

# All Questions



SDLC-STLC-AGILE

FUNCTIONAL  
TESTING

JAVA

SQL

TELL ME ABOUT  
YOURSELF/PROJECT  
QUESTIONS

BEHAVIORAL

API



# What is your Responsibility as an SDET?

- Turn manually executed test scenarios into automatically executed test scenarios via Automation Tools.
- Design and develop test plans that verify user stories and system requirements.
- Detect Defects in the application and document them.
- Involving yourself in the following test types:  
Functional Testing, Regression Testing, Smoke Testing

# Agile experience in your most recent project?

- Our sprint is 2 weeks and we have release every 3 sprints as a release cycle. We have 11 people in my team. 5 developers , 3 testers, also 1 SM, 1 BA and 1 PO.
- We start a sprint with Sprint Planning Meeting and we learn the part of the application which we are going to develop. We groom the planned user stories created by the PO.
- After sprint starts, we do Daily Standup Meeting everyday morning and we discuss what did we do yesterday, what will we do today and is there any blocker. It's a daily team sync up to keep us on the same page.
- End of the sprint, we do Sprint Demo/Review Meeting . It is just to show customer what we build throughout the sprint. PO provides feedback. As an SDET in my team, I have presented the features developed by the team. Client or stakeholders or business people can ask questions about the features developed.
- After Sprint Demo, we do Sprint Retrospective Meeting. In Sprint Retro, we talk we should start doing, stop doing and continue doing. We go over them and make sure that we don't make the same mistakes again.
- This sums up what happens in a typical sprint.

# STLC vs SDLC



- STLC is part of SDLC. It can be said that STLC is a subset of the SDLC set. The complete Verification and Validation of software is done in SDLC, while STLC only does Validation of the system.

## **SDLC (Software Development Life Cycle)**

1. Requirement gathering and analysis
2. Design
3. Coding
4. Testing
5. Deployment
6. Maintenance

## **STLC (Software Testing Life Cycle)**

1. Requirement / Design Analysis
2. Test Planning
  - Test Plan
  - Test Estimation
  - Test Schedule
3. Test Case Development (Designing)
  - Test Cases / Test Scripts / Test Data
  - Requirements Traceability Matrix
4. Test Environment Setup
5. Test Execution
  - Test Results (Incremental)
  - Defect Reports
6. Test Closure Activity (Reporting)
  - Test Results (Final)
  - Test Metrics
  - Test Closure Report

# WHERE are your requirement documents?



- Requirements convey the expectation of users for the software or product. In other words, all the expected functionalities out of the application are documented in terms of “Requirements”.
- Currently in my project my requirements are documented as Acceptance Criteria for each User Story.
- User Stories are created by the PO.
- Acceptance Criteria are created by the Business Analyst.
- All Acceptance Criteria will be documented in Confluence.

# Where is the requirement coming from?



- PO provide requirements for the application by Talking to the End-users that will be using this application the most.
- Talking to Partners/Sponsors.
- Talk to Domain Experts – coders and developers that have already build this application similar before or someone that is an expert the type of product being built.
- Industry Analysts and Information about competitors.

# Is the requirement is good or bad?



- Requirement must be (SMART Or INVEST)  
Specific, Measurable, Attainable, Realistic, Testable.

- Example:

Given I am an Authorized user with valid username and password  
When I login by entering my username and password on the login page  
Then I should be logged into the application in 2 seconds or less  
And I should should be able to login Measurable User should able to  
login very fast (in 2 second after clicking login button).

# Agile



- Agile is flexible methodology used for Software Development.
- You can change requirements and modify requirements anytime.
- It is dynamic, we are consistently gathering feedback and making adjustments to requirements. .
- Changes are always welcome.



# Agile Framework?



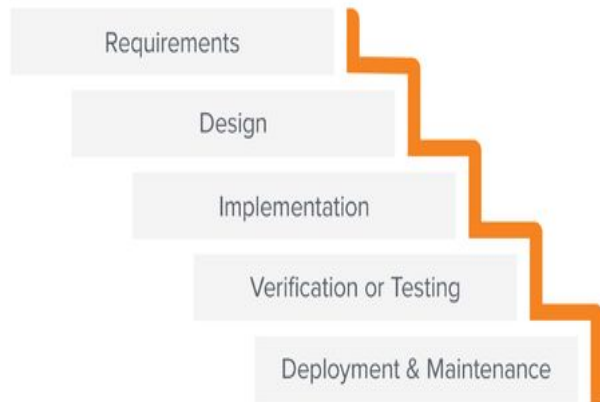
- **Role** : PO, SM, Team
- **Ceremonies** : Sprint Planning, Daily Scrum, Sprint Review, Sprint Retro, Grooming Session
- **Artifacts** : Product backlog, Sprint backlog, Burnout chart

# Waterfall?



- Waterfall methodology is the sequential method using for Software Development.
- You can not go back and have to finish the phase before you move on.
- 

The Waterfall Method



# Which Agile framework did you use in your previous projects ?



- I have heard Extreme programming(XP) , Kanban, Feature Driven Development and Scrum.
- But I have only worked with scrum only.

# Relation between Agile and Scrum? What is Scrum?



- Agile is the software development methodology that focuses on customer satisfaction by frequent software delivery.
- Scrum is one of the many approaches to implement Agile.
- Scrum is an Agile framework.
- Scrum is suitable for certain type of projects where there are rapidly changing requirements.
- In simple words, Agile is the methodology and scrum is the process/framework to follow this practice.

# Why do we need Agile? Waterfall and Agile?

- Because waterfall methodologies have following disadvantage;  
Requirement can not be change or hard to change once document is signed.
- In waterfall before completing the one phase you can't move to the next phase.  
For example, before coding phase is completed testing can not be started.
- Customer can't see what they are going to get until very late stage in development life cycle.  
It takes longer time to go to the production. By the time product goes to the market might be outdated already.
- Agile has following advantages:  
The change is welcomed. For example after the sprint demo if client does not like something we can take their feedback and improve the product.  
Requirement change is OK.

Since it is iterative development process, the development team can developed piece of functionality, get feedback and improve next iteration. So the product will be continuously improve.

# What are Different roles in Scrum?



- **Product owner** He is responsible to have a vision of what to build and convey his detailed vision to the team. He is the starting point of an agile scrum software development project. PO creates the user stories and prioritizes the product/sprint backlog.
- **Scrum team** is formed by the collective contribution of individuals who perform for the accomplishment of a particular project. The team is bound to work for the timely delivery of the requested product.
- **Scrum master** – Scrum master is the leader and the coach for the scrum team who tracks whether the scrum team is executing committed tasks properly. He is also responsible to increase the efficiency and productivity of the team so that they can achieve the sprint goal effectively.

# Describe your scrum team?



- Currently in my project we have 11 members in our scrum team.
  - . PO - Stephen.
  - Dev - Linda, Raj, John, James, Ryan, and Karan.
  - Test - Myself, Arjun, and Shelly.
  - SM - Roger.

# Have you heard of scrum of scrums?



- In case, there are multiple teams involved in the project, scrum of scrums is used to focus on projects that collaborate with multiple teams.
- It supports agile teams to collaborate and coordinate their work with other teams.
- It helps focus the meeting towards specific agenda items.



# Burn-down and burn-up charts



- To track the progress of an ongoing project, these charts are used.
- **Burn-up charts** indicate the work that has been completed while
- **Burn-down chart** shows the amount of remaining work in a project.

# Sprint?



- In Scrum, the project is divided into Sprints.
- Each Sprint has a specified timeline (2 weeks to 1 month).
- This timeline will be set for the duration of the project.
- Here, User Stories are split into different modules.
- The end result of every Sprint should be a potentially shippable product.

# How long is your Sprint?



- My current sprint duration is 2 weeks.

# Product backlog & Sprint Backlog?



- **Product backlog** is maintained by the project owner which contains every feature and requirement of the product.
- **Sprint backlog** can be treated as subset of product backlog which contains features and requirements related to that particular sprint only.

# Velocity of a sprint and how it is measured?

- **Velocity** is one of the planning tool used to estimate the speed of the work and time of completion of the project.
- The calculation of velocity is done by reviewing the work team has successfully completed during earlier sprints;
- for example, if the team completed 5 stories during a two-week sprint and each story was worth 3 story points, then the team's velocity is 15 story points per sprint.

# How often do you release?



- Our Release Cycle is dependent on the Roadmap created by the product owner.
- Typically we release features every 3 months.
- However when a critical defect has been reported in production we may do Hotfix releases which are emergency releases to resolve customer issues in our production environment.

## How What happens when requirements change in middle of sprint?



- As part of being in an Agile project we expect requirements to change.
- When a requirement changes in the middle of a sprint I will analyze the changes.
- Based on the changes I will determine the impact and how it affects the work that has been completed. (Does it add more scope or reduces scope ?)
- Based on the impact analysis we will need to re-estimate the user story to ensure the Level of Effort is updated.

# Software Testing?



- Process of executing a program or application with the intent of find software bugs using functional and automation tools
- Process of validating/verifying a software program/application



# Why we test?



- To build bug free application.
- To satisfied end user and client.
- To build great product to generate more revenue.

# Is 100% testing possible?



- We can't test the application 100% since there are unlimited scenarios that we can't even imagine.
- Software testing is risk based activity based on priority of the functionality we can test as much as much as possible.
- Even though 100% testing is not possible but I believe 100% customer satisfaction is certainly possible.

# Test Plan VS Test Strategy?



- A **Test Plan** is a formal document derived from requirement documents, describing in detail the scope of testing and the different activities performed in testing. Whereas, a **test strategy** is a high-level document describing the way testing will be carried out in an organization.

#	Test Plan	Test Strategy
1	A test plan is derived from Software Requirement Specification (SRS), describing in detail the scope of testing and the different activities performed in testing.	A test strategy is a high-level document describing the way testing is carried out.
2	A test plan is specific to a particular project.	A test strategy is normally for a complete organization. Although it can be specified for a particular project as well.
3	It describes the whole testing activities in detail - the techniques used, schedule, resources etc.	It describes the high-level test design techniques to be used, environment specifications etc.
4	It is prepared by test lead or test manager.	It is generally prepared by the project manager.

# Test Plan?



- Test plan is a word document that described the testing scope
  - High level test cycle
  - Defect life cycle
  - Entrance Criteria (defines what all need to start the testing)
  - Exit Criteria (defines what the testing is finished)
- If you don't know where to start and where to finish then your goals are not clear.

# Test Case?



- Test case describes the functionality and test steps.
  - Test Case ID
  - Step number
  - Description of the functionality
  - Expected result
  - Actual Result

# When does testing start?



- Testing starts from detailed review of the requirements.
- We have to make sure the requirement is correct in first place.
- With the wrong requirement it is impossible to build bug free application.

# Defect Life Cycle?



- New - When the defect has been newly created.
- Assigned - When the defect has been accepted and assigned to a developer.
- Rejected - When the bug is rejected due to its invalidity.
- Deferred - When the bug is prioritized for a later sprint.
- Open - The defect is awaiting assignment.
- Fixed - The defect has been fixed.
- Retested - The defect has been retested.
- Closed - The defect has been closed.

# Epic, User stories & Tasks?



- **User Stories:** User Stories defines the actual business requirement. Generally created by Product Owner.
- **Task:** To accomplish the business requirements development/testing teams create tasks.
- **Epic:** A group of related user stories is called an Epic.



# What is testing hierarchy?



- **Unit testing** – Developers test each module or block of code during development.
- **Component Testing** – Component is a standalone functionality that can work by itself. Ex. Amazon Buyer Functionality, Seller Functionality, Prime Video Functionality.
- **Integration Testing** – Combine all of the Functionalities. When I integrate them, can I still use all of the functions? Make sure they all still work.
- **System Testing** – EndtoEnd testing. Test everything from beginning to end.
- **Acceptance Testing** – Hire a UAT (User Acceptance Testing) Team or Business Analyst can also do Acceptance Testing.  
After testing has been complete you have to get another team to do acceptance testing so they can confirm the QA teams testing was successful and have the product ready for the customer.

# Requirement Traceability Matrix (RTM)



- RTM is used to make sure that all test cases cover the requirement or not. It is like excel sheet. We can also say that RTM is an document which shows all the acceptance criteria and the user stories are covered by the Test Cases (Scenario)

# Defect



- When expected result does not match actual result it is a defect.

# What to do when you find a defect?



- If I find a defect, before report it I re-produce the bug that I need to make sure that is a valid defect.
- If it is a small issue, I will go to the developer desk, and he can fix it right away.
- If it is a big issue, then I open my JIRA and log the defect.
- If I am not sure it is bug or not I will talk to SME (subject matter expert it means the person who knows the application better than anyone).

# If Developer says not a defect, what to do?



- I always make sure that it is a real defect that's why I reproduce it.
- I take a screenshots and give all the steps to reproduce the defect
- Actually one of my biggest challenges that I faced in my current project is that...

# When will you Automate?



- If it is taking a lot of manual effort. I run at least once manually and after that I automate it. Automation is good for most repetitive functionality.
  - If the test cases are high priority test cases.
  - If the functionality is critical functionality.
  - If the test cases are too long and too difficult to execute. The regression test cases based on the priority.
  - We should automate as much as possible.

# When will you not Automate?



- -If functionality keeps changing
  - If functionality is used only once during the entire project
  - Ad-Hoc test can not be automated.
- Captcha can not be automated

# Debugging vs testing?



- The main difference between debugging and testing is that debugging is typically conducted by a developer who also fixes errors during the debugging phase. Testing on the other hand, finds errors rather than fixes them. When a tester finds a bug, they usually report it so that a developer can fix it.



# Peer review?



- Peer reviews are reviews conducted among people that work on the same team. For example, a test case that was written by one QA engineer may be reviewed by a developer and/or another QA engineer.

# Who writes test plans and test cases?



- Test plans are typically written by the quality assurance lead while testers usually write test cases.

# Bug severity vs bug priority



- **Bug Severity** refers to the level of impact that the bug has on the application or system while **Bug Priority** refers to the level of urgency in the need for a fix.
- Usually the **severity** is defined in terms of financial loss, damage to environment, company's reputation and loss of life.
- **Priority** of a defect is related to how quickly a bug should be fixed and deployed to live servers.

