Jan 2022 Batch Notes

Jan 2022 Batch Notes:

Trainer Name : Naveen Khunteta (Naveen AutomationLabs)

Open Cart App URLs:

https://www.softaculous.com/demos/OpenCart

https://demo1.opencart3.com/

https://demo.opencart.com/index.php?route=account/login

http://opencart.antropy.co.uk/index.php?route=account/login

https://naveenautomationlabs.com/opencart/

JavaSession -01

Topic : Data_Types

Data Types : There are 8 primitive data types in Java.

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Details	Written By
JavaSession -02 **Topic:** **StringConcat_Incremental_Decremental_BasicMathematicalOperations**	
• String is not a primitive data type., string is a class, which is already available in Java (pre-defined class) • String S = " " always to be written in double quotes only. and default value of string is null. • add two values or merge two values • **Collection of different characters is called String.** Ex:- String x = "Hello"; String y = "Selenium"; System.out.println(x+y); o/p:- HelloSelenium • **Adding integer with string is also String concatenation.** Ex: String x = "Hello"; String y = "Selenium"; int a = 100; int b = 200; System.out.println(a+b+x+y); o/p:- 300HelloSelenium (first integers got added and then got concatenated with String. output will be a new string) • Whenever we are doing arithmetic operators in character we have to follow the ASCII values. **ASCII TABLE:-** a. a to z -> 97 - 122 b. A to Z -> 65 - 90 c. 0 to 9 -> 48 - 57	@anupama **@Naveen AutomationLabs** **@Saranyaa** **@shivom**

String comparison: String is non-primitive data type. Use equals method to compare String variables. Use == operator to compare primitive data types String str1= "Hello"; String str2="Hello"; if(str1.equals(str2)) { System.out.println("Both the strings are same"); }	
1.Post increment:- **i++** assigning the value, then increase the value by 1 2. Post decrement:- **i--** assigning the value, then decrease the value by 1 3. Pre increment:- **++i** increase the value by 1, then assign the value **Trick** : Blindly add 1 to both of the Numbers , H=9 and J = ++H, so the o/p would be 10 & 10. 4. Pre decrement:- **--i** decrease the value by 1, then assign the value Note - Keep the actual value in mind while deducing the value of a decrement/increment eg- a = -99 , a-- or --a - would result in a having value -100 a++ or ++a would result in a having value -98	**@Naveen AutomationLabs** @Anuradha @AkshathaJain
Sorted in ASCII number Byte or int will print within ASCII table reach only. The range of ASCII numbers: a-z: 97-122 A-Z: 65-90 0-9: 48-57 Max number is 122 If there is any mathematical operator involved, then java will only consider ASCII value of it and perform the operation Any whole number divided by 0, Java throws arithmetic exception	@Naveen AutomationLabs @akshathaJain

Numeric floating number 8.9(Float)/0 (Int) or 2(Int)/0.0 (Float)= Java throws infinity (Infinity is considered during floating number) 2/0.0 0/0.0 or 0.0/0.0 or 0.0/0 throws NAN (Not a number) 0/0 Also throws arithmetic exception	
• *We should always use == operator while comparing two primitive data types.* (int, short, byte, long) if(a==b) ....where a and b are both primitive data types. • We never use == operator while comparing two non-primitive data types.(Strings, Arrays, Classes) • *We should use .equals() method to compare two non primitive data types.* String x="abc"; if(x.**equals**(abc)){ } • The main difference between the .equals() and == is - .equals() is a method while "==" is a operator. • == used for reference comparison(Address comparison) while .equals() method is used for content comparison. • == checks if both objects point to same memory location while .equals() mainly comparing the values of objects.	@Amol **@Dhrumil**

JavaSession -03

Topic: ConditionalOperators_IfElse_SwitchCase.

Conditional Operators: *Interview question*	***What is Dead Code?*** The code that is never executed is called Dead code.If the first condition is always satisfied then it will never go to the else part. Java will not give you any error but warning message to remove the code that is occupied in the memory unnecessarily. fb **Ex:1** **if** (*true*) { System.out.println("Hi.."); } **else** {*// dead code* System.out.println("Bye.."); }	@Anita

	---------- **Ex:2** if(false){ System.out.println("Hello");//dead code } else{ System.out.println("Testing"); }	
Conditional Operators :	another example: its purely dependent on flag variable . //we never know where is this flag variable coming from it can be selenium or API or XML file..... boolean flag=true; if(flag){ System.out.prinltn(" Testing"); }else{ System.out.prinltn("developer"); }	@Supriya
Short Circuit Operator(&& , ||)	In short circuit, when any one of the condition is not satisfied then the further expression will not be evaluated. Here the result is clear even before the evaluation of the complete expression and the result is returned. In case of Logical AND(&&), if the first expression is false , then short circuit evaluation happens and the second condition is not executed and returns False. Since in AND, if an expression is False, the outcome will be false. **Ex :** int x = 75 ; y = 400; z=300; if((x>y) && (y>z){ System.out.println("x is the greatest"); } else{ System.out.println("x is not the greatest"); } Note : Here x>y is false and y>z is true hence the second expression is	@Abhay @Roopali @Vibha

not executed since the result would be false and is a short circuit.

In case of Logical OR(||), if the first expression is true, then short circuit evaluation happens and the second condition is not executed and returns True. Since in OR, if at least expression is True, the outcome will be true
If the first expression is False, the the total evaluation happens and there will be no short circuit.

Ex : int x = 400 ; y = 100; z=300;

```
if((x>y) || (y>z){
System.out.println("x is greater");
}
```

Note : Here x>y is true and y>z is false hence it here is a short circuit with true as result.

Ex:2 no short circuit
int a = 100 ; b = 400; c=300;d=60

```
if((a>b)|| (a>c)||(a>d)){
System.out.println("a is greater");
}
```

Note : Here a>b is false and a>c is false hence here there is no short circuit and it has to completely evaluate since a>d is true with true as result.

All Divide By Zero cases

Cases	Results	Example	Tricky examples
Any integer divided by 0	Arithmetic Exception	9/0.	0/0 is also exception
0 divided by Any Integer.	0	0/9=0	
Any integer or Float divided by 0 or 0.0	Infinity	2.5/0,2.5/0.0	9/0.0 = infinity 9.5/0.0=infinity
int divided by int	integer	5/2=2	
Int divided by float or float divide by integer	float	5.0/2=2.5 , 5/2.0=2.5	
Int 0 divided by float 0 or float 0 divided by int 0	NaN- Not a number	0/0.0=naN 0.0/0=naN 0.0/0.0=naN	

