JAVA-SELENIUM

SDLC-STLC

SQL -API

MUHTEMEL

MÜLAKAT SORULARI

By Akif Rencber

akifrencber@gmail.com

1. KAÇ FARKLI LOCATOR VARDIR?

Sekiz farklı locators vardır. Locator, web sayfası içinde unique bir web elementi tanımlayan bir adres olarak tarif edilebilir. Bu sayede Selenium'da farklı locator tiplerini kullanarak doğru ve isabetli bir şekilde bir web elementi tanımlayabiliriz. Web elementin doğru tanımlanması ilgili otomasyon case'nin başarılı bir şekilde execute edilmesini sağlayacaktır.

By.id() = driver.findElement(By.*id* ("add-to-cart-button"));

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

By.tagName() = driver.findElements(By.tagName("a"));

By.className() = driver.findElements(By.*className*("s-image"));

By.linkText() = driver.findElement(By.*linkText*("Addresses"));

By.partialLinkText() = driver.findElement(By.*linkText*("dresses"));

By.xpath () = driver.findElement(By.*xpath*("//input[@id='twotabsearchtextbox']"));

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

2. XPATH VE CSS_SELECTOR ARASINDAKİ FARKLAR NELERDİR?

CSS_SELECTOR	XPATH
Xpath'e göre hızlıdır	Css selector'e göre yavaştır.
Text () syntax'ini desteklemez	Text () syntax'ini destekler
Parent child ilişkisine sahip olmadığı için geriye dönük hareket edemez bu sebeple dinamik değildir.	Parent child ilişkisine sahip olduğu için geriye dönük hareket edebilir. Bu sebeple dinamiktir. Dinamik olduğu için CssSelector'e göre elementi bulma konusunda daha garantidir.
groupIndex'i desteklemez	groupIndex destekler

By.xpath () = driver.findElement(By.*xpath*("//input[@id='twotabsearchtextbox']"));

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

3. ABSOLUTE VE RELATİVE XPATH ARASINDAKİ FARKLAR NELERDİR?

Absolute xpath, HTML kodlarindaki hiyerarsi kullanilarak parent/child silsilesi yazilip istenen elemente ulasmak icin kullanilir. HTML kodlari dinamik oldugundan, developer'lar sürekli ekleme veya cikarmalar yapabilirler. Absolute xpath yazildiktan sonra hiyerarsideki ekleme veya cikarmalar kodumuzun istenen webElementi bulamamasina sebep olur. Bu sebeple kullanımı tavsiye edilmez.

/html[1]/body[1]/div[3]/main[1]/div[2]/div[1]/div[7]/div[1]/div[2]/div[1]/div[3]/div[1]/div[1]/div[1]/div[1]/div[1]/div[1]/div[1]/ul[1]/li[3]/div[1]/a[1]/div[1]/div[1]/div[3]/div[2]

Relative xpath ise tag, attribute ve value bilgilerinin unique sonuc olusturacak sekilde birlesmesiyle olusur. O element degistirilmedikce relative xpath de degismez. Sadece, relative xpath icin **index kullandığımızda** HTML kodlarında eklenecek veya cikarılacak webelementler xpath'i etkileyebilir.

Örnek:

By.xpath ( ) = driver.findElement(By.*xpath*("(//input[@id='twotabsearchtextbox']) [1]"));

4. SELENİUM'DA SCREENSHOT NASIL YAPILIR, NEDEN ÖNEMLİ VE STEPLERİNİ KISACA BELİRTİNİZ?

Selenium'da TakesScreenshot interface'den getScreenshotAs() methodunu kullanarak screenshots alabiliriz. Screenshot almak failed olmuş testlerin bir nevi belgelendirilmiş halidir. Bu sayede bir tester yapmış olduğu otomasyonda bulduğu defect'i developer'a bildirerek gerekli eksikliklerin giderilmesini sağlar ve bulduğu defect'i ispatlamış olur.

Screenshot almak için öncelikle bir obje oluşturarak bunu ilgili driver'a cast ederiz.

TakesScreenshot tsc = (TakesScreenshot) Driver.*getDriver* (); // TestNG biçimi

Daha sonra objeye verdiğimiz isim (tsc) tsc. getScreenshotAs (OutputType.*FILE*); kodunu çağırıyoruz.

En sonunda FileUtils classını kullanarak ekran görüntülerini projede istenilen yerde saklarız.

```
// naming the screenshot with the current date to avoid duplication
String date = new SimpleDateFormat("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") + "/target/Screenshots/" + name + date +
".png"; File finalDestination = new File(target);

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

NOT: Ben projemde hooks classını kullanarak bu class içinde testin fail olması durumunda ekran resminin alınmasına yönelik kodlarımı yerleştirerek ekran resmini bu şekilde alıyordum.

```
@After
public void tearDown(Scenario scenario) {
  final byte[] screenshot = ((TakesScreenshot)
Driver.getDriver()).getScreenshotAs(OutputType.BYTES);
  if (scenario.isFailed()) {
    scenario.attach(screenshot, "image/png", "screenshots");
  }
  Driver.closeDriver();
}
```

5. BROWSER İÇİNDE NAVİGATE (GEZİNME) NASIL YAPILIR?

navigate() methodları kullanarak bunu yapmak mümkündür.

Driver.*getDriver*().navigate().back(); Browser'da sayfayı geri götürür.

Driver.*getDriver*().navigate().forward(); Browser'da sayfayı ileri götürür.

Driver.*getDriver*().navigate().refresh(); Browser'da sayfayı yeniler.

Driver.getDriver().navigate().to("www.amazon.com"); Bizi ilgili web sayfasına götürür.

NOT: Sayfayı refresh etmek için farklı methodlar da mevcuttur.

➔ **driver.get("URL");** veya **driver.getCurrentUrl();**

➔ **driver.navigate().to("URL");** **veya** **driver.navigate().to(driver.getCurrentUrl());**

➔ **sendKeys (Keys.F5);**

NOT: Actions class'ını kullanarak bir işlem yaptığımız zaman perform(); diyerek kodu bitirmezsek ne olur?

6. JAVA'DA ABSTRACT CLASS İLE INTERFACE ARASINDAKİ FARKI AÇIKLAYINIZ?

Öncelikle şunu ifade edelim: Object Oriented Programming'de (OOP) classların soyutlanmasını Abstract Class ve Interface ikilisi gerçekleştirmektedir.

Peki Abstract Class Nedir? Abstract class bir araba ya da işleyen bir mekanik aksam içindeki teknik yapılanmadır. Örnekle açıklamak gerekirse; Biz arabaya biner kontağı çevirir ve gaza basarak arabayı hareket ettirmeye odaklanırız. Arabanın mekanik aksamında meydana gelen olaylar ile ilgilenmeyiz. Burada gerçekleşen durumlar ve olaylar abstact class'ın konusuna girer.

➔ Abstract Class, static methodlar içerebilir. Ancak Interface, static method içermez.

```
➔ public static void sanzuman() {
   System.out.println("Abstract class'da static keyword ile body oluşturulabilir");
  }
```

```
void motor();
```

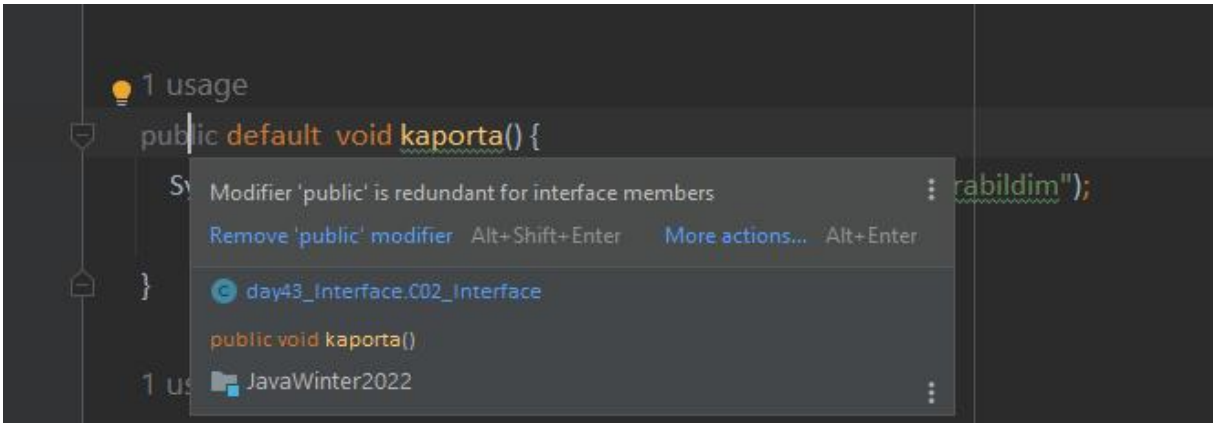
➔ Abstract classlar hız açısından Interface'lere göre daha hızlıdır.

➔ Abstract class'da constructor bulunur, Interface'de constructor olmaz.

Abstract class'da class olması sebebiyle constructor olmasına rağmen new keyword'ü kullanılmaz ve dolayısıyla obje oluşturulamaz. Aynı şekilde

interface'de de new keyword'ü kullanılmaz ve class olmadığı için consturctor olmaz.

➔ Abstract classlar public dışında access modifier alabilirken Interface'ler sadece public alabilir.



➔ Bir class sadece bir tane Abstarct class'ı inherit edebilirken, Birden fazla Interface'i inherit edebilir.

➔ Abstract class'larda method body'si olurken, Interface'lerde method body olmaz.

```
void motor();
```

Ancak; interface'de body'si olan bir method yazmak isterseniz Java'da iki alternatif yöntem vardır. Birincisi methoda default keyword'ü koymak, ikincisi ise yine method'da static keyword'ünü kullanmak Aşağıdaki örnekte görüldüğü gibi normalde method badisi olmayan interface body'yi iki şekilde koyabiliriz.

```
public default  void kaporta() {
 System.out.println("default keyword sayesinde body olusturabildim");

}

public static void sanzuman() {
 System.out.println("static keyword ile body olusturabildim");
}
```

➔ Abstract class, objelerin ne yapması ve nasıl yapması gerektiğini belirler. Interface ise sadece ne yapması gerektiğini belirler.

➔ Abstract class'ın tanımlaması için abstarct keyword'ü kullanılırken Interface için interface keyword'u kullanılır.

```
public abstract class  AbstractClassDeneme
```

```
public interface InterfaceDeneme
```

➔ Bir class interface'i inherit etmek isterse implement keyword'ünü kullanırız

```
public class  Deneme  implements InterfaceTest{
```

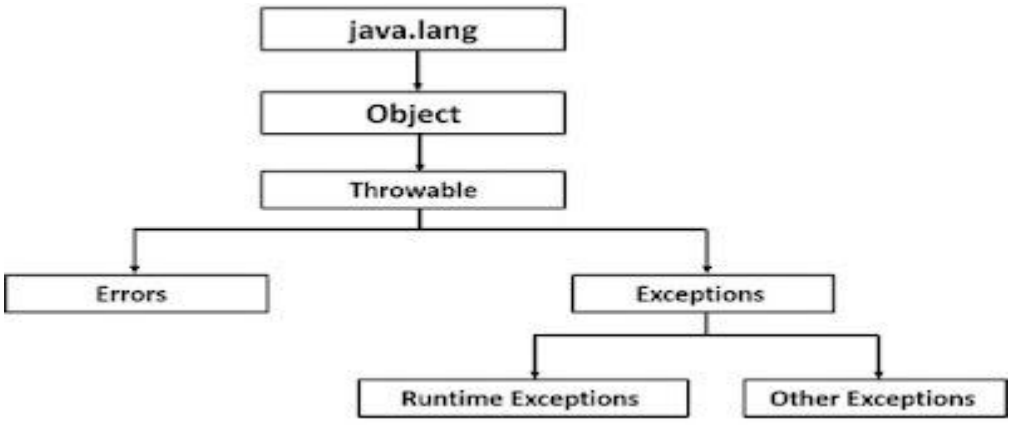
➔ Bir class Abstract class'ı inherit etmek isterse extends keyword'ünü kullanırız

