# Full Stack QA

## Week 1
## Introduction

- SDLC
- Software Testing
- Testing Types
- Test Process
- Test Levels

## Week 2
## Requirements and Agile Methodology

- Requirements Management
- Writing test cases and test scenarios
- Bug Reporting
- Agile Fundamentals
- Whole Team Approach
- Collaborative User Story Creation
- Frequent Feedback

## Week 3
## Manual Testing concepts and JIRA

- Black Box Testing
- Equivalence Partitioning
- Boundary-value Analysis
- Decision Table Testing
- State-Transition Testing
- Creating a JIRA Project
- Creating Components
- Creating User Stories, Epics
- Sprint Planning
- Bug Reporting

## Week 4
## API Testing

- Rest API vs Soup API
- API Methods
- API Authorisation
- JSON
- XML
- Postman
- Rest Assured Introduction

## Week 5
## API Testing and UI Testing

- Rest Assured
  a. GET Method
  b. POST Method
  c. PUT Method
  d. Delete Method
- UI Manual Testing

## Week 6
## UI Testing And JAVA

- Selenium Locators
- Selenium Webdriver
- Selenium Test Cases
- Handling different UI Web Elements
- OOPS

## Week 7
## JAVA and DB

- Access Modifiers
- Conditional Statements
- Data types
- this and super
- JAVA Collections
- Loops
- Relational DB
- No sql

## Week 8
## Test Framework

- TestNG
- TDD
- BDD
- Cucumber
- Test Reports
- Maven

## Week 9
## Performance Testing

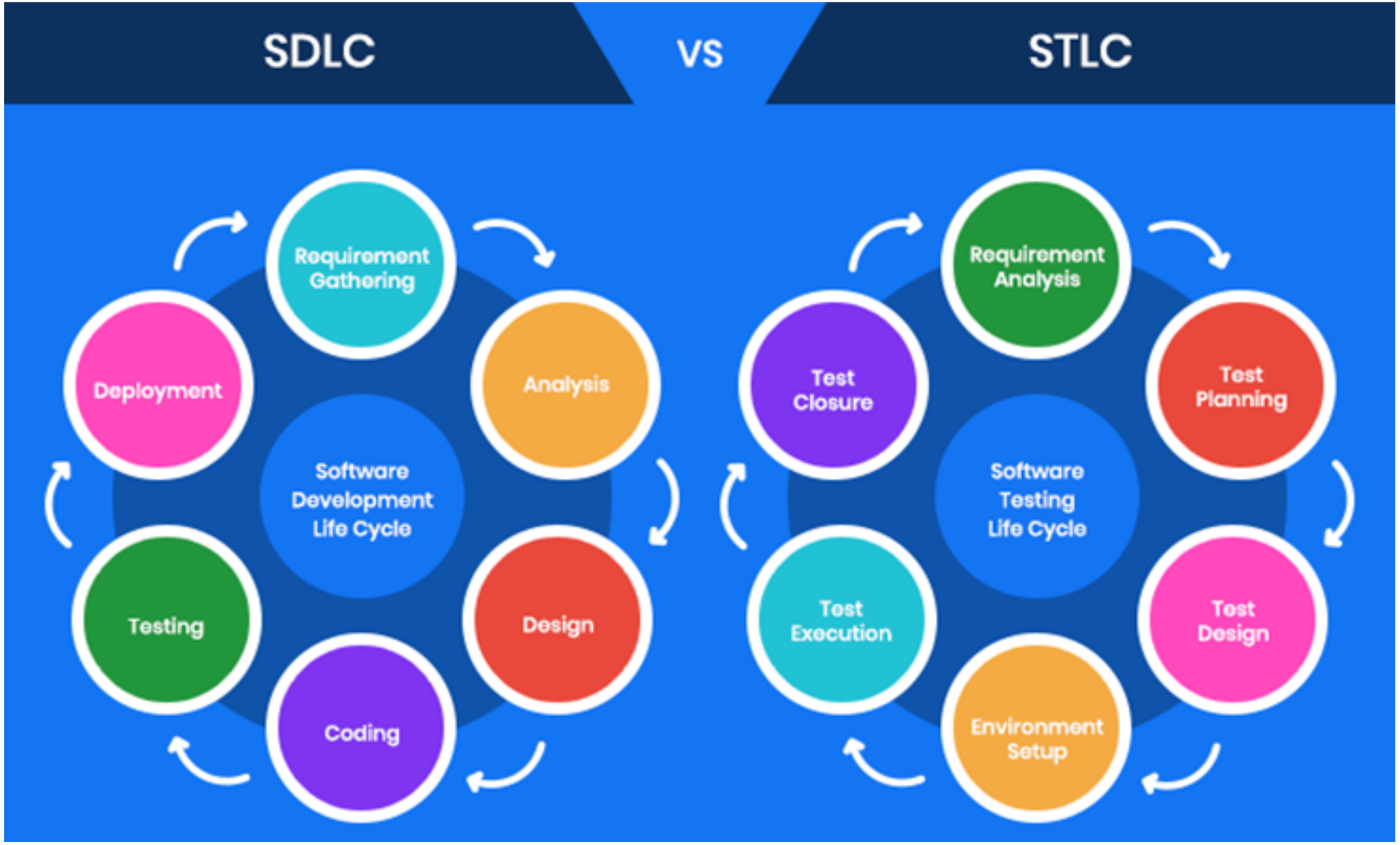
- What is Performance Testing
- Load Profiling
- Types of Performance Testing
- Test Reports
- UI and API Performance Testing

## Week 10
## CI/CD

- Git push , pull, merge
- Working with Jenkins
- Jenkins Pipeline
- Grovvy Scripting

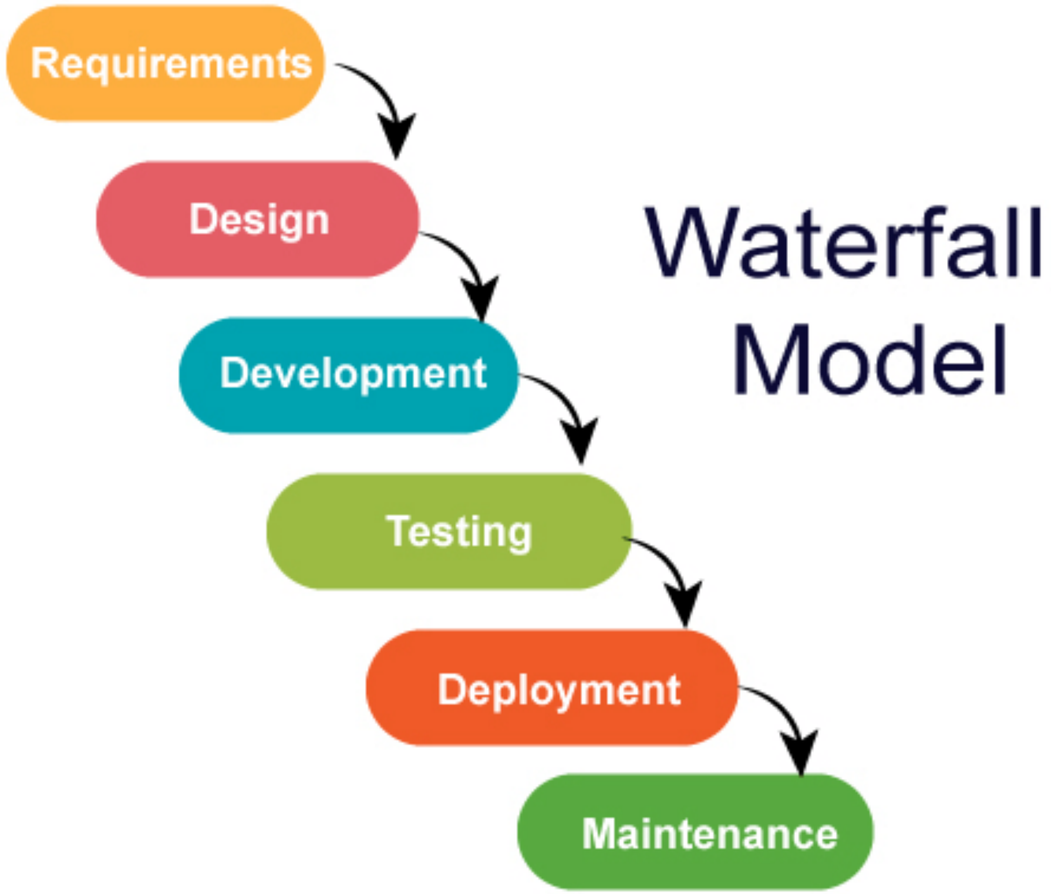
# Why Testing?

Week 1:
Day 1

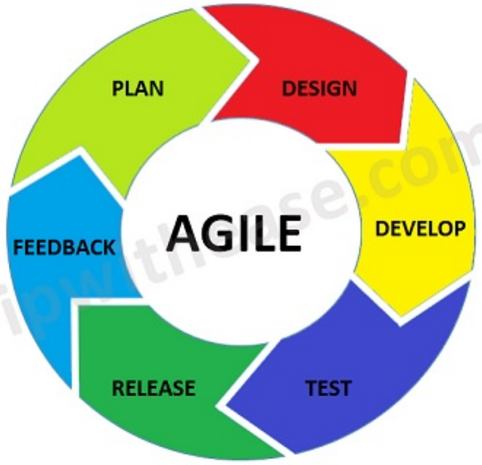



**Assessment**:
STLC Use cases for ATM machine

Week 1:
Day 4


Requirements
Design
Development
Testing
Deployment
Maintenance
Waterfall
Model

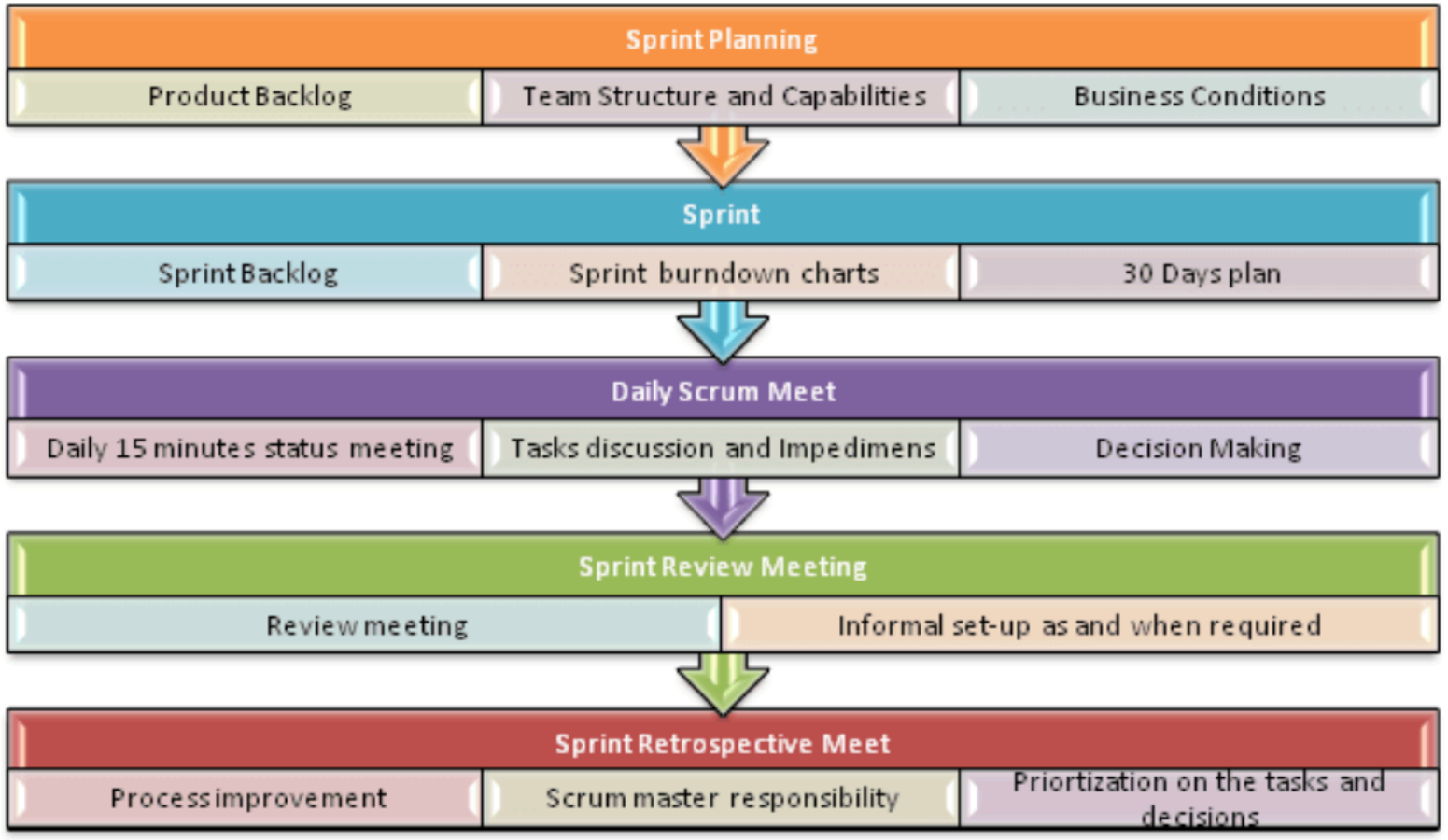
# AGILE METHODOLOGY

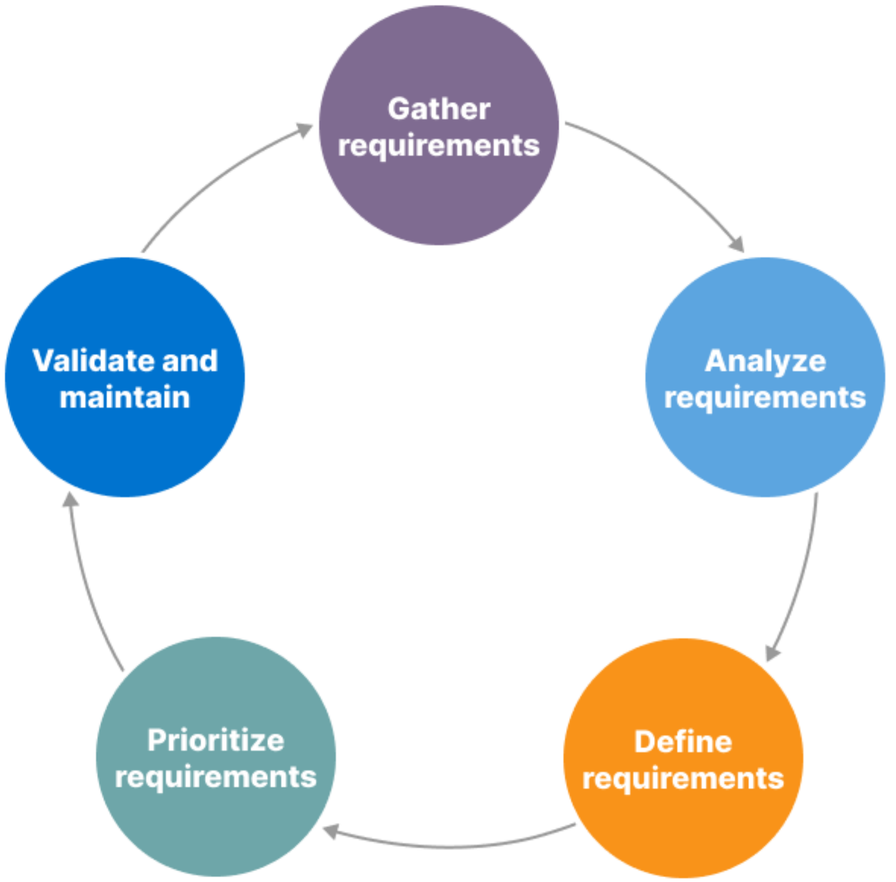



networkinterview.com

(An Initiative By ipwithease.com)







-Functional Testing
  - UI
  - API
- Non Functional Testing
  - Performance
  - Security

1.Manual Testing
2.Automation Testing

- Test Plan
- Test Scenario
- Test cases
- Requirement Traceability Matrix

## Template for Test cases

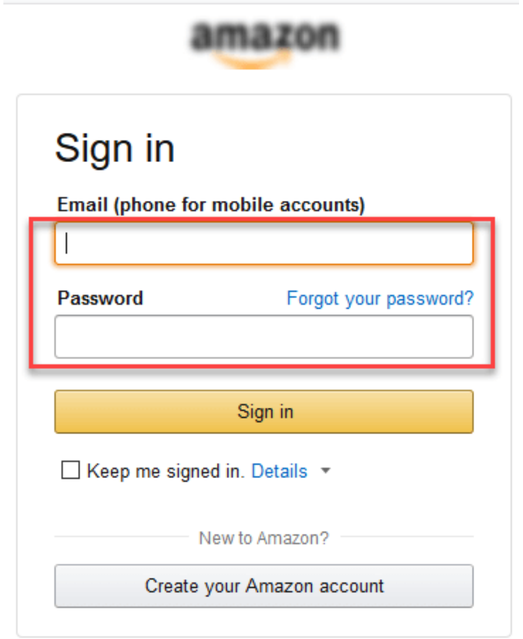
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Test case Name/ Id | | | | | | |
| | Test case Type | | | | | | |
| H | Requirement # | | | | | | |
| E | Module Name | | | | | | |
| A | Status | | | | | | |
| D | Sevarity | | | | | | |
| E | Precondition | | | | | | |
| R | Test Data | | | | | | |
| | Release | | | | | | |
| | Version | | | | | | |
| | Brief Description | | | | | | |
| | Steps | Description | Inputs | Expected Output | Actual Result | Status | Comment |
| B | | | | | | | |
| O | | | | | | | |
| D | | | | | | | |
| Y | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| F | | | | | | | |
| O | Author | | | | | | |
| O | Created Date | | | | | | |
| T | Review By | | | | | | |
| E | Approved By | | | | | | |
| R | | | | | | | |

RTM Template: -

## Requirement Traceability Matrix Template

| Requirement No | Module Name | High Level Requirement | Low Level Requirement | Test Case Name |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**Test Scenario 1:**Check the Login Functionality




1. Check system behavior when valid email id and password is entered.
2. Check system behavior when *invalid* email id and *valid* password is entered.
3. Check system behavior when *valid* email id and *invalid* password is entered.
4. Check system behavior when *invalid* email id and *invalid* password is entered.
5. Check system behavior when email id and password are left blank and Sign in entered.
6. Check Forgot your password is working as expected
7. Check system behavior when valid/invalid phone number and password is entered.
8. Check system behavior when “Keep me signed” is checked

Example 2:  Test Scenarios for a Banking Site Test Scenario
1: Check the Login and Authentication Functionality  Test Scenario
2: Check Money Transfer can be done  Test Scenario
3: Check Account Statement can be viewed  Test Scenario
4: Check Fixed Deposit/Recurring Deposit can be created

Feature -2 : Place a order in Amazon
Feature -3: Add a Friend in Facebook/any social media
Feature -4: Check the upload feature in Google drive
Feature -5 : Checkout a order in amazon
Feature - 6 : Refund a order in amazon

- Boundary Value Analysis (BVA)
- Equivalence Class Partitioning
- Decision Table based testing.
- State Transition
- Error Guessing

- Boundary Value Analysis (BVA)

**Example:**

```
Input condition is valid between 1 to 10

Boundary values 0,1,2 and 9,10,11
```

# Equivalence Class Partitioning

**Example:**
Input conditions are valid between

```
1 to 10 and 20 to 30
```

Hence there are five equivalence classes

```
--- to 0 (invalid)
1 to 10 (valid)
11 to 19 (invalid)
20 to 30 (valid)
31 to --- (invalid)
```

You select values from each class, i.e.,

```
-2, 3, 15, 25, 45
```

### Example 2: Equivalence and Boundary Value

Following password field accepts minimum 6 characters and maximum 10 characters

- Decision Table based testing.

Let’s create a decision table for a login screen.

The condition is simple if the user provides correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.

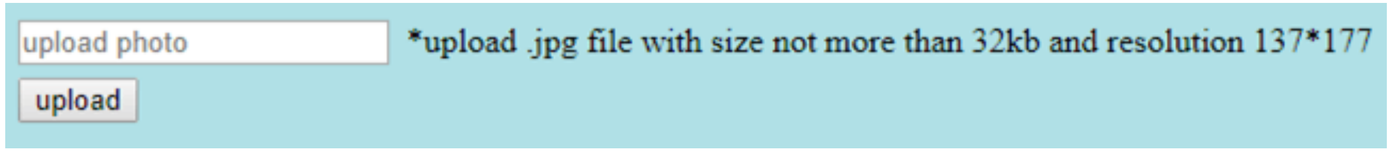
| Conditions | Rule 1 | Rule 2 | Rule 3 | Rule 4 |
|---|---|---|---|---|
| **Username (T/F)** | F | T | F | T |
| **Password (T/F)** | F | F | T | T |
| **Output (E/P)** | E | E | E | P |


**Example 2: How to make Decision Table for Upload Screen**

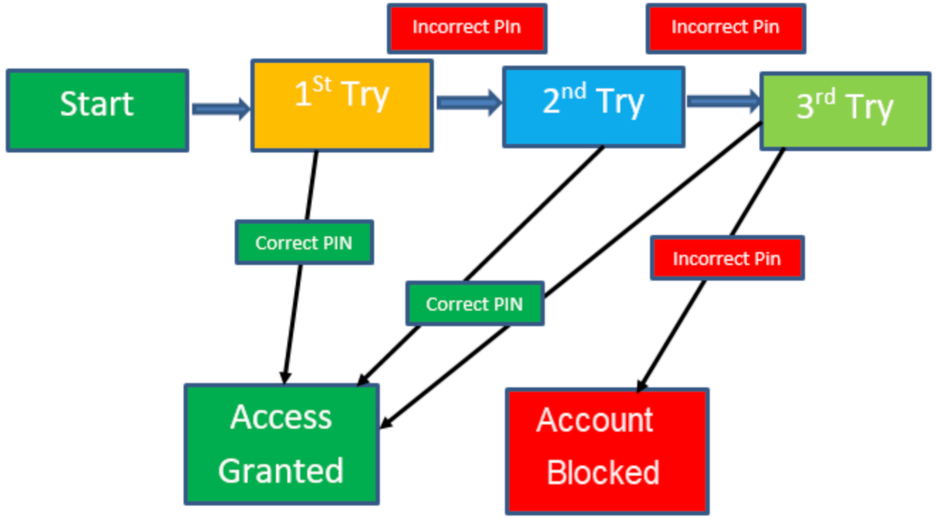
Now consider a dialogue box which will ask the user to upload photo with certain conditions like –

1. You can upload only ‘.jpg’ format image
2. file size less than 32kb
3. resolution 137*177.

If any of the conditions fails the system will throw corresponding error message stating the issue and if all conditions are met photo will be updated successfully



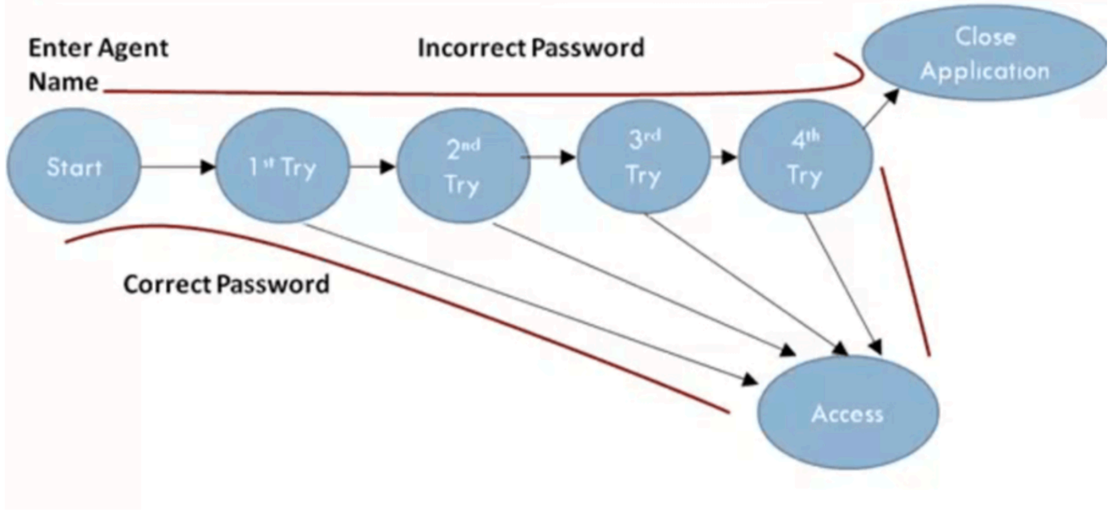
## State transition diagram




### State Transition Table

| | Correct PIN | Incorrect PIN |
|---|---|---|
| S1) Start | S5 | S2 |
| S2) 1st attempt | S5 | S3 |
| S3) 2nd attempt | S5 | S4 |
| S4) 3rd attempt | S5 | S6 |
| S5) Access Granted | – | – |
| S6) Account blocked | – | – |

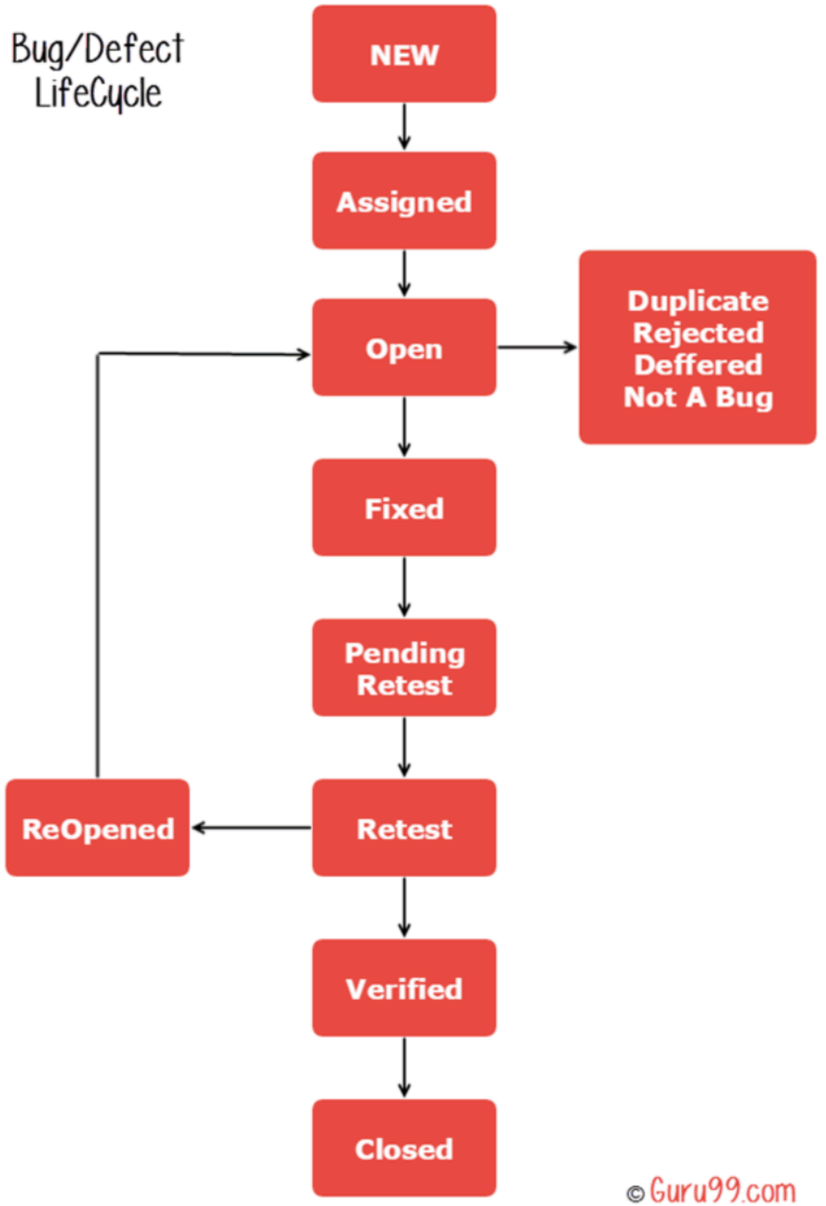
In the flight reservation login screen, consider you have to enter correct agent name and password to access the flight reservation application.



