Infosys (38 min)

1. Tell me about different locators

2.Tell me about Xpath axes

3. Synchronization methods

4. Where you had used Encapsulation

5.WHat is difference between Stringbuffer and StringBuilder which is synchrinized

Why we use synchronization

6.Write code for Waits

7. Write code for taking screenshots

8.Write code for Scrollingwindows

9.Write code for handling alerts

10. Write code for Handling windows

11. Tell me about API methods

12. Tell me about API codes

13 Tell me about Feature file

14. Tell me about TestNG

**1.Tell me about yourself**

Hello everyone I am Shubhangi Sirsat and I am from Pune. Currently I am working as QA in Expleo solutions and I am having total 4 years of experience in which i performed AUtomation as well as manual Testing. Whenever I talk about manual testing my roles and responsibilities are like whenever I receive user stories I have to analyse that and write test cases for same and after that we need to review test cases by 4 ways ie Self review, peer review,internal review, and external review.After review we have to create traceability matrix in which we have to map user stories against the test cases and once the test review part is done and I receive my test cases I start for execution and before starting with execution we need to perform smoke testing on the latest built. then I have to do functional testing non functional testing,Retesting. Basically in my organization once done with retesting for regression testing we use automation testing. For automation we use Selenium with java. Also I worked on cucumber with BDD framework Maven, cucumber TestNG and POM concepts as well. Apart from this I worked on Agile methodologies and we need to attend different ceremonies like Sprint planning meeting daily standup call,review meetings and retrospective meetings.

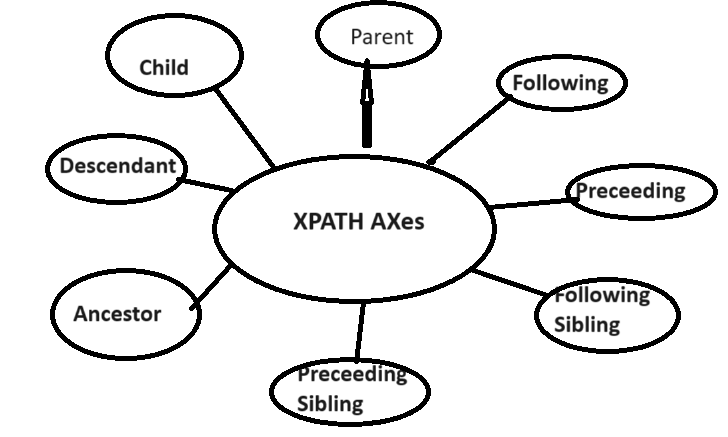
**2. Tell me about different locators**

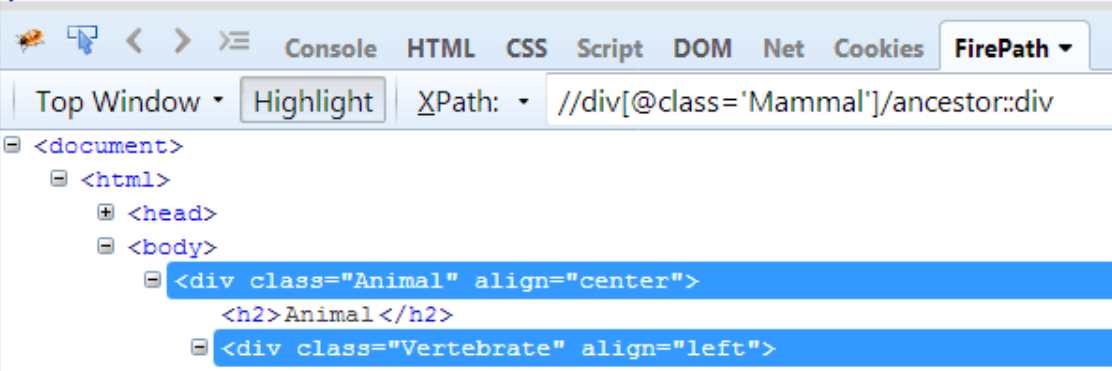
----> Basically locators are used to fined the location of webelements in dom(document obj model) structure. There are different locators like

ID, Name, ClassName, Link text, partiallinkext, TagName, CSS Selector

**2.Tell me about Xpath axes**

------> It is method to search multiple nodes in current DOM from the current node context. Mainly used for finding dynamic web elements like which changes dynamically when we refresh pages or do any other options and which are not possible to find with help of ID,name, class name, link text or css selector.





**3. Synchronization methods**

---->Thread.sleep()

Implicit wait() - driver. manage(). timeouts().implicitlyWait(Duration.ofSeconds(10));

Explicit Wait() -

WebDriverWait wait= new WebDriverWait(driver,20);

WebElemet e= wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(" xpath")));

Other methods related to conditions :

alertIsPresent()

elementToBeClickable()

elementToBeSelected()

visibilityOfElementLocated()

frameToBeAvaliableAndSwitchToIt()

invisibilityOfTheElementLocated()

invisibilityOfElementWithText()

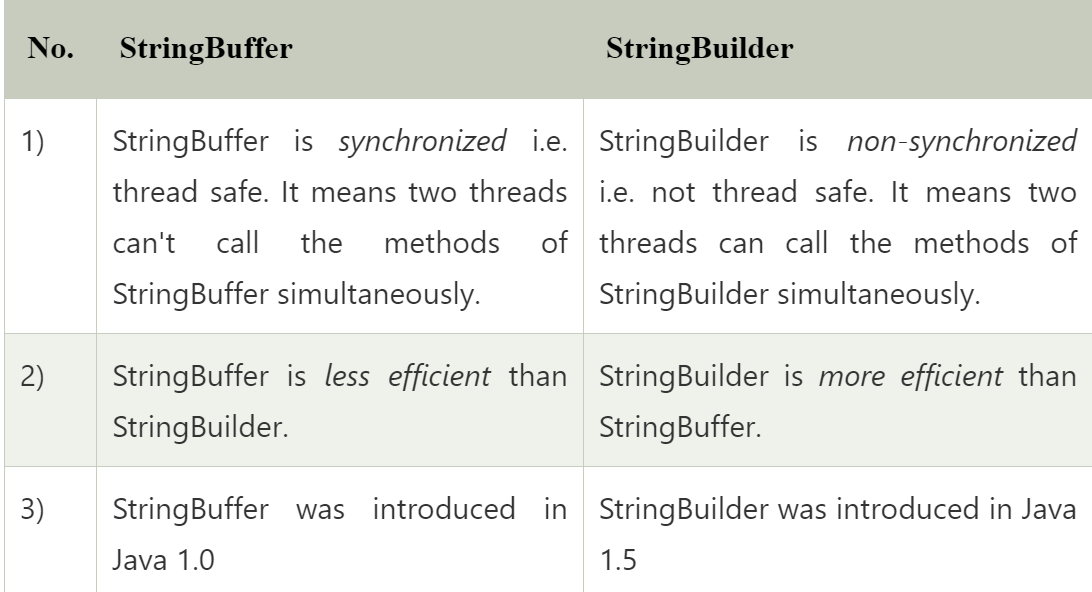
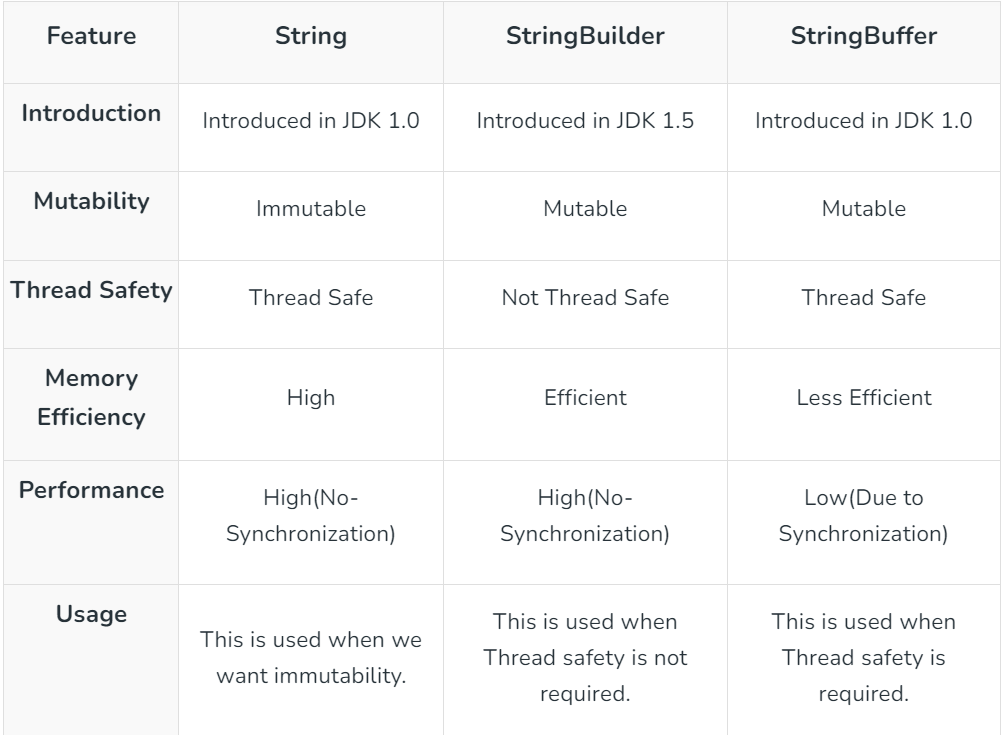
**4. Where you had used Encapsulation**

--->

***Encapsulation*** in Java is a mechanism of wrapping the data members and member functions together as a single unit. In encapsulation the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class, therefore it is also known as data hiding.

