Your framework demonstrates the use of several Object-Oriented Programming (OOP) concepts. Below is an analysis of the OOP concepts used in your framework:

1. **Encapsulation**
   * **Definition:** Encapsulation is the bundling of data (attributes) and methods (behaviors) that operate on the data into a single unit (class). It also restricts direct access to some of an object's components, which is a way of preventing unintended interference and misuse of the data.
   * **Examples in Your Framework:**
     + **Private Fields:** Many classes in your framework use private fields (e.g., private WebDriver driver; in LoginPage). These fields are accessed and modified only through public methods (getters/setters or other methods).
     + **Getter Methods:** For example, getDriver() in UIBaseTest provides controlled access to the WebDriver instance.
     + **Utility Classes:** Classes like ConfigManager and ConfigReader encapsulate configuration-related logic and expose only necessary methods (e.g., getBrowser(), getBaseUrl()).
2. **Abstraction**
   * **Definition:** Abstraction is the concept of hiding complex implementation details and exposing only the necessary features or functionalities.
   * **Examples in Your Framework:**
     + **Abstracting WebDriver Initialization:** The BrowserManager class abstracts the logic of initializing different browsers (Chrome, Firefox, Edge). Users of this class don't need to know the internal implementation details.
     + **WaitManager:** This class abstracts the complexity of handling explicit and implicit waits. Users only need to call methods like waitForVisibility() or waitForElementToBeClickable() without worrying about the underlying WebDriverWait implementation.
     + **CommonMethods:** This class abstracts common Selenium actions (e.g., moveToElementAndClick(), isElementFullyVisible()) into reusable methods.
3. **Inheritance**
   * **Definition:** Inheritance allows a class to inherit properties and behaviors (methods and fields) from another class. It promotes code reusability and establishes a relationship between the parent (superclass) and child (subclass) classes.
   * **Examples in Your Framework:**
     + **UIBaseTest as a Base Class:** The UIBaseTest class serves as a base class for test classes like AdminPageTest. It provides common setup and teardown methods (@BeforeMethod, @AfterMethod) that are reused across test classes.
     + **FrameworkException:** This custom exception class extends RuntimeException, inheriting its behavior while adding custom functionality.
4. **Polymorphism**
   * **Definition:** Polymorphism allows objects of different classes to be treated as objects of a common superclass. It can be achieved through method overriding (runtime polymorphism) or method overloading (compile-time polymorphism).
   * **Examples in Your Framework:**
     + **Method Overloading:** The switchToFrame() method in CommonMethods is overloaded to accept different parameters (int, String, WebElement).
     + **ThreadLocal with WebDriver:** The ThreadLocal<WebDriver> in UIBaseTest allows different threads to have their own instances of WebDriver, demonstrating runtime polymorphism.
     + **Custom Exceptions:** The FrameworkException class can be used polymorphically wherever a RuntimeException is expected.
5. **Composition**
   * **Definition:** Composition is a design principle where a class is composed of one or more objects of other classes. It represents a "has-a" relationship.
   * **Examples in Your Framework:**
     + **PageFactory:** The PageFactory class composes multiple page objects (e.g., LoginPage, DashBoardPage). It initializes and provides access to these page objects.
     + **CommonMethods:** This class uses composition by leveraging WebDriver, Actions, and JavascriptExecutor to perform actions on web elements.
6. **Singleton Pattern (Design Pattern, but related to OOP)**
   * **Definition:** The Singleton pattern ensures that a class has only one instance and provides a global point of access to it.
   * **Examples in Your Framework:**
     + **ConfigReader:** The ConfigReader class uses a static block to load properties only once, ensuring a single instance of the configuration is used throughout the framework.
     + **ThreadLocal in UIBaseTest:** While not a strict Singleton, the ThreadLocal<WebDriver> ensures that each thread has a single instance of WebDriver.
7. **Exception Handling (Custom Exceptions)**
   * **Definition:** Custom exceptions are used to handle specific error scenarios in a more controlled and meaningful way.
   * **Examples in Your Framework:**
     + **FrameworkException:** This custom exception is used to handle framework-specific errors (e.g., invalid environment configuration, WebDriver initialization failures).
8. **Modularity and Reusability**
   * **Definition:** Modularity refers to dividing the framework into smaller, independent, and reusable components.
   * **Examples in Your Framework:**
     + **CommonMethods:** Provides reusable methods for common Selenium actions.
     + **WaitManager:** Encapsulates wait-related functionality, making it reusable across the framework.
     + **PageFactory:** Centralizes the creation and management of page objects, promoting reusability.
9. **Thread Safety**
   * **Definition:** Thread safety ensures that shared resources are accessed in a way that prevents race conditions and inconsistent states in multi-threaded environments.
   * **Examples in Your Framework:**
     + **ThreadLocal:** Ensures each thread has its own instance of WebDriver, preventing conflicts during parallel test execution.
     + **Synchronized Blocks:** Used in ConfigReader to ensure thread-safe loading of properties.

**Summary of OOP Concepts Used:**

|  |  |
| --- | --- |
| **OOP Concept** | **Examples in Framework** |
| **Encapsulation** | Private fields, getter methods, utility classes like ConfigManager. |
| **Abstraction** | BrowserManager, WaitManager, CommonMethods. |
| **Inheritance** | UIBaseTest as a base class, FrameworkException extending RuntimeException. |
| **Polymorphism** | Overloaded methods (switchToFrame), custom exceptions, ThreadLocal with WebDriver. |
| **Composition** | PageFactory composing page objects, CommonMethods using WebDriver and Actions. |
| **Singleton Pattern** | ConfigReader loading properties once, ThreadLocal for WebDriver. |
| **Exception Handling** | Custom FrameworkException for framework-specific errors. |
| **Modularity** | Reusable components like CommonMethods, WaitManager, and PageFactory. |
| **Thread Safety** | ThreadLocal for WebDriver, synchronized blocks in ConfigReader. |

**Interview Perspective:**

**Strengths:**

* Your framework demonstrates a strong understanding of OOP principles, especially encapsulation, abstraction, and modularity.
* The use of design patterns like Singleton and ThreadLocal for thread safety is a plus.

**Areas to Highlight:**

* How you ensured thread safety in a parallel execution environment.
* How you achieved reusability through abstraction and composition.
* How you handled exceptions and custom error scenarios.

**Potential Improvements:**

* Consider using interfaces or abstract classes for further abstraction (e.g., a base page class for common page methods).
* Explore dependency injection for better testability and flexibility.

This framework is well-structured and adheres to OOP principles, making it maintainable, scalable, and suitable for interview discussions.