Singleton Design pattern - Properties

Download this video

- Creational design pattern
- Only one instance of the class should exist
- Other classes should be able to get Instance of Singleton class
- Used in Logging, Cache, Session, Drivers

Subscribe

Singleton Design pattern - Implementation

Download this video

- Constructor should be private
- Public method for returning instance
- Instance type – private static

Initialisation Type:

- Eager Initialisation
- Lazy Initialisation
- Thread safe Method Initialisation
- Thread safe block Initialisation

Subscribe

Factory Design pattern - Properties

- Creational design pattern
- Used when we have multiple sub-classes of a Super class & based on input we want to return one class instance
- It provides abstraction between implementation & client classes
- Remove the instantiation of client classes from client code

Subscribe

Factory Design pattern - Implementation

- Super class can be interface or abstract class or basic class
- Factory class has a static method which returns the instance of Sub-class based on input

Subscribe

Builder Design pattern - Properties

- Creational design pattern
- Used when we have too many arguments to send in Constructor & it's hard to maintain the order.
- When we don't want to send all parameters in Object initialisation

(Generally we send optional parameters as Null)

Subscribe

Builder Design pattern - Implementation

- We create a ‘static nested class’, which contains all arguments of outer class.
- As per naming convention, If class name is ‘Vehicle’, builder class should be ‘VehicleBuilder’
- Builder class have a public constructor with all required parameters.
- Builder class should have methods for optional parameters. Method will return the Builder object.
- A ‘build()’ method that will return the final Object.

Subscribe

Builder Design pattern - Implementation

- The main class ‘Vehicle’ has private constructor so to create instance only via Builder class.
- The main class ‘Vehicle’ has only getters.

Subscribe

FlyWeight Design pattern - Properties

- Structural design pattern
- Used when we need to create many Object of a class. We use it to reduce creation of Object.
- Intrinsic Properties: Which are same for a Object.
- Extrinsic Properties: Which are different for a Object.

Subscribe

FlyWeight Design pattern - Implementation

- Interface: Which contain common method: Employee
- Object: Individual Class: Developer, Tester
- Intrinsic Property (Developer: Fix the issue, Tester: Test the issue)
- Extrinsic Property: Skills
- We use Factory to use return Object: EmployeeFactory
- Client: Client class

- We'll assign issues as per skills

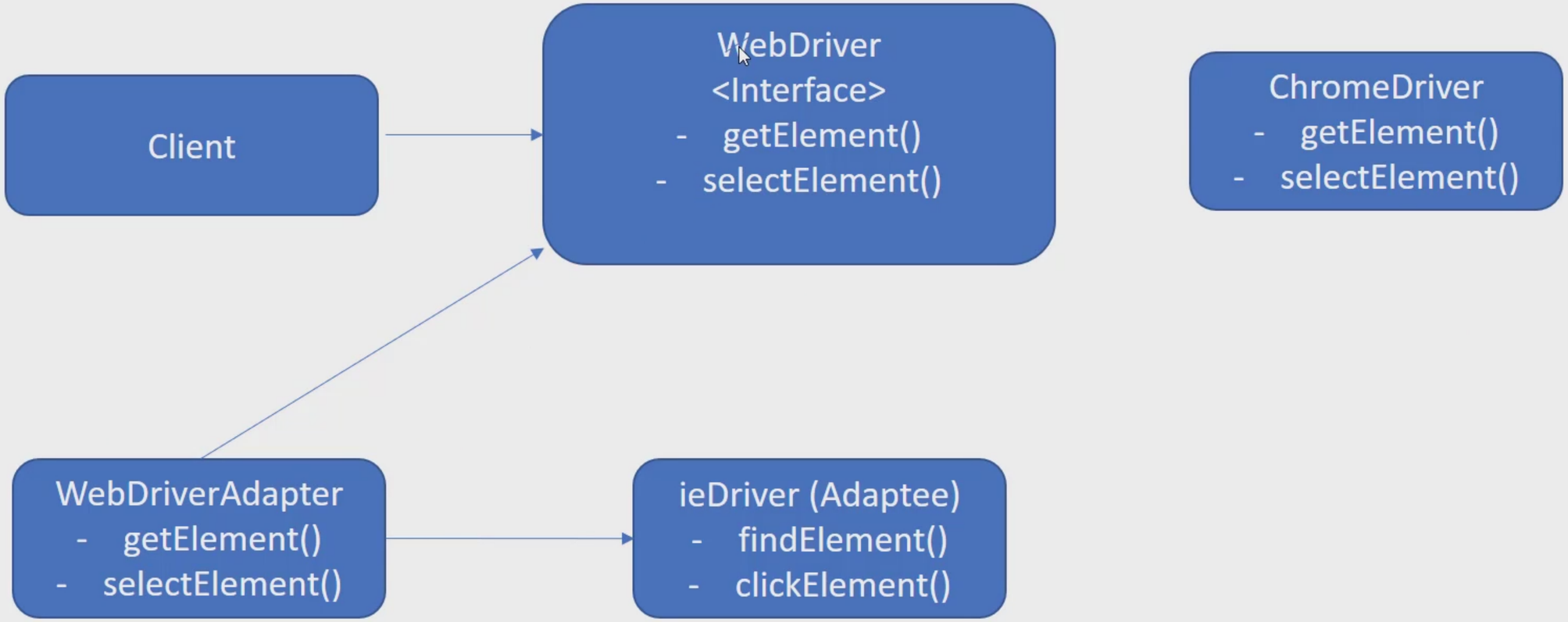
Subscribe

Adapter Design pattern - Properties

- Structural design pattern
- When objects offering same features, but has different interface. i.e Charging adapter, USB to Ethernet Adapter
- It allows exiting classes to be used with others without modifying their source code.
- i.e: WebDriver Adapter

Subscribe

Adapter Design pattern - Properties



Adapter Design pattern - Implementation

- Interface: WebDriver
- Interface Implementation: ChromeDriver, WebDriverAdaptor
- Adapter: WebDriverAdapter
- Adaptee: ieDriver
- Client: AdapterPatternTest

Subscribe