# Topic-wise assignments:

## Java and Java 8:

1 - Implement stack using queues.  
2 - Implement Oops concept for Car Class.  
3 - What is Object-Level Locking and Class-Level Locking  
4 - What are Java-8 related changes for Interfaces?  
5 - What is double-checked locking?  
6 - What are the disadvantages of normal singleton design pattern implementation and ways to overcome the same.  
7. - (Lambda expression usage): Adding values/data in application popup  
8. Java 8 features:  
Implement with Collections:  
a. forEach() method in Iterable interface (Implemented with Collections)  
b. Implementing multiple inheritance in Classes : Provide implementation logic in the class implementing the interfaces  
c. Add reference to static method using Functional interface {Using :: keywork)  
d. Performing filter operation through specified conditions, while booking a ticket on Cleartrip application, using the Stream API (java.util.stream)  
e. Display the current time (By converting datetime objects to strings) parse dates and time in 2 different time zones (java.time.zone).  
f. Implement Comparator interface and print the given strong in reverse order

## Selenium:

1 - What are the other ways to insert text into a text-box, if sendKeys is not allowed? For e.g.: Using JavascriptExecutor, Actions Class  
2 - How to wait and check for presence of an element on a page, every second(give code)? For E.g. FluentWait  
3 - Explain firefox driver internal architecture  
4 - How to drag and drop an element from one location to another? For E.g. Using Actions Class  
5 - "Write code to implement testng listeners? -- Use below:  
IAnnotationTransformer  
IAnnotationTransformer2  
IHookable  
IInvokedMethodListener  
IMethodInterceptor  
IReporter  
ISuiteListener  
ITestListener"  
6 - How to create a new fireFox profile and set different capabilities for the same?  
7 - How to take screenshots using selenium-java?  
8 - How to operate on windows based pop-up? Is it feasible via Selenium? For Eg: Using AutoIT, Robot Class etc  
9 - Is it possible to automate captcha and bar-code? Please explain?  
10 - "What are the various ways/alternatives in/using which, one can operate on drop-downs?  
For example, using Select class, Actions Class, Normal SendKeys, Using JavaScriptExecutor, Creating List of WebElements etc"  
11 - How to handle changing element locator values/dynamic locator values? For Example, using Preceding-sibling, Descendant, Ancestor etc  
12 - Creating the firefox profile by setting up the capabilities to set the download folder (with some properties to be set for the acceptance of security certificate)  
13 - Creating test cases by using/exhausting parallel, invocationCount, DataProviderThreadPoolCount attributes/annotations (Explore all the properties/annotations for <test and <suit tags)

## Spring:

1 - Write Spring-Java code to implement xml based & annotation based auto-wiring.  
2 - Write Spring-Java code to implement @Component annotation.  
3 - Write Spring-Java code to implement Constructor and Setter/Getter based dependency-injection.  
4 - Write Spring-Java code to showcase collection-injection.

## GIT:

1 - Git Installation & Configuration  
2 - Create local and remote GIT Repository -- You may use GitLab or similar for creating remote repository  
3 - Map your selenium project(Implement BDD & Create some sample test-cases) with your local GIT Repository  
4 - Perform GIT Operations as per this link -> <https://kb.epam.com/display/EPMCES/Git> (This includes the basic Git CI implementation as well)

## Sonar:

1. Configure SonarCloud as explained in : <https://about.sonarcloud.io/>  
2. Execute the Sonar job, analyse the Blocker, Critical and Major issues, and fix them.

## Maven:

1 - Convert your 'Assignment-Git' Project code into Maven Project and work on different phases (validate, compile, test etc).  
2 - Try integrating Maven & Git  
3 - Set System Properties/Code related properties using Maven  
4 - Implement transitive dependency, Include exclude files/directories

