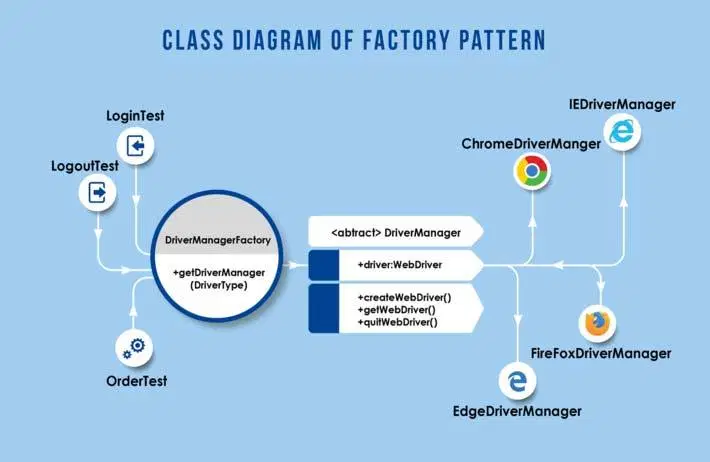
**Browser Driver Factory**

****

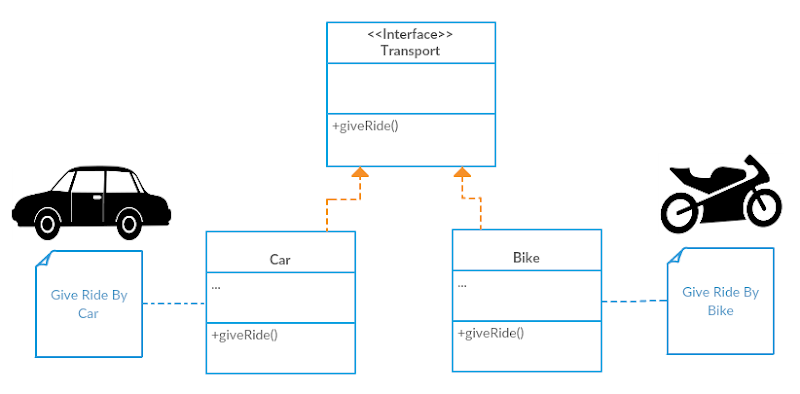
***DAO MINH DAM  
Email:***[*daominhdam@gmail.com*](mailto:daominhdam@gmail.com)

***Skype:****daominhdam (Đào Minh Đảm)*

[***https://automationfc.vn***](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://automationfc.vn%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991187000%26amp;usg%3DAOvVaw0EKB7mBt0r6jNiwuGjRsNT&sa=D&source=editors&ust=1628669991254958&usg=AOvVaw3XlhIoaEhasBDA0MoweYnn)*-*[***https://automationfc.com***](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://automationfc.com%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991187000%26amp;usg%3DAOvVaw0ExQv6gmpw9ElIcCmZ2m03&sa=D&source=editors&ust=1628669991255165&usg=AOvVaw1zcAAVq7D43aI-dsFT3JSK)