# System testing and integration testing?



- For system testing, the entire system as a whole is checked,
- whereas for integration testing, the interaction between the individual modules are tested.

# Black box vs white box testing?



- **White Box Testing** has many names such as Glass Box, Clear Box, or Structural Testing. It requires the testers to gain code-level perspective, design cases to exploit code and find potential bugs.
- **Black Box Testing** is a standard software testing approach which requires testers to assess the functionality of the software as per the business requirements. They treat the software as a black box and validate as per the end user point of view. It applies to all levels of software testing such as Unit, Integration, System or Acceptance Testing.

# Front End Testing and Back End testing?



- Front End Testing is performed on the Graphical User Interface (GUI), whereas Back End Testing involves databases testing. Front end consist of web site look where user can interact whereas in case of back end it is the database which is required to store the data.
- When user enters data in GUI of the front end application, then this entered data is stored in the database. To save this data into the database we write SQL queries.

# Functional testing & non-functional testing



- **Functional testing** is a type of testing which verifies that each function of the software application operates in conformance with the requirement specification.
  - -Unit Testing
  - -System Testing
  - -Smoke Testing
  - -User Acceptance Testing
  - -Integration Testing
  - -Regression Testing
- **Non-functional testing** is a type of testing to check non-functional aspects (performance, usability, reliability, etc.) of a software application.
  - -Performance Testing
  - -Stress Testing
  - -Load Testing
  - -Security Testing

# What do you do when a production defect is reported?



- In my current company we have a separate JIRA board to monitor production defects.
- All production defects reported are screened by the product owner where they provide a priority on the ticket.
- Once the priority is determined, I will begin analyzing the issue reported.
- I will need to reproduce the defect in an environment that mirrors production like staging.
- During my analysis I collaborated with my development team to review their code or clarify questions about functionality.
- Once the analysis is complete it may require a hotfix deployment which is an emergency targeted release to resolve the defect for the user.



# Can you describe the testing process in your company ?



- In my current company we practice Agile Scrum.
- Prior to the start of the sprint, we review all the user stories and estimate it based on testing complexity.
- During Sprint planning we finalize the user stories and our test lead assigns the testing responsibility between our team mates.
- When I am ready to test my user story here are the steps I will follow:

Requirement analysis

Test planning

Test case development

Test environment setup

Test execution

Test cycle closure



# What information do you include in a bug report?



- Here are the following elements I will be sure to include in my defect report:
  1. Title.
  2. Steps to reproduce.
  3. Expected Result.
  4. Actual Result.
  5. Evidence.
  6. Severity.
  7. Environment Details.
  8. Link to User Story and Test Case.

# What is the difference between bug release and bug leakage?



**Bug leakage:** When a bug is missed by the testing team and found by a customer on the user end, that is considered bug leakage.

**Bug Release:** Application releases that are intended to resolve production defects reported by users.

# What is the difference between smoke testing and sanity testing?



**Smoke:** A daily process where we test the core functionalities of the application to ensure its stability before we begin testing it for the day. The Smoke is fully automated and run at 7AM every day. If we our smoke test fails we will report this to the development team for further analysis and hold off on testing the application further until this is resolved.

**Sanity:** When we focus our testing on end to end for a specific set of modules in our application. Sanity allows us to isolate specific modules to test if we need to verify that there no outstanding defects with the module.

## How do you approach a situation where requirements are not clearly defined?



- When requirements are not clear here are some strategies I will do:
  1. Set up a meeting with the business analyst to clarify the requirements.
  2. Asking the product owner for some insights on the business case for the feature.
  3. Reviewing similar application requirements.
  4. Performing Ad Hoc testing to understand the behavior of the functionality.

# Imagine you have a tight deadline, and the application has critical issues. What steps would you take?



- Managing deadlines and having adequate time management skills are important as a tester.
- First I would determine the priority of my tasks, and focus on the highest priority items.
- I would next look which modules have the critical issues, and would sort the issues by riskiest module.
- If I am unable to deliver my tasks on time, I would proactively inform the scrum master to add additional resources or work overtime to deliver the tasks on time.
- I will my team updated by providing them a daily status in my scrum stand up meeting.

# What are different Test Techniques you are aware of?



Here are the Testing Techniques that I am familiar with:

1. Equivalence Class Partitioning.
2. Boundary Value Analysis.
3. State Transition.
4. Error Guessing.
5. Use Case Testing.
6. Decision Table Testing.

# How do you manage your test cases?



Currently we using JIRA as a agile project management tool, we integrated JIRA with X-RAY which the test management tool we are using.

With X-Ray we can do the following:

- Create Test Cases.
- Group Test Cases.
- Execute and document test results.
- Link test cases to user stories.
- Link Defects to user stories and test cases.
- Create customized analytics and reporting.



# What is JAVA?



Java is a multiplatform, object-oriented programming language. Currently I am using JAVA to automated functional tests to create a efficiency when it comes to the testing process.

# What are the JAVA components?



JVM: Java Virtual machine(JVM) is an abstract machine. It doesn't physically exist. Whatever Java program we want to run, goes into JVM. And JVM is responsible for loading, verifying and executing the java program. JVM is responsible for converting the byte code to the machine-specific code.

JRE: JRE stands for “Java Runtime Environment”. It physically exists. The Java Runtime Environment provides the minimum requirements such as libraries and Class Loader for executing a Java application on JVM. It consists of the Java Virtual Machine (JVM), core classes, and supporting files. *JRE = JVM + Library Classes*

JDK: The Java Development Kit (JDK) is a software development environment used for developing Java applications.  
*JDK = JRE*

+ *Development tools*. It includes:

- Java Virtual Machine,
- Java Runtime Environment,
- Loader,
- Java compiler,
- Documentation generator
- Archiver (jar),
- Other tools needed in Java development.

# Can you explain the main method in JAVA?



The main() is the starting point for JVM to start execution of a Java program. During the run time of the class JVM will specifically look for the main method to execute the program.

# Can you explain the syntax of the main method in JAVA?



-public:

public is an access modifier which is used to specify who can access this method. Public means that this method will be accessible by any Class.

-static:

It is a keyword in java which identifies that it belongs to the class its in.

-void:

it is the return type of the method void means no value will be returned.

-main:

it is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main executions occurs.

-String[] args:

it is the parameter passed to the main method.



# Local, Instance and Static variables

- **Local variable** is typically used inside a method, constructor, or a **block** and has only local scope. The best benefit of having a local variable is that other methods in the class won't be even aware of that variable.
- **Instance variable** is a variable which is declared within a **class**, but outside a method. Instance Variable belongs to the OBJECT. Can be called by object name. Every object of that class will create it's own copy of the variable while using it. Thus, any changes made to the variable won't reflect in any other instances of that class and will be bound to that particular instance only.  
  
Numbers default to 0, Objects default to null, Booleans default to false
- **Static (class) variable** belongs to the CLASS, can be called through class name. Static variables are

declared with the static keyword in a class, but outside a method. Static variables are also called shared variable, it will have only one copy of it. Every object of the class will share the value of it.

```
public class VariableExample{
    int myVariable;//instance variable
    static int data = 30; //static variable
    public static void main(String args[]){ //main method
        int a = 100; //local variable
        VariableExample obj = new VariableExample(); //declare object
        System.out.println("Value of instance variable myVariable: "+obj.myVariable); //instance variables can be called by object name
        System.out.println("Value of static variable data: "+VariableExample.data); //instance variables can be called by className
        System.out.println("Value of local variable a: "+a);
    }
}
```

## Output:

- Value of instance variable myVariable: 0
- Value of static variable data: 30
- Value of local variable a: 100

# What is the String class in JAVA?



In Java, a `String` is an object that represents a sequence of characters.

Strings in Java are immutable, which means that once a `String` object is created, its value cannot be changed. Any modification to a `String` results in the creation of a new `String` object.

The String class provides methods that can be used to manipulate and work with strings:

List of String methods:

[https://www.w3schools.com/java/java\\_ref\\_string.asp](https://www.w3schools.com/java/java_ref_string.asp)

# What is the String pool?



The string pool, also known as the interned string pool or string intern pool, is a special memory area in Java where string literals are stored.

This pool helps in optimizing memory usage and improving the performance of string operations.

When JAVA String literals are created, the String pool will be checked if the string value already exists, if the string value does not exist then a new string value will be inserted into String pool.

Advantages of the String pool:

- Memory Management.
- Performance.
- Immutability.

# What is the difference between '==' or .equals() for Strings?



The `==` is a comparison operator. It compares one value to another. The comparison operator is not ideal for Strings since Strings can be in literal or object form.

The `==` operator checks if two references point to the same object in memory. Literal and String objects have different references point and therefore even if a literal/object had the same value it will return false.

The `equals()` method of the `String` class compares the content of the strings. The `equals()` method should be used when comparing content of strings.



# What is the difference between String, StringBuffer, and StringBuilder?



## **String**

- **Immutability:** Strings are immutable, meaning their values cannot be changed once created.
- **Thread Safety:** Not thread-safe.
- **Performance:** Suitable for scenarios where the string value doesn't change frequently.

- **StringBuilder**

- **Mutability:** Mutable, allows modification of the string content without creating new objects.
- **Thread Safety:** Not thread-safe.
- **Performance:** Faster than `StringBuffer` due to the lack of synchronization. Suitable for single-threaded environments.

## **StringBuffer**

- **Mutability:** Mutable, allows modification of the string content without creating new objects.
- **Thread Safety:** Thread-safe, with synchronized methods to ensure safe use in multi-threaded environments.
- **Performance:** Slower than `StringBuilder` due to synchronization overhead. Suitable for multi-threaded environments.

# What are JAVA Loops?



Loops in Java are control flow statements that allow code to be executed repeatedly based on a condition. They are used to perform repetitive tasks efficiently.

In JAVA here are the following loops available:

1. for loop.
2. for each loop. (enhanced for loop)
3. while loop.
4. do while loop.

**For Loop:** Ideal for iterating a specific number of times.

**While Loop:** Ideal when the number of iterations depends on a condition.

**Do-While Loop:** Ensures at least one iteration before checking the condition.

**Enhanced For Loop:** Simplifies iteration over arrays and collections.

**Break Statement:** Exits the loop immediately.

**Continue Statement:** Skips the current iteration and continues with the next one.

# What are JAVA Operators?



## 1. Arithmetic operators

+, -, \*, /, %

## 2. Assignment

=, +=, -=, \*=, /=, %=

## 3. Comparison

- Compare 2 numeric values
- Always return boolean true or false.
- 

>, <, >=, <=, ==, !=

## 4. Logical operators

- Compare two boolean values
  - Always return boolean true or false
- &&, ||

## 5. Unary Operators

- Only one operand is needed

++, --,

- ++ will increment the numeric variable value by 1
- -- will decrement the numeric variable value by 1
- post decrement: variableName--
- Post increment: variableName++
- Pre decrement: -- variableName
- Pre increment: ++ variableName

# What is an if statement in JAVA?



An `if` statement in Java is a control flow statement that allows you to execute a block of code only if a specified condition is true. It is used to perform conditional operations in a program.

`if` statements are fundamental in Java for making decisions based on conditions.

They help control the flow of execution in a program by selectively executing blocks of code.

`if-else` statements provide an alternative execution path when the main condition evaluates to false.

`if-else-if` ladders are used for sequential checks of multiple conditions.

Nested `if` statements allow for more intricate conditional logic by nesting one `if` statement inside another.

# What is a JAVA Constructor?



**Purpose:** Constructors are used to initialize the state (instance variables) of an object when it is created.

**Name:** Constructors have the same name as the class they belong to.

**No Return Type:** Constructors do not have a return type, not even `void`.

**Invocation:** Constructors are automatically invoked when an object of the class is created using the `new` keyword.

**Initialization:** They are used to initialize the instance variables of an object to their initial or default values.

**Overloading:** Java allows constructor overloading, meaning a class can have multiple constructors with different parameter lists.

**Default Constructor:** If no constructor is defined in a class, Java provides a default constructor (no-argument constructor) that initializes the object with default values.

**Purpose:** Constructors are essential for setting up the initial state of objects, ensuring they are in a valid and usable state upon creation.

# What are types of JAVA Constructor?



## Default Constructor

- **Definition:** A constructor with no parameters.
- **Usage:** Automatically provided by Java if no other constructors are defined in the class.
- **Purpose:** Initializes object variables to default values (e.g., `0`, `null`, `false`).

## Parameterized Constructor

- **Definition:** A constructor with parameters.
- **Usage:** Accepts arguments to initialize instance variables with specific values during object creation.
- **Purpose:** Provides flexibility to initialize objects with different initial states.

## No-Argument Constructor

- **Definition:**
  - A constructor with no parameters.
  - It does not accept any arguments during object creation.
- **Automatic Generation:**
  - If no constructors are explicitly defined in a class, Java automatically provides a default constructor.
  - This default constructor initializes the object with default values (e.g., `null` for reference types, `0` for numeric types, `false` for boolean).

# Can you have multiple constructors in a class?



Yes you can multiple constructors within a class.

For example you can have:

1. No Arg Constructor.
2. Parametrized constructor that accepts 2 integer parameters.
3. Parametrized constructor that accepts 2 string parameters.
4. Parametrized constructor that accepts 1 string and 1 int parameters.

This is known as 'Method Overloading' where the method name(constructor) is the same but the parameters are different.

# What is the meaning of the static keyword?



In Java, the `static` keyword is used to define members (variables and methods) that belong to the class itself rather than instances (objects) of the class. Here are the key points:

- 1. Static Variables (Class Variables):**
  - Static variables are shared among all instances (objects) of a class.
  - There is only one copy of a static variable that is maintained in memory, regardless of how many instances of the class are created.
  - They are typically used for constants (`final static`) or variables that need to maintain their value across all instances of the class.
- 2. Static Methods:**
  - Static methods belong to the class rather than any specific instance.
  - They can be called using the class name directly, without needing to instantiate an object of the class.
  - Static methods cannot directly access instance variables or instance methods of the class (unless through an object reference).
- 3. Static Blocks:**
  - Static blocks are used for static initialization of a class.
  - They are executed only once when the class is loaded into memory by the Java Virtual Machine (JVM).
  - Static blocks are useful for initializing static variables or performing one-time initialization tasks for the class.

