Page Object Model – OOPS is primary requirement to implement the Page Object Model – Specially Inheritance and Encapsulation

When relation is of type – “is a” = Inheritance is in picture, for example “Audi is a Car” – Audi inherits all the functions of the Car object. You can create an object of Audi car and can use the object to call all the features of the Car object. Car is Parent and Audi is child.

When relation is of type – “Has a” – Encapsulation is defined. Object within another object. You can instantiate an object inside another object and call the functions of the instantiated object in that object.

Challenges in developing a Page Object Model – Without Page Factory

1. Driver must be shared across the page classes – Solution Page Factory Design Pattern. This can be resolved by using the page Factory design pattern. Like usage of PageFactory.initElements(driver, Object.class). Create a constructor of the object and pass the driver. Define the WebDriver driver in the base class and extend all the test classes to baseclass.
2. Common functions for the pages – All common functions must be defined in the baseclass. Each test class should be extended to baseclass to call the common functions.
3. One action taking to multiple pages (multiple return types of a common action) – you can return either objects based on condition. return type defined is that of type “Object”. “Object” is super class of all classes. Once a function returns either object, catch that returned Object under Super class “Object”. Later based on what object is returned, you can cast it to the object of reference and the start calling the functions under that object.
4. Common objects – like similar menu bar – call constructor super class and pass the driver. Create another package for commonpages. Any class under the commonpages should have a “Has a “ relationship with basepage. Define the encapsulated object under the basepage(import that class in basepage) and create variable to refer to that class. Using that definition call the common class functions from other classes , since all other classes anyways extending the basepage.
5. Building Base Test Class – Initialize function , to have all the browser initialization functions. Provide Extent Manager definitions in the base test case class
6. Remove hardcoding – Create another package for the project with name util. Define all the constants in this class using public static – So that it can be accessed from anywhere and can be utilized by just using the class name
7. Design – Accommodate Grid – Basic motive is that test cases should be designed in such a way that they are independent of each other so that running them parallelly on grid can be implemented.
8. Implementing Reporting – Extent Report. Provide Extent Manager definitions in the base test case class.
9. Logging in the Page Classes - Build logging so that they can be called from the class files not just from the testcases.
10. Screenshots and adding to the report –
11. Reading data from the excel -

Public – It can be accessed from any where

Static – it can be accessed directly using the class name