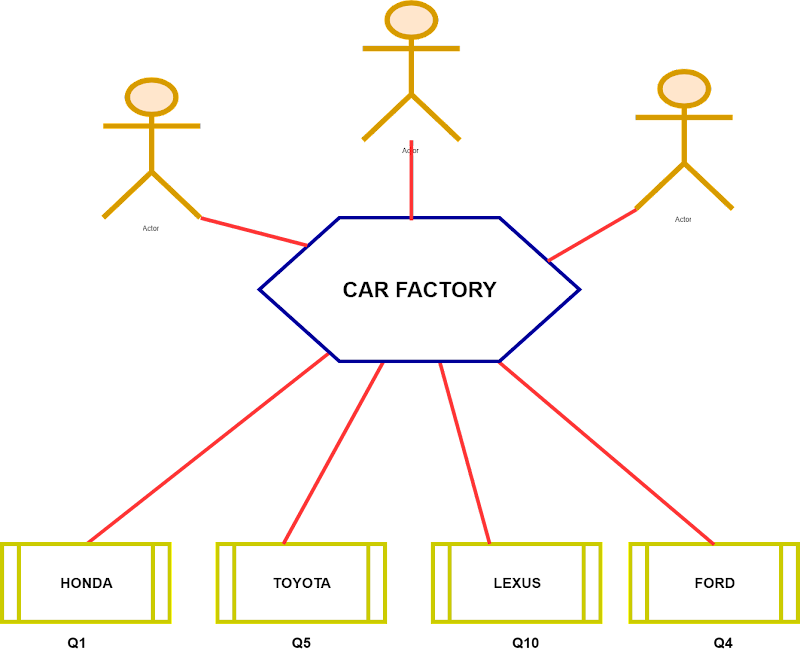
***Tài liệu này chỉ sử dụng trong khóa học - vui lòng không chia sẻ công khai***  
-------------------------------------------------------------------------------------------------------------------------------

**Factory Pattern Basic**

****

* **Bài toán mua xe ôtô**

|  |
| --- |
| *- Hai vợ chồng A đang sinh sống tại Sài Gòn có nhu cầu mua một chiếc oto, vì chưa có kinh nghiệm gì về xe oto và các hãng xe nên quyết định sẽ đi tới các cửa hàng để xem 1 số loại xe trước khi quyết định mua xe nào. - Buổi sáng 2 vợ chồng ghé qua cửa hàng xe Honda ở quận 1 để xem về các mẫu xe của loại xe Honda, buổi chiều họ ghé qua cửa hàng xe Toyota ở quận 5, buổi tối họ ghé qua cửa hàng xe Ford ở quận 4 và ngày mai sẽ qua cửa hàng xe Lexus tại quận 10*  *- Vì các cửa hàng không nằm tại cùng 1 vị trí nên việc đi lại giữa các cửa hàng tốn rất nhiều thời gian, và trường hợp muốn xem thêm loại xe khác thì khó khăn trong việc đi lại*  *- Làm sao giải quyết được bài toán họ không cần phải đi qua tất cả các cửa hàng mà vẫn có thể xem được bất kì hãng xe mà họ muốn (Ford/ Honda/ Lexus/..) và khi có 1 hãng xe mới vẫn có thể thêm vào cho khách hàng chọn lựa. => Tập trung tất cả các loại xe về 1 địa điểm (Factory - Nhà máy sản xuất) - khi khách hàng yêu cầu xem loại xe nào chỉ cần gọi đúng ra hãng xe (tên loại) mà họ cần và lấy ra để thao tác. Khi thêm vào 1 hãng xe mới không ảnh hưởng đến những chức năng cũ (vẫn hoạt động được)* |



* Không apply Factory Pattern:

|  |
| --- |
| public abstract Car {    abstract void view(); } |

|  |
| --- |
| public class Honda extends Car {    @Override    public void view() {        System.out.printf("Honda view");    } } |

|  |
| --- |
| public class Lexus extends Car {    @Override    public void view() {        System.out.printf("Lexus view");    } } |

|  |
| --- |
| public class Toyota extends Car {    @Override    public void view() {        System.out.printf("Toyota view");    } } |

|  |
| --- |
| public class Customer {    public void viewHonda() {        Honda honda = new Honda();        honda.view();    }    public void viewNexus() {        Lexus nexus = new Lexus();        lexus.view();    }    public void viewToyota(){        Toyota toyota = new Toyota();        toyota.view();    } } |

