

Q U A I T Y U E R S E  
A C A D E M Y

Selenium

Ders-01

Selenium Giriş

Webdriver Method'lari

WebElements

unityverseacademy.com

Egitmen : Ahmet BULUTLUOZ

# Software Testing Nedir ?

Software testing var olan veya gelistirilmekte olan bir uygulamanin, tasrim asamasinda planlanan ozellikleri tasiyip tasimadiginin belirlenmesi icin yapılan faaliyetlerin butunudur.

Software testi icin tasrim asamasinda belirlenen sonuclar (**Expected Result**) ile uygulamanin kendisinden alınan sonuclar (**Actual Result**) karsilastirilir.

Expected ve actual result birbirine esit ise test basarili (**Test Passed**), Expected ve actual result birbirine esit degilse test basarisiz (**Test Failed**) olarak raporlanir.

Test gelistirme dongusunde developer'lar bir feature icin kod yazmaya basladiklarinda, tester'larda o feature'i analiz ederek acceptance criteria cercevesinde expected result'lari tespit etmeli, yazılımin bu ihtiyacları karşıladıgından emin olmak icin positive ve negative test senaryolari olusturmali ve bu testleri otomasyonla yapacak test case'leri olusturmalıdır.



# Software Testing Neden Önemlidir ?

Gunumuzdeki rekabetci piyasa kosullari, uygulamalari bugs - free olmaya zorlamaktadir.

Ayrıca developer'larin user case'den anladiklari ile end – user'larin kullanım aliskanliklari her zaman uyusmayabilir.

Uygulamaya sonradan eklenen bazi feature'lar calisan uygulamada bazi islevleri negatif etkileyebilir.

Kullanicinin beklentilerini karsilamayan uygulamalar basarisiz olur.



- Urunun end-user kullanımına hazır olduğundan emin olmak
- End-user tarafından karşılanan sorunlarda düzeltme ve yeniden yapma maliyetlerini azaltmak
- Uygulamanın marketteki algısının üst seviyede olmasını sağlamak
- Sonradan eklenen işlevlerin eski işlevleri bozmadığından emin olmak için software testing yazılım geliştirme sürecinin vazgeçilmez bir parçası olmuştur.

# Manual Software Testing Nedir ?

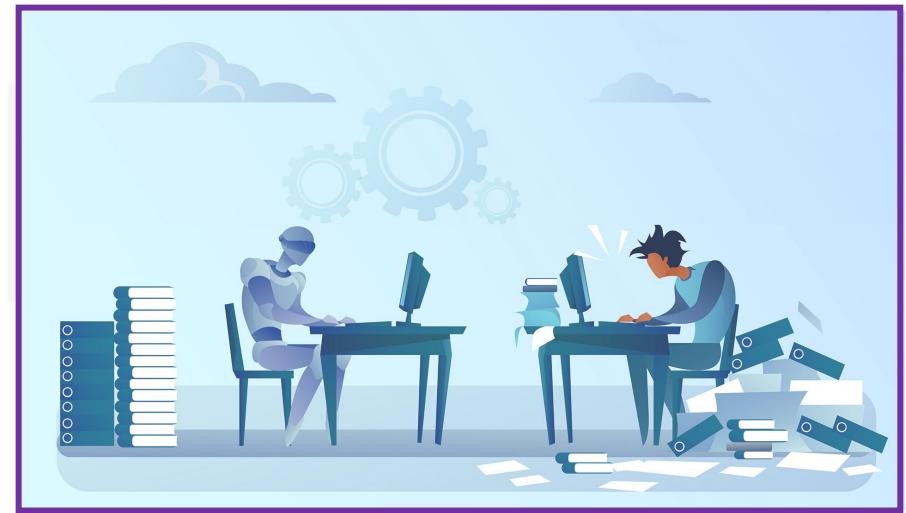
Manual testing, uygulamanın planlanan sonuçlara uygun çalışıp çalışmadığını hiç bir otomasyon aracı kullanmadan, bir end-user gibi test edilmesidir.

Ancak insan gücüyle yapılan bu testler için hem çok fazla insan gücüne ve zamana ihtiyaç duyulur hem de insanın özelliklerimizden dolayı testlerde yanlışlıklar yapılabilir.

Zaman ve ihtiyac duyulan insan gücünü azaltmak, testler çalıştırılırken ortaya çıkabilecek hataları minimum'a indirmek için test otomasyonu gereklidir.

Manual tester'lar uygulama üzerinde çok fazla zaman harcadıkları ve her adımı manual yaptıkları için uygulama bilgileri daha iyidir.

Automation tester'lar uygulamayı daha iyi anlamak ve sistem ihtiyaçlarını görmek için başlangıçta bir kaç kez manual test yapıp sonra otomasyon yapmalıdır.



# Test Otomasyonu Nedir ?

Test otomasyonu, insan gucu ile klavye ve mouse kullanilarak yapilabilecek bir yazılım testinin, bir otomasyon aracı kullanilarak kodlar aracılıgi ile yapılmasidir.

Test otomasyon sayesinde

- klavye ve mouse kullanilarak yapilabilecek islemlerin cogu yapilabilir,
- yapılan islemler sonucunda gereklesen sonuclar kaydedilebilir
- Elde edilen sonuclarla, expected sonuclar karsilastirilip, testin sonunu bulunabilir,
- Ve istenirse otomatik raporlar olusturulabilir



Otomasyonu yapılan bir test, istenen aralıklarla tekrar calistirilabilir. Hatta belirli aralıklarla olusturulan tum testler calistirilarak sistemin saglikli olarak calismaya devam ettiginden emin olunabilir (Regression Test)

Test otomasyonu insan gucu ihtiyacini azaltmasi, sorunsuz calismasi gibi ozellikleri sayesinde her gecen gun daha çok talep gormektedir.

# Automation & Manual Testing

Yandaki kod sizce nedir ?

- A- Test Case
- B- Manuel tester icin test adimlari
- C- Otomasyon ile test yapan kodlar

Feature: US1010 herokuapp Delete testi

@heroku @sirali @pr1

Scenario: TC15 herokuapp'dan delete butonu calismali

Given kullanici "herokuappUrl" anasayfasinda

And add element butonuna basar

And kullanici 3 sn bekler

Then Delete butonu gorunur oluncaya kadar bekler

And Delete butonunun gorunur oldugunu test eder

Then Delete butonuna basar

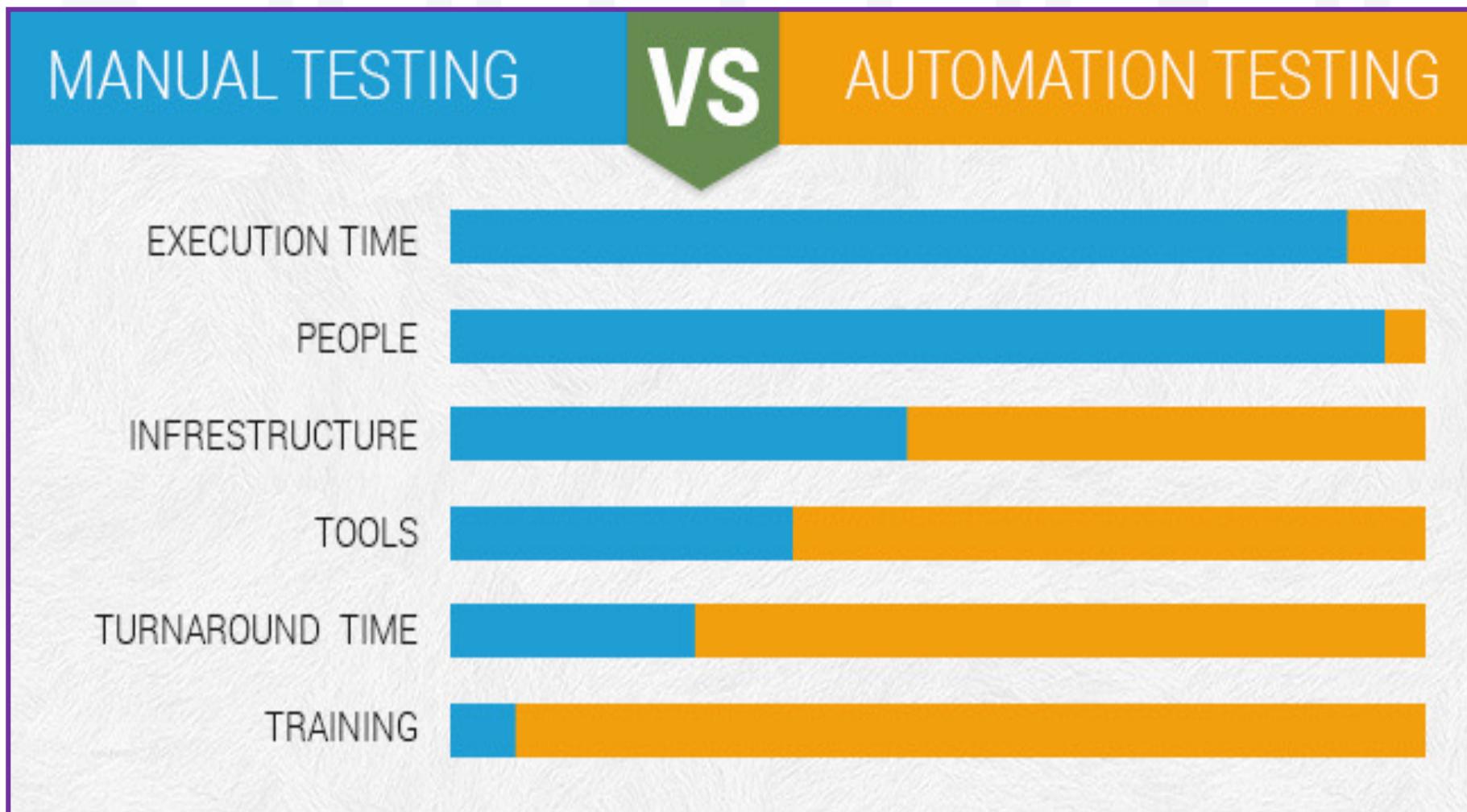
And Delete butonunun gorunmedigini test eder

And sayfayı kapatır

Test otomasyonu sayesinde herkesin anlayacagi test case'ler olusturabilir, daha kisa surede, daha az insan kaynagi ile testlerinizi gerceklestirebilir, istediginiz raporlari otomatik olarak olusturabilirsiniz.

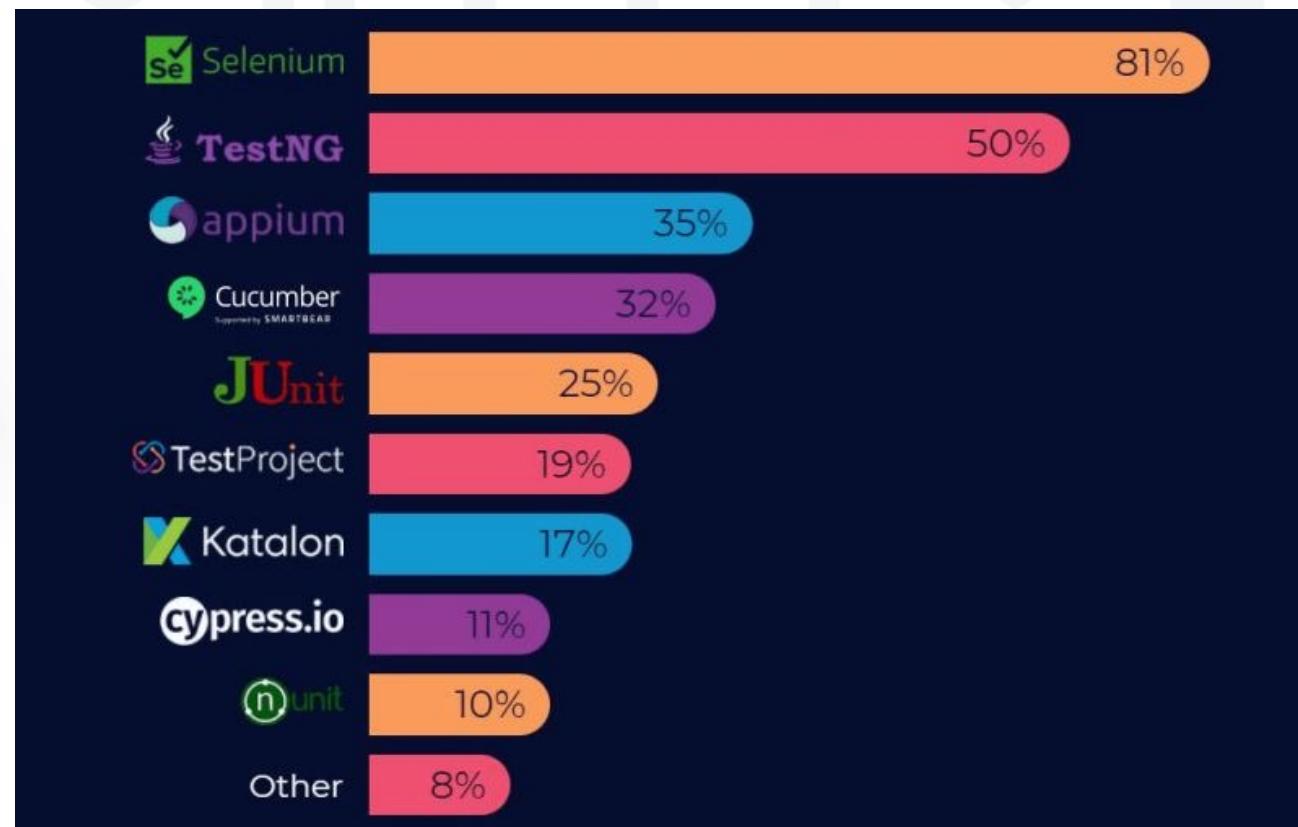
Manuel Test sayesinde kod bilgisi olmasa bile insanlara test yapabilir, temel test ihtiyaclarinizi karsilayabilirsiniz.

# Automation & Manual Testing



# En Çok Kullanılan Tool'lar

En çok kullanılan test otomasyon tool'lari



# Selenium Nedir ?

## About Selenium

Selenium is a suite of tools for automating web browsers.

Selenium browser'ları otomasyon ile çalıştıracak tool'ların çalışma için oluşturulan bir paketdir.

Selenium farklı programlama dilleri ile çalışarak günümüzde kullanılan browser'ların tamamını otomasyon ile çalıştırabilme için oluşturulan class ve method'lara sahiptir.

Selenium'u kullanabilmek için bu class'lar çalışılan projeye eklenmelidir.

Selenium'un class'larını, kendi sitesinden indireceğimiz jar dosyalarını projeye ekleyerek projemize dahil edebilir veya bu işi bizim adımıza yapacak maven gibi tool'lari kullanarak class'lari direkt projemize ekleyebiliriz.

# Selenium Nedir ?

**Selenium automates browsers. That's it!**

What you do with that power is entirely up to you.

Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that.

Boring web-based administration tasks can (and should) also be automated as well.

Selenium browser'lari otomasyonla calistirir, bu otomasyon gucu ile ne yapacagini tamamen size kalmistir.

Selenium web uygulamalarini test etmek icin kullanilan acik kaynakli, ucretsiz bir uygulamadir.

2021 yilinda Selenium 4 piyasaya ciktig ve Selenium'a yeni yetenekler(method'lar) kazandirdi.

Selenium, Java, Phyton, .Net gibi en çok kullanılan programlama dilleri ile kullanılabilir.

# IntelliJ Nedir ?

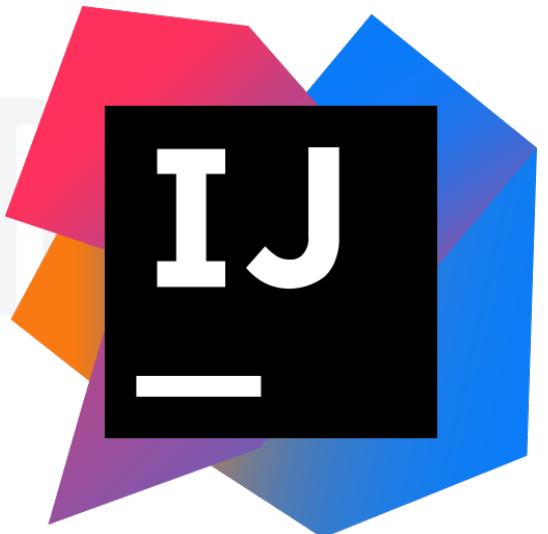
High Level programlama dilleri calisabilmek icin derleyicilere ihtiyac duyarlar.

IntelliJ IDEA 2000 yılında kurulmuş olan JetBrains firmasına ait olan, popüler bir kod gelistirme ortamı (**I**ntegrated **D**evelopment **E**nvironment) dir.

IntelliJ uretkenligi en ust duzeye tasiyacak akilli kodlama yardimi, kod tamamlama ve ergonomic tasarrim gibi ozelliklerle kod yazimini sadece verimli degil, ayni zamanda keyifli hale getirmistir.

Bir çok framework ve plugin ile calisma imkani saglar.

Kısaca, IntelliJ IDE ihtiyaçlarınızı tahmin eder ve sıkıcı ve tekrarlayan geliştirme görevlerini otomatikleştirir, böylece büyük resme odaklanabilirsiniz.