## **Purpose and Usage of `static`:**

- **Memory Efficiency:** Static variables are allocated memory once per class, conserving memory compared to instance variables that are allocated memory for each object.
- **Global Access:** Static methods and variables can be accessed directly using the class name, making them accessible globally within the package or program.
- **Utility Methods:** Static methods are commonly used for defining utility methods that perform common tasks and do not require access to instance-specific data.
- **Class Initialization:** Static blocks ensure that necessary initialization tasks are performed when the class is first loaded, such as setting up static variables or establishing connections.



# What is the difference between a block, static block, and method?



*Block*: is a block of code surrounded by { }

Which will be 1st block of code executed when object is created

*Static Block*: is a block code starts with static keyword and follow by  
{ } - static{ }

Which will be the first block of code executed when the class is referenced

*Method*: is a block of code that has a name, return type and access modifiers, method will only be executed when called

# What is an Array?



An **array** is a data structure that stores a fixed-size sequential collection of elements of the same type. Arrays are used to store multiple values of the same data type under a single variable name.

## **Key Points about Arrays:**

1. **Fixed Size:**
  - Arrays in Java have a fixed size, meaning once they are created, their size cannot be changed.
2. **Element Type:**
  - All elements in an array must be of the same data type (e.g., `int`, `double`, `String`).
3. **Zero-based Indexing:**
  - Elements in an array are accessed via an index, starting from `0` up to `length - 1`, where `length` is the size of the array.
4. **Declaration and Initialization:**
  - Arrays are declared using square brackets `[]` after the data type, either implicitly or explicitly specifying the size during initialization.
5. **Accessing Elements:**
  - Individual elements of an array are accessed using the index within square brackets, e.g., `array[index]`.
6. **Length Property:**
  - Arrays have a `length` property that specifies the number of elements in the array.

# What is the difference between a class and object?



## Class

- **Definition:**
  - A class is a blueprint or template for creating objects.
  - It defines the properties (attributes) and behaviors (methods) that objects of the class will have.
- **Usage:**
  - Classes are used to encapsulate data (attributes) and functionality (methods) into a single unit.
  - They serve as a template from which multiple objects can be created.

## Object

- **Definition:**
  - An object is an instance of a class.
  - It is a runtime entity that occupies memory and has its own state (attributes) and behavior (methods) defined by its class.
- **Creation:**
  - Objects are created using the `new` keyword followed by a constructor of the class.
  - Each object created from the same class has its own set of instance variables, independent of other objects of the same class.

# What is object oriented programming?



Object-oriented programming provides a structured approach to software development by focusing on objects, their interactions, and their ability to model real-world entities and relationships. It enhances code organization, promotes reuse, and supports scalable and maintainable software solutions.

# What are the pillars of Object Oriented Programming in JAVA?



The pillars of Object-Oriented Programming (OOP) in Java refer to the fundamental principles that guide the design and implementation of object-oriented systems. These pillars, also known as the four main concepts of OOP, are:

## 1. **Encapsulation:**

- **Definition:** Encapsulation refers to the bundling of data (attributes) and methods (behaviors) that operate on the data into a single unit (object).
- **Purpose:** It hides the internal state of an object and restricts access to certain components, promoting data integrity and security. Encapsulation also allows objects to present a clean and consistent interface to the outside world.

## 2. **Inheritance:**

- **Definition:** Inheritance is the mechanism by which one class (subclass or derived class) can inherit properties and behaviors from another class (superclass or base class).
- **Purpose:** It promotes code reuse and allows hierarchical classification of classes. Subclasses can extend the functionality of their superclass by adding new methods or overriding existing ones while inheriting common attributes and methods.

## 3. **Polymorphism:**

- **Definition:** Polymorphism allows objects of different classes to be treated as objects of a common superclass through a shared interface.
- **Purpose:** It enables flexibility and extensibility in programming by allowing methods to be called on objects of different types. Polymorphism includes method overriding (where a subclass provides a specific implementation of a method defined in its superclass) and method overloading (where multiple methods have the same name but different parameter lists).

## 4. **Abstraction:**

- **Definition:** Abstraction refers to the process of hiding complex implementation details and showing only the essential features of an object.
- **Purpose:** It simplifies complex systems by focusing on the relevant aspects while hiding unnecessary details. Abstraction is achieved through abstract classes and interfaces in Java, which define methods without implementation, leaving it to subclasses to provide specific implementations.

# What is encapsulation ?



Encapsulation is when data is hidden from external classes.

We use the private access modifiers for variables to restrict access.

To allow selective access to data we can create public setter and getter methods to set and retrieve values.

# What is the difference between Method overloading and overriding ?



Method overloading also known as Static polymorphism is when a class contains several methods with the same name but different method parameters.

Method overriding also known as Dynamic Polymorphism is when a child class extends a parent class and creates its own implementation of the parent class method.

```
1 package InterviewDemo;
2
3 public class Overloading {
4     //default constructor
5     public Overloading() {
6         System.out.println("default");
7     }
8     // another constructor
9     public Overloading(String name) {
10        System.out.println("Constructor with 1 param");
11    }
12
13    public void print() {
14        System.out.println("This is print method");
15    }
16
17    public int print(int num) {
18        System.out.println("This is print method");
19        return num;
20    }
21    // this will give error
22    // public int print() {
23    //     System.out.println("This is print method");
24    //     return 10;
25    // }
26    //
27
28
29    public void print(String msg) {
30        System.out.println("This is print method: " + msg);
31    }
32
33
34 }
```

```
ProgrammingLanguage.java x
1 package InterviewDemo;
2
3 public class ProgrammingLanguage {
4
5
6     public void printName() {
7         System.out.println("C++");
8     }
9 }
10 |

Demo.java x
1 package InterviewDemo;
2
3 public class Demo {
4
5     public static void main(String[] args) {
6         ProgrammingLanguage java = new Java();
7         java.printName();
8     }
9 }
10
11
12
13 }
14
```

## Can you overload the main method?



Yes we can overload the main method. However JVM will specifically look for the expected main method signature (String[] args) during program execution.



## Can you have multiple main methods in a class?



Yes we can overload the main method. However JVM will specifically look for the expected main method signature (String[] args) during program execution.

## Can you have multiple main methods in a class?



Yes we can overload the main method. However JVM will specifically look for the expected main method signature (String[] args) during program execution.

# What is an abstract class?



An abstract class in Java:

- Cannot be instantiated directly.
- Is declared with the `abstract` keyword.
- Can have both abstract methods (without a body) and concrete methods (with a body).
- Must be subclassed, with subclasses providing implementations for the abstract methods.
- Provides a way to define a common base with shared behavior and enforced structure.

# What is an interface?



An interface in Java:

- Is a reference type, similar to a class, that can contain only abstract methods (until Java 8) and static final variables.
- Starting from Java 8, can also have default methods (with a body) and static methods.
- Is used to specify a set of methods that a class must implement.
- Supports multiple inheritance, allowing a class to implement multiple interfaces.
- Provides a way to achieve abstraction and define a contract for what a class can do, without dictating how it should be done.

# What is the difference between Interface and Abstract class?



The key differences between an abstract class and an interface in Java are:

1. **Instantiation:**
  - **Abstract Class:** Cannot be instantiated directly. It needs to be subclassed.
  - **Interface:** Cannot be instantiated directly. A class needs to implement the interface.
2. **Methods:**
  - **Abstract Class:** Can have both abstract methods (without a body) and concrete methods (with a body).
  - **Interface:** Until Java 8, could have only abstract methods. From Java 8 onwards, can also have default methods (with a body) and static methods.
3. **Variables:**
  - **Abstract Class:** Can have instance variables (non-static fields).
  - **Interface:** Can only have static final variables (constants).
4. **Inheritance:**
  - **Abstract Class:** Supports single inheritance (a class can inherit from only one abstract class).
  - **Interface:** Supports multiple inheritance (a class can implement multiple interfaces).
5. **Use Case:**
  - **Abstract Class:** Used when classes share a common base and some common implementation. Suitable for providing a common base class with some default behavior.
  - **Interface:** Used to define a contract for what a class can do. Suitable for completely abstracting the behavior and allowing multiple inheritance.
6. **Access Modifiers:**
  - **Abstract Class:** Can have any access modifiers for methods and fields.
  - **Interface:** Methods are implicitly `public`, and fields are implicitly `public static final`.
7. **Constructor:**
  - **Abstract Class:** Can have constructors.
  - **Interface:** Cannot have constructors.

In summary, use an abstract class when you want to share code among several closely related classes, and use an interface when you want to define a contract that can be implemented by any class, regardless of its position in the class hierarchy.

Let's say you have *class A*, how can you use all methods from *class A* inside *class B*?



We can use inheritance. Class B will extend Class A. Through inheritance Class A will inherit all the methods/data from Class B.

# What is Inheritance?



Inheritance in Java is a mechanism where one class (the subclass) inherits the properties and behaviors of another class (the superclass). It allows for code reuse and the creation of a hierarchical class structure. Java supports single inheritance, where a class can inherit from only one superclass. Key concepts include method overriding, where a subclass provides a specific implementation of a superclass method, and the use of the `super` keyword to access superclass methods and constructors. Inheritance represents an IS-A relationship, making it fundamental for polymorphism and code organization.

# What are the different types of Inheritance available in JAVA?



## Single Inheritance:

- A class inherits from one superclass.
- Example: `class Dog extends Animal { }`

## Multiple Inheritance (via Interfaces):

- A class can implement multiple interfaces, thus inheriting behaviors from multiple sources.
- Example: `class Dog implements Animal, Pet { }`

## Multilevel Inheritance:

- A class inherits from another subclass, forming a chain of inheritance.
- Example: `class Puppy extends Dog { } // Dog extends Animal`

## Hierarchical Inheritance:

- Multiple classes inherit from a single superclass.
- Example: `class Dog extends Animal { }` and `class Cat extends Animal { }`



# What is Polymorphism in JAVA?



Polymorphism in Java allows objects to be treated as instances of their parent class, enabling one interface to handle different underlying forms (or classes). It comes in two types:

1. **Compile-time Polymorphism** (Method Overloading): Achieved by defining multiple methods with the same name but different parameters. The method to be executed is determined at compile time.
2. **Runtime Polymorphism** (Method Overriding): Achieved when a subclass provides a specific implementation of a method already defined in its superclass. The method to be executed is determined at runtime.

Polymorphism promotes flexibility, code reuse, and the ability to maintain and extend code more easily.

# What is the difference between a constructor and a method?



A Method is a specific function contained within a class.  
A Method can not have the same name as the class name.  
A Method must provide void or a specific return type.  
A Method can be static, final, or abstract.

A constructor is a special type of method with the same name as the class.  
Constructor can not have a return type.  
Constructor's can have parameters.  
Constructors are invoked during object creation of the class.  
Constructors can be used to initialize fields for an object of the class.  
Constructors have the same name as the class Name.  
Constructors can not be static but can use access modifiers such as private or public.

## Can you override the main method?



Since the main method is static that means the method exclusively belongs to the class itself.

Therefore its not possible to override the main method of the class or any static method.

## Can you override a static method?



No, it's not possible to override a static method.

## What is the difference between this and super?



super keyword is used to reference the superclass of the current object. With the super keyword we can access superclass methods, constructors and data from the child class.

this refers to the current object. It is used to access the current object's fields, methods, constructors.

super is used with JAVA inheritance.

this is used exclusively with an object that is an instance of a class.

# What is the final keyword?



**final class:** A class that cannot be subclassed. It prevents any class from inheriting from it.

**final method:** A method that cannot be overridden by subclasses. It provides a concrete implementation that must be used as-is by any subclass.

**final variable:** A constant variable whose value cannot be changed once initialized.

# What are the access modifiers available in JAVA?



**public:** Accessible from any class.

**protected:** Accessible within the same package and by subclasses.

**default (no modifier):** Accessible only within the same package.

**private:** Not accessible from subclasses or other classes.

## Can a subclass access private members of its superclass?



Private members of a superclass can not be accessed unless the superclass provides public setter and getter methods that the subclass can use to access private members.



## What is an ArrayList in Java?



`ArrayList` is a part of the Java Collections Framework. It is a resizable array implementation of the `List` interface. It allows for dynamic resizing and provides methods to manipulate the size of the list, such as adding, removing, and accessing elements. Array List maintains insertion order and uses Index based ordering.

## Can an ArrayList be re-sized?



`ArrayList` is a dynamic array that will automatically resize when additional elements are added.

# What are some methods you have heard of as part of Array List?



Here's a summary of the common methods available in `ArrayList`:

## Adding Elements

- `add(E e)`: Adds an element to the end of the list.
- `add(int index, E element)`: Inserts an element at the specified position.
- `addAll(Collection<? extends E> c)`: Appends all elements from a collection to the end of the list.
- `addAll(int index, Collection<? extends E> c)`: Inserts all elements from a collection at a specified position.

## Accessing Elements

- `get(int index)`: Retrieves the element at the specified position.
- `indexOf(Object o)`: Finds the index of the first occurrence of an element.
- `size()`: Returns the number of elements in the list.
- `isEmpty()`: Checks if the list is empty.

## Modifying Elements

- `set(int index, E element)`: Replaces the element at a specified position with a new element.
- `remove(Object o)`: Removes the first occurrence of a specified element.
- `remove(int index)`: Removes the element at a specified position.
- `clear()`: Removes all elements from the list.

## Querying Elements

- `contains(Object o)`: Checks if the list contains a specified element.
- `toArray()`: Converts the list to an array.
- `toArray(T[] a)`: Converts the list to an array of a specified type.

# What is JAVA collections ?



The Java Collections Framework is a set of classes and interfaces that handle groups of objects. It provides various types of collections, such as:

- **List**: Ordered collections allowing duplicates (e.g., `ArrayList`, `LinkedList`).
- **Set**: Unordered collections without duplicates (e.g., `HashSet`, `TreeSet`).
- **Map**: Collections of key-value pairs (e.g., `HashMap`, `TreeMap`).

It also includes utility classes like `Collections` for common operations and `Arrays` for array manipulations. The framework provides efficient, flexible ways to manage and manipulate data.

## What is Hashset in java?



**HashSet:** Uses a hash table for storage. It does not maintain any order. Hashset implements the set interface.

## Can you store null in a Hashset ?



Yes you can store null as an element in a Hashset. However this can only be done once since Hashset only stores unique values.

