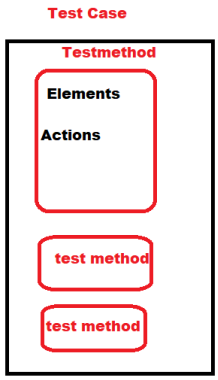
Page Object Model (POM) in Playwright

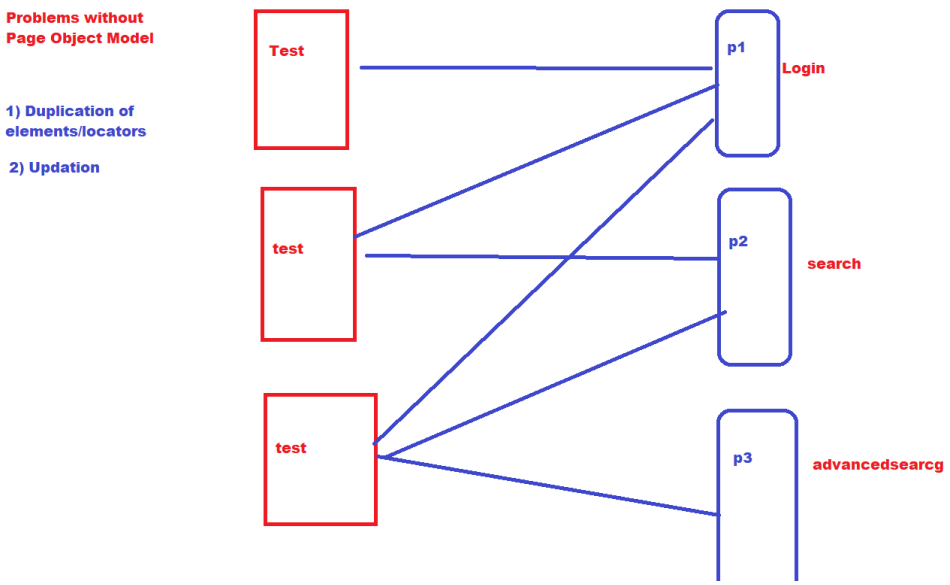
Page Object Model (POM) is a **design pattern** in test automation that helps in creating **maintainable, reusable, and readable** test code by separating the test logic from the **UI interactions**.

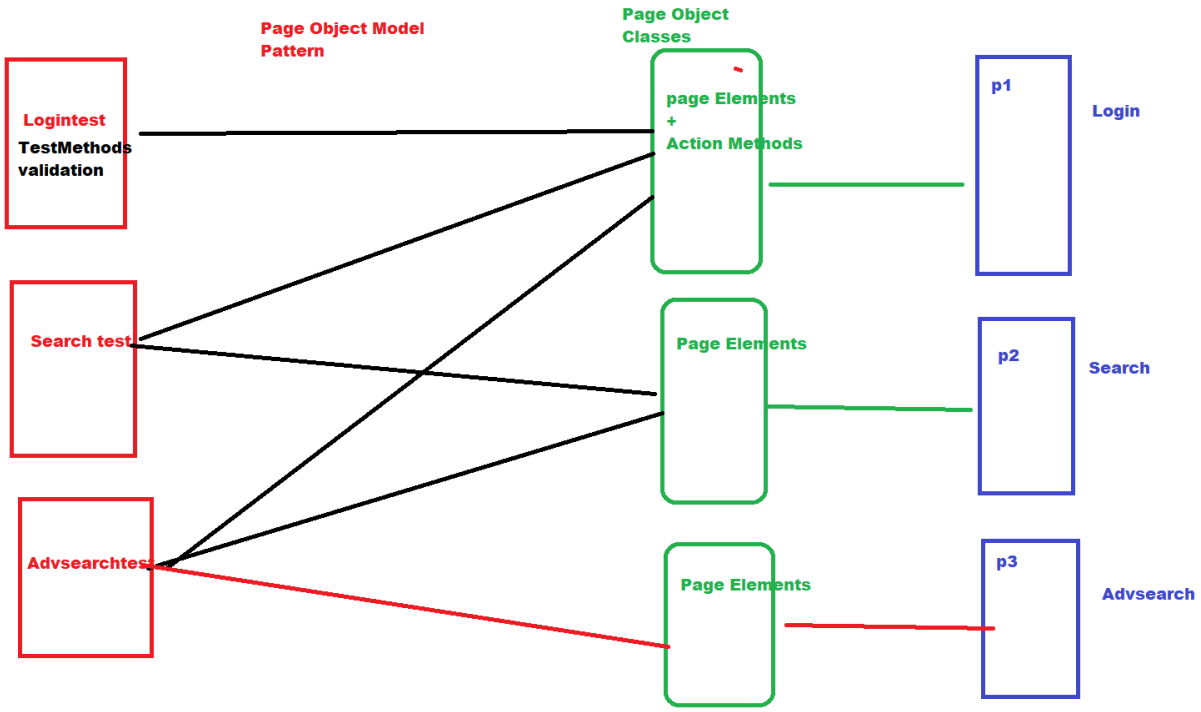
- Each web page is represented as a **class**.
- The **elements** on the page are represented as **variables**.
- The **actions** on the elements (like click, fill, etc.) are written as **methods**.

Why Use POM in Playwright?

- **Code reusability**: Page functions can be reused across test files.
- **Easy maintenance**: If UI changes, only the corresponding page file needs an update.
- **Better readability**: Clear separation of test logic and page structure.
- **Scalability**: Ideal for large-scale projects with many test cases.







Example: Login Page with POM

LoginPage.ts

```
import { Page, Locator } from '@playwright/test';
export class LoginPage {
  readonly page: Page;
  readonly usernameInput: Locator;
  readonly passwordInput: Locator;
  readonly loginButton: Locator;

  constructor(page: Page) {
    this.page = page;
    this.usernameInput = page.locator('#username');
    this.passwordInput = page.locator('#password');
    this.loginButton = page.locator('#login');
  }

  async goto() {
    await this.page.goto('https://example.com/login');
  }

  async login(username: string, password: string) {
    await this.usernameInput.fill(username);
    await this.passwordInput.fill(password);
    await this.loginButton.click();
  }
}
```

login.spec.ts

```
import { test, expect } from '@playwright/test';
import { LoginPage } from '../pages/LoginPage';

test('Login with valid credentials', async ({ page }) => {
  const loginPage = new LoginPage(page);
  await loginPage.goto();
  await loginPage.login('pavan123', 'securePassword');

  await expect(page).toHaveURL('https://example.com/dashboard');
});
```