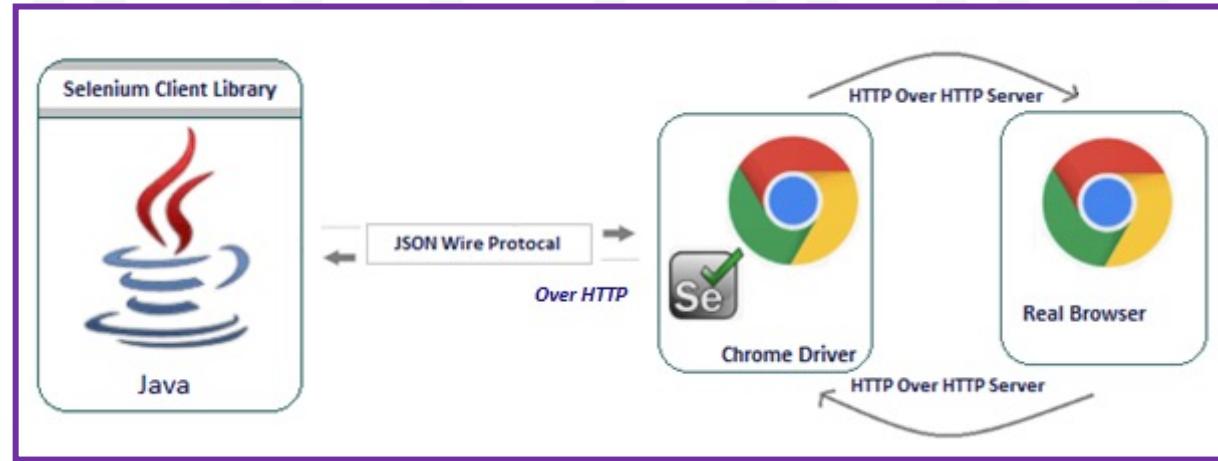
# Selenium Bileşenleri

**Selenium'un dört bileşeni vardır;**

- Selenium Integrated Development Environment (IDE) (Selenyum Entegre Geliştirme Ortamı (IDE))
- Selenium Remote Control (RC)(Selenyum Uzaktan Kumanda (RC))
- WebDriver ( Biz Selenium WebDriver kullanacağız)
- Selenium Grid ( paralel test için kullanılıyor)



# Selenium Nasil Calisir ?



Selenium test otomasyonunu WebDriver ile gerceklestirir.

Java ile yazdigimiz kodlar ile kullanilacak browser'a uygun bir webDriver objesi olusturulur.

Selenium kullanarak WebDriver class'indan olusturulan driver objesi bizim elimiz, gozumuz gibi calisir. Gonderildigi web sayfasinda klavye ve mouse ile yapabilecek islemleri yapar, elementlere tiklama, yazi gonderme, elementler uzerindeki yazilari alma gibi pekcoek islemi gerceklestirir. Elde ettigi sonucları Java kodlarinin oldugu ortama döndürür.

# Selenium'un Avantajlari & Dezavantajlari



- 1 ) Ücretsiz ve acik kaynaklidir. ( Open source )
- 2 ) Bir çok programlama dilini destekler  
(Java, Python, PHP, C#, Ruby vs.)
- 3 ) Çoklu işletim sistemleriyle çalışır.  
Multiple operating systems (Windows, MacOS, Linux)
- 4 ) Birden çok tarayıcı ile çalışır.  
Multiple browsers (Edge, Safari, Chrome, Firefox vs.)

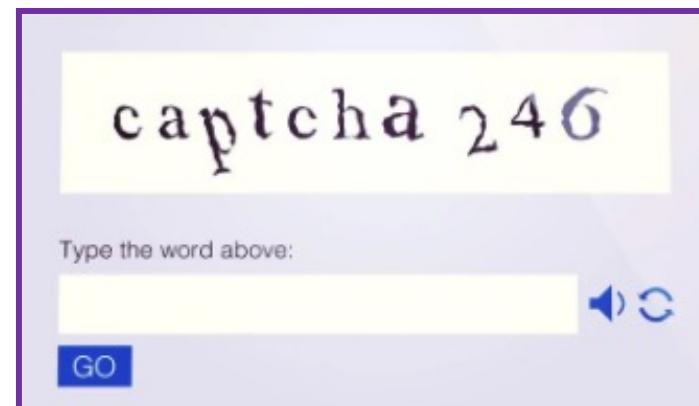
Programlama bilgisi gerektirir (Biz Java biliyoruz)

Yalnızca web tabanlı uygulamaları test eder

Profesyonel desteği sahip değil

performans testleri yapamaz

Captcha'yı asamaz(düger tüm otomasyon araçları gibi)



# Framework Nedir ?

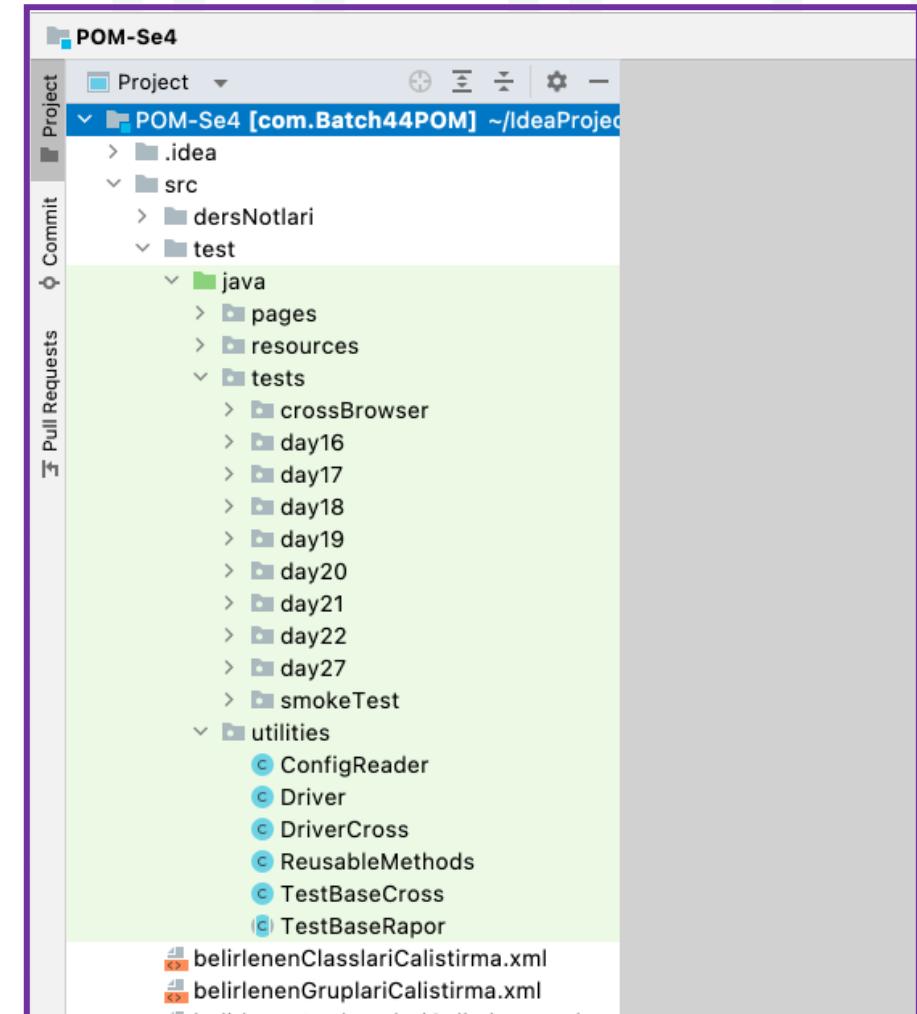
Framework, uzerine kodlarimizi yazarak projelerin olusturulabilecegi bir yapidir.

Test otomasyonu yaparken herseye sifirdan baslayip, herseyi sifirdan kurgulayip olusturmak yerine,

Herkesin anlayabilecegi, test olusturma ve gozden gecirme sureclerini kolaylastirmak, tum ekibin ortak calisabilecegi bir yapı olusturmak icin test framework'lari olusturulmustur.

Framework, test yapmak icin kullandigimiz tum enstrumanlari birlestirir.

Framework ile UI, API ve Database testleri yapılandırılabilir.



# Jar Dosyaları ile Selenium Kurulumu

- 1) <https://www.selenium.dev/downloads/> adresine gidin
- 2) Selenium Client & WebDriver Language Bindings altında Java driver'ini download edin
- 3) Browsers altında Chrome documentation linkini tıklayalım

Chrome'un kendi sayfasına gidip Current stable release'i tıklayıp size uygun olani download edin  
Indirilen surum ile bilgisayarınızdaki Chrome browser surumunun aynı olduğundan emin olun
- 4) src altında resources director'si olusturun
- 5) Bu klasor altında drivers ve libraries klasorleri olusturun
- 6) Indirdigimiz chromedriver'i drivers klasorune, selenium-java dosyasını ise libraries klasorune cikartın
- 7) intelliJ 'de yeni project / package / class olusturalım ve class icinde main method olusturun
- 8) File/Project Structure/Modules/Dependencies kismindan jar dosyalarini yukleyin

# WebDriver Objesi Olusturma

Selenium jar dosyaları ile projeye eklendiğinde, kullanmak istenen tüm browser'ların driver'larının da projeye eklenmesi gerekmektedir.

Kullanılacak browser'a ait driver projeye eklendikten sonra, her class'da bilgisayardaki browser'i yönetecek bir WebDriver objesi oluşturulur ve o obje yardımıyla WebDriver Class'ındaki hazır method'lar kullanılabilir.

WebDriver objesi oluşturmak ve objeye kullanılacak browser'a uygun değeri atamak için main method içerisinde

- 1) Java'daki setProperty(" webdriver.chrome.driver ", " driverPath"); ile sistem ayarları yapılır.

```
System.setProperty("webdriver.chrome.driver" , "src/driver/chromedriver"); /MAC
```

```
System.setProperty("webdriver.chrome.driver","src/driver/chromedriver.exe"); \\WINDOWS
```

- 2) WebDriver driver= new ChromeDriver( ); ile webdriver objesi oluşturulur ve istenen browser'a uygun değer ataması yapılır.

# WebDriver Objesi Kullanma

Selenium ile otomasyon yapabilmenin ilk adımı WebDriver Class'ından obje olusturmaktr.

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class C01_DriverMethods {

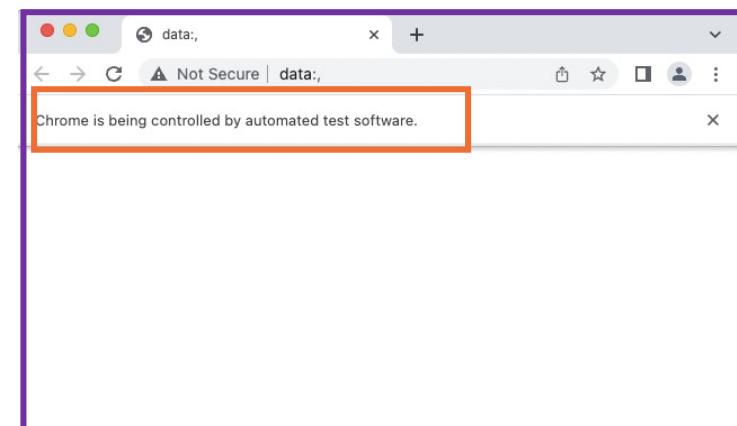
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "src/resources/chromedriver");

        WebDriver driver= new ChromeDriver();
```

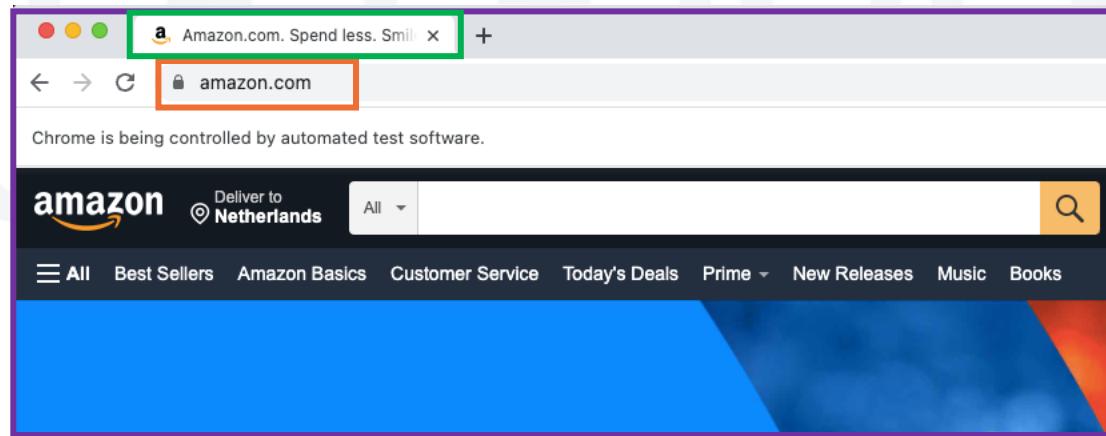
Ilgili ayarları yapıp bir driver objesi olusturdugumuzda, Selenium bu driver objesi sayesinde otomasyon yapabilecegimiz bir browser acar.

Bu browser'un Selenium tarafından kontrol edildigi yazılıdır.

Chrome disinda bir browser kullanilacaksa, o browser'in driver'ini da projeye eklenmeli ve class icindeki ayarlarda o driver'in dosya yolu driver'a gosterilmelidir.



# driver.get...() Method'lari



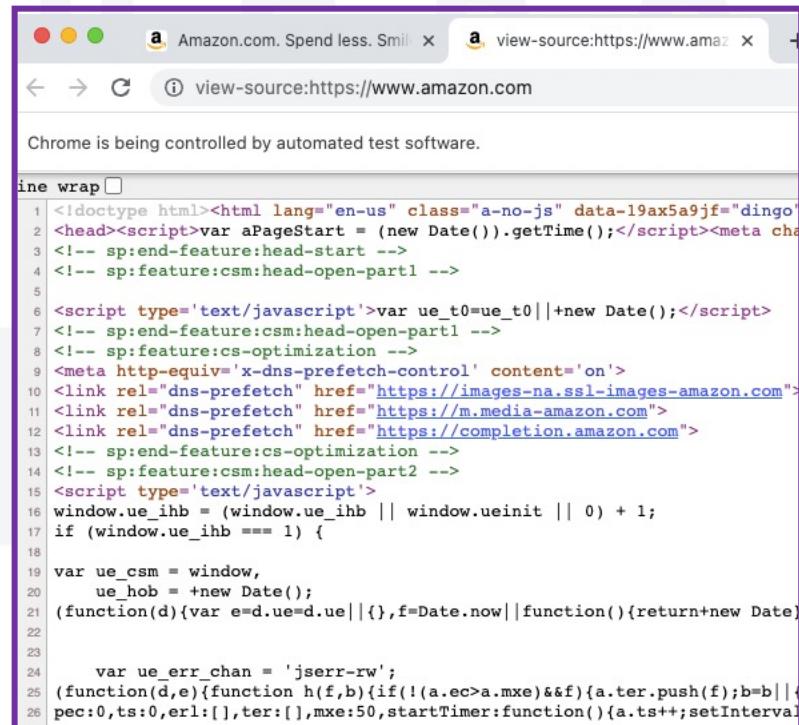
- 1- `driver.get("https://www.amazon.com");` driver'i istenen url'e goturur.
- 2- `driver.getCurrentUrl();` Gidilen Web sayfasinin URL bilgisini döndürür.
- 3- `driver.getTitle();` Gidilen Web sayfasinin title (baslik) bilgisini döndürür.
- 4- `driver.close();` Acilmis olan driver'i kapatir
- 5- `driver.quit();` Test sirasinda birden fazla window acilmissa, tumunu kapatir.

## driver.get...() Method'lari

## 6- driver.getPageSource()

Gidilen sayfanın kaynak kodlarını döndürür.

Sayfa kaynak kod'lari cok test otomasyonunda cok kullanilmaz, sadece ozel olarak bu kodlarda bir kelimenin var olup olmadigi gibi ozel bir test istenirse sayfa kodlari String olarak kaydedilip, istenen arama yapilir.



7- driver.getWindowHandle()

CDwindow-8C07925B8CBA4C8EE3039E660C30DDA1

Acilan window'a isletim sistemi tarafindan verilen unique bir deger olan **window handle** degerini döndürür.

8- driver.getWindowHandles()

Test sırasında driver birden fazla window actiysa , bir **Set** olarak acilan tum window'larin **window handle** degerlerini döndürür.

# İlk Test Otomasyonu

Software testi için tasarım aşamasında belirlenen sonuçlar (**Expected Result**) ile uygulamanın kendisinden alınan sonuçlar (**Actual Result**) karşılaştırılır.

Expected ve actual result birbirine eşit ise test başarılı (**Test Passed**), Expected ve actual result birbirine eşit değilse test başarısız (**Test Failed**) olarak raporlanır.

Test aşamalarının ve test sonuçlarını anlasılabilir olması, testin kısa olmasından önemlidir.