@Lavnya

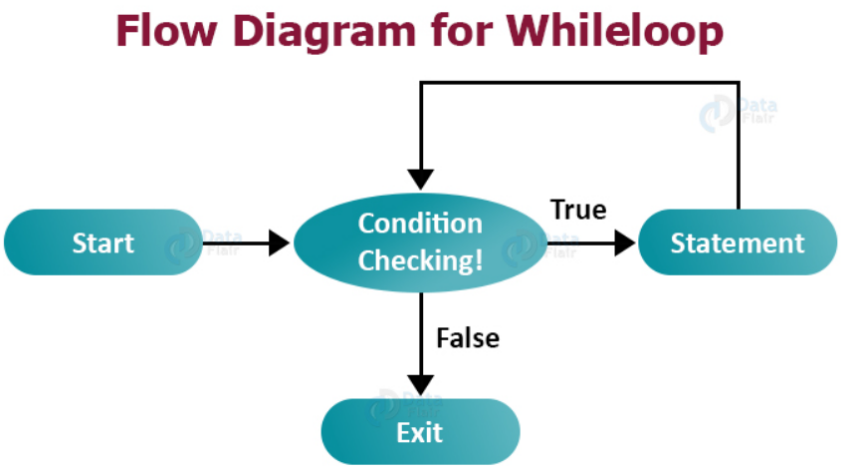
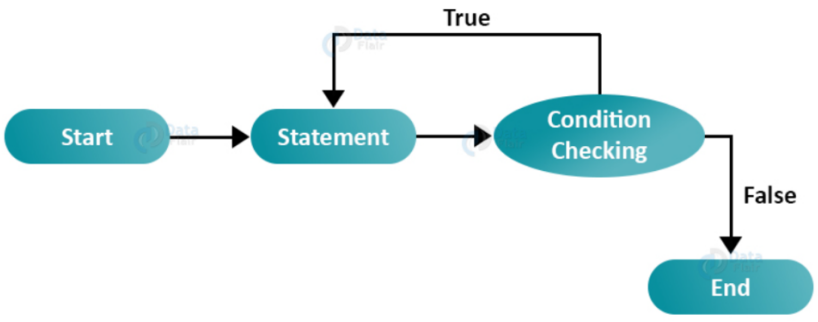
Switch Case - Note - works with Char/Integer /String Keys	1. Limitation of Switch case is, it is only applicable for integer 2. (byte,short,int), string and Character values and cannot be used for Boolean values. 3. Switch Case logic can be best use for Cross browsers, environment selection, user access-based role, Choosing payment options etc 4. Switch case - does not work with Float / Double 5. Switch case does not work with integer type-Long 6. Switch case does not work with boolean 7. Whenever there is no break statement involved in a switch case block. All the statements are executed even if the test expression is satisfied 8. Does not work with variable conditions. 9. We can not use a continue with the switch statement 10. Java is case sensitive language ,every keyword in java should be lower case only. Example: `public void openBrowser(String browserType) {` `switch (browserType) {` `case "firefox":` `driver = new FirefoxDriver();` `break;` `case "chrome":` `driver = new ChromeDriver();` `break;` `case "IE":` `driver = new InternetExplorerDriver();` `break;` `default:` `System.out.println("browser : " + browserType + " is invalid, Launching Firefox as browse r of choice..");` `driver = new FirefoxDriver();` `}` `}`	@Anita @Anuradha @Archana @Siddesha @Anitha @Subhan @Saranyaa @Babita @anupama @Dhrumil
System.out.println()	***int*** **a** = 10 ,**b** = 20; • *System.out.println* (a+b); - Output= **30** • Here '+' sign is behave as Arithmetic Operator • *System.out.println* (a+b+" test data"); Output= "**30test data**" • Here 1st '+' sign is behave as Arithmetic Operator, while 2nd '+' sign Concatenation Operator	@Anuradha @anupama @Abhishek @Roopali

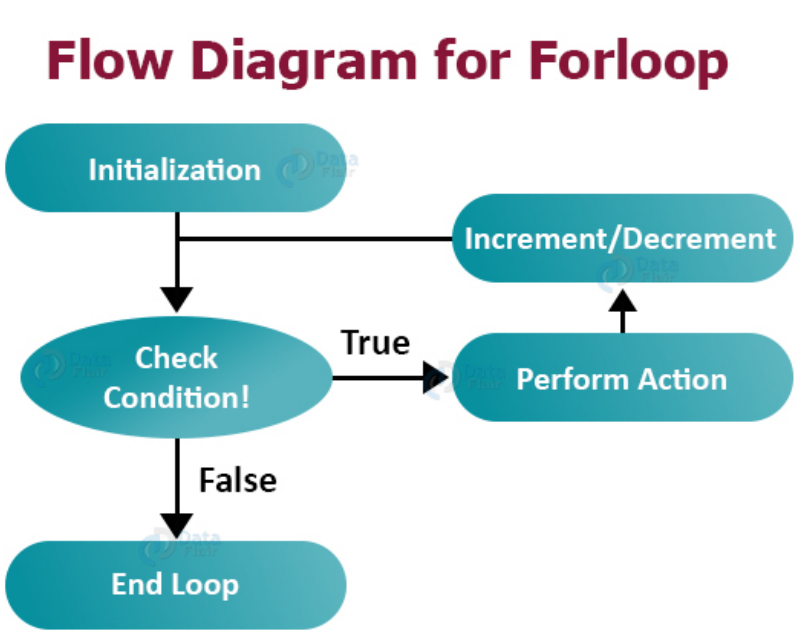
	• *System.out.println* ("test data" + a); Output= "**test data10**" • Here '+' sign is behave as Concatenation Operator • *System.out.println* ("test data" + a+b) Output= "**test data1020**" • Here '+' sign is behave as Concatenation Operator • **Note** - Since execution happens left to right , the placement of the string matters while printing to get the correct data • **System** is a class ***out*** is variable and **println()** is method.	
break;	• '**break**' can be used only with loops or switch cases. • But we can use '**break**' with '**if**' statement in case if it's used inside '**for**' loop. **while**() { **if**() { **break**; } else { } } • break; - It will completely come out of the while loop once the if condition satisfies. • **break;** Statement is a loop control statement that is used to terminate the loop. As soon as the break statement is encountered from within a loop, the loop iterations stop there, and control returns from the loop immediately to the first statement after the loop.	@Abhay @*Amol* @Loknath @Dhrumil

JavaSession -04

Topic: LoopsConcepts_For_While_DoWhile.

Loops	• Repeated steps where the number of repetition is controlled by a condition.	@Kethari
Why do we use loops? ***Types of loops*** ***Elements involved in loops*** ***Syntax***	**Loops:** • Loops are used to execute a certain set of statements repeatedly until it meets the given condition. **Types of Loops:** • for loop • while loop • do while loop **Different components involved:** • Initialization • Condition • Expression (Increment / decrement) • Body of the loop **Syntax:** **for** (initialization; condition; increment / decrement) { //statement to be executed } eg: **for**(int i=10;i<1;i++) { System.out.println(i); } o/p : no o/p since the condition is validated and its false so nothing will be printed. ---------- **while** (condition) { //statement to be executed increment / decrement ; } ---------- **do** { //statement to be executed	@Poorani @Loknath @Akanksha

	increment / decrement ; }**while** (condition); --- ☐ **Important: While writing loop make sure exit condition must be present otherwise it will exhaust the memory and application might crash.**	@Akanksha
Key difference between For loop versus While loop	Use **For** loop when number of iterations is fixed and use **While** loop when number of iterations is unknown. Example - For loop - Menu items on the page - calendar handling while loop - waiting for the element - waiting for the page to load	@Manish @Ujjval
Flow Diagram for ***While loop***		@Manas
Flow Diagram for ***Do While loop***		@Manas

Flow Diagram for ***For loop***		@Manas
For Loop without initialization/condition	**for (; ;){** system.out.println("Hello"); } In such case, ***Hello will be printed infinite times because by default it will assume condition is true.*** **for (; ;){** system.out.println("Hello"); break; } o/p:-Hello will be printed once **Infinity** is the property of loop.	@ Saranya Sakthivel @Simran @Simran
Use Cases for while Loop	1. waiting for element on the page 2. waiting for the page to be loaded 3. pagination - Loop until the element is found (exp- Finding a person name in multiple pages of a web table) 4. For increment operators we can use i++, ++i or i=i+1 5. For Decrement operators we can use i--,--i or i=i-1	@anupama @Saranyaa @Babita
Use Cases for for Loop	1. drop-down traversing 2. menu items 3. calendar handling 4. We use for loop when number of iterations are fixed	@anupama @jeetendra

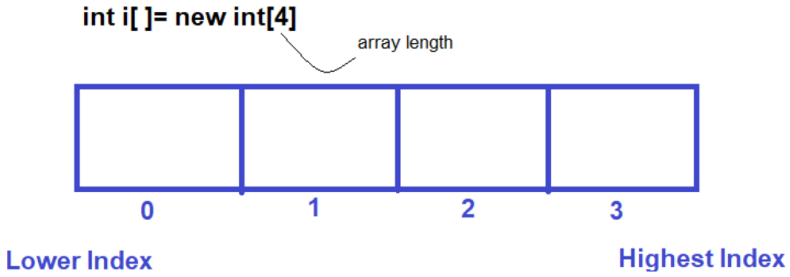
Use Cases for do-while loop	1. The Java do-while loop is used to iterate a part of the program several times. 2. do-while loop will check condition at the end of the block so it will be executed minimum 1 time.	@Zeeshan @Saranyaa
Infinite loop-> while, for & do-while and use cases	while **Ex: 1** int i=1; while(i<=10) { System.out.println(i); } o/p:111111111111..... infinite times loop will be executed. --- **Ex:2** while(true) { System.out.println("Welcome to taj hotel"); } Welcome to taj hotelWelcome to taj hotelWelcome to taj hotelWelcome to taj hotel..... **UseCase:** 24/7 display hotel name --- do while: int p=1; do { System.out.println(p) } while(p<=10); o/p- 111111... --- for: **Example 1:** for(int i=10;i>1;i++) {	@anupama @Saranyaa @Manas @Vibha @Gagan