* **Apply Factory Pattern**

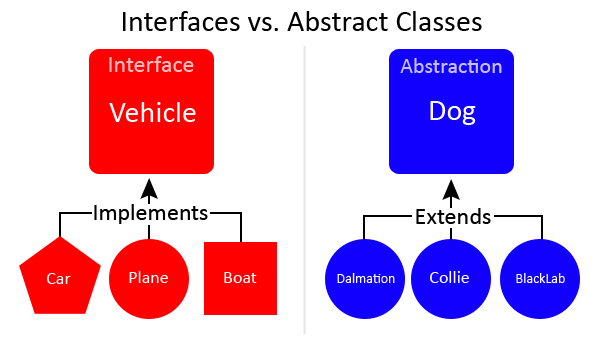
|  |
| --- |
| public class CarFactory {     public void viewCar(String carType) {        Car car;        if (carType.equalsIgnoreCase("HONDA")) {            car = new Honda();            car.view();        } else if (carType.equalsIgnoreCase("LEXUS")) {            car = new Lexus();            car.view();        } else if (carType.equalsIgnoreCase("TOYOTA")) {            car = new Toyota();            car.view();        }    } }  public class Customer {         public static void main(String[] args) {                CarFactory carFactory = new CarFactory();                carFactory.viewCar("HONDA");                carFactory.viewCar("LEXUS");                carFactory.viewCar("FORD");         } } |

* **AbstractTest (Factory Pattern)**

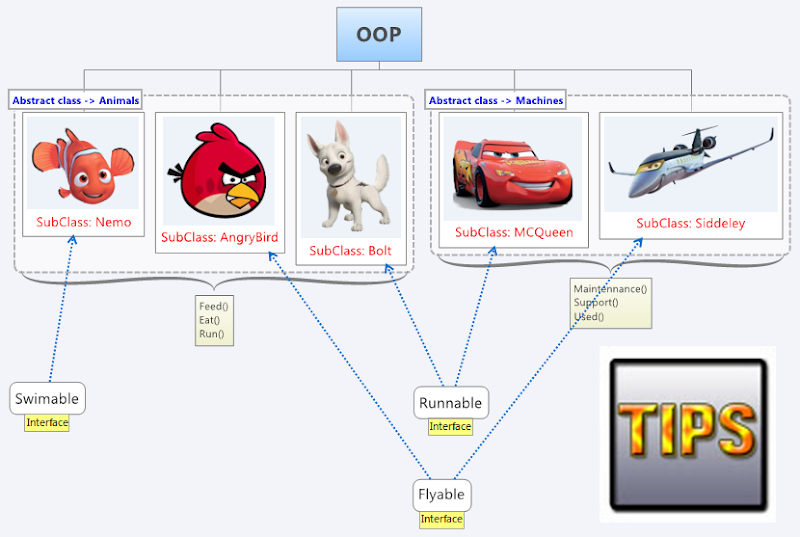
|  |
| --- |
| protected WebDriver openMultiBrowser(String browserName) {                 if (browserName.equalsIgnoreCase("firefox")) {                         driver = new FirefoxDriver();                 } else if (browserName.equalsIgnoreCase("chrome")) {                         System.setProperty("webdriver.chrome.driver", "..\\chromedriver.exe");                         driver = new ChromeDriver();                 } else if (browserName.equalsIgnoreCase("ie")) {                         System.setProperty("webdriver.chrome.driver", "..\\IEDriverServer.exe");                         driver = new InternetExplorerDriver();                 } else if (browserName.equalsIgnoreCase("chromeheadless")) {                         System.setProperty("webdriver.chrome.driver", "..\\chromedriver.exe");                         ChromeOptions options = new ChromeOptions();                         options.addArguments("headless");                         options.addArguments("window-size=1366x768");                         driver = new ChromeDriver(options);                 }                  driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);                 driver.manage().window().maximize();                 driver.get(Constants.DEV\_URL);                 return driver; } |