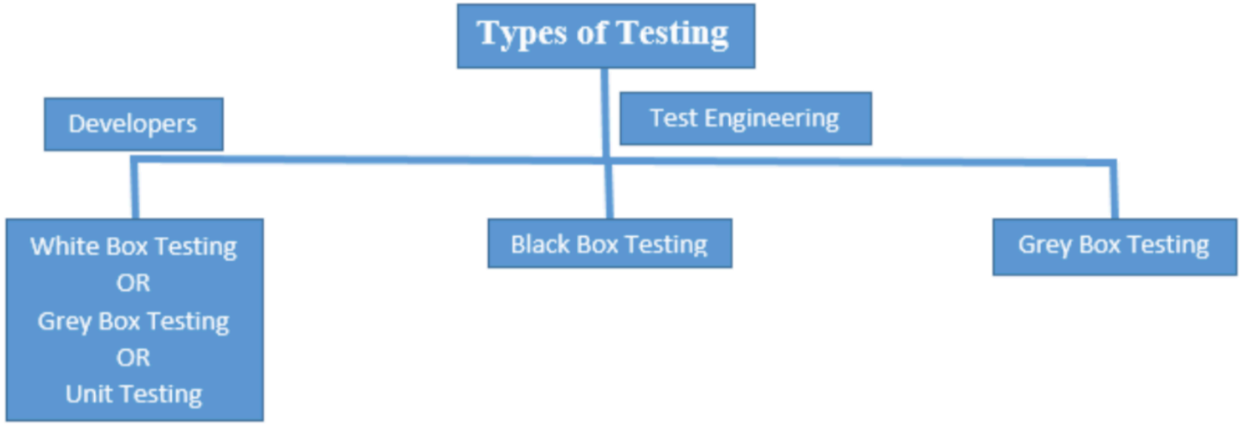

State Transition Graph

Test Data

- **No data**: Check system response when no data is submitted
- **Valid data**: Check system response when Valid test data is submitted
- **Invalid data**: Check system response when *InValid* test data is submitted
- **Illegal data format**: Check system response when test data is in an invalid format
- **Boundary Condition Dataset**: Test data meeting boundary value conditions
- **Equivalence Partition Data Set**: Test data qualifying your equivalence partitions.
- **Decision Table Data Set**: Test data qualifying your decision table testing strategy
- **State Transition Test Data Set**: Test data meeting your state transition testing strategy
- **Use Case Test Data**: Test Data in-sync with your use cases.

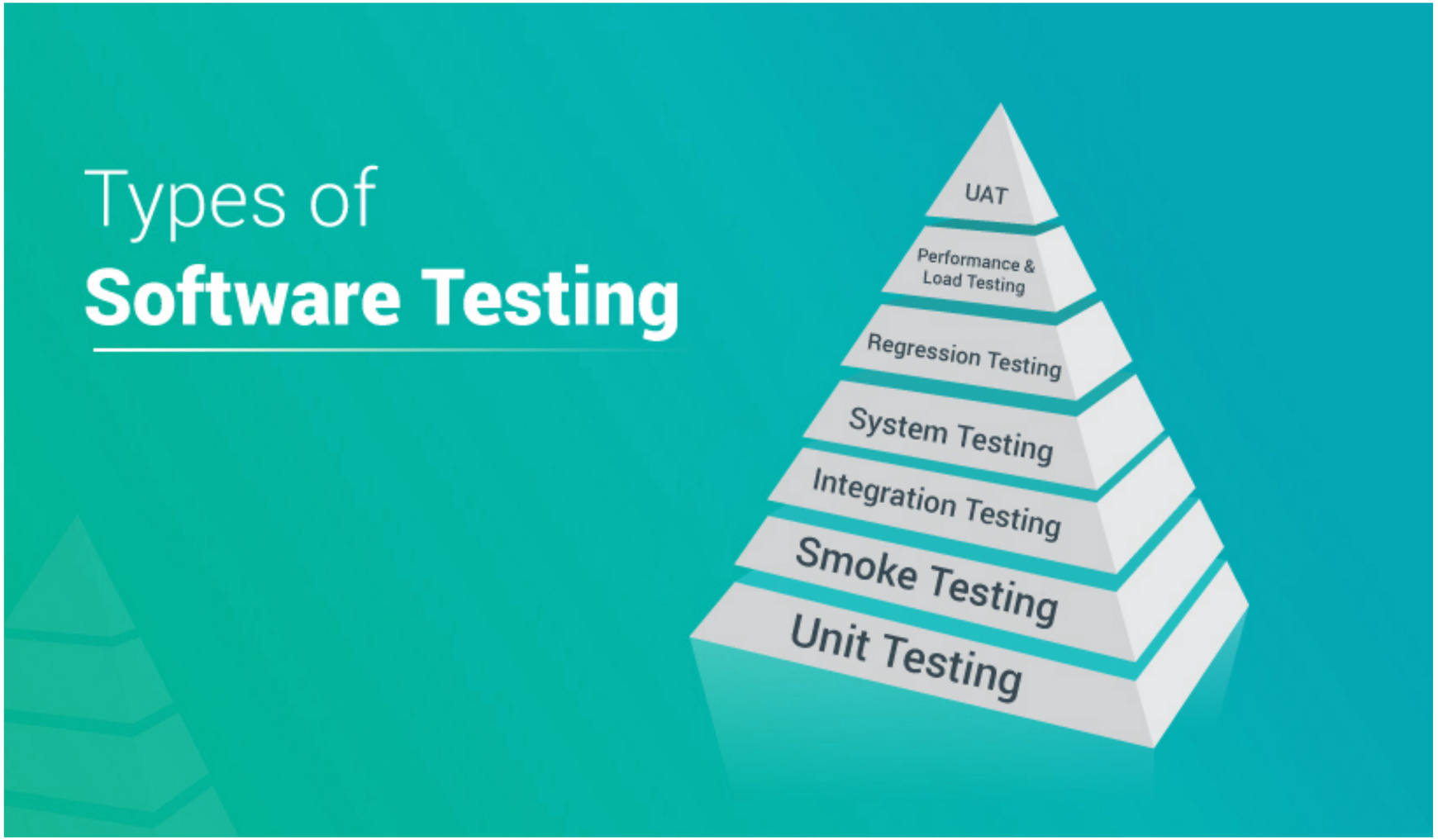

Bug/Defect
LifeCycle
NEW
Assigned
Open
Duplicate
Rejected
Deffered
Not A Bug
Fixed
Pending
Retest
ReOpened
Retest
Verified
Closed

# Types of Testing

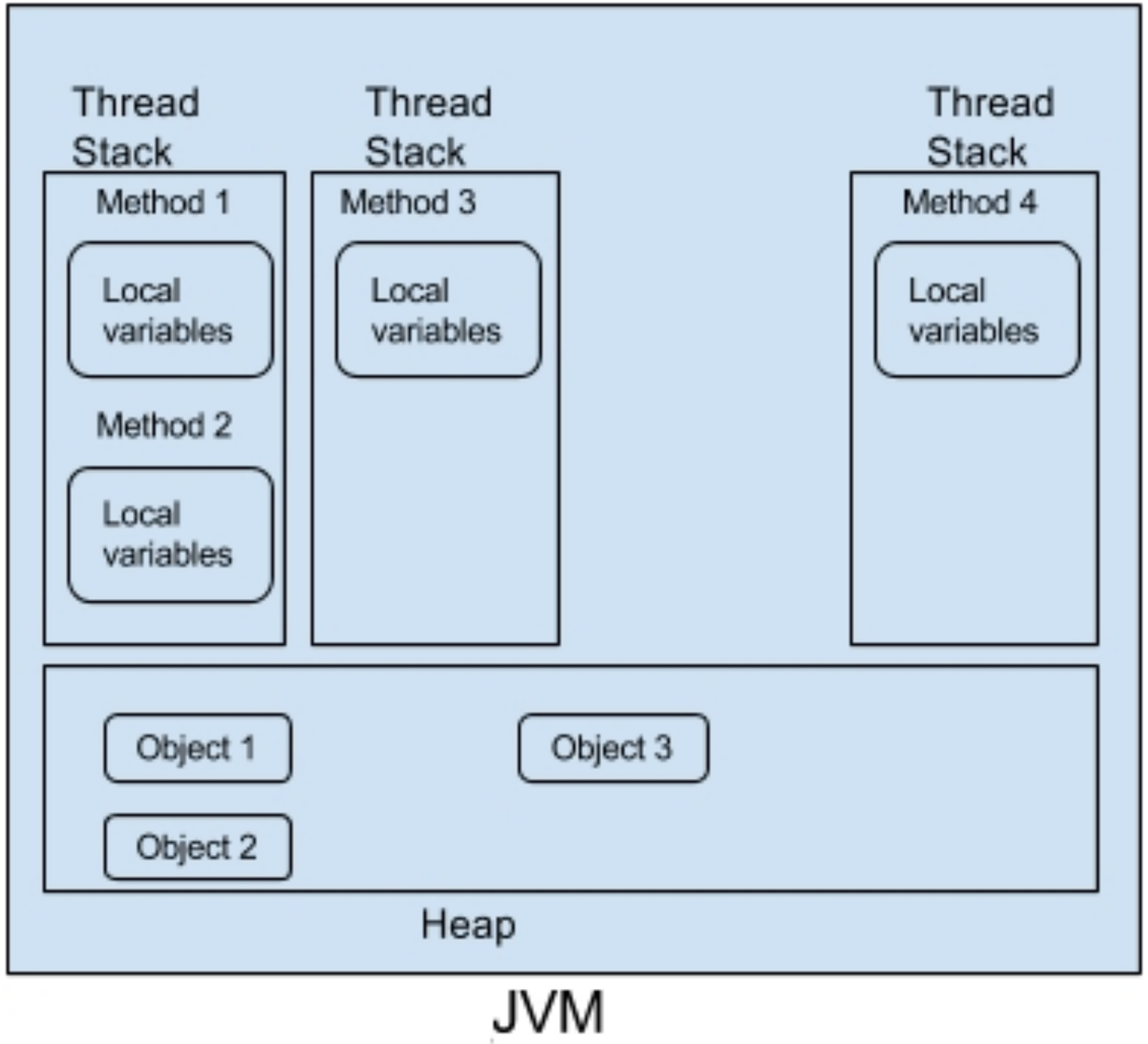



Smoke Testing/Sanity

Week 1:
Day 2 and 3




**Assessment**:
Develop test scenarios for different test levels for Calculator Application


Thread Stack
Thread Stack
Thread Stack
Method 1
Local variables
Method 2
Local variables
Method 3
Local variables
Method 4
Local variables
Object 1
Object 2
Object 3
Heap
JVM

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include String, Classes, Interfaces, and Arrays.

Byte Data Type -> -128 to 127 (inclusive)

```
byte a = 10, byte b = -20
```

Short Data Type -> -32,768 to 32,767

```
short s = 10000, short r = -5000
```

Int Data Type -> - 2,147,483,648 (-2^31) to 2,147,483,647 (2^31 -1) (inclusive)

Long Data Type -> -9,223,372,036,854,775,808(-2^63) to 9,223,372,036,854,775,807(2^63 -1)(inclusive)

```
char letterA = 'A'
```

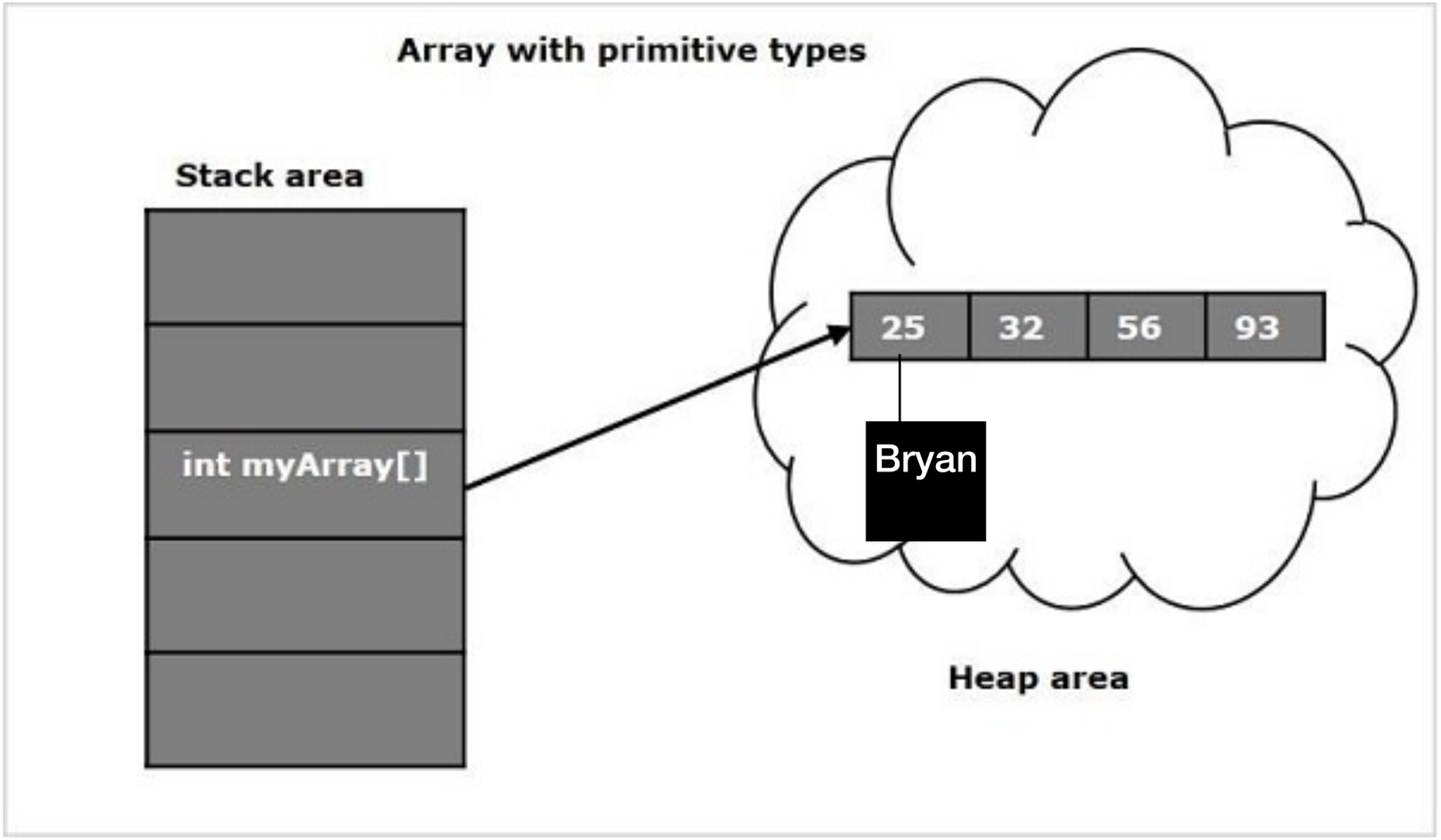
```java
if (1<0 && 1/0 >0){
    System.out.println();
}
```

## Operators in Java

| | | |
|---|---|---|
| Relational | comparison | < > <= >= instanceof |
| | equality | == != |
| Bitwise | bitwise AND | & |
| | bitwise exclusive OR | ^ |
| | bitwise inclusive OR | \| |
| Logical | logical AND | && |
| | logical OR | \|\| |
| Ternary | ternary | ? : |
| Assignment | assignment | = += -= *= /= %= &= ^= \|= <<= >>= >>>= |

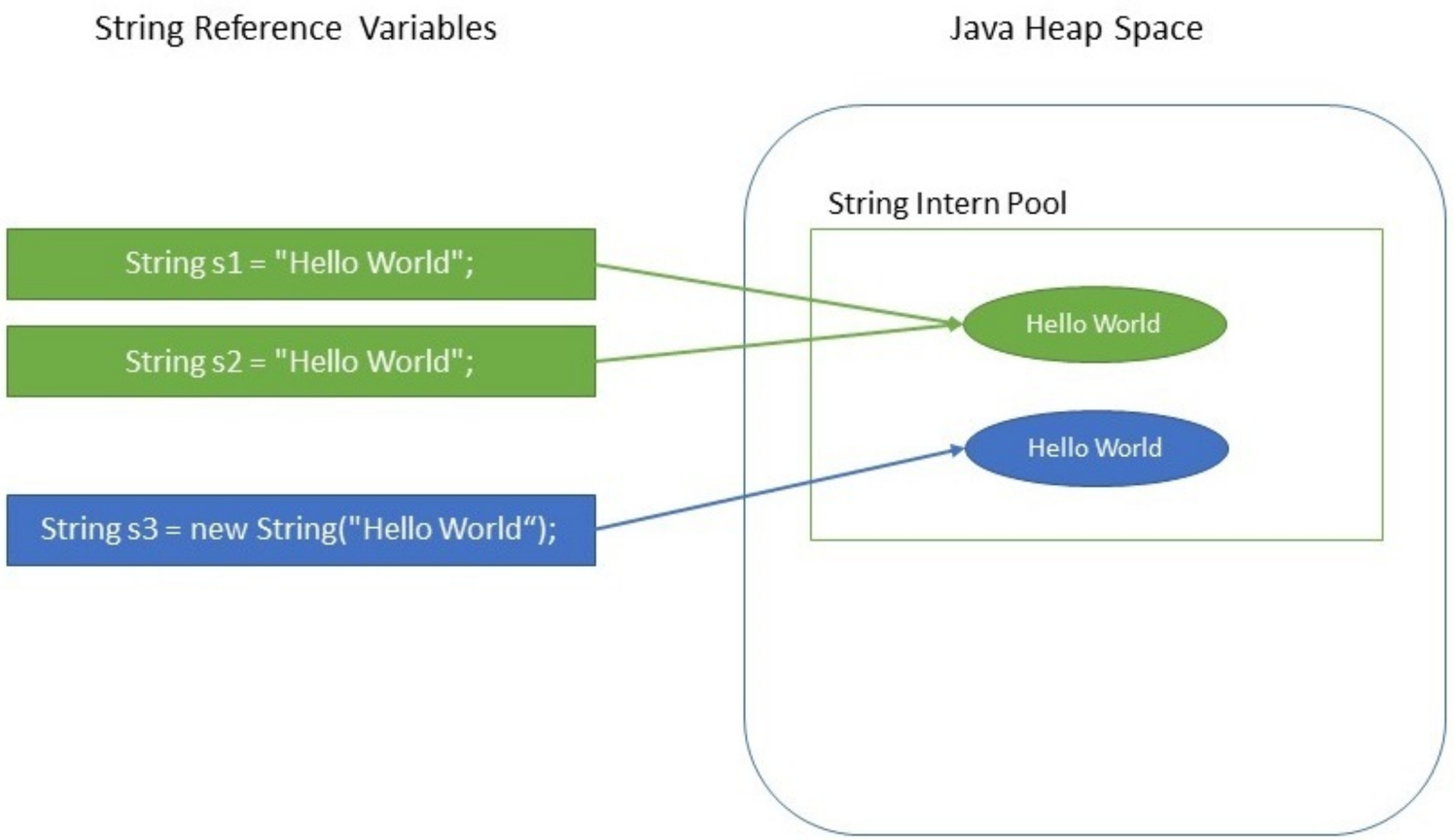
## Function/Method

- Write a function to add two numbers.
-Write a function to find the greatest of  two numbers.


Array with primitive types
Stack area
int myArray[]
25
32
56
93
Bryan
Heap area

# Table 1

| Method | Description | Return Type |
|---|---|---|
| **charAt()** | Returns the character at the specified index (position) | char |
| **compareTo()** | Compares two strings lexicographically | int |
| **compareToIgnoreCase()** | Compares two strings lexicographically, ignoring case differences | int |
| **concat()** | Appends a string to the end of another string | String |
| **contains()** | Checks whether a string contains a sequence of characters | boolean |
| **contentEquals()** | ng contains the exact same sequence of characters of the specified CharSeq | boolean |
| **endsWith()** | Checks whether a string ends with the specified character(s) | boolean |
| **equals()** | Compares two strings. Returns true if the strings are equal, and false if not | boolean |
| **equalsIgnoreCase()** | Compares two strings, ignoring case considerations | boolean |
| **indexOf()** | turns the position of the first found occurrence of specified characters in a str | int |
| **isEmpty()** | Checks whether a string is empty or not | boolean |
| **lastIndexOf()** | turns the position of the last found occurrence of specified characters in a str | int |
| **length()** | Returns the length of a specified string | int |
| **replace()** | tring for a specified value, and returns a new string where the specified values | String |
| **replaceFirst()** | occurrence of a substring that matches the given regular expression with the | String |
| **replaceAll()** | substring of this string that matches the given regular expression with the giv | String |
| **split()** | Splits a string into an array of substrings | String[] |
| **startsWith()** | Checks whether a string starts with specified characters | boolean |
| **substring()** | Returns a new string which is the substring of a specified string | String |
| **toCharArray()** | Converts this string to a new character array | char[] |
| **toLowerCase()** | Converts a string to lower case letters | String |
| **toString()** | Returns the value of a String object | String |
| **toUpperCase()** | Converts a string to upper case letters | String |
| **trim()** | Removes whitespace from both ends of a string | String |




```
1. public static void main(String args[]){
2.   String s="Sachin";
3.   s.concat(" Tendulkar");//concat() method appends the string at the end
4.   System.out.println(s);//will print Sachin because strings are immutable objects
5. }
```

-Write a Java program to get the character at the given index within the String
Original String = Java Exercises!
The character at position 0 is J
The character at position 10 is i

-Write a Java program to compare two strings lexicographically
String 1: This is Exercise 1
String 2: This is Exercise 2
"This is Exercise 1" is less than "This is Exercise 2"

-Write a Java program to compare two strings lexicographically, ignoring case differences
String 1: This is exercise 1
String 2: This is Exercise 1
"This is exercise 1" is equal to "This is Exercise 1"

-Write a Java program to concatenate a given string to the end of another string
String 1: PHP Exercises and
String 2: Python Exercises
The concatenated string: PHP Exercises and Python Exercises

-

Java provides three types of control flow statements.