## Jenkins:

1 - Download Jenkins.war  
2 - Go through the folder where Jenkins.war downloaded.  
3 - Type "java -jar jenkins.war"  
4 - Please check logs in cmd  
5 - Type 'localhost:8080' in browser  
6 - Install required plugins  
7 - You will get default admin and password  
8 - Log in with admin  
9 - Install Maven related plugins  
10 - Click on New Item  
11 - Enter project name  
12 - Select freestyle project-Maven project  
13 - Enter description  
14 - 14)Select Source code management tool.  
15 - In Build Section,mention project pom.xml path  
16 - Set goals and options.  
17 - Install Testng Result plugin  
18 - Select Post-build Actions->Publish testng results  
19 - click on apply and save  
20 - Click on Build now  
21 - You are able to see results in Build details.  
22 - Now, Select Build Triggers->Build periodically check box.  
23 - Set Build Triggers->Build periodically  
24 - Set Values (15 13 \* \* \*)-schedule the build every day of every month of every year at the 15th minute of the 13th hour of the day, and track execution.  
25 - Now, execute some sample tests of your project code by integrating GIT, Maven and Jenkins. Please make Git related Jenkins configuration accordingly.  
26 - Now, perform parallel execution using selenium-grid, in combination with above mentioned steps.  
27 - Generate Serenity reports for your project

## API Automation:

1. Structure of a REST based request and response

<http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069>

 RESTful services should have following properties and features

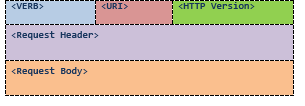
* Representations
* Messages
* URIs
* Uniform interface
* Stateless
* Links between resources
* Caching
* Representations

You can use any format for representing the resources, as REST does not put a restriction on the format of a representation. Depending on your requirement, you can decide to use JSON or XML.

* Messages

##### HTTP Request

An HTTP request has the format shown in Figure:

  
Figure 1: HTTP request format.

<VERB> is one of the HTTP methods like GET, PUT, POST, DELETE, OPTIONS, etc

<URI> is the URI of the resource on which the operation is going to be performed. It is combination of the “Endpoint URL” and “API URL”.

<HTTP Version> is the version of HTTP, generally "HTTP v1.1" .

<Request Header> contains the metadata as a collection of key-value pairs of headers and their values. These settings contain information about the message and its sender like client type, the formats client supports, format type of the message body, cache settings for the response, and a lot more information.

<Request Body> is the actual message content. In a RESTful service, that's where the representations of resources sit in a message.

There are no tags or markups to denote the beginning or end of a section in an HTML message.

Listing Three is a sample POST request message, which is supposed to insert a new resource Person.

URI:

REST requires each resource to have at least one URI. A RESTful service uses a directory hierarchy like human readable URIs to address its resources. The job of a URI is to identify a resource or a collection of resources. The actual operation is determined by an HTTP verb. The URI should not say anything about the operation or action. This enables us to call the same URI with different HTTP verbs to perform different operations.

##### Difference between PUT and POST

The short descriptions of these two methods I provided above are almost the same. These two methods confuse a lot of developers. So let's discuss these separately.

The key difference between PUT and POST is that PUT is idempotent while POST is not. No matter how many times you send a PUT request, the results will be same. POST is not an idempotent method. Making a POST multiple times may result in multiple resources getting created on the server.

Another difference is that, with PUT, you must always specify the complete URI of the resource. This implies that the client should be able to construct the URI of a resource even if it does not yet exist on the server. This is possible when it is the client's job to choose a unique name or ID for the resource, just like creating a user on the server requires the client to choose a user ID. If a client is not able to guess the complete URI of the resource, then you have no option but to use POST.

|  |  |
| --- | --- |
| **Request** | **Operation** |
| PUT http://MyService/Persons/ | Won't work. PUT requires a complete URI |
| PUT http://MyService/Persons/1 | Insert a new person with PersonID=1 if it does not already exist, or else update the existing resource |
|  |  |
| POST http://MyService/Persons/ | Insert a new person every time this request is made and generate a new PersonID. |
| POST http://MyService/Persons/1 | Update the existing person where PersonID=1 |

It is clear from the above table that a PUT request will not modify or create more than one resource no matter how many times it is fired (if the URI is same). There is no difference between PUT and POST if the resource already exists, both update the existing resource. The third request (POST http://MyService/Persons/) will create a resource each time it is fired. A lot of developers think that REST does not allow POST to be used for update operation; however, REST imposes no such restrictions.