```
public abstract class AbsMuhasebe extends AbsPersonel
```

7. JAVA'DA EXCEPTİONLAR

Exception, kod blokları arasında oluşan anormal duruma denir. Örnek vermek gerekirse bir sayının sıfıra bölünmesi, kod bloğunun noktalı virgül (;) ile kapatılmaması, String bir değer döndüren data type'ına int bir data type girilmesi gibi nedeler exception'a sebep olacaktır. Basit hatalar eksik bilgilerin girilmesi, ya da yanlış girilen bilgilerin düzeltilmesi ile handle edilebilecekken, sıfıra bölünme *"arithmeticException"* gibi exception sorunlarında try catch bloğu kullanmak gerekir.

Aşağıdaki tabloda görüleceği üzere Java'da iki tür exception mevcuttur. Kontrolsüz olanlar (UncheckedExceptions) ve Kontrollü olanlar (CheckedExceptions). Kontrolsüz exceptionlar aynı zamanda run time exception olarak da bilinirken kontrollü exceptionlara compile time exception da denir. Aşağıdaki tabloda görüldüğü gibi "Errors" ise exception'ların ata classı olan Throwable'dan gelmektedir.



NOT: Java'da bütün hata türlerinin ata class'ı "Throwable" sınıfıdır.

Hata (Error) : Ölümcül bir hatayı işaret eder ve telafisi çok zordur. Örneğin OutOfMemoryError (Yetersiz Bellek) hatası oluşmuş ise uygulamanın buna müdahele edip düzeltmesi olanaksınızdır.

Kontrolsüz İstisnalar (UncheckedExceptions) : Bu istisna tiplerine Çalışma anı İstisnaları da (Run-Time Exceptions) denilir. Çünkü çalışma anında meydana gelen istisnalardır. Eğer uygulama normal seyrinde giderse ortaya çıkmaması gerekir. Örneğin, **ArrayIndexOutOfBoundsException** istisna tipi, bir dizinin olmayan elemanına eriştiğimiz zaman ortaya çıkar. Yani kontrolsüz kodlamadan dolayı meydana gelen istisna tipleridir. Java bu tür istisnalar için önceden bir önlem alınmasını şart koşmaz; yine de önlem almakta özgürsünüzdür.

Kontrollü İstisnalar (CheckedExceptions) : Bu istisna tiplerine Derleme Anı İstisnaları da (Compile-Time Exceptions) denilir. Çünkü derleme anında ide'ler tarafından uyarılırız. Eğer derleyici derleme zamanında exceptionlar için try-catch bloğu göremezse hata verecektir ve kodumuz biz handle edene kadar derlenmeyecektir. Bu istisnalar çevresel koşullardan dolayı oluşabilirler. Örneğin erişilmek istenilen dosyanın yerinde olmaması (FileNotFoundException) veya ağ (Network) bağlantısının kopması sonucu ortaya çıkabilecek olan istisnalardır. Bu istisnalar için önceden önlem alınması gereklidir.

1. **NumberFormatException**: Format dışı kullanımında gerçekleşen durumdur. Örneğin sayı girilmesi gereken yere karakter girilmesi gibi. **= compile time**

2. **ArrayIndexOutOfBoundsException:** Array'da aranan index dışında bir değer girilmişse bu exception ortaya çıkar. **= run time**

3. **FileNotFoundException:** Oluşturulmamış bir dosyaya erişim durumunda meydana gelen hatadır. **= compile time**

NOT: Java böyle bir durumda bize altı kırmızı çizili bir şekilde uyarı verir ve bizden method signature'ına throws keyword'lü bir exception yazmamızı ister. Bu şekilde ilgili exception handle edilmiş olur.

4. **VirtualMachineError:** JVM'nin çalışmasını etkileyen durumları inceler. **= Error**

5. **AWTError:** Grafik arayüze ait hataları inceler. **= Error**

6. **ArrayStoreException:** Array'a kendi türü dışında bir veri girilmesi durumunda gerçekleşir. = **run time**

7. **OutOfMemoryError:** Bellek yetersizliği durumlarını inceler. = **Error**

8. **ClassNotFoundException:** Olmayan bir class'a erişme istediği durumlarını inceler.

= **run time**

9. **IOException:** Giriş çıkış işlemlerindeki istenmeyen durumları inceler. **= compile time**

10. **ClassCastException:** Herhangi bir nesne değişkenine farklı bir tip değer girilmesi = **run time**

Örnek

```
Object i = Integer.valueOf (42);
String s = (String) i;          // ClassCastException thrown here.
```

11. **StringIndexOutOfBoundsException:** String'de var olmayan bir indekse erişilmeye çalışıldığında gerçekleşen hata. = **run time**

12. **AritmeticException:** Sıfıra bölme gibi mantıksal matematik hatalarında oluşur. = **compile time**

Örneğin

(İnt x = 10;

İnt y = 0;

System.out.println (x/y)

Komutu exception fırlatır.)

13. NullPointerException: Herhangi bir nesne değişkenine ilk değer atanmadan kullanılmaya çalışıldığında ortaya çıkar. = **run time**

14. IllegalArgumentException: Metotlara geçersiz argüman atamalarında fırlatılır.= **run time**

NOT: Compile Time Exceptionları kontol altına almak için **try catch** bloğu kullanırız.

Örnek syntax:

```
 try {
  int a= 42 / 0;
} catch (ArithmeticException Hata) {
  System.out.print("Hata oluştu :" + Hata.getMessage());
}
```

8. SELENİUM'DAKİ EXCEPTİON'LARDAN ÖNEMLİ OLAN BİR KAÇ TANESİNİNİN İSMİNİ SÖYLEYİNİZ?

1. **ElementNotVisibleException**: Web sayfasında görünür olmayan bir elementi bulmaya çalıştığında ortaya çıkar.
2. **ElementNotSelectableException:** Aranan webElement'in seçilemediği durumlarda,
3. **NoAlertPresentException:** Olmayan bir pop-up şeklindeki diyalog üzerinde işlem yapılmaya çalışıldığında ortaya çıkar.
4. **NoSuchAttributeException**: Attribute'un bulunamadığı durumlarda gerçekleşir.
5. **NoSuchElementException**: sayfada bulunmayan bir elemente erişim sağlanmaya çalışıldığında.
6. **NoSuchWindowException**: Yeni bir pencereye geçmeye çalışıldığında,
7. **TimeoutException:** bir komutun tamamlanması bekleme süresinden daha uzun sürdüğünde ortaya çıkar.
8. **WebDriverException**: herhangi bir kod bloğu Selenium WebDriver'ı çalıştıramadığında oluşan exception türüdür.

9. SELENIUM'UN AVANTAJLARI VE DEZAVANTAJLARI NEDİR?

Bu soru, "Neden Selenium'u tercih edersiniz" sorusu şeklinde de sorulabilir"

Browser'lari otomat eden tool'larin calismasi icin bir suite olan Selenium,

- Acik kaynakli olması,
- Bir cok browser icin destek sağlaması,
- Farkli programlama dilleri ile calisması,
- Tüm isletim sistemleri ile uyumlu olması yönüyle avantajlara sahiptir.

NOT: Bu sebepler onun aynı zamanda tercih edilme nedenidir?

---Buna karşılık (Dezavantajları)

***Bu soru, "Selenium ile otomate edilemeyen durumlar" şeklinde de sorulabilir.**

- Programlama dili bilmeniz gerekir.
- Sadece web tabanli uygulamalar ile calisir. (Appium ile birlikte mobil testing yapilabilir)
- Performans testleri yapmaz, daha cok funcktional testlere odaklanir
- Handle captcha mümkün degildir. **(çok büyük uğraş ile manuel destek alarak belki...)**
- Acik kaynakli oldugu icin profesyonel bir musteri hizmetleri yoktur ama cok genis bir kullanici kitlesi ve bunlarin bilgi paylasim portallari mevcuttur.

10. TEST SENARYOSU OLUŞTURMAYA NASIL KARAR VERİRSİNİZ?

Öncelikle kabul kriterlerine bakarak takım halinde bizden istenen gereksinimlerin neler olduğunu tespit etmeye çalışırım takım halinde yapılan değerlendirmeler ile kabul kriterleri sprint toplantısında görev dağılımı yapılarak paylaştırılır ve artık test senaryosu yapmanın zamanı gelir. Elbette test senaryosu yapmadan önce sağlıklı bir işleyiş olması için kabul kriterleri bağlamında manuel test yapmak yani üzerinde çalışılacak web sitesini (web uygulamasının) tanımak gereklidir.

11. AGİLE VE WATERFALL METHODOLOJİ ARASINDAKİ FARKLAR NELERDİR?

Agile methodoloji 2-4 haftalık sprintler halinde bütün aşamaların birbiri ve stockholder ile temas halinde olduğu yazılım yaşam döngüsü sürecidir. Waterfall metodolojide bir aşama bitmeden diğer aşamaya geçilmez. Bu konuda oldukça katı olan bu metdolojide hataların teşhisi oldukça geç gerçekleşebilir. Ayrıca stockholderın olmasını beklediği sonuç, projenin yapım aşamasındaki iletişim eksikliğinden kaynaklı aksi netice ortaya çıkarabilir. Oysa agile metodolojide sürekli takip ve projeyi sprintlere bölerek yürütme gerçekleştiğinden olası sorunlara çok hızlı müdahale edilir ve projenin daha sağlıklı bir şekilde ilerlemesi sağlanır.

Waterfall modeli baştan sona her ayrıntıyı adım adım ve uzun vadeli planlarken; Agile modelinde kısa vadeli planlar söz konusu olur. Çünkü Agile yaklaşıma göre proje süreci devam ederken değişiklik yapılabilir. Her aşama Waterfall modelinde olduğu kadar net değildir. Waterfall incelemelerinde dokümanlar ön plandadır. Her aşamanın tamamlanmasından sonra incelemeler yapılır. Agile modelinde analizler, müşterilerle yapılan iletişim neticesinde sık sık gerçekleşmektedir. Waterfall modelinin uzun vadede maliyeti artırma riski vardır. Agile ise ürüne dair en iyi versiyonu bekler. Bu nedenle de ürünle ilgili potansiyeli zorlar ve görece daha düşük riskler taşır. Agile yaratıcılığa imkân tanıyan esnekliğe sahipken Waterfall modelinde bundan söz etmek pek mümkün olmaz. Bu noktada en başta alınan kararlara bağlılık söz konusudur.

12. LOGİN BUTONUNA CLİCK ETMENİN ALTERNATİF YOLU NEDİR?

Bunun için **use submit ();** methodunu kullanabilir. Fakat attribute type'ı submit olduğu zaman bu kullanım geçerli olabilir.

13. CHECKBOX/RADİO BOTUNLARININ DOĞRULAMA İŞLEMİ HANGİ METHOD İLE YAPILIR?

Bunun için **isSelected();** methodunu kullanırız. (seçiliyor mu)

Syntax olarak aşağıdaki dizilim örnek verilebilir.

```
driver.findElement (By.xpath("xpath of the checkbox/radio button")).isSelected();
```

NOT: **Bu methodun return değeri true ise doğrulama gerçekleşir değilse gerçekleşmez**

14. TESTNG'DE KULLANILAN NOTASYONLAR HANGİLERİDİR?

@Test = Method'u test case olarak işaretler ve calistirir. Ayrica bir Main Method'a ihtiyac duymaz.

@BeforeSuite = Bir suite'te bulunan bütün testlerden önce (sadece bir defa) çalışır.

@AfterSuite = Bir suite'te bulunan bütün testlerden sonra (sadece bir defa) çalışır.

@BeforeTest = @Test notasyonundan önce (sadece bir defa) çalıştırılır.

@AfterTest = @Test notasyonundan sonra (sadece bir defa) çalıştırılır.

@BeforeClass = Bu notasyonun tanımlandığı method, ilk test methodundan önce (sadece bir kere) çalıştırılır.

@AfterClass = Bu notasyonun tanımlandığı method, classdaki testler çalıştıktan sonra (sadece bir kere) çalıştırılır.

@BeforeMethod = Her methodtan önce çalışır. (@Test notasyonu kadar çalışır)

@AfterMethod = Her methodtan sonra çalışır. (@Test notasyonu kadar çalışır)

15. EXCEL'DEN DATA OKUMA NASIL YAPILIR?

Öncelikle şunu belirtelim ki WebDriver, excel dosyalarının veri okumasını doğrudan desteklemez. Bu nedenle, herhangi bir Microsoft ofis belgesini okumak/yazmak için **Apache POI gibi bir eklenti kullanmanız gerekir.**

```
String dosyaYolu = "src/resources/ulkeler.xlsx";

FileInputStream fis = new FileInputStream (dosyaYolu);
Workbook workbook = WorkbookFactory.create (fis);
String actualData = workbook
    .getSheet("Sayfa1")
    .getRow(3)
    .getCell(3)
```

.toString();
System.*out*.println(actualData);

→ Bu komut ile önce dosya yoluna ulaşmak ardından da dosyayı açabilmek için FileInputStream objesi create etmek gerekir. Daha sonra FileInputStream objesinin parametresine dosya yoluna verdiğimiz isim parametre olarak gönderilir. Bu iki adımdan sonra Workbook objesi create edilir ve parametresine FileInputStream objesinin ismi gönderilir. Workbook objesine verdiğimiz isim String bir dataya atanır ve sırasıyla istenilen sayfanın, istenilen satırının, istenilen hücresi bir **toString();** methodu aracılığıyla String'e atanır ve yazdırılır.

16. ASSERT ve VERİFY ARASINDAKİ FARK NEDİR?

Assert- (hardAssert) Sonucu doğrulamak için kullanılır. Testin ilk fail olduğu aşamada execution durur ve diğer aşamalar test edilmeden test başarısız olur.

Verify- (softAssert)- Bu da doğrulama için kullanılır. Ancak hard assert gibi ilk fail olma durumunda execution kesilmez. Test sonuna kadar devam eder.

NOT: hardAssert JUnit'den itibaren kullanılan bir method iken softAssert methodu TestNG ile birlikte gelmiştir. Bu sebeple ikisini ayırmak için bu tanımlama kullanılmaktadır.

17. SOFTASSERT'ÜN KULLANIM ŞEKLİ NASIL OLMALIDIR.

Öncelikle softAssert'ü kullanabilmek için ilgili test classı içinde obje create etmek gerekir. Şayet birden fazla **@Test** notasyonunun olduğu bir çalışma ortamı varsa ve soft assert bunların hepsinde kullanılacaksa o zaman softAssert objesi test classı altında create edilmelidir. Sadece belirli bir **@Test** methodu içinde kullanılacaksa o zaman sadece o test methodu içinde create edilir. Obje create edilmesinden sonra yapılması gereken softAssert işlemlerinin yapılması ve en sonunda assert'un doğru olup olmadığı test edebilmek için AssertAll () methodunun çağrılması gerekir. Bu satir yazilmazsa **(AssertAll)** assertion gorevi yapilmamis olur. *Softassertion baslangic ve bitis satirlari arasindaki tum assertion'lari yapip bitis satirina geldiginde bize buldugu tum hatalari rapor eder.* Hiç hata yoksa ve assertAll ()'dan sonra kod devam ediyorsa otomasyonun da çalışması devam eder.

Örnek syntax aşağıdaki gibidir.