1. Decision Making statements
   - if statements [ **int** min=(a<b)?a:b;  ]
   - switch statement
2. Loop statements
   - do while loop
   - while loop
   - for loop
   - for-each loop
3. Jump statements
   - break statement
   - continue statement

1.Simple if statement
2.if-else statement
3.if-else-if ladder
4.Nested if-statement

```
class Main {
public static void main(String[] args) {

int number = 0;

// checks if number is greater than 0
if (number > 0) {
System.out.println("The number is positive.");
}

// checks if number is less than 0
else if (number < 0) {
System.out.println("The number is negative.");
}

// if both condition is false
else {
System.out.println("The number is 0.");
}
}
}
```

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

```
class IfStatement {
  public static void main(String[] args) {
   int number = 10;      // checks if number is less than 0
    if (number < 0) {
          System.out.println("The number is negative.");
        }

 System.out.println("Statement outside if block");
    }
}
```

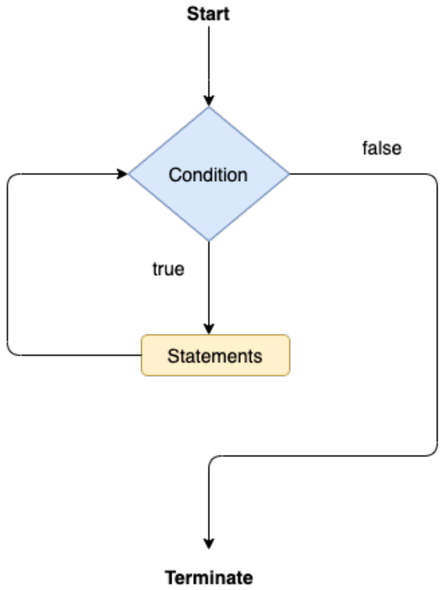
```
class Main {
   public static void main(String[] args) {
      int number = 10;

      // checks if number is greater than 0
      if (number > 0) {
         System.out.println("The number is positive.");
      }

      // execute this block
      // if number is not greater than 0
      else {
         System.out.println("The number is not positive.");
      }

      System.out.println("Statement outside if...else block");
   }
}
```

# Switch Statement:

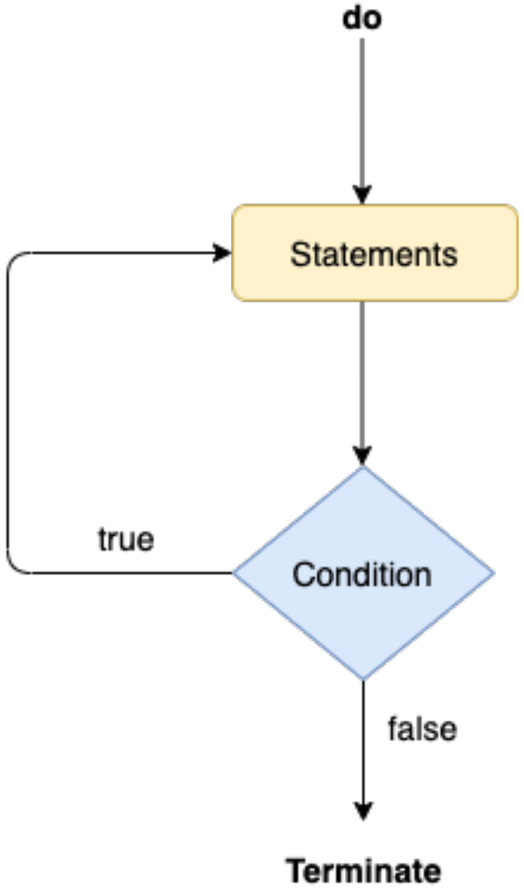
```
1.public static void main(String[] args) {
2.int num = 2;
3.switch (num){
4.case 0:
5.System.out.println("number is 0");
6.break;
7.case 1:
8.System.out.println("number is 1");
9.break;
10.default:
11.System.out.println(num);
12.}
13.}
14.
```

1.for loop
2.while loop

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```




3.do-while loop



# Java Break and Continue

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  System.out.println(i);
}
```

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  System.out.println(i);
}
```

- Write a program to  assign grades (A, B, C) based on the percentage obtained by a student.
  - if the percentage is above **90**, assign grade **A**
  - if the percentage is above **75**, assign grade **B**
  - if the percentage is above **65**, assign grade **C**

- Write a program to print the maximum of the given three numbers 100,50,200
- Java Program to check if number is even or odd
- Java Program to check if number is positive or negative

- Write a program to print values from 1 to 10.
- Write a program to print values from 100 to 1.
- Write a program to print the values from an array.
- Write a program to print the values from an array in reverse order.
- Display a Text Five Times
- Display Sum of n Natural Numbers

Java Type Casting

Type casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of casting:

- **Widening Casting** (automatically) - converting a smaller type to a larger type size
- `byte` -> `short` -> `char` -> `int` -> `long` -> `float` -> `double`

- **Narrowing Casting** (manually) - converting a larger type to a smaller size type
- `double` -> `float` -> `long` -> `int` -> `char` -> `short` -> `byte`

```
public class Main {
  public static void main(String[] args) {
    int myInt = 9;
    double myDouble = myInt; // Automatic casting: int to double

    System.out.println(myInt);      // Outputs 9
    System.out.println(myDouble);   // Outputs 9.0
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    double myDouble = 9.78d;
    int myInt = (int) myDouble; // Manual casting: double to int

    System.out.println(myDouble);   // Outputs 9.78
    System.out.println(myInt);      // Outputs 9
  }
}
```

Math.max(*x*,*y*)

Math.min(*x*,*y*)

Math.sqrt(*x*)

Math.abs(*x*)

`Math.random()`

# Java Method Parameters

**Return Statement**

```
public class Main {
  static int myMethod(int x) {
    return 5 + x;
  }

  public static void main(String[] args) {
    System.out.println(myMethod(3));
  }
}
```

```
public class Main {
  static int myMethod(int x, int y) {
    return x + y;
  }

  public static void main(String[] args) {
    System.out.println(myMethod(5, 3));
  }
}
```

```
public class Main {
  static int myMethod(int x, int y) {
    return x + y;
  }

  public static void main(String[] args) {
    int z = myMethod(5, 3);
    System.out.println(z);
  }
}
```

# Java Scope

```
public class Main {
  public static void main(String[] args) {

    // Code here CANNOT use x

    int x = 100;

    // Code here can use x
    System.out.println(x);
  }
}
```

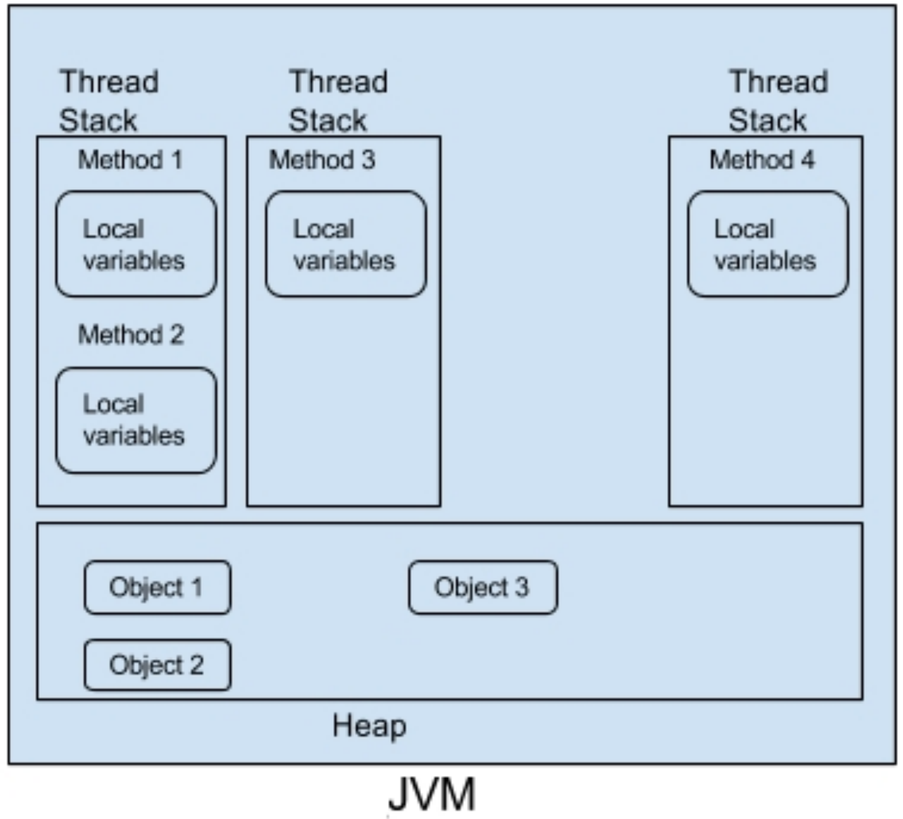
```
public class Main {
    int x = 100;
  public static void main(String[] args) {
    // Code here can use x
    System.out.println(x);
  }

public static void function() {
    // Code here can use x
    System.out.println(x);
  }

}
```

# Java OOP

| class | objects |
| --- | --- |
| Fruit | Apple |
| | Banana |
| | Mango |

Another example:

| class | objects |
| --- | --- |
| Car | Volvo |
| | Audi |
| | Toyota |




So, a class is a template for objects, and an object is an instance of a class.
When the individual objects are created, they inherit all the variables and methods from the class.

# Accessing Attributes

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj1 = new Main();  // Object 1
    Main myObj2 = new Main();  // Object 2
    System.out.println(myObj1.x);
    System.out.println(myObj2.x);
  }
}
```

## Multiple Classes

Main.java

```
public class Main {
  int x = 5;
}
```

Second.java

```
class Second {
  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

# Modify Attributes

```
public class Main {
  int x;

  public static void main(String[] args) {
    Main myObj = new Main();
    myObj.x = 40;
    System.out.println(myObj.x);
  }
}
```

# Multiple Objects

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj1 = new Main();  // Object 1
    Main myObj2 = new Main();  // Object 2
    myObj2.x = 25;
    System.out.println(myObj1.x);  // Outputs 5
    System.out.println(myObj2.x);  // Outputs 25
  }
}
```

# Multiple Attributes

```
public class Main {
  String fname = "John";
  String lname = "Doe";
  int age = 24;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println("Name: " + myObj.fname + " " + myObj.lname);
    System.out.println("Age: " + myObj.age);
  }
}
```

# Java Class Methods

## Static vs. Non-Static

```
public class Main {
  // Static method
  static void myStaticMethod() {
    System.out.println("Static methods can be called without creating objects");
  }

  // Public method
  public void myPublicMethod() {
    System.out.println("Public methods must be called by creating objects");
  }

  // Main method
  public static void main(String[] args) {
    myStaticMethod(); // Call the static method
    // myPublicMethod(); This would compile an error

    Main myObj = new Main(); // Create an object of Main
    myObj.myPublicMethod(); // Call the public method on the object
  }
}
```

# Java Constructors

```
// Create a Main class
public class Main {
  int x;  // Create a class attribute

  // Create a class constructor for the Main class
  public Main() {
    x = 5;  // Set the initial value for the class attribute x
  }

  public static void main(String[] args) {
    Main myObj = new Main(); // Create an object of class Main (This will call the constructor)
    System.out.println(myObj.x); // Print the value of x
  }
}

// Outputs 5
```

## Constructor Parameters

```
public class Main {
  int x;

  public Main(int y) {
    x = y;
  }

  public static void main(String[] args) {
    Main myObj = new Main(5);
    System.out.println(myObj.x);
  }
}
```

1.
Write a program to print the area of a rectangle by creating a class named 'Area' having two methods. First method named as 'setDim' takes length and breadth of rectangle as parameters and the second method named as 'getArea' returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.

2.
Create a class named 'Student' with String variable 'name' and integer variable 'roll_no'. Assign the value of roll_no as '2' and that of name as "John" by creating an object of the class Student.

3.
Assign and print the roll number, phone number and address of two students having names "Sam" and "John" respectively by creating two objects of class 'Student'.

4.
Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' without any parameter in its constructor.

5.
Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' with constructor having the three sides as its parameters.

6.
Write a program to print the area of two rectangles having sides (4,5) and (5,8) respectively by creating a class named 'Rectangle' with a method named 'Area' which returns the area and length and breadth passed as parameters to its constructor.

7.
Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a method named 'returnArea' which returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.

8.
Print the average of three numbers entered by user by creating a class named 'Average' having a method to calculate and print the average.

9.
Print the sum, difference and product of two complex numbers by creating a class named 'Complex' with separate methods for each operation whose real and imaginary parts are entered by user.

10.
Write a program that would print the information (name, year of joining, salary, address) of three employees by creating a class named 'Employee'. The output should be as follows:

| Name | Year of joining | Address |
|---|---|---|
| Robert | 1994 | 64C- WallsStreat |
| Sam | 2000 | 68D- WallsStreat |
| John | 1999 | 26B- WallsStreat |

11.
Add two distances in inch-feet by creating a class named 'AddDistance'.

# Access Modifiers

| Modifier | Description |
| --- | --- |
| `public` | The code is accessible for all classes |
| `private` | The code is only accessible within the declared class |
| *default* | The code is only accessible in the same package. This is used when you don't specify a modifier. You will learn more about packages in the Packages chapter |
| `protected` | The code is accessible in the same package and **subclasses**. You will learn more about subclasses and superclasses in the Inheritance chapter |

# Non-Access Modifiers

| Modifier | Description |
| --- | --- |
| `final` | The class cannot be inherited by other classes (You will learn more about inheritance in the Inheritance chapter) |
| `abstract` | The class cannot be used to create objects (To access an abstract class, it must be inherited from another class. You will learn more about inheritance and abstraction in the Inheritance and Abstraction chapters) |

# Java Encapsulation

The meaning of **Encapsulation**, is to make sure that "sensitive" data is hidden from users. To achieve this, you must:

- declare class variables/attributes as `private`
- provide public **get** and **set** methods to access and update the value of a `private` variable

```
public class Person {
  private String name; // private = restricted access

  // Getter
  public String getName() {
    return name;
  }

  // Setter
  public void setName(String newName) {
    this.name = newName;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.setName("John"); // Set the value of the name variable to "John"
    System.out.println(myObj.getName());
  }
}

// Outputs "John"
```

# Java Packages

```
import package.name.Class;   // Import a single class
import package.name.*;   // Import the whole package
```

```
import java.util.Scanner;

class MyClass {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);
    System.out.println("Enter username");

    String userName = myObj.nextLine();
    System.out.println("Username is: " + userName);
  }
}
```

# Java Inheritance

In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

- **subclass** (child) - the class that inherits from another class
- **superclass** (parent) - the class being inherited from

```
class Vehicle {
  protected String brand = "Ford";        // Vehicle attribute
  public void honk() {                    // Vehicle method
    System.out.println("Tuut, tuut!");
  }
}

class Car extends Vehicle {
  private String modelName = "Mustang";    // Car attribute
  public static void main(String[] args) {

    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (from the Vehicle class) on the myCar object
    myCar.honk();

    // Display the value of the brand attribute (from the Vehicle class) and the value of the modelName from the Car class
    System.out.println(myCar.brand + " " + myCar.modelName);
  }
}
```

## The final Keyword

```
final class Vehicle {
  ...
}

class Car extends Vehicle {
  ...
}
```

# Java Polymorphism

**Polymorphism** uses those methods to perform different tasks. This allows us to perform a single action in different ways.

```
class Animal {
  public void animalSound() {
    System.out.println("The animal makes a sound");
  }
}

class Pig extends Animal {
  public void animalSound() {
    System.out.println("The pig says: wee wee");
  }
}

class Dog extends Animal {
  public void animalSound() {
    System.out.println("The dog says: bow wow");
  }
}

class Main {
  public static void main(String[] args) {
    Animal myAnimal = new Animal();  // Create a Animal object
    Animal myPig = new Pig();  // Create a Pig object
    Animal myDog = new Dog();  // Create a Dog object
    myAnimal.animalSound();
    myPig.animalSound();
    myDog.animalSound();
  }
}
```

# Java Abstraction

Data **abstraction** is the process of hiding certain details and showing only essential information to the user.
Abstraction can be achieved with either **abstract classes** or **interfaces** (which you will learn more about in the next chapter).
The `abstract` keyword is a non-access modifier, used for classes and methods:

- **Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- **Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

An abstract class can have both abstract and regular methods:

```
// Abstract class
abstract class Animal {
  // Abstract method (does not have a body)
  public abstract void animalSound();
  // Regular method
  public void sleep() {
    System.out.println("Zzz");
  }
}

// Subclass (inherit from Animal)
class Pig extends Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig(); // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

# Java Interface

An `interface` is a completely "**abstract class**" that is used to group related methods with empty bodies:

```
// interface
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void run(); // interface method (does not have a body)
}
```

To access the interface methods, the interface must be "implemented" (kinda like inherited) by another class with the `implements` keyword (instead of `extends`).
The body of the interface method is provided by the "implement" class:

```
// Interface
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void sleep(); // interface method (does not have a body)
}

// Pig "implements" the Animal interface
class Pig implements Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
  public void sleep() {
    // The body of sleep() is provided here
    System.out.println("Zzz");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig();  // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

Notes on Interfaces:

- Like **abstract classes**, interfaces **cannot** be used to create objects (in the example above, it is not possible to create an "Animal" object in the MyMainClass)
- Interface methods do not have a body - the body is provided by the "implement" class
- On implementation of an interface, you must override all of its methods
- Interface methods are by default `abstract` and `public`
- Interface attributes are by default `public`, `static` and `final`
- An interface cannot contain a constructor (as it cannot be used to create objects)

Why And When To Use Interfaces?

1) To achieve security - hide certain details and only show the important details of an object (interface).
2) Java does not support "multiple inheritance" (a class can only inherit from one superclass). However, it can be achieved with interfaces, because the class can **implement** multiple interfaces. **Note:** To implement multiple interfaces, separate them with a comma (see example below).

```
interface FirstInterface {
  public void myMethod(); // interface method
}

interface SecondInterface {
  public void myOtherMethod(); // interface method
}

class DemoClass implements FirstInterface, SecondInterface {
  public void myMethod() {
    System.out.println("Some text..");
  }
  public void myOtherMethod() {
    System.out.println("Some other text...");
  }
}

class Main {
  public static void main(String[] args) {
    DemoClass myObj = new DemoClass();
    myObj.myMethod();
    myObj.myOtherMethod();
  }
}
```

# Java Enums

**Enum** is short for "enumerations", which means "specifically listed".

```
public class Main {
  enum Level {
    LOW,
    MEDIUM,
    HIGH
  }

  public static void main(String[] args) {
    Level myVar = Level.MEDIUM;
    System.out.println(myVar);
  }
}
```

```
enum Level {
  LOW,
  MEDIUM,
  HIGH
}

public class Main {
  public static void main(String[] args) {
    Level myVar = Level.MEDIUM;

    switch(myVar) {
      case LOW:
        System.out.println("Low level");
        break;
      case MEDIUM:
         System.out.println("Medium level");
        break;
      case HIGH:
        System.out.println("High level");
        break;
    }
  }
}
```

```
for (Level myVar : Level.values()) {
  System.out.println(myVar);
}
```

### Difference between Enums and Classes

An `enum` can, just like a `class`, have attributes and methods. The only difference is that enum constants are `public`, `static` and `final` (unchangeable - cannot be overridden).

An `enum` cannot be used to create objects, and it cannot extend other classes (but it can implement interfaces).

### Why And When To Use Enums?

Use enums when you have values that you know aren't going to change, like month days, days, colors, deck of cards, etc.

# Java User Input (Scanner)

```
import java.util.Scanner;  // Import the Scanner class

class Main {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);  // Create a Scanner object
    System.out.println("Enter username");

    String userName = myObj.nextLine();  // Read user input
    System.out.println("Username is: " + userName);  // Output user input
  }
}
```

```
import java.util.Scanner;

class Main {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);

    System.out.println("Enter name, age and salary:");

    // String input
    String name = myObj.nextLine();

    // Numerical input
    int age = myObj.nextInt();
    double salary = myObj.nextDouble();

    // Output input by user
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
    System.out.println("Salary: " + salary);
  }
}
```

**Note:** If you enter wrong input (e.g. text in a numerical input), you will get an exception/error message (like "InputMismatchException").

# Java Date and Time

```
import java.time.LocalDate; // import the LocalDate class

public class Main {
  public static void main(String[] args) {
    LocalDate myObj = LocalDate.now(); // Create a date object
    System.out.println(myObj); // Display the current date
  }
}
```

```
import java.time.LocalTime; // import the LocalTime class

public class Main {
  public static void main(String[] args) {
    LocalTime myObj = LocalTime.now();
    System.out.println(myObj);
  }
}
```

```
import java.time.LocalDateTime; // import the LocalDateTime class

public class Main {
  public static void main(String[] args) {
    LocalDateTime myObj = LocalDateTime.now();
    System.out.println(myObj);
  }
}
```

# Formatting Date and Time

```
import java.time.LocalDateTime; // Import the LocalDateTime class
import java.time.format.DateTimeFormatter; // Import the DateTimeFormatter class

public class Main {
  public static void main(String[] args) {
    LocalDateTime myDateObj = LocalDateTime.now();
    System.out.println("Before formatting: " + myDateObj);
    DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

    String formattedDate = myDateObj.format(myFormatObj);
    System.out.println("After formatting: " + formattedDate);
  }
}
```

| Value | Example |
| --- | --- |
| *yyyy-MM-dd* | "1988-09-29" |
| *dd/MM/yyyy* | "29/09/1988" |
| *dd-MMM-yyyy* | "29-Sep-1988" |
| *E, MMM dd yyyy* | "Thu, Sep 29 1988" |

# Java ArrayList

The `ArrayList` class is a resizable array, which can be found in the `java.util` package.
The difference between a built-in array and an `ArrayList` in Java, is that the size of an array cannot be modified (if you want to add or remove elements to/from an array, you have to create a new one). While elements can be added and removed from an `ArrayList` whenever you want. The syntax is also slightly different:

```
import java.util.ArrayList; // import the ArrayList class

ArrayList<String> cars = new ArrayList<String>(); // Create an ArrayList object
```

## Add Items

```
import java.util.ArrayList;

public class Main {
  public static void main(String[] args) {
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    System.out.println(cars);
  }
}
```

## Access an Item

```
cars.get(0);
```

## Change an Item

```
cars.set(0, "Opel");
```

## Remove an Item

```
cars.remove(0);
```

To remove all the elements in the `ArrayList`, use the `clear()` method:

```
cars.clear();
```

ArrayList Size `cars.size();`

## Loop Through an ArrayList

```
public class Main {
  public static void main(String[] args) {
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    for (int i = 0; i < cars.size(); i++) {
      System.out.println(cars.get(i));
    }
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    for (String i : cars) {
      System.out.println(i);
    }
  }
}
```

## Other Types

Elements in an ArrayList are actually objects. In the examples above, we created elements (objects) of type "String". Remember that a String in Java is an object (not a primitive type). To use other types, such as int, you must specify an equivalent wrapper class: `Integer`. For other primitive types, use: `Boolean` for boolean, `Character` for char, `Double` for double, etc:

Create an `ArrayList` to store numbers (add elements of type `Integer`):

```
import java.util.ArrayList;

public class Main {
  public static void main(String[] args) {
    ArrayList<Integer> myNumbers = new ArrayList<Integer>();
    myNumbers.add(10);
    myNumbers.add(15);
    myNumbers.add(20);
    myNumbers.add(25);
    for (int i : myNumbers) {
      System.out.println(i);
    }
  }
}
```

# Sort an ArrayList

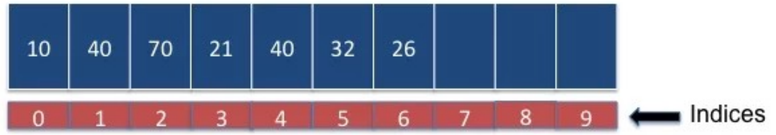
Another useful class in the `java.util` package is the `Collections` class, which include the `sort()` method for sorting lists alphabetically or numerically:

```
import java.util.ArrayList;
import java.util.Collections;  // Import the Collections class

public class Main {
  public static void main(String[] args) {
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    Collections.sort(cars);  // Sort cars
    for (String i : cars) {
      System.out.println(i);
    }
  }
}
```

## Sort an ArrayList of Integers:

```
import java.util.ArrayList;
import java.util.Collections;  // Import the Collections class

public class Main {
  public static void main(String[] args) {
    ArrayList<Integer> myNumbers = new ArrayList<Integer>();
    myNumbers.add(33);
    myNumbers.add(15);
    myNumbers.add(20);
    myNumbers.add(34);
    myNumbers.add(8);
    myNumbers.add(12);

    Collections.sort(myNumbers);  // Sort myNumbers

    for (int i : myNumbers) {
      System.out.println(i);
    }
  }
}
```

Default size of ArrayList is 10

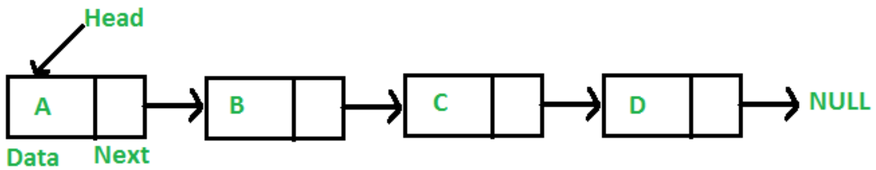



ArrayList

You can get or set element from any index

You can get index of any element using indexOf method

LinkedList



# Java LinkedList

In the previous chapter, you learned about the `ArrayList` class. The `LinkedList` class is almost identical to the `ArrayList`:

```
// Import the LinkedList class
import java.util.LinkedList;

public class Main {
  public static void main(String[] args) {
    LinkedList<String> cars = new LinkedList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    System.out.println(cars);
  }
}
```

## ArrayList vs. LinkedList

The `LinkedList` class is a collection which can contain many objects of the same type, just like the `ArrayList`.
The `LinkedList` class has all of the same methods as the `ArrayList` class because they both implement the `List` interface. This means that you can add items, change items, remove items and clear the list in the same way.
However, while the `ArrayList` class and the `LinkedList` class can be used in the same way, they are built very differently.

## How the ArrayList works

The `ArrayList` class has a regular array inside it. When an element is added, it is placed into the array. If the array is not big enough, a new, larger array is created to replace the old one and the old one is removed.

## How the LinkedList works

The `LinkedList` stores its items in "containers." The list has a link to the first container and each container has a link to the next container in the list. To add an element to the list, the element is placed into a new container and that container is linked to one of the other containers in the list.

## When To Use

Use an `ArrayList` for storing and accessing data, and `LinkedList` to manipulate data.

# LinkedList Methods

For many cases, the `ArrayList` is more efficient as it is common to need access to random items in the list, but the `LinkedList` provides several methods to do certain operations more efficiently:

| **Method** | **Description** |
| --- | --- |
| addFirst() | Adds an item to the beginning of the list. |
| addLast() | Add an item to the end of the list |
| removeFirst() | Remove an item from the beginning of the list. |
| removeLast() | Remove an item from the end of the list |
| getFirst() | Get the item at the beginning of the list |
| getLast() | Get the item at the end of the list |

## addFirst()

```
import java.util.LinkedList;

public class Main {
  public static void main(String[] args) {
    LinkedList<String> cars = new LinkedList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");

    // Use addFirst() to add the item to the beginning
    cars.addFirst("Mazda");
    System.out.println(cars);
  }
}
```

## addLast()

```
import java.util.LinkedList;

public class Main {
  public static void main(String[] args) {
    LinkedList<String> cars = new LinkedList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");

    // Use addLast() to add the item to the end
    cars.addLast("Mazda");
    System.out.println(cars);
  }
}
```

removeFirst()

```
import java.util.LinkedList;

public class Main {
public static void main(String[] args) {
LinkedList<String> cars = new LinkedList<String>();
cars.add("Volvo");
cars.add("BMW");
cars.add("Ford");
cars.add("Mazda");

// Use removeFirst() remove the first item from the list
cars.removeFirst();
System.out.println(cars);
}
}
```

# Java HashMap

```
import java.util.HashMap; // import the HashMap class

HashMap<String, String> capitalCities = new HashMap<String, String>();
```

## Add Items

```
// Import the HashMap class
import java.util.HashMap;

public class Main {
  public static void main(String[] args) {
    // Create a HashMap object called capitalCities
    HashMap<String, String> capitalCities = new HashMap<String, String>();

    // Add keys and values (Country, City)
    capitalCities.put("England", "London");
    capitalCities.put("Germany", "Berlin");
    capitalCities.put("Norway", "Oslo");
    capitalCities.put("USA", "Washington DC");
    System.out.println(capitalCities);
  }
}
```

## Access an Item

```
capitalCities.get("England");
```

## Remove an Item

```
capitalCities.remove("England");

capitalCities.clear();
```

## HashMap Size

```
capitalCities.size();
```

## Loop Through a HashMap

```
// Print keys
for (String i : capitalCities.keySet()) {
  System.out.println(i);
}

// Print values
for (String i : capitalCities.values()) {
  System.out.println(i);
}
```

```
// Print keys and values
for (String i : capitalCities.keySet()) {
  System.out.println("key: " + i + " value: " + capitalCities.get(i));
}
```

# Other Types

```
// Import the HashMap class
import java.util.HashMap;

public class Main {
  public static void main(String[] args) {

    // Create a HashMap object called people
    HashMap<String, Integer> people = new HashMap<String, Integer>();


    // Add keys and values (Name, Age)
    people.put("John", 32);
    people.put("Steve", 30);
    people.put("Angie", 33);

    for (String i : people.keySet()) {
      System.out.println("key: " + i + " value: " + people.get(i));
    }
  }
}
```

# Java HashSet

A HashSet is a collection of items where every item is unique, and it is found in the `java.util` package:

```
// Import the HashSet class
import java.util.HashSet;

public class Main {
  public static void main(String[] args) {
    HashSet<String> cars = new HashSet<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("BMW");
    cars.add("Mazda");
    System.out.println(cars);
  }
}
```

**Note:** In the example above, even though BMW is added twice it only appears once in the set because every item in a set has to be unique.

## Check If an Item Exists

```
cars.contains("Mazda");
```

## Remove an Item

```
cars.remove("Volvo");

cars.clear();
```

## HashSet Size

```
cars.size();
```

## Loop Through a HashSet

```
for (String i : cars) {
  System.out.println(i);
}
```

```
import java.util.HashSet;

public class Main {
  public static void main(String[] args) {

    // Create a HashSet object called numbers
    HashSet<Integer> numbers = new HashSet<Integer>();

    // Add values to the set
    numbers.add(4);
    numbers.add(7);
    numbers.add(8);

    // Show which numbers between 1 and 10 are in the set
    for(int i = 1; i <= 10; i++) {
      if(numbers.contains(i)) {
        System.out.println(i + " was found in the set.");
      } else {
        System.out.println(i + " was not found in the set.");
      }
    }
  }
}
```

# Java Iterator

An `Iterator` is an object that can be used to loop through collections, like ArrayList and HashSet. It is called an "iterator" because "iterating" is the technical term for looping.

## Getting an Iterator

```
// Import the ArrayList class and the Iterator class
import java.util.ArrayList;
import java.util.Iterator;

public class Main {
  public static void main(String[] args) {

    // Make a collection
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");

    // Get the iterator
    Iterator<String> it = cars.iterator();

    // Print the first item
    System.out.println(it.next());
  }
}
```

## Looping Through a Collection

```
while(it.hasNext()) {
  System.out.println(it.next());
}
```

# Removing Items from a Collection

```
import java.util.ArrayList;
import java.util.Iterator;

public class Main {
  public static void main(String[] args) {
    ArrayList<Integer> numbers = new ArrayList<Integer>();
    numbers.add(12);
    numbers.add(8);
    numbers.add(2);
    numbers.add(23);
    Iterator<Integer> it = numbers.iterator();
    while(it.hasNext()) {
      Integer i = it.next();
      if(i < 10) {
        it.remove();
      }
    }
    System.out.println(numbers);
  }
}
```

**Note:** Trying to remove items using a **for loop** or a **for-each loop** would not work correctly because the collection is changing size at the same time that the code is trying to loop.

# Java Wrapper Classes

| Primitive Data Type | Wrapper Class |
|---|---|
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| boolean | Boolean |
| char | Character |

```
ArrayList<int> myNumbers = new ArrayList<int>(); // Invalid
ArrayList<Integer> myNumbers = new ArrayList<Integer>(); // Valid
```

```
public class Main {
  public static void main(String[] args) {
    Integer myInt = 5;
    Double myDouble = 5.99;
    Character myChar = 'A';
    System.out.println(myInt.intValue());
    System.out.println(myDouble.doubleValue());
    System.out.println(myChar.charValue());
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    Integer myInt = 100;
    String myString = myInt.toString();
    System.out.println(myString.length());
  }
}
```

**Java Collection: ArrayList Exercises [22 exercises with solution]**

**1.** Write a Java program to create a new array list, add some colors (string) and print out the collection.

**2.** Write a Java program to iterate through all elements in a array list.

**3.** Write a Java program to insert an element into the array list at the first position.

**4.** Write a Java program to retrieve an element (at a specified index) from a given array list.

**5.** Write a Java program to update specific array element by given element.

**6.** Write a Java program to remove the third element from a array list.

**7.** Write a Java program to search an element in a array list.

**8.** Write a Java program to sort a given array list.

**9.** Write a Java program to copy one array list into another.

**10.** Write a Java program to shuffle elements in a array list.

**11.** Write a Java program to reverse elements in a array list.

**12.** Write a Java program to extract a portion of a array list.

**13.** Write a Java program to compare two array lists.

**14.** Write a Java program of swap two elements in an array list.

**15.** Write a Java program to join two array lists.

**16.** Write a Java program to clone an array list to another array list.

**17.** Write a Java program to empty an array list.

**18.** Write a Java program to test an array list is empty or not.

**19.** Write a Java program to trim the capacity of an array list the current list size.

**20.** Write a Java program to increase the size of an array list.

**21.** Write a Java program to replace the second element of a ArrayList with the specified element.

**22.** Write a Java program to print all the elements of a ArrayList using the position of the elements.

**Java Collection: LinkedList Exercises [26 exercises with solution]**

**1.** Write a Java program to append the specified element to the end of a linked list.

**2.** Write a Java program to iterate through all elements in a linked list.

**3.** Write a Java program to iterate through all elements in a linked list starting at the specified position.

**4.** Write a Java program to iterate a linked list in reverse order.

**5.** Write a Java program to insert the specified element at the specified position in the linked list.

**6.** Write a Java program to insert elements into the linked list at the first and last position.

**7.** Write a Java program to insert the specified element at the front of a linked list.

**8.** Write a Java program to insert the specified element at the end of a linked list.

**9.** Write a Java program to insert some elements at the specified position into a linked list.

**10.** Write a Java program to get the first and last occurrence of the specified elements in a linked list.

**11.** Write a Java program to display the elements and their positions in a linked list.

**12.** Write a Java program to remove a specified element from a linked list.

**13.** Write a Java program to remove first and last element from a linked list.

**14.** Write a Java program to remove all the elements from a linked list.

**15.** Write a Java program of swap two elements in a linked list.

**16.** Write a Java program to shuffle the elements in a linked list.

**17.** Write a Java program to join two linked lists.

**18.** Write a Java program to clone an linked list to another linked list.

**19.** Write a Java program to remove and return the first element of a linked list.

**20.** Write a Java program to retrieve but does not remove, the first element of a linked list.

**21.** Write a Java program to retrieve but does not remove, the last element of a linked list.

**22.** Write a Java program to check if a particular element exists in a linked list.

**23.** Write a Java program to convert a linked list to array list.

**24.** Write a Java program to compare two linked lists.

**25.** Write a Java program to test an linked list is empty or not.

**26.** Write a Java program to replace an element in a linked list.

**Java Collection: HashSet Exercises**

1. Write a Java program to append the specified element to the end of a hash set.

**2.** Write a Java program to iterate through all elements in a hash list.

**3.** Write a Java program to get the number of elements in a hash set.

**4.** Write a Java program to empty an hash set.

**5.** Write a Java program to test a hash set is empty or not.

**6.** Write a Java program to clone a hash set to another hash set.

**7.** Write a Java program to convert a hash set to an array.

**8.** Write a Java program to convert a hash set to a tree set.

**9.** Write a Java program to convert a hash set to a List/ArrayList.

**10.** Write a Java program to compare two hash set.

**11.** Write a Java program to compare two sets and retain elements which are same on both sets.

**12.** Write a Java program to remove all of the elements from a hash set.

Java Collection: HashMap Exercises

**1.** Write a Java program to associate the specified value with the specified key in a HashMap.
**2.** Write a Java program to count the number of key-value (size) mappings in a map.
**3.** Write a Java program to copy all of the mappings from the specified map to another map.
**4.** Write a Java program to remove all of the mappings from a map.
**5.** Write a Java program to check whether a map contains key-value mappings (empty) or not.
**6.** Write a Java program to get a shallow copy of a HashMap instance.
**7.** Write a Java program to test if a map contains a mapping for the specified key.
**8.** Write a Java program to test if a map contains a mapping for the specified value.
**9.** Write a Java program to create a set view of the mappings contained in a map.
**10.** Write a Java program to get the value of a specified key in a map.
**11.** Write a Java program to get a set view of the keys contained in this map.
**12.** Write a Java program to get a collection view of the values contained in this map.

# Java Exceptions - Try...Catch

## Java Exceptions

When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.
When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an **exception** (throw an error).

## Java try and catch

The `try` statement allows you to define a block of code to be tested for errors while it is being executed.
The `catch` statement allows you to define a block of code to be executed, if an error occurs in the try block.
The `try` and `catch` keywords come in pairs:

```
try {
  // Block of code to try
}
catch(Exception e) {
  // Block of code to handle errors
}
```

This will generate an error, because **myNumbers[10]** does not exist.

```
public class Main {
  public static void main(String[ ] args) {
    int[] myNumbers = {1, 2, 3};
    System.out.println(myNumbers[10]); // error!
  }
}
```

```
public class Main {
  public static void main(String[ ] args) {
    try {
      int[] myNumbers = {1, 2, 3};
      System.out.println(myNumbers[10]);
    } catch (Exception e) {
      System.out.println("Something went wrong.");
    }
  }
}
```

# Finally

The `finally` statement lets you execute code, after `try...catch`, regardless of the result:

```
public class Main {
  public static void main(String[] args) {
    try {
      int[] myNumbers = {1, 2, 3};
      System.out.println(myNumbers[10]);
    } catch (Exception e) {
      System.out.println("Something went wrong.");
    } finally {
      System.out.println("The 'try catch' is finished.");
    }
  }
}
```

# The throw keyword

```
checkAge(20);
```

The `throw` statement allows you to create a custom error.
The `throw` statement is used together with an **exception type**. There are many exception types available in Java: `ArithmeticException`, `FileNotFoundException`, `ArrayIndexOutOfBoundsException`, `SecurityException`, etc:

Throw an exception if **age** is below 18 (print "Access denied"). If age is 18 or older, print "Access granted":

```
public class Main {
  static void checkAge(int age) {
    if (age < 18) {
      throw new ArithmeticException("Access denied - You must be at least 18 years old.");
    }
    else {
      System.out.println("Access granted - You are old enough!");
    }
  }

  public static void main(String[] args) {
    checkAge(15); // Set age to 15 (which is below 18...)
  }
}
```

The output will be:

```
Exception in thread "main" java.lang.ArithmeticException: Access denied - You must be at least 18 years old.
        at Main.checkAge(Main.java:4)
        at Main.main(Main.java:12)
```

# Comparable

```
package com.journaldev.sort;

import java.util.Comparator;

public class Employee implements Comparable<Employee> {

	private int id;
	private String name;
	private int age;
	private long salary;

	public int getId() {
		return id;
		}

	public String getName() {
		return name;
		}

	public int getAge() {
		return age;
		}

	public long getSalary() {
		return salary;
		}

	public Employee(int id, String name, int age, int salary) {
		this.id = id;
		this.name = name;
		this.age = age;
		this.salary = salary;
		}

		@Override
		public int compareTo(Employee emp) {
		//let's sort the employee based on an id in ascending order
		//returns a negative integer, zero, or a positive integer as this employee id
		//is less than, equal to, or greater than the specified object.
		return (this.id - emp.id);
		}


	}
```

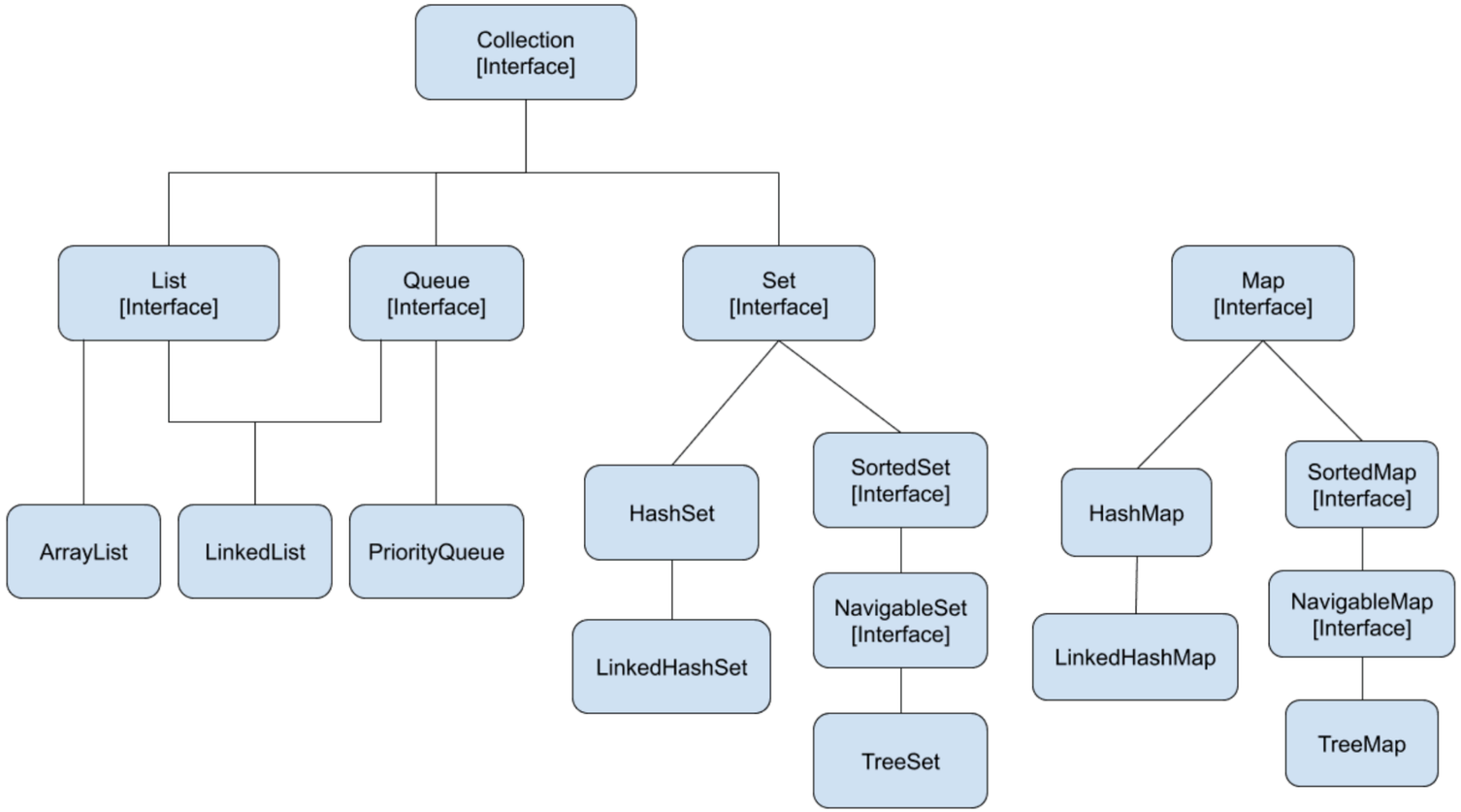
```
//sorting object array
Employee[] empArr = new Employee[4];
empArr[0] = new Employee(10, "Mikey", 25, 10000);
empArr[1] = new Employee(20, "Arun", 29, 20000);
empArr[2] = new Employee(5, "Lisa", 35, 5000);
empArr[3] = new Employee(1, "Pankaj", 32, 50000);

//sorting employees array using Comparable interface
implementation
Arrays.sort(empArr);
System.out.println("Default Sorting of Employees list:
\n"+Arrays.toString(empArr));
```

# Comparator

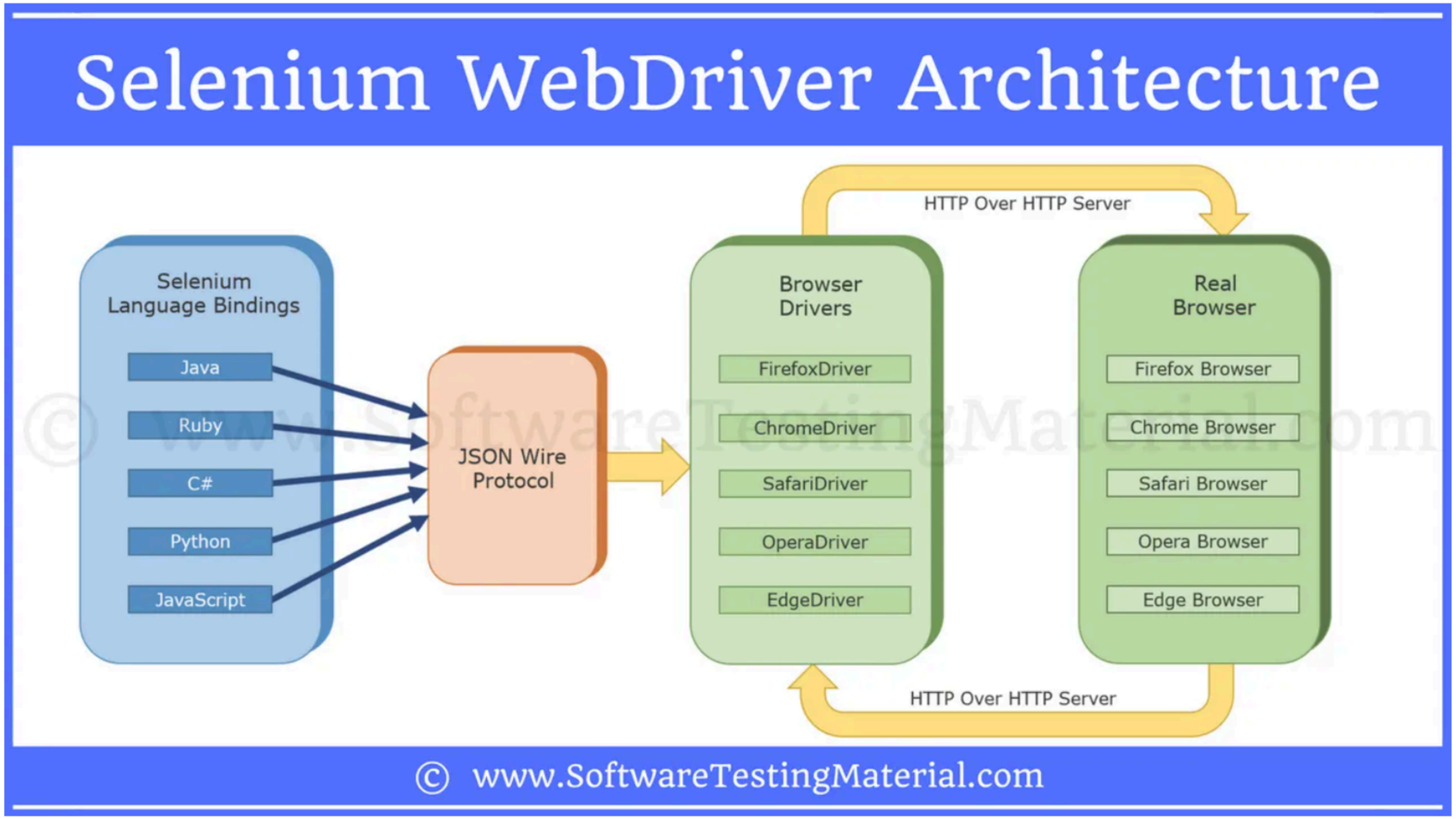
```
package com.journaldev.sort;

import java.util.Comparator;

public class EmployeeComparatorByIdAndName implements Comparator<Employee> {

@Override
public int compare(Employee o1, Employee o2) {
int flag = o1.getId() - o2.getId();
if(flag==0) flag = o1.getName().compareTo(o2.getName());
return flag;
}

}
```

```
package com.journaldev.sort;

import java.util.Arrays;

public class JavaObjectSorting {

/**
* This class shows how to sort custom objects array/list
* implementing Comparable and Comparator interfaces
* @param args
*/
public static void main(String[] args) {

//sorting custom object array
Employee[] empArr = new Employee[4];
empArr[0] = new Employee(10, "Mikey", 25, 10000);
empArr[1] = new Employee(20, "Arun", 29, 20000);
empArr[2] = new Employee(5, "Lisa", 35, 5000);
empArr[3] = new Employee(1, "Pankaj", 32, 50000);

//sorting employees array using Comparable interface implementation
Arrays.sort(empArr);
System.out.println("Default Sorting of Employees list:\n"+Arrays.toString(empArr));

//sort employees array using Comparator by Salary
Arrays.sort(empArr, Employee.SalaryComparator);
System.out.println("Employees list sorted by Salary:\n"+Arrays.toString(empArr));

//sort employees array using Comparator by Age
Arrays.sort(empArr, Employee.AgeComparator);
System.out.println("Employees list sorted by Age:\n"+Arrays.toString(empArr));

//sort employees array using Comparator by Name
Arrays.sort(empArr, Employee.NameComparator);
System.out.println("Employees list sorted by Name:\n"+Arrays.toString(empArr));

//Employees list sorted by ID and then name using Comparator class
empArr[0] = new Employee(1, "Mikey", 25, 10000);
Arrays.sort(empArr, new EmployeeComparatorByIdAndName());
System.out.println("Employees list sorted by ID and Name:\n"+Arrays.toString(empArr));
}

}
```

# Collections Framework hierarchy



# Selenium


Selenium WebDriver Architecture
HTTP Over HTTP Server
Selenium Language Bindings
Java
Ruby
C#
Python
JavaScript
JSON Wire Protocol
Browser Drivers
FirefoxDriver
ChromeDriver
SafariDriver
OperaDriver
EdgeDriver
Real Browser
Firefox Browser
Chrome Browser
Safari Browser
Opera Browser
Edge Browser
HTTP Over HTTP Server
© www.SoftwareTestingMaterial.com

# XPath

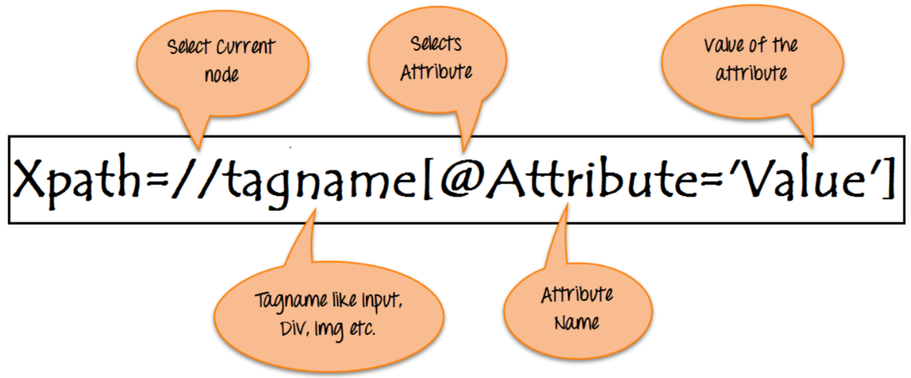
There are two types of XPath:

**1) Absolute XPath**

**2) Relative XPath**

Id
xPath
Css selector
Link text
Partial link text
class

The basic format of XPath in selenium is explained below with screen shot.




Basic Format of XPath