	System.out.println(i); } **Example 2:** for(;;) { System.out.println("Test"); } By default the condition is taken as true,so enters into infinite loop **Example 3:** Print even numbers from 1 to 100 String evenNumberList = ""; for(int c=1;c<=100;c++) { if(c%2==0) { evenNumberList = (evenNumberList + c +","); } } System.out.println(evenNumberList); o/p:- 2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,78,80,82,84,86,88,90,92,94,96,98,100,	
Why we use for() loop?	1. For loops are used in java to execute statements repeatedly for a given number of times. 2. For loops are used when number of times to execute the statements is known to programmer.	@Raza
What is the distinction between for\while & do while loop?	• In a do while the statement or body of the loop will be executed at least once before the condition is validated.	@Mithun
symbol of %	◦ **%** is known as **modulus operator** . it gives us the remainder when we divide 2 numbers. • for example : 10%2 = Remainder is 0. 5%2 = Remainder is 1.	@jeetendra **@Amol**

	• More examples: ▪ **10%1=0** ▪ **1%10 (any number)=1** ▪ **10/1=10** ▪ **1/10 (any number)=0**	
while loop	• Break can be used in the while loop. • Exit condition is must in while loop.	@jeetendra
When to use for Loop	• When no of iterations are fixed.	@Simran
When to use do while loop	• Do while loop - Irrespective of condition it executes the body of loop once.	@Simran @Sangeetha
When to use while loop	• When number of iterations are not fixed.	@Simran

Array Concepts:

Array	• Collection of similar data types is called an array. **Syntax:** • int arr[]=new int[4];---**declaration of array** • 4=size /length of array • []->we can write before arr/after arr • **intialization of array** • arr[0]=10; • arr[1]=20; • arr[2]=30; • arr[3]=40; • We can declare arrays in byte,int,short,char,string,double data types • The elements in the array allocated by *new* will automatically be initialized to **zero** (for numeric types), **false** (for boolean), or **null** (for reference types).	@anupama @Anuradha

	Note: Arrays by default is static to overcome this , arrayList is used • Memory is used when a declaration is done, for the no of indexes/segments irrespective of whether we set a value to the array[index] or not • Index starts from 0 to (length-1)	
	Lowest Index=0 Highest Index= length-1 Length of Array= highest index+1 **Note:** Array index always starts with 0 • unlike Python we do not have -ve indexes for array  	@Manas @Anuradha
When to use arrays	If we want to store same data type for multiple values use an array Ex: int i=10; i=20; i=30; i=40 in this example **i** can store only **40** value ,so if we declare **array** then will store **multiple** values for same data type. int **i**[]=new int[4]; a[0]=10; a[1]=20; a[2]=30; a[3]=40;	@anupama
AIOB(Array Index Out Of Bound Exception) • Whenever you used an –ve value or,	**Ex 1:** int i[]=new int[2]; i[0]=1; i[1]=2; i[2]=3;//AIOB) **Ex 2:**	@anupama @Anuradha @Dhrumil

the value greater than or equal to the size of the array, then the **ArrayIndexOutOfBoundsException** is thrown. ***Please Note*** : • AIOB exception is thrown either at the assignment for the index or when we try to access the value at the arr[index] when index is in -ve or index is >=arr.length • AIOB is thrown at runtime and is not a compiler exception	int i[]=new int[2]; i[0]=1; i[1]=2; System.out.println(i[2]);//AIOB **Ex 3:** int i[]=new int[2]; i[0]=1; i[1]=2; System.out.println(i[-1]);//AIOB Same like string,float,char dat types...	
Array Types	Static array: Array size is fixed **Use cases:** 1. Handling menu items 2. Drop-down with fixed values 3. Handling calendar month, day **Limitations:** 1. Size is fixed. To overcome this , dynamic array is used 2. Stores only similar data type. To overcome this, use object array	@Saranyaa

	Dynamic array : Array size will vary	
How to fetch/print all the values from an Array	**Using for loop** int[] a= {10,20,30,40}; for (int i=0;i<a.length;i++){ System.out.println(a[i]); } **for-each loop** int[] a= {10,20,30,40}; for(int e: a) { system.out.println(e); }	@Manas @anupama
for each loop	for-each loop used to traverse or to iterate through elements of Array or collection one by one. **Drawbacks:** 1. No index concept Ex: Print numbers in reverse order cannot be achieved using **For each** loop ex: 10 to 1	@payal @anupama @Anuradha
Object array	Object array is also **static** in nature,but you can store different data types of data Syntax: **Object a[]=new Object[5]**; Ex: employee data, student data	@anupama
Code to print characters/ASCII values	Print all the characters a-z: for(char i = 'a' ; i<= 'z'; i++){ System.out.println (i) ; } Output: a-z ========== ASCII values of character: for (char b = 'a' ; b<='z' ; b++) {	@Babita @Dhrumil

	System.out.println ((int) b) ; } Output: 97-122	

Static Arrays : @Naveen AutomationLabs

Arrays Concept : Array are used when we want to store multiple values of same datatype in a variable	whereas dynamic arrays are flexible in size. **Dynamic Arrays are nothing but Collections** (List , Set , Map)	Syntax for Array : int a[]=new int[5]; Same syntax for double, char, String arrays.	@Naveen AutomationLabs
If we try to fetch value which is beyond array size, then compiler will give exception as "**ArrayIndexOutofBound**" exception at run time.	To print values of array, we use for loop for(int i=0;i<a.length;i++) { Syso(a[i]); }	For each (Enhanced for loop) : for(int e : a) { Syso(e); }	@Naveen AutomationLabs
int b[]=new int[100]; b[0]=10; b[1]=20; /**This is very bad coding** as we are wasting lot of memory	In order to store different datatypes in one single array, we use **Object Array**: Object arr[]=new Object[5]; arr[0]="Testing"; arr[1]=100; arr[2]=false;	for(int i=0;i<arr.length; i++) { Syso(arr[i]); }	@Naveen AutomationLabs

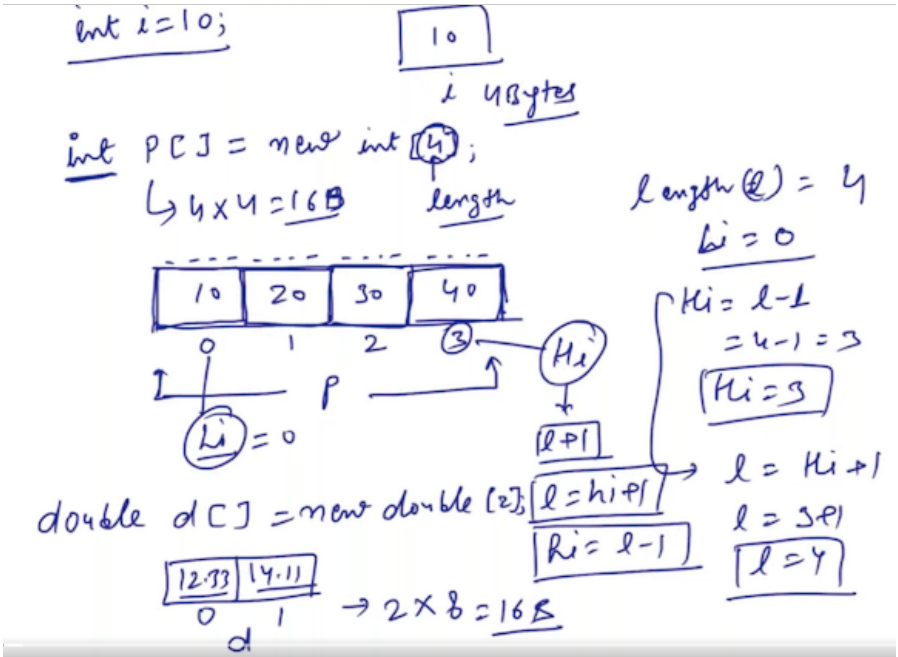
JavaSession -05

Topic: StaticArray.