```
// softassert baslangici obje olusturmaktir
SoftAssert softAssert=new SoftAssert();
driver.get ("https://www.amazon.com");
String expectedTitle = "amazon";
String actualTitle = driver.getTitle();
softAssert.assertTrue (actualTitle.contains(expectedTitle),"title amazon icermiyor");
WebElement aramaKutusu = driver.findElement (By.id("twotabsearchtextbox"));
softAssert.assertTrue (aramaKutusu.isEnabled(),"arama kutusuna erisilemiyor");
aramaKutusu.sendKeys ("Nutella" + Keys.ENTER);
WebElement sonucYaziElementi= driver.findElement (By.xpath("//div[@class='a-section a-spacing-small a-spacing-top-small']"));
softAssert.assertTrue (sonucYaziElementi.isDisplayed(),"arama yapilamadi");

softAssert.assertTrue (sonucYaziElementi.getText().contains ("Kutella"),"sonuc yazisi Kutella icermiyor");
// kodlardan birinde hata varsa softassert burada durur.

softAssert.assertAll();
```

18. DEKSTOP POPUP'LAR VE WEB POPUP'LAR NASIL HANDLE EDİLİR?

Dekstop popup'lar selenium ile handle edilmez. Bunun için robot objesi create etmek gerekir. Ancak web popup'lar selenium ile handle edilebilir. Web tabanlı popupları handle etmek için uyarı penceresine geçmemiz ve Selenium WebDriver Alert API yöntemlerini çağırmamız gerekir.

→ **dismiss():** Popup'ı reddetmek etmek için,

→**accept():** Popup'ı kabul etmek için,

→**getText():** Alert üzerindeki yazıyı almak için,

→**sendKeys():** Alert box'a yazı göndermek için kullanılır.

Örnek Synatx:

```
Driver.getDriver().switchTo().alert().accept(); // TestNG formatı
```

driver. switchTo().alert().accept(); **// JUnit formatı**

19. SELENİUM'DA DROPDOWN MENU NASIL HANDLE EDİLİR?

Selenium'da dropdown menu select objesi, WebElement ve select objesinden türetilen

- ➔ **selectByVisibleText ();**
- ➔ **selectByValue ();**
- ➔ **selectByIndex ();** methodları ile ile handle edilir. **(İndex sıfırdan başlar)**

NOT: Bir webElement'in dropdown olduğundan emin olmak için HTML kodlarına bakmak gerekir. Burada select html kodu varsa dropdown menu de vardır.

NOT: getOptions(); methodu dropdown menude bulunan tüm öğeleri listelemek için kullanılır.

20. SELENIUMDA MOUSE İŞLEMLERİ NASIL YAPILIR. BİRKAÇ ÖRNEK İLE AÇIKLAYIN?

Mouse işlemlerini Actions classından türetilen methodlar ile handle edebilir. Öncelikle Actions classından action objesi create etmek ve parametreye driver'ı göndermek gerekir.

Syntax = **Actions actions = new Actions(driver);**

Örneğin;

- ➔ **doubleClick();** ile bir webElement'e iki defa click yapılır.
- ➔ **dragAndDrop();** ile hedef webElement bir başka webElement'e sürüklenerek götürülür.
- ➔ **clickAndHold();** ile webElement üzerine click yapılarak mouse bası halde üzerinde bekletilir.
- ➔ **moveToElement();** ile bir webElement istenilen yere götürülerek orada bekletilir burada istenilen webElement'e tıklanılması için mouse bekletmiş olur.
- ➔ **contextClick();** ile istenilen webElement'e sağ tıklama yapılır.

NOT: Klavye üzerindeki işlemlerde de yine Actions classından istifade edilir.

- ➔ **sendKeys();** Sends a series of keys to the element

- ➔ **keyUp();** Performs key release
- ➔ **keyDown()**; Performs keypress without release

NOT: Actions classındaki methodları kullandığımız zaman kodu bitirince mutlaka **perform ();** methodunu kullanırız.

Örnek syntax= actions.sendKeys ("selenium"). perform();

21. SELENİUM İLE HANGİ TESTLER OTAMATE EDİLİR?

- ➔ Functional Tests (positive/negative, User Interface)
- ➔ Smoke/Sanity Tests
- ➔ Regression Tests
- ➔ Integration Tests
- ➔ End-to-end (System) Testing
- ➔ **Data Driven**

22. SELENİUM İLE OTOMATE EDİLMEYEN TEST TÜRLERİ HANGİLERİDİR?

Selenium sadece fonksiyonel test türlerini otomate ettiği için fonksiyonel olmayan testleri otomate edemez. Yani performans, stres, load test gibi testler kapsam dışıdır. Yine dinamik testlerin dışında kalan static testler de Selenium'un kapsamı dışındadır. Manuel ve Unit testler de selenium'un dışındadır. Ayrıca şunu ilave edelim ki database testing yapılacaksa genellikle Selenium'da ihtiyaç olmaz. (Appıum ile yapılabilir) Örneğin çok yaygın olarak kullanılan API, selenium'dan bağımsızdır.

23. FİNDELEMENT VE FİNDELEMENTS ARASINDAKİ FARKLAR NELERDİR?

Her iki method da **WebDriver interface'nin abstract methodudur** ve bir web sayfasında WebElement bulmak için kullanılır.

findElement (); web sayfasında bir adet elementi bulmak için kullanılır ve sadece bir WebElement tipi döndürür. WebElement'i bulamazsa exception (**NoSuchElementException**) fırlatır.

findElements(); birden fazla web element bulmak için kullanılır ve WebElements listesi döndürür. Şayet elementleri bulamazsa exception fırlatmaz boş liste döndürür.

24. TEST SIRASINDA KARŞILAŞTIĞINIZ BİR DEFECT'İ NASIL HANDLE EDERSİNİZ?

Öncelikle bunun bir bug olup olmadığını daha doğrusu canlıya çıkmadan önceki tanımlaması ile defect olup olmadığından emin olmak için bir kaç kez manuel ve otomasyon testi yaparım. Daha sonra ekip arkadaşıma danışarak bunun bir bug olup olmadığını sorarım. Buradan da bug olduğuna dair bir kanaat çıkarsa o zaman bunu rapor etmeden önce görev assing edilen developer arkadaşıma bu durumu bildiririm. Hızlı fix edilecek bir defect ise raporlamaya ihtiyaç duymadan sorunu takip ederim. Fakat developer ile görüşmeden sonra bunun hızlı fix edilmeyecek bir kusur olduğu anlaşılırsa o zaman defect olarak raporlar ve süreci tekrar takip ederim.

25. TESTNG İLE JUNİT ARASINDAKİ FARKLARDAN BAHSEDER MİSİNİZ?

- ➔ TestNG, JUnit ve NUnit'in gelişmiş versiyonu olmakla birlikte bazı yönlerden JUnit ile ayrışır. İlk olarak şunu diyebiliriz ki TestNg'de daha fazla notasyon vardır ve bu durum Page Object Model (POM) bağlamında işimizi oldukça kolaylaştırır. Bundan başka priority ve dependsOn methodları TestNG'de JUnit'den farklı olarak bulunmaktadır. Bu methodlar sayesinde testleri öncelik sırasına ve bağımlılığa göre çalıştırabilir ve elde ettiğimiz bu esneklik sayesinde otomasyonda handle edilmesi zor durumların üstesinden gelebiliriz.
- ➔ Paralel Testing ile birden fazla browser'in çalışmasını sağlar.
- ➔ @Data Provider sayesinde tek bir method ile birden fazla test case'in çalışmasını sağlayarak daha az kod yazma imkânı vermektedir. Bu pratikliği sayesinde kullanım kolaylığı getirir.
- ➔ JUnit daha çok birim (unit) test ile bilinirken TestNG daha geniş kapsalı test işlevselliğine sahiptir. Bu bağlamda Functional Testing, E2E (System) Testing ve Entegration Testing gibi testleri yapabilme yeteneğine sahiptir.
- ➔ Dependencies gerektirmeden JDK'nın default olarak tanımladığı runtime ve işlem kaydı (logging) özelliklerini kullanma kabiliyetine sahiptir.
- ➔ Xlm dosyası ile istenilen tüm package'lar, test methodları ya da tüm testler çalıştırılabilir.

NOT: <u>JUnit, birim testi için kullanılan Java tabanlı bir framework'dür.</u>

26. IFRAME HAKKINDA BİLGİ VERİNİZ?

IFrame bir web sayfasının içinden başka bir web sayfasına geçmek için kullanılan HTML etiketidir. IFrame genellikle bir Web sayfasına dokuman, video veya interaktif media gibi başka bir kaynaktan içerik eklemek için kullanılır. <iframe> tag'ı bir inline frame belirtir.

IFrame'i handle etmek için switchTo () mettod'unu kullanmak gerekir. Bunun 3 yolu vardir. İndex ile Id veya Name Value ile ve WebElement ile yapabiliriz.

- ➔ **driver (). switchTo (). frame (iframe'in index değeri)** // indexin sıfırdan başladığını belirtmek gerekir.
- ➔ **driver (). switchTo (). frame (" iframe'in id veya name değer")** // String olarak girilir.
- ➔ **driver (). switchTo (). frame (iframe'in webelementi)**

NOT: driver (). switchTo (). parentFrame (); methodu ile bir üstteki frame'e geri dönülürken,

- ➔ **driver (). switchTo (). defaultContent ();** methodu ile en üstteki frame geçiş yapılır.

27. AUTOMATİON TESTİNG NEDİR VE FAYDALARI NELERDİR?

Automation Testing; Herhangi bir yazılım veya uygulamanın beklenen kritelerini bir otomasyon test aracı kullanarak test etme sürecidir. Otomasyon testi yaparken amacımız actual sonucu excepted sonuç ile karşılaştırmak ve defectleri bulmaktır.

Automation Testing'in Faydaları;

- ➔ Kodun yeniden kullanılabilirliğini sağlar
- ➔ Kolay raporlama imkânı sunar
- ➔ Manuel Testing'e göre zaman ve paradan tasarruf ettirir.
- ➔ Farklı browser'larda paralele testing imkanı sağladığı için entegrasyon testi için uygundur.

NOT: Otomasyon testinin handikapları nedir?

28. JAVA'DAKİ THREAD.SLEEP(); METHODU HAKKINDA NE BİLİYORSUNUZ?

Kodların parametrede belirtilen süre kadar durdurulmasını sağlayan bu komut, hard wait olarak ifade edilebilir. Browser ile driver arasında meydana gelen senkronizasyon sorunlarını handle edebilmek için kullanılan bu method, kodların uyum içinde çalışması, kullanıcının veriyi ekranda görebilmesi gibi bir süreliğine tanınan zaman dilimidir. Selenium'dan gelen implicitly ve expilicitly wait komutlarının aksine kendisine tanınan süre boyunca kesin ve net bir şekilde bekleme yapar. Exception fırlatması nedeniyle kodların sağlıklı işlemesine mani olma ihtimali de olduğu için pek fazla tavsiye edilmez. Onun yerine Selenium'dan gelen wait komutlarını kullanmak daha faydalı olacaktır.

29. DİNAMİK TEST TİPLERİ TEMEL'DE KAÇA AYRILIR?

Test tipleri temelde **Functional** ve **Non-fuctional** olmak üzere ikiye ayrılır.

➔ Functional testler: Sistemin fonksiyonel özelliklerinin doğrulanmasını içeren test çeşitlerinin genel ismidir.

➔ Non-Functional testler: Sistemin; performans, güvenlik, dayanıklılık, ölçebilirlilik gibi işlevsel olmayan gereksinimlerinin test edilmesini içeren test türlerinin genel adıdır.

30. BİR QA OLARAK HANGİ TÜR TESTLERİ YAPTINIZ. BU TESTLER HAKKINDA KISACA BİLGİ VERİR MİSİNİZ?

➔ Smoke Test, Sanity Test ve Regression Testi.

➔ **SMOKE TEST;** Web uygulamasında hayati derecede önemli olan noktaların olabildiğince sık bir şekilde test edilmesidir. Örneğin bilet satış sitesinde bir kullanıcının bilet satın alma sürecinde bilgilerini girerek ödeme yapmasına kadar tüm detayların doğru bir şekilde çalışıp çalışmadığını test etmek smoke test kapsamına girer. Mesai başlangıcından önce yapılan ve yaklaşık 15 dakika süren bu test, gün içinde birden fazla kez de yapılabilir.

➔ **SANİTY TEST;** Daha derine inmeden uygulamanın ana işlevlerini kontrol etmek için sürüm aşamasında (Sağlık) Sanity Testi yapılır. Regresyon testinin bir alt kümesi olarak da adlandırılır. "Yayın düzeyinde" yapılır.

➔ **REGRESSİON TESTİ;** Yakin zamanda yapilan degisikliklerin mevcut ozelligi etkileyip etkilemedigini anlamak icin bu test yapilir. Yeni functionality eklendiginde

onlarin eski kodlara zarar verip vermediginin test edilmesidir. UPDATE ya da SURUM KONTROL TESTI de denilebilir.