## Can you store null in a Hashset ?



Yes you can store null as an element in a Hashset. However this can only be done once since Hashset only stores unique values.

## What is a Hashmap ?



**HashMap** is a part of the Java Collections Framework and implements the **Map** interface. It stores key-value pairs and allows for efficient retrieval, insertion, and deletion based on the key's. It does not guarantee the order of its elements.



## Can you have null as a key in Hashmap ?



Yes you can have null as a key only once since keys have to be unique for Hashmap.

## Do values have to be unique in a Hashmap ?



No only keys have to be unique values do not have to be unique.

## What is an exception in java ?



An exception is an event that disrupts the normal flow of the program's instructions. It is an object that represents an error or an exceptional condition.

# What is the difference between Error and Exception ?



**Error:** Represents serious problems that a reasonable application should not try to catch (e.g., `OutOfMemoryError`, `StackOverflowError`).

**Exception:** Represents conditions that a program might want to catch and handle (e.g., `IOException`, `SQLException`).

# How do you handle exceptions in JAVA ?



Exceptions are handled using `try`, `catch`, `finally`, and `throw` keywords:

- **try**: Block of code where exceptions might occur.
- **catch**: Block of code that handles the exception.
- **finally**: Block of code that executes regardless of whether an exception occurred or not.
- **throw**: Used to explicitly throw an exception.

## What is the difference between throw and throws ?



- **throw**: Used to explicitly throw an exception from a method or block.
- **throws**: Used in a method declaration to specify that a method can throw one or more exceptions.

## What is the purpose of the 'finally' block ?



The `finally` block is used to execute code that must run regardless of whether an exception was thrown or not. It is typically used for cleanup activities like closing resources.

## Can you have multiple catch blocks for a single try block ?



Yes, you can have multiple `catch` blocks to handle different types of exceptions. The order of `catch` blocks matters; more specific exceptions should be caught before more general exceptions.



# What is the difference between checked and unchecked exception ?



## Checked Exceptions:

- **Checked at compile-time:** Must be handled or declared in method signatures.
- **Examples:** `IOException`, `SQLException`.
- **Purpose:** Used for recoverable conditions like I/O operations.
- **Handling:** Must use `try-catch` or declare with `throws`.

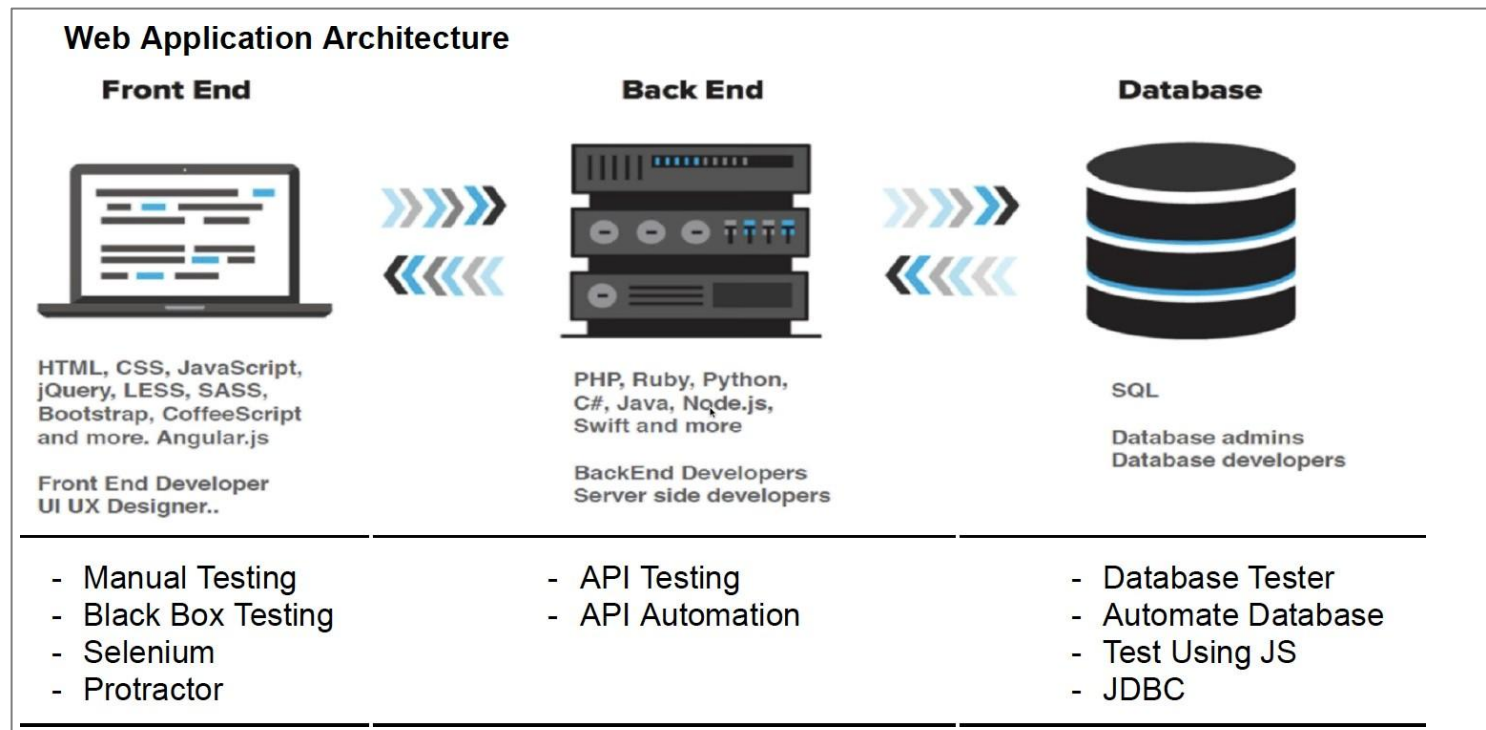
## Unchecked Exceptions:

- **Not checked at compile-time:** Handling is optional.
- **Examples:** `NullPointerException`, `ArrayIndexOutOfBoundsException`.
- **Purpose:** Indicates programming errors that typically need fixing.
- **Handling:** Not required to be caught or declared.



# What is SQL?

- SQL stands for Structured Query Language. It is the standard language for relational database management systems. It is especially useful in querying data related to software applications using a relational database management system.



# What is a Database?



- A database is an organized collection of data that is stored and managed on a computer system. It allows users to efficiently store, retrieve, and manipulate data.

## ***Examples of different databases:***

Examples of databases include MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. They are commonly used in applications like websites, business management systems, and mobile apps to manage the data they work with.

# RDBMS?



- **Relational Database Management System (RDBMS)** means that *tables in database are related using **primary/foreign key relationship**. Used to store, modify and retrieve data in the database.*
- **How are they related?**
  - Primary Key (unique and not NULL)
  - Foreign Key (duplicate and NULL)
- **What type of database system you have expertise with?**
  - RDBMS, such as SQL and Oracle

# Subsets / Categories of SQL?



## i. DML (Data Manipulation Language)

- DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

## ii. DDL (Data Definition Language)

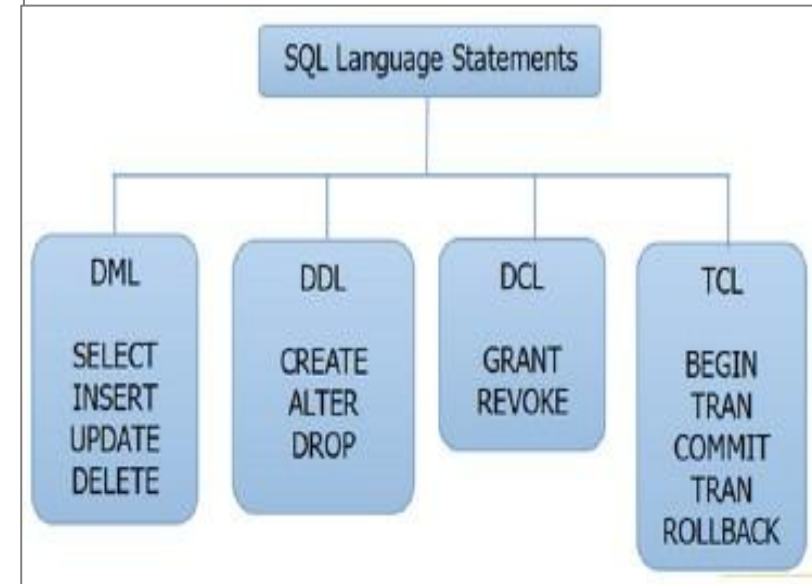
- DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

## iii. DCL (Data Control Language)

- DCL statements control the level of access that users have on database objects.

## iv. TCL (Transaction Control Language)

- TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.



# SQL vs MySQL

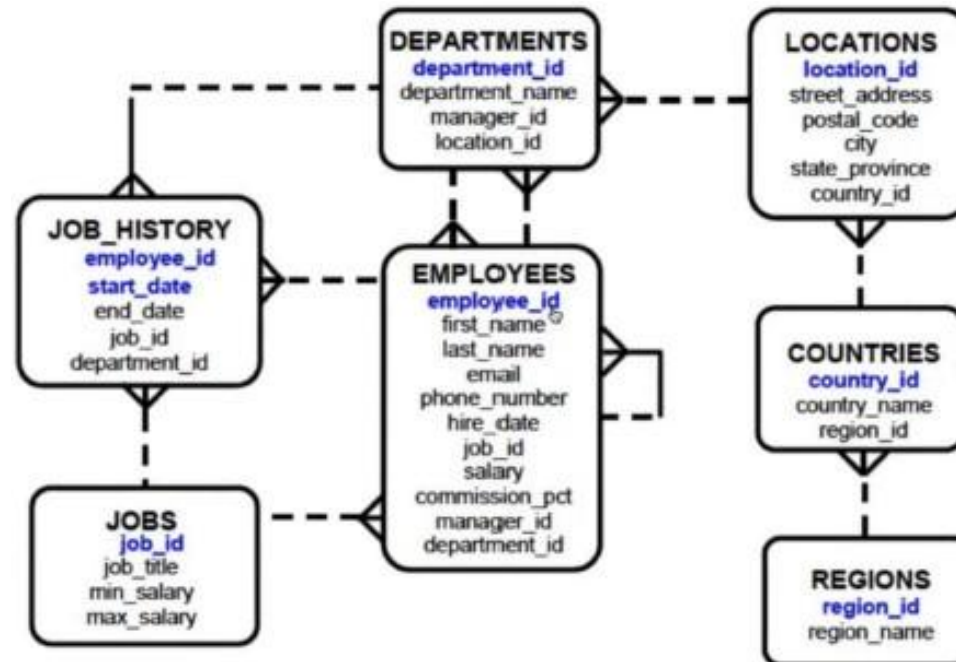


- SQL is a standard language for retrieving and manipulating structured databases. On the contrary, MySQL is a relational database management system, like SQL Server, Oracle or IBM DB2, that is used to manage SQL databases.

# DATABASE SCHEMA



- **DATABASE SCHEMA:** is a chart that shows all the tables and how they are related to each other.
- If there is no schema:
  1. Oracle ==> **SELECT** table\_name **FROM** user\_tables;
  2. MySQL ==> show tables;



# DML and DDL Commands



DML command actions can be restored.

Commands:

- **SELECT** from tablename; (**read**)
- **INSERT** into tablename values (...); (**add**)
- **UPDATE** tablename SET value WHERE location;
- **DELETE** from tablename WHERE location; (**rows**)
- **MERGE**

DDL command actions **cannot** be restored / undone.

Commands:

- **CREATE** table tablename (column1, column2 ...);
- **ALTER** table tablename modify value;
- **TRUNCATE** table tablename; (**delete whole table data**)
- **DROP TABLE**; (**delete whole table with structure**)
- **RENAME**
- **COMMENT**



# What are constraints?



- Properties that table column must comply with.
- Columns have constraints that defined how data can be stored.
- Having Constraints maintain data integrity.
- **Primary Key:** should be unique and NOT NULL  
**Foreign Key:** can be duplicate and NULL
- Data which is not in PK
  - Unique Key:** only unique value
  - Null:** can have null
  - Not null:** cannot have null

# Primary Key and Foreign Key



## What is Primary Key?

- It is unique column in every table in a database
- It can ONLY accept;
  - nonduplicate values
  - cannot be NULL

## What is Foreign Key?

- It is a column that comes from a different table and using Foreign key tables are related each other
- It is the primary key of another table
- It can be duplicate or null for another table

## Primary Key vs UNIQUE



- A Primary key can be a combination of one or more columns. A Primary must be unique and can not be null. There can only be a single primary key for each table.
- A Unique key must only have unique values. It is allowed to have null as value. There can be many Unique keys within a table.

# Data Types in SQL



- `number(num)` - whole numbers up to `num` digits
- `number(num,num2)` - `num` whole numbers up to `num2` decimals
- `char(num)` - fixed length character/string
- `varchar2(num)` - used for varying length data
- `date` - full date
- `currency` - used for prices

# DISTINCT



The **DISTINCT** operator is used to return unique values from a column.

Ex: `SELECT DISTINCT name  
FROM table;`

Returns unique names.

# COUNT



The COUNT function is an aggregate function that returns the number of rows in a table.

```
SELECT COUNT(column_name)
```

```
FROM table name;
```

# WHERE, ORDER BY, GROUP BY, HAVING



- SQL clause helps to limit the result set by providing a condition to the query. A clause helps to filter the rows from the entire set of records. For example – WHERE, HAVING clause.
- **WHERE** clause in SQL is used to filter records that are necessary, based on specific conditions. WHERE is a row operation.

```
SELECT *  
FROM employees  
WHERE first_name LIKE 'N%';
```

- **HAVING** clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since WHERE clause cannot filter aggregated records. HAVING is a column operation.

```
SELECT department_id, MIN (salary)  
FROM employees  
GROUP BY department_id  
HAVING MIN (salary) < 3500;
```

- **ORDER BY** clause in SQL is used to sort the records based on some field(s) in ascending (**ASC**) or descending order (**DESC**).