I have used encapsulation in POM pages where we declare our data members for webelement as private and all other getter and setter methods as public that can be accessed by outside test methods.

**5.WHat is difference between Stringbuffer and StringBuilder which is synchroinized**



**Why we use synchronization**

For thread safe means two threads can access method of String builder class simultaniously.

**6.Write code for Waits**

Inplicit wait: driver.manage().timeOuts().implicitlyWait(Duration.OfSeconds(5));

Explicit wait :

WebDriverWait wait=new WebDriverWait(driver,20);

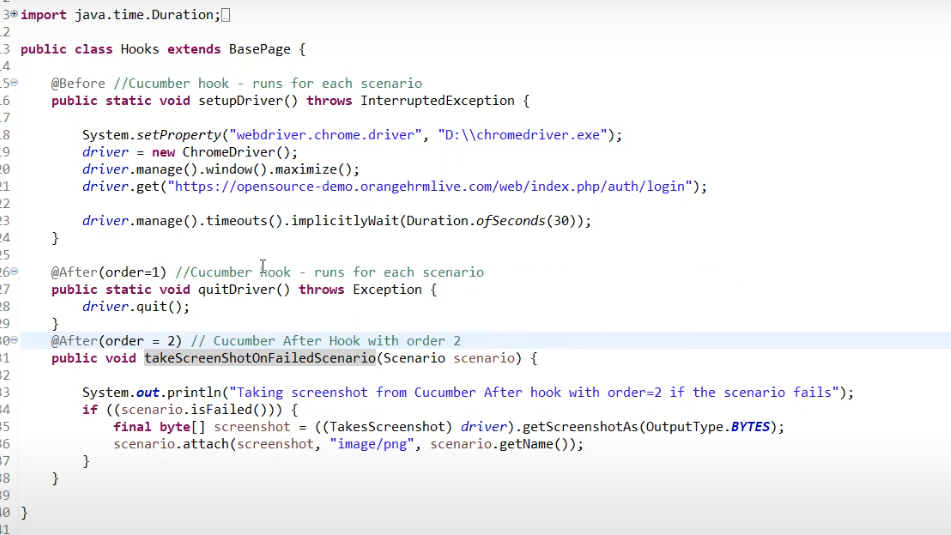
WebElement w= wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(“xpath”)));

**7. Write code for taking screenshots**

File src= ((TakeScreenshot)driver).getScreenshotAs(OutputType.FILE);

File dest=new File(“Path”);

FileHandler.copy(src,dest);



**8.Write code for Scrollingwindows**

JavaScriptExecutor js= JavaScriptExecutor(driver);

Js.executeScript(scroll(100,200));

**9.Write code for handling alerts**

**Driver.switchTo().alert().accept();**

**Driver.switchTo().alert().dismiss();**

**Driver.switchTo().alert().sendKeys(“Text”);**

**Driver.switchTo().alert().getText();**

**10. Write code for Handling windows**

//Printing window handles

**String parentHandle=driver.getWindowHandle();**

**Set<String> handle= driver.getWindowHandles();**

**For(String handles:handle)**

**{**

**//System.out.println(handles);**

**If(!handles.equals(parentHandle))**

**{**

**Driver.switchTo().window(handle);**

**Driver.findElement(By.Id(“ text”)).sendKeys(“Value”);**

**}**

**}**

// how to find how many number of widows open

Set<String> allWindowHandles = driver.getWindowHandles();  
ArrayList<String> tabs = new ArrayList<String>(allWindowHandles);  
System.out.println("No. of tabs: " + tabs.size());

//

**11. Tell me about API methods**

### **GET**

The GET method is used to retrieve data on a server. Clients can use the GET method to access all of the resources of a given type, or they can use it to access a specific resource. while a GET request to the /products/123 endpoint would return the specific product with an ID of 123. GET requests typically do not include a request body

**POST**

The POST method is used to create new resources. For instance, if the manager of an e-commerce store wanted to add a new product to the database, they would send a POST request to the /products endpoint. POST requests typically include a request body, which is where the client specifies the attributes of the resource to be created. For example, a POST request to the /products endpoint might have a request body that looks like this:

{  
 "name": "Sneakers",  
 "color": "blue",  
 "price": 59.95,  
 "currency": "USD"  
}

### **PUT**

The PUT method is used to replace an existing resource with an updated version. This method works by replacing the entire resource (i.e., the specific product located at the /products/123 endpoint) with the data that is included in the request’s body.

**PATCH**

The PATCH method is used to update an existing resource. It is similar to PUT, except that PATCH enables clients to update specific properties on a resource—without overwriting the others. For instance, if you have a product resource with fields for name, brand, and price, but you only want to update the price, you could use the PATCH method to send a request that only includes the new value for the price field. The rest of the resource would remain unchanged. This behavior makes the PATCH method more flexible and efficient than PUT.

### **DELETE**

The DELETE method is used to remove data from a database. When a client sends a DELETE request, it is requesting that the resource at the specified URL be removed. For example, a DELETE request to the /products/123 endpoint will permanently remove the product with an ID of 123 from the database. Some APIs may leverage authorization mechanisms to ensure that only clients with the appropriate permissions are able to delete resources.

**12. Tell me about API codes**

### **1xx informational responses**

These codes indicate that the server has received the request and is processing it. They are primarily used to manage communication between the client and server during the early stages of a request-response cycle. Some examples of this type of code are:

**100 Continue**: This status code indicates that the initial part of the request has been received and the server would like the client to send the rest of it.

**101 Switching Protocols**: This status code is used to inform the client that the server is changing the protocol that is being used in the connection.

**102 Processing**: This status code is an interim response that indicates the server is still processing the request.

### **2xx success responses**

These codes indicate that the client’s request was successfully received, understood, and processed by the server. Some of the most common 200 responses are:

**200 OK:** This status code indicates that the request was successful, and the server returned the requested data.

**201 Created:** This status code means that the request was successful, and the server created a new resource.

**204 No Content:** This status code indicates that the request was successful, but the server did not return any data.

### **3xx redirection responses**

These codes indicate that the client needs to take additional actions to fulfill the request. They are often used when the requested resource has moved to a different location. Some examples include:

**301 Moved Permanently:** This status code indicates that the requested resource has been permanently moved to a new URL. Clients should respond by updating their bookmarks and links to point to the new URL, and search engines should update their indexes with the new location.

**303 See Other:** This status code indicates that the response is available at a different URL, and the client should perform a GET request to that URL to retrieve the resource.

### **4xx client error responses**

These codes indicate that there was an issue with the client’s request, such as a mistyped URL or invalid credentials. The most common 4xx responses include:

**400 Bad Request:** This status code indicates that the request was malformed or invalid.

**401 Unauthorized:** This status code lets the client know that it is not authorized to access the requested resource.

**403 Forbidden:** This status code communicates that the client is authenticated but not authorized to access the requested resource.

**404 Not Found:** This status code indicates that the requested resource was not found on the server.

### **5xx server error responses**

These codes, which indicate that the server encountered an error while trying to fulfill the client’s request, include:

**500 Internal Server Error:** This generic error code indicates the server encountered an unexpected condition that prevented it from fulfilling the request.

**502 Bad Gateway:** This status code indicates that a server acting as a gateway or proxy received an invalid response from an upstream server.

**503 Service Unavailable:** This status code is returned when the server is temporarily unable to handle the request. It’s often seen during periods of increased traffic or when the server is undergoing maintenance.