- Regression Testi; Major Regression ve Minor Regression olmak üzere ikiye ayrılır.
- **Major Regression;** 3 aylik zaman dilimlerinde yapilan releaselerden sonra tüm user storylerin bastan sona test edilmesinden ibaret olan testtir.
- **Minor Regression;** Yakin zamanda yapilan degisikligin mevcut ozelligi etkileyip etkilemedigini anlamak icin yapilir. Sprint sonunda yapilir. O sprintte olusturulan functionalitylerin birbirine zarar verip vermediginin test edilmesidir. Her sprint sonunda_o sprintte gorusulen user storyleri bastan sona test etmeye denir.

31. UNİT TEST (WHITE-BOX) HAKKINDA BİLGİ VERİNİZ?

White-box testinin alt kategorisi olan Unit (Birim) Test, daha ziyade developerlar tarafından yapılır. Ancak gerekli kaynak kodlarının verilmesi halinde QA'ler de bu testi yapabilme becerisine sahiptir.

Unit Test, yaptığımız tüm fonksiyonların tek tek birbirinden bağımsız bir şekilde test edilmesidir. Entegrasyon testinden önce gelir ve yazılım testinin ilk seviyesidir.

32. BLACK BOX TESTİNG NEDİR? ÇEŞİTLERİ HANGİLERİDİR?

Kara Kutu Testi de denilen bu test türü functional testler kategorisindedir. Sistemin iç mimarisi ve kaynak kodlara gerek olmadan sadece girdiler ve çıktıların doğrulanmasının yeterli olduğu test türlerinin genel adıdır.

- ➔ FUNCTIONAL TESTING
- ➔ REGRESSION TESTING(Major/Minor)
- ➔ SMOKE TESTING (sanity test)
- ➔ USER ACCEPTANCE TESTING (UAT)
- ➔ SYSTEM TESTING (End to End Testing)
- ➔ MONKEY TESTING(Ad-Hoc Testing)

33. SELENIUM'UN TANIMI VE BILEŞENLERI HAKKINDA BILGI VERINIZ?

Selenium, web uygulamalarını test etmek için kullanılan açık kaynaklı bir otomasyonu suiti'dir. Dört tane bileşene sahiptir.

- ➔ **1. Selenium IDE (Selenium Integrated Development Environment- entegre geliştirme ortamı):** Selenium IDE (Entegre Geliştirme Ortamı) bir Firefox eklentisidir. Selenium Suite'teki en basit çerçevedir. Komut dosyasını kaydetmemizi ve oynatmamızı sağlar.
- ➔ **2. Selenium RC (Selenium Remote Control):** Birden fazla testin sürekli olarak yapılmasını sağlar. Bir Selenium RC sunucusu ile çalışır.
- ➔ **3. Selenium WebDriver:** Komutları kabul eden ve bunları bir tarayıcıya gönderen bir tarayıcı otomasyon çerçevesidir. Tarayıcıya özgü bir sürücü aracılığıyla uygulanır. Doğrudan onunla iletişim kurarak tarayıcıyı kontrol eder.
- ➔ **4. Selenium Grid:** Selenium Grid, farklı makinelerde paralel olarak farklı tarayıcılarda testler yapmak için Selenium RC ile birlikte kullanılan bir araçtır.

34. SELENİUM GRİD NE ZAMAN KULLANILIR VE AVANTAJLARI NELERDİR?

Eş zamanlı bir şekilde çoklu platformlar ve browser'larda yazılan farklı veya benzer testlerin execute edilmesi işleminde kullanılır.

Bu sayede;

- ➔ Test senaryolarının paralel olarak yürütülmesine izin vererek test yürütme süresinden tasarruf sağlanır.
- ➔ Çoklu tarayıcı testine izin verir.
- ➔ Çoklu platformda test senaryoları yürütmemize olanak tanır.

NOT 1: SELENİUM GRİD'DE HUB: farklı makinelerdeki test yürütmelerini kontrol eden bir sunucu veya merkezi bir noktadır.

NOT 2: SELENİUM GRİD'DE NODE: Hub'a bağlı olan makinedir. Selenium Grid 'de birden fazla Node olabilir.

35. GRAY BOX TESTİNG HAKKINDA NE BİLİYORSUNUZ?

Beyaz kutu testinde testin yapılması için gerekli olan kaynak kodlarının bilinmesi durumu ile buna gerek olmayan kara kutu testlerinin bir arada yapılması anlamına gelen bir test türüdür. Bu test türü tester'ın yaptığı ve developer'ın yükünü azaltan bir test türüdür. Ancak tester'ın bu testi yapabilmesi için developerın geliştirdiği kaynak kodlara sahip olması ve bunları kullanabilecek beceriye sahip olması gerekir. Bir tester gerekli datalar elinde olduğunda bu test türünü yapabilecek yetkinliğe sahiptir. Özetle bu test ile hem sistemin iç gereksinimleri hem de girdiler ile çıktıların uyumu test edilir.

36. PERFORMANS TESTLERİ HAKKINDA BİLGİ VERİR MİSİNİZ?

Sistemin performansının beklenen ya da daha yüksek yük altında değerlendirildiği Non-Functional test türüdür. Performans testi non functional bir test olduğu için hataları bulma değil performans odaklı tıkanmaları ortadan kaldırmayı amaçlar. Bu sayede maliyetlerin azaltılması, müşteriden önce olası sorunların handle edilmesi, geliştirme süresinin ve altyapı maliyetlerinin azaltılması gibi faydalar sağlar.

Performans Testi Türleri;

- **Load Testing**: Uygulamanın beklenen kullanıcı yükleri altında gerçekleştirme yeteneğini kontrol eder. Amaç, yazılım uygulaması yayınlanmadan önce **performans darboğazlarını** tespit etmektir. Parametrelerin belirli olması gerekir, (örneğin 1000 kişi) kullanıcı sayısı vb. gibi.

- **Stress Testing**: Yüksek trafik veya veri işlemeyi nasıl işlediğini görmek için bir uygulamayı aşırı iş yükleri altında test etmeyi içerir. Amaç, bir uygulamanın kırılma noktasını belirlemektir. Sistemin hangi aşamada cevap veremediği önemlidir.

NOT: Tasarlanan sistem için belirlenen kişi sayısının üzerinde bir kullanıcı sayısı ile test etme işlemi.

- **Endurance Testing**: Sistemin sürekli beklenen yükü ne kadar devam ettirebileceğini belirlemek için kullanılır. Örneğin 1000 kullanıcı limitine sahip bir uygulamda bu 1000 kullanıcının belirlenen süre kadar sistemde kalabildiğini test etme.

- **Spike Testing**: Anlık yükselmelerde sistem nasıl davranıyor ya da kullanıcı sayısı anlık düştüğünde sistem nasıl davranıyor?

- **Volume (Flood) Testing:** Amaç, yazılım uygulamasının performansını değişken veritabanı hacimleri altında kontrol etmektir. Veritabanındaki anlık yük artışında sistem nasıl davranıyor kontrol etmektir.

- **Scalability Testing: Ölçeklenebilirlik testinin amacı, kullanıcı yükündeki artışı desteklemek için yazılım uygulamasının "ölçeklendirme" konusundaki etkinliğini belirlemektir. Amaç, uygulamanın ölçeklendirmeyi hangi noktada durdurduğunu ölçmek ve bunun arkasındaki nedeni tespit etmektir. Kullanıcı sayısı arttıkça sistem davranışı nasıl scale edilebiliyor mu gibi.**

37. TESTLERİNİZİ HANGİ ORTAMDA KOŞTURURSUNUZ?

Testlerimizi DevOps'un testerler için özel olarak hazırladığı ortamda execute ederiz. Buna *"Test Environment"* denir.

38. SQL'İN TARİFİNİ YAPAR MISINIZ?

Açılımı Yapılandırılmış Sorgu Dili (Structured Query Language) olan SQL, bir veri tabanı uygulamasıdır. SQL ile veri tabanındaki verileri getirebilir, update edebilir, değiştirebilir ya da silebiliriz. Böylece SQL, database'deki bilgilerin sağlıklı bir şekilde yönetilmesini sağlar. Veri tabanına ihtiyaç duyan her web uygulaması SQL'e de ihtiyaç duyacaktır. SQL'den başka sorgulama dilleri olmasına rağmen SQL'in ücretsiz olması, açık kaynaklı bir sorgulama dili olması ve kolay anlaşılır bir syntax'e sahip olması gibi avantajları onun muadillerine göre (PL/SQL, TCL, T-SQL) daha fazla tercih edilmesine neden olmaktadır.

39. SQL KOMUTLARINI KULLANABİLECEĞİMİZ İDE VEYA EDİTÖRLER HANGİLERİDİR?

➔ **MySQL**

➔ **Oracle**

➔ **Microsoft SQL Server**

➔ **MsSQL**

➔ **Access**

➔ **IBM Database**

➔ **Progress**

➔ **Sybase**

➔ **Firebird**

40. SELENİUM'DA PAGE OBJECT MODEL (POM) NEDİR? TARİF EDER MİSİNİZ?

POM, Selenium'da bir çeşit dizayn pattern (tasarım modeli)'dir. Bu tasarım modeli, yaygın olarak ortaya çıkan yazılım problemlerinin çözümü için bir standart ortaya koyar. Bu standart ile clean kod yazımı, daha az kod, daha kullanışlı bir tasarım ve zaman ile paradan tasarruf sağlanabilir. Naming Convention gereği doğru isimlendirmeler ile takım çalışmasında başka bir arkadaşımızın da kolaylıkla bu tasarımı kullanmasını sağlar ve daha sürdürülebilir bir framework meydana getirebiliriz. Bunu daha da somutlaştırmak için POM'a uygun bir framework'te nasıl dizaynettiğimizden de bahsetmek yerinde olacaktır. POM temelli bir framework'te (örneğin TestNG ya da Cucumber) pages, tests ve utilities package'ları oluşturularak bu packageler içine ilgili classlar yerleştirilir. Utilities package'ında driver'ımızı sürekli kullanacağımız için Driver classı, testlerimizin raporlarını almak için Test Base class'ı, **configuration properties dosyası ile test classlarımız arasında yine POM mantığı gereği key value alışverişi yapacak olan Config Reader classı** ve gerekirse bize kolaylık sağlaması için bir Reusable Method Classı koyarız. Pages package'ı içine çalışacağımız web uygulamasına ait web elementlerin yer aldığı classı ve test package'ı içine de testlerimizi koşacağımız classları koyarız. Test classları içinde gereksinimlerimize göre Driver classı ve page classlarını obje yardımı ile çağırır ve daha reusable bir şekilde testlerimizi execute edebiliriz. Özetle POM ile dizayn edilmiş framework, kodların daha reusable ve daha readable olmasını sağladığı gibi değişikliklerin daha kolay ve hızlı bir şekilde uygulanabilir olması dinamik bir framework ile çalışma imkânı da sağlar. Böylece zamandan ve paradan da tasarruf edilebilir.

41. SELENİUM'DA PAGE FACTORY NEDİR?

Page Factory, Selenium'da Page Object Model'in bir uygulamasıdır. Web elementleri bulmak için kullandığımız **@FindBy** notasyonunu kullanmamıza imkân sağlar. Ayrıca **PageFactory.initElements();** methodu, @FindBy notasyonunu ile tanımlanan tüm web elementlerin başlangıç noktasıdır. Page Factory Pom'un temel taşlarındandır. Pages package'ı içinde oluşturduğumuz page classı içinde bir constructor oluşturup **PageFactory.initElements();** methodu'nu kullanarak driver'ımızı page sayfasında kullanılabilir hale getiririz. Bu sayede page classı içinde yer alan web elementlerin driver'ı buradan bularak elde edilmesini sağlamış oluruz. Daha sonra elde edilen bu web elementler test classları içinde obje yardımı ile çağrılır ve testlerimizi POM mantığı çerçevesinde koşturabiliriz.

NOT: Hatırlayalım: @FindBy notasyonu bizim çalıştığımız TestNG ve Cucumber frameworklerinin her ikisinde de kullanılan bir notasyondur.

42. CUCUMBER'DA dryRun NEDİR VE NASIL KULLANILIR?

dryRun, cucumber ile inşa ettiğimiz framework'umuzde Runner class'ı içinde yer alan bir parametredir. İki değer alabilir.

- **dryRun=false** bu default değerdir ve Runner classından testlerimizi çalıştırmak isteğimiz zaman bu değeri seçmiş olmamız gerekir.
- **dryRun= true** testlerimizi calistirmadan sadece eksik adim olup olmadigini kontrol eder ve eksik adimlarin method'larını bizim icin olusturur.

NOT: true degeri secildiginde test scenario'su failed olacak sekilde bile olsa, eksik adim yoksa test passed yazar.

43. SENDKEYS() KULLANMADAN TEXT GİRİŞİNİ NASIL YAPARIZ?

Javascript ve wrapper class kombinasyonu ile bunu yapmak mümkündür. Aşağıdaki kod dizinini kullanarak girmek istediğimiz metni ilgili yere iletebiliriz.

