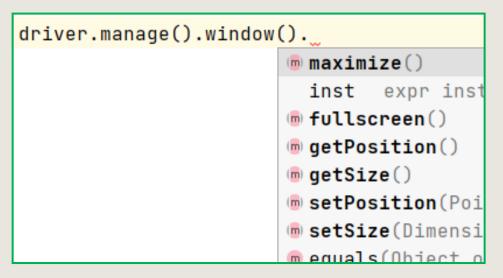
SELENIUM 4 (YENILIKLER VE DEGISIKLIKLER

Selenium 4 ile kullanabileceginiz yeni ozellikleri gunluk olarak sizlerle paylasacagim

Selenim 4 ve Selenium 3 karsilastirmasi

1) driver.manage.window().minimize()







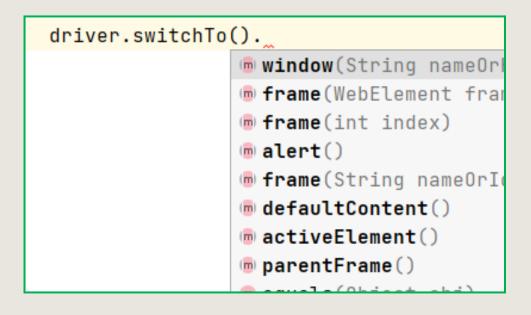
```
driver.manage().window().

maximize()
inst expr instanced
fullscreen()
getPosition()
getSize()
minimize()
winst expr instanced
fullscreen()
setPosition()
setPosition(Point tale)
setSize(Dimension tale)
```

Selenium 3'de var olan window manage ozelliklerine ek olarak minimize() eklendi. minimize() ile normal hayatta yaptiginiz sekilde window'u sembol olarak asagi indirebilirsiniz.

2) driver.switchTo ().newWindow();









Selenium 3'de var olan switchTo () ozelliklerine ek olarak newWindow() eklendi. Yeni pencereyi isterseniz TAB, isterseniz yeni WINDOW olarak acabilirsiniz.

3) webElement.get





webElement.get

m getText()
m getAttribute(String name)
m getSize()
m getAccessibleName()
m getAriaRole()
m getCssValue(String propertyName)
m getDomAttribute(String name)
m getDomProperty(String name)
m getLocation()
m getRect()
m getShadowRoot()
m getTagName()

Selenium 3'de Web Element'ler icin var olan Size, Text, Location ve Rect method' larina ilave olarak

Web Element'in AccessibleName ve Aria role ozelliklerini ogrenebilecegimiz method'lar eklendi

```
=======Selenium 3 de olanlar=======

Size : (344, 38)

Text :

Location : (267, 11)

Rect.Height : 38

======Selenium 4 de yenilikler=======

AccessibleName : Search

Aria role : textbox
```

4) webElement.get





Selenium 3'de Web Element'ler icin var olan istenen attribute'un value'su, web elementin tagName'i ve spesifik CSS ozelliklerinin yanina

Web Element'in Dom ile erisebilmeniz icin attribute'lerin Dom degeri ve Dom property degerini ogrenebileceginiz method'lar eklendi

```
webElement.get

m getText()
m getAttribute(String name)
m getSize()
m getAccessibleName()
m getAriaRole()
m getCssValue(String propertyName)
m getDomAttribute(String name)
m getDomProperty(String name)
m getLocation()
m getRect()
m getShadowRoot()
m getTagName()
```

```
=======Selenium 3 de olanlar=======

Id attribute'nun degeri : twotabsearchtextbox

webElement'in Css background color degeri : rgba(0, 0, 0, 0)

webElement Tag name'i : input

======Selenium 4 de yenilikler======

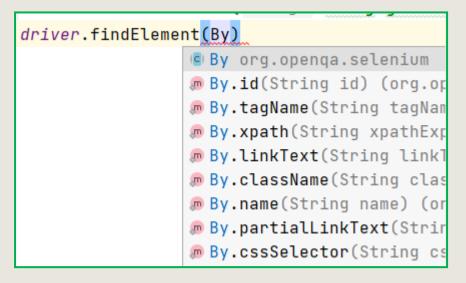
Id attribute'nun Dom attribute degeri : twotabsearchtextbox

Id attribute'nun Dom property degeri : null
```

5) Relative Locators







driver.findElement(with(B))

By org.openqa.selenium

By.id(String id) (org.open

By.tagName(String tagName)

By.linkText(String linkTex

By.xpath(String xpathExpre

By.className(String classN

By.name(String name) (org.

By.partialLinkText(String mane)

By.cssSelector(String cssS

Selenium 3'de Web Element'leri locate etmek icin kullandigimiz 8 adet locator Selenium 4'de de var Var olan bu method'larin yanina Selenium 4 Devrim gibi bir ozellik eklemis

Eklenen RelativeLocator Class'indaki method'lar sayesinde bir web elementi etrafindaki diger web elementler ile tanimlayabilirsiniz

6) Relative Locators ornek

Bazi web elementlerde developer'lar dinamik ozellikler kullandiklarindan sabit bir locator bulamayabilirsiniz.

Bu durumda Selenium 4 imdadiniza yetisip, gunluk hayatimizda kullandigimiz adres tariflerine benzer bir imkan sunuyor.