**13 Tell me about Feature file**

A ***Feature File*** is an entry point to the Cucumber tests. Which is created with .feature extention. This is a file where you will describe your tests in Descriptive language (*Like English*). Where we use Gherking language.A feature file can contain a scenario or can contain many scenarios in a single feature file . Here we can write test steps in Given, When,Then, And format.and every step is this file is mapped with Steps in step definition files where we write logic for responsible step.For comman features we can use Background keyword

**14. Tell me about TestNG**

**-->**

*TestNG organizes tests into suites and classes. It utilizes annotations to define the execution order and dependencies between tests within a suite.*

*Some of TestNG’s notable features include:*

* ***Annotations:*** *@BeforeTest, @AfterTest, and others provide control over test execution flow.*
* ***Data-Driven Testing:*** *Allows running the same test with different data sets for comprehensive testing.*
* ***Grouping:*** *Organize related tests together for better management and execution.*
* ***Parallel Testing:*** *Enables running tests concurrently to reduce overall test execution time.*
* ***Reporting:*** *Generates detailed reports on test execution results.*
* [@BeforeSuite](https://www.geeksforgeeks.org/testng-annotations-beforesuite/)
* [@AfterSuite](https://www.geeksforgeeks.org/testng-annotations-aftersuite/amp/)
* [@BeforeTest](https://www.geeksforgeeks.org/testng-annotations-beforetest/)
* [@AfterTest](https://www.geeksforgeeks.org/testng-aftertest-annotation/)
* [@BeforeClass](https://www.geeksforgeeks.org/testng-annotations-beforeclass/)
* [@AfterClass](https://www.geeksforgeeks.org/testng-afterclass-annotations/)
* [@BeforeMethod](https://www.geeksforgeeks.org/testng-beforemethod-annotations/)
* [@AfterMethod](https://www.geeksforgeeks.org/testng-aftermethod-annotations/)
* [@BeforeGroups](https://www.geeksforgeeks.org/testng-beforegroups-annotations/)
* [@AfterGroups](https://www.geeksforgeeks.org/testng-annotations-aftergroups/)

**15. Tell me about your automation experience.**

**16. How you will execute smoke tests in Cucumber**

*Cucumber has already provided a way to organize your scenario execution by using* ***tags*** *in feature file. We can define each scenario with a useful tag. Later, in the runner file, we can decide which specific tag (and so as the scenario(s)) we want Cucumber to execute. Tag starts with “***@***”. After “***@***” you can have any relevant text to define your tag like* ***@SmokeTests*** *just above the scenarios you like to mark. Then to target these tagged scenarios just specify the tags names in the* ***CucumberOptions*** *as* ***tags = {"@SmokeTests"}****.*

*Other than Interview :*

### Is there a way to type in a textbox without using sendKeys()?

Yes! Text can be entered into a textbox using JavaScriptExecutor

JavascriptExecutor jse = (JavascriptExecutor) driver;

[jse.executeScript("document.getElementById(‘email').value=“abc.efg@xyz.com](mailto:jse.executeScript("document.getElementById(‘email').value=“abc.efg@xyz.com)”);

### How to upload a file in Selenium WebDriver?

You can achieve this by using sendkeys() or Robot class method. Locate the text box and set the file path using sendkeys() and click on submit button

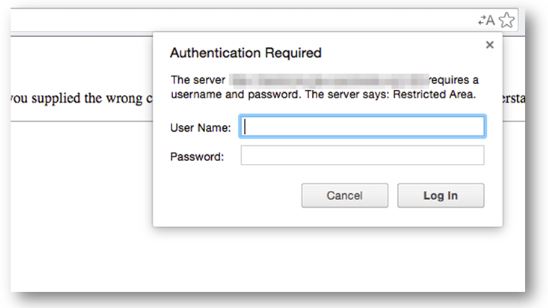
Locate the browse button

WebElement browse =driver.findElement(By.id("uploadfile"));

Pass the path of the file to be uploaded using sendKeys method

browse.sendKeys("D:\\SeleniumInterview\\UploadFile.txt");

### How to login to any site if it is showing an Authentication Pop-Up for Username and Password?



To handle authentication pop-ups, verify its appearance and then handle them using an explicit wait command.

Use the explicit wait command

WebDriverWait wait = new WebDriverWait(driver, 10);

Alert class is used to verify the alert

Alert alert = wait.until(ExpectedConditions.alertIsPresent());

Once verified, provide the credentials

alert.authenticateUsing(new UserAndPassword(<username>, <password>));

**Siemens Interview questions:**

***1. How to find Broken links :***

*Create a list of type WebElement to store all the Link elements in it.*

*for(int i=0; i<links.size(); i++)*

*{ WebElement element = links.get(i);*

*String url=element.getAttribute("href");*

*verifyLink(url);*

*}*

*Now Create a Connection using URL object( i.e ., link)*

*URL link = new URL(urlLink);*

*Connect using Connect Method*

*HttpURLConnection httpConn =(HttpURLConnection)link.openConnection();*

*Use getResponseCode () to get response code*

*if(httpConn.getResponseCode()!== 200)*

*Through exception, if any error occurred*

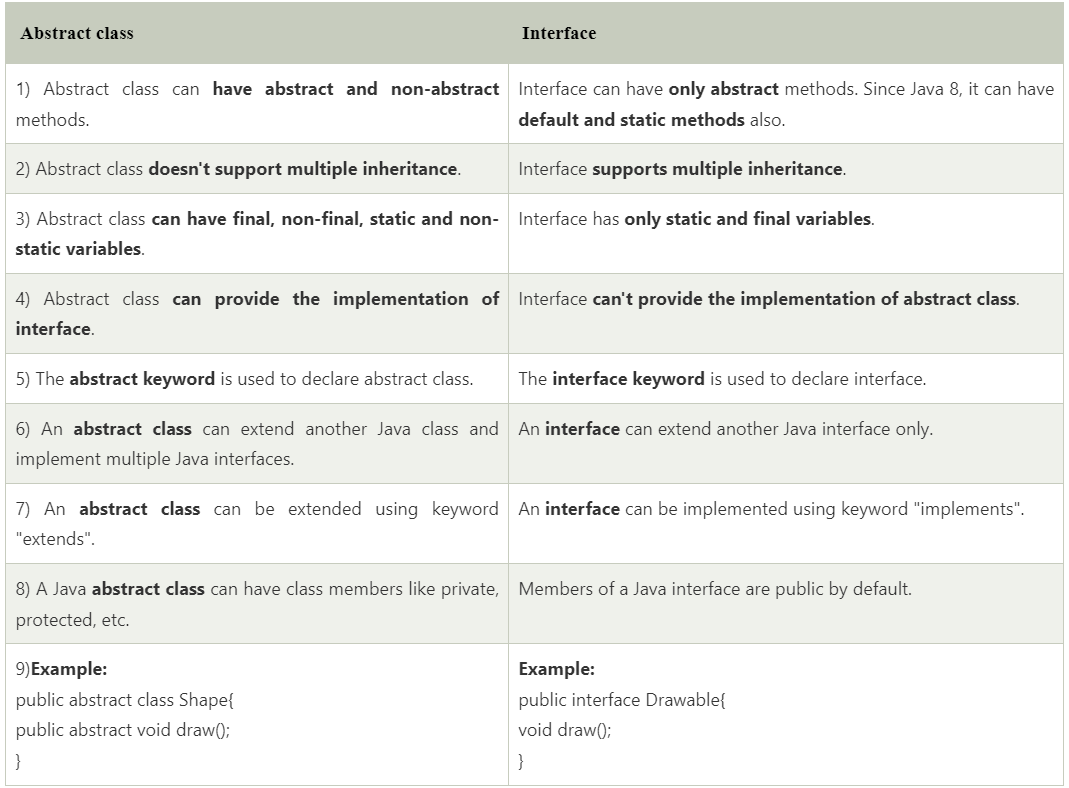
*System.out.println(“Broken Link”);*

**Git Pull Method?**

* The [git pull command](https://www.simplilearn.com/tutorials/git-tutorial/git-pull-request) is used to fetch and merge changes from the remote repository to the local repository.
* The command "git pull origin master" copies all the files from the master branch of the remote repository to the local repository.
* git pull <branch\_name> <remote URL>
* Git pull origin master

**One feature file to run 100 times how to do it ?**

**Abstact and Interface difference?**

  
 **Count characters from word**

String str= ”Count Characters”;

Char ch[] =str.toCharArray();

HashMap<Character,Integer> map= new HashMap<Character,Integer>();

For(int i=0;i<=ch.length();i++)

{

If(map.containsKey(ch[i]))

{

Int count=map.get(ch[i]);

Map.put(ch[i],count+1);