```
public static void setAttribute (WebElement element, String attributeName, String value)
```

```
        {
            WrapsDriver wrappedElement = (WrapsDriver) element;
                                JavascriptExecutor driver = (JavascriptExecutor)
wrappedElement.getWrappedDriver();
                driver.executeScript ("arguments[0].setAttribute(arguments[1],arguments[2])",
element, attributeName, value);
        }
```

44. SELENIUM'DA SENKRONİZASYON DENİNCE AKLINIZA NE GELMEKTEDİR?

Sekronizasyon, testlerimizi çalıştırmak için kullandığımız driver objesi ile browser'ımızın uyumlu bir şekilde çalışmasını sağlamak demektir. İki aracın senkronize olmasının önünde bilgisayarın ve internetin hızından kaynaklı engeller olabileceği gibi üzerinde çalışılan web uygulaması ile ilgili de bazı sorunlar olabilir. Çünkü **kod ortamında çalışan driver, browser'dan daha hızlı bir şekilde ilerler.** Bunun önüne geçmek için wait komutunu kullanabiliriz.

NOT: Selenium'dan gelen bekleme komutları hangileridir sorusu sorulabilir:

Cevap: ***İmplicitly wait: ExplicitlyWait (); FluentWait ();***

45. WAIT KOMUTLARI HAKKINDA BİLGİ VERİNİZ?

Driver ile browser arasındaki uyumu sağlama ve testlerimizin sağlıklı bir şekilde çalışmasına olanak sağlamaya yarayan wait komutları genel olarak hard wait ve dinamik wait olmak üzere ikiye ayrılır.

Hard wait olarak nitelendireceğimiz **Thread.sleep();** komutu Java'dan gelir, kendisine verilen sure kadar kodların calismasini durdurur. Thread.sleep'in oldugu satirdan bir sonraki satira gecis sadece ve sadece kendisine tanimlanan surenin bitmesi ile mümkündür.

Dinamik wait'ler ise selenium'dan gelir. Belirlenen süreden önce elementi bulması halinde belirlenen süre kadar beklemez ve kodun çalışmasını devam ettirir. Tanımlanan süre içinde elementi bulamazsa exception fırlatır.

Dimanik wait'ler; implicitly, explicitly ve fluent olmak üzere üçe ayrılır.

➔ **İmplicitly wait ();** web sayfasının yüklenmesi ve web elementlerin bulunabilmesi için global bir bekleme süresi tanır. Belirlenen süreden önce element bulunursa bekleme süresi kadar beklemeden kodun çalışması devam eder.

Syntax:

***driver*.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(15));**

➔ **ExplicitlyWait ();** Genel bir bekleme suresi degil spesifik bir web element icin tanimlanmis, spesifik bir gorev icin beklenecek max sureyi belirler Bunun icin ilgili webElementin locate'i ve wait objesine ihtiyacimiz vardir.

NOT: Eger kullanilacak Webelement baslangicta html kodlarinda bulunamiyorsa, locate ve bekleme ayni satirda yapilabilir.

wait.until (ExpectedConditions.visibilityOf(HerakuappPage.deleteButton));

➔ **WebElement element = wait.until (ExpectedConditions.visibilityOfElementLocated(By.xpath("...")));**

46. SELENIUM'DA IMPLICITLY WAİT VE EXPLICITLY WAIT ARASINDAKİ FARKLAR NELERDİR?

İmplicitly wait (); Bir element ve birden fazla element için tekrar eden aramalar için kullanılır.

ExplicitlyWait (); özel bir web elementi için tanımlanmıştır. Tekrar kullanılması halinde bir kez daha ilgili komutun yazılması gerekir.

47. EXPLİCİT WAİT İÇİN KULLANILAN EXPECTED CONDİTİON'LAR HANGİLERİDİR?

➔ alertIsPresent()

➔ elementSelectionStateToBe()

- ➔ elementToBeClickable()
- ➔ elementToBeSelected()
- ➔ frameToBeAvaliableAndSwitchToIt()
- ➔ invisibilityOfTheElementLocated()
- ➔ invisibilityOfElementWithText()
- ➔ presenceOfAllElementsLocatedBy()
- ➔ presenceOfElementLocated()
- ➔ textToBePresentInElement()
- ➔ textToBePresentInElementLocated()
- ➔ textToBePresentInElementValue()
- ➔ titleIs()
- ➔ titleContains()
- ➔ visibilityOf()
- ➔ visibilityOfAllElements()
- ➔ visibilityOfAllElementsLocatedBy()
- ➔ visibilityOfElementLocated()

48. FLUENT WAIT HAKKINDA NE BİLİYORSUNUZ?

Fluent wait, dinamik waitler içinde selenium'dan gelen bir bekleme komutudur. Kullanımı çok yaygın olmamakla birlikte gerekli görüldüğünde kullanılabilir. Bu bekleme komutu driver'ın bir koşulu beklemesi için maksimum süreyi ve ***"ElementNotVisibleException"*** atmadan önce koşulu kontrol etmek istediğimiz sıklığı tanımlamak için kullanılır. Nesne bulunana veya zaman aşımı gerçekleşene kadar düzenli aralıklarla web elementini kontrol eder.

Örneğin bir web elementin yüklenme süresinin her zaman aynı olmadığı bir senaryo düşünelim Bu durumda ilgili web element, bazen 10 bazen ise 20 saniye veya daha uzun bir süre içinde yüklenebilir. Bu durumda 20 saniyelik Explicit Wait tanımlarsak bir exception atmadan önce belirtilen süreye kadar bekleyecektir. Bu tür senaryolarda, Fluent Wait öğeyi bulana kadar veya son zamanlayıcı bitene kadar farklı frekansta bulmaya çalışacağından, kullanımı ideal olacaktır.

Syntax:

Wait wait = new FluentWait(WebDriver reference)
.withTimeout (timeout, SECONDS)
.pollingEvery (timeout, SECONDS)
.ignoring (Exception.class);

Değer vererek bir örnek oluşturacak olursak;

Wait <WebDriver> wait = new FluentWait <WebDriver> (driver)
.withTimeout (30, TimeUnit.SECONDS)
.pollingEvery (5, TimeUnit.SECONDS)
.ignoring (NoSuchElementException.class);

Yukarıdaki örnek syntax'da görüldüğü gibi maksimum bekleme süresi 30 saniyedir ve 30 saniye olana kadar 5 saniyelik aralıklarla elementin görünürlüğü kontrol edilecektir.

49. SELENIUM WEB DRİVER'IN DESTEKLEDİĞİ DİLLER HANGİLERİDİR?

- ➔ **Java**
- ➔ **Python**
- ➔ **C#**
- ➔ **C++**
- ➔ **Php**
- ➔ **Perl**
- ➔ **JavaScript**

50. SELENIUM'DA TEXT BOX TEMİZLEME İŞLEMİ NASIL YAPILIR?

Otomasyonunu yaptığımız web uygulamasında text box kutucuklarına bazen göndermek istediğimiz text html kodlarının yazılma tarzından kaynaklı gitmeyebilir. Şayet locator doğru ise ilgili yerde temizlik yapmak gerekecektir. Bunun için **clear ();** methodunu kullanmak gerekir.

Syntax: **driver.findElement (By.xpath ("xpathofbox")).clear();**

51. SELENİUM ÖZELİNDE OOP KONSEPT'E ÖRNEK VERİR MİSİNİZ?

Utilities package'i içinde kullandığımız Driver class'ı OOP konseptin bizatihi kendisidir. Çünkü Selenium WebDriver bir Interface'dir. Biz bu interface'in child classlarından oluşturduğumuz obje ile **WebDriver driver = new ChromeDriver();** OOP konseptin temellerinden olan **polimorphism, abstraction ve inheritence** bu işlem ile bir defada gerçekleşir.

52. FRAMEWORK NEDİR VE TÜRLERİ NELERDİR?

Bir framework testin tutarlılığını sürdürmeye devam eden bir çeşit otomasyon yönergesidir. Bu sayede test yapılandırılması geliştirilir, daha az kod kullanılır, daha az kod bakımı yapılır, kodun yeniden kullanılabilirliliği geliştirilir. Framework sayesinde teknik olmayan test uzmanları koda dâhil olabilir.

Yazılım otomasyon testinde 5 farklı framework tipi kullanılır.

1-Data Driven Automation Framework: Veri odaklı test yapma ve aynı testte farklı dataların kullanılması işlemidir. **API veya Postman gibi**

2-Method Driven Automation Framework: Method temelli bir frameworkdür.

3-Modular Automation Framework: Test uygulamalarını küçük parçalara bölerek çalışma prensibi.

4-Keyword Driven Automation Framework: Kelime temeli test yaklaşımıdır.

5-Hybrid Automation Framework: Temel olarak farklı framework'lerin kombinasyonundan oluşur. Örneğin data driven ile keyword driven'ın kombinasyonundan selenium'da hibrit bir framework elde edilebilir.

53. DATA DRİVEN AUTOMATİON FRAMEWORK'UN AVANTAJLARI NELERDİR?

- ➔ Regresyon testi sırasında uygulamanın birden çok veri değeri seti ile test edilmesini sağlar.
- ➔ Test senaryosu verilerini yürütülebilir test komut dosyasından ayırır.
- ➔ Eylemlerin ve İşlevlerin farklı testlerde yeniden kullanılmasına izin verir.

- ➔ Test verilerini otomatik olarak oluşturur. Bu, büyük hacimli rastgele test verilerinin gerekli olduğu durumlarda yararlıdır.
- ➔ Esnek ve bakımı kolay kapsamlı kodun oluşturulmasıyla sonuçlanır.
- ➔ Geliştiricilerin ve test uzmanlarının test senaryolarının/komut dosyalarının mantığını test verilerinden ayırmasına olanak tanır.
- ➔ Test senaryolarının ve komut dosyalarının azaltılmasına yardımcı olan test senaryolarının birkaç kez yürütülmesine izin verir.
- ➔ Test komut dosyalarındaki değişikliklerin test verilerini etkilemesine izin vermez.

54. TEST OTOMASYON ARAÇLARINA NEDEN İHTİYAÇ DUYULUR?

Test süreci; test planının hazırlanması, testlerin çalıştırılması, testlerin raporlanması ve bakım gibi birçok süreçten geçer. Bir proje tamamlanmış olsa bile yeni eklentiler ve değişiklikler ile tekrar ele alınması ve update edilmesi gerekir. Bu süreçlerin sağlıklı ve eksiksiz bir şekilde yerine getirilmesi malum olduğu üzere oldukça külfetlidir. Zamandan ve paradan tasarruf sağlayarak tekrar eden bu adımları daha efektif bir şekilde yürütmek için test otomasyon araçlarının kullanılması faydalı olacaktır. Fakat ne kadar testler için otomasyon araçları kullanılsa da projedeki adımların sağlıklı ilerlemesi için manuel testlerin otomasyon testlerinden önce kesinlikle yapılmasında fayda vardır.

55. KEYWORD DRİVEN AUTOMATİON FRAMEWORK HAKKINDA BİLGİ VERİR MİSİNİZ?

Test adımlarının veya otomasyon scriptlerinin geliştirilmesini, test senaryolarından veya ilgili iş akından ayırmayı sağlayan dolayısıyla karmaşıklığı azaltan ve modülerliği arttıran bir yaklaşımdır. Bu framework tipinde hazırlanan keywordler farklı parametreler ile birden fazla test senaryosunda kullanılabilir hale gelir. Yazılan kodların tekrar kullanımı sağlanmış olur. Keyword temelli test framework'ü data driven'a çok benzerdir. Fakat bu yaklaşımda test sistemi verileri yerine oluşturulan kelimeler ile ilgili parametreler ön plandadır.

56. TESTİN HEDEFLERİ NELERDİR?

- ➔ Bütünlük, doğruluk ve uygunluk gibi fonksiyonel kalite özelliklerinin değerlendirilmesi,
- ➔ Güvenilirlik, performans, güvenlik, uyumluluk ve kullanılabilirlik gibi fonksiyonel olmayan kalite özelliklerinin değerlendirilmesi,

- ➔ Sistemin yapısının veya mimarisinin doğru, eksiksiz ve gereksinimlerde belirtildiği gibi olup olmadığının değerlendirilmesi,
- ➔ Değişikliklerin etkilerinin değerlendirilmesi: hataların giderildiğini onaylama,
- ➔ Hatanın veya yapılan değişikliğin istenmeyen değişiklikleri tetikleyip tetiklemediğini bulma.

57. TESTLERİN OLABİLDİĞİNCE ERKEN AŞAMADA BAŞLAMASININ TEMEL FAYDASI NEDİR?

Testlerin yazılım geliştirme yaşam döngüsünün erken aşamalarında yapılması, sonradan çıkacak yüksek maliyetli değişikliklerin azaltılmasını veya ortadan kaldırılmasını sağlar. Çünkü bir hatanın erken aşamada bulunması kullanıcıya ulaşmadan önce kolayca handle edilebilirken kullanıcıya ulaşmasından sonra ortaya çıkaracağı sorun daha büyük olabilir. Bu sebeple sorunun daha fazla büyümesine mani olacak şekilde ne kadar erken bulunursa o kadar maliyet yönünden etkisi az olacaktır.

58. API'DA "DE-SERİALİZATİON" NEDİR VE NASIL YAPILIR?

De-Serialization, Json formatını java objesine dönüştürme işlemidir. Bu işlem iki türlü yapılır. Birincisi Google tarafından üretilen "Gson" yöntemi, diğeri ise kullanımı Gson'a göre daha yaygın olan Object Mapper yöntemidir. Key-Value olarak bulunan datanın okunabilmesi için bu işlem yapılır.

59. SDLC'DE MAİNTENANCE'IN ÖNEMİNİ IZAH EDER MİSİNİZ?

Software Development Life Cycle'ın aşamalarından biri olan maintenance, bir projeyi gelistirmek, onu anlasilir kilmak, tekrar eden kodlardan kacinmak, updateleri tek merkezden kolayca yapabilmek ve projenin surdurulebilirligini saglamak için son derece önemlidir. Tanımdan da anlaşılacağı üzere bir proje müşteriye teslim edildikten sonra da eklemeler ve güncellemelerin yanı sıra ortaya çıkacak hataların düzeltilmesi için yeniden gözden geçirilmek zorundadır.

60. MAVEN NEDİR, NİÇİN GEREKLİDİR VE AVANTAJLARI NELERDİR?

Selenium ile kullandigimiz bir projede ihtiyacimiz olan tum kutuphaneleri kolayca projeye eklemek ve ihtiyacimiz oldugunda kolayca update etmek icin kullandigimiz bir tool'dur. Yani Maven, bir proje inşa ve yönetim aracıdır. Maven'in getirdigi standartlar

cercevesinde tum tool'lar GitHub'a ekledikleri kutuphanelerini dependency denen kucuk kod bloklarini kullanarak projemize ekleyebiliriz.

- ➔ Maven, Page Object Model (POM) konseptine dayalıdır.
- ➔ Geliştiricileri birçok ayrıntıdan korur.
- ➔ Açık kaynaklıdır.
- ➔ Farklı ide (eclipse, intellij vb.) lerde çalışabilir.
- ➔ Geniş tabanlı bir kullanıcı kitlesine sahiptir.
- ➔ Maven, en iyi uygulamaların geliştirilmesi için güncel ilkeleri bir araya getirmeyi ve bir projeye bu yönde rehberlik etmeyi kolaylaştırmayı amaçlamaktadır.
- ➔ Konfigrasyon için pom.xml dosyasını kullanır. Bu dosya projenin insasi, raporlaması ve dokümantasyonu için gerekli bütün bilgileri içerir.

61. BİR WEB SAYFASINA GİTMEK İÇİN KULLANDIĞIMIZ URL İÇİNDE "HTTPS" PROTOKULÜNÜ KULLANMAK GEREKLİ MİDİR?

Url içinde http veya https protokollerinin kısaltması girilmemişse ilgili web sayfasına driver'ımız bizi götürmeyecektir. Ancak "www" ifadesi olmadan örneğin; **driver.get ("https://amazon.com");** şeklindeki bir syntax ile ilgili url'u gidilebilir. Fakat url, **driver.get ("www.amazon.com");** şeklinde ise ilgili web sayfasına ulaşılamaz. Exception ortaya çıkabilir.

Açılımı Secure Hyper Text Transfer Protocol (Güvenli Metin Aktarma Protokolü) olan Https, web sitesinin güvenli olduğunu beyan etme şeklidir. Bu protokol ile web sitesine bir giriş isteği göndermiş olursunuz. Şayet ilgili web sitesine bu istek gitmezse haliyle giriş gerçekleşmeyecektir.

62. SELENİUM'DA DRİVER. GETWİNDOWHANDLE(); İLE DRİVER. GETWİNDOWHANDLES(); FARKİ NEDİR?

driver.getWindowHandle(); İçinde bulunulan sayfasının window handle değerini döndürür. Bu benzersiz bir değerdir.

driver.getWindowHandles(); İse Mevcut tüm sayfaların window handle değerini döndürür.

63. SELENİUM'DA HANDLE WİNDOWS METHODLARI HAKKINDA BİLGİ VERİNİZ?

➔ **Pencereler arasında geçiş yapma (switch)**

driver.switchTo().window(sayfa1HandleDegeri);

➔ **Yeni tab olusturup geçiş yapma (switch)**

driver.switchTo().newWindow(WindowType.*TAB*);

➔ **Yeni window olusturup geçiş yapma (switch)**

driver.switchTo().newWindow(WindowType.*WINDOW*);

64. BEHAVİOR DRİVEN DEVELOPMENT (BDD) DENİLİCE AKLINIZA NE GELİYOR?

BDD, Test Driven Development (TDD) yaklaşımının karmaşıklığını gidermek, daha basit ve anlaşılır bir test dili ortaya koymak için tasarlanmıştır. Konuşma dili formatında olan bu yaklaşım gerkhin language olarak "given", "when" ve "then" keyword'leri ile yaygın olarak kullanılır. TDD'de olduğu gibi test senaryoları yazılmakla birlikte müşteri ile bir telefon görüşmesi basitliğinde olan bir dil kullanılır.

➔ Koda dökülen bu dil ile paydaşlar ile işbirliğinin gelişmesi,
➔ Test gereksinimlerindeki beklenen çıktılar ile gerçekleşen çıktıların büyük oranda uyuşması,
➔ Proje bakım maliyetlerinin ve risklerinin en aza indirilmesi,
➔ Müşteri ihtiyaçlarının daha efektif bir şekilde karşılanması,
➔ Test dilinin kolay olması sebebiyle kodların yazımında ve güvenilirliliğinde kesinlik derecesinin yüksek olması gibi avantajlar sağlamaktadır.

NOT: Örnek vermek gerekirse Cucumber, Rest Apı ve Cypress ile BDD temelli test yaklaşımı gerçekleştirmek mümkündür.

65. JAVA'DA INHERITENCE'IN FAYDALARI NELEDİR?

OPP konseptin temel taşlarından biri olan inheritence, parent child ilişkisi çerçevesinde geliştiriciler için;

➔ Daha az kod yazma (less code)

➔ Bir class'da bir kez yazarak birçok yerde kullanma (reusability)

➔ Update edebilme kolaylığı (maintenance)

➔ Kod sadeliği ile classlar arası organizasyon sağlama (well organization)

Gibi birçok açıdan kolaylık sağlar.

NOT: *İnheritence'da classlar arası ilişki aşağıdan yukarıya doğrudur. Yani child class verileri Parent'tan alır. Parent asla child class'dan veri almaz.*

66. JAVA'DA THİS () VE SUPER () ANAHTAR KELİMELERİNİN ARALARINDAKİ FARKLAR NELERDİR?

➔ **super();** parent class'dan constructor ve variable cagirmak icin,

➔ **this();** icinde olunan class'da baska bir constructor ve variable cagirmak icin kullanilir.