```
String expectedTitleIcerik="amazon";
String actualTitle= driver.getTitle();

// url test yapalim

if (actualUrl.contains(expectedUrlIcerik)){
    System.out.println("Url test PASSED");
}else {
    System.out.println("Url test FAILED");
    System.out.println("actual Url : " + actualUrl);
    System.out.println("Actual Url aranan " + expectedUrlIcerik + " kelimesini icermiyor");
}
```

Örneğin; gidilen sayfanın title değerinin belirli bir kelimeyi içerdigi test edilmek isteniyorsa, expected ve actual değerler kaydedilip, bir if else blogu içerisinde istenen test yapıp, sonuc yazdırılabilir.

# WebDriver Method'ları

1. Yeni bir package olusturalim : day01
2. Yeni bir class olusturalim : CO3\_GetMethods
3. Amazon sayfasina gidelim. <https://www.amazon.com/>
4. Sayfa basligini(title) yazdirin
5. Sayfa basliginin “Amazon” icerdigini test edin
6. Sayfa adresini(url) yazdirin
7. Sayfa url’inin “amazon” icerdigini test edin.
8. Sayfa handle degerini yazdirin
9. Sayfa HTML kodlarinda “alisveris” kelimesi gectigini test edin
10. Sayfayı kapatın.

# driver.navigate...() Method'lari

9- `driver.navigate().to( url: "https://www.amazon.com");`

driver'i verile URL'e götürür. driver.get( )den farkı navigate method'lari ile gidilen sayfaların back, forward gibi fonksiyonları saglayabilmektedir.

10- `driver.navigate().back();`

Gidilen web sayfasını bir önceki sayfaya döndürür.

11- `driver.navigate().forward();`

Gidilen web sayfasından navigate( ).back( ) ile bir önceki sayfaya donulmusse yeniden ilk sayfaya götürür.

12- `driver.navigate().refresh();`

Içinde olunan web sayfasını yeniler.

# driver.navigate...( ) Method'lari

1. Yeni bir Class olusturalim.C05\_NavigationMethods
2. Youtube ana sayfasina gidelim . <https://www.youtube.com/>
3. Amazon soyfasina gidelim. <https://www.amazon.com/>
4. Tekrar YouTube'sayfasina donelim
5. Yeniden Amazon sayfasina gidelim
6. Sayfayı Refresh(yenile) yapalim
7. Sayfayı kapatalim / Tüm sayfalari kapatalim

# driver.manage( )... Method'lari

13- `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));`

driver'in gittiği sayfayı açması ve orada kullanacağı her bir web elementi bulması için tanımlanan maximum bekleme süresini tanımlar.

Wait konusu ayrı bir konu olarak anlatılacak, ancak yazılan otomasyon yapılmırken, internet bağlantısı veya bilgisayarın hızı gibi sebeplerle gecikmeler yaşanması durumunda ne yapacağını net olmasının her testin basında max.bekleme süresi belirlenmelidir.

14- `driver.manage().window().maximize();`

Açılan driver'i tam sayfa yapar.

`driver.manage().window()`.... ile kullanılabilen farklı method'lar vardır ancak açılan web sayfasında tüm webelement'lerin görülebilir ve ulaşılabilir olması için her testin basında `maximize()` method'u kullanmasında fayda vardır.

# driver.manage( )... Method'lari

15- 

```
driver.manage().window().fullscreen();
driver.manage().window().maximize();
driver.manage().window().minimize();
```

Acilan driver'i onceden belirlenmis standart buyukluklere getirir.

16- 

```
driver.manage().window().setSize(new Dimension( width: 1000, height: 700 ) );
driver.manage().window().setPosition(new Point( x: 100, y: 100));
```

Acilan driver'i kullanıcının istedigi ozel olculere getirir ve istenen noktaya tasir.

17- 

```
driver.manage().window().getPosition();
driver.manage().window().getSize();
```

Acilan driver'in bulunduğu pozisyonu ve boyutlarini döndürür.

# driver.manage( )... Method'lari

1. Yeni bir Class olusturalim.C06\_ManageWindow
2. Amazon soyfasina gidelim. <https://www.amazon.com/>
3. Sayfanin konumunu ve boyutlarini yazdirin
4. Sayfayi simge durumuna getirin
5. simge durumunda 3 saniye bekleyip sayfayı maximize yapın
6. Sayfanin konumunu ve boyutlarini maximize durumunda yazdirin
7. Sayfayı fullscreen yapın
8. Sayfanin konumunu ve boyutlarini fullscreen durumunda yazdirin
9. Sayfayı kapatın

# driver.manage( )... Method'lari

1. Yeni bir Class olusturalim.C07\_ManageWindowSet
2. Amazon soyfasina gidelim. <https://www.amazon.com/>
3. Sayfanin konumunu ve boyutlarini yazdirin
4. Sayfanin konumunu ve boyutunu istediginiz sekilde ayarlayın
5. Sayfanin sizin istediginiz konum ve boyuta geldigini test edin
8. Sayfayı kapatın

Q U A I T Y U E R S E  
A C A D E M Y

# Selenium

## Ders-02

### WebElements

### Locators

unityverseacademy.com

Egitmen : Ahmet BULUTLUOZ

# WebDriver Method'lari

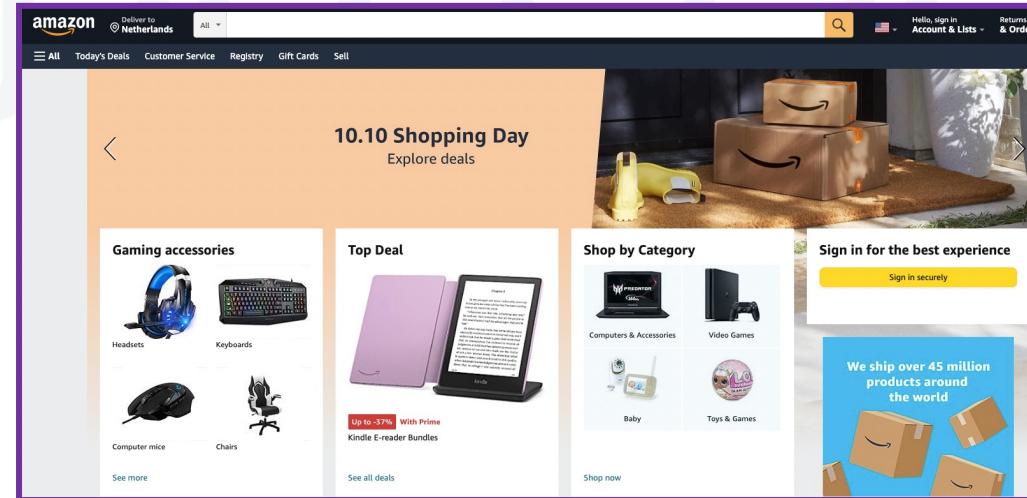
- 1.Yeni bir class olusturalim (Homework)
- 2.ChromeDriver kullanarak, facebook sayfasina gidin ve sayfa basliginin (title) “facebook” oldugunu doğrulayin (verify), degilse dogru basligi yazdirin.
- 3.Sayfa URLinin “facebook” kelimesi icerdigini doğrulayin, icermiyorsa “actual” URL’i yazdirin.
- 4.<https://www.walmart.com/> sayfasina gidin.
5. Sayfa basliginin “Walmart.com” icerdigini doğrulayin.
6. Tekrar “facebook” sayfasina donun
7. Sayfayı yenileyin
8. Sayfayı tam sayfa (maximize) yapın
- 9.Browser'i kapatın

# WebDriver Method'ları

1. Yeni bir class olusturun (TekrarTesti)
2. Youtube web sayfasına gidin ve sayfa başlığının "youtube" olup olmadığını doğrulayın (verify), eğer değilse doğru başlığı(Actual Title) konsolda yazdırın.
3. Sayfa URL'sinin "youtube" içerip içermediğini (contains) doğrulayın, içermiyorsa doğru URL'yi yazdırın.
4. Daha sonra Amazon sayfasına gidin <https://www.amazon.com/>
5. Youtube sayfasına geri donun
6. Sayfayı yenileyin
7. Amazon sayfasına donun
8. Sayfayı tamsayfa yapın
9. Ardından sayfa başlığının "Amazon" içerip içermediğini (contains) doğrulayın, Yoksa doğru başlığı(Actual Title) yazdırın.
- 10.Sayfa URL'sinin <https://www.amazon.com/> olup olmadığını doğrulayın, degilse doğru URL'yi yazdırın
- 11.Sayfayı kapatın

# WebElements

Bir web sayfasında kullanılan herseye web element denir.



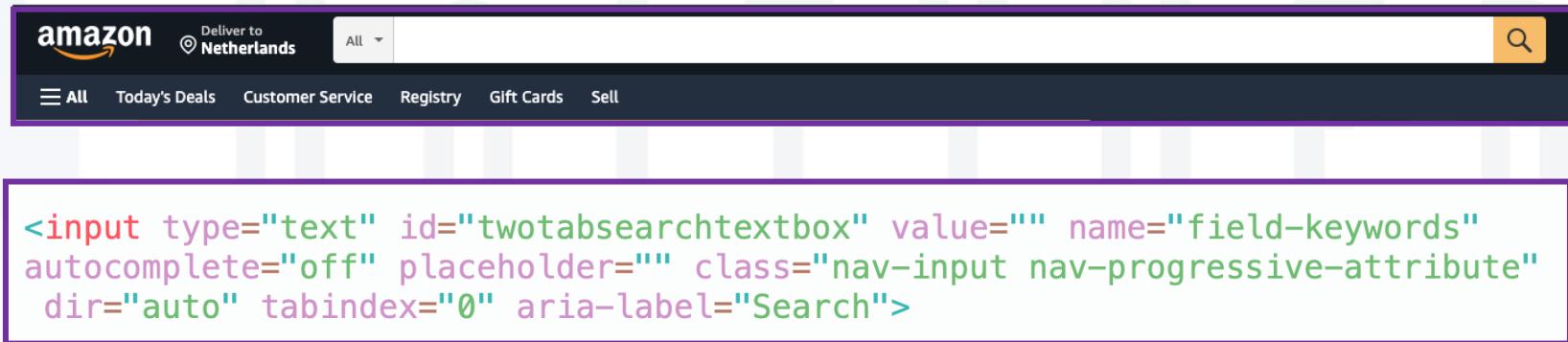
Her web element farklı özelliklerde olur. Link, açılır menu, button gibi etkilesimli web elementler olduğu gibi resim, background gibi etkilesimsiz web elementler de olur.

Görünən her web element aslında developer'lar tarafından yazılan bir HTML kodun görselleştirilmiş halidir.

Selenium WebDriver görsel elementleri değil, HTML kodları kullanır.

Otomasyon sırasında kullanılmak istenen web elementler HTML kodları kullanılarak unique olarak WebDriver'a tarif edilmelidir.

# WebElements



Her bir web element yapısına uygun olarak farklı tag ve attribute'ler bulundurur.

Web elementi unique olarak tarif edebilmek için tag ve attribute'ler tekil olarak kullanılabilir.

Tekil kullanım unique tarif için yeterli olmazsa, birden fazla bilginin kombinasyonu kullanılır.

**Tag :** input

**Attributes :** type, id, value, name, autocomplete, placeholder, class, dir, tabindex, aria-label

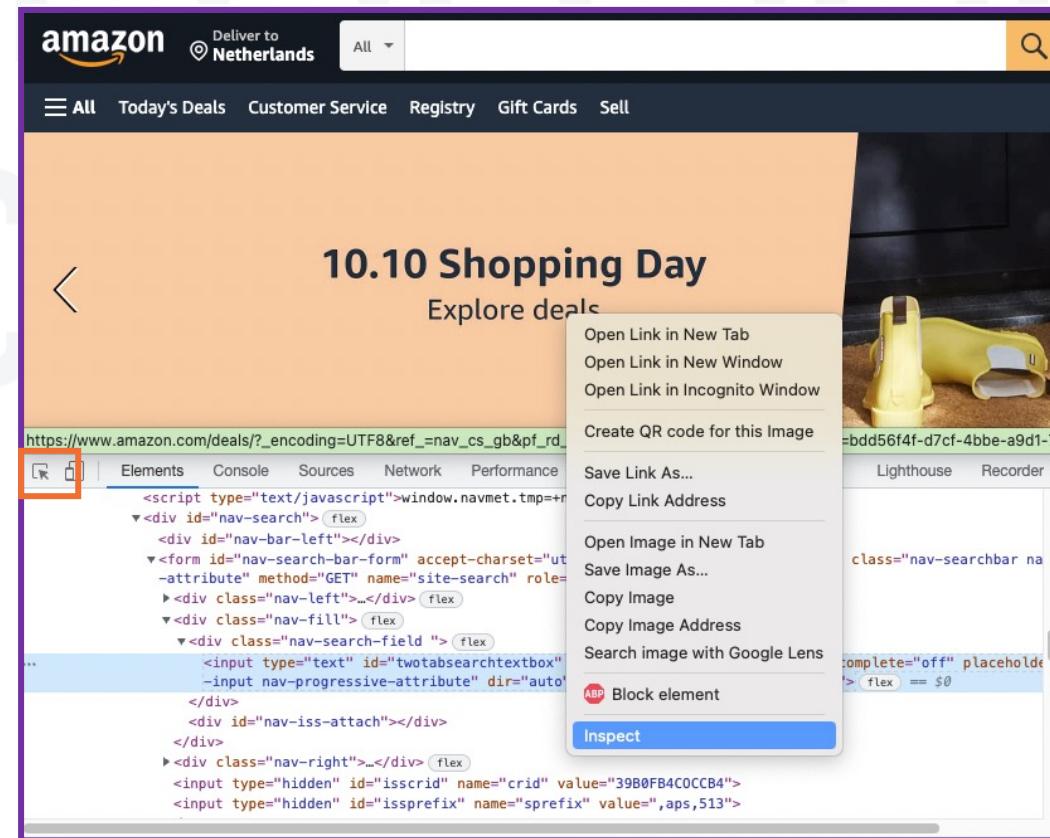
# WebElements

Web sayfasında HTML kodlarını görebilmek için mouse'da sağ click yapıp inspect/incele seçilmelidir.

Istenen web elementin HTML kodunu bulmak için, o element üzerinde yeniden sağ click yapıp inspect denebilir,

Veya menudeki → işaretini seçip, mavi iken mouse ile istenilen element seçilebilir.

HTML kodları açık iken ctrl+f tuslarına basılıncaya acılan bölümde webelement'in özellikleri aratılırsa, o özellikte kaç webelement bulunduğu görülebilir.



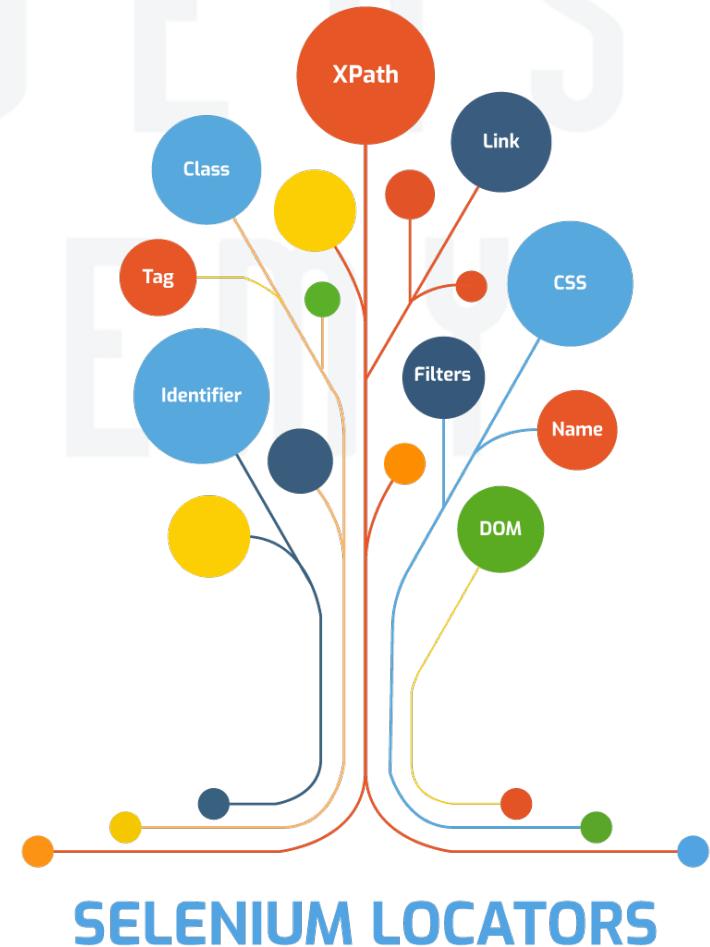
# Locators

Selenium LOCATORS, web sayfasındaki web öğelerini tanımlamak için kullanılır.

Selenium'da; metin kutuları, onay kutuları, linkler, radyo butonları, liste kutuları ve diğer **tüm web öğeler** üzerinde eylemler gerçekleştirmek için **LOCATORS'a** ihtiyacımız vardır.

Konum belirleyiciler bize web elementleri tanımlamada yardımcı olur.

Web Elementlerine ulaşmak için tag veya bazı attribute'lerin kullanıldığı 6 adet locators bulunur, bunlarla ulaşamayan webelementleri için özel olarak tanımlanan Xpath ve css locator'lari kullanılır.



# Locators