}

Else{

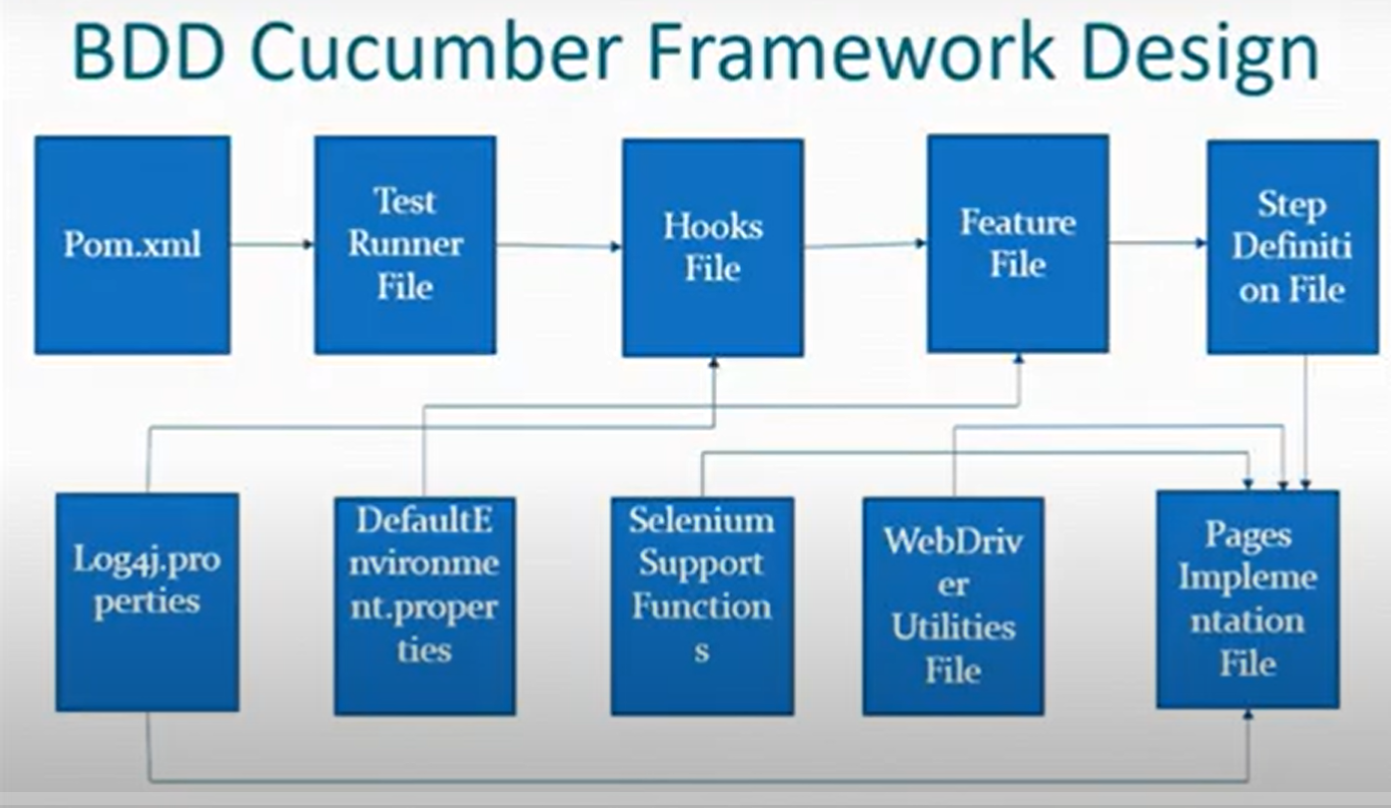
map.put(ch[i],1);

}

System.out.println(map);

}

**Framework folder structure**

  
 **How to automate Rest API?**   
 **Cucumber Disadvantages?**

### **Maintainability**

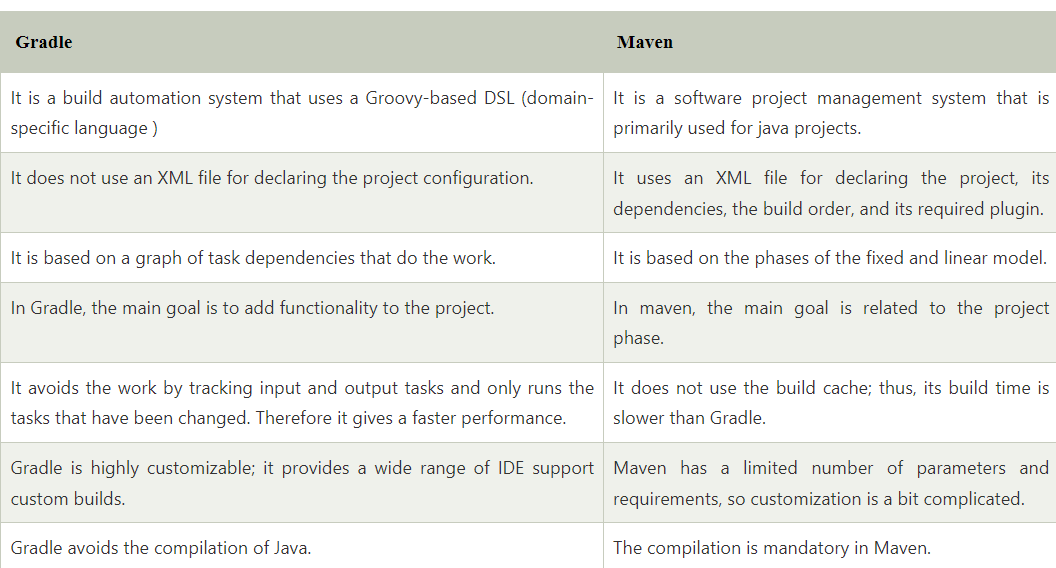
First, if you don't actually need to have others involved in the automation development, just having well-written methods & classes is more than enough. Cucumber can add an additional layer that just more tedious to maintain. As you can see, we have three different step definition classes just in this "simple" skeleton project.

### **Technical limitations**

Cucumber also has some technical limitations by design, chief among which being that you can't easily share information (aka state) between steps. You need to rely either on class-level fields or use additional libraries like Cucumber Picocontainer, Spring, or Guice.

A trivial example can be seen in the **frontend.feature** file. My expected page title contains a pipe character. This character is used by Cucumber to delimitate cells within a data table. Unless I escape the character, it would fail compilation due to a mismatch in table cell count.

**Mavan and Gradle difference?Maven disadvantage**



**Maven is limited to Java only**, whereas Gradle supports multiple languages and can be used to build projects in Java, C++, Python, and more.

  
 **When JAVA file created and when class file created**

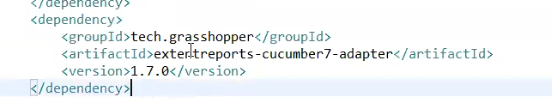
When we compile the Java source code (. java file), it generates a . class file. If a Java program has more than one class, in such cases after compiling the source file, we get the same number of .  
 **Where you give path for running feature file and class file ?**  
 **Scenario : If you are going on leave for one week for emergency reason (last week of sprint) , You have 20 scenarios to complete within sprint how you will do it ?**

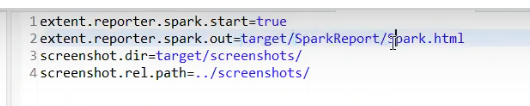
In this situation, Instead of spilling over all the sprint work, I would try and minimize the impact on the sprint. Run the priority test cases which are required to finish 1 or 2 critical stories.

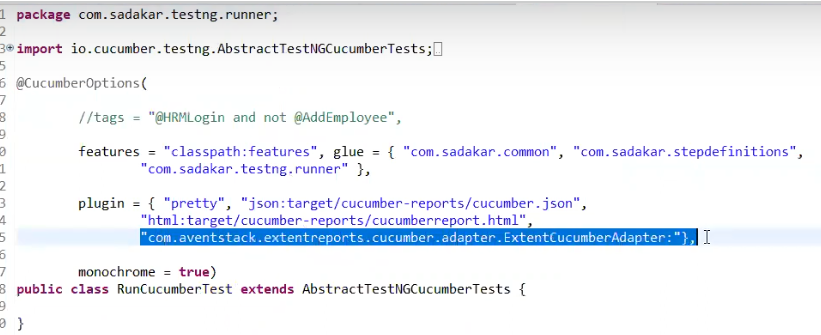
In case there are automation scripts available then impact on sprint can further be minimized.

Having said this, team will have to identify concrete action items so that such situation can be avoided in future. Inspect and Adapt!