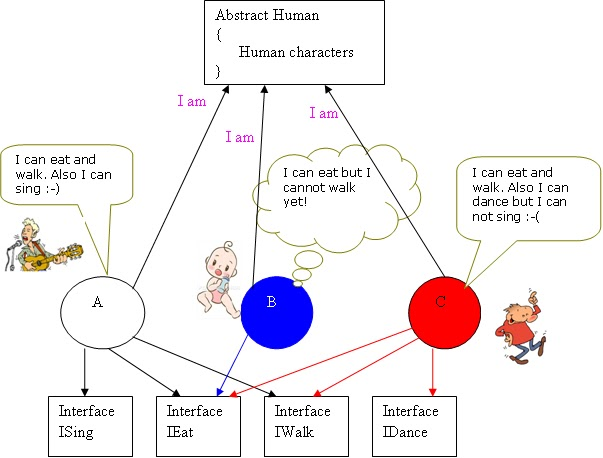
**Abstract vs Interface**

* Tính trừu tượng (abstraction) là một trong bốn tính chất cơ bản của lập trình hướng đối tượng
* Là một tiến trình ẩn các chi tiết trình triển khai và chỉ hiển thị tính năng tới người dùng.
* Tính trừu tượng cho phép bạn loại bỏ tính chất phức tạp của đối tượng bằng cách chỉ đưa ra các thuộc tính và phương thức cần thiết của đối tượng trong lập trình
* Tính trừu tượng giúp bạn tập trung vào những cốt lõi cần thiết của đối tượng thay vì quan tâm đến cách nó thực hiện
* Sử dụng abstract Class và Interface để có tính trừu tượng



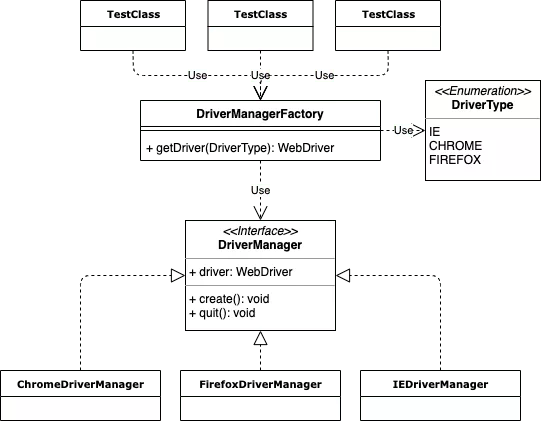
* **Interface:**
* Khi muốn tạo dựng một bộ khung chuẩn gồm các chức năng mà những module cần phải có. Sau khi nhận yêu cầu của khách hàng cả team ngồi với nhau và phân tích các đầu mục các tính năng của từng module - sau đó triển khai vào code viết các interface như đã phân tích để các bạn có thể nhìn vào đó thực hiện đủ các tính năng (khi đã implement rồi thì không sót một tính năng nào)
* **Lưu ý với Interface:**
* Không khởi tạo từ Interface được vì Interface không có constructor
* Xây dựng được bộ khung mẫu mà các lớp phải follow theo
* Chỉ chứa method trống không có phần thân - phần thân sẽ được code bởi những class implement (cài đặt) interface đó
* Cho dù method ko viết rõ nhưng đều hiểu method này là public astract method (mặc định là public abstract)
* Các biến (thuộc tính) trong Interface mặc định là public static final (hằng số)
* Các lớp có thể cài đặt (implements) một hoặc nhiều Interface (đa kế thừa)
* Chỉ có duy nhất 1 access modifier là public (áp dụng cho cả thuộc tính/ phương thức)
* Từ Java 8 thì Interface có thêm 2 loại phương thức là default và static
* **Abstract Class:**
* Giống như Interface nhưng khi định nghĩa một đối tượng có những chức năng A/B/C thì trong đó tính năng A/B chắc chắn sẽ thực thi còn tính năng C tùy thuộc vào đối tượng cụ thể là gì - các lớp extend phải hoàn thành nốt tính năng này - còn những tính năng đã hoàn thành vẫn sử dụng như bình thường vì đây là những tính năng chung
* Sử dụng làm class parent (lớp cha), khai báo biến (thuộc tính) và cài đặt các phương thức (hành vi) giống nhau. Các thuộc tính/ hành vi riêng biệt sẽ được cài đặt ở sub class (lớp con)
* **Lưu ý với Abstract:**
* Không thể khởi tạo 1 đối tượng trực tiếp từ một class trừu tượng (tương tự như Interface)
* Có thể linh động các method giống như một class thông thường
* Có thể không khai báo hoặc khai báo nhiều abstract method bên trong
* Có thể chứa cả các method không trừu tượng (thường) nên các class kế thừa có thể override hoặc không override các method thường
* Chỉ hỗ trợ đơn kế thừa/ ko hỗ trợ đa kế thừa như Interface
* Bao gồm tất cả các access modifier (public/ protected/ private/ default)
* Class abstract có thể implement các Interface nhưng Interface thì không thể implement class abstract
* Class abstract có thể kế thừa được một class khác và Interface có thể kế thừa được nhiều Interface khác