### Statelessness

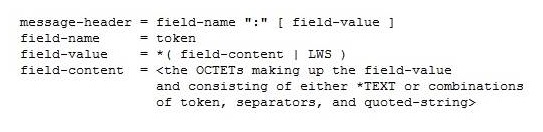
1. What are headers and how many different types can be there in headers.

<https://www.httpdebugger.com/http/http_header.html>

**HTTP headers** provide vital information required for a HTTP transaction send via [http protocol](https://www.httpdebugger.com/http/http_protocol.html).

The general HTTP header format contains colon-separated name - value pairs in the header field. Each of the name-value pair end with a carriage return (CR) and a line feed (LF) character sequence. Empty fields at the end of each header indicate the end of the header.

The common header format followed by applications looks like:



### **Types of HTTP headers**

There are four types of HTTP message headers. They are:

* General Header

General Header fields have **common applicability in request and response messages**. **The header fields apply only to the transmitted message and do not apply on the transferred entity.**

* Request Header

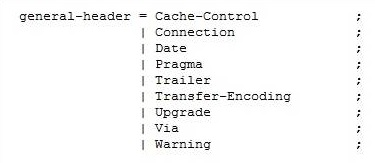
The request header field allows clients to additionally pass **request information and client information** to the server.

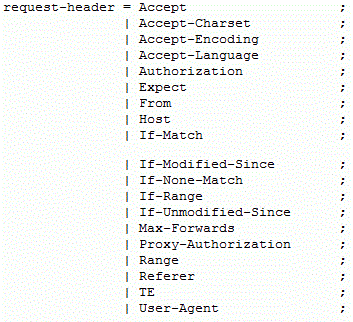
* Response Header

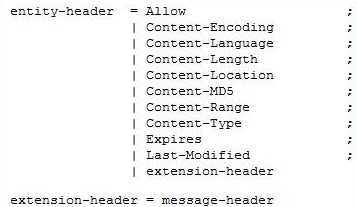
The response header field allows server to pass **additional information through the responses other than simple Status-Line response.**

* Entity Header

Entity header fields define **meta-information about the entity body or the requested resource.**







1. What’s Authorization? In how many ways authorization can be achieved?

<https://www.toolsqa.com/rest-assured/authentication-and-authorization-in-rest-webservices/>

Authorization is the process of giving access to someone. If you are Authorized then you have access to that resource. Now to Authorize you need to present credentials and as we discussed earlier that process is called Authentication. Hence Authorization and Authentication are closely related terms and often used interchangeably.

Before ending the tutorial let us see the contents of the private resource in the URL mentioned above. To do that enter the following credentials

***Username: ToolsQA***

***Password: TestPassword***

Server will be able to **Authenticate**and then **Authorize** you to access the private resource content. Below image shows the content after successful Authentication.



With this basic understanding of Authentication and Authorization, read the coming tutorials where we will discuss the specif types of **Authentication** models in **REST API**.

1. What message body, how many types it can be of

Request/Response body. It can be of XML/JSON/Plain Text.

1. RequestSpecification interface and its methods. In how many ways can we build RequestSpecification.

<http://static.javadoc.io/com.jayway.restassured/rest-assured/2.4.1/com/jayway/restassured/specification/RequestSpecification.html>

1. What’s a response, in how many ways can we get and manipulate the response?
2. What are Request and ResponseSpecBuilder?
3. How can we log the request and response? (there is a built in library in RestAssured)
4. How to use conditional logging?
5. How to store a variable so that it can be used in another tests?
6. What’s JSONPath? How can we invoke it? How to use it for validation?

<https://github.com/rest-assured/rest-assured/wiki/GettingStarted>

### JsonPath

Standalone JsonPath (included if you depend on the rest-assured artifact). Makes it easy to parse JSON documents. Note that this JsonPath implementation uses [Groovy's GPath](http://groovy-lang.org/processing-xml.html" \l "_gpath) syntax and is not to be confused with Kalle Stenflo's [JsonPath](https://github.com/json-path/JsonPath) implementation.