**Which type of extent reports you are using in current framework ?**





  
 **JAVA code : count words from input string ? If you are written code then they will ask alternate method to write code(Tell them with collections such as set and HashMap)**  
 **From command line how you execute java code?**

Javac finename.java java filename

**Can Static method be overridden ?**

The short answer is No. Static methods in Java cannot be overridden. This is because static methods are not associated with the instance of a class, but with the class itself. Therefore, when a subclass inherits a static method from its parent class, it cannot modify the behavior of the static method in any way.

**Webtable having data to fill**

Feature :Login to Luma website

Scenario Outline: As a valid user user can login to website

Given User is on homepage of Luma website

When User gives valid <userId>

When User gives valid <password>

Then User successfully login with his account

Examples:

|userId | password |

| admin1| 12345 |

| user1| 123456 |

**How to upload files through selenium?**

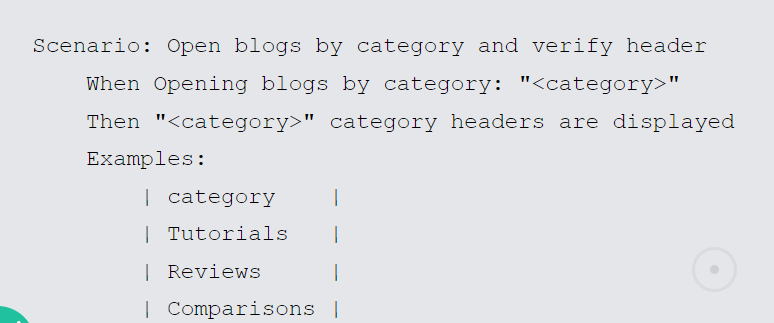
WebElement chooseFile = driver.findElement(By.id("custom-input"));

chooseFile.sendKeys("/Users/ankita/Downloads/edureka.png");

**When to use Scenario outline?**

The Scenario Outline keyword can be used to run the same Scenario multiple times, with different combinations of values. The keyword Scenario Template is a synonym of the keyword Scenario Outline . We can collapse these two similar scenarios into a Scenario Outline .  
 **How to execute One scenario with different data?**

That can be achieved using the Cucumber feature called [Examples](https://cucumber.io/docs/gherkin/reference/#examples). A special keyword that includes header names as a first row and their values starting from the second row. If for example, we have 3 rows in the examples that means that the scenario will run two times for different parameters.

  
 **How to swap two string without using third Variable?**

public class SwapString {

static String string1;

static String string2;

public static void main(String[] args) {

s1=s1+s2;

s2=s1.substring(0,(s1.length()-s2.length()));

s1=s1.substring(s2.length());

System.out.println("s1="+s1+" s2="+s2);

}

}

**How to find second max number and max number from array?**

public class MaxNumber {

public static void main(String[] args) {

int arr[]= {23,12,45,1,90};

int len=arr.length;

int max1=0;

int max2=0;

for(int i=0;i<len;i++)

{

if(arr[i]>max1)

{

max2=max1;

max1=arr[i];

}

}

System.out.println("max1="+max1+"max2="+max2);

}

**How to create Abstract class?**

**Abstraction** is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

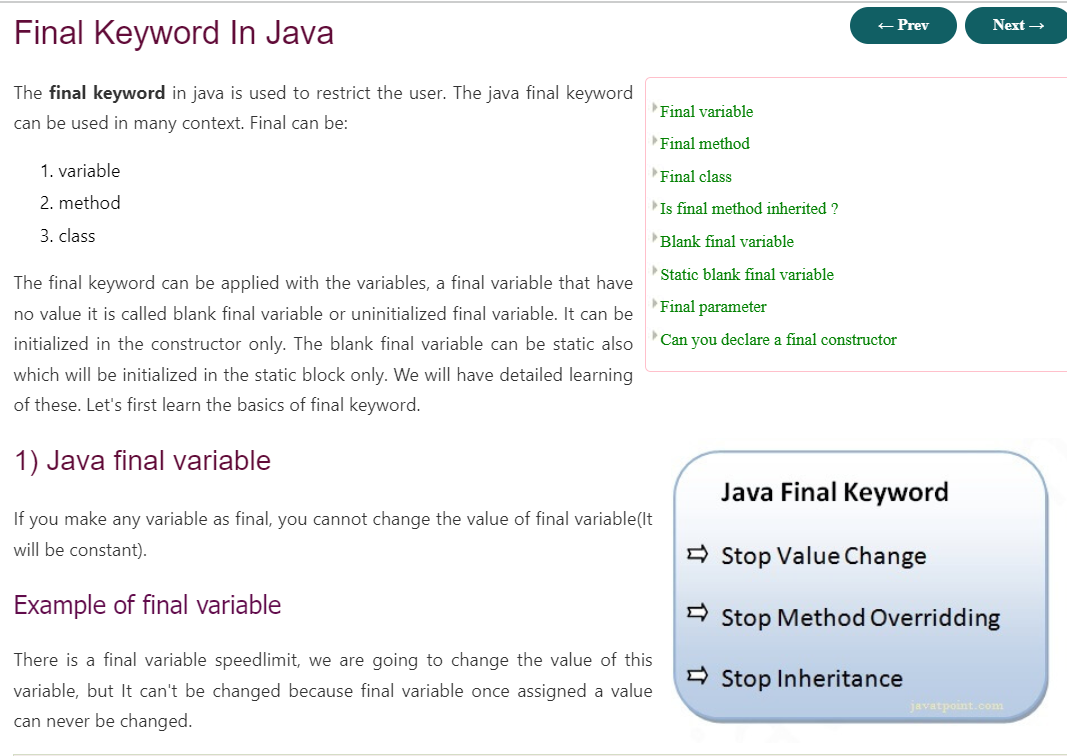
A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

#### Points to Remember

* An abstract class must be declared with an abstract keyword.
* It can have abstract and non-abstract methods.
* It cannot be instantiated.
* It can have [constructors](https://www.javatpoint.com/java-constructor) and static methods also.
* It can have final methods which will force the subclass not to change the body of the method.

1. **abstract** **class** Bike{
2. **abstract** **void** run();
3. }
4. **class** Honda4 **extends** Bike{
5. **void** run(){System.out.println("running safely");}
6. **public** **static** **void** main(String args[]){
7. Bike obj = **new** Honda4();
8. obj.run();
9. }
10. }

**Final keyword**

  
 **Git Push process**

git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin <https://github.com/Shubhangi-Sirsat/Testing-notes.git>git push -u origin main

### …or push an existing repository from the command line

git remote add origin <https://github.com/Shubhangi-Sirsat/Testing-notes.git>git branch -M main  
git push -u origin main

**Status code -2XX, 4XX, 5XX**

2XX –Client related

200 – request is successful returns OK

201 – Resource creted successfully on server

203 -

**204 No Content:** This status code indicates that the request was successful, but the server did not return any data.

### **3xx redirection responses**

These codes indicate that the client needs to take additional actions to fulfill the request. They are often used when the requested resource has moved to a different location. Some examples include:

**301 Moved Permanently:** This status code indicates that the requested resource has been permanently moved to a new URL. Clients should respond by updating their bookmarks and links to point to the new URL, and search engines should update their indexes with the new location.

**303 See Other:** This status code indicates that the response is available at a different URL, and the client should perform a GET request to that URL to retrieve the resource.

### **4xx client error responses**

These codes indicate that there was an issue with the client’s request, such as a mistyped URL or invalid credentials. The most common 4xx responses include:

**400 Bad Request:** This status code indicates that the request was invalid.

**401 Unauthorized:** This status code indicates that client is not authorized to access the requested resource.

**403 Forbidden:** This status code communicates that the client is authenticated but not authorized to access the requested resource.

**404 Not Found:** This status code indicates that the requested resource was not found on the server.

### **5xx server error responses**

These codes, which indicate that the server encountered an error while trying to fulfill the client’s request, include:

**500 Internal Server Error:** This generic error code indicates the server encountered an unexpected condition that prevented it from fulfilling the request.

**502 Bad Gateway:** This status code indicates that a server acting as a gateway or proxy received an invalid response from an upstream server.

**503 Service Unavailable:** This status code is returned when the server is temporarily unable to handle the request. It’s often seen during periods of increased traffic or when the server is undergoing maintenance.

**How to validate response code ?**

***If there is high priority bug then what will you do?***

*-> I will highlight it in DSM that it is blocker for us and we can not proceed because of that bug.*

**Who signs off on user stories?**

The team also relies on acceptance criteria to determine when the work for that story is completed. The product owner is in charge of accepting or rejecting the story and will only sign off if all the acceptance criteria are met.