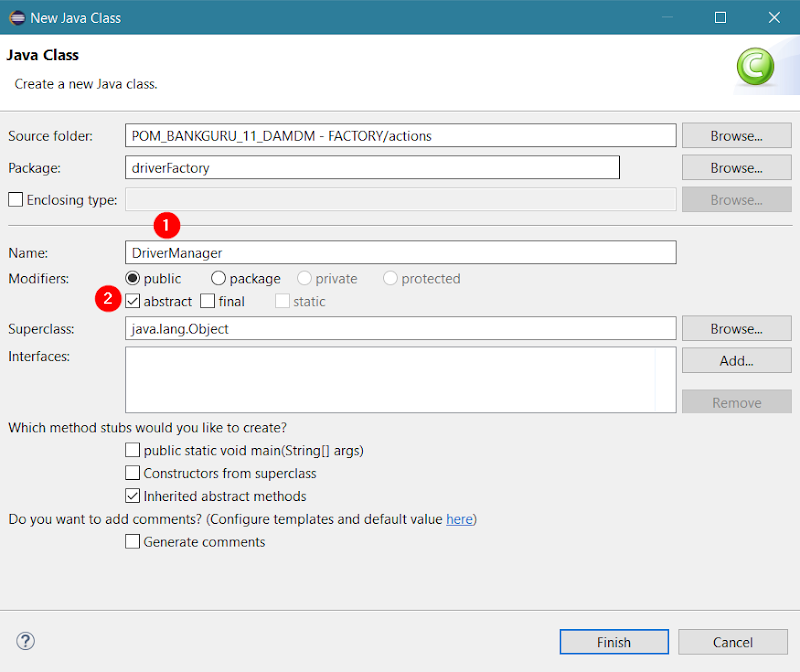
* **Refrences:**
* [https://o7planning.org/vi/10205/abstract-class-va-interface-trong-java](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://o7planning.org/vi/10205/abstract-class-va-interface-trong-java%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991205000%26amp;usg%3DAOvVaw0O5RmFuHwsdiKCdEQUB1b_&sa=D&source=editors&ust=1628669991262022&usg=AOvVaw3Z1qgTWbfgzn7gJ5JlCVPm)
* [https://gpcoder.com/2295-abstract-class-va-interface-trong-java/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://gpcoder.com/2295-abstract-class-va-interface-trong-java/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991206000%26amp;usg%3DAOvVaw3yt8juU-DHiyimv4H0WzRv&sa=D&source=editors&ust=1628669991262268&usg=AOvVaw2cOPEkd3FF8a-YEhUahuhO)
* [https://levunguyen.com/laptrinhjava/2020/04/02/su-khac-nhau-giua-abstract-interface/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://levunguyen.com/laptrinhjava/2020/04/02/su-khac-nhau-giua-abstract-interface/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991206000%26amp;usg%3DAOvVaw3djyGQkmTmHWaswaHc75LM&sa=D&source=editors&ust=1628669991262388&usg=AOvVaw1roC8X2w2Rf8BoC4fxoSvi)