Array	Types: 1.Static 2. Dynamic	@Naveen AutomationLabs
Static Array:	• Size of array is fixed. This type of array is used when the size is defined • Ex. Number of days of Months • int p[] = new int[4]; // size of array p is 4 • Every segment of array has index, starting from 0 Li (lowest index) • Length(l) = 4 • Li= 0 • Highest index(Hi)= l-1 • Hi =4-1 • Hi= 3 • length of the array ->p.length	@Naveen AutomationLabs
Issues with Static array:	1. This type of array leads to memory wastage, if all the allotted space is not used 2. Also scalability is the issue, there will be downtime to increase the size if the requirement goes up 3. It cannot store different data type values. To overcome these limitation we have to use Dynamic arrays	@Naveen AutomationLabs
Issue of Static array with example	Ex: int price[]=new int[200]; price[0]=200; price[1]=10; Since this is an integer array: integer type - uses 4 bytes ; in this case : 4x200=800 bytes only two elements are occupied : 2x4=8 bytes Total of 792 bytes are unused (wasted) ---------- int pages[]=new int [5]; if there are more pages to be added (dynamic website), then there is downtime to increase the size of the array to accommodate for the new requirement.	@Naveen AutomationLabs

for loop is used to get thru all the array elements	int arr[]=new int[] {0,100,20}; for(int i=0;i<arr.length;i++){ Sop("[" + i+ "] : " +arr[i]);	@Naveen AutomationLabs
for each loop(enhanced loop) can also be used to do the same thing	syntax of for each loop: for(<array_datatype> <var_name> : <array_name>){ Sop(<var_name>); } Ex: int arr[]=new int[] {0,100,20}; int index=0;//can include a index for(int a : arr){ Sop(index + ": " +a); index++; }	@Naveen AutomationLabs
Time complexity of for loop and enhanced for loop are same	O(n)	@Naveen AutomationLabs
for each loop	enhanced for loop Syntax: for(int e : i) { Sop(e); }	@Naveen AutomationLabs
Object	inbuild class in Java is a **parent class of all the classes** in Java Syntax: Object arrayName[] = new Object[size]; yg6tgyt5f	@Naveen AutomationLabs

Array declaration	**// Integer array:** **int** p[] = **new int**[] { 10, 2, 3, 5 }; // using this syntax we can assign values to an array **int** a[] = **new int**[4]; a[0] = 1; a[1] = 2; a[2] = 3; a[3] = 4; **// **** iterate array through typical for loop** **for** (**int** i = 0; i < p.length; i++) { System.***out***.println("The value at index[" + i + "]" + "=" + p[i]); } Using typical for loop, we are iterating using index and hence we need to use integer while iterating. ***/**** iterate an array using enhanced for loop. Here while using enhanced*** ***for loop we use similar data type variable with respect to the array we are iterating */*** **int** count = 0; **for** (**int** e : p) { System.***out***.println("The value at index[" + count + "]" + "=" + e); } /************* Float array ********************** While assigning values to the float array, there is no need to specify decimal values to the array elements. By default all floating literals are considered as Double literals. If we mention decimal values in float array then we have to append each value with f or F otherwise we get error - Type mismatch: cannot convert from double to float **float** q[] = **new float**[] { 1, 2, 3, 4 }; **float** f[] = **new float**[] { 1.0f, 2.0f, 5.0f, 6.0f }; **float** k[] = **new float**[4]; k[0] = 1.0f; k[1] = 2; k[2] = 3.0f; k[3] = 4.0f; **// The concept of object array comes into picture wherein if we want to save heterogeneous data.** Object obj[]=**new** Object[4]; Object obj1[]=**new** Object[] {"John","Male",5.7,57.2}; **for**(Object e:obj1) {	@Naveen AutomationLabs

	System.***out***.println(e); } 	
What is array? Types of array..	If we want to store similar type of data, then we should not create different variables. We should go with Arrays. We can combine all data together, using a single variable name and allocate common memory. Example: store all student names, store all product names. **There are 2 types of array:** 1- static array 2- dynamic array	@Naveen AutomationLabs
Static Array		@Naveen AutomationLabs
About static array..	The size of the array will be fixed in this case. **How to declare an 'int' type array:** int p[] = new int[4]; * Here, length of the array is 4. * How much memory is allocated? -> int occupies 4 bytes each. so, 4 * 4=16 bytes * There is Li(Lowest Index) and Hi (Highest Index), where Li is always 0 and Hi is Length-1. **How to declare double type array:**	@Naveen AutomationLabs

double d[]= new double[2];

* Since each double value occupies 8 bytes of memory, here the above declared array will occupy 2*8=16 bytes

Problems with static array:
1-either the memory is over-allocated. i.e., initially declared with size 499, and if currently only 100 size is required to store products, it is wastage of 399 memory slots.
2-or there will be scarcity of memory. i.e, if initially declared with size 499, and if currently 600 size is required to store similar data, then we would have to shut our app and then increase the array size and then start the server. If this was taken care in 30 mins, then there will be huge business loss in 30 mins.

Where to use static array:
*where count of data is static. example: number of days in a calendar, number of months, in an application where menu items will always be same etc.

Note: If we try to store a value in index greater than the size of the array, then we see **arrayIndexOutOfBoundException**. Also, if we try to assign a value to index '-1', then also we see **arrayIndexOutOfBoundException**.

Example:
int a[]=new int[4]; //array declared with size 4
a[4]=98; //this will throw arrayIndexOutOfBoundException as the 'Hi' is 3.
a[-1]=98; //this will throw arrayIndexOutOfBoundException as the 'Li' is 0.

- **-ve indexing is not allowed in Java but it is allowed in Python.**

How to print all the values of an array -> by using 'for' loop or 'for each' loop

The array we can define with below types:
-> int
-> char
-> String

@Naveen Automati

	-> Double -> Object // If we are storing different types of data in array, then use Object type of array. Example below: Object employee[]=new employee[3]; employee[0]="Diksha"; // denoting name-string type employee[1]='F'; //denoting gender-char type employee[2]=25; //denoting age-int type Similarly, another example is-> cab info about an uber car.	onLabs

JavaSession -06

Topic : ArrayList_DynamicArrayConcepts

• Size of the ArrayList can be get by: ar.**size();** • ***Note***: Size of Array **can't** be get by ar.**length()** method unlike strings , but Size of ArrayList can be get by ArrayList.size() method. • ar.**size() method always gives the current physical capacity of the ArrayList not the Virtual Capacity.**	@Dhrumil @Amol
• Iterate through ArrayList using Loop: Example: ***add*** to add the data and ***get*** to retrieve the data al.add(12.33); al.add("h"); al.add(true); al.add("Testing"); for(int i=0; i<=al.size()-1; i++) { System.***out***.println(al.get(i)); } ---------- // Advanced For Loop for(Object e: al) { System.***out***.println(e); }	@Dhrumil @Amol

- **Virtual Capacity -** Whenever an instance of ArrayList is created then by default the *Virtual capacity* of ArrayList is **10** and the *physical capacity* is **0.**
- Once the default VP is occupied completely i;e when PC=**10** then VP becomes **0**.
- Once the virtual capacity is occupied by elements then it resizes based on the elements added **(size()/2).**

- ArrayList ar1= **new** ArrayList(5);
- [ ] Here the virtual capacity is 5 you can create your own virtual capacity.
- [ ] If you don't pass any number then the default VC is **10**.

Eg : Suppose there is ArrayList which occupied all the 10 arrays. Then it will be resize by n/2 i.e 5(Here n=10 is physical capacity). So it will add **5** more Virtual Capacity.

Now the Physical Capacity is 10 and Virtual Capacity is 5. Similarly if there will be more elements added > Suppose 5 more then physical capacity will be 15 and Virtual Will be added Physical Capacity/2 that will be 15/2 = 7.5 ~ 7 . So 7 more will be added as Virtual Capacity.

@Abhay
@Amol
@Vibha

- **Array:**

```
int i[] = new int[5];
int i[3]=40;
System.out.println(i[3]);
System.out.println(i[0]);  // default value
```

OUTPUT: **40**
0

- Arrays are static in nature so we can add value at any position within defined range.
- If we don't provide any value at any position in static array then it will always give default value in return in above example we got zero 0 as default of integer.

- **ArrayList:**

```
ArrayList ar= new ArrayList(5);
ar.add(3, 40);
System.out.println(ar.get(3));
```

OUTPUT: Exception in thread "main"
java.lang.IndexOutOfBoundsException
: Index: 1, Size: 0

- When an element is removed the next element is moved/adjust accordingly.