***What happens after end of sprint***

A "sprint sign-out" typically refers to a process in Agile project management, particularly in Scrum methodology. At the end of a sprint, which is a set period during which specific work has to be completed and made ready for review, a "sprint sign-out" is a step where the team concludes the sprint by reviewing the work done and preparing for the next steps.

### **Who Does It?**

1. **Development Team**: The team members who have been working on the tasks during the sprint.
2. **Product Owner**: The person responsible for defining the features of the product and ensuring that the team delivers value to the business.
3. **Scrum Master**: The facilitator who ensures that the team follows Scrum practices and helps remove any impediments.
4. **Stakeholders (Optional)**: Sometimes, stakeholders or other relevant parties may be involved, especially if their input is needed for the next steps.

### **Key Activities During Sprint Sign-Out**

1. **Review Completed Work**: The team reviews the work that was planned for the sprint and confirms what was completed.
2. **Demonstration (Sprint Review)**: The team may demo the work to the Product Owner and stakeholders.
3. **Retrospective**: The team discusses what went well, what didn't, and how they can improve in the next sprint.
4. **Planning for Next Sprint**: The team may begin planning the next sprint, including selecting new tasks and setting goals.

The "sprint sign-out" can sometimes be part of the broader Sprint Review or Sprint Retrospective events, depending on how a team structures their processes. The goal is to ensure that the sprint is properly closed, feedback is gathered, and the team is prepared for the next cycle.

**scrum master-**

Scrum master duties may include:

* **Facilitating conversations** between members of the same team or across teams to resolve conflict and improve collaboration
* **Managing impediments** so the team can focus on getting work done
* **Coaching the team** toward better self-management

All of these elements can ultimately help the team use the iterative process of scrum to make products, services, and features their customers love.

**how you manage the risk:**

In Agile Scrum, managing risks is vital to ensuring the success of your projects.

**Identify and Document Risks**

-The first step is recognizing potential risks.

-Gather your team and stakeholders regularly to list possible issues related to scope, technology, dependencies, and team dynamics.

-Write down these risks in a central repository called a "risk register," including a clear description, expected impact, likelihood, and a plan to tackle them.

**Prioritize Risks**

Not all risks are equally important. Use a risk assessment matrix to rank them based on their potential impact and concentrate your efforts on high-priority risks.

**Transform Risks into Backlog**

Transform the high-priority risks to user stories, and place them in your project backlog. This ensures that the team and stakeholders are aware of these risks and that they become part of your project's work.

**Regular Risk Review Meetings**

We discuss the status of each risk in Sprint planning meeting update our mitigation plans. Use visual indicators like burndown charts to track how well you're managing these risks over time.

**Daily Stand-ups**

During daily stand-up meetings, we voice any concerns, impediments, or emerging risks. This allows you to address issues promptly.

**Retrospectives**

In your Sprint Retrospectives, focus on risk management and assess how the team is handling risks. Identify areas for improvement and make necessary adjustments.

**Adapt and Adjust**

If new risks arise or circumstances change, be ready to modify your risk management approach and update the risk register accordingly.

**Transparency and Communication**

Keep everyone informed. Share the risk register and updates with all relevant parties. Open and honest communication ensures that everyone knows the current status of risks.

**Collaborative Problem Solving**

Encourage teamwork. When risks arise, engage team members and stakeholders in resolving them. Assign risk owners and make sure they take responsibility for managing and resolving the risks.

**Root Cause Analysis**

When a risk arise, conduct a root cause analysis to understand why it happened. This insight will help prevent similar risks in the future.

**Regularly Update Risk Register**

Always keep the risk register up-to-date to reflect the current status of each risk. Make sure it's easily accessible to the team and stakeholders.

**Escalate When Necessary**

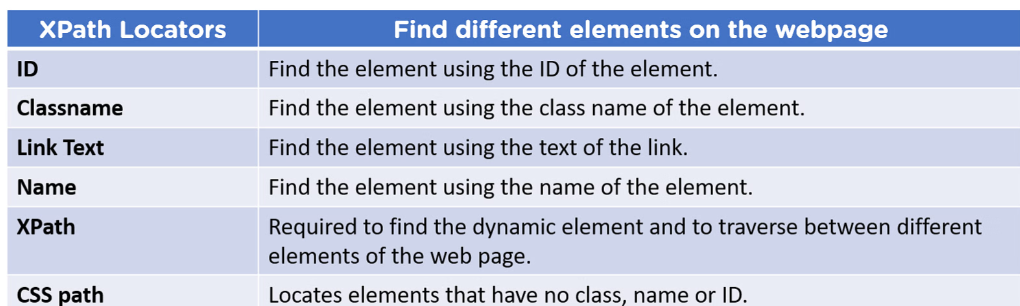
If a risk becomes too severe or beyond the team's control, we escalate it to higher management or stakeholders for resolution.

**Continuous Improvement**

Keep improving your risk management process based on feedback and lessons learned from previous projects.I

**1.How do you use locators and locator strategies?**

---> A locator is a way to identify elements in DOM. It is the argument passed to the [Finding element](https://www.selenium.dev/documentation/webdriver/elements/finders/) methods.



1. driver.findElement(By.className("information"));
2. driver.findElement(By.cssSelector("#fname"));
3. driver.findElement(By.id("lname"));
4. driver.findElement(By.name("newsletter"));
5. driver.findElement(By.linkText("Selenium Official Page"));
6. driver.findElement(By.partialLinkText("Official Page"));
7. driver.findElement(By.tagName("a"));
8. driver.findElement(By.xpath("//input[@value='f']"));

**Types of XPath in Selenium**

* Absolute XPath: Begins from the root of the HTML document and specifies the complete path to the element. It's not as flexible and can break if the page structure changes.
* Relative XPath: Starts from a specific element and navigates through the DOM hierarchy to locate the desired element.
* XPath Contains() function

It is used if part of the value of any attribute changes dynamically, like login information, etc. The function can navigate to the web element with the partial text present.

//tag\_name[contains(@attribute,’value\_of\_attribute’)]

### XPath Text() function

The XPath Text() is a function used to locate the element on a web page using the web element's text.

//tag\_name[text()= ’Text of the element’]

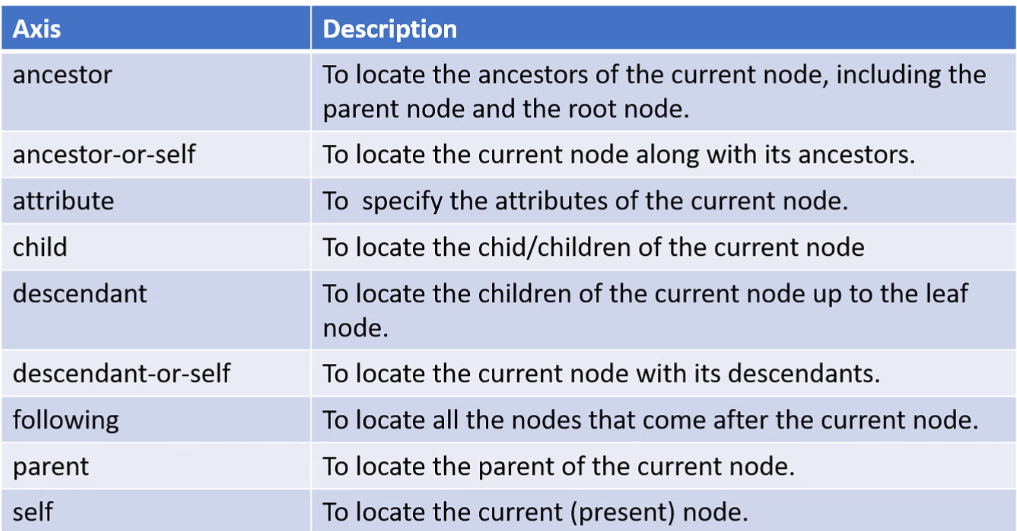
**XPath Starts-with() function**

The XPath Starts-with() function is used to find the element in which the attribute value starts with some specific character or a sequence of characters. The function plays a major role while working with the dynamic web pages.

//tag\_name[starts-with(@attribute,’Part\_of\_Attribute\_value’)]

## XPath Axes

Here, an axis shows a relationship to the current node and helps locate the relative nodes concerning the tree's current node. So, the XPath Axis uses the relation between several nodes to find those nodes in the DOM structure.

**XPATH for webtable**

**Maven command line execution :**

**2.How do you handle synchronization issues?**

--->Using waits : Thread.sleep(),implicit wait(), Explicit wait()

**3.How do you use multiple data sets in Cucumber?**

By using Scenario outline, and Example where we give variable names as headers and we give data seperated by delimeters under particular variables.