```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search">
```

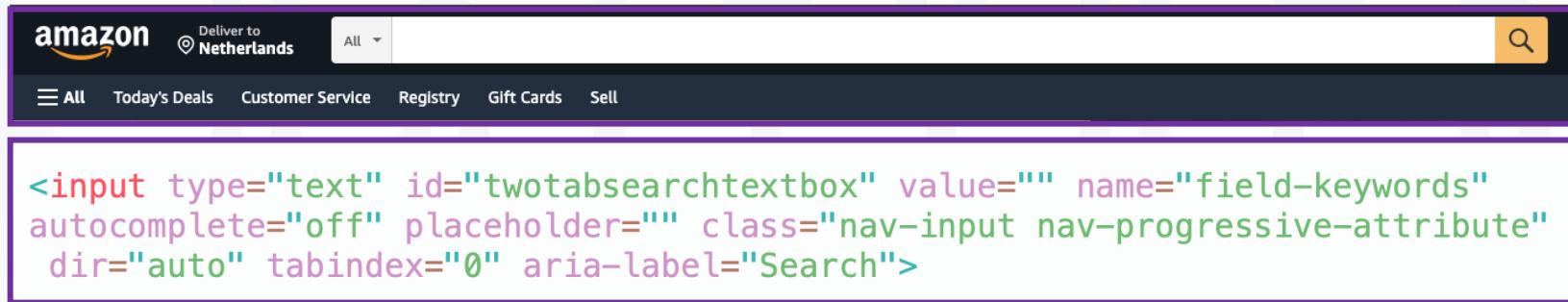
## 1- By.id("uniqueld")

Web elementi tanımlamak için ilk bakacagımız locator id olabilir.

Id genellikle unique olduğu için locate etmekte sıkça kullanılır. Ancak developer'ların aynı id ile birden fazla webelementi tanımlayabilecekleri de unutulmamalıdır.

Hangi locator kullanılırsa kullanılsın, web sayfasının HTML kodlarında locator aratılarak, unique sonuca ulaşıldığı gözlemlenmelidir.

# driver.findElement( ) Method'u



Unique locator'i tespit edilen web element kullanilmak icin, driver objesi ile LOCATE edilip, WebElement class'indan olusturulan objeye atanmalidir.

**Driver.findElement(By.locator ("uniqueLocatorDegeri"))**

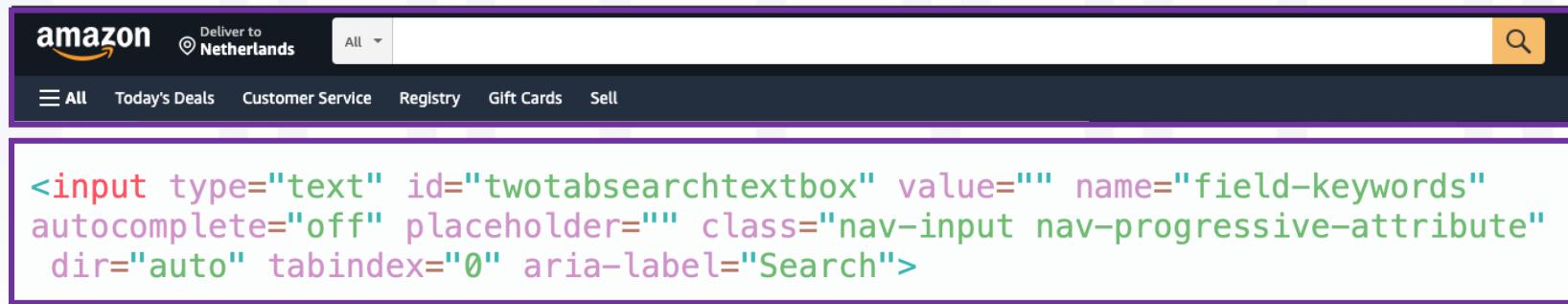
```
WebElement amazonAramaKutusu = driver.findElement(By.id("twotabsearchtextbox"));
```

Driver, findElement( ) ile objeyi bulamazsa **NoSuchElementException** verir.

findElement( ) ile locate edilip, objeye atanen webElement testler sirasinda kullanilabilir.

webElement bir obje oldugu icin direkt yazdirilamaz, hazir method'lar kullanilarak manipule edilebilir.

# WebElement Objesi Olusturma



## Amazon Arama Testi

- 1- Bir test class'i olusturun ilgili ayarlari yapin
- 2- <https://www.amazon.com> adresine gidin
- 3- amazon arama kutusunu locate edin
- 4- arama kutusuna “Nutella” yazdirin
- 5- arama islemini yapabilmek icin ENTER tusuna basin

# Locators



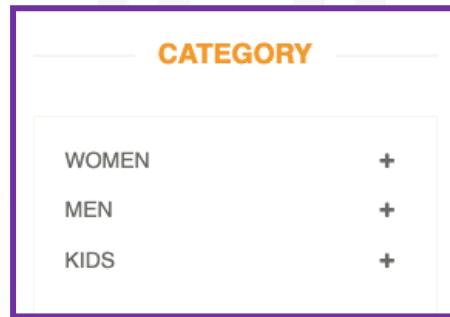
```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search">
```

## 2- By.name("uniqueName")

WebElement'in HTML kodlarında name attribute'u varsa ve unique ise locate etmek için By.name( ) kullanılır

```
WebElement amazonAramaKutusu= driver.findElement(By.name("field-keywords"));
```

# Locators



## 3- By.className("uniqueClassName")

class attribute'u genellikle benzer ozellikleri barindiran web elementleri gruplandirmak icin kullanilir.

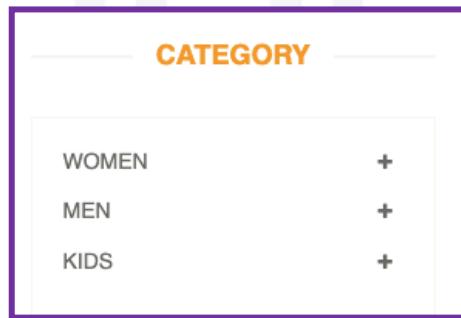
Bu sebeple class attribute'u ile yapacagimiz locate islemleri genellikle 1 web element degil, birden fazla element döndürür.

bu elementleri store edebilmek icin bir web element degil, web elementlerden olusan bir list gereklidir.

**NOT :** class value'sunde bosluk (space) varsa By.className ile locate islemlerinde sorunlar yasanabilir.

```
<div class="panel-group category-products" id="accordian">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">...</h4>
    </div>
    <div id="Women" class="panel-collapse collapse">...</div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">...</h4>
    </div>
    <div id="Men" class="panel-collapse collapse">...</div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">...</div>
    <div id="Kids" class="panel-collapse collapse">...</div>
  </div>
<div class="brands_products">
```

# driver.findElements( ) Method'u



driver.findElements(.....)

Locator'a uygun tüm web elementlerini döndürür.

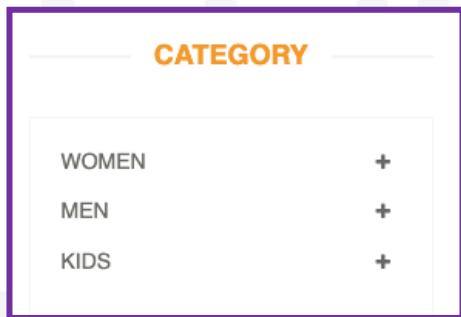
```
<div class="panel-group category-products" id="accordian">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">...</h4>
    </div>
    <div id="Women" class="panel-collapse collapse">...</div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">...</h4>
    </div>
    <div id="Men" class="panel-collapse collapse">...</div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">...</h4>
    </div>
    <div id="Kids" class="panel-collapse collapse">...</div>
  </div>
<div class="brands_products">
```

findElements( ) birden fazla web element döndürebileceği için dönen elementleri store etmek için bir list kullanılmalıdır.

Locator'a uyan hicbir webelement olmasa da exception olusmaz, bos bir list olusur.

List'teki tüm elementler web element oldugu için direkt yazdırılamaz, bir for-each loop kullanılarak elementlere istenen işlemler yapılabilir.

# Locators



## Automation Exercise Category Testi

- 1- Bir test class'i olusturun ilgili ayarları yapın
- 2- <https://www.automationexercise.com/> adresine gidin
- 3- Category bolumundeki elementleri locate edin
- 4- Category bolumunde 3 element olduğunu test edin
- 5- Category isimlerini yazdırın
- 6- Sayfayı kapatın

# Locators

## 4- By.tagName("tagName")

```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords"  
autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute"  
dir="auto" tabindex="0" aria-label="Search">
```

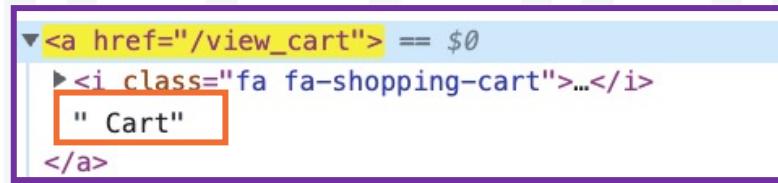
```
List<WebElement> inputTagList =driver.findElements(By.tagName("input"));
```

Bir web sayfasında herhangi bir tagName'in unique olması nadiren karsilasilabilir bir durumdur.

Tag ismi ile yapılan locate'ler unique bir elemente ulasmaktan daha çok sayfadaki tüm link'leri bulmak gibi amaclarla kullanılabilir.

Birden fazla web element döndüreceği için driver.findElements(..) ile kullanılması daha çok karşılaşılan bir durumdur.

# Locators



```
<a href="/view_cart" == $0
  ><i class="fa fa-shopping-cart">...</i>
    " Cart"
</a>
```

5- By.linkText("linkYazisininTamami")

6- By.partialLinkText("linkYazisininBirBolumu")

Sadece link'ler icin kullanilabilirler.

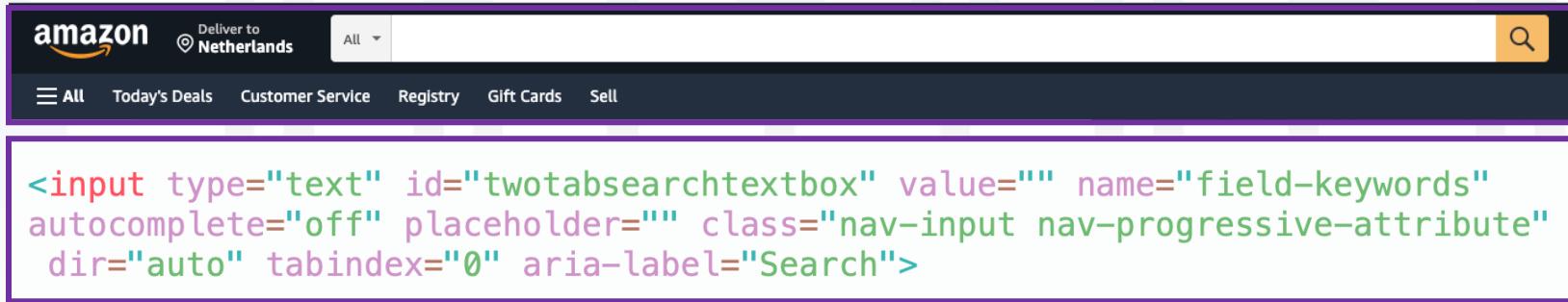
Her link uzerinde bulunan yazi kullanilarak locate yapmamizi saglar.

Link uzerinde bulunan yazi String data turunde oldugundan case sensite'dir.

By.linkText ( ) icin bosluklar da dikkate alınarak tum metin yazılmalıdır.

Tum metnin yazılamaması, yazının kısmı olarak kullanılması isteniyorsa By.partialLinkText ( ) kullanılmalıdır.

# WebElement Method'lari



Bir web element'i locate etmek her testin vazgecilmez bir adimidir.

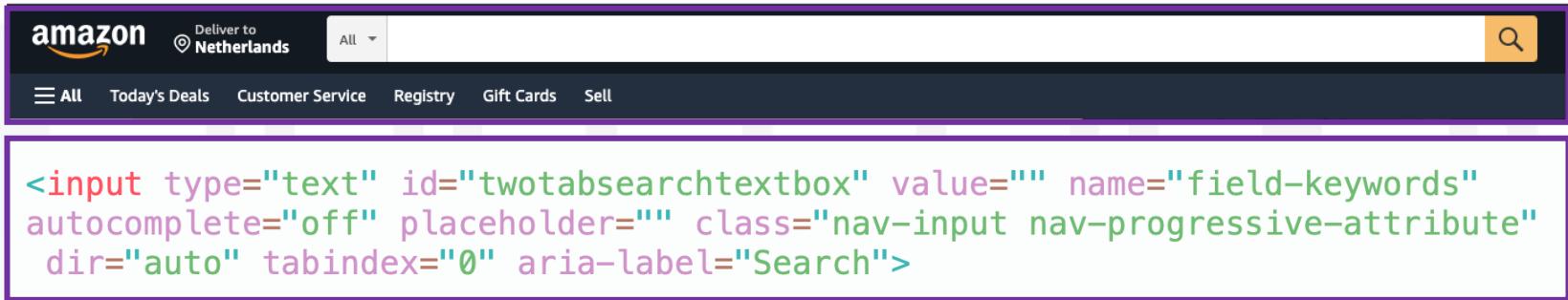
Webelement'i locate ettikten sonra variable atamak veya atamadan direk kullanmak test adimlarina ve belirlenen genel test stratejisine baglidir.

Webelement ile yapabilecegimiz islemler icin hazir method'lari kullaniriz.

1- `webElement.click();` Web element'e click yapar.

2- `webElement.sendKeys( ...keysToSend: "Istenen Metin");` Web element'e istenen metni yollar

# WebElement Method'lari



3- `webElement.submit();` Web element ile işlem yaparken ENTER tusuna basma işlemini yapar.

4- `webElement.sendKeys( ...keysToSend: "Istenen Metin" + Keys.ENTER);`

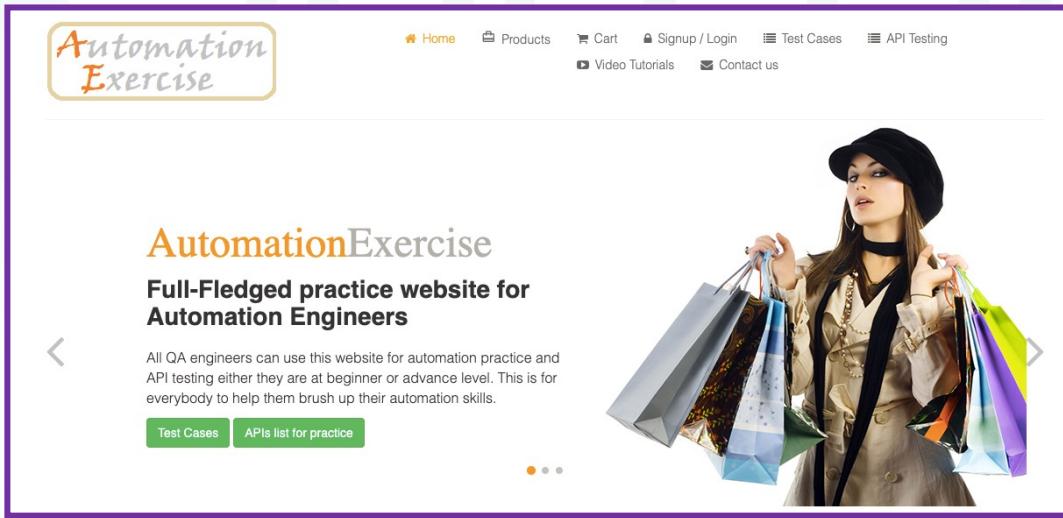
Web element'e istenen metni yollayıp, sonra ENTER tusuna basar

5- `webElement.isEnabled();` Web element erişilebilir ise true, yoksa false döner.

6- `webElement.isDisplayed();` Web element gorunuyor ise true, yoksa false döner.

7- `webElement.isSelected();` Web element secili ise true, yoksa false döner.

# Locators



## Automation Exercise Link Testi

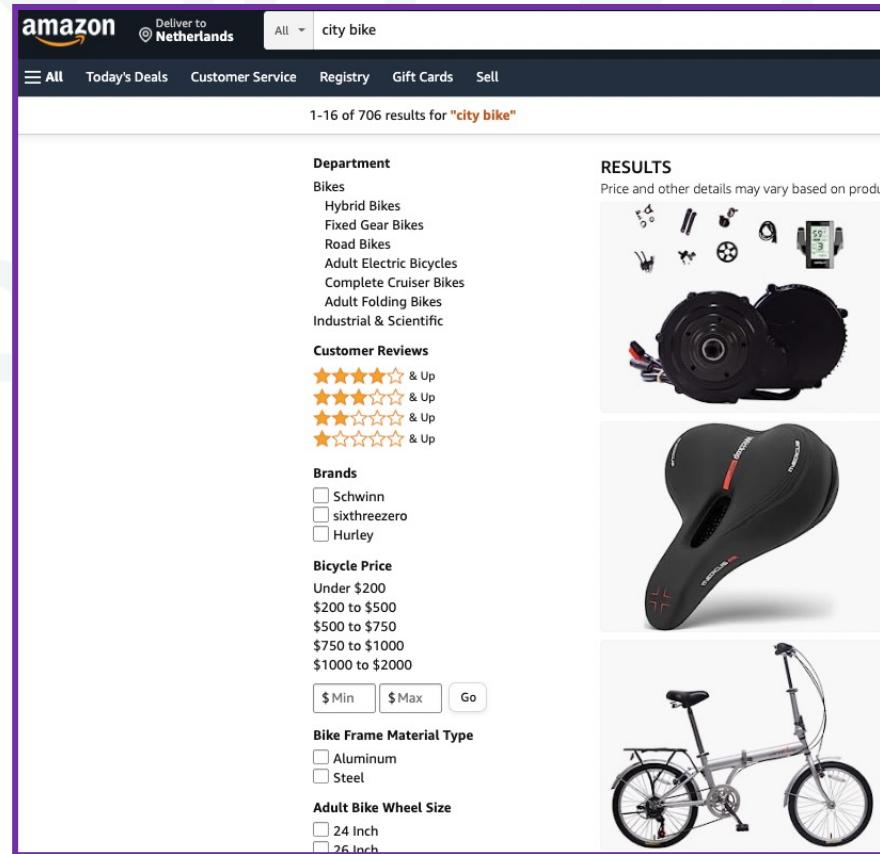
- 1- Bir test class'i olusturun ilgili ayarlari yapin
- 2- <https://www.automationexercise.com/> adresine gidin
- 3- Sayfada 147 adet link bulundugunu test edin.
- 4- Products linkine tiklayin
- 5- special offer yazisinin gorundugunu test edin
- 6- Sayfayi kapatin

# Locators

	<code>findElement( )</code>	<code>findElements( )</code>
websayfasında birden fazla Web Element Locator ile uyusursa	İlk elemani dondurur	Tüm elemanları ondurur
websayfasında hiçbir Web Element Locator ile uyuşmazsa	NoSuchElementException fırlatır	Exception fırlatmaz, boş bir liste dondurur
Return Type	<code>WebElement</code>	<code>List&lt;WebElement&gt;</code>
Elemana erişim	Direk ulaşılabilir	Liste'den index veya iterator ile ulaşılabilir

# Locators

- 1- <https://www.amazon.com/> sayfasına gidin.
- 2- Arama kutusuna “city bike” yazip aratin
- 3- Görüntülenen sonuçların sayısını yazdırın
- 4- Listeden ilk ürünün resmine tıklayın.



# Locators - Xpath

Bir WebElement'i locate etmek icin kullanabilecegimiz en etkin yöntemdir.