@Amol

- We cannot assign the element in the middle of the ArrayList;It gives array Index out of bounds exception.first value is always stored in 0th location,we cannot jump directly to 3/4th location without allocation in 0/1/2th location.
- We need to come sequentially. ArrayList is a continuous memory allocation.

@vibha @Anuradha

	Array	ArrayList
Resizable	No	Yes
Primitives	Yes	No
Iterating values	for, for each	Iterator , for each
Length	length variable	size method
Performance	Fast	Slow in
Multidimensional	Yes	No
Add Elements	Assignment operator	add method
Allocation of elements	can be random	continous

Note : You can set a value to any index between lowest and highest index after initializing it Array are not sorted .
ex int arr1[] =new int[4] ;
arr1[2] = 5 ;

@Akanksha

All methods performed on Physical capacity of Arraylist.
These methods are from java.util.ArrayList .
1 - add(index,element) - Used to add elements in Arraylist at specific indices.
2 - remove(index) - removes elements from specific index.

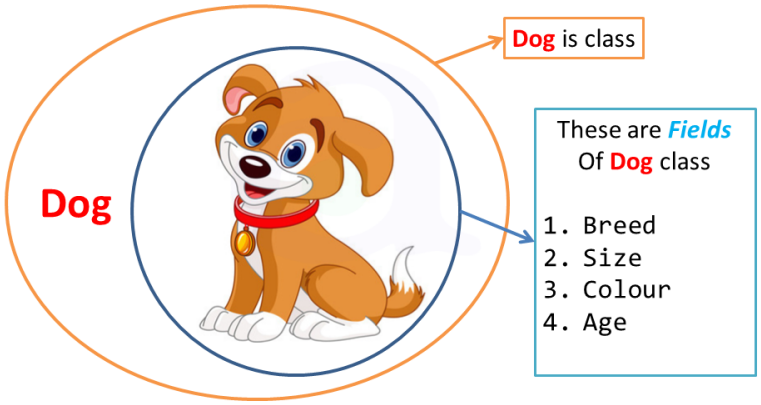
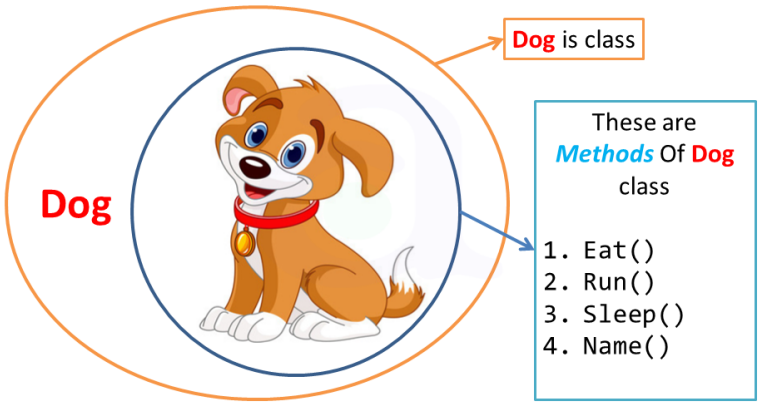
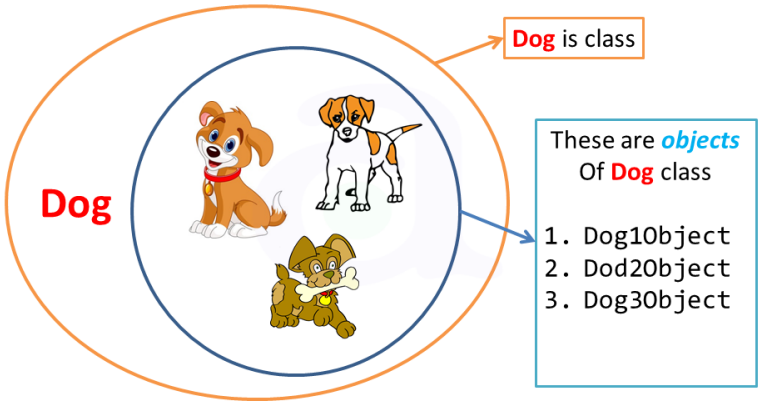
Generics :- 1-Used to store specific type of data in ArrayList. 2- Supports Type-Safety by defining Datatype. ex- If we want to store Strings ArrayList<String> list=new ArrayList<String>(); we can use Object to store all kinds. ArrayList<Objects> list=new ArrayList<Objects>();	@Akanksha

JavaSession -07

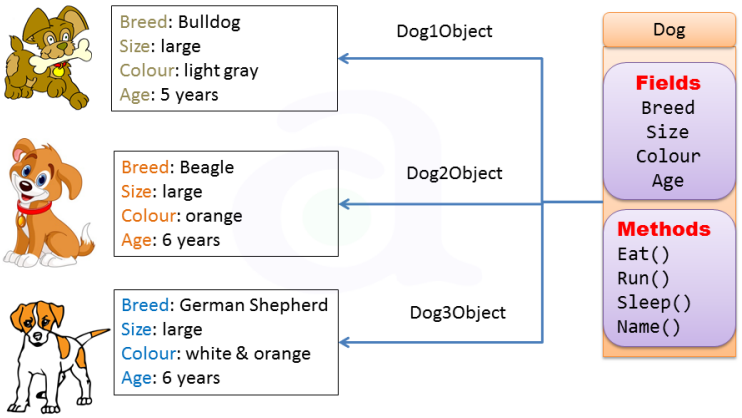
Topic : Class_And_Objects_References_NullReference

Class	*Object*	
• It is a ***blueprint*** or ***template*** or ***category***. • It is used to declare or create object. • You can create multiple objects by using a single class.	• Object is an ***instance of class.*** • Multiple references are allowed for a single object.	@Saranyaa @Komal @vibha @Amol
• ***No memory is allocated*** when a class is declared.	• Memory is created as soon as an object is created (Run time). • The **new** keyword is used to allocate memory ***at runtime.*** • All objects get memory in ***Heap memory*** area.	
• Class is a ***logical entity*** (blue print).	• Object is a ***physical entity.***	@Amol
• Class can only be declared once.	• Multiple objects can be created as per requirement.	
• For example Car is a class	• Object of class car can be Tata, Jaguar,Audi	
• Class c=new Class();	• c is stored in stack, where as the object resides in heap.The memory related to heap is taken care by Garbage collector of JVM.	

We

Class and Object Illustrations		@Amol
		@Amol
		@Amol
		@Amol

What is Object in Java?



- An object is an instance of the class which has state and behavior.
  - **State:** represents the data (value/variables) of an object. (what it has?)
  - **Behavior:** represents the behavior (functionality) of an object. (what it can do?)

@Amol

How to create an Object in Java?

- There are 3 ways to initialize object in Java. Initializing an object means storing data into the object.
  i. By reference variable
  ii. By method
  iii. By constructor

@Amol

No Reference Object

- **Employee e1= new Employee();**
  - **Employee :** Class name
  - **e1 :** Object_Reference_Name/ reference variable
  - **new Employee():** is the real object(RHS)
  - Here **e1** is not an object the real object **new Employee()** is being referred by the reference variable e1.

@Amol

@Amol

Null Reference Object

--

- **new Employee();**
  - We can create an object without a reference variable it's called as ***No Reference Object***.
  - But it's not a good practice to create unnecessary many objects without a reference.
  - How many times this statement is executed that many number of objects are created and corresponding memory is allocated in heap.