For single data we directly give data under test steps :

When I entered

|userId|admin|

|username|nikita|

|password|ffff|

**4.If there are multiple feature files and you want to run only a few, what will you do?**

- If you use IntelliJ then I suggest installing the Cucumber for Java plugin. Then you can right-click on the Test annotation in the feature file and run that single test scenario.

In Cucumber, tags are a powerful feature that allows you to filter which scenarios or features to run during your tests. You can assign tags to scenarios or feature files and then run only the scenarios that have specific tags.

### **How to Use Tags in Cucumber**

**Adding Tags to Scenarios:**  
You can add tags directly above the scenario or feature definition. Tags start with an @ symbol. For example:

@smoke   
Scenario: Verify login functionality   
 Given the user is on the login page   
 When the user enters valid credentials   
 Then the user should be logged in successfully   
  
@regression   
Scenario: Verify logout functionality   
 Given the user is logged in   
 When the user clicks on the logout button   
 Then the user should be logged out

**Tags on Feature Files:**  
You can also tag entire feature files. This is done at the beginning of the feature definition:

@regression   
Feature: User login   
@smoke   
 Scenario: Verify successful login   
 Scenario: Verify failed login with wrong password

**Running Tagged Scenarios:**  
When you run Cucumber, you can use the --tags option to specify which tags to include or exclude. Here are some examples:

**Run scenarios with a specific tag:**

cucumber --tags @smoke

**Exclude scenarios with a specific tag:**

cucumber --tags ~@regression

**Combining tags:**  
You can combine tags to include or exclude multiple tags:

cucumber --tags "@smoke or @regression" # Include either tag   
cucumber --tags "@smoke and not @regression" # Include @smoke but exclude @regression

**Using Tags with a Test Runner:**  
If you are using a specific test runner, such as JUnit in Java, you can specify the tags in the runner configuration. For example, in a JUnit test runner:

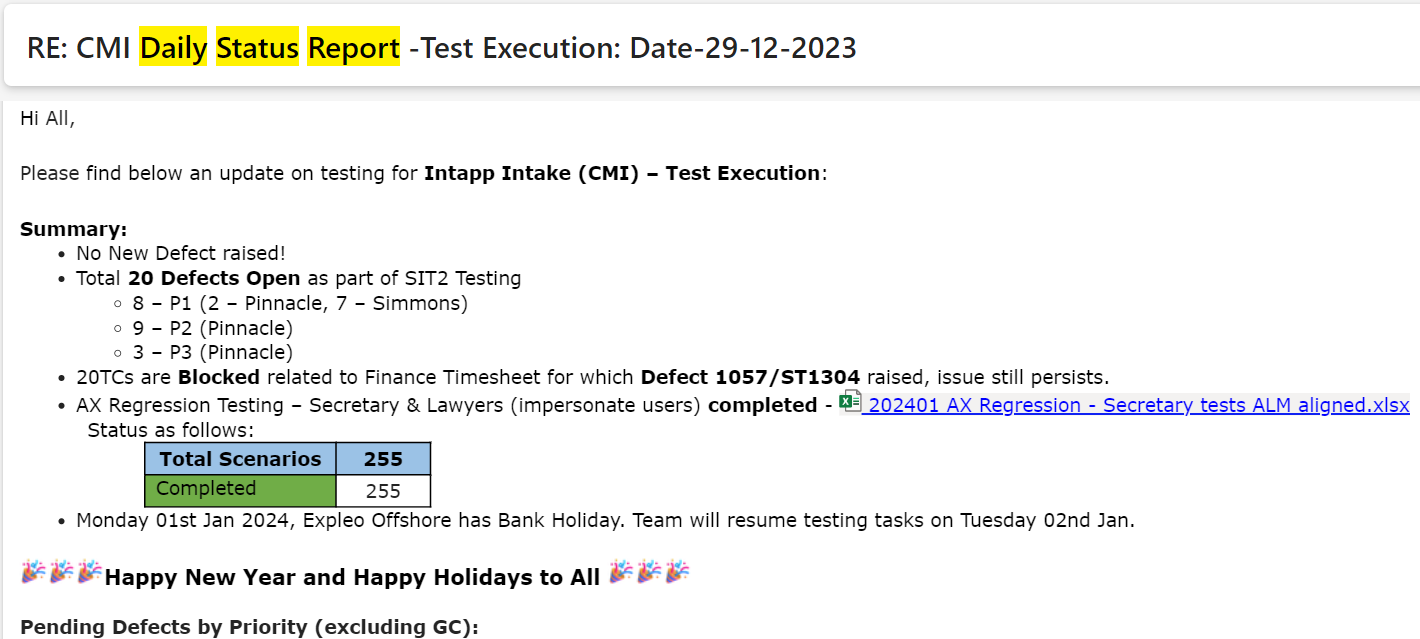
@CucumberOptions(   
 features = "src/test/resources/features",   
 glue = "stepdefinitions",   
 tags = "@smoke",   
 plugin = {"pretty", "html:target/cucumber-reports"}   
)   
public class RunCucumberTest {   
}

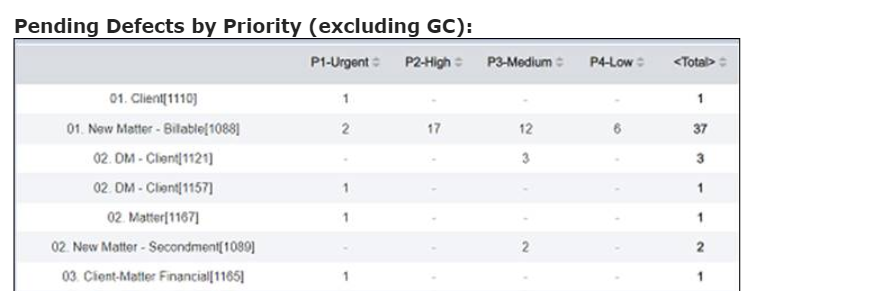
**5.How do you find unique or unused test data?**

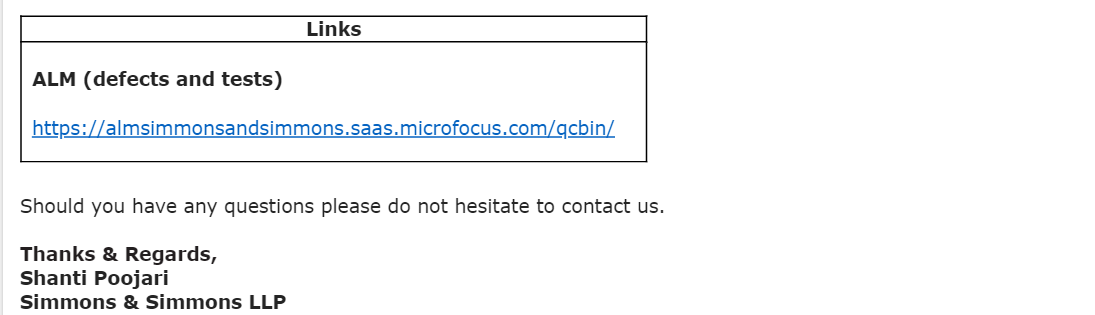
Preparing data for testing can be very time-consuming. Around 30-60% of a tester's time is spent searching, maintaining, and generating data for testing and development. Here are the possible reasons for this scenario:

* Testing teams do not have access and depend on others for the data they need, limiting the testing team's ability to identify and fix defects.
* While multiple teams work on the same project, each team uses its database and thus has different data sets. This creates conflicts, as the data set is only sometimes up-to-date when it comes time for another team to test the app.
* Searching a production database for information related to your application is like searching for a pin in a haystack. You need special cases to perform good tests, but you can only find them if you dig through dozens of terabytes.

6.**How do you handle date-related components, such as future dates?**  
 **7.Do you send daily status reports? To whom do you send them, and what format do you follow?**





Daily status reports, if used, are typically sent to:

* The Agile team (including developers, testers, and Scrum Master)
* The Product Owner
* Any relevant stakeholders who need updates

**8.What is the difference between a Daily Status Report (DSR) and a Weekly Status Report (WSR)?**

### **Key Differences**

1. **Frequency:** DSRs are daily; WSRs are weekly.
2. **Scope:** DSRs focus on daily activities; WSRs provide a broader view of weekly progress and issues.
3. **Detail Level:** DSRs are concise and immediate; WSRs are more detailed and strategic.
4. **Audience:** DSRs are usually for the immediate team; WSRs may be shared with a broader group, including higher management and external stakeholders.