NOT 1: super() ve this() constructor çağırırken constructor'in ilk satirinda olmalidirlar. Bu durumda bir constructor'da ikisinin birden olmasi mumkun degildir.

NOT 2: Ancak variable çağırdıkları zaman ilk satirda olma sarti olmadigi icin ikisi birlikte kullanılabilirler.

67. OVERRİDİNG VE OVERLOADİNG NEDİR?

Overriding; Methodun signature'ını değiştirmeden iki methoddan sadece birini çalıştırmaktır. Overriding üzerine yazma anlamına gelir. Override edilmiş method child class'da varsa buradaki method çalışır. Şayet child classta yoksa o zaman parent class'a gider ve buradaki override edilmiş methodu bulup onu çalıştırır.

Overloading; Java'da aynı isim ve parametre ile birden fazla method tanımlanmasına izin vermez. Bu sebeple overloading işlemi methodun signature'ını değiştirerek aynı isimde farklı iki method yaparak elde edilir. Methoda yeni bir özellik ekleyerek işlem yapılır.

Overloading işleminde method signature'ının değiştirilmesi;

➔ Parametre sayısının değiştirilmesi,

➔ Parametrelerin data type'larının değiştirilmesi

➔ Parametre sayısı aynı olmak zorundaysa farklı data type'larındaki parametrelerin sırasının değiştirilmesi ile mümkündür.

NOT: *Overriding ve Overloading methodlar, OOP konseptin temel taşlarında biri olan Polymorphism'e örnektir.*

68. JAVA'DA OOP KONSEPT'İ DÖRT TEMEL PRENSİP ÜZERİNDEN AÇIKLAYINIZ?

Açılımı Object Oriented Programming olan OOP, nesneye yönelik programlama demektir. Java, nesneye yönelik bir programlama olarak classları kullanarak objeler üretir ve sonrasında application'lar ile bunu tamamlar. Böylece ortaya OOP konsept çıkar. OOP konseptin dört temel unsuru;

➔ **İnheritence:** Daha önce oluşturulan class'ların üzerine yeni class'lar oluşturarak parent child ilişkisi çerçevesinde daha kullanışlı bir konsept oluşturmuş oluruz. Çünkü inheritence bize yeni oluşturulan class'ın var olan class'daki tüm method ve variable'ları kullanma imkânı verir. Bu sayede kodu bir kez daha yazmadan daha anlaşılır ve sade bir kod düzeni elde etmiş oluruz. İnheritence kullandığımız zaman Access modifier olarak protected tercih etmek faydamıza olacaktır. Çünkü bilindiği üzere protected erişim belirleyici bize aynı package'den başka child class olarak extends edilmiş proje içindeki tüm classlara erişim imkânı sağlar. **Şayet Access modifier'ı private olarak belirleşek ilgili method ya da variable'a ulaşmak mümkün olmayacaktır.** Elbette public kullanmak da inherit edilecek class üyelerine (variable ve methodlar) ulaşmak mümkün olsa da kısıtlama ile OOP konsepte en uygun olan protected erişim belirleyici seçmek daha mantıklıdır.

➔ **Encapsulation:** Hassas dataları korumak için kullanılan bir çeşit data saklama yöntemidir. Datalar private erişim belirleyici ile saklanır. Getter ve setter methodları ile istediğimiz methodların uygun olanını kullanarak diğer classlardan istediğimiz dataya ulaşabiliriz. Get methodu bize saklanan dataların değiştirilmeden korunmasını sağlarken set methodu saklanan dataların obje üzerinden update edilmesine imkan sağlar.

➔ **Polymorphism:** Çok biçimlilik anlamına gelen bu kavram inheritence ile iç içedir. Bir nesnenin farklı nesneler gibi davranması işlemidir. Overriding işlemi buna somut örnek olarak verilebilir.

➔ **Abstraction:** class isminde abstract keyword'ünü kullanmak zorundayız. Bu classı inherit edecek diğer classların tamamı class içinde tanımlanılan soyut methodu override etmek zorundadır. **Abstract class'larda obje oluşturamayız.** Bu sebeple üzerinde obje ile değişiklik yapılmasını istemediğimiz classlarımızı abstract yapabiliriz. Elbette bu bir zorunluluk değildir. Abstract classlarda constructor create edilebilir.

69. JAVA'DA GARBAGE COLLECTOR'UN ÇALIŞMA PRENSİBİ HAKKINDA BİLGİ VEREBİLİR MİSİNİZ?

Garbage collector gereksiz nesnelerin ve bilhassa null olarak tanımlanmış olanların toplanarak hafızada yer açılması işlemidir. Bu sayede gereksiz bilgiler temizlenerek çalışmanın hızlanması sağlanır. Garbage collector herhangi bir komut verilmeden de otomatik olarak çalışabilir. Ancak biz de kod yazarak **Runtime.getRuntime.gc();** garbage collector'un devreye girmesini sağlayabiliriz. Bir şekilde hafızada biriken nesneler, OutOfMemoryException fırlatabilir. Ayrıca şunu ifade etmek gerekirse garbage collector heap memory'de temizlik yapar ve JVM (Java Virtual Machine)'nin kontrolünde çalışır.

NOT: Runtime.getRuntime.gc(); methodu yazmasak da carbage collector otomatik çalıaşacaktır.

70. OVERLOADİNG VE OVERRİDİNG ARASINDAKİ FARKI AÇIKLAYINIZ?

OVERLOADİNG	OVERRİDİNG
Aynı class içinde yapılır bu sebeple interit içermez	Subclass ve Superclass olmak üzere iki class arasında gerçekleştiği için inherit içerir.
Return type'ın aynı olmasına gerek yoktur.	Return type'ın aynı olmalıdır.
Parametreler farklı olmalıdır.	Parametreler aynı olmalıdır.
Static polymorphism kuralı geçerlidir.	Dinamik polymorphism kuralı geçerlidir.
Bir method diğerini gizleyemez	Subclass'daki overriding super class'dakini gizler

71. ACCESS MODİFİER'LAR VE ARALARINDAKİ FARKLAR NELERDİR?

Java'da class'ların method'ların ve variable'ların erişim sınırını belirleyen erişim belirleyicilere Access modifier denir. En geniş kapsamdan en dar kapsama Java'daki erişim belirleyiciler public, protected default ve private olmak üzere dört tanedir.

- ➔ **public:** Proje içinde her classtan erişim sağlanabilir.
- ➔ **protected:** Aynı package içinden ve farklı package'larda olsa bile child classlarından erişim sağlanabilir.
- ➔ **default:** Sadece aynı package içinde olan classlardan erişim sağlanabilir. Metoda özellikle default yazmaya gerek yoktur.
- ➔ **private:** Sadece içinde bulunulan classtan erişim sağlanabilir.

72. LOCAL, İNSTANCE VE STATİK VARİABLE'LAR HAKKINDA NE BİLİYORSUNUZ?

- ➔ Java'da method içinde tanımladığımız variable'lar local variable olarak bilinir.
- ➔ Static Variable'ın bir diğer ismi class variable'dır. Class içinde bir constructor veya scope dışında static keyword'ü ile tanımlanır.
- ➔ Class içinde tanımlanmakla birlikte static olmayan variable'lara da instance variable denilir.

73. JAVA NEDEN BAĞIMSIZ BİR PLATFORM OLARAK BİLİNİR?

Java, Java Virtual Machine (JVM) sayesinde herhangi bir işletim sistemine bağımlı olmadan çalışabilir. JVM, sanal bir makinedir ve bu sayede Java kodları herhangi bir sistemde byte kodlara dönüşerek çalışabilir. Bu sebeple Java, bağımsız bir yazılım dilidir.

74. JAVA'DA DATA TYPE'LAR HAKKINDA BİLGİ VERİNİZ?

Data type'lar primitive ve Non-primitive olmak üzere temelde ikiye ayrılır. Primitive data type'lar char, boolean, byte, short, int, long, double ve float olmak üzere sekiz tanedir.

Primtive Data Type'lar;

- ➔ **boolean;** true ya da false döndürür.
- ➔ **char;** tek tırnak ' ' içinde özel karakter, sayısal değer ya da bir harf alabililir. Ancak sadece bir karakter alma özelliğine sahiptir.

➔ **byte;** Sayısal değerler içinde en dar çerçeveye sahiptir. 8 bit hafızası vardır.

➔ **short;** Sayısal değer alan primitive data type'lardandır. 16 bit hafızası vardır

➔ **int;** En yaygın kullanılan ve tam sayı değeri alan data type'lardandır. 32 bit hafızaya sahiptir.

➔ **long;** tam sayı değeri alan bir diğer sayısal data type'lardandır ve 64 bit hafızaya sahiptir.

➔ **double;** Ondalıklı sayılar için kullanılır. 64 bit hafızaya sahiptir. Kullanımı float'a göre daha yaygındır

➔ **float;** 64 bit hafızaya sahiptir. Ondalık sayılar için kullanılır. Bu data type'ı kullanırken sayısal verinin sonuna f harfi konulur.

Non-Primitive Data Type'lar;

String; double quat içinde her türlü metni yazabildiğimiz bir data türüdür. İmmutable'dır. Yani değiştirilemez bir yapıya sahiptir. Datası belleğin Heap memory kısmında tutulur. Referansı da stack memory de bulunur. Object sınıfına bağlıdır ve bir classı vardır.

Non-Primitive Data Type'lar oluşturulan her obje için geçerlidir ve primitive data type'lar gibi sınırlı değildir. İstediğimiz kadar Non-Primitive data type üretebiliriz. Object sınıfına bağlı olan tüm data type'lar birer nonprimitive data türüdür.

75. PRİMİTİVE VE NONPRİMİTİVE DATA TÜRLERİ ARASINDAKİ FARKLAR NELERDİR?

PRİMİTİVE	NONPRİMİTİVE
Bilgiyi hafızada tutar ve küçük harfle başlar.	Büyük harfle başlarlar. Bilgiyi sadece hafızada tutmazlar aynı zamanda methodları vardır.
Sabit bir sayıdadırlar yani sekiz tanedir.	Sınırsızdır. İstediğimiz kadar oluşturabiliriz.

76. JAVA'NIN ÇALIŞMA PRENSİBİ HAKKINDA BİLGİ VEREBİLİR MİSİZ?

Java run time'dır. Yani çalıştığı sırada hangi bilgilere sahipse ona göre çalışır ve verileri depolamaz. Yukarıdan aşağıya ve soldan sağa doğru kodları okur. Ancak bir constructor çağırmışsak çağrılan bu constructor'un sıralamasına göre satırlar arasında geçiş yapabilir. Main method varsa çalışmaya ilk olarak buradan başlar. Ancak main methoddan önce bir static blok varsa ilk çalışacak yer burasıdır. Yani statik blok her şeyden önce çalışır.

Java ayrıca case sensetive'dir. Yani küçük büyük karaktere karşı duyarlıdır. Örneğin "code" ile "Code" Java'da birbirine eşit değildir. Class isimleri büyük, method isimleri ise küçük harf ile başlar. Yazım stili olarak "camelCase" tercih edilir.

77. JAVA'DA AUTO WİDENİNG VE DATA CASTİNG NE DEMEKTİR?

Auto Widening; Dar bir data type'dan daha geniş bir data type'e otomatik olarak geçiş demektir. Örneğin int olarak tanımlanmış bir data türünün double'a assing edilmesi auto widening'e bir örnektir.

Data Casting; Narrowing yani daraltma işlemidir. Auto Widening'in tersine geniş bir data type'ın daha dar olan bir data type'a dönüştürülmesi işlemidir. Ancak bu durum otomatik olarak yapılmaz ve Java bir seçenek olarak inisiyatifi geliştiriciye verir. Cast yapma işlemi opsiyoneldir ve veri kayıplarının yaşanmasına karşı dikkatli olmak gerekir. Çünkü primitive data type'ların belirli değer aralıkları vardır ve bunların aşılması durumunda veri kayıpları yaşanabilir.

78. CONSTRUCTOR VE METHOD ARASINDAKİ FARKLARI AÇIKLAR MISINIZ?

CONSTRUCTOR	METHOD
Return type yoktur.	Return type olabilir.
Class adı ile aynı olmalıdır.	Class adı ile aynı olmayabilir.
Overload edilir fakat override edilmez	Hem overload hem de override edilebilir.
Örtülü (implicity) bir şekilde call edilir.	Açıkça (explicitly) call edilmelidir.

Obje'yi aktif hale getirmek için kullanılır	Bir objenin nasıl davranadığını temsil eder.

79. ARRAY VE ARRAYLİST ARASINDAKİ FARKLAR NELERDİR?

➔ ArrayList, esnektir ve bir sınırı yoktur. Buna karşılık Array'in bir sınırı vardır.

➔ ArrayList'in içinde sadece wrapper classları **(Integer String Character vs.)** kullanabiliriz. Array'da ise hem primitive data type'ları hem de wrapper class'ları kullanabiliriz.

➔ ArrayList, nesnelerin depolanması, Array ise primitive data type'ların tanımlanmasında kullanılır.

NOT: Bazı durumlarda Array mi yoksa ArrayList kullanılacağı konusundaki ikileme düşülebilir. Bu durumda öğe sayısı belli ve değişiklik ihtimali düşük ise Array seçmek mantıklı olabilir. Ancak öğe sayısı net değil ve değişme ihtimali yüksek ise o zaman ArrayList seçmek gerekir.

80. JAVA'DA STACK VE HEAP MEMORY HAKKINDA BİLGİ VERİP FARKLARINI BELİRTİNİZ?

➔ Her iki hafıza da Ram'da bulunur. Ancak stack, bellekten static olarak yer tahsisi için, Heap ise dinamik olarak yer tahsisi için görev yapar.

➔ Stack'da yer alan datalar doğrudan bellek içine yerleştirilir ve erişim hızlıdır. Heap ise runtime'da kullanılır ve erişimi uzun sürer.

➔ Stack'da veriler hemen silinebilirken Heap'de verinin silinmesi için Garbage Collector çalışmalıdır.

➔ Stack memory'nin alanı sınırlı olduğundan çok sayıda ve büyük veri type'ları belleğin hemen dolmasına neden olabilir.

81. AGİLE METODOLOJİ HAKKINDA GENEL BİR ÇERÇEVE ÇİZEBİLİR MİSİNİZ?

Kelime manası ile çevik demek olan Agile, yazılım geliştirmede kullanılan proje yönetimine özel bir yaklaşımdır. Sprint adı verilen proje yönetim süreçlerinde projenin ekip ruhu çerçevesinde olabildiğince gereksinimleri karşılamasını öngören bir yaklaşımdır.

Agile metodolojinin başlıca prensipleri:

➔ Azami derecede müşteri memnuniyeti,

- ➔ Olabildiğince erken teslimat,
- ➔ En yüksek düzeyde kaliteli bir çıktı,
- ➔ Müşteri ile süreç boyunca temasta olma,
- ➔ İyi organize olmuş ekip ruhu ile süreci yürütme,
- ➔ Takım ruhunu, disiplinini ve motivasyonunu sürekli yüksek tutma.

Faydaları;

- ➔ Waterfall methodojisine karşı bir tepki olarak ortaya çıkan bu yöntem riskleri azaltma ve projenin kaliteli bir şekilde bitirilmesini sağlama açısından oldukça faydalıdır.
- ➔ Planlı ve ne yaptığını bilen bir ekip ile disiplinli bir ortamda süreklilik sağlanır
- ➔ Projenin gereksinimlerini doğru tespit ile sağlıklı bir ortam sağlar.
- ➔ Ürünün mümkün olduğunca pazarda rekabetçi olmasını sağlar.
- ➔ **Hem çalışanın hem de müşterinin kazançlı çıktığı bir ortam sağlar.**

82. MAP'LAR VE ARRAY'LAR BİRDEN FAZLA NESNEDEN OLUŞMASINA RAĞMEN NEDEN COLLECTİON SINIFINA DÂHİL EDİLMEZLER?

Arraylar esnek olmadıkları yani uzunlukları değiştirilemediği için collection sınıfına dâhil edilmezken, Map'lar yapı olarak karmaşık ve farklı olduğu için bu sınıf içinde değerlendirilmezler.

83. JAVA'DA NULL POİNTER NEDİR?

Null, bir değer ataması değildir ve kendine has bir yapıdır. Primitive data type'larda kullanılmaz, sadece non primitive data type'larda kullanılır. Hiçbir değere eşit değildir. Örneğin String bir variable "null" ataması yaparsak hata vermeden konsol'da "null" çıktısı verir.

Syntax:

String str = null;

84. JAVA'DA CONSTRUCTOR'U TANIMLAYINIZ?

- ➔ Java'da obje oluşturmak için kullanılan kod bloğudur.

- ➔ Yeni bir obje create etmek için kullandığımız bu yapı kendine has bir özelliğe sahiptir ve ne method ne de variable olarak bilinir.
- ➔ Constructor'lar hem parametreli hem de parametresiz olur.
- ➔ Return type olmaz ve class ismi ile aynı olur.
- ➔ Class create edilir edilmez default olarak bir constructor Java tarafından oluşturulur. Kodların sağlıklı çalışması ve daha sonraki kod eklentilerini göz önüne alarak Java tarafından oluşturulan default constructor'ı yazarak görünür hale getirmek yerinde olacaktır. Access modifier'ı public'tir.

85. EXCEPTİON HANDLİNG NEDİR? BÖYLE BİR DURUMLA KARŞILAŞIRSANIZ NASIL HANDLE EDERSİNİZ?

Exception, kod blokları arasında oluşan anormal duruma denir. Örnek vermek gerekirse bir sayının sıfıra bölünmesi, kod bloğunun noktalı virgül (;) ile kapatılmaması, String bir değer döndüren data type'ına int bir data type girilmesi gibi nedeler exception'a sebep olacaktır. Basit hatalar eksik bilgilerin girilmesi, ya da yanlış girilen bilgilerin düzeltilmesi ile handle edilebilecekken, sıfıra bölünme *"arithmeticException"* gibi exception sorunlarında try catch bloğu kullanmak gerekir.

Try catch bloğu ile kodun exception fırlatmasına imkan vermeden kullanıcıya hata mesajı vermemiz sağlanır. Muhtemel hata try bloğu içine yazılır. Catch bloğu içine de hatanın tipi ve ardından istersek hata mesajı String olarak yazdırılabilir. Başka bir örnek verecek olursak Örneğin *"ArrayIndexOutOfBoundsException"* hatasında index dışında bir Array'in dönütünü istersek bize yukarıda ifade edilen exception mesajı dönecektir. Bunu da yine handle etmek için try catch bloğu içine alabiliriz.

NOT: Java, bazı durumlarda (örneğin excel'den dosya okuma gibi) exception gerçekleşmesi ihtimali karşı bizi uyarır ve method signature'ına ilgili exception türünü yazmamızı ister.

86. KARŞINIZA BİRDEN FAZLA HANDLE ETMENİZ GEREKEN EXCEPTİON'LAR ÇIKARSA NE YAPMANIZ GEREKİR?

Bu durumda birden fazla catch bloğu kullanmak gerekir. Yani bir defa try iki defa ise catch kullanarak hatayı handle edebiliriz. Ayrıca şunu ifade etmek gerekirse Java 7'den sonra gelen yeni bir güncelleme ile farklı exceptionları bitwise (|) kullanarak yan yana yazıp aynı catch bloğu içinde de handle etmek mümkündür.

Aşağıdaki örnekte bunu görmek mümkündür.

