



OOPs Concepts Explained Clearly (Selenium Usage)

1 Abstraction

Abstraction is the process of **hiding complex implementation details** and showing **only the essential features** to the user.

The main goal of abstraction is to **reduce complexity, improve readability, and make systems easier to use and maintain.**

In simple words:

👉 You know what to do, but you don't need to know how it is done internally.

Why abstraction is important:

- Makes code simple to understand
- Reduces dependency on implementation
- Allows changes without affecting test cases

Real-life example:

When you drive a car, you use **steering, accelerator, brake**.

You don't know how the engine or gearbox works internally.

Selenium usage:

In Selenium frameworks, we use **Page Object Model (POM)** to achieve abstraction.

Tests only call:

`loginPage.login(username, password);`

They don't know:

- XPath
- WebDriver logic
- Wait conditions

👉 Test cases focus on **business flow**, not Selenium code.

2 Encapsulation

Encapsulation is the practice of **wrapping data (variables) and methods (functions)** into a single unit (class) and **restricting direct access** to them.

It protects the internal state of an object from being changed accidentally.

In simple words:

👉 Data is safe inside the class and can be accessed only through controlled methods.

Why encapsulation is important:

- Improves security
- Prevents misuse of data
- Makes code easier to maintain

Real-life example:

ATM machine — you can withdraw money only by entering a PIN.

You cannot directly access the cash inside.

Selenium usage:

Web elements are declared **private** inside page classes.

```
private WebElement username;  
private WebElement password;
```

Only public methods like `login()` are allowed.

If locator changes, we update it in **one place** — tests remain untouched.

3 Inheritance

Inheritance allows one class to **reuse the properties and behavior** of another class.

It supports **code reusability** and helps avoid duplication.

In simple words:

👉 Common functionality is written once and shared everywhere.

Why inheritance is important:

- Reduces duplicate code
- Improves consistency
- Easier framework scaling

Real-life example:

All employees follow **same company rules**, but roles differ.

Selenium usage:

A `BaseTest` class contains:

- WebDriver initialization
- Browser setup
- Tear down

All test classes extend it:

```
public class LoginTest extends BaseTest
```

Every test automatically gets driver access.

4 Polymorphism

Polymorphism means **one method name having multiple forms**.

The same action behaves differently based on input or context.

There are two types:

- **Method Overloading** (same method, different parameters)
- **Method Overriding** (same method, different implementation)

Why polymorphism is important:

- Improves flexibility
- Makes code extensible
- Supports framework customization

Real-life example:

The word “Open”:

- Open a door
- Open a book
- Open an app

Same word, different actions.

Selenium usage:

```
login(user);  
login(user, role);
```

Or overriding base methods for:

- Different browsers
 - Custom reporting
 - Environment-specific behavior
-

Simple Framework Flow

`BaseTest` → initializes driver



`Page Classes` → contain locators & actions



`Test Classes` → call only business methods

- ✓ Clean framework
 - ✓ Easy maintenance
 - ✓ Interview-ready explanation
-

Interview Tip

If you **explain OOPs like this**, interviewers know:

- You didn't just learn theory
- You **used it practically**

This is the difference between:

✗ “I know OOPs”

✓ “I build frameworks using OOPs”