**Factory Pattern Advanced**



* **Hàm openMultiBrowser sẽ trở nên cồng kềnh khi add các service (capability) của từng browser vào**
* [https://chromedriver.chromium.org/capabilities](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://chromedriver.chromium.org/capabilities%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991207000%26amp;usg%3DAOvVaw0qBbRRnJ_kUTrRF1uJiCyR&sa=D&source=editors&ust=1628669991262667&usg=AOvVaw1t1sOSQyy-L8Dg37LTDYjP)
* [https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/ie/InternetExplorerOptions.html](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/ie/InternetExplorerOptions.html%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991207000%26amp;usg%3DAOvVaw10anLO_h74iS1AaZO3l-6-&sa=D&source=editors&ust=1628669991262813&usg=AOvVaw3WEqxk9TqFPxlAoKMUut9Y)

|  |
| --- |
| protected WebDriver openMultiBrowser(String browserName, String version) {                 if (browserName.equalsIgnoreCase("firefox")) {                         WebDriverManager.firefoxdriver().version(version).setup();                         System.setProperty(FirefoxDriver.SystemProperty.DRIVER\_USE\_MARIONETTE, "true");                         System.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE, workingDir + "\\FirefoxUILog.txt");                         FirefoxOptions options = new FirefoxOptions();                         options.addArguments("-private");                         driver = new FirefoxDriver(options);                 } else if (browserName.equalsIgnoreCase("firefoxheadless")) {                         WebDriverManager.firefoxdriver().version(version).setup();                         FirefoxOptions options = new FirefoxOptions();                         System.setProperty(FirefoxDriver.SystemProperty.DRIVER\_USE\_MARIONETTE, "true");                         System.setProperty(FirefoxDriver.SystemProperty.BROWSER\_LOGFILE, workingDir + "\\FirefoxHeadlessLog.txt");                         options.setHeadless(true);                         driver = new FirefoxDriver(options);                 } else if (browserName.equalsIgnoreCase("chrome")) {                         WebDriverManager.chromedriver().version(version).setup();                         DesiredCapabilities capabilities = DesiredCapabilities.chrome();                         ChromeOptions options = new ChromeOptions();                         options.addArguments("--incognito");                         options.setExperimentalOption("useAutomationExtension", false);                         options.setExperimentalOption("excludeSwitches", Collections.singletonList("enable-automation"));                         capabilities.setCapability(ChromeOptions.CAPABILITY, options);                         driver = new ChromeDriver(options);                 } else if (browserName.equalsIgnoreCase("chromeheadless")) {                         WebDriverManager.chromedriver().version(version).setup();                         ChromeOptions options = new ChromeOptions();                         options.addArguments("headless");                         options.addArguments("window-size=" + Constants.HEADLESS\_RESOLUTION);                         driver = new ChromeDriver(options);                 } else if (browserName.equalsIgnoreCase("ie")) {                         WebDriverManager.iedriver().arch32().version(version).setup();                         driver = new InternetExplorerDriver();                 } else if (browserName.equalsIgnoreCase("edge")) {                         WebDriverManager.edgedriver().version(version).setup();                         driver = new EdgeDriver();                 } else {                         System.out.println("Please choose your browser name in TestNG xml file.");                 }                  driver.get(Constants.DEV\_URL);                 driver.manage().timeouts().implicitlyWait(Constants.LONG\_TIMEOUT, TimeUnit.SECONDS);                 return driver;         } |

