```
driver.switchTo().frame(String  
nameOrId);
```

```
driver.switchTo().frame(  
WebElement frameElement);
```

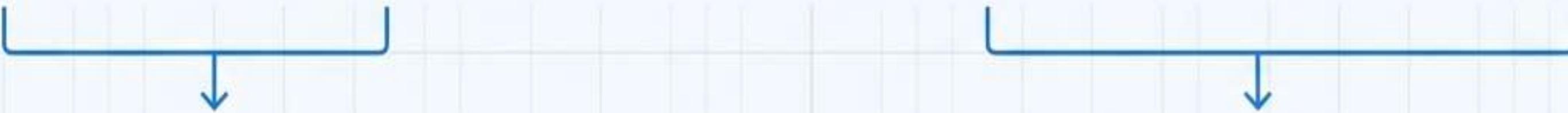
Dynamic Polymorphism: Method Overriding

Based on IS-A Relationship (Inheritance)



The Golden Rule of Selenium Architecture

```
WebDriver driver = new ChromeDriver();
```



Interface
(Parent Reference Type)

Implementation
(Child Object Type)

This is Dynamic Polymorphism. By writing our tests against the WebDriver interface, we can swap new ChromeDriver() with new FirefoxDriver() and the entire test suite runs on a different browser without changing a single line of logic.

Interview Prep Kit: Overloading

Ambiguity Check

Can we overload a method based only on the return type?

No.

The compiler checks arguments only. If signatures are identical, the return type is irrelevant to ambiguity.

The Main Method

Can we overload the main method?

Yes.

You can have multiple main methods, but the JVM will only look for and execute public static void main(String[] args).

Interview Prep Kit: Overriding

Static Methods

Can we override static methods?

No.

This is called 'Method Hiding.' Static methods belong to the class, not the object instance.

Visibility Constraints

Can we reduce the visibility of an overridden method?

No.

If the parent method is public, the child cannot be private. You can only maintain or increase visibility.

Summary: The Polymorphism Mind Map