Maven:

<dependency>

<groupId>io.rest-assured</groupId>

<artifactId>json-path</artifactId>

<version>3.3.0</version>

</dependency>

1. What’s XMLPath? How can we invoke it? How to use it for Validation?

### XmlPath

Stand-alone XmlPath (included if you depend on the rest-assured artifact). Makes it easy to parse XML documents.

Maven:

<dependency>

<groupId>io.rest-assured</groupId>

<artifactId>xml-path</artifactId>

<version>3.3.0</version>

</dependency>

1. Can we validate the response without storing response in the Response object?
2. What can be the structure of REST API automation framework using Cucumber? Please explain with the flow diagram.
3. (Optional) How to use Postman for REST based testing
4. Demonstrate RestAssured usage for API automation (Create a new Test/Module on the same use case framework)
5. Use cucumber and excel to feed the input data and expected result
6. [POST]Post a tweet using twitter public API
7. [POST]Post a tweet using twitter public API with wrong credentials(negative)
8. [DELETE]Delete the same tweet
9. [GET]get last 10 published tweet and display those on console. If there are less than 10 tweets, display what’s available
10. API testing refresher
11. What is SOAP
12. What are constraints in SOAP?
13. What is REST, Difference between SOAP and REST
14. What’s request metadata
15. REST request and response structure

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# Use-Case:

Automation round trip multi-city flight reservation ticket, using the ClearTrip Application.

**Expected Output:**

\*Maven

1) Create a Maven Project

2) Add required dependencies -- Serenity, JBehave-Core/Cucumber, Junit, TestNG, Maven Plugins, Spring, Rest-Assured, Hemcrest etc

\*BDD

3) Create/Write Story/Feature for any 2 scenarios from the below list for ClearTrip Application (<https://www.cleartrip.com/>):

\*Java & Selenium (Java 8 features to be implemented)

3.a) Use collections for storing/retrieving data from excel/application. For example -- Various Dropdowns like Adults, Children, Infants etc

3.b) Use OOPs Concepts like Interfaces, Inheritance, Polymorphism wherever applicable.

3.c) Follow common-coding-lags sheet(shared already) wherever possible.

Scenarios:

a. Search Flights One-Way

b. Search Flights Round-Trip

c. Search Flights Multi-City

\*Test Designs using Excel

4) Create TestData Repository for the scenarios being automated. Use Excel for reading/writing data.

\*Java & Selenium

5) Implement page object model for ClearTrip Application. Try using preceding-sibling, descendants, ancestor etc within element locator values. Try using all if possible.

\*API

5.1) For rest Web-Services test-cases pertaining to above mentioned scenarios, please validate input & output data responses. For example, Status-Code checks, header-level validations, content validations etc.

\*Java & Selenium

6) Create Step-Definition classes.

\*Java & Selenium

7) Create workflow classes. Example -- Read Test-Data & Insert various element values within Clear-Trip app, on Search-Flights navigations etc.

\*Java & Selenium

8) Create Selenium utility classes. -- For example: Waits, Actions Class, Select Class, JavaScript Executor etc. Write code to handle Ajax Calls efficiently.

\*Spring - Autowiring

9) Use Spring container for wiring required configurations with respective classes (use @Autowired & @Component as well, wherever applicable). All POM Pages & Step Definition Classes should be pre-initialized before usage.

Sample: UserName Password, Application URL etc.

\*TestNG & script execution

10) Include Runner file into your pom.xml

\*Git

11) Create a fresh Local & Remote git repository, attach your project with the local git repository and push your entire project-code over git. You can use gitHub.

\*Jenkins

12) Install Jenkins and create new Maven Project. Provide required details. Select Source Code Management as GIT, provide your GIT repository URL. Add Git Credentials if required.

\*Git

13) Check for required installation & configuration related to Git, Maven & Jenkins for Jenkins. Do required Global Tools configurations settings(). Afterwards, Under Build Section, provide Maven Version and Goals and save the configuration.

\*Jenkins

14) Trigger Jenkins Job manually as well as using Cron-Jobs for build scheduling.

15) Validate Serenity Reports generation and execution status, along with analysis of failed tests.