* Tạo class DriverManager (**abstract**): cho phép các class test get driver
* abstract void **createDriver**()
* quitDriver()/ getDriver() -> method thường

|  |
| --- |
| public abstract class DriverManager {         protected WebDriver driver;          protected abstract void createDriver();          public void quitDriver() {                 if (driver != null) {                         driver.quit();                         driver = null;                 }          }          public WebDriver getDriver() {                 if (driver == null) {                         createDriver();                 }                 return driver;         } } |

* Tạo class ChromeDriverManager: Quản lí các service của trình duyệt Chrome

|  |
| --- |
| public class ChromeDriverManager extends DriverManager {         @Override         protected void createDriver() {                 DesiredCapabilities capabilities = DesiredCapabilities.chrome();                 ChromeOptions options = new ChromeOptions();                 options.addArguments("--incognito");                 capabilities.setCapability(ChromeOptions.CAPABILITY, options);                 driver = new ChromeDriver(capabilities);         } } |

* Tạo class FirefoxDriverManager: Quản lí các service của trình duyệt Firefox

|  |
| --- |
| public class FirefoxDriverManager extends DriverManager {         @Override         protected void createDriver() {                 String rootFolder = System.getProperty("user.dir");                 FirefoxProfile profile = new FirefoxProfile();                 DesiredCapabilities capability = DesiredCapabilities.firefox();                 profile.setAcceptUntrustedCertificates(false);                 profile.setAssumeUntrustedCertificateIssuer(true);                 profile.setPreference("dom.webnotifications.enabled", false);                 profile.setPreference("browser.download.folderList", 2);                 profile.setPreference("browser.helperApps.alwaysAsk.force", false);                 profile.setPreference("browser.download.manager.showWhenStarting", false);                 profile.setPreference("browser.download.dir", rootFolder + "\\downloadFiles");                 profile.setPreference("browser.download.downloadDir", rootFolder + "\\downloadFiles");                 profile.setPreference("browser.download.defaultFolder", rootFolder + "\\downloadFiles");                 profile.setPreference("browser.helperApps.neverAsk.saveToDisk", "text/anytext ,text/plain,text/html,application/plain");                 capability = DesiredCapabilities.firefox();                 capability.setCapability(FirefoxDriver.PROFILE, profile);                 driver = new FirefoxDriver(capability);         } } |

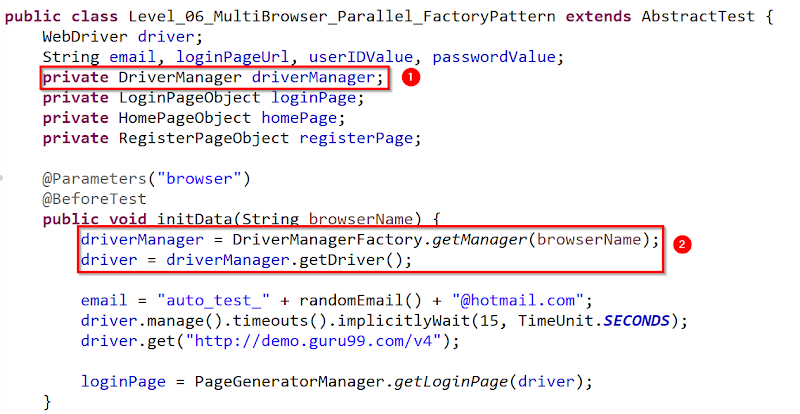
* Tạo class ChromeHeadlessDriverManager: Quản lí các service của trình duyệt Chrome Headless

|  |
| --- |
| public class ChromeHeadlessDriverManager extends DriverManager {         @Override         protected void createDriver() {                 DesiredCapabilities capabilities = DesiredCapabilities.chrome();                 ChromeOptions options = new ChromeOptions();                 options.addArguments("headless");                 options.addArguments("window-size=1920x1080");                 capabilities.setCapability(ChromeOptions.CAPABILITY, options);                 driver = new ChromeDriver(capabilities);         } } |