```
WebElement webElement= driver.findElement(By.xpath( xpathExpression: "bulunan xpath"));
```

Onceki 6 locator, HTML element olusturulurken developer'in yazdigı kodlara göre yapılır.

Ornegin; HTML element'de id attribute'u varsa By.id( ) method'u kullanabilir ama developer id attribute'u koymedi ise kullanamayız.

Aynı şekilde HTML elementi bir link ise By.linkText( ) veya By.partialLinkText( ) kullanabiliriz, link degilse kullanamayız.



Xpath de HTML kodu kullanır ancak farklı kombinasyonlar kullanıldığı için dinamiktir ve her webelement için mutlaka bir xpath bulunabilir.

2 çeşit Xpath yazılabilir

1. **Absolute** xpath (mutlak)
2. **Relative** xpath (bağılı)

# Locators

HTML kodlarda Parent – Child – Sibling ilişkisi

```
▼<div class="navFooterLine navFooterLinkLine navFooterDescLine" role="navigation" aria-label="More on Amazon.com">
  ▼<table class="navFooterMoreOnAmazon" cellspacing="0">
    ▼<tbody>
      ▼<tr>
        ▶<td class="navFooterDescItem">..</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        ▼<td class="navFooterDescItem">
          ▼<a href="https://advertising.amazon.com/?ref=footer_advtsing_amzn_com" class="nav_a">
            "Amazon Advertising"
            <br>
            ▶<span class="navFooterDescText">..</span>
          </a>
        </td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        ▶<td class="navFooterDescItem">..</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        ▶<td class="navFooterDescItem">..</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
```

Her HTML sayfasi <Html> </Html> taglari arasina yazilir.

Bir HTML tag'inin arasina yeni bir tag acildiginda konum olarak bir tab icerden baslar.

HTML tag'inin dusey hizasina bakarak parent-child-sibling ilişkisi anlasilabilir.

# Locators

## 1. Absolute Xpath

```
<div class="navFooterLine navFooterLinkLine navFooterDescLine" role="navigation" aria-label="More on Amazon.com">
  <table class="navFooterMoreOnAmazon" cellspacing="0">
    <tbody> // div/ table/ tbody
      <tr>
        <td class="navFooterDescItem">...</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        <td class="navFooterDescItem">
          <a href="https://advertising.amazon.com/?ref=footer_advtsing_amzn_com" class="nav_a">
            "Amazon Advertising"
            <br>
            <span class="navFooterDescText">...</span> // tbody / tr / td[3] // span
          </a>
        </td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        <td class="navFooterDescItem">...</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        <td class="navFooterDescItem">...</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
      </tr>
    </tbody>
  </table>
</div>
```

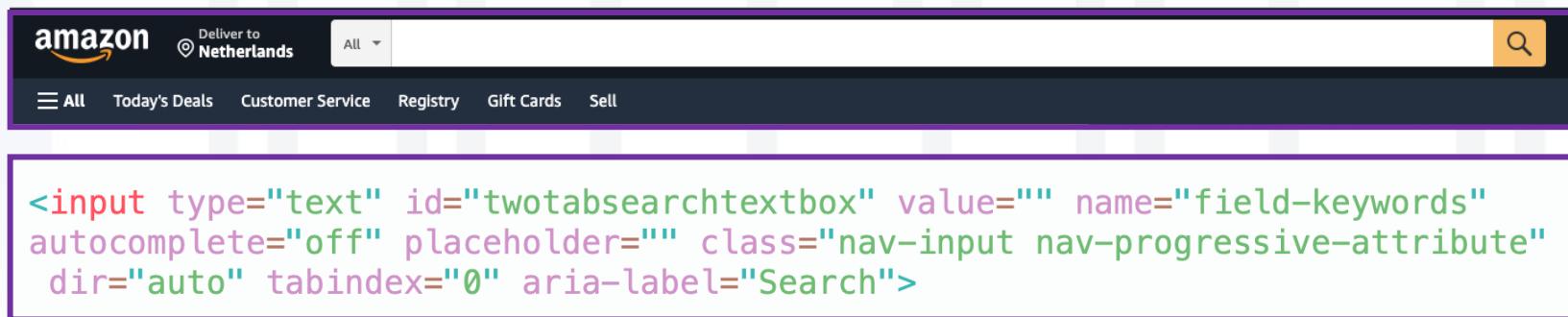
Absolute xpath yazmak icin en basa // sonraki her adimda / yazarak hedef web element'e kadar tum tag'lar yazilir.

Eger ayni path'e sahip birden fazla element varsa index kullanilabilir. [2] gibi

Eger bir parent'in grand child'lari icinde unique bir tag varsa parent // grand child yazilabilir

# Locators

## 2.Relative Xpath



Bir web element'in 3 bileseni bulunur.

1- Tag : input

2- Attributes : type, id, value, name, autocomplete, placeholder, class, dir, tabindex, aria-label

3- Attribute Values : text, twotabsearchtextbox, field-keywords .....

Relative Xpath bu 3 bilesenin belirlenen sekilde birlikte kullanilmasi ile olusur. Her Xpath ile unique bir sonuc elde edilemeyebilir ancak unique bir deger mutlaka bulunur.

//tagName[@attributelsmi='attributeValue']

Q U A I T Y U E R S E  
A C A D E M Y

Selenium

Ders-03

Locators

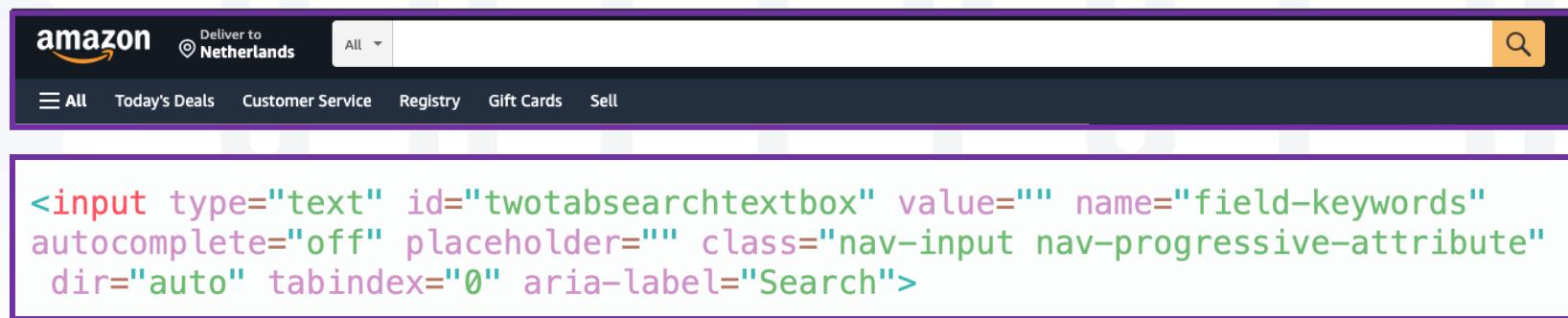
JUnit Framework

unityverseacademy.com

Egitmen : Ahmet BULUTLUOZ

# Locators

## 2.Relative Xpath



Unique deger icin 3 bilesenin tamami kullanilmak zorunda degildir.

Hedef unique olarak webelement'i locate etmektir. Daha az bilesenle de unique degere ulasabilen Xpath'ler de kullanilabilir.

```
driver.findElement(By.xpath( xpathExpression: "//input" ));
```

Sadece tag ismi ile

```
driver.findElement(By.xpath( xpathExpression: "// * [@type='text']" ));
```

tag ismi farketmeksizin

```
driver.findElement(By.xpath( xpathExpression: "//input[@ *= 'text'" ));
```

Attribute farketmeksizin

```
driver.findElement(By.xpath( xpathExpression: "//input[@type]" ));
```

Attribute value farketmeksizin

```
driver.findElement(By.xpath( xpathExpression: "//div[@id='logo' or class='flex-col logo' ] "));
```

```
driver.findElement(By.xpath( xpathExpression: "//div[@id='logo' and class='flex-col logo' ] "));
```

# Locators

## 2. Relative Xpath

Link disindaki bazi webelementler'inde de text bulunabilir.

Bu text'ler o webelement'e ozel oldugu icin unique bir xpath elde etmek icin kullanisli olabilirler.

Text ile locate yazmak icin kullanılan genel syntax :

```
//tagName[text()='yazinin tamami']
```

Genel xpath kullanimina uygun olarak tagname veya attribute ismi yazilmadan da text ile xpath yazilabilir.

```
//tagname[.='yazinin tamami']
```

```
//*[@.='yazinin tamami'] "
```

Metnin sadece bir kismi kullanilacaksa

```
//*[@contains(text(),'yazinin bir bolumu')] "
```

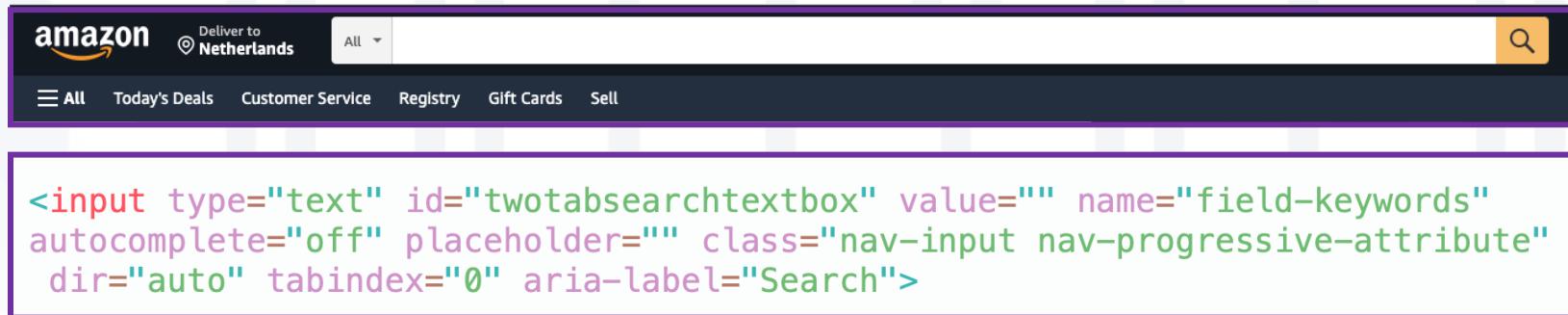
# Locators

## Relative Xpath Soru

- 1- [https://the-internet.herokuapp.com/add\\_remove\\_elements/](https://the-internet.herokuapp.com/add_remove_elements/) adresine gidin
- 2- Add Element butonuna basin
- 3- Delete butonu'nun gorunur oldugunu test edin
- 4- Delete tusuna basin
- 5- “Add/Remove Elements” yazisinin gorunur oldugunu test edin

# Locators

## 8.cssSelector



cssSelector de xpath'e benzer bir kullanıma sahiptir. Tag ismi, attribute ismi ve attribute value ile yapılacak kombinasyonlarla olusturulur.

```
driver.findElement(By.cssSelector("input[id='twotabsearchtextbox']"));
```

Hedef unique olarak WebElement'i locate etmektir. Daha az bilesenle de unique degere ulaşılabilen Xpath'ler de kullanılabilir.

cssSelector özellikle class ve id attribute'leri ile kısa şekilde yazılabilir

```
driver.findElement(By.cssSelector("#twotabsearchtextbox")); Id attribute ile
```

```
driver.findElement(By.cssSelector(".nav-input nav-progresive-attribute")); Class attribute ile
```

# Tekrar Sorusu

- 1- bir class olusturun
- 2- <https://www.amazon.com/> adresine gidin
- 3- Browseri tam sayfa yapin
- 4- Sayfayı "refresh" yapın
- 5- Sayfa basliginin "Spend less" ifadesi icerdigini test edin
- 6- Gift Cards sekmesine basin
- 7- Birthday butonuna basin
- 8- Best Seller bolumunden ilk urunu tiklayin
- 9- Gift card details'den 25 \$'i secin
- 10-Urun ucretinin 25\$ oldugunu test edin
- 11-Sayfayı kapatın

# Relative Locators

Relative Locators nedir ?

Bir web elementi direk locate edemedigimiz durumlarda gunluk hayatimizda kullandigimiz sekilde o web elementi etrafindaki web elementlerin referansı ile tarif edebiliriz.

Ornegin yandaki resimde Berlin icin bir cok relative locator tanimlayabiliriz.

- Boston'in saginda , Sailor'in ustunde
- NYC'nin altinda, Bay Area'nin solunda
- Boston yakinalarinda Bay Areanin solunda ve Toronto'nun saginda vb..



Bu ozellik Selenium 4 ile gelen yeniliklerden biridir.

<https://www.diemol.com/selenium-4-demo/relative-locators-demo.html>

# Relative Locators

## Class Work: Relative Locators

- 1 ) <https://www.diemol.com/selenium-4-demo/relative-locators-demo.html>  
adresine gidin
- 2 ) Berlin'i 3 farkli relative locator ile locate edin
- 3 ) Relative locator'larin dogru calistigini test edin



# Relative Locators

```
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(with(By.tagName("li")).above(sailor).toRightOf(boston));