--

- Here ObjectReference e3 pointing to **new Employee()** object

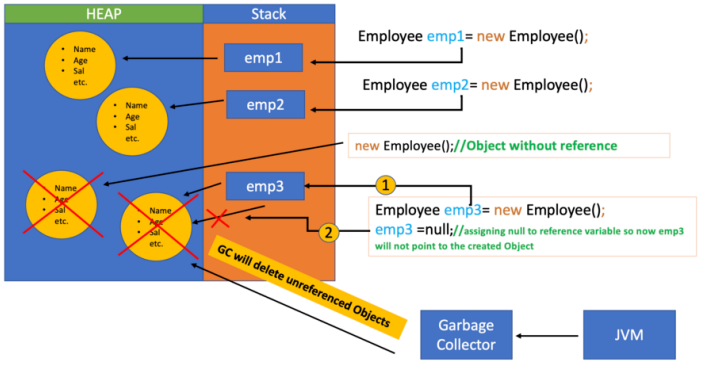
```
Employee e3= new Employee();
e3.name="Peter";
//System.out.println(e3.name);  ------o/p= 'Peter'

e3=null;
```

- Now ObjectReference e3 pointing to **null**

```
System.out.println(e3.name);
//System.out.println(e3.name);  ---> (null.name)
```

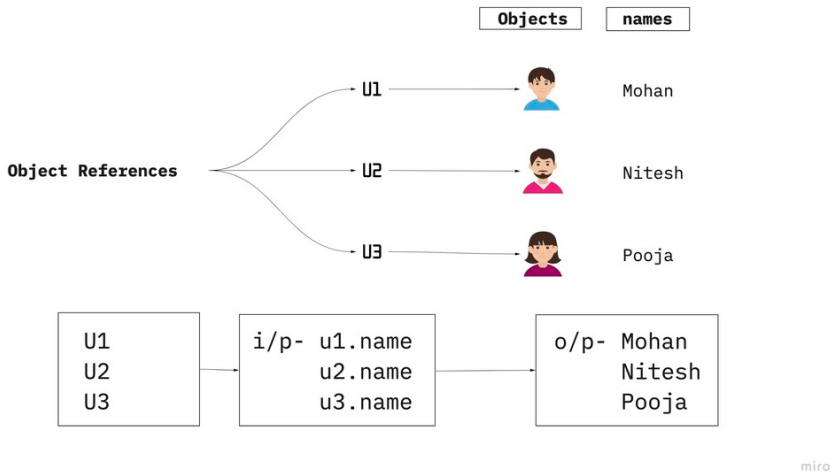
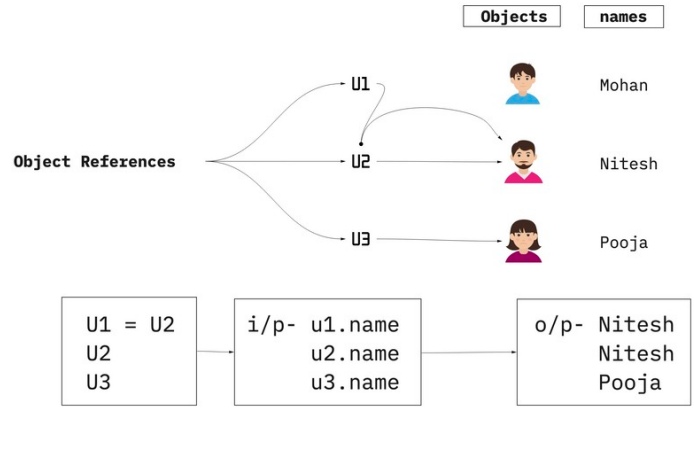
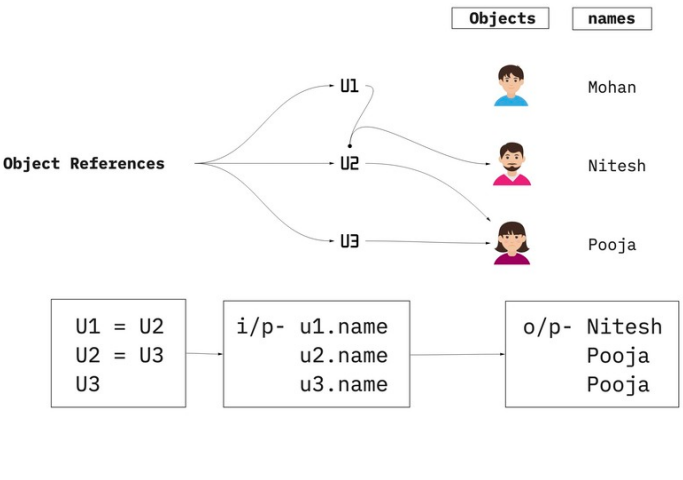
  - [ ] **o/p** NullPointerException: Cannot read field "name" because "e3" is null

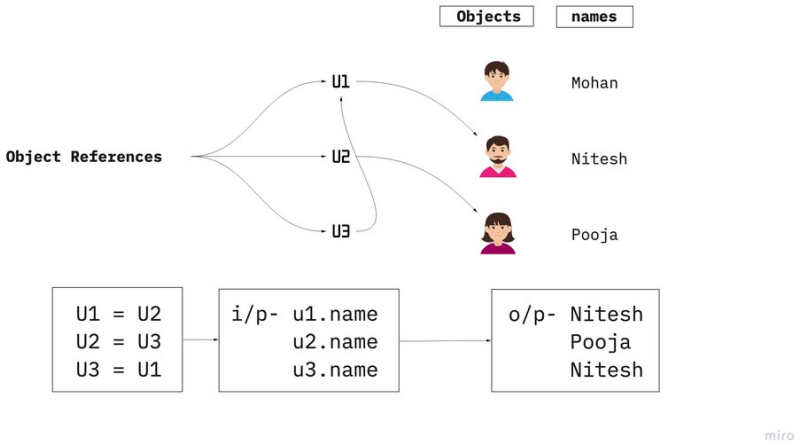
- Now **new** Employee() is being **NullReferenceObject** ready for garbage collection.

Memory management Java	 • Object is always created in Heap memory. • Reference variable is created in Stack memory.	@Lavnya @Saranyaa
Role of Garbage Collector and JVM in memory management	• GC will destroy those objects references which are referring to the Null from heap memory. For ex: **emp3**=Null; ◦ It will break the connection and If we try to print then we will get Null pointer exception • We can also request GC to collect non-referenced objects by using ***System.gc();*** • After GC receives the users request, but waits for instructions from JVM for cleaning the heap memory. ☐ **NOTE**: ***Garbage collector is defined only for heap memory GC can't access stack memory.***	@madhumati @Vibha @Saranyaa @Amol
NullPointerException	• ObjectReference e3 pointing to **new Employee()** object Employee e3= **new** Employee(); e3.name="peter"; e3.age=24; //System.out.println(e3.name); ------o/p= Peter • Now ObjectReference e3 pointing to **null** e3=**null**; System.***out***.println(e3.name); //System.out.println(e3.name); ------o/p (null.name)	@Amol

	☐ NullPointerException: Cannot read field "name" because "e3" is null • Now **new** Employee() is being **NullReferenceObject** ready for garbage collection. • Here In NullReferenceObject reference is still there but pointing to null.	

NOTE	☐ One object can have ***multiple references.***	@Rishabh
Default Values for Different Data Types	• Default value of **String** is **null** • Default value of **Integer** is **0** • Default value of **double** is **0.0** • Default value of **char** is **blank space** • Default value of **boolean** is **false**	@Vishnupriya
Garbage collector eligible for	• NonReferenceObject. • NullReferenceObject i.e reference pointing to null. • Java GC operates only in the heap memory section not in the stack memory section.	@Simran @Sohel
System.gc()	• It might call the jvm or not.Garbage collector will run to reclaim the unused memory space upon instructions from JVM.	@Rishabh

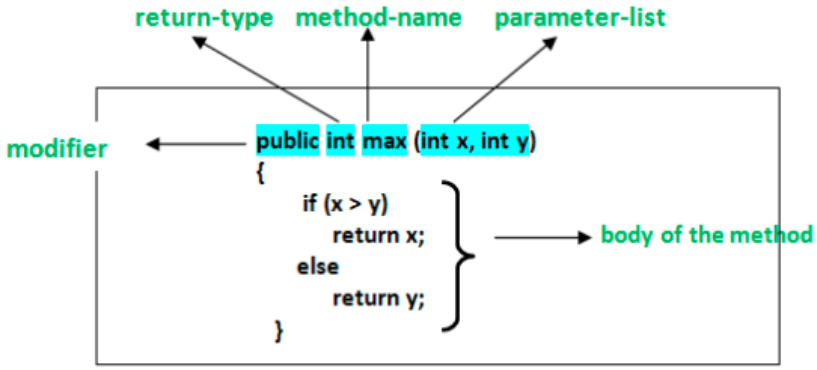
Object References U1,U2,U3 pointing to their respective objects.		@Amol
ObjectReference U1 pointing to U2.		@Amol
Before U2 pointing towards U3 ,U2 was pointing towards itself that's why U1 gives Nitesh as o/p		@Amol