**GROUP BY** clause in SQL is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.

# Joins



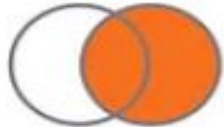
## LEFT JOIN



Everything on the left  
+  
anything on the right that  
matches

```
SELECT *  
FROM TABLE_1  
LEFT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

## RIGHT JOIN



Everything on the right  
+  
anything on the left that matches

```
SELECT *  
FROM TABLE_1  
RIGHT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

## OUTER JOIN



Everything on the right  
+  
Everything on the left

```
SELECT *  
FROM TABLE_1  
OUTER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

## INNER JOIN



Only the things that match on the  
left AND the right

```
SELECT *  
FROM TABLE_1  
INNER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```



# INNER JOIN



**INNER JOIN** ==> used to display data from multiple tables and returns only matching records

```
SELECT employee_id, last_name, employees.department_id, department_name  
FROM employees JOIN departments  
ON employees.department_id = departments.department_id;
```

```
SELECT employee_id, last_name, employees.department_id, department_name  
FROM employees JOIN departments  
USING (department_id);
```

# OUTER JOIN (Left Outer-Right Outer-Full Outer)



2.1. **RIGHT OUTER JOIN** — used to display data from multiple tables and returns matching records and non-matching records from right hand side table

```
SELECT student_id, student_lastname, c.course_id, course_name
FROM students s RIGHT OUTER JOIN courses c      -> right table
ON s.course_id = c.course_id;                  -> right table
```

2.2. **LEFT OUTER JOIN** — used to display data from multiple tables and returns matching records and non-matching records from left hand side table

```
SELECT student_id, student_lastname, c.course_id, course_name
FROM students s LEFT OUTER JOIN courses c      -> left table
ON s.course_id = c.course_id;                  -> left table
```

2.3. **FULL OUTER JOIN** — display data from both tables

# SELF JOIN



**SELF JOIN** ==> | used to display data from same table

```
SELECT e1.last_name, manager_id, e2.last_name  
FROM employees e1 JOIN employees e2  
ON e1.manager_id = e2.employee_id;
```

# Aggregate functions in SQL?



- SQL Aggregate functions determine and calculate values from multiple columns in a table and **return a single value**.
- There are 7 aggregate functions in SQL:
  - **COUNT()** : Returns number of table rows.
  - **MAX()** : Returns the largest value among the records.
  - **MIN()** : Returns smallest value among the records.
  - **AVG()** : Returns the average value from specified columns.
  - **SUM()** : Returns the sum of specified column values.
  - **FIRST()** : Returns the first value.
  - **LAST()** : Returns last value.

# 'BETWEEN' vs 'IN' condition operators?



- BETWEEN operator is used to display rows based on a range of values in a row whereas the IN condition operator is used to check for values contained in a specific set of values.

- **Example of BETWEEN:**

```
SELECT * FROM Students  
WHERE ROLL_NO BETWEEN 10 AND 50;
```

- **Example of IN:**

```
SELECT * FROM students  
WHERE ROLL_NO IN (8,15,25);
```

# DELETE vs TRUNCATE vs DROP statements?

- **DELETE** is used to delete or remove one or more existing rows.
- **TRUNCATE** deletes all the data from inside a table, but the table remains.
- **DROP** deletes everything, data, table and structure from the database.

```
//to delete values  
DELETE FROM students  
WHERE student_id = 104;
```

```
//clears table, keeps the table  
TRUNCATE TABLE students; -> can not be rolled back  
//deletes the table  
DROP TABLE students; -> can not be rolled back
```

# Views vs Tables



Views	Tables
It is a virtual table that is extracted from a database.	A table is structured with a set number of columns and a boundless number of rows.
Views do not hold data themselves.	Table contains data and stores the data in databases.
A view is also utilized to query certain information contained in a few distinct tables.	A table holds fundamental client information and the cases of a characterized object.
In a view, we will get frequently queried information.	In a table, changing the information in the database changes the information that appears in the view

# SET OPERATORS (UNION, UNION ALL, MINUS, INTERSECT)



1. Number of columns must be same
2. Data type should be same

- **UNION** (*returns combined rows from 2 independent queries and removes duplicates and sorts them*)
- **UNION ALL** (*returns combined rows from 2 independent queries but DOES NOT remove duplicates or sort them*)
- **MINUS** (*returns records from 1 query that are not present in 2 query*)
- **INTERSECT** (*returns only common for both queries data*)



## Add a column to a table?



- To add another column in the table, I write the command like::

```
ALTER TABLE table_name ADD column_name varchar(50);
```

# Order of SQL SELECT



Order of SQL SELECT clauses is:

- SELECT
- FROM
- WHERE
- GROUP BY
- HAVING
- ORDER BY
- Only the SELECT and FROM clauses are mandatory

# How do you use SQL to test ?



## 1. Data Validation

- **Data Retrieval:** Check that data is correctly stored by retrieving records based on specific conditions.
- **Aggregations:** Validate calculations like totals or averages to ensure accuracy.

## 2. CRUD Operation Testing

- **Insert Testing:** After adding a record through your application, verify that it appears correctly in the database.
- **Update Testing:** Confirm that updates made through the application are reflected in the database.
- **Delete Testing:** Ensure that records deleted via the application are actually removed from the database.

## 3. Data Integrity Testing

- **Foreign Key Constraints:** Ensure relationships between tables are maintained and enforced correctly.
- **Unique Constraints:** Verify that fields meant to be unique, like email addresses, are not allowing duplicates.

# UPDATING ROW - DELETE ROW



## UPDATING ROW

```
Update TableName set ColumnName = value where condition;  
update scrumteam set firstname = 'Martin' where EmployeeID='1';  
update scrumteam set lastname = 'Murtin' where firstname='Tom';
```

## DELETING ROW

```
delete from TableName where condition;  
delete from scrumteam where firstname='Jack';  
delete from scrumteam where JobTitle='SDET';
```

# CREATE TABLE



```
//to create a table
```

```
CREATE TABLE students  
(  
    student_id number(4) primary key,  
    last_name varchar2(30) NOT NULL,  
    course_id number(4) NULL );
```

```
//to insert values
```

```
INSERT INTO students VALUES (200, 'Jones', 101);  
INSERT INTO students VALUES (201, 'Smith', 101);  
INSERT INTO students VALUES (202, 'Lee', 102);  
COMMIT; -> to save changes
```

## TO INSERT VALUES



```
//to insert values  
INSERT INTO students VALUES (200, 'Jones', 101);  
INSERT INTO students VALUES (201, 'Smith', 101);  
INSERT INTO students VALUES (202, 'Lee', 102);  
COMMIT; → to save changes
```

# TO UPDATE VALUES



```
//to update values
```

```
UPDATE students SET course_id = 102
```

```
WHERE last_name = 'Jones'; -> if there is no condition it will update all!
```

# WHAT IS A SUBQUERY ?



A subquery is a query within a query.

A Subquery can be used when you need another query to filter a specific value that becomes an input to your main query.

```
SELECT *  
FROM table where column = (Inner query);
```

Need to find the employees that make more than the average salary?

```
Select employee_name, phone, email  
From employees  
Where avg salary > (SELECT avg salary from employees);
```



# WHAT Is the difference between UNION and UNION all ?



UNION  $\Rightarrow$  When you combine the output between multiple independent select statements.  
(removes duplicate)

UNION ALL  $\Rightarrow$  (output will be duplicates and non duplicates)

Rules:

1. Same number of columns in each query.
2. Order of the column and data type needs to match.

# How do you retrieve duplicate values in the database?



We can retrieve duplicate values by using GROUP BY with HAVING clause to filter on the number of occurrences.

Example:

```
Select first_name, count(*)  
FROM table  
WHERE first_name != 'value'  
group by first_name  
Having count(*) > 1;
```

# Tell me about yourself



First of all, I would like to thank you for giving me this opportunity and I really appreciate your time.

I've been working in the IT industry for 8 years. Throughout my career I've specialized in automation, but I was also involved and I am very comfortable with manual and back-end testing.

Currently, I am holding an SDET position in my team and my main responsibilities are to design, develop and maintain Test Automation Framework that verifies user stories and system requirements.

I have hands on experience on UI, API and DataBase testing. My role in my company includes running FUNCTIONAL TEST, SMOKE TEST and REGRESSION TEST for UI part and also API Testing for the backend.

CUCUMBER BDD framework. I developed my "testing framework"

- I have specialized in JAVA. I used SELENIUM as a testing tool with

from scratch based on Page Object Model (POM) and I have worked with MAVEN as a build management tool and SELENIUM with JUnit

testing framework. I have also worked with TestNG and Data Driven Testing using Apache POI.

- I use GIT for version control and I also use JIRA for bug tracking and project management tool.

- I have extensive knowledge of SQL queries and I used POSTGRESQL

to interact with a relational database and and JDBC for database testing. I am also comfortable with the Data Driven (DDD) Frameworks.

- I have worked on API testing on my project and I used POSTMAN and REST ASSURED LIBRARY for API testing.

- I achieve continuous integration and schedule my test executions by using Jenkins.

- I'm familiar with Agile ENVIRONMENTS and currently I'm working in a Scrum Team and proficient in all sprint-related Scrum ceremonies.

-I am a cross- functional team member. As far as soft skill

concerned, I consider myself a positive person, adaptable to changing circumstances,

- I can work well individually and in a team. I am detail oriented, a

technology, and always make sure I meet the deadlines. That's pretty much about myself.

# Tell me about your Project



- **My current project is a web-based loan application.**
- As you can see from my resume, I am currently working in a loan company that works with small companies. I am working for loan authorization department.
- The application we are responsible for is mainly developed to store the all required information about the customer and share it with the company headquarter. It also generates the details of the status of all pending, denied and authorized loan applications.
- As an SDET my main responsibility is automating the UI part of the application, but I am also aware of the back end and database part of the application.
- My role in this project includes running regression, smoke and functional testing. As a cross functional team member, I also I help manual testing when required.

# Daily Activities



- My daily activities at work, I go to work early in the morning and check result report of Smoke Test to make sure that environment is up and running and the application is stable or not for the day.
- If something goes wrong, I will send out an email to my team so they can take care of it asap before everyone comes to work, to reach maximum productivity.
- And then I check my email if there are any important tasks or notices, also check my schedule if there are any meetings for the day and also check Jira to review what needs to be done that day in which priority.
- Then I go to attend daily standup meeting at 10:00 am. with my scrum team to talk about what we did yesterday, what we will do today and are there any impediments in my way. This meeting is simply to synchronize our team and it takes about 15 minutes.
- After that, I go back to my desk and start automating test cases from regression suits. And also, I automate test cases from sprint backlog after doing manually if it is passed.
- That is pretty much about my daily activities at work.

# Role



- As an automation engineer, I develop my 'testing framework' based on POM (Page Object Model). I performed various types of testing, like; functional testing, smoke testing, regression testing and back-end testing. I am responsible executing Regression test when developers add new functionality to the application or end of the sprints.
- I am also responsible to check report of Smoke Test to make sure that environment is up and running first thing in the morning.
- If there are any issues, I analyze them.
- If it is service issue, I will immediately contact developers.
- If it is about my scripts, I debug my scripts and fix it.
- If it is a bug, I re-produce it and log the defect.
- And also I am using Jira as bug management tool. Once the bug reports fixed by the developers, I re-test it and if it is passed I close it. If the defect is not fixed, I re-open it. Also, as a part of the Agile Scrum Team, I participate in the several walkthroughs meeting for the requirement reviews and provide valuable feedback to the BA. Lastly, I am cross-functional team member that is always willing to help my team in any way to achieve our sprint goal.
- That is pretty much about my role as an automation engineer.

# Framework



- My framework is written using Java OOP language. Java is a powerful and robust programming language and selenium with java is an open source and has a large community that support each other.
- Selenium WebDriver is a library/tool/API which is used to automate the browser, it interacts with the browser. It used for UI automation testing.
- My framework is created as a maven project; maven is used to manage dependencies and also run our tests, as mvn goals from terminal.
- We use Junit as a testing tool. Junit is used to kick off cucumber tests and also do assertions.
- In my framework we are using POM according to which has separate class for every pages of the application.
- My framework uses a singleton pattern to share the WebDriver instance between different classes.
- I have in my framework configuration.properties file to store the important test data, like usernames, passwords, url's, browsers etc..
- In my Utilities package I have reusable utilities which can be used across different classes of my framework, like waits that I need to use always, ConfigurationReader that reads my configuration.properties file, and My Driver that I can choose which browser I will use in this test case...
- (Types of tests) My framework can test the UI, Back-end and API
- We used Cucumber to write test cases, requirements, specifications in a GHERKIN language understandable by non-tech people
- My framework generates detailed HTML and JSON reports with is easy to read and understand to non-technical team members. My reports have details test steps and screenshots for any failures that may occur. It can also do metrics on what percentage is passing, failing, skipped etc.
- We use Git, Github for Version Control.
- As a Continuous Integration/Deployment tool we use Jenkins.

# Team Structure



- My team consist of adaptive, cross-functional and self-organized individuals that highly motivated and knowledgeable. We have 11 people in my team.
  - 5 developers (Jacobs, Suzan, Tim, Alejandra, Megane),
  - 3 Testers (Irina, Alex, me)
  - 1 SM (John)
  - 1 PO (Brian)
  - 1 BA (Shaun)





# Main Responsibility as an SDET?

- Turn manually executed test scenarios into automatically executed test scenarios via a Selenium, Java and Cucumber testing framework and Gherkin.
- Design and develop test plans that verify user stories and system requirements.
- Develop and automate test cases to ensure what we build meets the highest levels of quality.
- Functional Testing, Regression Testing, Smoke Testing

# Agile experience in your most recent project?

- Our sprint is 2 weeks and we have release every 3 sprints as a release cycle. We have 11 people in my team. 5 developers , 3 testers, also 1 SM, 1 BA and 1 PO.
- We start a sprint with Sprint Planning Meeting and we learn the part of the application which we are going to develop. We get general idea than we do Sprint Grooming for giving some estimates for the tasks.
- After sprint starts, we do Daily Standup Meeting everyday morning and we discuss what did we do yesterday, what will we do today and is there any blocker. Just we synchronize info about the sprint.
- End of the sprint, usually we do Sprint Demo/Review Meeting . It is just to show customer what we build throughout the sprint. PO puts feedback. As an SDET in my team, I have done presentation sometimes and go over through the functionalities in the meeting room. Client or stakeholders or business people they ask questions what they don't know or what they want to know.
- After Sprint Demo, we do Sprint Retrospective Meeting. In Sprint Retro, we talk about what was good in last sprint, what kind of mistakes we made. We go over them and make sure that we don't make the same mistakes again. If we did something good , we would continue doing it.
- This is pretty much our Sprint process

# Biggest Accomplishment?



- I would say is establishing a great trustworthy relationship within the team. If you are asking for technical: When I joined my last project, the application had very less “id” so I had to spend hours to locate one WebPage elements in my POM project so I communicated with developers and other team members and all together we come up with the solution which I got the access to put id in the application by myself. That was great for me it saved my and others time. So instead of spending time to locating elements I spend my time to more creating automation test scripts and executing them.