WebElement mountie=driver.findElement(with(By.className("ui-li-has-thumb")).below(boston));
```



# Maven



Apache Maven yazılım projelerinin kolay anlasılmasına ve yönetilmesine odaklanmış bir tool'dur.

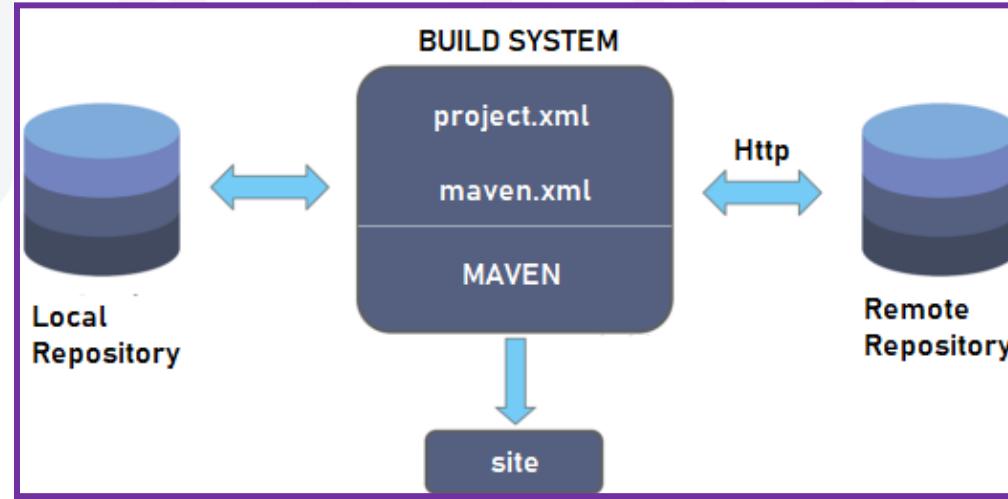
Proje nesne modeli (POM) konseptine dayalı olarak Maven, bir projenin inşasını, raporlamasını ve dokümantasyonunu merkezi bir bilgi parçasından(dependency) yönetebilir.

- 1- Projeleri oluşturma için standart belirleme,
- 2- projenin nelerden olduğunu net olarak tanımlama,
- 3- proje bilgilerini yayılamanın kolay bir yolunu bulma
- 4- JAR dosyalarını farklı projeler arasında paylaşma

Amaçları çerçevesinde başlayan bir çalışma sonucu ortaya çıkmıştır.

Sadece Java tabanlı projeleri oluşturmak ve yönetmek için kullanılabilecek bir araçtır.

# Maven



Maven bir Java olusturma aracıdır (**build tool**). Maven proje otomasyon ve yönetim aracıdır (**automation and management tool**).

Maven, konfigürasyon için pom.xml dosyasını kullanır.

pom.xml projenin insası, raporlaması ve dokümantasyonu için gerekli bütün bilgileri içerir ( dependencies , plugins v)

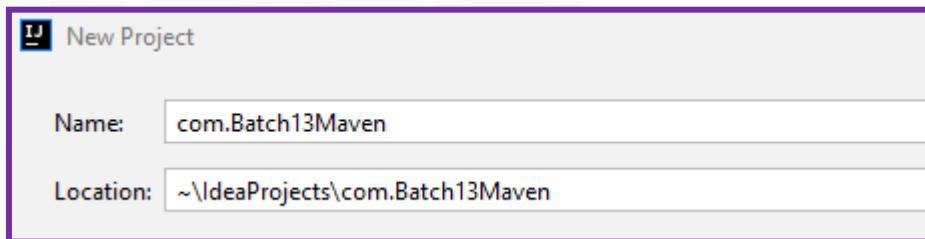
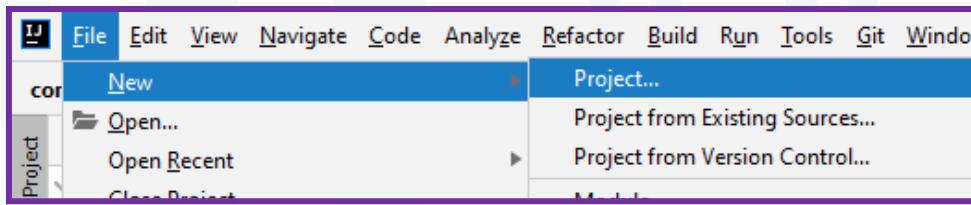
Maven ile çalışan tüm projeler aynı konfigürasyonu kullandığı için yeni bir projede çalışmaya başlandığında projenin anlaşılması çok kolay olacaktır.

# Neden Maven ?

- 1- Proje yönetiminde oluşturma, belgeleme, yayınılama ve dağıtım gibi tüm süreçlerin yönetilmesine yardımcı olur.
- 2- Projenin ve yapım sürecinin performansını artırır.
- 3- Jar dosyalarını ve diğer bağımlılıkları (dependencies) indirme görevi otomatik olarak yapılır
- 4- Gerekli tüm bilgilere kolay erişim sağlar
- 5- Geliştiricinin, bağımlılıklar, süreçler vb. hakkında endişelenmeden farklı ortamlarda bir proje oluşturmasını kolaylaştırır.
- 6- Open source olduğundan ücretsizdir, geniş bir kullanıcı tabanı olduğu için karşılaşılan sorunlara çözüm bulmakta zorluk yaşanmaz.



# Maven Proje Olusturma



2. Select Maven -> click next

Java versiyonunun bilgisayarinizdaki version ile ayni oldugunu control edin

3. Name: com.projeAdi -> click finish

EnableAutoImport sorarsa click

4. Package olusturun, name : day04

5. Classolusturun, name : FirstMavenClass

# pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>SeleniumInt</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
  </properties>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>...</dependency>

    <!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
    <dependency>...</dependency>
  </dependencies>
</project>
```

Project Object Model (pom) Maven projesinin temelini oluşturur.

pom.xml proje hakkında bazi bilgiler ve Maven tarafından projeyi olusturmak icin kullanılan konfigurasyon detaylarini gösterir.

Bir projenin hangi framework'u kullandigini anlamak icin pom.xml'e bakmak yeterlidir.

POM'da belirtilebilecek yapılandırmalardan bazıları proje bağımlılıkları(dependencies), yürütülebilecek eklentiler(plugin) veya hedefler(goal), yapı profilleri vb. Proje sürümü, açıklama, geliştiriciler (artifact id, group id, version) ve benzeri gibi diğer bilgiler de belirtilebilir.

Projeye eklenecek dependency'ler mvnrepository.com'dan bulunabilir.

Istenen dependency icin version secilirken guncellik, stabil versiyon olma ve kullanılma sayiları dikkate alınmalıdır.

# pom.xml'e Dependency Ekleme

```
<properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
</properties>

<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency...>
        <!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
        <dependency...>
    </dependencies>
</project>
```

- 1- pom.xml'de </properties> kapanis tagi ile </project> kapanis tagi arasında <dependencies> acilis ve </dependencies> kapanis tag'larini olusturun.
- 2- mvnrepositories.com adresinden WebDriverManager ve Selenium Java dependency'lerini kopyalayip, dependencies taglari arasina yapistirin
- 3- Bu dependency'leri projenize ilk defa yuklediginiz icin versiyon bolumleri kirmizi cikacaktir. Kirmiziligi gidermek icin 4. adimi takip edin.

```
<version>4.5.0</version>
```

# pom.xml'e Dependency Ekleme

- 4- projenin sag ust kisminda bulunan Maven yazisini tiklayin, acilan bolumde yenile butonuna tiklayin ve yuklediginiz dependency'lerin projenize eklendigini kontrol edin.

The screenshot shows an IDE interface with several tabs at the top: Test.java, pom.xml (SeleniumInt), C03\_linkText.java, C05\_noSuchElementExc.java, C06\_webElementMethodlari.java, C01\_Xpath.java, and C01\_... . The pom.xml tab is active. On the right side, there is a 'Maven' tool window. The 'Dependencies' section is expanded, showing two entries: org.seleniumhq.selenium:selenium-java:4.4.0 and io.github.bonigarcia:webdrivermanager:5.3.0. A red box highlights the 'Dependencies' section in the tool window and the corresponding XML code in the pom.xml file. Another red box highlights the 'SeleniumInt' project name in the tool window and its corresponding XML code in the pom.xml file. The XML code in the pom.xml file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/pom-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>SeleniumInt</artifactId>
    <version>1.0-SNAPSHOT</version>

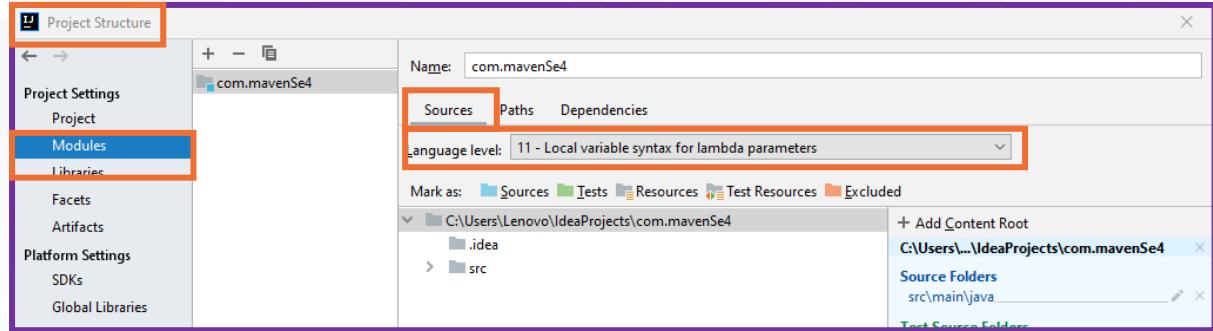
    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>4.4.0</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
        <dependency...>
    </dependencies>

```

# pom.xml'e Dependency Ekleme



File  
Project Structure  
Modules  
Sources

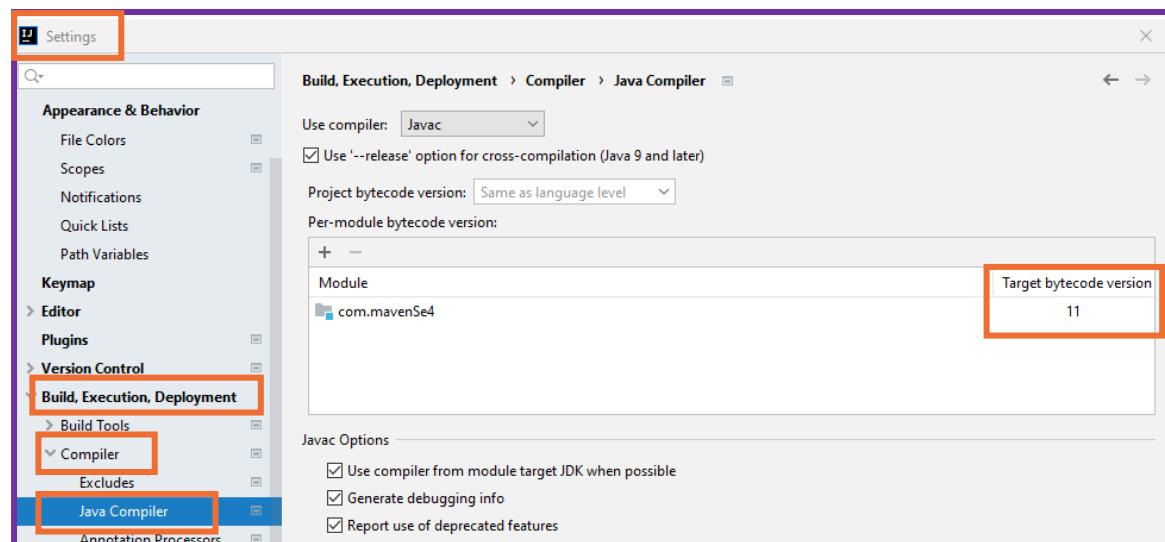
Language level : min 8 yapın, bilgisayarınızda kurulu java 8 veya üstü ise java versiyonu aynı olmalı

File  
Settings  
Build,Execution,Deployment

Compiler

Java Compiler

Target bytecode version : min 8 yapın,  
bilgisayarınızda kurulu java 8 veya üstü ise  
java versiyonu aynı olmalı



# Maven WebDriver Olusturma

Maven ile Selenium Java ve WebDriverManager dependency'lerini projemize eklendiği için, her seferinde driver.exe dosyasını driver'a tanıtmaya mecburiyeti kalmadı.

```
System.setProperty("webdriver.chrome.driver","src/resources/chromedriver");
WebDriver driver=new ChromeDriver();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));
driver.manage().window().maximize();
```

Bundan sonra driver ayarları yapılmırken ilk satırda sistem ayarlarını bonigarcia WebDriverManager kullanarak yapacağız.

```
WebDriverManager.chromedriver().setup();
WebDriver driver=new ChromeDriver();
driver.manage().window().maximize();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));
```

Chrome之外のブラウザを使用する場合は、1. および 2. 行で Chrome 代わりにそのブラウザの選択が有効になります。

# Maven ClassWork

## Class Work Amazon Search Test

- 1- <https://www.amazon.com/> sayfasina gidelim
- 2- arama kutusunu locate edelim
- 3- “Samsung headphones” ile arama yapalim
- 4- Bulunan sonuc sayisini yazdiralim
- 5- Ilk urunu tiklayalim
- 6- Sayfadaki tum basliklari yazdiralim

# Maven ClassWork

1. <http://zero.webappsecurity.com> sayfasina gidin
2. Signin buttonuna tiklayın
3. Login alanine “username” yazdırın
4. Password alanina “password” yazdırın
5. Sign in buttonuna tiklayın
6. Back tusu ile sayfaya donun
7. Online Banking menusunden Pay Bills sayfasina gidin
8. amount kismina yatırmak istediğiniz herhangi bir miktarı yazın
9. tarih kismina “2020-09-10” yazdırın
10. Pay buttonuna tiklayın
11. “The payment was successfully submitted.” mesajının çıktığını test edin

# Maven Tekrar Testi

- 1- C01\_TekrarTesti isimli bir class olusturun
- 2- <https://www.google.com/> adresine gidin
- 3- cookies uyarisini kabul ederek kapatın
- 4- Sayfa basliginin “Google” ifadesi icerdigini test edin
- 5- Arama cubuguna “Nutella” yazip aratin
- 6- Bulunan sonuc sayisini yazdirin
- 7- sonuc sayisinin 10 milyon’dan fazla oldugunu test edin
- 8- Sayfayı kapatın

# Maven Tekrar Testi

1. “<https://www.saucedemo.com>” Adresine gidin
2. Username kutusuna “standard\_user” yazdirin
3. Password kutusuna “secret\_sauce” yazdirin
4. Login tusuna basin
5. Ilk urunun ismini kaydedin ve bu urunun sayfasina gidin
6. Add to Cart butonuna basin
7. Alisveris sepetine tiklayin
8. Sectiginiz urunun basarili olarak sepete eklendigini control edin
9. Sayfayı kapatın

# JUnit

Java ile olusturulabilecek en temel Test Framework'udur.

Open-source bir kutuphanedir.

Developer'lar tarafından yapılan unit test'ler için de kullanılır.

Testlerimizi yaparken bize kolaylık sağlamak amacıyla oluşturulan Assert, Test, Before, After gibi bir çok class'a sahiptir.

JUnit, Maven altyapısını kullanır. JUnit framework için ihtiyacımız olan kutuphaneleri mvn.repository.com adresinden bulabileceğimiz dependency'ler ile projemize ekleyebiliriz.

JUnit'de testler için ihtiyaç duyulan pek çok olsa da TestNG next generation olarak daha fazla özellik ile piyasaya çıkmış ve piyasayı domine etmiştir.



# JUnit

## Annotations

JUnit ile Java class'larinin vazgecilmez ogesi olan main method ihtiyaci ortadan kalkar

Annotations ile compiler'a talimatlar verilebilir

Annotations kodların daha iyi anlasilabilmesi ve daha iyi bir yapı kurulabilmesi icin gerekli meta-data (basit bilgi)'lari Java'ya iletmek icin kullanılır.

Testler sirasinda en cok kullanılan Junit notasyonları

@Test

@Ignore

@Before , @After

@BeforeClass , @AfterClass



# JUnit

## @Test ve @Ignore Annotations

@Test notasyonu bir method'un bagimsiz olarak calisabilmesini saglar.

@Test notasyonu olan method'un calismasi icin Java'daki gibi method call yapilmasina ihtiyac yoktur, bagimsiz olarak calistirilabilir.

Class level'daki run butonu ile class'daki tum test method'lari da birlikte calistirilabilir.

Junit'in method'lari calistirma sirasi ongorulemez. Bunun icin her test method'u bagimsiz olarak calistirilabilir olmalıdır.

@Ignore notasyonun tanimli olduğu metotlar test sirasinda calistirilmayacaktir. Ayrca istenilirse @Ignore("açıklama") şeklinde yazilarak metodun neden test edilmesini istemedigimizide yazabiliriz.

```
6  public class C01_JUnitIlkTest {  
7  
8      @Test  
9      public void test01(){  
10         System.out.println("test01");  
11     }  
12  
13     @Test @Ignore  
14     public void test02(){  
15         System.out.println("test02");  
16     }  
17  
18     @Test  
19     public void test03(){  
20         System.out.println("test03");  
21     }  
22 }
```

# JUnit

## @Test Annotation

JUnit standart olarak, calistirilan her test method'unun sorunsuz olarak calisip calismadigini raporlar.

Calistirilan testlerin kac tanesinin passed, kac tanesinin failed oldugunu yazar.

Passed olan testler yesil tik ile, failed olan testler kirmizi carpi isareti ile, ignore edilen testler ise gri bir daire uzerine bir cizgi cekilerek isaretlenir,

Ancak JUnit'in dogru raporlama yapabilmesi icin Assert class'indan method'lar kullanilmalidir. Aksi takdirde sadece test method'unun sorunsuz calismis oldugunu raporlar.

```
JUnit Test Report - C01_JUnitlikTest (ders05_notations)
```

Method	Status	Time (ms)
test01	Passed	1ms
test02	Ignored	0ms
test03	Failed	4ms

Tests failed: 1, passed: 1, ignored: 1 of 3 tests - 5ms

/Users/ahmetbulutluoz/Library/test01

Test ignored.

java.lang.AssertionError:  
Expected :5  
Actual :7  
[Click to see difference](#)

```
JUnit Test Report - C01_JUnitlikTest (ders05_notations)
```

Method	Status	Time (ms)
test01	Passed	1ms
test02	Ignored	0ms
test03	Failed	4ms

Tests failed: 1, passed: 1, ignored: 1 of 3 tests - 5ms

/Users/ahmetbulutluoz/Library/test01

Test ignored.

java.lang.AssertionError:  
Expected :5  
Actual :7  
[Click to see difference](#)

# JUnit

## @Before - @After Annotations

@Before ve @After notasyonuna sahip method'lar her @Test method'undan once ve sonra calisirlar.

Bu method'lar sayesinde driver olusturulurken yaptigimiz ayarlari ve test method'u bittikten sonra driver'in kapatilmasi gibi gorevler icin tekrar tekrar kod yazmak mecburiyeti ortadan kalkar.

Genellikle @Before notasyonu kullanan method icin **setup**, @After notasyonu kullanan method icin **teardown** isimleri kullanilir.