Only think of current situations . Where the given reference is pointing at this moment of time???		@Amol

JavaSession -08

Topic : Functions/Methods_InJava_with_DifferentExamples

What are the functions/methods in java?	• A **method/function** is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation. • It is used to achieve the **re-usability** of code. We write a method once and use it many times. We do not require to write code again and again. • It also provides the **easy modification** and **readability** of code, just by adding or removing a chunk of code. ☐ ***NOTE: The method is executed only when we call or invoke it.***	@Amol
Features of Methods/functions	• A function is also called as ***method***. • A function ***cannot be created inside a function***. - no nested functions. • Functions are ***parallel*** to each other. • Functions are always ***independent of each other***. • To call a function we have to create the object of the class inside main() method. if it is non static. • We cannot create function inside main method. ☐ ***We cannot use return and break together inside any method.***	@Jeetendra @Simran @Sachin @Manas @Jyothsna @ Sangeeta @Smitha @Anuradha @Babita @Amol

NOTE *Method body*	• If using **return**, it should be the last statement of your function. • **Return** keyword is not used in the main method. ☐ ***void and return keyword should not be used together.*** • Function name can start with small letter if its single word else should start with small letter and have capital letter for second word. • For Function naming use camel casing-- ◦ example: Public void launchBrowser() ◦ Eg: public void sum(); public void getSum(); ☐ ***Only Non-static methods and variables are called as a data-members of the class.***  	
What is method signature?	• Every method has a method signature. It is a part of the method declaration. • It includes the **method name** and **parameter list**. • The return type and exceptions are not considered as part of it. Above ex. : ***max(int x, int y).***	@Amol
How to call a non-static (Instance) method inside the	• The method of the class is known as a non-static/instance method. It is a non-static method defined in the class without the static keyword. Before calling or invoking the instance	@Amol

main() method?	method, it is necessary to create an object of its class. • ***Create an object of a class Math*** (where the above max() method is defined) inside the main() method. ◦ Here the max() method should be non-static. *Math m1= new Math();* • ***Call the method using ObjectRefrence variable inside main () method*** *m1.max(2,4);* //here 2,4 are called arguments	
How to call a static method inside the main() method?	• A method that has **static** keyword is known as static method. In other words, a method that belongs to a class rather than an instance of a class is known as a static method. • We can also create a static method by using the keyword static before the method name. • The main advantage of a static method is that we can call it without creating an object. ☐ ***Inside the same class*** where max() method is defined as static Call the method directly as below: ***MethodName(arguments);*** *max(2,4); (in case max() method is static)* ☐ ***Outside the class*** Call the method as : ***ClassName.MethodName(arg);*** *Math.max(2,4);* • The best example of a static method is the main() method.	@Amol
void	• void means it cannot return any value.	@Jeetendra
Default Function/Zero parameterised/ No input no return	**public void** getMarks() { **int** a=10; **int** b=20; **int** c=30; **int** total=a+b+c; System.***out***.println("total marks"+total);	@Simran @Sangeetha

```
        }
```

where

- getMarks()- Function name/ method name
- void- Return type i.e cannot return any value

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

                Math m1= new Math();
                int t=m1.getMarks();
                System.out.println(t+20);
    }
```

- A method which is preceding by void keyword wont return any value.

Types of Functions:-

- ***No input no return***
- ***No input and some return***
- ***Some input and some return***
- ***Some input and no return***

@Amol

No input and some return

```
public int getMarks()
    {
        int a=10;
        int b=20;
        int c=30;
        int total=a+b+c;
        System.out.println("total marks"+total);
        return total;
        }
```

NOTE:

- ☐ ***return statement should be the last statement of any function with return type.***

@Amol

Some input and some return

```
public int add(int a, int b ) {.   // int a and int b are parameters
```

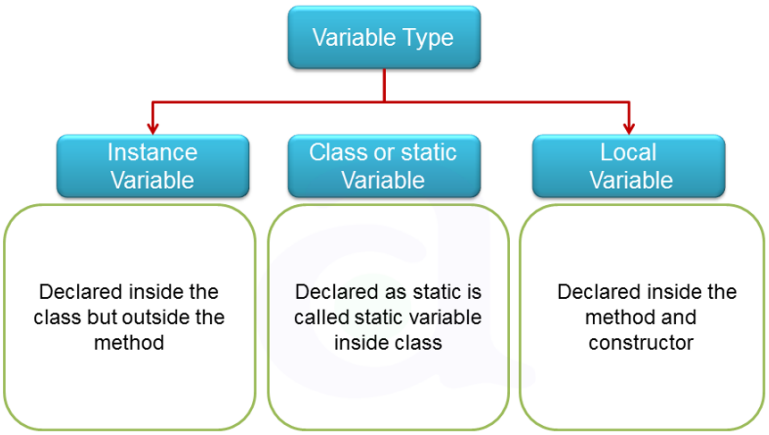
```
            System.out.println("add-method");
            int sum=a+b;
            return sum;