```
public class main {
public static vois main (String) [] args) {
try {
int x = 10; int y = 0;
System.out.println (x/y);
}
Catch (ArrayIndexOutOfBoundsException | ArithmeticException ex)
{
System.out.println (ex. toString ())); }
}
}
```

87. DOUBLE EQUAL SİGN (==) İLE EQUALS ARASINDAKİ FARK NEDİR?

Double Equal Sign (==) iki variable'ın birbirine eşit olup olmadığını kontrol ederken hem değere hem de referansa bakar. True döndürmesi için ikisinin de aynı olması gerekir.

Equals ise iki variable'ın birbirine eşit olup olmadığını kontrol ederken sadece değere bakar. Değer aynı ise true döndürür.

88. JAVA'DA VARARGS NEDİR?

Variable Arity ifadesinin kısaltması olan varargs, belirli bir türden sıfır veya daha fazla sayıda argümanı bir methoda tek seferde aktarmamızı sağlayan bir mekanizmadır. Arka planda bu mekanizma değişken sayıdaki argümanların toplanıp tek bir dizi içinde saklanması ve dizinin methoda geçirilmesi mantığıyla çalışır. (int … sayi) kod örneğinde olduğu gibi değişken sayıdaki argümanlar 3 nokta ile ifade edilir ve bu aynı zamanda dinamik bir yapıdır. İstenildiği kadar int data türünde argüman buraya gönderilebilir. Bir parametre içinde varargs sadece bir tane olur ve en başta yer alır. Varargs içinde aynı data türünde olan argümanlar olmalıdır. Farklı olan data türleri varargs olarak kullanılamaz.