```
WebDriver driver;
@Before
public void setUp(){
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();
    driver.manage().window().maximize();
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));
}
@Test
public void amazonTest(){
    driver.get("https://www.amazon.com");
    System.out.println(driver.getTitle());
}
@Test
public void facebookTest(){
    driver.get("https://www.facebook.com");
    System.out.println(driver.getTitle());
}
@Test
public void bestbuyTest(){
    driver.get("https://www.bestbuy.com");
    System.out.println(driver.getTitle());
}
@After
public void tearDown(){
    driver.close();
}
```

# JUnit

## Test Vs Test Method'u

Test method'u `@Test` notasyonu kullanilarak olusturulan, tek basina da veya baska test methodlari ile birlikte calistirilabilen bir test case'dir.

Test ise genellikle birden fazla method, class veya package icerebilen, belirli bir amac icin calistirilan test method'larinin bütündür.

Ornek olarak, smoke test, regression test veya end2end testlerini söyleyebiliriz.

```
WebDriver driver;
@Before
public void setUp(){
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();
    driver.manage().window().maximize();
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));
}
@Test
public void amazonTest(){
    driver.get("https://www.amazon.com");
    System.out.println(driver.getTitle());
}
@Test
public void facebookTest(){
    driver.get("https://www.facebook.com");
    System.out.println(driver.getTitle());
}
@Test
public void bestbuyTest(){
    driver.get("https://www.bestbuy.com");
    System.out.println(driver.getTitle());
}
@After
public void tearDown(){
    driver.close();
}
```

# JUnit

## @BeforeClass - @AfterClass Annotations

Eger bir class'daki birden fazla test method'u icin driver'in bir kere olusturulmasi ve tum testleri bittikten sonra driver'in kapatilmasi yeterli olacaksa kullanilirlar.

Boylece driver'i tekrar tekrar acip kapatmak zorunda kalinmaz.

**NOT :** @BeforeClass ve @AfterClass notasyonu kullanacak method'lar static olmak zorundadir.

```
// amazon sayfasina giden
// uc ayri test method'u olusturup
// Nutella, java ve Selenium icin arama yapip, arama sonuclarini yazdirin

static WebDriver driver;

@BeforeClass
public static void setup(){
    WebDriverManager.chromedriver().setup();
    driver=new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));
    driver.manage().window().maximize();
}

@Test
public void testNutella(){...}

@Test
public void testJava(){...}

@Test
public void testSelenium(){...}

@AfterClass
public static void teardown(){
    driver.close();
}
```

# JUnit

## Annotations

Bir class'da hem @Before - @After notasyonlari, hem de @BeforeClass-@AfterClass notasyonlari birlikte kullanilabilir.

Birlikte kullanimda method'larin yapacaklari islemler ve yapilari, calisma sirasi ve notasyon ozelliklerine gore düzenlenmelidir.

Ornek : 2 Test method'u

@Before - @After notasyonlari

@BeforeClass-@AfterClass notasyonlari olan bir class calistirilirsa, yandaki gibi bir calisma olacak ve toplam 8 method calistirilmis olacaktir.



# JUnit

## Assertions

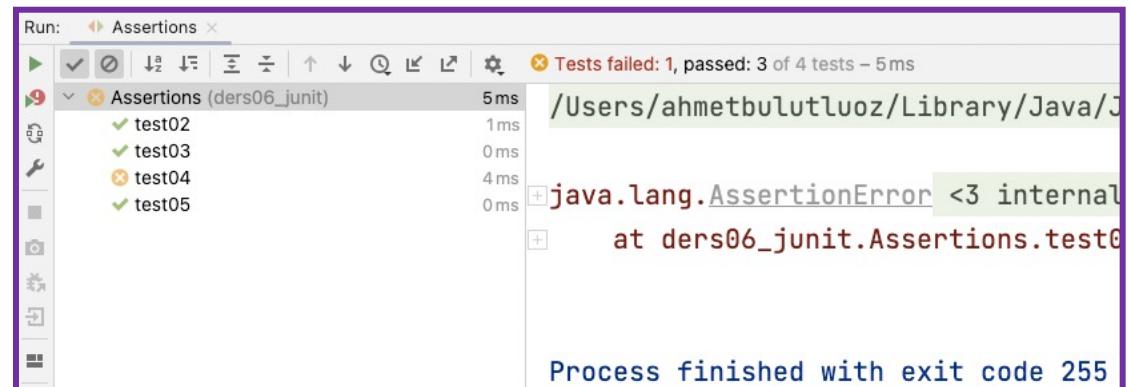
JUnit standart olarak, calistirilan her test method'unun sorunsuz olarak calisip calismadigini raporlar.

JUnit Assert Class'indan static method'lar kullanilarak Actual Ressult - Expected Result karsilastirilir.

Bu karsilastirma bir boolean sonuc uretir.

Assert islemi boolean sonucun **true** veya **false** olmasina degil, **beklenen sonuc ile ayni olup olmadigina** gore FAILED veya PASSED sonunu uretir.

```
public class Assertions {  
    int sayi1= 10;  
    int sayi2= 20;  
    int sayi3= 30;  
  
    @Test  
    public void test02(){  
        Assert.assertEquals(sayi3, actual: sayi1+sayi2) // PASSED  
    }  
    @Test  
    public void test03(){  
        Assert.assertNotEquals(sayi3, sayi2) // PASSED  
    }  
  
    @Test  
    public void test04(){  
        Assert.assertTrue(condition: sayi3==sayi2) // FAILED  
    }  
  
    @Test  
    public void test05(){  
        Assert.assertFalse(condition: sayi3==sayi2); // Passed  
    }  
}
```



# JUnit

## Assertions

JUnit ile assertion icin en cok kullanilan 4 method vardir.

```
Assert.assertEquals(a, c);
Assert.assertNotEquals(b, c);
Assert.assertTrue( condition: a>b );
Assert.assertFalse( condition: a<=b );
```

Kullanilacak method secilirken, assert isleminin icerisine yazilan boolean islem sonucunun ne olmasi beklendiği incelenmeli, assert method'u da beklenen sonuca uyumlu olarak secilmelidir.

Sonucun 20 oldugunu test edin → Assert.assertEquals( sonuc , 20 )

Sonucun succesfull olmadigini test edin. → Assert.assertNotEquals(sonuc, "succesfull")

Sayinin 50'den buyuk oldugunu test edin. → Assert.assertTrue( sayi>50)

Sayinin 50'den buyuk olmadigini test edin. → Assert.assertFalse( sayi>50)

# JUnit

```
@Test
```

```
public void test02(){  
    int a= 10, b=20, c=30;  
    String str1="Ali", str2="ALI";
```

```
    Assert.assertEquals(str1,str2);
```

Not equals

Failed

```
    Assert.assertNotEquals(str1,str2);
```

Not equals

Passed

```
    Assert.assertTrue(str1.equalsIgnoreCase(str2));
```

True

Passed

```
    Assert.assertFalse(str1.equals(str2));
```

False

Passed

```
    Assert.assertFalse(condition: a>b);
```

False

Passed

```
    Assert.assertEquals(c, actual: a+b);
```

equals

Passed

```
    Assert.assertNotEquals(b,c);
```

Not equals

Passed

```
    Assert.assertTrue(condition: c>b);
```

true

Passed

# JUnit

## Assertions

- 1) Bir class oluşturun: BestBuyAssertions
- 2) <https://www.bestbuy.com/> Adresine gidin farkli test method'lari olusturarak asagidaki testleri yapin
  - o Sayfa URL'inin <https://www.bestbuy.com/> 'a esit oldugunu test edin
  - o titleTest => Sayfa başlığının "Rest" içermediğini(contains) test edin
  - o logoTest => BestBuy logosunun görüntünlendiğini test edin
  - o FrancaisLinkTest => Fransizca Linkin görüntülendiğini test edin

# JUnit

## Assertions

- 1) Bir class oluşturun: YoutubeAssertions
- 2) <https://www.youtube.com> adresine gidin
- 3) Aşağıdaki adları kullanarak 4 test metodu oluşturun ve gerekli testleri yapın
  - titleTest => Sayfa başlığının “YouTube” olduğunu test edin
  - imageTest => YouTube resminin görüntünlendiğini (isDisplayed()) test edin
  - Search Box 'in erisilebilir olduğunu test edin (isEnabled())
  - wrongTitleTest => Sayfa basliginin “youtube” olmadığını doğrulayın

# JUnit

## Assertions

1. Bir Class olusturalim YanlisEmailTesti
2. <http://automationpractice.com/index.php> sayfasina gidelim
3. Sign in butonuna basalim
4. Email kutusuna @isareti olmayan bir mail yazip enter'a bastigimizda “Invalid email address” uyarisi ciktigini test edelim

# JUnit

## Check Box

Gerekli yapıyi olusturun ve aşağıdaki görevi tamamlayın.

- Verilen web sayfasına gidin.

<https://the-internet.herokuapp.com/checkboxes>

- Checkbox1 ve checkbox2 elementlerini locate edin.
- Checkbox1 seçili değilse onay kutusunu tıklayın
- Checkbox2 seçili değilse onay kutusunu tıklayın
- Checkbox1 ve Checkbox2'nin seçili olduğunu test edin

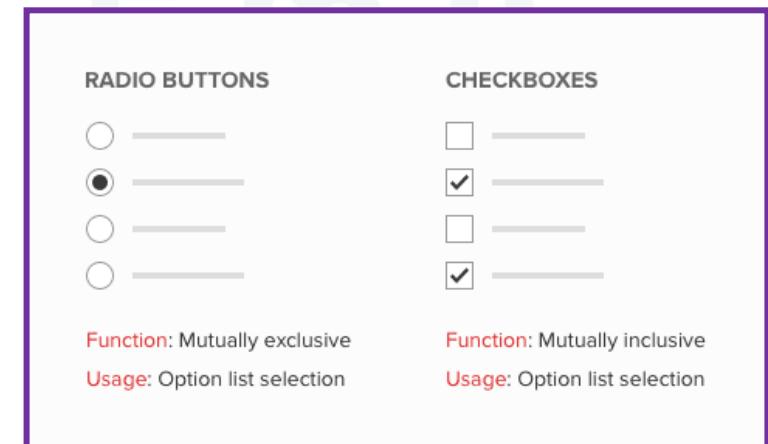
<b>Specialty Food Type</b>
<input type="checkbox"/> GMO-Free
<input type="checkbox"/> Organic
<input type="checkbox"/> USDA Organic
<b>Calories Per Serving</b>
<input type="checkbox"/> 0 Calories
<input type="checkbox"/> Up to 40 Calories
<input type="checkbox"/> 40 to 100 Calories
<input type="checkbox"/> 100 to 200 Calories
<input type="checkbox"/> 200 to 300 Calories
<b>Fat Calories Per Serving</b>
<input type="checkbox"/> 40-100 Calories
<b>Nutrition Facts Per Serving</b>
<input type="checkbox"/> Fat Free (<0.5g)
<input type="checkbox"/> Low Fat (<3g)
<input type="checkbox"/> Free of Saturated Fat (<0.5g)
<input type="checkbox"/> Free of Trans Fat (0g)
<input type="checkbox"/> Cholesterol Free (<2mg)
<input type="checkbox"/> Sodium Free (<5mg)
<input type="checkbox"/> Low Sodium (<140mg)
<input type="checkbox"/> Carbohydrate Free (<0.5g)
<input type="checkbox"/> Sugar Free (<0.5g)
<input type="checkbox"/> Dietary Fiber (>10g)
<input type="checkbox"/> Protein (>10g)
<b>Food Diet Type</b>
<input type="checkbox"/> Vegan
<b>Availability</b>
<input type="checkbox"/> Include Out of Stock

# JUnit

## Radio Buttons

Gerekli yapıyi olusturun ve aşağıdaki görevi tamamlayın.

- a. Verilen web sayfasına gidin.  
<https://facebook.com>
- b. Cookies'i kabul edin
- c. Create an account buton'una basin
- d. Radio button elementlerini locate edin ve size uygun olani secin
- e. Sectiginiz radio button'un secili, ötekilerin secili olmadigini test edin



# JUnit

```
@Test  
public void test02(){  
    int a= 10, b=20, c=30;  
    String str1="Ali", str2="ALI";  
  
    Assert.assertEquals(str1,str2); → Failed  
    Not equals  
    Assert.assertNotEquals(str1,str2); → Passed  
    Not equals  
    Assert.assertTrue(str1.equalsIgnoreCase(str2)); → Passed  
    True  
    Assert.assertFalse(str1.equals(str2)); → Passed  
    False  
    Assert.assertFalse(condition: a>b); → Passed  
    False  
    Assert.assertEquals(c, actual: a+b); → Passed  
    equals  
    Assert.assertNotEquals(b,c); → Passed  
    Not equals  
    Assert.assertTrue(condition: c>b); → Passed  
    true
```

# JUnit

## TestBase Class

Her test method'u calisirken webDriver objesine ve bu obje icin ayarlara ihtiyac vardır. Bu islemleri her class icin yeniden yasmak yerine Java'daki **Inheritance** ozelligi kullanilabilir.

Bu islemlerin yapildigi **@Before** ve **@After** notasyonuna sahip method'lar olusturulan bir TestBase class'ina konulabilir.

Testlerin yapilacagi class'lar extends keyword ile TestBase class'ina child class yapip, oradaki setup ve teardown method'lari direk kullanilabilir.

```
public class TestBase {  
  
    protected WebDriver driver;  
  
    @Before  
    public void setup(){  
        WebDriverManager.chromedriver().setup();  
        driver=new ChromeDriver();  
        driver.manage().window().maximize();  
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));  
    }  
  
    @After  
    public void teardown(){  
  
        driver.close();  
    }  
}
```

TestBase class'inda setup ve teardown method'lari disinda tekrar tekrar yapacagimiz islemleri yapan hazir method'lar da konulabilir.

Olusturulan driver objesi sadece child class'larin kullanabilmesi icin protected yapilabilir

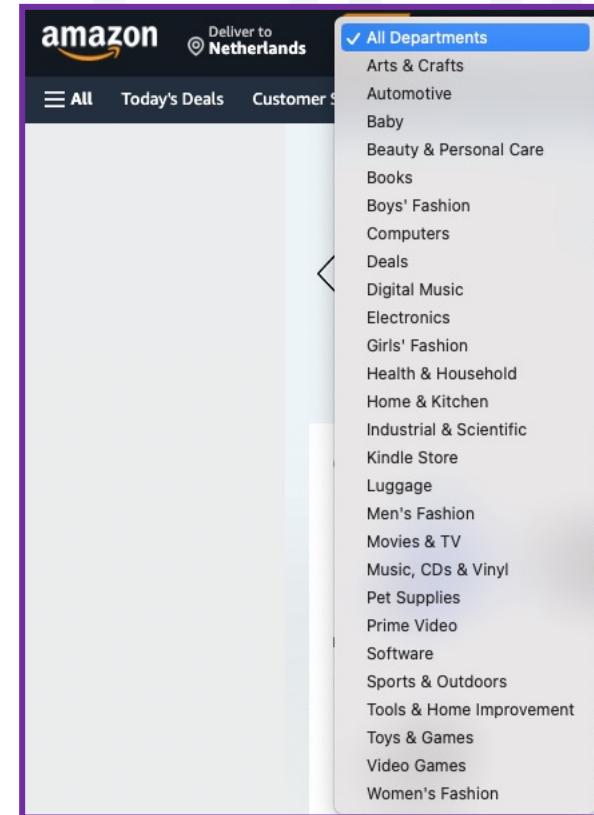
# JUnit

## Handle Dropdown

Dropdown(acilir menu) ozel bir HTML kodu ile olusturulur.

HTML sayfalarda farkli acilir menuler yapilabilir. Dropdown'i digerlerinden ayiran tag'inin **<select>** olmasi ve secilebilecek opsiyonların da **<option>** tag'i ile olusturulmasidir.

Selenium dropdown menu ile islem yapilabilmesi icin ozel bir Select class'i olusturmustur. Select class'indan olusturacak obje yardimi ile bu class'daki method'lar kullanilabilir.



# JUnit

## Handle Dropdown

Dropdown'daki opsiyonlardan birini secmek icin 3 islem yapilir.

1- Dropdown webelement'i locate edin

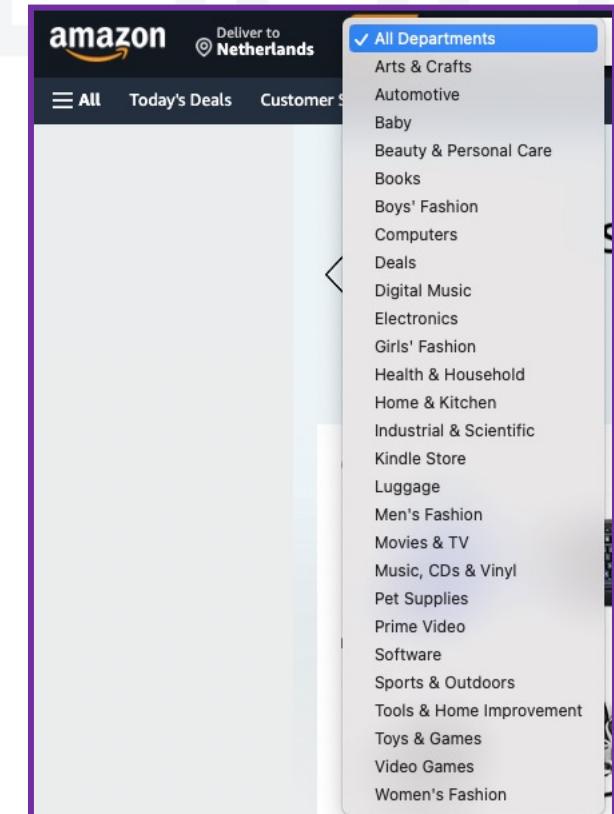
```
WebElement ddm= driver.findElement(By.id("searchDropdownBox"));
```

2- Select class'indan bir obje olusturun ve locate edilen dropdown elementi parametre olarak yazin

```
Select select= new Select(ddm);
```

3- select objesi ile Select class'inda bulunan method'lardan uygun olani ile istediginiz option'i secin.