# Why did you apply for this position?



- After looking at the job description, I think it matches my day-to-day activity and my experience.
- I was confident with the job description that's why I applied.
- Also, I have done some research on the company and I am really excited about the company's product and services like...

# Where do you see yourself 5 years from now?

- I want to learn as much as possible to be more technical. I would like to see myself SDET. I want to be technically very competitive person 5 years from now.

# Why should we hire you?



- I think you should hire the candidate that has the best qualifications for this position. Since I don't know the other candidates, I can represent only myself. I think my experience and technical expertise will bring a lot of values and benefits to the company and the project. I think that's why you should hire me.

# Weakness?



- Well, I think my weakness is that whenever I am given some responsibilities and there is a deadline for it, I work day and night, sometimes 7 days a week. This is bad for my family life, the reality is I can not sleep unless I am done with my assignments.

# Strength?



- Well, I am very detail oriented person. I have the sense of urgency. I can prioritize my job according to the deadline. I am very much dedicated towards my job. I am honest. I have the skills and expertise in QA process. These are some of my strengths.



# Challenge you faced during your last project?



- I think, one of the biggest challenges that I faced with in my current project is that...
- ... everytime I found a bug, the developer disagreed to accept it and most of the time we had to ask QA for clarification. Then realize the requirement itself was not as specific as you thought the project understood. We fed it to the developer. In the end, you solved more. I think, the most important problem is misunderstanding and the lack of communication in the business life. If we come together as a group and discuss it, there is nothing we cannot solve. I'm really grateful and blessed to have been in the team that I was in, because we were able to collaborate and come together to solve the problem.
- The challenge I have faced is locating dynamic elements by retrieving the right HTML code. One of my recent challenge is that another coworker who is also QA had to leave.

# How do you handle stress? Or Conflict?



- Nothing is personal. Everyone thinks company's benefits so I would like to explain my concern and his/her explanation makes sense for me. Of course, I can do the things which is most helpful to my company. So, I try to communicate with his/her and I would try to understand the concern. Because everyone have the same goal and wants to get job done successfully. Also in scrum environment we working as a team . I always maintain good communication and relationship with my colleagues. So they trust me and they can communicate with me very easily. . . I always avoid miscommunication and my team believe me every time.

# Can you start tomorrow?



- My team won't be happy with me if I leave tomorrow, and I don't think it is professional and I have never done that before. I have to transfer the automation framework knowledge to other team members before I leave.

# What do you do if I hire you?



- In first week, you know, I will get done all the paper works, getting the machine to the project, databases etc. Then I will have to learn the company culture. I have to learn about my projects and my teammates. I think, understanding what the project is important if I want to be more productive.

# how long are you planning to stay?



- As long as there is a project to work, I am willing to stay as long as possible.

# Can you work under pressure?



- I don't remember any project that I worked had no pressure. Pressure is good thing sometimes. It forces you to work harder and smarter.

# What do you like the most about testing?



- Testing is fun job for me because you are very important person to the client and testing because as end user I want to buy better product that is piece of art and helping others to make sure their product has top quality.

# What to do in case of you have too much work



## and you can not finish for the deadline?

- When developers don't deploy their code on time, our tester team don't have enough time for completion. And the upper management keeps asking for us for completion. - Some of my team members simply focuses on task completion and not on the test coverage and quality of work. - So, at the Sprint Grooming Meeting, I suggested that we should work very closely with the developer and make sure that we are communicating on daily base. And also, the developers prioritize the important tasks and work on them first. Any scenarios left, would be pushed to the next sprint since it is not as important as the other ones. - Lastly, I try to prioritize my work and follow my test lead and manager whatever they see is more important I start with that.



# Do you have any question for us?

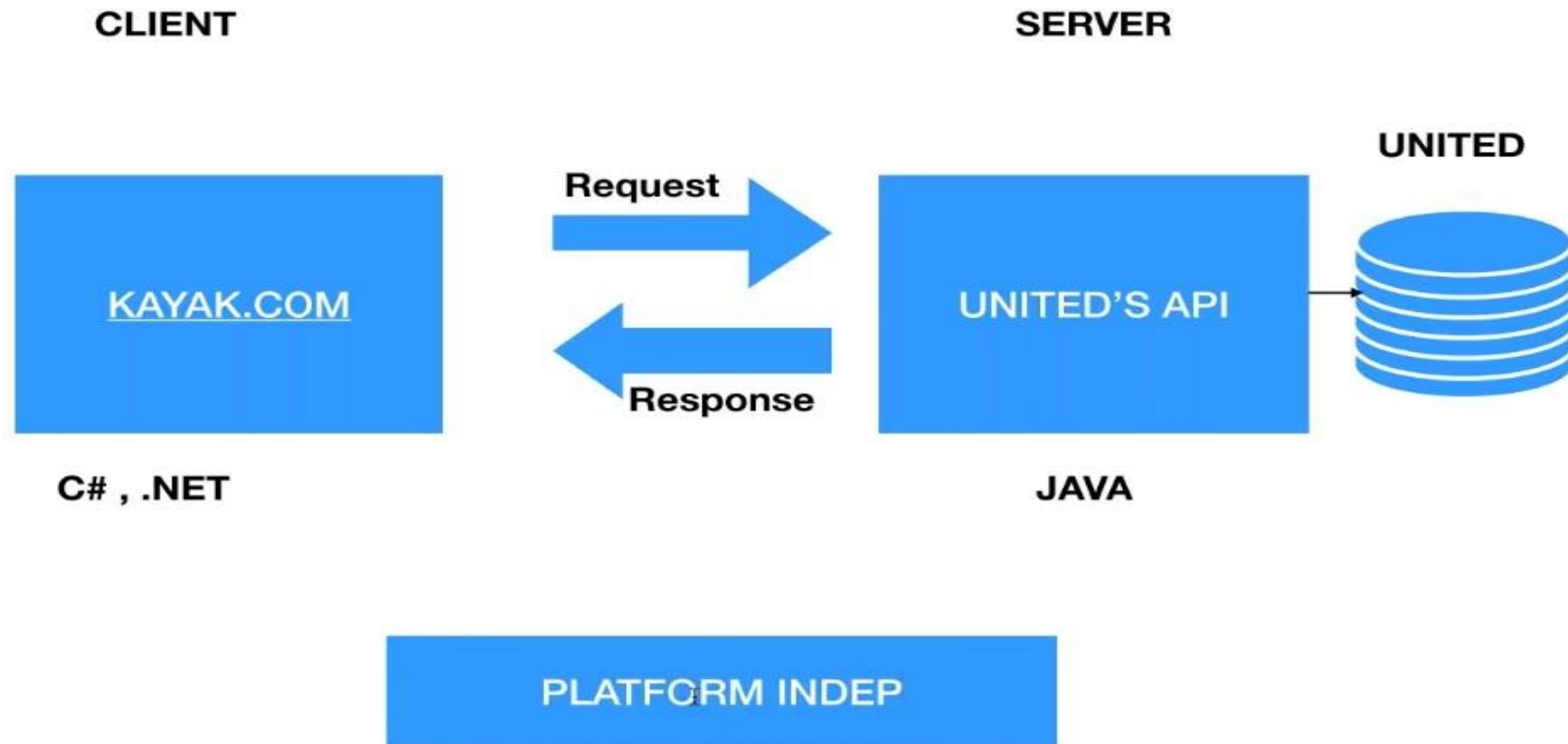


- it's important to me that I continually improve and try to achieve excellence in my position and the best way to do this is to continually learn. I'm always trying to learn new things or learn old things better.
- Do you provide trainings, seminars or anything to support the education of your employees?
- What are the next steps in the interview process?

# API -> APPLICATION PROGRAMMING INTERFACE



CLIENT -> -> -> SERVER(API  
SERVER) CLIENT <- <- <-  
SERVER(API SERVER)



API is a middle man between database and client.



# API Interview Questions

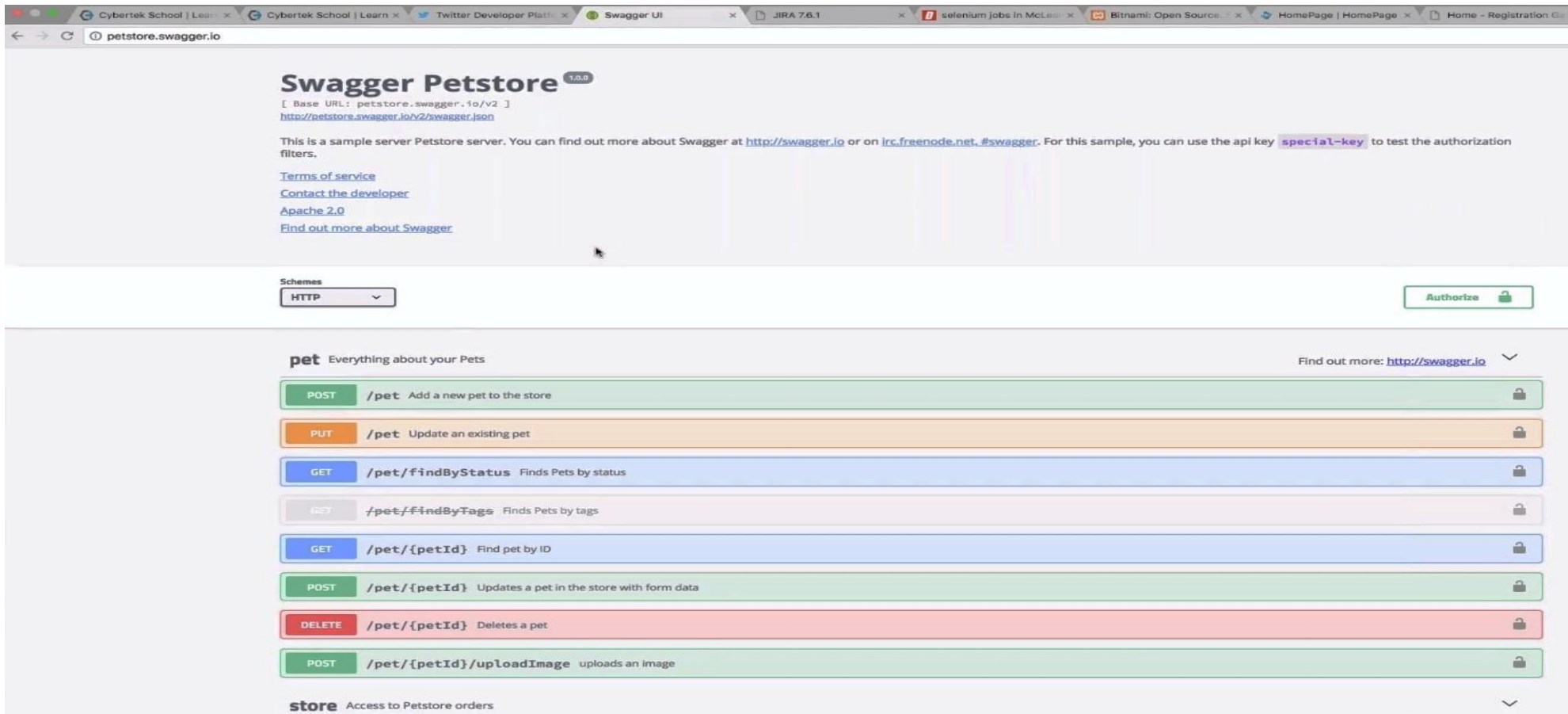
- If API communication happens through internet , we can also call it **web service**.
- **HOW DO WE TEST APIs?**
- As we know in API, there is request and response communication happens between client and server.
- As testers, we send a request to an API and verify the response.
- Request -> types of requests in RestApi:
  - > GET request -> READ data (like SELECT in sql)
  - > POST request -> is to CREATE data
  - > PUT request -> UPDATE data
  - > DELETE request -> DELETE data

# What are the **Http methods** and **request types**

- **Get** does not requires body
  - (retrieves data from given server using a given URI)
- **Put** requires body means UPDATE information
  - (Replaces all current representations of the target resource with the uploaded content)
- **Post** requires body means CREATE information
  - (send data to the server)
- **Delete** does not requires body
  - (Removes all current representations of the target resource given by a URI.)

# API Documentation - Swagger is a tool for API documentation .

- Swagger is a tool for API documentation.
- API documentation is a technical content deliverable, containing instructions about how to effectively use and integrate with an API. It's a concise reference manual containing all the information required to work with the API



The screenshot shows the Swagger UI for the Petstore API. The browser address bar displays `petstore.swagger.io`. The page title is "Swagger Petstore 1.0.0" with a sub-header "[ Base URL: petstore.swagger.io/v2 ]" and a link to the Swagger JSON file: `http://petstore.swagger.io/v2/swagger.json`. A descriptive paragraph states: "This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on [#swagger](https://irc.freenode.net). For this sample, you can use the api key `special-key` to test the authorization filters." Below this are links for "Terms of service", "Contact the developer", "Apache 2.0", and "Find out more about Swagger".