```
Xpath=//*[contains(@type,'sub')]

Xpath=//*[@type='submit' or @name='btnReset']

Xpath=//input[@type='submit' and @name='btnLogin']

Xpath=//label[starts-with(@id,'message')]

Xpath=//td[text()='UserID']

Xpath=//*[@type='text']//following::input

Xpath=//*[text()='Enterprise Testing']//ancestor::div

Xpath=//*[@id='java_technologies']//child::li
```

```
Xpath=//*[@id='rt-feature']//parent::div

Xpath=//*[@id='rt-feature']//descendant::a[1]
```

## Css Selector

```
node[attribute_name = ‘attribute_value’]
```

```
node[attribute_name^ = ‘attribute_value’]
```

```
node[attribute_name$ = ‘attribute_value’]
```

```
.Class_name
```

```
#id
```

```java
@Test
public void testBrokenLink() throws IOException {
    driver.get("http://www.leafground.com/pages/Link.html");
    List<WebElement> webElementList = driver.findElements(By.xpath("//a"));
    for (int i = 0; i < webElementList.size(); i++) {
        WebElement brokenLink = webElementList.get(i);
        String url = brokenLink.getAttribute("href");
        URL u = new URL(url);
        HttpURLConnection huc = (HttpURLConnection) u.openConnection();
        huc.setRequestMethod("GET");
        huc.connect();
        int respCode = huc.getResponseCode();
        if (respCode >= 400) {
            System.out.println("Broken link");
        } else {
            System.out.println("Valid link");
        }
    }
}
```

*Implicit Wait:*

*driver.manage().timeouts().implicitlyWait()*

Explicit Wait:

```
WebDriverWait wait = new WebDriverWait(AppPage.driver, AppPage.WAIT_TWO_MIN);
wait.until((driver) -> ExpectedConditions.elementToBeClickable(locator));
```

Fluent Wait:

```
Wait<WebDriver> wait = new FluentWait<WebDriver>(AppPage.driver)
        .withTimeout(AppPage.WAIT_TWO_MIN, TimeUnit.SECONDS).pollingEvery(5, TimeUnit.SECONDS)
        .ignoring(NoSuchElementException.class);
wait.until((driver) -> e.isEnabled());
```

**Page Object Model**

```
public class Guru99Login {

    WebDriver driver;

    By user99GuruName = By.name("uid");

    By password99Guru = By.name("password");

    By titleText =By.className("barone");

    By login = By.name("btnLogin");

    public Guru99Login(WebDriver driver){

        this.driver = driver;

    }

    //Set user name in textbox

    public void setUserName(String strUserName){

        driver.findElement(user99GuruName).sendKeys(strUserName);

    }

    //Set password in password textbox

    public void setPassword(String strPassword){

         driver.findElement(password99Guru).sendKeys(strPassword);

    }
```

**Page Factory Model**

```
public class Guru99Login {

    /**
     * All WebElements are identified by @FindBy annotation
     */
    WebDriver driver;
    @FindBy(name="uid")
    WebElement user99GuruName;
    @FindBy(name="password")
    WebElement password99Guru;
    @FindBy(className="barone")
    WebElement titleText;
    @FindBy(name="btnLogin")
    WebElement login;
    public Guru99Login(WebDriver driver){
        this.driver = driver;
        //This initElements method will create all WebElements
        PageFactory.initElements(driver, this);
    }
    //Set user name in textbox
    public void setUserName(String strUserName){
        user99GuruName.sendKeys(strUserName);
    }
```

```
AjaxElementLocatorFactory factory = new AjaxElementLocatorFactory(driver,10);
PageFactory.initElements(factory, this)
```

POM
Page Factory[ base page]
Base test
TestNG[ data provider, priority , expectedExceptions, enabled , invocationCount ,timeOut = millliseconds ,
@BeforeTest [ in parent class]
@BeforeSuite …
@AfterSuite

Create a testng xml

```
dependsOnMethods = { “test name” }
```

```
@Test(groups= {"Include Group"})

public void test1() throws InterruptedException {

   System.out.println("from 1");

   Thread.sleep(100);

}

<suite name="regression suite">

   <groups>

       <run>

           <include name="Include Group"/>

       </run>

   </groups>

   <test name="test">

       <classes>

           <class name="test.Testng"></class>

       </classes>

   </test>

</suite>
```

- ```
  @Parameters({"testtt",})
  @Test()
  public void test1(@Optional("Optional Parameter Selected") String s)
  ```
- ```
  <parameter name="testtt" value="one"></parameter>
  - parallel
  <test verbose="2" preserve-order="true" name="/Users/cb-sarathkumar/work/cb-ui-middleware/GensparkMavenUI" parallel="classes"
   thread-count=“5”>
  ```
- ```
  Thread.currentThread().getId()
  ```
  ```

## Configuring the Test Methods to run parallelly in TestNG

```
public class TestNG
{
    @Test(threadPoolSize = 4, invocationCount = 4, timeOut = 1000)
    public void testMethod()
    {
        System.out.println("Thread ID Is : " + Thread.currentThread().getId());
    }
}
```

## Parallel test execution using DataProviders in TestNG

```
public class TestNG
{
@Test(dataProvider = "dp")

public void testMethod(int number)
{
    System.out.println("The Thread ID for "+ number + " Is :  " + Thread.currentThread().getId());
}


@DataProvider(name = "dp",parallel=true)
public Object[][] dp1() {
  return new Object[][] {
      new Object[] { 1 },
      new Object[] { 2 },
      new Object[] { 3 },
      new Object[] { 4 },
      new Object[] { 5 },
      new Object[] { 6 },
      new Object[] { 7 },
      new Object[] { 8 }

  };
}
}
```

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="ParallelExecutionDP" data-provider-thread-count="3" >
    <test name="ExecuteDPParallelly">
        <classes>
            <class name="TestNG"></class>
        </classes>
    </test>
</suite>
```

## What is Cross-Browser Testing?

***Cross-browser testing*** is the process of testing our website on different browsers and operating systems. With cross-browser testing, we make sure that the site is rendered the same in every browser. We can perform cross-browser testing either manually or in an automated way, but the manual method is very tedious. The reason being that while performing cross-browser testing, we do not only care about the browsers but their different versions and the operating systems too. So just imagine the permutations of so many browsers with so many versions (*Chrome is on 83*) and operating systems. Thus, a better way is to choose an automated way.

***Since the engine is different, the way a browser understands the web languages (HTML, JavaScript, and CSS) is different***. So, the position property of  CSS would run fine on Chrome but will choke on Safari because of the absence of -*WebKit* tag. It makes cross-browser testing using TestNG a very crucial job. If I summarize the main big things that make cross-browser testing using TestNG, then it would be as follows:

- *Cross-browser testing using TestNG ensure a better performance on different browsers and OS.*
- *Image orientations mess up a lot of the time. We can take care of it.*
- *The tester and the developer can assure how JavaScript renders on different browsers.*
- *One can track Font-size issues.*
- *The unsupported tags can be revealed, which can be taken care of by turnaround codes.*

# Capturing ScreenShot in Selenium

```
//Take the screenshot
        File screenshot = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

        //Copy the file to a location and use try catch block to handle exception
        try {
            FileUtils.copyFile(screenshot, new File("C:\\projectScreenshots\\homePageScreenshot.png"));
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
```

```
JavascriptExecutor js = (JavascriptExecutor)driver;
js.executeScript("arguments[0].click();", element);
```

```
Calendar calendar = Calendar.getInstance();
SimpleDateFormat formater = new SimpleDateFormat("dd_MM_yyyy_hh_mm_ss");
String methodName = result.getName();
if(!result.isSuccess()){
    File scrFile = ((TakesScreenshot)getDriver()).getScreenshotAs(OutputType.FILE);
    try {
        String reportDirectory = new File(System.getProperty("user.dir")).getAbsolutePath() + "/target/surefire-reports";
        File destFile = new File((String) reportDirectory+"/failure_screenshots/"+methodName+"_"+formater.format(calendar.getTime())+".png");
        FileUtils.copyFile(scrFile, destFile);
        Reporter.log("<a href='"+ destFile.getAbsolutePath() + "'> <img src='"+ destFile.getAbsolutePath() + "' height='100' width='100'/> </a>");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

# Selenium WebDriver Event Listener

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.events.WebDriverEventListener;

public class EventHandler implements WebDriverEventListener{

	public void afterChangeValueOf(WebElement arg0, WebDriver arg1) {
		// TODO Auto-generated method stub

		System.out.println("inside method afterChangeValueOf on " + arg0.toString());
	}

	public void afterClickOn(WebElement arg0, WebDriver arg1) {
		// TODO Auto-generated method stub
		System.out.println("inside method afterClickOn on " + arg0.toString());
	}

	public void afterFindBy(By arg0, WebElement arg1, WebDriver arg2) {
		// TODO Auto-generated method stub
		System.out.println("Find happened on " + arg1.toString()
				+ " Using method " + arg0.toString());
	}

	public void afterNavigateBack(WebDriver arg0) {
		// TODO Auto-generated method stub

		System.out.println("Inside the after navigateback to " + arg0.getCurrentUrl());
	}

	public void afterNavigateForward(WebDriver arg0) {
		// TODO Auto-generated method stub
		System.out.println("Inside the afterNavigateForward to " + arg0.getCurrentUrl());
	}

	public void afterNavigateTo(String arg0, WebDriver arg1) {
		// TODO Auto-generated method stub
		System.out.println("Inside the afterNavigateTo to " + arg0);
	}

	public void afterScript(String arg0, WebDriver arg1) {
		// TODO Auto-generated method stub
		System.out.println("Inside the afterScript to, Script is " + arg0);
	}

	public void beforeChangeValueOf(WebElement arg0, WebDriver arg1) {
		// TODO Auto-generated method stub

		System.out.println("Inside the beforeChangeValueOf method");
	}

	public void beforeClickOn(WebElement arg0, WebDriver arg1) {
		// TODO Auto-generated method stub
		System.out.println("About to click on the " + arg0.toString());

	}

	public void beforeFindBy(By arg0, WebElement arg1, WebDriver arg2) {
		// TODO Auto-generated method stub
		System.out.println("Just before finding element " + arg1.toString());

	}

	public void beforeNavigateBack(WebDriver arg0) {
		// TODO Auto-generated method stub
		System.out.println("Just before beforeNavigateBack " + arg0.getCurrentUrl());

	}

	public void beforeNavigateForward(WebDriver arg0) {
		// TODO Auto-generated method stub
		System.out.println("Just before beforeNavigateForward " + arg0.getCurrentUrl());

	}

	public void beforeNavigateTo(String arg0, WebDriver arg1) {
		// TODO Auto-generated method stub
		System.out.println("Just before beforeNavigateTo " + arg0);
	}

	public void beforeScript(String arg0, WebDriver arg1) {
		// TODO Auto-generated method stub
		System.out.println("Just before beforeScript " + arg0);
	}

	public void onException(Throwable arg0, WebDriver arg1) {
		// TODO Auto-generated method stub
		System.out.println("Exception occured at " + arg0.getMessage());

	}

}
```

```
public static void main(String[] args) {
		// TODO Auto-generated method stub

		 FirefoxDriver driver = new FirefoxDriver();
		 EventFiringWebDriver eventDriver = new EventFiringWebDriver(driver);

		EventHandler handler = new EventHandler();
		eventDriver.register(handler);
		eventDriver.get("https://toolsqa.com/automation-practice-switch-windows/");
		WebElement element = eventDriver.findElement(By.id("target"));
		element.click();

	}
```

# ITestListener

```
@Listeners(ListenersTestNG.class)
```

```
import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;
public class ListenersTestNG implements ITestListener {
    public void onStart(ITestContext context) {
        System.out.println("onStart method started");
    }

    public void onFinish(ITestContext context) {
        System.out.println("onFinish method started");
    }

        public void onTestStart(ITestResult result) {
            System.out.println("New Test Started" +result.getName());
        }

        public void onTestSuccess(ITestResult result) {
            System.out.println("onTestSuccess Method" +result.getName());
        }

        public void onTestFailure(ITestResult result) {
            System.out.println("onTestFailure Method" +result.getName());
        }

        public void onTestSkipped(ITestResult result) {
            System.out.println("onTestSkipped Method" +result.getName());
        }

        public void onTestFailedButWithinSuccessPercentage(ITestResult result) {
            System.out.println("onTestFailedButWithinSuccessPercentage" +result.getName());
        }
}
```

```
<suite name="Test Suite" parallel="methods" thread-count="2" verbose="1">
    <listeners>
        <listener class-name="com.vsysq.base.TestListener" />
    </listeners>
```

```
private ITestClass testClass;

private void checkAndInitTestClass(ITestResult result) {
    if (result.getMethod().getTestClass() != testClass) {
        testClass = result.getMethod().getTestClass();
    }
}

@Override
public void onTestStart(ITestResult result) {
    checkAndInitTestClass(result);
    System.out.println(testClass.getRealClass().getSimpleName());
    System.out.println(testClass.getRealClass().getName());
    System.out.println(result.getMethod().getMethodName());
}
```

```
private ITestClass testClass;

private void checkAndInitTestClass(ITestResult result) {
    if (result.getMethod().getTestClass() != testClass) {
        testClass = result.getMethod().getTestClass();
    }
}

@Override
public void onTestStart(ITestResult result) {
    checkAndInitTestClass(result);
    System.out.println(testClass.getRealClass().getSimpleName());
    System.out.println(testClass.getRealClass().getName());
    System.out.println(result.getMethod().getMethodName());
}
```

Configure it under Edit configuration

IRetryAnalyzer

```java
public class RetryAnalyzer implements IRetryAnalyzer {

    int counter = 0;
    int retryLimit = 4;

    @Override
    public boolean retry(ITestResult result) {

        if(counter < retryLimit)
        {
            counter++;
            return true;
        }
        return false;
    }
}
```

```java
    @Test(retryAnalyzer = Liste.class)
public void test2() {
    Assert.assertEquals("1",0);
    System.out.println("from 22");
}
```

```java
public class AnnotationTransformer implements IAnnotationTransformer {

    @Override
    public void transform(ITestAnnotation annotation,
                                            Class testClass,
                                            Constructor testConstructor,
                                            Method testMethod) {
        annotation.setRetryAnalyzer(RetryAnalyzer.class);
    }
}
```

```xml
<listeners>
    <listener class-name="test.AnnotationTransformer"></listener>
</listeners>
```

```
   <dependency>
       <groupId>org.apache.poi</groupId>
       <artifactId>poi-ooxml</artifactId>
       <version>4.1.1</version>
   </dependency>
```

```
public static void main(String[] args) throws InterruptedException, IOException {
    File file =    new File("/Users/cb-sarathkumar/Downloads/demo.xlsx");
    FileInputStream inputStream = new FileInputStream(file);
    XSSFWorkbook wb=new XSSFWorkbook(inputStream);
    XSSFSheet ws = wb.getSheet("Sheet1");
    int rows = ws.getRow(0).getLastCellNum();
    XSSFRow row = ws.getRow(0);
    int cols = row.getLastCellNum();
    System.out.println(rows);
    System.out.println(cols);
    for (int i = 0; i < rows; i++) {
        row =  ws.getRow(i);
        for (int j = 0; j < row.getLastCellNum(); j++) {
            System.out.println(row.getCell(j));
        }
    }
}
```

# Data Driven Framework

```
public class ExcelUtils {
    private static XSSFWorkbook workbook;
    private static XSSFSheet sheet;
    private static XSSFRow row;
    private static XSSFCell cell;

    public void setExcelFile(String excelFilePath, String sheetName) throws IOException {
        //Create an object of File class to open xls file
        File file = new File(excelFilePath);

        //Create an object of FileInputStream class to read excel file
        FileInputStream inputStream = new FileInputStream(file);

        //creating workbook instance that refers to .xls file
        workbook = new XSSFWorkbook(inputStream);

        //creating a Sheet object
        sheet = workbook.getSheet(sheetName);

    }

    public String getCellData(int rowNumber, int cellNumber) {
        //getting the cell value from rowNumber and cell Number
        cell = sheet.getRow(rowNumber).getCell(cellNumber);

        //returning the cell value as string
        return cell.getStringCellValue();
    }

    public int getRowCountInSheet() {
        int rowcount = sheet.getLastRowNum() - sheet.getFirstRowNum();
        return rowcount;
    }

    public void setCellValue(int rowNum, int cellNum, String cellValue, String excelFilePath) throws IOException {
        //creating a new cell in row and setting value to it
        sheet.getRow(rowNum).createCell(cellNum).setCellValue(cellValue);
        FileOutputStream outputStream = new FileOutputStream(excelFilePath);
        workbook.write(outputStream);
outputStream.flush();
outputStream.close();

    }
}

class demo {
    public static void main(String[] args) throws IOException {
        ExcelUtils excelUtils = new ExcelUtils();
        excelUtils.setExcelFile("/Users/cb-sarathkumar/Downloads/demo.xlsx","Sheet1");
        System.out.println(excelUtils.getCellData(1,1));
    }
}
```

```java
public static Map<String, Map<String, String>> readTestData() throws IOException {
    String path = "/Users/cb-sarathkumar/Downloads/demo.xlsx";
    FileInputStream fis = new FileInputStream(path);
    Workbook wb = new XSSFWorkbook(fis);
    Sheet sh = wb.getSheetAt(0);
    int lastRow = sh.getLastRowNum();
    Map<String, Map<String, String>> fileMap = new HashMap<>();
    Map<String, String> dataMap = new HashMap<>();
    for (int i = 1; i <= lastRow; i++) {
        Row headers = sh.getRow(0);
        Row row = sh.getRow(i);
        Cell keyCol = row.getCell(0);
        for (int j = 1; j < row.getLastCellNum(); j++) {
            dataMap.put(headers.getCell(j).toString(), row.getCell(j).toString());
        }
        fileMap.put(keyCol.toString(),dataMap);
    }
    return fileMap;
}
```

<testName,<colHeaders,colValues>>

```java
private ITestClass testClass;

private void checkAndInitTestClass(ITestResult result) {
    if (result.getMethod().getTestClass() != testClass) {
        testClass = result.getMethod().getTestClass();
    }
}

@Override
public void onTestStart(ITestResult result) {
    checkAndInitTestClass(result);
    System.out.println(testClass.getRealClass().getSimpleName());
    System.out.println(testClass.getRealClass().getName());
    System.out.println(result.getMethod().getMethodName());
}
```

```
    ChromeOptions chromeOptions = new ChromeOptions();
//            chromeOptions.setHeadless(true);
//            chromeOptions.addArguments("disable-infobars");
            Map<String, Object> prefs = new HashMap<String, Object>();
            prefs.put("download.default_directory", System.getProperty("user.dir") + File.separator + "externalFiles" + File.separator + "downloadFiles");
            chromeOptions.setExperimentalOption("prefs", prefs);
```

## Factory pattern

*log4j.properties*

```
# Define the root logger with appender file
log4j.rootLogger = DEBUG, console, file

#Define console appender
log4j.appender.console=org.apache.log4j.ConsoleAppender
logrj.appender.console.Target=System.out
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%-5p %c{1} - %m%n

#Define rolling file appender
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=logs/main.log
log4j.appender.file.Append=false
log4j.appender.file.ImmediateFlush=true
log4j.appender.file.MaxFileSize=10MB
log4j.appender.file.MaxBackupIndex=5
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d %d{Z} [%t] %-5p (%F:%L) - %m%n
```

```
Logger Log = Logger.getLogger(className.class);

@BeforeTest
public void setUp() {
        PropertyConfigurator.configure("log4j.properties");
    Log.info("New driver instantiated");
}
```

Download create testing xml plugin

mvn clean test -DsuiteXmlFile=testng.xml

```
System.out.println(System.getProperty("browser"));
```

Mvn clean - will delete the target folder contents

Comment dependency in pom.xml

Mvn compile - to generate compile files

```
<properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <browser>${browser}</browser>
</properties>
```

```
<build>
  <sourceDirectory>src</sourceDirectory>
  <testSourceDirectory>test</testSourceDirectory>
</build>
```

Mvn test - will run the test cases

Mvn surefire plugin - will give us reports and need to pass testing.xml to this

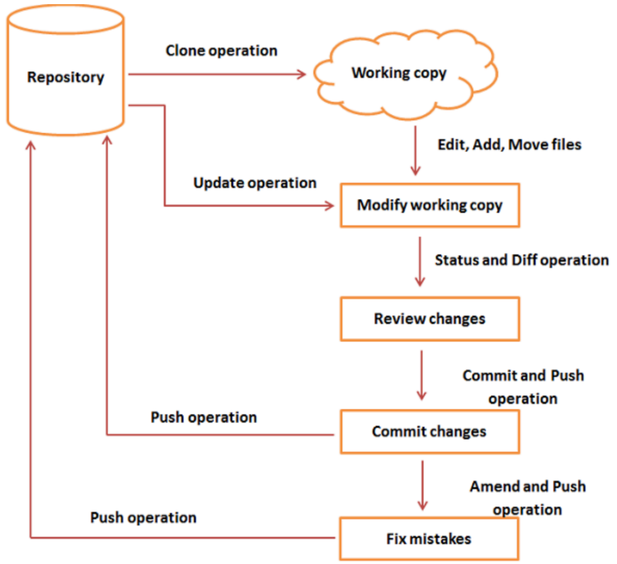
```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.0.0-M5</version>
            <configuration>
                <suiteXmlFiles>
                        <suiteXmlFile>
                            testng.xml
                        </suiteXmlFile>
                </suiteXmlFiles>
            </configuration>
        </plugin>
    </plugins>
    <sourceDirectory>src</sourceDirectory>
    <testSourceDirectory>src/test</testSourceDirectory>
</build>
```

Mvn package -DskipTests will create a Jar file

Mvn install will copy the Jar file in .m2 so that other projects can use this

- `validate` - validate the project is correct and all necessary information is available
- `compile` - compile the source code of the project
- `test` - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- `package` - take the compiled code and package it in its distributable format, such as a JAR.
- `verify` - run any checks on results of integration tests to ensure quality criteria are met
- `install` - install the package into the local repository, for use as a dependency in other projects locally
- `deploy` - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

# GIT




https://education.github.com/git-cheat-sheet-education.pdf

Merge conflict:
git checkout --ours filename.c
git checkout --theirs filename.c
git add filename.c
git commit -m "using theirs"

By setting the schedule period to `15 13 * * *` you tell Jenkins to schedule the build every day of every month of every year at the 15th minute of the 13th hour of the day.
Jenkins used a cron expression, and the different fields are:

1. MINUTES Minutes in one hour (0-59)
2. HOURS Hours in one day (0-23)
3. DAYMONTH Day in a month (1-31)
4. MONTH Month in a year (1-12)
5. DAYWEEK Day of the week (0-7) where 0 and 7 are sunday

If you want to schedule your build every 5 minutes, this will do the job : `*/5 * * * *`
If you want to schedule your build every day at 8h00, this will do the job : `0 8 * * *`
For the past few versions (2014), Jenkins have a new parameter, `H` (extract from the Jenkins code documentation):
To allow periodically scheduled tasks to produce even load on the system, the symbol `H` (for “hash”) should be used wherever possible.
For example, using `0 0 * * *` for a dozen daily jobs will cause a large spike at midnight. In contrast, using `H H * * *` would still execute each job once a day, but not all at the same time, better using limited resources.
Note also that:
The `H` symbol can be thought of as a random value over a range, but it actually is a hash of the job name, not a random function, so that the value remains stable for any given project.