**9.What Agile ceremonies do you follow in your project?**

--- >

1)Sprint Planning meeting –

- This meeting is near about 60 min

- Project Owner and Scrum master are host of this meeting

- Basic purpose of this meeting is to sort out user stories among all user stories for developing a module.

-Those user stories will be explained by Product owner

-Scrum master allocates those user stories on the basis of three factors like knowledge, efforts, complexity

**Daily Standup meeting -**

-Scrum master hosts this meeting

-Every team member attends this meeting

- Every team members answers three questions

\* What we did yesterday

\* What we are going to do today

\* Any obstacles or questions

**Sprint Review Meetings -**

-Scrum master hosts this meeting.It is also known as demo meeting.

-Every team member and stakeholder are invited to participate

-Sprint review meeting is to inspect the outcome of the sprint and determine future adaptation

-The scrum team present the results of their work to stakeholder

-They will discuss about progress towards the product goal.

**Sprint Retrospective meeting -**

* Scrum master hosts this meeting
* It will happen after sprint review meeting
* Evaluate how the last sprint, iteration, or work item went, specifically around the team dynamic, processes, and tools. Articulate and stack rank the items that went well, and those items that did not. Create and implement a plan for improving the way the team does work.

Example :

### **Evaluation of the Last Sprint**

#### **What Went Well**

1. **Team Collaboration:** The team worked well together, communicating effectively and helping each other out.
2. **Clear Goals:** The goals for the sprint were well-defined, making it easy for everyone to know what to focus on.
3. **On-time Delivery:** Most tasks were completed on time, which helped keep the project on track.

#### **What Didn't Go Well**

1. **Tool Issues:** There were problems with some of the tools we use, causing delays and confusion.
2. **Unclear Roles:** At times, it wasn't clear who was responsible for certain tasks, leading to some overlap and confusion.
3. **Missed Deadlines:** A few tasks took longer than expected, which pushed back the timeline.

### **Improvement Plan**

1. **Improve Tools:** Review and fix the issues with the tools, and provide training if needed.
2. **Clarify Roles:** Make sure everyone knows their specific responsibilities at the start of each sprint.
3. **Better Time Management:** Set more realistic deadlines and check in more frequently to ensure tasks are on track.

**10.What kinds of risks have you faced?**

* **Requirement Risks:**
* **Unclear Requirements:** Ambiguity or lack of clarity in requirements can lead to incorrect or incomplete testing.
* **Changing Requirements:** Frequent changes can disrupt testing plans and timelines.
* **Technical Risks:**
* **Integration Issues:** Problems integrating various system components can lead to delays and defects.
* **Tool Limitations:** Inadequate or malfunctioning testing tools can affect the efficiency and accuracy of testing.
* **Performance Issues:** Unexpected performance bottlenecks can be challenging to identify and resolve.
* **Project Risks:**
* **Tight Schedules:** Short timelines can lead to insufficient testing, increasing the likelihood of undetected bugs.
* **Resource Constraints:** Lack of skilled personnel, hardware, or other resources can limit testing effectiveness.
* **Dependency Delays:** Delays in receiving builds or other dependencies from other teams can push back testing schedules.
* **Operational Risks:**
* **Environment Issues:** Inconsistent or unstable testing environments can cause inaccurate test results.
* **Data Risks:** Lack of proper test data or data privacy issues can limit the scope of testing.
* **Communication Risks:**
* **Miscommunication:** Poor communication among team members, stakeholders, or other teams can lead to misunderstandings and mistakes.
* **Lack of Documentation:** Insufficient documentation can make it difficult to understand and test the software effectively.
* **Quality Risks:**
* **Undetected Defects:** The possibility of defects remaining undetected despite testing efforts.
* **Incomplete Testing:** Limited time or resources can lead to insufficient testing coverage.
* **Security Risks:**
* **Vulnerabilities:** Potential security vulnerabilities can be overlooked if not properly tested, leading to security breaches.

**11.How do you perform System Integration Testing (SIT), and what is your approach?**

**System Integration Testing (SIT)** is a phase in the software testing process where individual modules or components are combined and tested as a group to ensure they work together correctly. The primary goal is to identify issues related to the interactions between integrated components. Here's a straightforward approach to performing SIT:

### **Approach to System Integration Testing (SIT)**

1. **Planning and Preparation**
   * **Define the Scope:** Identify the components to be integrated and tested. This includes specifying the modules, interfaces, and interactions.
   * **Identify Test Scenarios:** Based on the requirements and design documents, create a list of test scenarios that cover all possible interactions between components.
   * **Set Up the Environment:** Prepare the test environment, ensuring that all necessary hardware, software, and tools are available and configured correctly.

* **Test Execution**
* **Develop Test Cases:** Write detailed test cases based on the identified scenarios, including input data, expected outputs, and step-by-step instructions.
* **Perform Testing:** Execute the test cases, focusing on verifying the data flow, interactions, and overall functionality of the integrated components.
* **Monitor and Record Results:** Log the results of each test case, noting any defects or issues encountered.
* **Defect Management**
* **Log Defects:** Document any defects found during testing, including a detailed description, steps to reproduce, and the severity.
* **Track and Fix:** Work with the development team to prioritize, track, and resolve defects. Retest the affected components once fixes are applied.
* **Verification and Validation**
* **Verify Interfaces:** Ensure that the interfaces between integrated components work as expected.
* **Data Integrity:** Check that data is correctly passed between modules and that there are no data corruption issues.

**12.If project timelines are slipping, what approach do you follow?**

When project timelines are slipping, it's important to address the issue promptly and strategically to minimize the impact. Here's a practical approach to manage slipping timelines:

### **1. Assess the Situation**

* **Identify the Root Cause:** Determine why the project is behind schedule. Common reasons include underestimated effort, scope creep, resource shortages, technical challenges, or dependencies on other teams.
* **Evaluate the Impact:** Understand the extent of the delay and how it affects the project's overall timeline and deliverables.

### **2. Prioritize and Adjust Scope**

* **Review and Prioritize Features:** Work with stakeholders to prioritize the most critical features and functionalities. Focus on delivering the most valuable aspects first.
* **Adjust Scope:** Consider reducing the scope of the project by deferring non-essential features to future iterations or phases. This helps in focusing on completing the core requirements.

### **3. Optimize Resource Allocation**

* **Reallocate Resources:** Reassign team members to the most critical tasks. Consider bringing in additional resources if possible.
* **Cross-Train Team Members:** If feasible, cross-train team members to handle multiple roles, increasing flexibility and efficiency.

### **4. Enhance Communication and Collaboration**

* **Increase Communication Frequency:** Hold more frequent meetings (e.g., daily stand-ups, status updates) to monitor progress closely and address issues quickly.
* **Improve Stakeholder Communication:** Keep stakeholders informed about the situation, changes in the plan, and any impact on deliverables.

### **5. Streamline Processes**

* **Optimize Workflows:** Identify and eliminate bottlenecks in the workflow. This may include improving testing processes, automating repetitive tasks, or refining code review practices.
* **Focus on High-Value Tasks:** Encourage the team to focus on tasks that provide the most value and can be completed within the adjusted timeline.

### **6. Increase Team Efficiency**

* **Implement Overtime Judiciously:** If necessary and agreed upon by the team, consider temporary overtime. However, be cautious of burnout and ensure it is a short-term solution.
* **Encourage Focus:** Minimize distractions and non-essential meetings to allow the team to concentrate on critical tasks.

### **7. Risk Management**

* **Identify and Mitigate Risks:** Continuously monitor for potential risks that could further delay the project. Develop contingency plans for high-risk areas.
* **Adapt Quickly:** Be prepared to adapt plans as new information becomes available or as circumstances change.

### **8. Re-evaluate Deadlines**

* **Re-assess Timelines:** After implementing changes, re-assess the project timelines. If necessary, negotiate new deadlines with stakeholders based on a realistic assessment of what can be achieved.
* **Document Changes:** Clearly document any changes to the project scope, timelines, and deliverables, and communicate these changes to all stakeholders.

### **9. Monitor and Review**

* **Continuous Monitoring:** Keep a close watch on progress and adjust the plan as needed. Use project management tools to track tasks, dependencies, and deadlines.
* **Post-Mortem Analysis:** After the project or sprint, conduct a retrospective to analyze what caused the delays and how they were managed. Use these insights to improve future project planning and execution.

### **10. Maintain Team Morale**

* **Support the Team:** Provide support and encouragement to the team. Recognize their hard work and efforts to get the project back on track.
* **Encourage a Positive Attitude:** Foster a positive and solution-oriented attitude to keep the team motivated and focused on finding solutions rather than dwelling on setbacks.