At the bottom of the page, there is a "Schemes" dropdown menu set to "HTTP" and an "Authorize" button with a lock icon. The main content area lists API endpoints under the "pet" category, described as "Everything about your Pets". A link "Find out more: <http://swagger.io>" is visible. The endpoints are:

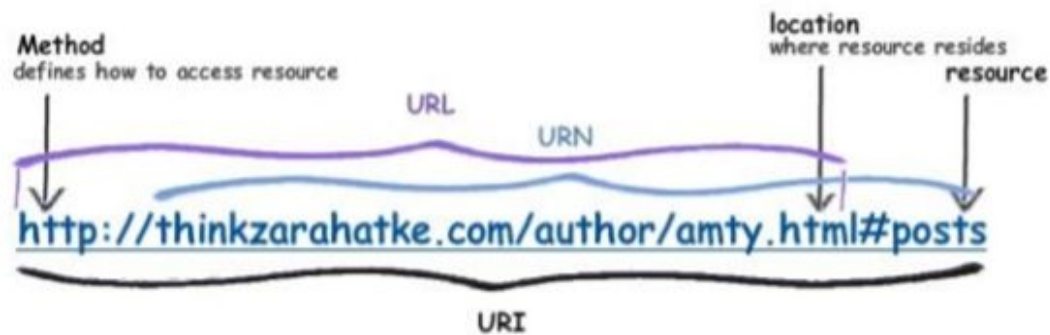
- POST** `/pet` Add a new pet to the store
- PUT** `/pet` Update an existing pet
- GET** `/pet/findByStatus` Finds Pets by status
- GET** `/pet/findByTags` Finds Pets by tags
- GET** `/pet/{petId}` Find pet by ID
- POST** `/pet/{petId}` Updates a pet in the store with form data
- DELETE** `/pet/{petId}` Deletes a pet
- POST** `/pet/{petId}/uploadImage` uploads an image

At the bottom, the "store" category is partially visible, described as "Access to Petstore orders".

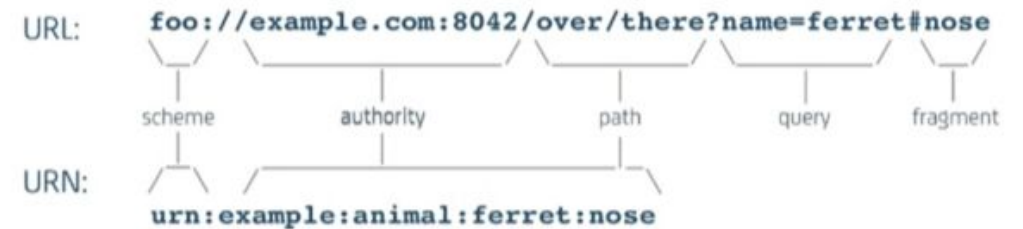


# Do you have API documentation website for your API?

- Yes we use swagger for our api documentation and this is where the description and guidelines of API endpoints are



The structure of URIs



- URL(Uniform Resource Locator) ==> <https://www.google.com/index.html>
- URN(Uniform Resource Name) ==> [www.google.com/index.html](http://www.google.com/index.html)
- URI(Uniform Resource Identifier) ==> <https://www.google.com/index.html>
- When we are doing something with API, it means that we are skipping UI and directly get the data/info from Web Services.

# 2 TYPES OF PARAMETERS IN REST SERVICES

## 1) QUERY/REQUEST PARAMETERS

-> is not part of url and passed in key+value format  
those parameters must be defined by API  
developer

<http://34.223.219.142:1212/ords/hr/employees?limit=100>

## 2) PATH PARAMETERS

-> is a part of URL and followed by the end of full resource url  
<http://34.223.219.142:1212/ords/hr/employees/100>

# When to use @QueryParam vs @PathParam

- If there is a scenario to retrieve a record **based on id**, for example you need to get the details of the employee whose id is 15, then you can have resource with @PathParam.

GET /employee/{id}

- If there is a scenario where you need to get the **details** of all employees but only 10 at a time, you may use query param

GET /employee?**start=1&size=10**

(This says that starting employee id 1 get ten records.)

- To summarize, use @PathParam for retrieval based on id. Use @QueryParam for filter



# How do you test rest api?



- I use POSTMAN for manual API testing and use RESTASSURED library in Java for automation.
- I verify if each REST API endpoint is working as expected.
- I send POST,PUT,GET, DELETE type of requests and verify response status code and response body and header.

# What are headers in REST API?



- I am using **Accept.(ContentType.JSON)** type checks what I am **receiving** should be in **JSON** or **XML** format
- I am using **ContentType.(ContentType.JSON)** checks what I am **sending** should be in **JSON** format



# What **Request Line** includes?

- **end point** —> address where we send the request

- **base url** where API is

- **resources** resources inside baseURL

- **parameters** separated from resources with ‘?’.

**GET** —> get some data without changing it in the server.

- accept type: response in Json or xml

- authorization tokens, credentials

**POST** —> add data to the server. Parameters sent in xml/json as part of request body/payload.

- accept type: response in Json or xml

- content type: request body in Json or xml

- authorization tokens, credentials

**PUT** —> replaces existing data in the server.

**DELETE** —> deletes data.

*// APIs do different types of operations: **C**reate **R**ead **U**ppdate **D**elete.*

# What **Response** includes?



- **Status code** —> defines if the request was successful.
- **Header** —> metadata (Time of execution, size etc.)
- **Body** —> returned information from the server. responses can be in different format (Json, XML, text, HTML)

# API TEST STRATAGY



- API testing involves APIs directly and checks whether the API meets expectation **functionality, reliability, performance, and security** of an application. My first of **testing** which ensures that the API functions correctly.
- The main objectives in **functional testing** of the API are:
  - to ensure that the implementation is working correctly as expected - no bugs!
  - to ensure that the implementation is working as specified according to API documentati
  - to prevent regressions between code merges and releases.



# I HAVE FOUR DIFFERENT PROCESS TO IMPLEMENT

## • Checking API contract –SWAGER:

An API is essentially a contract between the client and the server or between two applications. Before any implementation test can begin, it is important to make sure that the contract is correct.

- Endpoints are correct,
  - Resource correctly reflects the object model (proper JSON/XML structure used in response),
  - There is no missing functionality or duplicate functionality.
- Now that we have verified the API contract, we are ready to think of what and how to test.
- Relationships between resources are reflected in the API correctly.

## • Creating test cases

- I mostly create :
- Basic positive test (happy paths)
- Extended positive testing with optional parameters and extra functionality.
- Negative testing with valid input (trying to add an existing username)
- Security, authorization, and permission tests (sending valid or invalid access tokens to permitted or unpermitted endpoints)
- Negative testing with invalid input (trying to add a username which is null)
- Destructive testing (sending null, empty string, integer or other types, odd date format, deleting necessary parameters)

## • Executing test cases

- For each API request I need to verify:
- I check **Data accuracy**: I check the request and response body whether those are as written on API documentation in terms of data type and data structure.
- I check **HTTP status code**: For example, creating a resource should return 201 CREATED and unpermitted requests should return 403 FORBIDDEN, etc.
- I check **Response headers**: HTTP server headers have implications on both security and performance.
- I check **Response body**: Check the JSON body and time out field names, types, and values - including in error responses.
- I check **Authorization checks**: Check authentication and authorization
- I check **Error messages**: Check the error code coverage in case API returns any error

## • Implementing different test flows

- Single-step workflow:
- Multi
- Combined API and UI test: -step workflow with several requests:

# How do you test rest api?



- I also do positive and negative testing of API.
- **When I do positive testing,**
  - I send
    - valid request parameters
    - valid headers,
    - valid request JSON body
  - verify that response status code is 200 successful and JSON response body data is also matching the expected.
- **When I do negative testing,**
  - I send
    - invalid, request
    - invalid headers, or
    - invalid request json body and
  - verify that response status code is not 200 and Json response body contains error message.

# Ways to navigate JSON and VERIFYING RESPONSE DATA



## 1) Not recommended:

treat the response json as a String and do contains assertions on it.

```
AssertTrue(response.body().asString().contains("Java"));
```

## 2) PATH() method.

Extracting city for response.path("employee.address.city") assertions for verification.

```
assertEquals("New York", city);
```

## 3) JSONPATH object:

Convert Response data into JsonPath object and use jsonpath getter methods to extract values. Do assertions using JUnit

```
JsonPath json =
```

```
response.jsonPath().get("Ayse",
```

## 4) HAMCREST MATCHERS WITH PATH USING CHAINING:

```
json.getString("teachers.first_name")
```

We can do assertions in single statement by chaining methods in restAssured. To find values in the json body , we use the same path syntax:

```
and().assertThat()
```

```
.body("teachers.firstName",contains("Esen"),
```

```
"teachers.lastName",contains("Niiazov"),
```

```
"teachers.emailAddress",contains("eniazov@gmail.com")));
```

## 5) Java Collections/Data Structures to manipulate Json Data. JSON RESPONSE --> JAVA DATA STRUCTURE/COLLECTION

Response Json Data:

```
{
    "year", 2000,
    "make", "Honda"
    "model", "f500",
    "mileage", 50234
}
```

```
Map<String, Object> dataMap = response.body().as(Map.class)
```

```
dataMap.get("year") => 2000
```

```
dataMap.get("make") => Honda
```

```
http://api.cybertektraining.com/teacher/name/{name}
{
  "teachers": [
    {
      "teacherId": 2381,
      "firstName": "Esen",
      "lastName": "Niiazov",
      "emailAddress": "eniazov@gmail.com",
      "joinDate": "01/01/2018",
      "password": "123456123456",
      "phone": "12345678910",
      "subject": "Java",
      "gender": "Male",
      "department": "Computer",
      "birthDate": "01/01/1992",
      "salary": 100000,
      "batch": 11,
      "section": "1",
      "permanentAddress": "2700 S. River, Des Plaines, IL 60600"
    }
  ]
}
```



treat the response json as a String and do contains assertions on it.



### 1) **asString() method convert body to**

```
String responseBody = response.body().asString();
```

```
assertTrue(responseBody.contains("Java")); //assertTrue(response.body().asString().contains("Java"));
```

### 2) **PATH() method.** Extract values from JSON using path() method, use Junit assertions for verification.

```
String city = response.path("employee.address.city");
```

```
assertEquals("New York", city);
```

### 3) **JSONPATH object:** Convert Response data into JsonPath object and use jsonpath getter methods to extract values. Do assertions using JUnit. json.getString("x", "y"), json.getInt(a, b)....

```
JsonPath json = response.jsonPath();
```

```
String tName = json.getString("teachers.first_name")
```

```
assertEquals("Ayse", tName); //assertEquals("Ayse",
```

```
json.getString("teachers.first_name")); long phone = json.getLong("teachers.phone")
```

```
assertEquals(12345678910, phone);
```

### 4) **HAMCREST MATCHERS WITH PATH USING CHAINING:** We can do assertions in single statement by chaining methods in RESTassured. To find values in the json body , we use the same path syntax:

```
.and().assertThat()
```

```
.body("teachers.firstName",contains("Esen"),
```

```
"teachers.lastName",contains("Niiazov"),
```

```
"teachers.emailAddress",contains("eniiazov@gmail.co
```

```
m"));
```

```
"mileage", 50234
```

```
}Map<String, Object> dataMap = response.body().as(Map.class);dataMap.get
```

```
=> 2000
```

### 5) **Java Collections/Data Structures to manipulate Json Data.**

**JSON RESPONSE --> JAVA DATA STRUCTURE/COLLECTION**

Response Json Data:

```
{
  "year": 2000,
  "make": "fiat",
  "model": "f500",
}
```

# When I do negative testing



```
4
5  /*
6   * When I send a GET request to REST URL:
7   * http://34.223.219.142:1212/ords/hr/employees/1234
8   * Then status code is 404
9   * And Response body error message is "Not Found"
10  *
11  */
12  @Test
13  public void negativeGet() {
14  //    when().get("http://34.223.219.142:1212/ords/hr/employees/1234")
15  //    .then().statusCode(404);
16  Response response = when().get("http://34.223.219.142:1212/ords/hr/employees/1234");
17  assertEquals(response.statusCode(), 404);
18  assertTrue(response.asString().contains("Not Found"));
19  }
```

What first thing you check when you get response? 

- Status code (200 always mean Ok)
- We always check the 404 means not found



# API STATUS CODES -HTTPS STATUS CODE

This page is created from HTTP status code information found at [ietf.org](http://ietf.org) and [Wikipedia](http://Wikipedia). Click on the **category heading** or the **status code** link to read more.

## 1xx Informational

100 Continue

101 Switching Protocols

102 Processing (WebDAV)

## 2xx Success

★ 200 OK

203 Non-Authoritative Information

206 Partial Content

226 IM Used

★ 201 Created

★ 204 No Content

207 Multi-Status (WebDAV)

202 Accepted

205 Reset Content

208 Already Reported (WebDAV)

## 3xx Redirection

300 Multiple Choices

303 See Other

306 (Unused)

301 Moved Permanently

★ 304 Not Modified

307 Temporary Redirect

302 Found

305 Use Proxy

308 Permanent Redirect (experimental)

## 4xx Client Error

★ 400 Bad Request

★ 403 Forbidden

406 Not Acceptable

★ 409 Conflict

412 Precondition Failed

415 Unsupported Media Type

418 I'm a teapot (RFC 2324)

423 Locked (WebDAV)

426 Upgrade Required

431 Request Header Fields Too Large

450 Blocked by Windows Parental Controls (Microsoft)

★ 401 Unauthorized

★ 404 Not Found

407 Proxy Authentication Required

410 Gone

413 Request Entity Too Large

416 Requested Range Not Satisfiable

420 Enhance Your Calm (Twitter)

424 Failed Dependency (WebDAV)

428 Precondition Required

444 No Response (Nginx)

451 Unavailable For Legal Reasons

402 Payment Required

405 Method Not Allowed

408 Request Timeout

411 Length Required

414 Request-URI Too Long

417 Expectation Failed

422 Unprocessable Entity (WebDAV)

425 Reserved for WebDAV

429 Too Many Requests

449 Retry With (Microsoft)

499 Client Closed Request (Nginx)

## 5xx Server Error

★ 500 Internal Server Error

503 Service Unavailable

506 Variant Also Negotiates (Experimental)

509 Bandwidth Limit Exceeded (Apache)

598 Network read timeout error 

501 Not Implemented

504 Gateway Timeout

507 Insufficient Storage (WebDAV)

510 Not Extended

599 Network connect timeout error

502 Bad Gateway

505 HTTP Version Not Supported

508 Loop Detected (WebDAV)

511 Network Authentication Required



# What methods are you using to verify the size of the response data?

```
@Test
public void testItemsCountFromResponseBody() {
    given().accept(ContentType.JSON)
        .when().get(ConfigurationReader.getProperty("hrapp.baseresturl")+"/regions")
        .then().assertThat().statusCode(200)
        .and().assertThat().contentType(ContentType.JSON)
        .and().assertThat().body("items.region_id", hasSize(4))
        .and().assertThat().body("items.region_name", hasItem("Americas"))
        .and().assertThat().body("items.region_name", hasItems("Americas", "Asia", "Middle East and Africa"));
}
```

I use Matchers from

Hamcrest

- hasItem()
- equalTo()



# Why do you use JSONPath?

```
APIDay3_JsonPath.java configuration.properties
105 public void testWithJsonPath() {
106
107     Map<String,Integer> rParamMap = new HashMap<>();
108     rParamMap.put("limit", 100);
109
110     Response response = given().accept(ContentType.JSON)//header
111         .and().params(rParamMap) //query param/request param
112         .and().pathParams("employee_id", 177) //path param
113         .when().get(ConfigurationReader.getProperty("hrapp.baseresturl")+"/emp
114
115     JsonPath json = response.jsonPath(); //get json body and assign to jsonPath object
116
117     System.out.println(json.getInt("employee_id"));
118     System.out.println(json.getString("last_name"));
119     System.out.println(json.getString("job_id"));
120     System.out.println(json.getInt("salary"));
121 }
```

JsonPath is used easily to identify, navigate and manipulate JSON data.