```
docker version
docker pull browserless/chrome
docker run browserless/chrome
docker pull hello-world
docker ps
docker images
```

https://dockerlabs.collabnix.com/docker/cheatsheet/

```
docker run -it selenium/node-chrome bash
Docker rm <container name>
```

docker-compose up

docker-compose up -d --scale chrome=10

```
chromeOptions.addArguments("“-disable-setuid-sandbox”"); // disabling infobars
chromeOptions.addArguments("--disable-extensions"); // disabling extensions
chromeOptions.addArguments("--disable-gpu"); // applicable to windows os only
chromeOptions.addArguments("--disable-dev-shm-usage"); // overcome limited resource problems
chromeOptions.addArguments(“--no-sandbox");

driver = new RemoteWebDriver(new URL("http:localhost:4444/wd/hub"),chromeOptions);
```

- Install the latest LTS version: `brew install jenkins-lts`
- Install a specific LTS version: `brew install jenkins-lts@YOUR_VERSION`
- Start the Jenkins service: `brew services start jenkins-lts`
- Restart the Jenkins service: `brew services restart jenkins-lts`
- Update the Jenkins version: `brew upgrade jenkins-lts`

https://www.jenkins.io/doc/book/installing/windows/
https://git-scm.com/downloads

export MAVEN_HOME=/usr/local/Cellar/maven/3.6.2/libexec
export PATH=$PATH:$MAVEN_HOME/bin
cd /Users/cb-sarathkumar/work/cb-ui-middleware/GensparkMavenUI
mvn test

Rest API **https://www.google.co.in/search?q=toolsqa**

URI - https://www.google.co.in/
Path Params: search
Request Params: q=toolsqa

Session-1
---
Agenda
---
1) What is Client & Server?
2) 3 Layers of Application
3) Client Server architecture
4) What is API & Webservice
---

Basic concepts of API & webservices...

High level overview on topics which we are going to learn

Client & Server

3 layers of application
---
1) Front end(Presentation layer)
2) Application Layer(Business Logic)
3) Backend (Data Layer)

Client/Server architecture
---
1- tier (PL, DL --> Single)
2- tier ( PL, DL --> 2 systems)
3 - tier (PL, BL, DY --3 OR N systems)

API & Webservices
---

API --> Application Programming Interface (Its not a webservice)
Webservices --> also a kind of API's available on webservers, we need to access them by using URL.

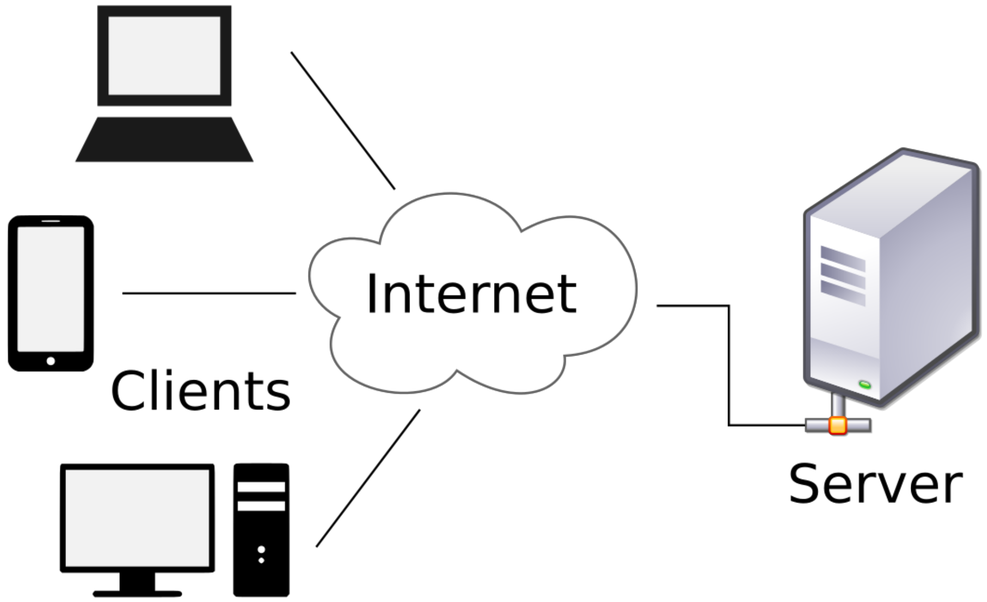
Web services
---
1) SOAP ---> First, older  XML
2) Rest --> More popular  JSON,XML,HTML

1) Basic concepts (Theory)
2) Postman (Rest & SOAP)
3) SOAPUI (SOAP & Rest)

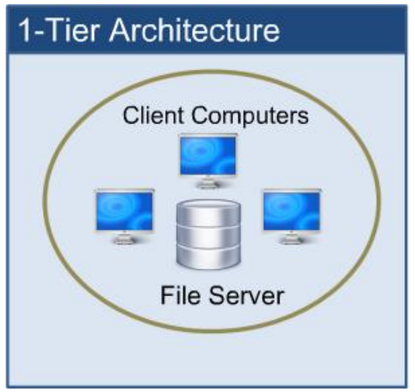
4) RestAssured API (Automation)
Java,TestNG,Maven,Jenkins

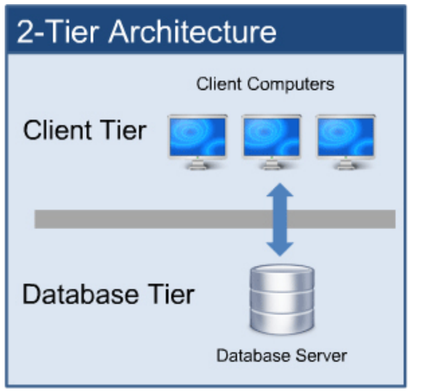
# What is Client and Server?

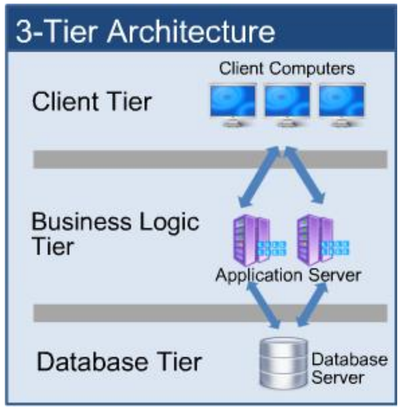
- What is a Client?
- A client is a computer hardware device or software that accesses a service made available by a server. The server is often (but not always) located on a separate physical computer.
- What is a Server?
- A server is a physical computer dedicated to run services to serve the needs of other computers. Depending on the service that is running, it could be a file server, database server, home media server, print server, or web server.



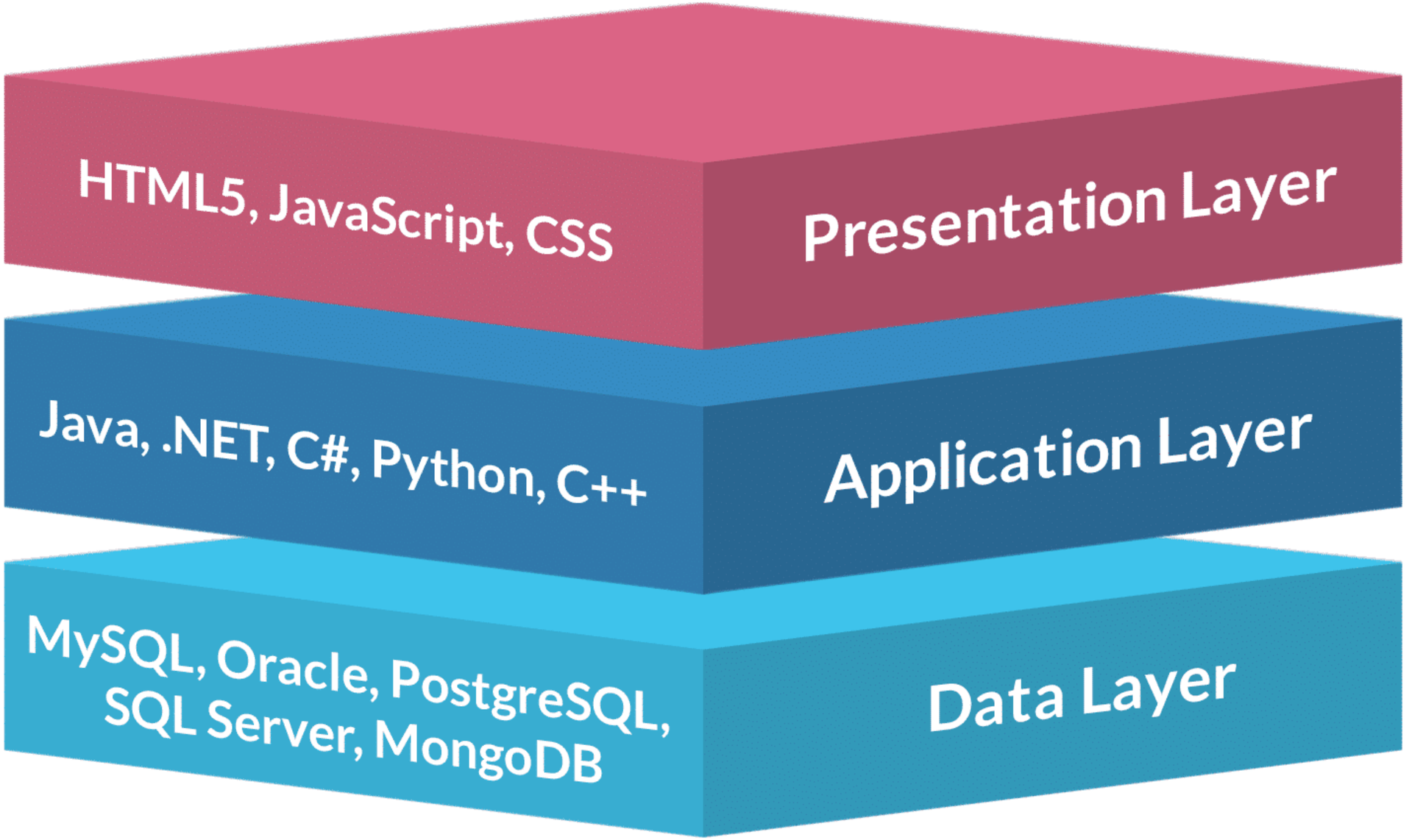
# Client/Server Architecture



Client/Server Architecture

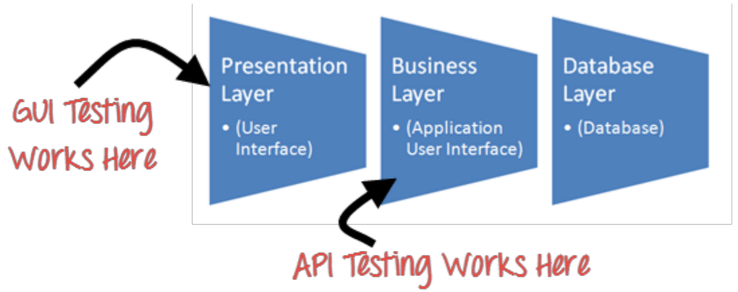


# What is an API?

- API is means **A**pplication **P**rogramming **I**nterface.
- It enables communication and data exchange between two separate software systems.
- A software system implementing an API contains functions/sub-routines which can be executed by another software system.

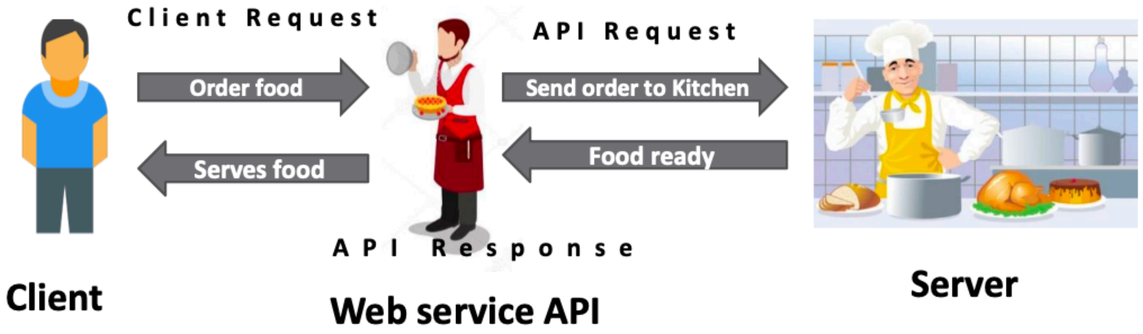
# What is API testing?

- API Testing is entirely different from GUI Testing and mainly concentrates on the business logic layer of the software architecture. This **testing won't concentrate on the look and feel of an application.**
- Instead of using standard user inputs(keyboard) and outputs, in API Testing, you use software to send calls to the API, get output, and note down the system's response.
- API Testing requires an application to interact with API. In order to test an API, you will need to
  - Use Testing Tool to drive the API
  - Write your own code to test the API



# What is Web Service?

- What is a Web Service: Web Service - service available over the web
- Enables communication between applications over the web
- Provides a standard protocol/format for communication
- **Why we use it?**
- Platform independent communication - using web services two different applications (implementation) can talk to each other and exchange data/information



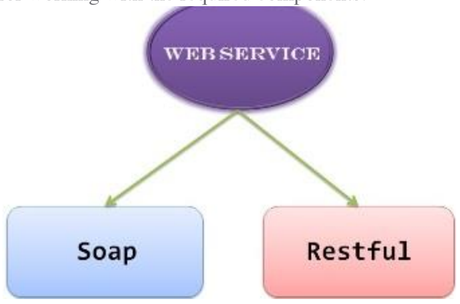
# Difference between API & Web service

1. Web Service is an API wrapped in HTTP.
2. All Web Services are API but APIs are not Web Services.
3. Web Service might not perform all the operations that an API would perform.
4. A Web Service needs a network while an API doesn't need a network for its operation.

# Types of Web Services

- There are mainly two types of web services.
  - SOAP web services. (Simple Object Access Protocol)
  - RESTful web services. (**Re**presentational **S**tate **T**ransfer

- **SOAP** (Simple Object Access Protocol) – SOAP is a protocol which was designed before REST and came into the picture. The main idea behind designing SOAP was to ensure that programs built on different platforms and programming languages could exchange data in an easy manner.
- **REST** – This was designed specifically for working with components such as media components, files, or even objects on a particular hardware device. Any web service that is defined on the principles of REST can be called a RestFul web service.
  - A Restful service would use the normal HTTP verbs of GET, POST, PUT and DELETE for working with the required components.



Day-2

---

Components of webservices

1) WSDL (Web Services Description Language)

2) UDDI (Universal Description Discovery and Integration)

SOAP & Rest services

---

SOAP will use XML request

Rest will use HTTP URL request

API ---> no internet, we can test them locally( Avaialbe in jars)

webservices--> Required internet, we can test them by sending http request(URL)

Tools available

---

Postman

SOAPUI

Jmeter

Katalon studio

etc...

Browsers

API Testing process

---

1) Review & understands Specification

2) Identify how to access, identify different parameters, response etc....

3) Test cases

4) Execute manually/automate test cases then execute

5) reprot defect s to the developer.

Rest Servies

---

Rest services use http protocol....

4 types of HTTP requests

1) POST ---> Create (C)

2) GET --> Retrieve (R)

3) PUT --> Update(U)

4) DELETE ---> Deleting(D)

CRUD OPerations...

Parts of URL ---> Domain + URI + Query String

Examples.....

https://reqres.in/

HTTP Request

---

https://reqres.in/api/users?page=2

status codes:..

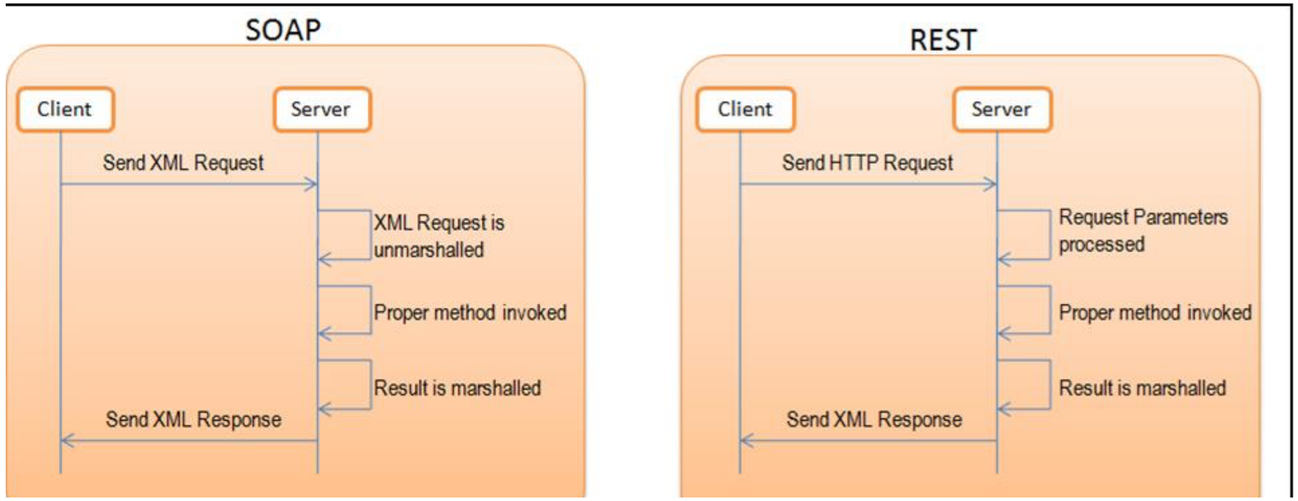
200 --> get successfull

201 --> post succcessful

204 --> delete succesfull

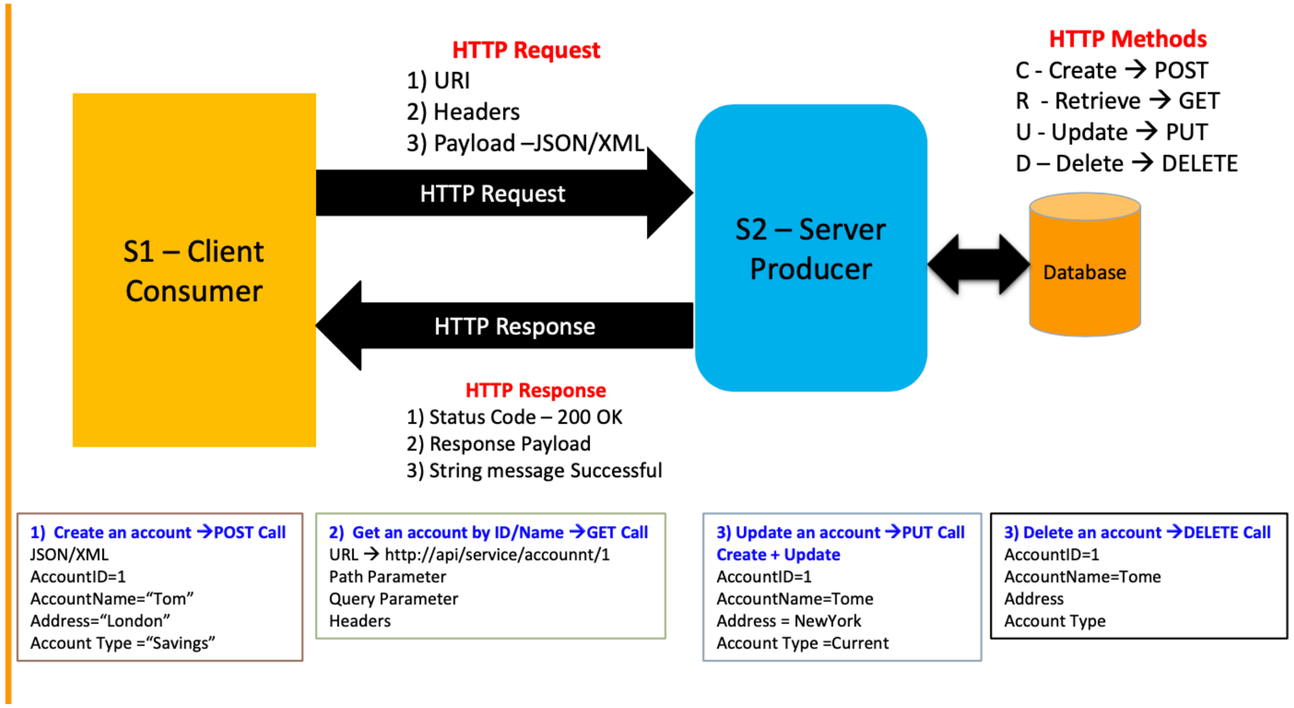
Demo: https://reqres.in/

# SOAP Vs Rest


SOAP
Client
Server
Send XML Request
XML Request is unmarshalled
Proper method invoked
Result is marshalled
Send XML Response
REST
Client
Server
Send HTTP Request
Request Parameters processed
Proper method invoked
Result is marshalled
Send XML Response

# RESTful web services

- REpresentational State Transfer
- A service represents a resource that can be accessed from the web
- Commonly makes use of different HTTP request methods to implement CRUD functions:
- POST – Create new resource
- GET – Retrieve resource
- PUT – Update resource
- DELETE – Delete resource




https://reqres.in

https://documenter.getpostman.com/view/6961294/S1TVXyAS

https://gorest.co.in/ https://dummy.restapiexample.com/

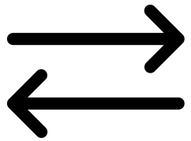
# What is an API?

**A**pplication **P**rogramming **I**nterface

# Application Programming Interface

**Client**

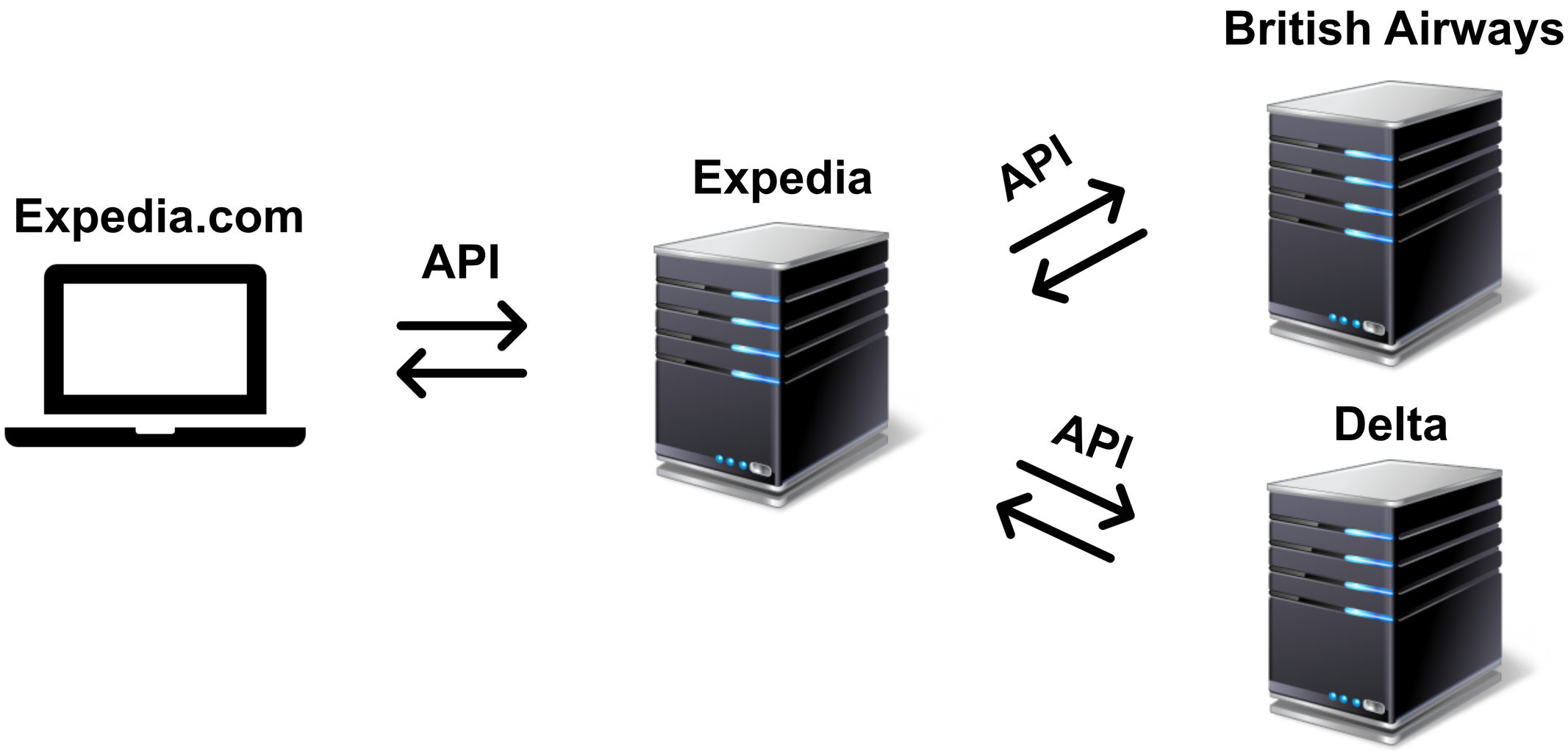
**API**



**Server**


Backend
Frontend
API's

# Application Programming Interface



# Social Login

LinkedIn

Make the most

Email or phone nu

Password (6 or m

By clicking Agr
Agreemer

Alrea

Get Started with Postm

Username

Email

Password

Passwords need to be at least 7 charact

Sign me up to get product updat
communications.

Create A

Sign Up W

By creating an account, I agree to the Term

Spotify

To continue, lo

CONTINUE W

CONTINU

CONTINUE

CONTINUE WITH

Email address or username

Email address or username

Password

Password

Forgot your password?

Remember me

Domino's Pizz

Login to unlock awesome new fe

GREAT FOOD

GREAT OFFERS

Login with your valid mobile num

+91

Mobile Number

Login with social accounts

FACEBOOK

KAYAK

Sign in and save

Track prices, organise travel plans and access member-only deals with your KAYAK account.

Continue with email

or

Booking.com

Apple

FB

Google

By signing up you accept our terms of use and privacy policy.

# What is REST API?

# Representational State Transfer

# REST Constraints

- **Client Server**
- **Stateless**
- **Cache**
- **Uniform Interface**
- **Layered System**
- **Code on Demand**

# What is JSON?

# JSON

- **JavaScript Object Notation**
- **Lightweight**
- **Human Readable**
- **Easy to understand**
- **Key – Value pairs**

# JSON

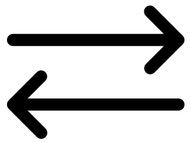
```
{
    "id": "1",
    "firstName": "Omprakash",
    "lastName": "Chavan",
    "email": "askomdch@gmail.com
"
}
```

# JSON

**Client**

**HTML**
**CSS**
**JavaScript**

**GET /user/1**