```java
public static void main(String[] args) {
  // verilen sayilari toplayan bir method olusturmak istiyorum

  int a=10;
  int b=20;
  int c=30;
  int d=40;
  int e=50;
  int f=60;

  topla(a,b);  // iki sayinin toplami: 30
  topla(a,b,c); // uc sayinin toplami: 60
  topla(a,b,c,d); // varargs calisir
  topla(a,b,c,d,e); // varargs calisir
  topla(a,b,c,d,e,f); // varargs calisir

}

private static void topla(int... a) {

  System.out.println("varargs calisir");
}
```

89. API VE API TESTİNG HAKKINDA BİLGİ VERİNİZ?

Açılımı Application Programming Interface (Uygulama Programlama Arayüzü) olan API, iki uygulamanın birbiri ile iletişim kurmasını sağlayan bir aracıdır. Başka bir deyişle bir protokol kümesidir. Apı testing ise uygulamanın güvenlik, performans, işlevsellik ve dayanıklılığını kontrol ettiğimiz bir doğrulama sürecidir. Test yaparken bize önceden verilen Url ve test case ile database göndereceğimiz format belli olmalıdır. Bize dönüş yapacak formatta aynı şekilde olur ve doğrulama işlemi bu sayede yapılır. API testing uygulamanın görüntüsü ile ilgilenmeden doğru çalışıp çalışmadığına bakar.

90. API VE WEB SERVİS ARASINDAKİ FARKLAR NELERDİR?

➔ Bu iki protokol arasındaki en temel fark web servisin testing yaparken internete ihtiyaç duyması, **API'ın ise bu ihtiyaca gerek olmadan sorgulamalarını yapabilmesidir.**

➔ Bütün web servisler API iken, bütün API'ler web servis değildir. Çünkü web servisler API'ın gerçekleştirebileceği tüm işlemleri yapamayabilir.

NOT: Ancak ikisinin ortak yanı uygulamalar arasında iletişim kurmasından gelir.

91. RESTFUL API NEDİR VE SOAP İLE ARASINDAKİ FARKLAR NELERDİR?

Rest temelinde HTTP protokolü üzerinden haberleşen bir mimari yaklaşımdır. Bu mimari yaklaşımı kullanan API'lara da Restful API olarak adlandırıyoruz. Restful standardı daha eski ve katı kuralları olan SOAP servislere bir çözüm olarak geliştirilmiş ve tercih edilmiştir.

SOAP	REST
Bir protokoldür.	http protokolünü kullanan bir mimari yapıdır.
WDSL ile tasarlanması gerektiğinden kullanması zordur.	http methodları ile tasarlandığı için SOAP'a göre oldukça kolaydır. Minumum içerik ile veri alıp gönderebilir.
Yalnızca XML formatında veri alışverişi sunar	XML'in yanı sıra JSON, HTML ve TXT formatında da request ve respons yapılabilir.
Ön belleği okuyamaz	Ön belleği okuyabilir.
Daha güvenlikli olmasına rağmen Rest'e göre daha yavaştır. (Daha çok bankalar ve ödeme noktaları bunu kullanır)	SOAP'a göre daha hızlıdır. Bu mimari yapıyı daha çok sosyal medya siteleri ve mobil uygulamalar tarafından kullanır.

92. RESTFUL API'DA KULLANILAN HTTPS TALEP TİPLERİ NELERDİR?

- ➔ **Get:** Database'den veri çekmek için kullanılır.
- ➔ **Put:** Var olan bir kaydı güncellemek için kullanılır. (kaydın tamamı için)
- ➔ **Patch**: Var olan kayıt üzerinde kısmi bir değişiklik yapılmak istendiğinde kullanır.
- ➔ **Post**: Yeni bir kayıt oluşturmak için kullanılır.
- ➔ **Delete**: Var olan bir kaydı silmek için kullanılır.

93. API'DA ENDPOİNT NEYİ İFADE EDER?

Endpoint, database testing yapacağımız zaman bize verilen URL'dir. Bu Url iki kısımdan oluşur. Parametreye kadar olan kısım Base url, parametreden sonraki kısım ise path param olarak adlandırılır. Bu iki kısmın tamamına ise Endpoint denilir.

94. POSTMAN'DA SORGU YAPMAYI BİLİYOR MUSUNUZ? BİZE BİRAZ POSTMAN'İ TARİF EDER MİSİNİZ?

Evet! Postman kullanmayı biliyorum. Postman, manuel test yapmamıza imkân sağlayan bir tool'dur. Bu sayede otomasyon öncesi verilerimizi kontrol ederek formatı belirler ve otomasyon için ön hazırlık yapabiliriz. **İnsomnia, Testfully ve Firecamp** gibi muadilleri olmasına rağmen Postman daha kullanışlı olduğu için daha fazla tercih edilir.

- ➔ Postman ile elimizdeki API'yı hızlıca test edebilir ve hazırladığımız requestleri export ederek arkadaşlarımızla da paylaşabiliriz.
- ➔ Postman API geliştirmek için tasarlanmış bir işbirliği platformudur. Hem kendi başımıza hem de takım arkadaşlarımız ile çalışma yapabiliriz.
- ➔ Postman, bireysel olarak ücretsizdir. Ancak sınırsız takım çalışması ve kaynak paylaşımı için ücretlidir.
- ➔ Http methodları ile (get, post, put, patch, delete) kolayca data'yı çağırıp, gerekli update'leri de doğrulamaları yapabilir ve veriyi silebiliriz.

95. SWAGGER DOKÜMAN'IN İŞLEVİ NEDİR?

Swagger, Rest Apı'leri tasarlamamız için kullandığımız bir çeşit kılavuzdur. Açık kaynaklı bir araçtır. Rest Api'ler için bir arayüz sağlar ve bu sayede kaynak koda erişmeden önce Rest Api'lerin özeliklerini görmemize ve inceleme yapmamıza imkân sağlar.

Daha detaylı bir açıklama yapacak olursak şunu diyebiliriz: İki bilgisayar sistemi arasında veri alışverişini daha ziyade Rest API'ler ile yapıyoruz. Ancak Rest API'lerin bir standadı olmadığı için haberleşme açısından sorunlu bir durum ortaya çıkıyor. İşte Swagger burada devreye giriyor. OpenApı standardını kullanarak API'lerimiz hem insanların hem de makinelerin anlayacağı formatta ortak bir dilde buluşabiliyor. Open API sözleşmeleri sayesinde servis içeriğinin ne olduğunu anlayabiliyoruz. Ayrıca makineler için sözleşme kurallarına uyan *client* veya *server* kodu üretebiliyor. 40 üzeri dil ve uygulama platformunu destekleyen swagger, bu sayede günümüzde API geliştiriminde iyi bir platform olarak karşımıza çıkmaktadır.

96. API'DA PATHPARAM İLE QUERYPARAM HAKKINDA BİLGİ VERİNİZ?

Bir kaynağı tanımlamak ya da daha ayrıntılı nesneleri üzerinde işlem yapmak istiyorsak **pathParam();** kullanırız. Ancak öğeleri sıralamak ve öğeler üzerinde filtreleme işlemi

yapmak istersek o zaman **queryParam();** kullanırız. Query parametreleri URL'de soru işaretinin (?) sağ tarafında görünür. Path parametreleri ise bu işaretin solunda olur. Path parametleri sorgulamanın bir parçasıdır ve kullanılması kaçınılmazdır. Query parametreleri Java'daki Map'lar gibi key value ilişkisi bağlamında çalışır.

97. SELENİUM'DA DRİVER.CLOSE (); İLE DRİVER. QUİT(); ARASINDAKİ FARK NEDİR?

➔ **driver.close();** test sırasında açılan web sayfasını kapatır. Sadece komut verilen ve bir tane sayfayı kapatma yetkisine sahiptir.

➔ **driver.quit();** ise test otomasyon sırasında açılmış tüm web sayfalarını kapatma yetkisine sahiptir.

98. GÜNDE KAÇ TEST SENARYOSUNU AUTOMATİON YAPIYORSUNUZ?

Buna kesin bir rakam vermek mümkün değildir. Çünkü test senaryoları her zaman aynı kolaylık veya zorlukta değildir. Karmaşık test senaryoları bizi yavaşlatırken nispeten daha kolay olanlar hızlı bir şekilde otomasyon yapmamıza imkân sağlayabilir. Bu sebeple test senaryosunun durumuna göre 1 ila 3 arasında bir test senaryosu yapabilirim.

99. FRAMEWORK OLUŞTURDUNUZ MU?

Evet! Bir framework oluşturmayı kendi başıma yapabilirim. Proje çalışmam kapsamında da TestNg ve Cucumber temelli framework kurdum. POM tabanlı frameworkler ile çalıştım.

100. MAVEN DEPENDENCY NEDİR? NASIL KULLANDİNİZ?

Maven, projelerimizi inşaa etmek icin kullandigimiz bir tool'dur. Maven framework'umuzde ihtiyacimiz olan tum kutuphaneleri projemize getirmek icin dependency'leri kullanir. dependency'lerde kullanacagimiz kutuphanenin github adresi, ismi ve version bilgileri bulunur. Biz version no'sunu degistirerek dependency'i guncelleyebiliriz. Dependency'lerde bir degisiklik yapinca projemize degisikligin islenmesi icin sag-ust taraftaki maven'dan yenile demeliyiz.

101. ELİNİZDE BİR LİSTE VAR VE BU LİSTEDEKİ TEKRARLI ELEMANLARI SİLMEK İSTİYORSUNUZ NE YAPARSINIZ?

Listenin elemanlarını Collection'lardan olan Set'e atarım. Ancak List'deki elemanlar henüz tekrarlı bir liste halinde olduğu için bu defa Set içinde yer alan tekrarsız elemanların

olduğu data'yı yeni bir List oluşturarak bunun içine atar ve kalıcı bir şekilde tekrarsız bir List elde etmiş olurum.

102. DEPENDENCY VE PLUGİNS ARASINDA NASIL BİR FARK VARDIR?

Hem dependecy hem de plugins jar dosyasıdır. Fakat dependecy daha geniş bir iş çerçevesine sahip olup genel işleyişi düzenler. Yani bir alet çantasına benzetebiliriz. Örneğin JUnit, TestNG ya da Cucumber'dan istifade etmek istiyorsak bunu pom xml dosyasının içine dependecy olarak yerleştirir ve kullanırız. Bu genel çerçeveyi ifade eder. Fakat saha spesifik işlemler için küçük bir eklenti mahiyetinde olan pluginler kullanılır. Plugin zaten kelime manası ile eklenti demektir. TestNG'de paralel browser çalıştırmak için kullanılan surefire ve failsafe pluginleri örnek olarak verilebilir.

103. BİR WEBELEMENT'E YAZILMIŞ BİR TEXT'İ NASIL SİLERSİNİZ?

Bu iş için **clear();** methodu kullanılır. **webElement.clear();**

104. GETTEXT( ); İLE GETATTRİBUTE( ); ARASİNDAKİ FARK NEDİR?

getText (); bir webElement üzerindeki görünen yazıyı alır, **getAttribute ("attributeIsmi");** parametre olarak yazdığımız attribute isminin değerini bize döndürür. Dolayısıyla bir webElement icin birden fazla **getAttribute();** method'u kullanılabilir.

105. SELECT TAGI OLMADAN DROPDOWN MENÜYÜ NASIL HANDLE EDERSİNİZ?

Dropdown menude select tagı yoksa seçenekleri tek tek locate ederek clikclemek gerekecektir. Yani her seçeneği ayrı bir element olarak düşünüp planlama yapmalıyız. Bu biraz daha zahmetli ve uğraştırıcı bir iş olacaktır. İlk olarak dropdown menünün locateni almak ve tıklamak gerekecektir. Daha sonra ise ilgili dropdown alt menüye aynı işlemi yapmamız gerekir.

106. BİR TESTER'IN SORUMLULUKLARI NELERDİR?

- ➔ Test planın takip etmek
- ➔ Hataları tarafsız ve gerçekçi bir şekilde raporlamak
- ➔ Yazımlımcıyı değil yazılımı test ettiğinin bilincinde olmak
- ➔ Riskleri tarafsız bir şekilde değerlendirmek

➔ Hataları önceliklendirmek

➔ Gerçekleri paylaşmak

107. SELENİUM'DA OBJECT REPOSİTORY NEDİR?

Page Object Model bağlamında projemizde kullandığımız locatorları bir arada tuttuğumuz bir çeşit depodur. TestNG veya Cucumber gibi POM mantığı ile dizayn edilmiş frameworkler içinde create ettiğimiz class'da bu yapıyı kullanabiliriz. Object repository'de depoladığımız locatorlara verdiğimiz isimler sayesinde bu locator'ları test classlarında önce obje create edip ardından kolayca çağırarak kullanabiliriz.

108. SELENİUM'DA JAVASCRİPT KODLARINA HANGİ DURUMLARDA İHTİYAÇ DUYARIZ?

Bazı web elementler, html kodlarının yapısından kaynaklı tıklanmayabilir ve bu sebeple locator'ını almış olmamıza rağmen bizden istenen gereksinimleri yerine getirmemiz noktasında bize zorluk çıkarabilir. Böyle bir durumda Javascript objesi create ederek sorunu handle edebiliriz. Çünkü JavascriptExecutors, HTML ve Document Object Model (DOM)'a ulaşabilir ve yönetebilir. Bu ayrıcalığı sayesinde test performansını arttırabiliriz. Javascript methodlarına ayrıca web sayfasını aşağı kaydırma, ekran resmi alma ve **bir kutucuğa text gönderme gibi farklı işlemlerde de ihtiyaç duyabiliriz.**

109. API'DA HANGİ HTTP METHODLARI BODY KULLANIR?

RestApı ile sorgu yaparken kulandığımız https methodlarından get ve delete hariç diğer methodlarda bir body'ye ihtiyaç duyarız. Syntax'da put, patch ve post methotlarında hemen önce body kullanılır.

110. API'DA @JsonIgnoreProperties NOTASYONUNUN İŞLEVİ NEDİR?

Pojo class'dan alacağımız variable'ların doğrulama işleminde doğrulama yapılacak veri tabanından gelen dataların sayısı aynı olmayabilir. Bu durumda assertion'da sorun çıkmaması için bu notasyonu kullanır ve bazı dataları görmezden gelebiliriz.

@ JsonIgnoreProperties (ignoreUnknown = true)

111. FİNAL KEYWORD'Ü HANGİ İŞLEMLERİ YAPMAK İÇİN KULLANILIR?

Değeri sabit olan ve değiştirilmesine gerek duyulmayan dataların hataen değiştirilmesinin önüne geçmek için final keyword'ünü kullanabiliriz. Örneğin pi sayısının değeri (3,14) sabittir ve değişmez. Bunun gibi datalar için final kullanılabilir.

- ➔ Bir class final olarak tanımlanırsa bu classtan inherit yapılmaz.
- ➔ Method final yapılmışsa override edilemez.
- ➔ Parametre final olarak tanımlanmışsa bu parametrenin değeri değiştirilemez.
- ➔ Aynı şekilde yukarıda ifade edildiği gibi bir variable final olarak tanımlanmışsa daha sonra değeri değiştirilemez

NOT: Normalde bir variable'ı tanımlayıp daha sonra değer ataması yapabilirken. Final olarak belirlenmiş bir variable'da bunu yapamayız. Değer ataması tanımlama ile birlikte yapılmalıdır.

112. THROWS VE THROW ARASINDAKİ FARKLAR NELERDİR?

--Throws class isminden sonra kullanılır. O class'ın exception atabileceğini gösterir.

--Throw ise programın içinde exception atılmasını istediğimiz yerde kullanılır. (try catch'de kullanılır.)

113. BİR WEB SAYFASINDAKİ BÜTÜN LİNKLERİ BULMAK İÇİN NE YAPMALIYIZ?

HTML'de web sayfaları arasında geçişi sağlamak yani linkleri vermek için **<a>** tagı kullanılır. Bu sebeple bir sayfada bütün linkleri bulmak için By.tagName locator'ını kullanırız. Örneğin; **List <WebElement> linkListesi = driver.findElements (By.tagName ("a"));** syntax'ini kullanarak bu işlemi yapmak mümkündür.