# Authentication VS



## Authorization

- **authentication** - *who are you?*
- **authorization** - *what rights do you have?*

# Authentication



**API KEY** = *one type authentication, we get it from the service provider (sent to account after signing up etc) and we include the key for all our requests as a parameter:*

```
given().queryParams("apikey", "a9faab96").
```

**BASIC AUTHENTICATION** = *using user name and password for authentication*

1. **Challenged basic authentication** —> *rest assured will not send username/password initially. It will only be send once server asks for it.*

```
given().auth().basic("username", "password").
```

2. **Preemptive basic authentication** —> *rest assured sends username/password before server asks for it.*

```
given().auth().preemptive().basic("username", "password").
```



# Authorization



**OAUTH Authorization** = *keys and tokens from 3rd party are used for authentication. To get token we are calling get("sign") end point. It requires username & password. API recognizes credentials and returns token.*

1. *OAuth1*
2. *OAuth2*

```
Response response = given().  
    param("email", "valid_email").  
    param("password", "valid_password").  
    when().get(endpoint); -> to pass credentials
```

```
String accessToken = response.path().get("accessToken"); -> to get token
```

```
given().header("Authorization", token). -> to use token
```

# HOW TO CONVERT JSON TO JAVA OBJECT

- First in first I add **GSON** dependency in to the pom.xml file

```
<dependency>  
    <groupId>com.google.code.gson</groupId>  
    <artifactId>gson</artifactId>  
    <version>2.8.2</version>  
</dependency>
```

- **GSON** -> is a json parser that is used to convert
  - from **java object** to **json** (serialization)
  - from **json** to **java object** (deserialization)

# DE-SERIALIZATION / SERIALIZATION:



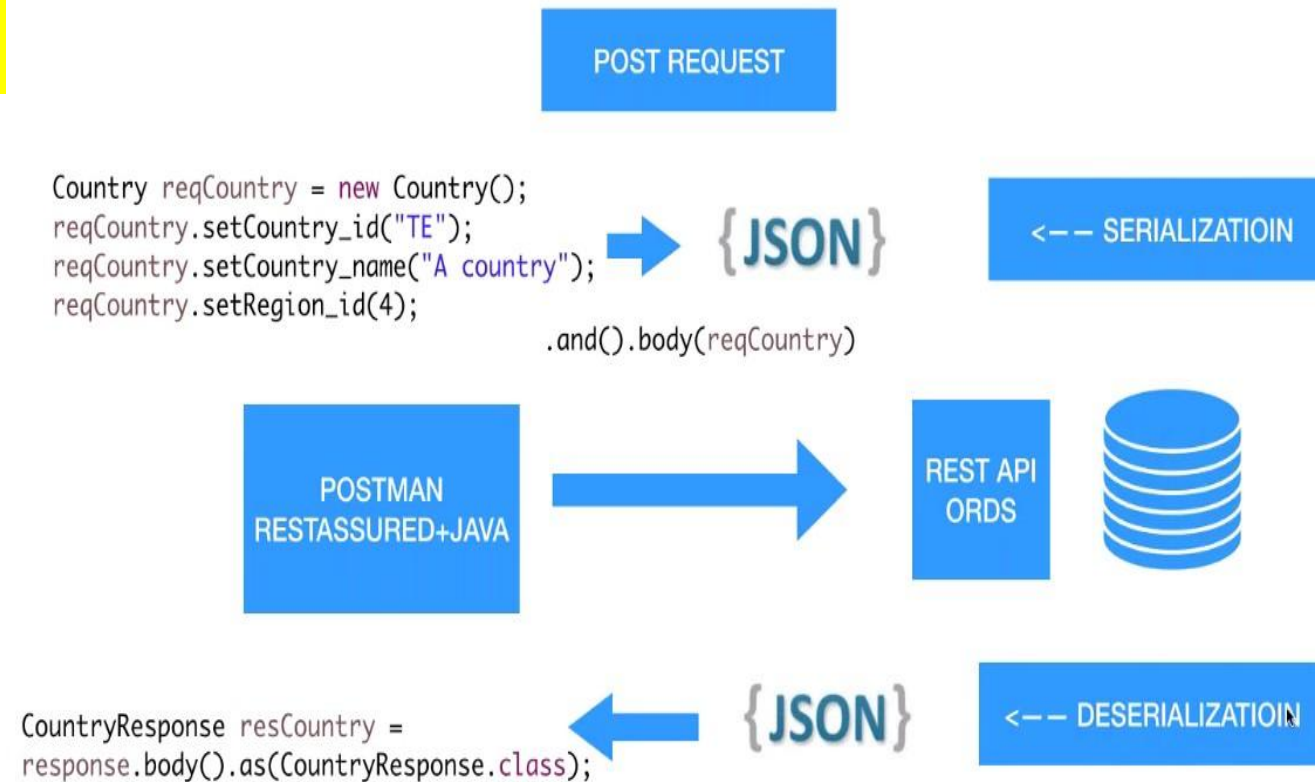
- DE-SERIALIZATION: CONVERT JSON -> JAVA OBJECT  
JSON TO MAP

```
17 public class APIDay3_GSON {
18
19     @Test
20     public void testWithJsonToHashMap() {
21         Response response = given().accept(ContentType.JSON)
22             .when().get(ConfigurationReader.getProperty("hrapp.baseresturl")+"/employees/1
23
24         Map<String,String> map =response.as(HashMap.class);
25
26         System.out.println(map.keySet());
27         System.out.println(map.values());
28
29         assertEquals(map.get("employee_id"),120);
30         assertEquals(map.get("job_id"), "AC_MGR");
31
32
```

# SERIALIZATION/ DE-SERIALIZATION



- SERIALIZATION: CONVERT JAVA OBJECT -> JSON



DE-SERIALIZATION: CONVERT JSON -> to JAVA Object



## Serialization (JAVA to JSON) and Deserialization (JSON to JAVA format)

```

@Test
public void gsonExample() {

    Gson gson = new Gson();

    //Serialization:
    Customer customer = new Customer(20, "Vlad", "male", 7033964165L);
    customer = {id=20, name='Vlad', gender='male', phone:7033964165} //java
    String jsonCustomer = gson.toJson(customer);
    System.out.println("to json format - Serialization: " + jsonCustomer); //json
    //to json format-Serialization:{"id":20,"name":"Vlad","gender":"male",
    "phone":7033964165} format
    //De-Serialization:

    String myJson = "{\"id\":25,\"name\":\"Roman\",\"gender\":\"male\",\"phone\":5712223366}"; //json
    format

    Customer javaCustomer = gson.fromJson(myJson, Customer.class);
    System.out.println("to java format - Deserialization: " + javaCustomer.toString());

    //to java format-Deserialization: {id=25, name='Roman', gender='male', phone=5712223366} //java
    format

    //fromJson(String json, Which.class) --> it will convert the json to object of the class
    //toJson(java object) -> it will take the java object and create json and return it

```



# Can All API endpoints use all of the Http protocols?

- It depends, API developer decides if that url works with GET,POST,PUT, or DELETE requests



# Difference between SOAP and RESTful web services?

- RESTful supports JSON, XML,
- SOAP supports only XML
- REST is faster than SOAP based web services
- SOAP is more secure
- REST is getting more popular



# End to End Testing Scenarios: UI, API, DB

- End to End Testing -> Involving Functionality  
End to End Testing -> Involving Functionality Plus Each Layer Of Application
- 1) Go to UI -> Add An Employee
    - 1) Go to DB and verify if employee is added and all data is matching
    - 2) API -> GET request and verify if employee is added successfully and all data is matching  
-> makes changes in front end and verify in database and REST API.
  - 2) Go to UI -> add an employee:  
check in UI search page.
  - 3) POST an employee using REST API:
    - 2) send a GET request with API and verify
    - 3) Go to DB and verify if employee is added successfully and all data is matching
    - 4) Go to front end(website) and verify that data posted is displayed  
-> makes changes using REST API then verify in DB and UI
  - 4) INSERT an employee into database:
    - 1) run select statement in DB and verify what you inserted is there in tables
    - 2) send API GET request and verify JSON is matching data you inserted to DB
    - 3) Go to front end(website) and verify that data inserted to DB is displayed  
-> make changes in DB using SQL and verify in REST API & front end



# Additional notes



- Big companies use database in a very simple way. Because SQL is very slow languages. It doesn't have OOP concept. So, instead of doing manipulation in database, companies would rather to do in Web Services. Because they use Java, Python, C++ in web service layer and those are much faster than SQL.
- JDBC is an API. It enables Java to interact with the Oracle DataBase.
- Applications such as Uber and Waze pay some money to google and get the permission to access google maps API to use in their own application. This communication is provided by API.
- Or Google use other newspapers to publish news on Google news. Google simply connects to these news sources (CNN, Fox, Newyork Times) by using API.



- Issues observed when performing API testing are
  - Stress, performance, and security issues
  - Duplicate or missing functionality
    - Reliability issues
    - Improper messaging
    - Incompatible error handling mechanism
    - Multi-threaded issues



# CHALLENGES IN API TESTING

- Some of the challenges we face while doing API testing are:
  - Selecting proper parameters and its combinations
  - Categorizing the parameters properly
  - Proper call sequencing is required as this may lead to inadequate coverage in testing (Orn:DELETE'den sonra GET kullanirsan 404 gelir)
  - Verifying and validating the output
  - Due to absence of GUI it is quite difficult to provide input values

# Test flows



We need to implement the next test flow if previous flow is success:

- Single-step workflow:** Executing a single API request and checking the response accordingly. Such basic tests are the minimal building blocks we should start with, and there's no reason to continue testing if these tests fail.
- Multi-step workflow with several requests:** For example, we execute a POST request that creates a resource with id and we then use this id to check if this resource is present in the list of elements received by a GET request. Then we use a PATCH endpoint to update new data, and we again invoke a GET request to validate the new data. Finally, we DELETE that resource and use GET again to verify it no longer exists.

```
POST https://raahulshettyacademy.com/maps/api/place/add/json?key=qaclck123

1 {
2   "location": {
3     "lat": -38.383494,
4     "lng": 33.427362
5   },
6   "accuracy": 50,
7   "name": "Frontline house",
8   "phone_number": "(+91) 983 893 3937",
9   "address": "29, side layout, cohen 09",
10  "types": [
11    "shoe park",
12    "shop"
13  ],
14  "website": "http://google.com",
15  "language": "French-IN"
16 }
17

Body Cookies Headers (10) Test Results Status: 200 OK Time: 1171 ms Size: 610 B Save Response

Pretty Raw Preview Visualize JSON

1 {
2   "status": "OK",
3   "place_id": "e638d41f7f81c47b6c70bf845b086480",
4   "scope": "APP",
5   "reference": "b740b746abae016fa6dca1b4ca292012b740b746abae016fa6dca1b4ca292012",
6   "id": "b740b746abae016fa6dca1b4ca292012"
7 }
```

- Combined API and UI test:** This is mostly relevant to manual testing, where we want to ensure data integrity between the UI and API. We execute requests via the API and verify the actions through the UI or vice versa. The purpose of these integrity test flows is to ensure that although the resources are affected via different mechanisms the system still maintains expected integrity and consistent flow.

# My API testing role in my current project



- As you know I am currently working in a loan company that works with small business companies.
- Once one of our customers wants to apply for a loan from my company, he should create an account using the "get started" button and then they should fill out an online form which consists of questions such as firstName, lastName, email, phoneNumber, SSN, etc. All the information is sent to our Database. Our customer can follow and monitor the latest status of his application that is pending, accepted or denied using my loan life cycle application. In terms of the API testing, I create all the required information on behalf of the customer and using the API POST method I send them to the database. I assert that the relevant information should be also visible at the UA part. Then using API GET, PUT, DELETE method I verify that the implementation and end point are working correctly as expected and there are no bugs. For this purpose;
- 1. First, I check the API contract which is Swagger to make sure that end points are correct and to ensure that the implementation is working as specified according to API documentation (Swagger).
- 2. Then I execute my API test via the POSTMAN;
- 3. I write my Test Cases in the feature file using the Gherkin language.
- I mostly create the following *test case* groups:
  - a. Basic positive test
  - b. Extended positive testing with optional parameters and extra functionality.
  - c. Negative testing with valid input (trying to add an existing username)
  - d. Negative testing with invalid input (trying to add a username which is null)
  - e. Destructive testing (sending null, empty string, integer or other types, odd date format, deleting necessary parameters)
  - f. Security, authorization, and permission tests (sending valid or invalid access tokens to permitted or unpermitted endpoints)
- 4. For each API request I need to verify that;
  - a. I check the request and response body whether those are as written on API documentation in terms of data type and data structure.
  - b. I check HTTP status code. For example, creating a resource should return 201 CREATED and unpermitted requests should return 403 FORBIDDEN, etc.
  - c. I check Response headers. HTTP server headers have implications on both security and performance.
  - d. I check Response body. I check valid JSON body and correct field names, types, and values, including in error responses.
  - e. I check Authorization checks. I check authentication and authorization
  - f. I check Error messages. I Check the error code coverage in case API returns any error
  - g. I check Respo<sub>n</sub>se time.