* Tương tự cho các loại browser khác:
* EdgeDriverManager
* IEDriverManager
* SafariDriverManager
* OperaDriverManager
* Android/ iOSDriverManager
* Tạo class **DriverManagerFactory**để quản lí việc gọi ra browser driver tương ứng

|  |
| --- |
| public class DriverManagerFactory {         public static DriverManager getManager(String browserName) {                 DriverManager driverManager;                 switch (browserName) {                 case "chrome":                         driverManager = new ChromeDriverManager();                         break;                 case "firefox":                         driverManager = new FirefoxDriverManager();                         break;                 default:                         driverManager = new ChromeHeadlessDriverManager();                         break;                 }                 return driverManager;         } } |

* Apply ở tầng testcase (class test)



**Apply**

* **Dự án nào nên apply Factory Pattern:**chạy trên nhiều platform/ browser/ os khác nhau
* Chrome
* Firefox
* IE
* Edge Old/ New
* Safari
* Android
* iOS
* Remote (Selenium Grid)
* Dự án quy mô nhỏ - chạy trên 1 vài loại browser (chrome/ firefox) thì không cần apply - chỉ cần apply hàm getBrowserDriver ở trong AbstractTest

**Reference**

* [https://stackjava.com/design-pattern/factory-pattern.html](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://stackjava.com/design-pattern/factory-pattern.html%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991227000%26amp;usg%3DAOvVaw2e5UcYoXmUYYus24WI85sb&sa=D&source=editors&ust=1628669991268376&usg=AOvVaw1fUTIsIYF_yXqZkwQpCijN)
* [https://gpcoder.com/4352-huong-dan-java-design-pattern-factory-method/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://gpcoder.com/4352-huong-dan-java-design-pattern-factory-method/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991228000%26amp;usg%3DAOvVaw02G3g4bQf4IIZFMBv0wMdH&sa=D&source=editors&ust=1628669991268635&usg=AOvVaw1S76PTxRLKIb_HJCF6KeQn)
* [https://gpcoder.com/4190-huong-dan-java-design-pattern-singleton/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://gpcoder.com/4190-huong-dan-java-design-pattern-singleton/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991228000%26amp;usg%3DAOvVaw1CANzDiau-I9sG-k6zqplE&sa=D&source=editors&ust=1628669991268863&usg=AOvVaw1gGwcXljoix2Ia2uCfkar0)
* [https://stackjava.com/design-pattern/singleton-pattern.html](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://stackjava.com/design-pattern/singleton-pattern.html%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991228000%26amp;usg%3DAOvVaw05tvxGgtwcTIPMg9rBuwLS&sa=D&source=editors&ust=1628669991268995&usg=AOvVaw1XuwS-jDx2I7Z9Sur8P2Bn)
* [https://www.automatetheplanet.com/category/series/designpatterns/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.automatetheplanet.com/category/series/designpatterns/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991229000%26amp;usg%3DAOvVaw3FXgxXD8v04NJmvlZ9tYw0&sa=D&source=editors&ust=1628669991269104&usg=AOvVaw38ZK8jqhBTeKQK8YlP04dj)
* [http://www.eliasnogueira.com/create-a-browser-factory-with-webdrivermanager/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttp://www.eliasnogueira.com/create-a-browser-factory-with-webdrivermanager/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991229000%26amp;usg%3DAOvVaw0iJCU24_Xy8vRbStBu0-qF&sa=D&source=editors&ust=1628669991269225&usg=AOvVaw0a9tlb66dDuBouHsPJxZ6_)
* [https://levunguyen.com/craftmanship/2020/05/06/su-dung-abstract-factory-design-pattern/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://levunguyen.com/craftmanship/2020/05/06/su-dung-abstract-factory-design-pattern/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628669991229000%26amp;usg%3DAOvVaw1luJUpDT0MRsaYA8d43LU_&sa=D&source=editors&ust=1628669991269347&usg=AOvVaw3i-OEbv4z-ZsX9eJSZOEwT)