Once with ile var olan her hangi bir ozelligini soyleyip sonra etrafindaki web elementleriyle tarif verebilirsiniz

Asagidaki ornekte **boston** ve **sailor** web elementlerini locate edip, onlar uzerinden **berlin ve mountie** web elementlerini locate ettim.





```
driver.get("https://www.diemol.com/selenium-4-demo/relative-locators-demo.html#");
WebElement boston=driver.findElement(By.id("boston"));
WebElement sailor = driver.findElement(By.id("sailor"));
WebElement berlin =driver.findElement(RelativeLocator.with(By.tagName("li")).above(sailor).toRightOf(boston));
WebElement mountie=driver.findElement(RelativeLocator.with(By.className("ui-li-has-thumb")).below(boston));
```

2022 Ocak Ahmet BULUTLUOZ

7) manage().timeouts()





Selenium 3'de wait icin zaman tanimlarken kullandigimiz long TimeUnit kaldirilmis, Yerine java util kutuphanesinden Duration class'indan kullanacagimiz ofSeconds(sure) metotlari getirilmis.

```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
driver.manage().timeouts().
                          m implicitlyWait(Duration duration)
                                                                   Tim
                         m implicitlyWait(long time, TimeUnit ...
                                                                   Tim
                           inst expr instanceof Type ? ((Type) expr
                         m scriptTimeout(Duration duration)
                                                                   Tim
                         m getScriptTimeout()
                                                                   Dur
                         m getImplicitWaitTimeout()
                                                                   Dur
                         m getPageLoadTimeout()
                                                                   Dur
                         pageLoadTimeout(Duration duration)
                                                                   Tim
```

8) Explicitly wait



WebDriverWait wait=new WebDriverWait(driver, timeOutInSeconds: 20);



WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));

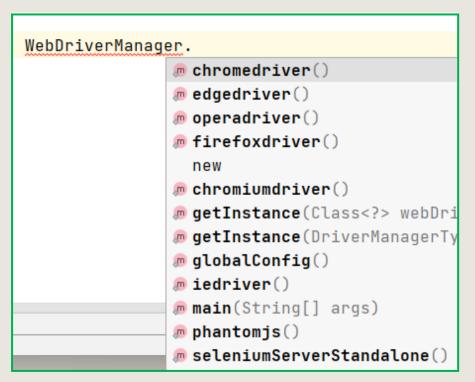
Selenium 3'de time icin kullanilan timeOutInSeconds Class'inin yerine Selenium 4'de Duration Class'i kullanilmaya basladi.

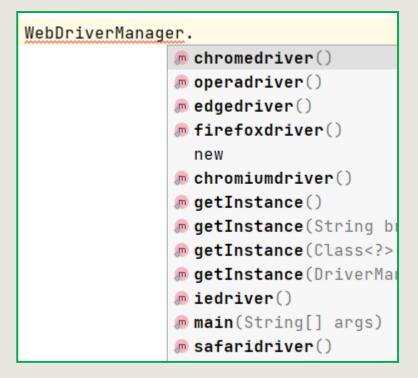
Bu degisim implicitly wait icin manage().timeouts()'da oldugu gibi explicitly wait'de wait objesi olustururken de karsimiza cikiyor.

9) WebDriverManager.









Selenium 3'de desteklenen browser'lardan phantomjs Selenium 4'de kaldirilirken, Selenium 3'de direk kullanilmayip getInstance () ile kullanilabilen safaridriver selenium 4'de direk kullanima acilmis

NOT : Bazi kaynaklarda Selenium 4'de Opera desteginin de kaldirildigi yazsa da yukarida gorebileceginiz sekilde operadriver listede yerini aliyor

10) ScreenShot





```
public static String getScreenshot(String name) throws IOException {
    // naming the screenshot with the current date to avoid duplication
    String date = new SimpleDateFormat( pattern: "yyyyMMddhhmmss").format(new Date());

    // TakesScreenshot is an interface of selenium that takes the screenshot
    TakesScreenshot ts = (TakesScreenshot) Driver.getDriver();
    File source = ts.getScreenshotAs(OutputType.FILE);
    // full path to the screenshot location
    String target = System.getProperty("user.dir") + "/test-output/Screenshots/" + name + date + ".png";

    File finalDestination = new File(target);
    // save the screenshot to the path given
    FileUtils.copyFile(source, finalDestination);
    return target;
}
```

```
public void test01() throws InterruptedException, IOException {
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();

    driver.get("https://www.amazon.com");
    Thread.sleep( millis: 3000);

WebElement aramaKutusu=driver.findElement(By.id("twotabsearchtextbox"));

File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
    FileUtils.copyFile(scrFile, new File( pathname: "./tumSayfa.png"));

File scrFile1 = aramaKutusu.getScreenshotAs(OutputType.FILE);
    FileUtils.copyFile(scrFile1, new File( pathname: "./aramaKutusu.png"));
}
```

Selenium 3 de ekran fotografi almak mumkundu ancak bir method olusturmak ve onu kullanmak gerekiyordu, soldaki ornekte onceki framework'de kullandigim, sadece tum sayfanin goruntusunu kaydeden metodu gorebilirsiniz.

Selenium 4'de tum ekranin fotografini veya istediginiz bir WebElement'in goruntusunu kolaylikla alabilirsiniz. Sagdaki ornekte Amazon sayfasina gidip hem tum sayfanin hem de arama kutusunun goruntusunu 2 satir kod yazarak aldim