```
select.selectByVisibleText("Electronics");
```



# JUnit

## Select Class Method'ları

1- istenen option'i secme

```
select.selectByIndex(1);
select.selectByValue("2");
select.selectByVisibleText("Option 1");
```

2- Bir option secildikten sonra secilen option'i webelement olarak döndürme

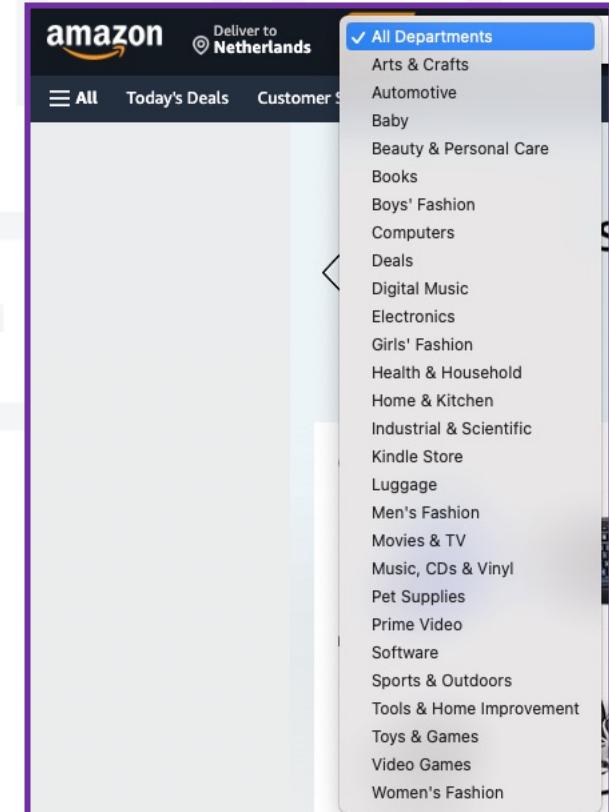
```
select.getFirstSelectedOption()
```

3- Bir option secildikten sonra secilen option'i text olarak döndürme

```
select.getFirstSelectedOption().getText()
```

4- Tum option'ları döndürüp kaydetme

```
List<WebElement> optionsList= select.getOptions();
```



# JUnit

## Handle Dropdown

- Bir class oluşturun: **DropDown**
- <https://the-internet.herokuapp.com/dropdown> adresine gidin.
  1. **Index** kullanarak Seçenek 1'i (Option 1) seçin ve yazdırın
  2. **Value** kullanarak Seçenek 2'yi (Option 2) seçin ve yazdırın
  3. **Visible Text**(Görünen metin) kullanarak Seçenek 1'i (Option 1) seçin ve yazdırın
  4. Tüm dropdown değerleri(value) yazdırın
  5. Dropdown'un boyutunun 4 olduğunu test edin

# JUnit

## Handle Dropdown

- Bir class olusturun: C3\_DropDownAmazon
- <https://www.amazon.com/> adresine gidin.
  - Test 1

Arama kutusunun yanindaki kategori menusundeki kategori sayisinin 45 oldugunu test edin

- Test 2
  - 1. Kategori menusunden Books secenegini secin
  - 2. Arama kutusuna Java yazin ve aratin
  - 3. Bulunan sonuc sayisini yazdirin
  - 4. Sonucun Java kelimesini icerdigini test edin

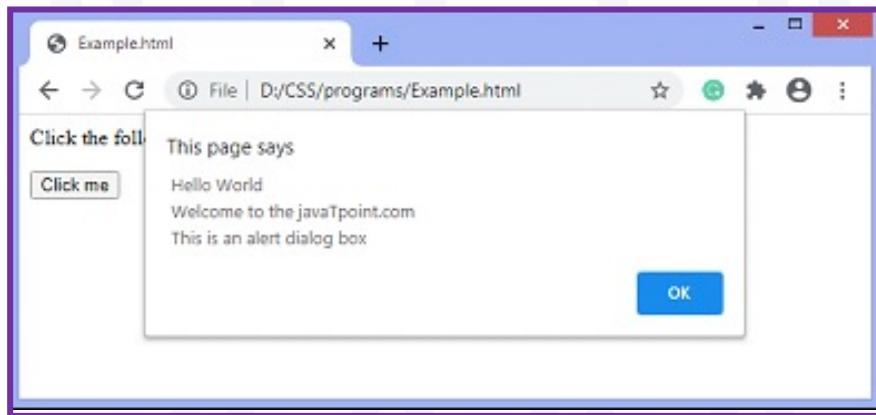
# JUnit

## Handle Dropdown

1. <http://zero.webappsecurity.com/> Adresine gidin
2. Sign in butonuna basin
3. Login kutusuna “username” yazın
4. Password kutusuna “password.” yazın
5. Sign in tusuna basin
6. Pay Bills sayfasına gidin
7. “Purchase Foreign Currency” tusuna basin
8. “Currency” drop down menusünden Eurozone’u secin
9. “amount” kutusuna bir sayı girin
10. “US Dollars” in secilmədigini test edin
11. “Selected currency” butonunu secin
12. “Calculate Costs” butonuna basin sonra “purchase” butonuna basin
13. “Foreign currency cash was successfully purchased.” yazısının çıktığını kontrol edin.

# JUnit

## JS Alerts



### Alert Nedir?

Alert kullanıcıya bir tür bilgi vermek veya belirli bir işlemi gerçekleştirmek istemek için ekran bildirimini görüntüleyen küçük bir mesaj kutusudur. Uyarı amacıyla da kullanılabilir.

#### 1- HTML Alerts

Bir alert çıktığında sağ click ile inspect yapabiliyorsak html alert'dir ve extra bir işleme gerek yoktur.

#### 2- Js Alerts

Js alerts inspect yapılamaz, ekstra işleme ihtiyaç vardır.

# JUnit

## JS Alerts

1. **Simple Alert** : Bu basit alert ekranında bazı bilgiler veya uyarılar görüntüler. Ok denilerek kapatılır
2. **Confirmation Alert** : Bu onay uyarısı bir tür işlem yapma izni ister. Alert onaylanıyorsa OK, onaylanmıyorsa Cancel butonuna basılır.
3. **Prompt Alert** : Bu Prompt Uyarısı kullanıcıdan bazı girdilerin girilmesini ister ve selenium webdriver metni sendkeys ("input....") kullanarak girebilir.



[https://the-internet.herokuapp.com/javascript\\_alerts](https://the-internet.herokuapp.com/javascript_alerts)

# JUnit

## JS Alerts

1. **accept( )** : Alert üzerindeki OK butonuna basmak için kullanılır.

```
driver.switchTo().alert().accept();
```

2. **Dismiss( )** : Alert üzerindeki OK butonuna basmak için kullanılır.

```
driver.switchTo().alert().dismiss();
```

3. **getText( )** : Alert üzerindeki yazıyı döndürür.

```
driver.switchTo().alert().getText();
```

4. **sendKeys("istenen yazı")** : Alert üzerindeki text kutusuna istenilen metni yazdırır.

```
driver.switchTo().alert().sendKeys( keysToSend: "Deneme");
```

Click for JS Alert

Click for JS Confirm

Click for JS Prompt

# JUnit

3 test method'u olusturup asagidaki gorevi tamamlayin

## 1. Test

- [https://the-internet.herokuapp.com/javascript\\_alerts](https://the-internet.herokuapp.com/javascript_alerts) adresine gidin
- 1.alert'e tiklayin
- Alert'deki yazinin "I am a JS Alert" oldugunu test edin
- OK tusuna basip alert'i kapatin

## 2.Test

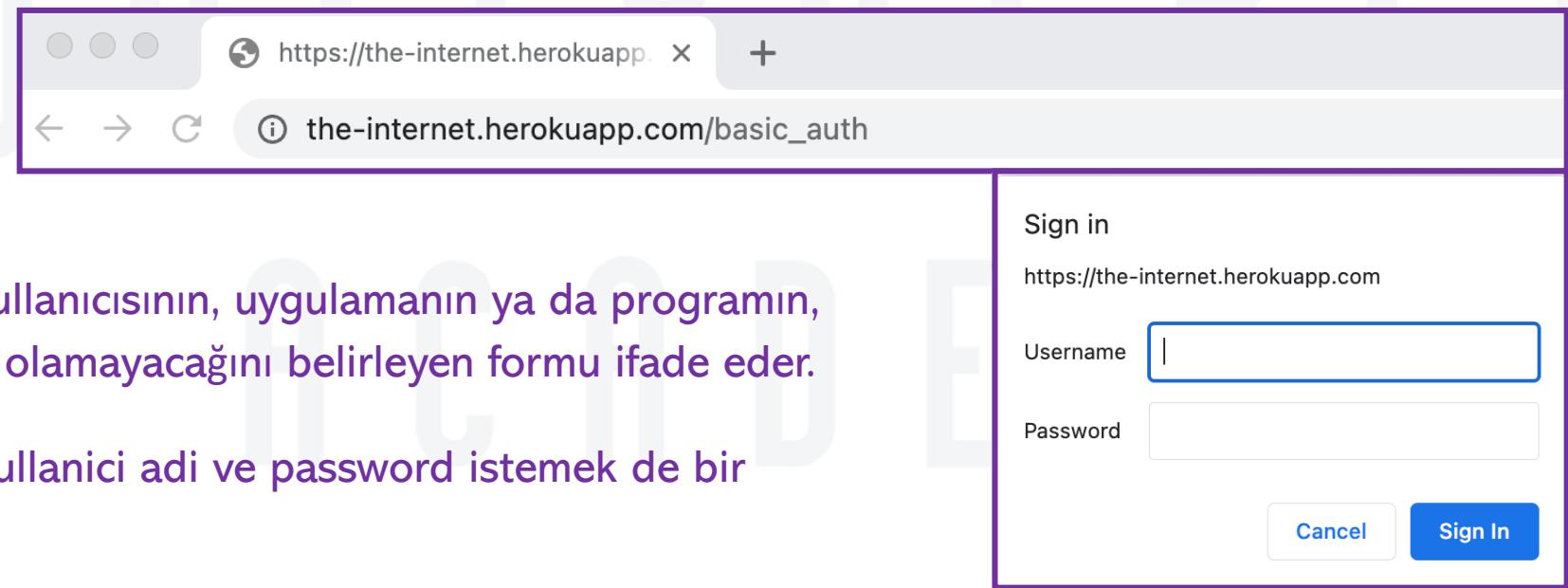
- [https://the-internet.herokuapp.com/javascript\\_alerts](https://the-internet.herokuapp.com/javascript_alerts) adresine gidin
- 2.alert'e tiklayalim
- Cancel'a basip, cikan sonuc yazisinin "You clicked: Cancel" oldugunu test edin

## 3.Test

- [https://the-internet.herokuapp.com/javascript\\_alerts](https://the-internet.herokuapp.com/javascript_alerts) adresine gidin
- 3.alert'e tiklayalim
- Cikan prompt ekranina "Abdullah" yazdiralim
- OK tusuna basarak alert'i kapatyalim
- Cikan sonuc yazisinin Abdullah icerdigini test edelim

# JUnit

## Basic Authentication



https://the-internet.herokuapp.com/basic\_auth

Sign in

https://the-internet.herokuapp.com

Username

Password

Cancel Sign In

### Authentication Nedir?

Kısaca, herhangi bir internet kullanıcısının, uygulamanın ya da programın, söz konusu sisteme dahil olup olamayacağını belirleyen formu ifade eder.

Uygulama ana sayfalarındaki kullanıcı adı ve password istemek de bir authentication'dır.

End user'lar için tasarlanmayan uygulamalarda(Ornegin API sorgularında) bu authentication HTML komutları ile de yapılabilir.

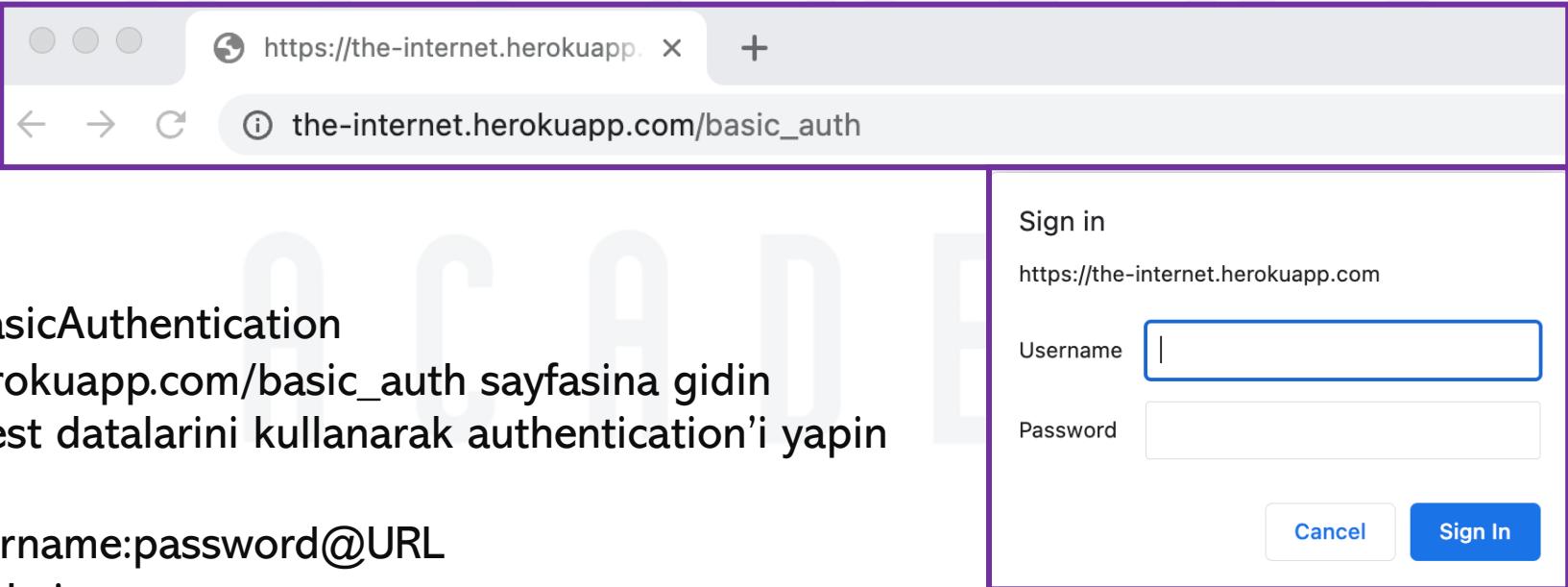
Bu authentication'i yapabilmek için uygulamanın kullanıcılarına authentication'i nasıl yapacağına dair bilgilendirme yapmış olması gereklidir.

Ornegin yandaki uygulama için authentication aşağıdaki gibi yapılabilir.

<https://username:password@URL>

# JUnit

## Basic Authentication



The screenshot shows a browser window with the URL [https://the-internet.herokuapp.com/basic\\_auth](https://the-internet.herokuapp.com/basic_auth). A purple box highlights the browser's title bar and the address bar. Another purple box highlights the 'Sign in' dialog box, which contains fields for 'Username' and 'Password' with placeholder text 'admin' and 'admin' respectively, and 'Cancel' and 'Sign In' buttons.

- 1- Bir class olusturun : BasicAuthentication
- 2- [https://the-internet.herokuapp.com/basic\\_auth](https://the-internet.herokuapp.com/basic_auth) sayfasina gidin
- 3- asagidaki yontem ve test datalarini kullanarak authentication'i yapin

Html komutu : <https://username:password@URL>

Username : admin  
password : admin

- 4- Basarili sekilde sayfaya girildigini dogrulayin

```
driver.get("https://admin:admin@the-internet.herokuapp.com/basic_auth");
```

**JUnit**  
Handle IFrame

UNITYVERSE  
ACADEMY

unityverseacademy.com