```
{
    "id": "1",
    "firstName": "Omprakash",
    "lastName": "Chavan",
    "email": "askomdch@gmail.co
m"
}
```

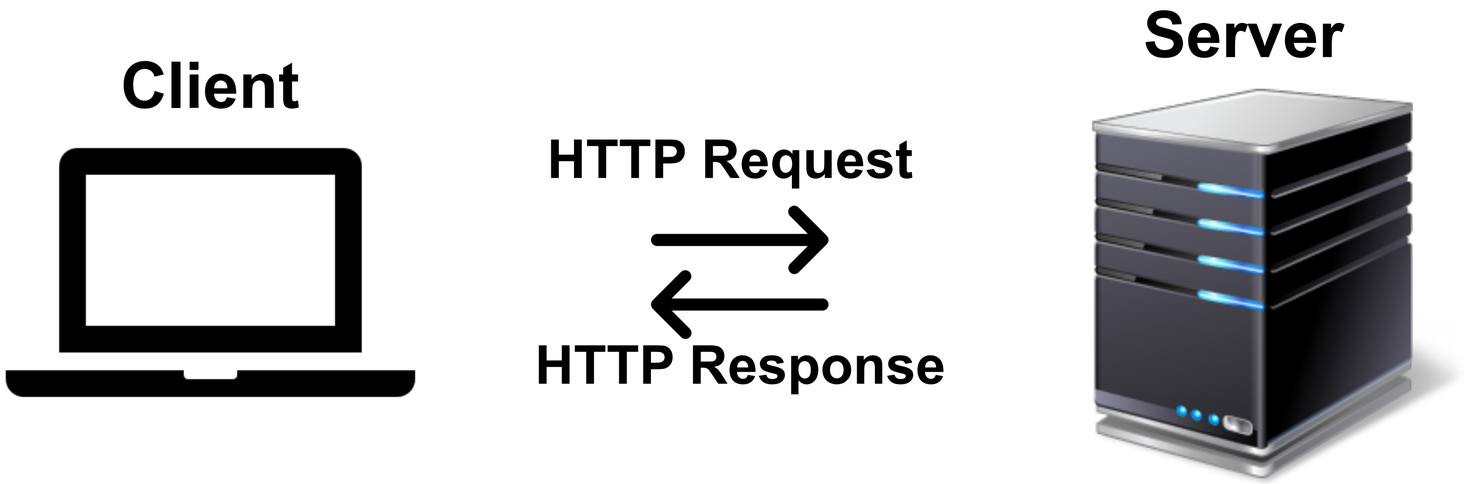
**Server**

**Java**
**Python**
**PHP**

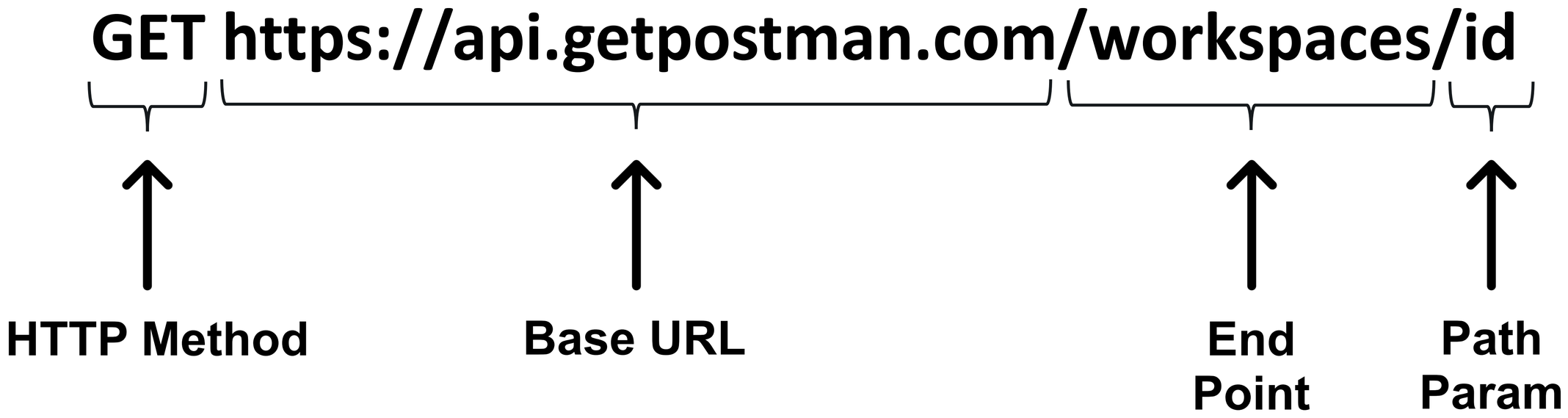
# HTTP Vs REST

# Hypertext Transfer Protocol

# Client Server Architecture



# Rest API Request



# HTTP Methods – RFC 7231/RFC 5789

| Method | Description | Request body | Response body | Safe | Idempotent | Cacheable |
|---|---|---|---|---|---|---|
| GET | Transfer a current representation of the target resource | No | Yes | Yes | Yes | Yes |
| HEAD | Same as GET, but only transfer the status line and header section | No | No | Yes | Yes | Yes |
| POST | Perform resource-specific processing on the request payload | Yes | Yes | No | No | In some cases |
| PUT | Replace all current representations of the target resource with the request payload | Yes | No | No | Yes | No |
| DELETE | Remove all current representations of the target resource | Optional | Optional | No | Yes | No |
| CONNECT | Establish a tunnel to the server identified by the target resource | No | Yes | No | No | No |
| OPTIONS | Describe the communication options for the target resource | No | Yes | Yes | Yes | No |
| TRACE | Perform a message loop-back test along the path to the target resource | No | No | Yes | Yes | No |
| PATCH | Perform partial modification of the target resource | Yes | Yes | No | No | No |

# HTTP Methods

- **GET**
  - **Retrieves data from a server at the specified resource**
  - **Often the default method for HTTP Clients**
  - **Safe and Idempotent**
  - **Cacheable**

# HTTP Methods

- **HEAD**
  - **Gets the resource but without the response body**
  - **Useful to verify resource availability before making a GET request**
  - **Useful for testing API availability**
  - **Safe and Idempotent**
  - **Cacheable**

# HTTP Methods

- **POST**
  - **Sends data to a server to create or update a resource**
  - **Unsafe and Non-idempotent**

# HTTP Methods

- **PUT**
  - **Sends data to a server to overwrite or create a resource**
  - **Unsafe and Idempotent**

# HTTP Methods

- **PATCH**
  - **Sends data to a server to partially modify a resource**
  - **Unsafe and Non-idempotent**

# HTTP Methods

- **DELETE**
  - **Sends data to a server to delete a resource**
  - **Unsafe and Idempotent**

# HTTP Methods

- **OPTIONS**
  - **Returns the HTTP methods supported by a server for a given URL**
  - **Safe and Idempotent**

```
    <dependency>
        <groupId>io.rest-assured</groupId>
        <artifactId>rest-assured</artifactId>
        <version>4.4.0</version>
        <scope>compile</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple -->
    <dependency>
        <groupId>org.json</groupId>
        <artifactId>json</artifactId>
    </dependency>
```

- Request builder
- Body as string
- Body as map
- Body as json file
-

```
RestAssured.requestSpecification = given().header("Accept", "application/json").header("Content-Type", "application/json")
        .header("Authorization", "Bearer ae11c5df458b41fbb06cc4bda6881b53f920f3f47cc38cf18251b28fa5b56e90")
        .log().all();
responseSpecification = new ResponseSpecBuilder().log(LogDetail.ALL).build();
```

- To log the req and response

```
PrintStream fileInputStream = new PrintStream(new File("logger.log"));
RestAssured.baseURI = "https://gorest.co.in/public/v1/";
RestAssured.requestSpecification = given().header("Accept", "application/json").header("Content-Type", "application/json")
        .header("Authorization", "Bearer ae11c5df458b41fbb06cc4bda6881b53f920f3f47cc38cf18251b28fa5b56e90")
        .filter(new RequestLoggingFilter(fileInputStream))
        .filter(new ResponseLoggingFilter(fileInputStream))
        .log().all();
```

```java
import io.restassured.RestAssured;
import io.restassured.builder.ResponseSpecBuilder;
import io.restassured.filter.log.LogDetail;
import io.restassured.filter.log.RequestLoggingFilter;
import io.restassured.filter.log.ResponseLoggingFilter;
import models.Data;
import models.User;
import org.json.JSONException;
import org.json.JSONObject;
import org.skyscreamer.jsonassert.JSONAssert;
import org.skyscreamer.jsonassert.JSONCompareMode;
import org.testng.Assert;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintStream;
import java.util.HashMap;

import static io.restassured.RestAssured.*;
import static io.restassured.module.jsv.JsonSchemaValidator.matchesJsonSchemaInClasspath;

public class GoRestTest {

    @BeforeTest
    public void setUp() throws FileNotFoundException {
        PrintStream fileInputStream = new PrintStream(new File("logger.log"));
        RestAssured.baseURI = "https://gorest.co.in/public/v1/";
        RestAssured.requestSpecification = given().header("Accept", "application/json").header("Content-Type", "application/json")
                .header("Authorization", "Bearer ae11c5df458b41fbb06cc4bda6881b53f920f3f47cc38cf18251b28fa5b56e90")
                .filter(new RequestLoggingFilter(fileInputStream))
                .filter(new ResponseLoggingFilter(fileInputStream))
                .log().all();
        RestAssured.responseSpecification = new ResponseSpecBuilder().log(LogDetail.ALL).build();
    }

    @Test
    public void testListUsers() {
        String responseBody = get("users")
                .then().spec(responseSpecification).extract().response().asString();
    }

    @Test
    public void testCreateUser() throws JSONException {
        String requestBody = "{\"name\":\"Tenali Ramakrishna\", \"gender\":\"male\", \"email\":\"tenali123.ramakrishna" + System.currentTimeMillis() + "@gmail.com\", \"status\":\"active\"}";
        HashMap<String, Object> reqMap = new HashMap<>();
        reqMap.put("gender", "male");
        reqMap.put("email", "tenali123.ramakrishna" + System.currentTimeMillis() + "@gmail.com");
        reqMap.put("name", "Tenali Ramakrishna");
        reqMap.put("status", "active");
        String responseBody = requestSpecification.body(reqMap)
                .when().post("users")
                .then().spec(responseSpecification).assertThat().statusCode(201).body(matchesJsonSchemaInClasspath("jsonSchema.json")).extract().response().asString();
        JSONObject jsonObject = new JSONObject(responseBody);
        JSONObject data = jsonObject.getJSONObject("data");
        String name = data.getString("name");
        Assert.assertEquals("Tenali Ramakrishna", name);
    }

    @Test
    public void testUpdateUser() throws JSONException, JsonProcessingException {
//        String requestBody = "{\"name\":\"Tenali Ramakrishna\", \"gender\":\"male\", \"email\":\"tenali123.ramakrishna" + System.currentTimeMillis() + "@gmail.com\", \"status\":\"active\"}";
        User user = new User();
        Data reqData = new Data();
        reqData.setEmail("tenali123.ramakrishna164q21119163439426@gmail.com");
        reqData.setName("tenali123.ramakrishna164191163439426@gmail.com");
        reqData.setGender("male");
        reqData.setStatus("active");
       reqData.setId(11);


        user = requestSpecification.body(reqData)
                .when().post("users")
                .then().spec(responseSpecification).assertThat().statusCode(201).extract().response().as(User.class);

        System.out.println(user.toString());
        ObjectMapper mapper = new ObjectMapper();
        String request = mapper.writeValueAsString(reqData);
        String response = mapper.writeValueAsString(user.getData());

        JSONAssert.assertEquals(request,response, JSONCompareMode.STRICT_ORDER);
```

```java
JSONAssert.assertEquals(request, response,
        new CustomComparator(JSONCompareMode.LENIENT,
            new Customization("id", new ValueMatcher<Object>() {
                public boolean equal(Object o, Object t1) {
                    return false;//true
                }
            })
        ));
```

```java
//        JSONObject jsonObject = new JSONObject(responseBody);
//        JSONObject data = jsonObject.getJSONObject("data");
//        int userID = data.getInt("id");
//        String requestBody = "{\"name\":\"Sarat\", \"gender\":\"male\", \"email\":\"tenali123.ramakrishna" + System.currentTimeMillis() + "@gmail.com\", \"status\":\"active\"}";
//        responseBody = requestSpecification
//                .body(requestBody)
//                .log().all()
//                .when().patch("users/" + userID)
//                .then().log().all().assertThat().statusCode(200).extract().response().asString();
```

https://www.baeldung.com/jsonassert

```java
package models;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON_NULL)
public class Data {
    private String email;
    private String name;

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    private String gender;
    private String status;
    private int id;
}
```

```java
package models;

public class User {
    private String meta;

    public String getMeta() {
        return meta;
    }

    public void setMeta(String meta) {
        this.meta = meta;
    }

    public Data getData() {
        return data;
    }

    public void setData(Data data) {
        this.data = data;
    }

    private Data data;

    public String toString(){
        return data.getEmail()+" "+ data.getName();
    }
}
```

# Authentication/Authorization

**Authentication →**

Who you are.
Proves your identity.

**Authorization →**

What you can do.
Proves your right to access.

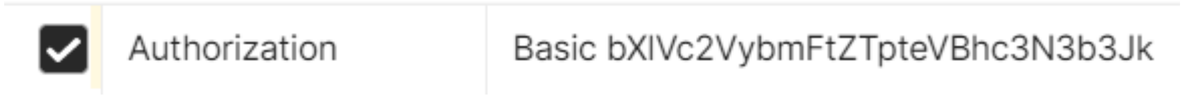
# HTTP Authentication Schemes

- **Basic**
- **Bearer**
- **Digest**
- **OAuth**

# Basic

- Base64 encoded *username:password* in the header
- Base64 encoding is not secure

Example:




**Basic Auth:**

```
String usernameColonPassword = "myUsername:myPassword";

String base64Encoded = Base64.getEncoder().encodeToString(usernameColonPassword.getBytes());
System.out.println("Encoded = " + base64Encoded);
byte[] decodedBytes = Base64.getDecoder().decode(base64Encoded);
System.out.println("Decoded = " + new String(decodedBytes));
```

# Bearer

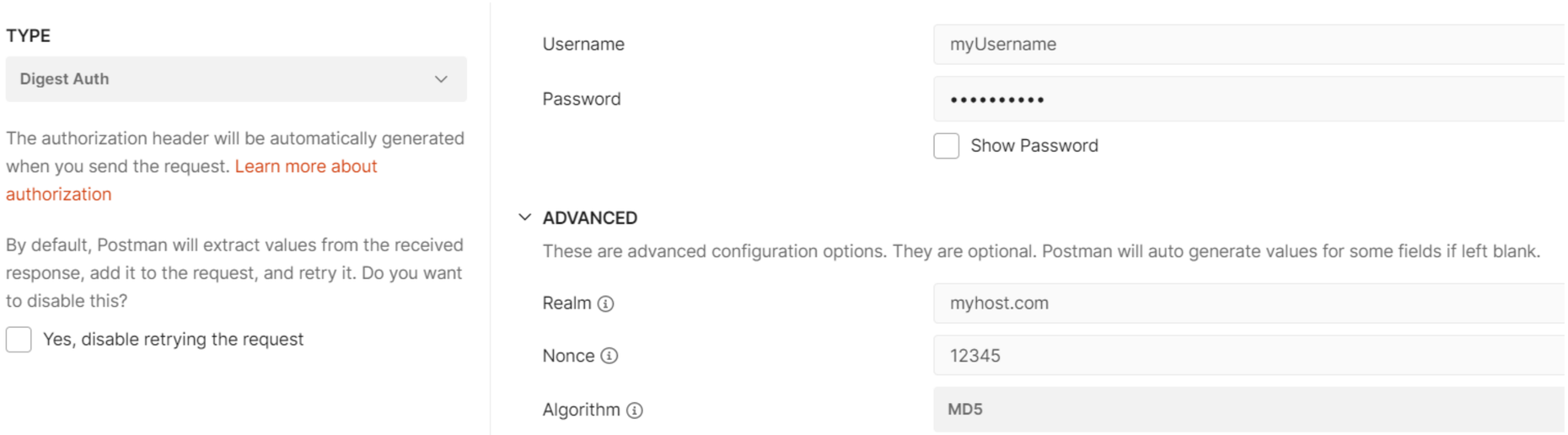
- Bearer token
- Bearer is a person or an entity who holds a security token to get access to a certain resource
- Cryptic: Generated by the server in response to login
- Originally created as part of OAuth2.0 spec

Example:

# Digest

- Challenge-response paradigm
- Username and password is hashed using MD5 algorithm

## Example:

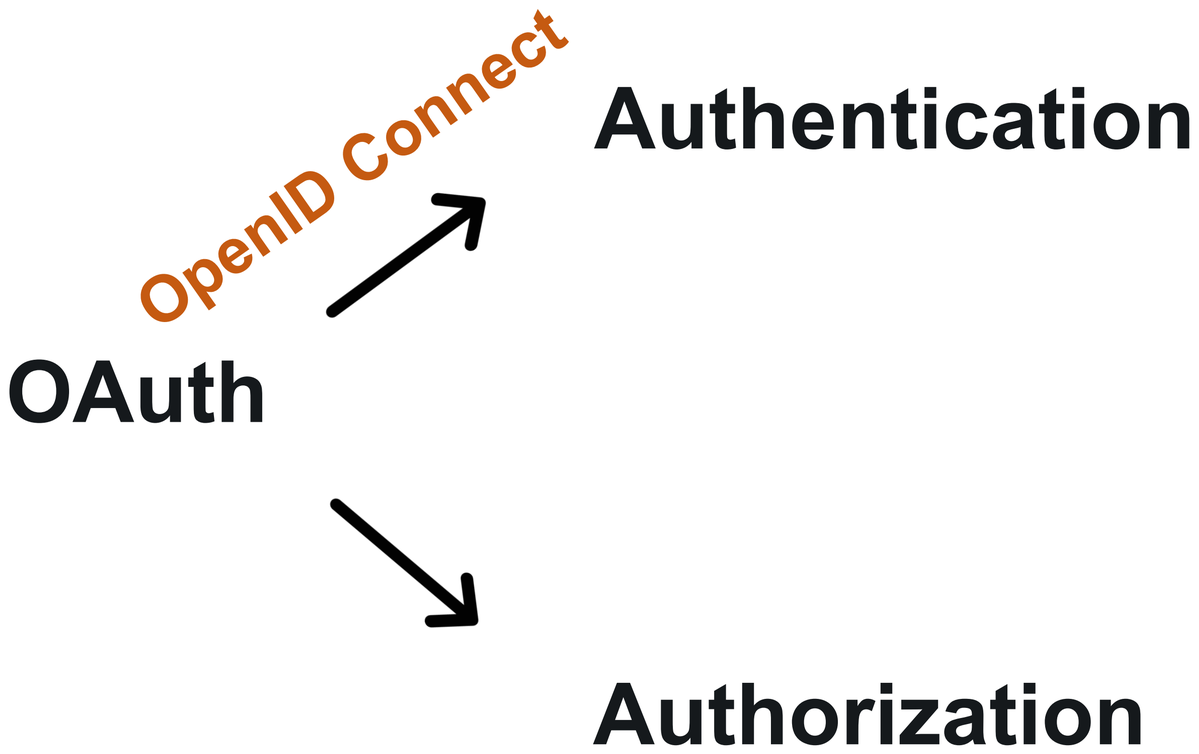


# API Key

- Usually generated during first time login or during sign up
- Used as a replacement for username and password
- Usually fetched from account settings and often it is possible to delete, regenerate and create multiple API keys
- Passed as a header or a query parameter or even in the request body.
- Not secure

Example:

# OAuth



# OAuth

**Why do we need OAuth?**

# OAuth

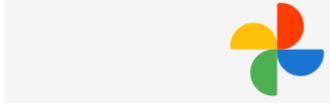
**Delegated Authorization**

# OAuth

Login

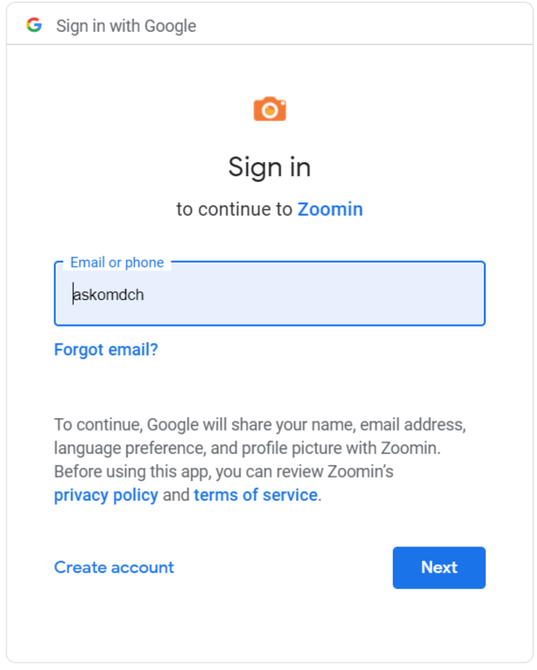
zoomin

Add Photos

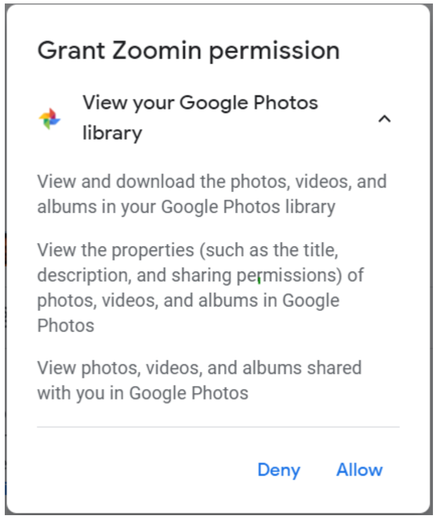


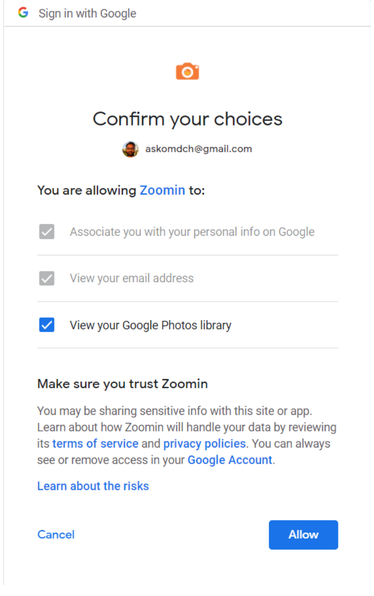
Please authenticate with Google Photos to select files

Connect to Google Photos




Login to Google




Allow

zoomin

# OpenID Connect + OAuth

LinkedIn

Make the most of your professional life

Email or phone number

Password (6 or more characters)

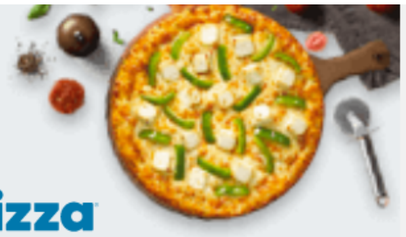
By clicking Agree & Join, you agree to the LinkedIn User Agreement, Privacy Policy, and Cookie Policy.

Agree & Join

or

Join with Google

Already on LinkedIn? Sign in



Domino's Pizza

**Login** to unlock awesome new features

GREAT FOOD

GREAT OFFERS

EASY REORDERING

Login with your valid mobile number

Mobile Number

+91

SUBMIT

Login with social accounts

FACEBOOK

GOOGLE

Spotify

Sign up for free to start listening.

SIGN UP WITH FACEBOOK

or

What's your email?

Enter your email.

Use phone number instead.

Confirm your email

Enter your email again.

# OAuth Terminologies

#Client

zoomin

#Resource Owner

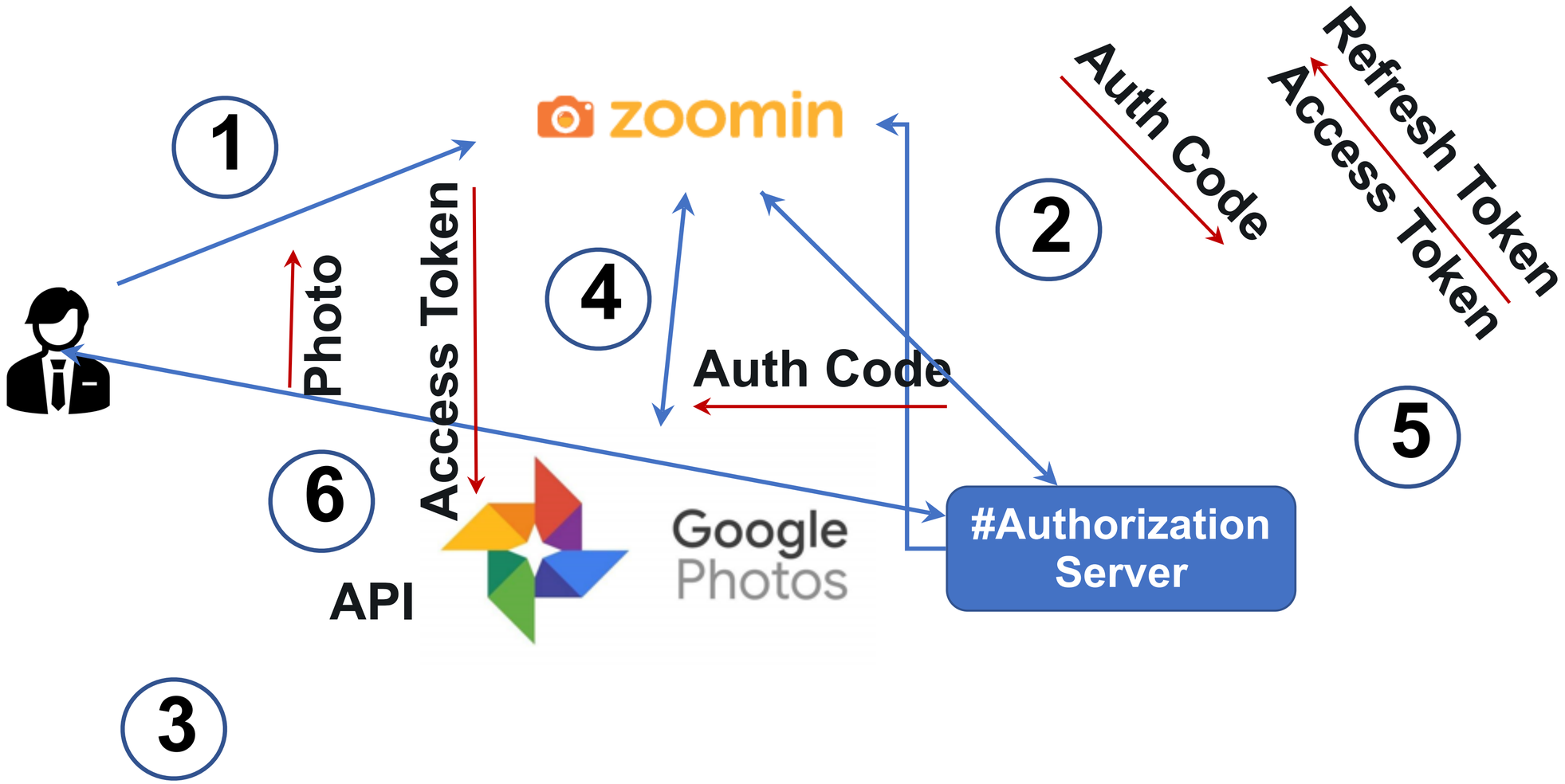
#Resource

Google Photos

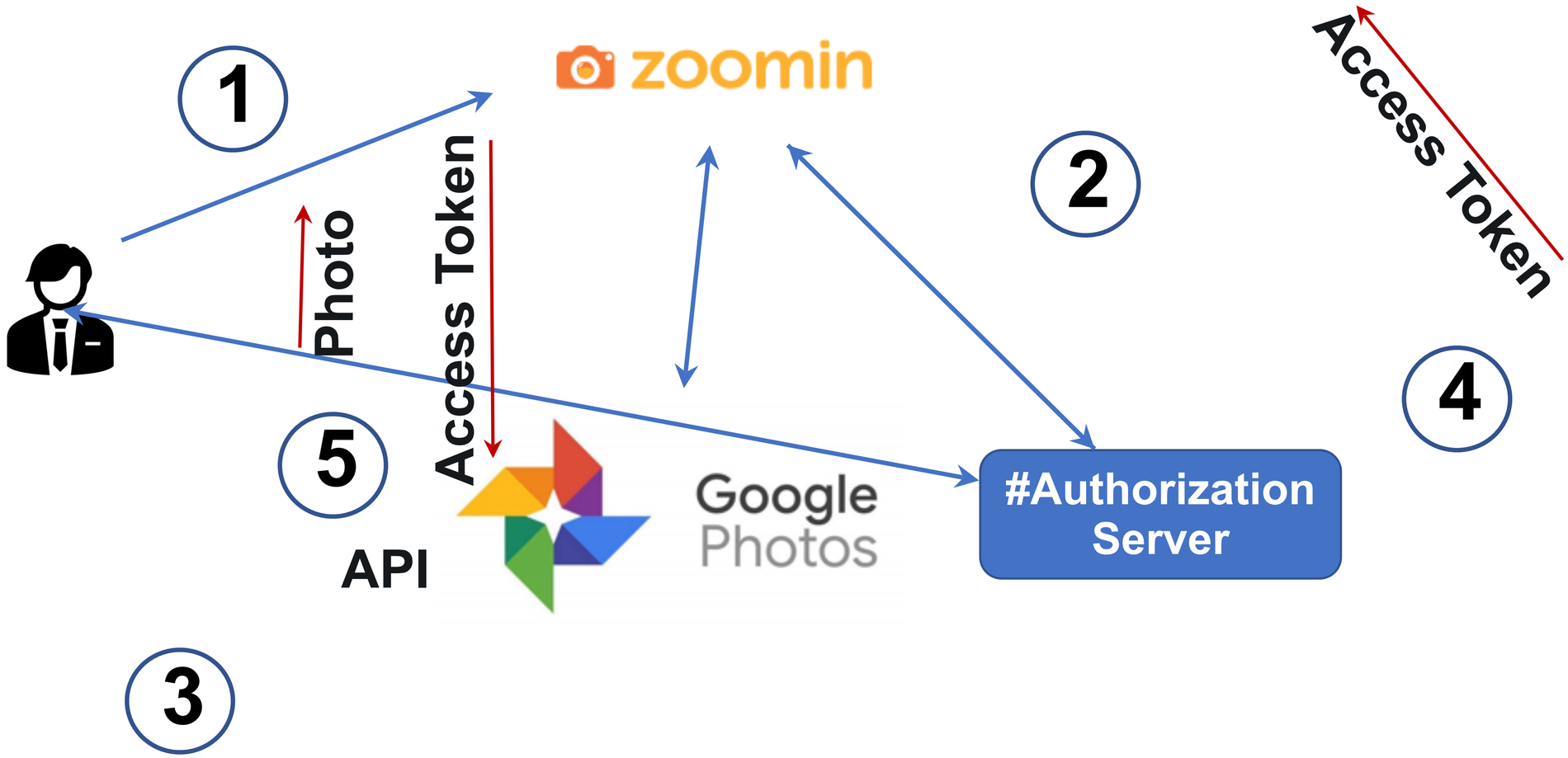
#Authorization Server

#Resource Server

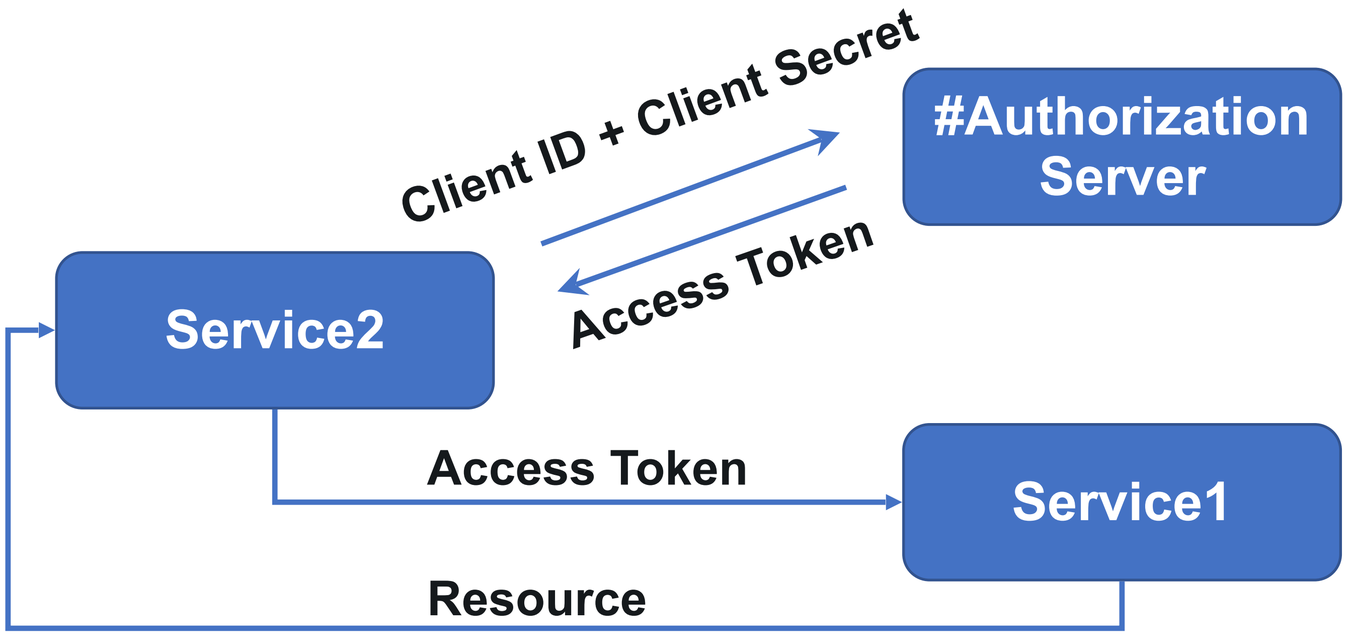
# OAuth – Authorization Grant Flow



# OAuth – Implicit Grant Flow



# OAuth – Client Credentials Flow



## Framework Goals:

- ✓ Scalable and extensible
- ✓ Reusable Rest Assured specifications
- ✓ Reusable Rest Assured API requests
- ✓ Separation of API layer from test layer
- ✓ POJOs for Serialization and Deserialization
- ✓ Singleton Design Pattern
- ✓ Lombok for reducing Boilerplate code
- ✓ Builder pattern for Setter methods in POJOs
- ✓ Robust reporting and logging using Allure
- ✓ Automate positive and negative scenarios
- ✓ Support parallel execution
- ✓ Data driven using TestNG Data Provider
- ✓ Automated access token renewal
- ✓ Maven command line execution
- ✓ Integration with Git
- ✓ Integration with Jenkins

## Tools and Technologies

- Rest Assured
- TestNG
- Java
- Allure Reports
- Hamcrest
- Jackson API
- Lombok

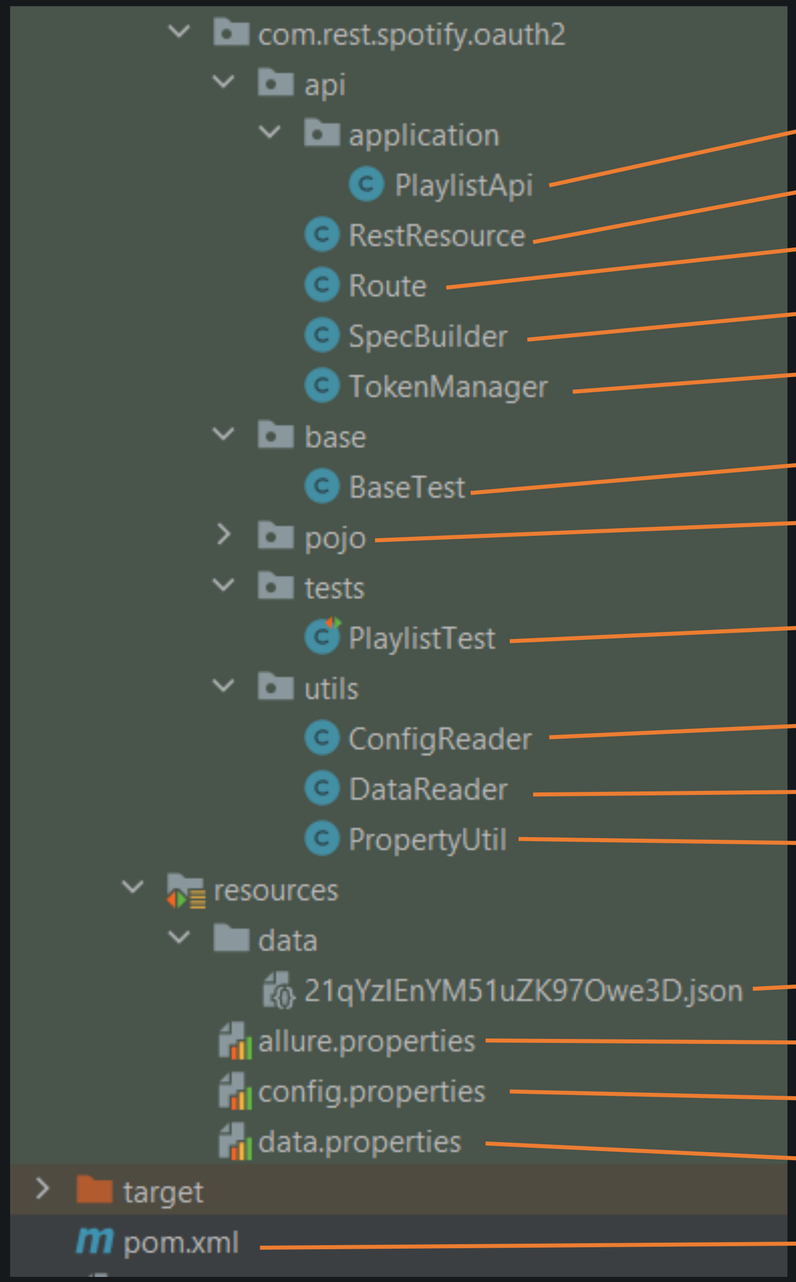
# What are we automating?

Spotify Playlists API using the OAuth 2.0 flow [Authorization Code Grant Flow]

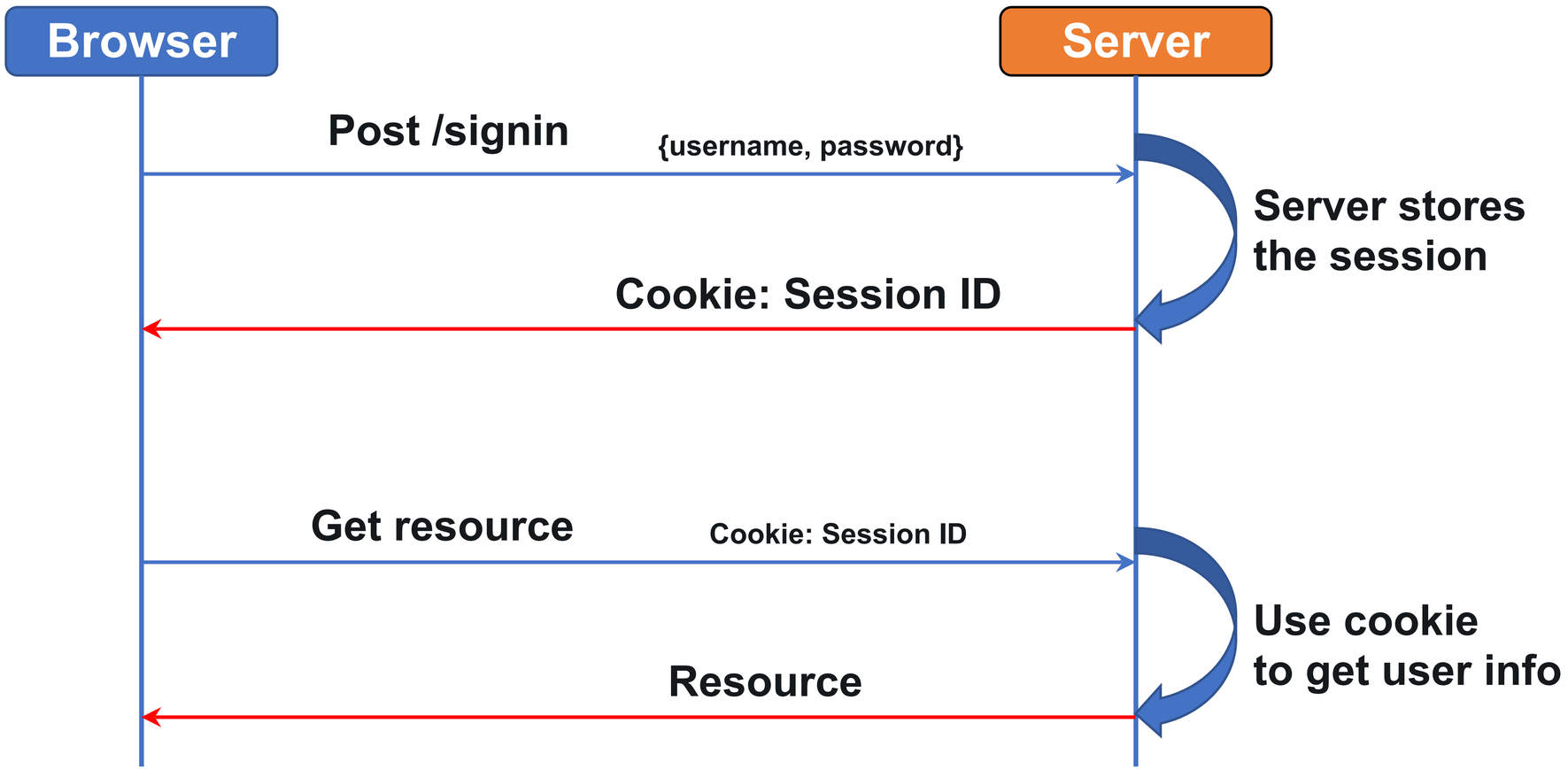
- ✓ Create a Playlist
- ✓ Get a Playlist
- ✓ Change a Playlist's Details

Test Cases:

- ✓ shouldBeAbleToCreateAPlaylist
- ✓ shouldNotBeAbleToCreateAPlaylistWithoutNam e
- ✓ shouldNotBeAbleToCreateAPlaylistIfTokenIsEx pired
- ✓ shouldBeAbleToFetchAGivenPlaylist
- ✓ shouldBeAbleToUpdateAGivenPlaylist


com.rest.spotify.oauth2
api
application
PlaylistApi
RestResource
Route
SpecBuilder
TokenManager
base
BaseTest
pojo
tests
PlaylistTest
utils
ConfigReader
DataReader
PropertyUtil
resources
data
21qYzIEnYM51uZK97Owe3D.json
allure.properties
config.properties
data.properties
target
pom.xml
Playlist API reusable methods
Rest Assured API reusable methods
Playlist end points
Reusable request and response specifications
Access token expiry check and renewal
TestNG Base test
POJO classes
Test class
Singleton config.properties reader
Singleton data.properties reader
Common property loader
JSON data
Allure properties
Config properties
Data properties
Dependency management

# Session based Authentication



# HTTP Cookie

**What is a Cookie?**

- Small piece of data stored in the browser storage

**Designed to,**

Authenticate the user – logged in or not?
Maintain the user session/state
Record user's browsing activity
Remembering user information

**Used for,**

Personalization
Session Management
Tracking

**Terminologies**

- Session Cookie
- Persistent Cookie
- Secure Cookie
- http-only Cookie
- Same site Cookie

**Setting a Cookie**

```
Set-Cookie: JSESSIONID=911642A574B571E6B1EFD79F92ACD064;
```

**Sending a Cookie**

```
Cookie: JSESSIONID=911642A574B571E6B1EFD79F92ACD064
```

# Unit Test - MockiTo

```java
package business;

import service.TodoService;

import java.util.ArrayList;
import java.util.List;

public class TodoBusinessImpl {
    private TodoService todoService;

    public TodoBusinessImpl(TodoService todoService) {
        this.todoService = todoService;
    }

    public List<String> retrieveTodosRelatedToSpring(String user) {
        List<String> filteredTodos = new ArrayList<String>();
        List<String> allTodos = todoService.retrieveTodos(user);
        for (String todo : allTodos) {
            if (todo.contains("Spring")) {
                filteredTodos.add(todo);
            }
        }
        return filteredTodos;
    }
}
```

```java
package service;

import java.util.List;

public interface TodoService {
    public List<String> retrieveTodos(String user);
}
```

```java
import business.TodoBusinessImpl;
import org.junit.Test;
import service.TodoService;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

public class FirstMockitoTest {
    @Test
    public void test() {
        TodoService todoService = mock(TodoService.class);
        List<String> allTodos = Arrays.asList("Learn Spring MVC",
                "Learn Spring", "Learn to Dance");
        when(todoService.retrieveTodos("test")).thenReturn(allTodos).thenReturn(new ArrayList<>());
        TodoBusinessImpl todoBusinessImpl = new TodoBusinessImpl(todoService);
        List<String> todos = todoBusinessImpl
                .retrieveTodosRelatedToSpring("test");
        assertEquals(2, todos.size());
        assertEquals(2, todos.size());
    }

}
```

```xml
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-all</artifactId>
    <version>1.10.19</version>
    <scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
</dependency>
```

# Annotation

```java
import business.TodoBusinessImpl;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.runners.MockitoJUnitRunner;
import service.TodoService;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.when;

@RunWith(MockitoJUnitRunner.class)
public class FirstMockitoTest {
    @Mock
    TodoService todoServiceMock;

    @InjectMocks
    TodoBusinessImpl todoBusinessImpl;


    @Test
    public void test() {
        List<String> allTodos = Arrays.asList("Learn Spring MVC",
                "Learn Spring", "Learn to Dance");
        when(todoServiceMock.retrieveTodos("test")).thenReturn(allTodos).thenReturn(new ArrayList<String>());
        List<String> todos = todoBusinessImpl
                .retrieveTodosRelatedToSpring("test");
        assertEquals(2, todos.size());
        assertEquals(2, todos.size());
    }

}
```

# Cucumber

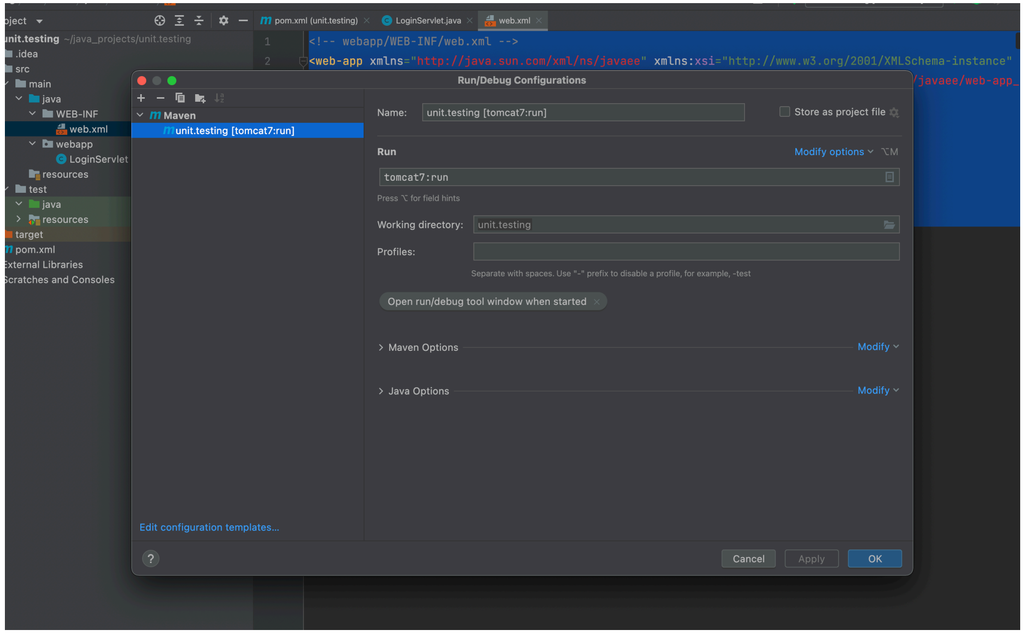
```
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>1.2.2</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>1.2.2</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.0.1</version>
</dependency>
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-core</artifactId>
    <version>1.2.5</version>
    <scope>test</scope>
</dependency>
```

```
Feature: Login Feature

  Scenario: Verify valid credentials
    Given :Open the browser
```

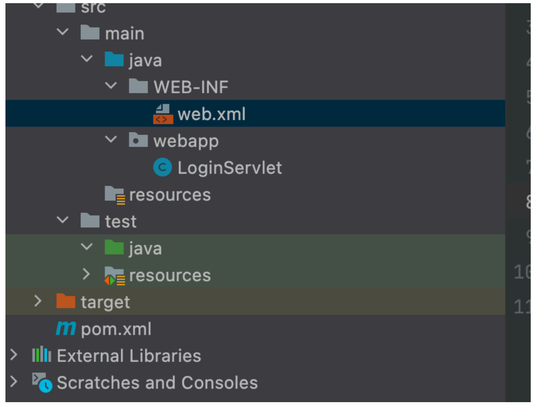
```
package stepDefs;

import cucumber.api.java.en.Given;

public class Steps {

    @Given("^:Open the browser$")
    public void open_the_browser() {
        // Write code here that turns the phrase above into concrete actions
    }

}
```

```
package runner;

import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(features = "src/test/resources/features",glue = "stepDefs")
public class Runner {
}
```

# Servlet




```
<!-- webapp/WEB-INF/web.xml -->
<web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">

    <display-name>To do List</display-name>
    <welcome-file-list>
        <welcome-file>login.do</welcome-file>
    </welcome-file-list>

</web-app>
```



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

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

    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-web-api</artifactId>
            <version>6.0</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
        <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.2</version>
                    <configuration>
                        <verbose>true</verbose>
                        <source>1.7</source>
                        <target>1.7</target>
                        <showWarnings>true</showWarnings>
                    </configuration>
                </plugin>
                <plugin>
                    <groupId>org.apache.tomcat.maven</groupId>
                    <artifactId>tomcat7-maven-plugin</artifactId>
                    <version>2.2</version>
                    <configuration>
                        <path>/</path>
                        <contextReloadable>true</contextReloadable>
                        <ignorePackaging>true</ignorePackaging>
                    </configuration>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>
</project>
```

```java
package webapp;

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

//1. extends javax.servlet.http.HttpServlet
//2. @WebServlet(urlPatterns = "/login.do")
//3. doGet(HttpServletRequest request, HttpServletResponse response)
//4. How is the response created?

@WebServlet(urlPatterns = "/login.do")
public class LoginServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Yahoo!!!!!!!!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("My First Servlet");
        out.println("</body>");
        out.println("</html>");

    }

}
```