        }
    public static void main(String[] args) {

                Math m1= new Math();
                int s1=m1.add(23,35);   // 23 and 35 are arguments
                System.out.println(s1);
        }
```

Difference between Parameters and Arguments	☐ **Parameters**- At the time of creating function what we give. ☐ **Arguments**- Actual values that we are passing while calling required method in main method.	@Simran

return Keyword:	• ***we cannot write 2 or more return keywords together in a function, return should be the last statement of the function.*** • Return can be any of these datatypes (string, int, boolean, arraylist,array,double,float..) • if we add a return statement to a function , but use void while writing a function , you will see a error in the code • return statement datatype should match the declaration of the type on the function • eg - public void sum(){ ◦ int a = 7 ; return a ; } - will throw a error while coding as we have declared return type for function as void, but returning a int. • holding variable is good practice to store values returned by the function in a class object and manipulate it further • A function can not be created inside the main method. • Call a function have to create the object of the class. **NOTE**: ☐ ***return statement should be the last statement of any function with return type.***	@Jeetendra @Anuradha @Babita @Meenakshi @Amol

	☐ ***There should be only one return statement per function.*** ☐ ***Key point: Any statement after return statement will result in compile-time error stating "Unreachable code".***	
Can we change return type of main() method in java?	• The *public static void main()* method is the entry point of the Java program. • Whenever you execute a program in Java, the JVM searches for the main method and starts executing from it. • You can write the main method in your program with return type other than *void*, the program gets compiled without compilation errors. But, at the time of execution JVM does not consider this new method (with return type other than *void*) as the entry point of the program. • It searches for the main method which is public, static, with return type void, and a String array as an argument. ◦ `public static int main(String[] args) { int a=34; return a; }` • If such a method is not found, a run time error is generated. ☐ Error: Main method must return a value of type void in class demo.Demoyt, please define the main method as: public static void main(String[] args)	@Amol
Return an array in function	We can return an array from function and also pass an array as parameters to a function. Refer below example:- `public class returnArray { int []arr= new int[4]; public int[] fillArray() { for(int i=0; i<4; i++) { arr[i]= i+2; } return arr; } public void printArray(int []b) { for(int i=0;i< 4; i++) { System.out.println(b[i]); } }`	@Roopali

	public static void main(String[] args) { returnArray a= new returnArray(); int b[] = a.fillArray(); a.printArray(b); } }	
Why main() method in Java is Void and Static?	• JVM call main() method to execute the program.Once main() method has finished executing, java program also terminates. • If we provide return statement in main(), JVM cant do anything with that return value. So main() method is always void in nature • main() method is static because the object is not required to call a static method. If it were a non-static method, JVMcreates an object first then call main() method that will lead the problem of extra memory allocation.	@Vishnupriya @Saranyaa @Amol

Types of Variables	 ▪ Instance variables : One copy per object. Every object has its own instance variable. ▪ E.g. x, y, r (centre and radius in the circle) ▪ Static variables : One copy per class. ▪ E.g. numCircles (total number of circle objects created)	@AMOL

Local vs Instance vs Static/Class variables

	Local variable	Instance variable	Static/Class variable
Declaration	Inside methods/ constructors/ blocks	Inside the class but outside the method/constructor/block	Declared as static inside the class but outside the method/ block/ constructor
Scope	Only inside the methods/ constructors/blocks	Inside all methods/ blocks/ constructors within a class.(not inside a static method directly ,need to create object.)	Everywhere inside the class including static methods
When variable get allocated in memory?	When method/constructor/ block get executed it allocates memory and get destroyed after exiting from block/method.	Instance variables are created when an object/instance of the class is created and destroyed when the object is destroyed.	Static variables are created at the start of program execution , when *.class* file executed and destroyed automatically when execution ends.
Stored in ..?	Stack memory	Heap memory	Non-Heap /Static memory
Default value	Don't have default values. Initialization of the local variable is mandatory before using it in the defined scope.	Int 0 Boolean false String null	Int 0 Boolean false String null
Access specifier /Modifiers	We can't use access specifiers with local variables.	Can be used.	Can be used
How to access?	Directly inside the block/ method/ constructor.	For static method call it directly. For simple method create object to access it.	Inside the class directly. Outside the class by using ClassName.b By using object reference name.

@AMOL

JavaSession -09

Topic: MethodOverloading_MainMethodOverloading_Static_MetaSpace.

Okie

Method Overloading	• If a class has multiple methods having ***same name*** but ***different in parameters,*** it is known as Method Overloading. • Also called as ***CompileTimePolymorphism*** or ***StaticBinding(Early Binding)*** . • Sequence(order) of parameters can be different in Method Overloading. eg testMethod(int a, string b) is different from testMethod(int b, string a) • It gives better readability. • Faster code execution. • Return type is not significant in method overloading. • Compiler uses method signature to check whether the method is overloaded or duplicated. Duplicate methods will have same method signatures i.e **same name, same number of arguments and same types of arguments, and in the same order** • Overloaded methods will also have same name but differ in number of arguments or type of arguments. • Compiler checks only method signature for overloading of methods not the visibility(public /private) of methods. ***NOTE***: ☐ ***Java doesn't support two duplicate methods with the same name and same parameter in a single class.*** ☐ ***Method overloading always happens to be within the same class.***	@Archana @Urvi @Roopali @vibha @Anuradha @Amol
Method can be overloaded by:	1. By ***changing the no. of parameters*** in method signature. 2. By ***changing the data-type*** of parameters. 3. By ***changing the order of parameters***: If the two methods have the same number of parameters and the same type of parameters but if the order of parameters is different, overloading works. • If all methods follow the above mandatory rules, then they may or may not:(not necessarily to be same) ◦ Have different ***return types.*** ◦ Have different ***access modifiers***. ◦ Throw different ***checked or unchecked exceptions***. **public class** Demo {	@Amol

	public void test() { System.out.println("test method"); } public void test(int a) { System.out.println("test method"+a); } //changing the no. of parameters public void test(int a, int b) { System.out.println("test method"+a+b); } //changing the data-type of parameters public void test(int a, String b) { System.out.println("test method"+a+b); } //changing the order of parameters: public void test(String a, int b) { System.out.println("test method"+a+b); } } 	@Amol
You can't overload method by:	☐ ***Changing the return type of method only:*** Method overloading does not work ***if the method name and parameters are same but the return type is different.*** public ~~int~~ test(int a, int b) { //Not allowed System.out.println("test method"+a+b); } public ~~void~~ test(int a, int b) { //Not allowed System.out.println("test method"+a+b); }	@Amol
Can we overload the main() method?	• The answer is, YES, we can overload the main() method. But remember that the ***JVM always calls the original main() method. It does not call the overloaded main() method.*** • public class Demo { public class MainMethodOverload {	@Amol

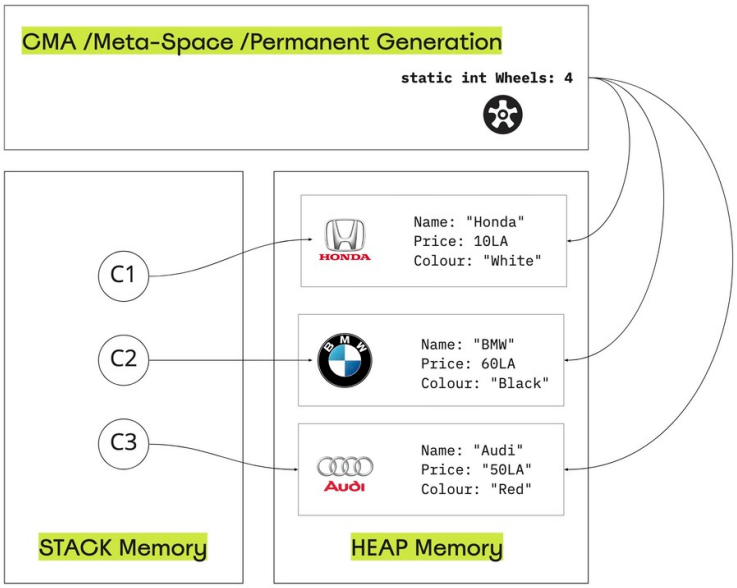
```
    // Overloaded main() method 1
    //invoked when an int value is passed
    public static void main(int a)
    {
    System.out.println(a);
    }

    // Overloaded main() method 2
    //invoked when a two integers are passed
    public static void main(int a, int b)
    {
    System.out.println(a+b);
    }

    //Original main() method
    public static void main(String[] args)
    {
    System.out.println("Original main() method invoked");
    main(12,23);     // Overloaded main() method 2
    main(10);     // Overloaded main() method 1
    }
  }
}
```

- The main() method is the starting point of any Java program. The JVM starts the execution of any Java program form the main() method. Without the main() method, JVM will not execute the program.

Holding Variable	• its a variable used in the class to receive the return value from a function called , useful in manipulating the return values	@Anuradha
Static keyword	• This means we will create only one instance of that static member that is shared across all instances/objects of the class. • The static keyword in Java is used for ***memory management*** mainly. It makes your program memory efficient (i.e., it saves memory). • The static keyword ***belongs to the class rather than an instance/object*** of the class because object can't hold any static data. • The static methods can be called directly without creating an	@Srinivas @Saranyaa @Anuradha @Amol

	Object. • If the class has property holding common value, then it will be declared as Static. • The static variable can be used to refer to the common property of all objects (which is not unique for each object), ◦ for example, the company name of employees, college name of students, etc. • In order for the property not to be overwritten by mistake we should declare it as final - example - wheels in a car class • Local variables cannot be declared as static , but can be declared as final.	
Static Keyword	• Static keyword applies to 1) Variables (Class variables) 2) Methods (Class methods) 3) Class 4) Blocks • Static keyword is not applicable for local variables	@Roopali @Anitha @Saranyaa @Meenakshi @Amol
Static memory allocation	• Always static memory allocation will be in Meta space. • The static variable gets memory only once in the class area at the time of class loading. ☐ ***static*** - means Only one copy ,now all objects will share this common one copy in meta-space.	@Supriya @Amol

Java Memory	is divided into following Sections. 1. Heap 2. Stack 3. CMA/Meta-Space/ Permanent Generation • ***The Stack section of memory contains regular methods, local variables, and reference variables.*** • ***The Heap section of memory contains Objects.***	@Srinivas @Vibha

	 • ***PermGen Memory:*** This is a special space in java heap which is separated from the main memory where ***all the static content*** is stored in this section. Apart from that, this memory also stores the application metadata required by the JVM. • The biggest disadvantage of PermGen is that it contains a limited size which leads to an Out of Memory Error. • Also the PermGen space cannot be made to auto increase. So, it is difficult to tune it. And also, the garbage collector is not efficient enough to clean the memory. • Due to the above problems, **PermGen has been completely removed in Java 8.** In the place of PermGen, a new feature called **Meta Space(Common Memory Allocation)**has been introduced. MetaSpace grows automatically by default. Here, the garbage collection is automatically triggered when the class metadata usage reaches its maximum metaspace size. • If CMA is fully *dynamic memory allocation* happens i,e if beyond 200mb, space will be shared from systems RAM.	@Amol
final keyword for a variable.	• If we initialize a variable with the final keyword, then we cannot modify its value. ◦ **ex:- static final int wheels = 4;** ◦ Now wheels is constant ,you can't change the value of wheels to any other value.	@Srinivas @Amol
How to call static variables and	☐ ***Inside the same class*** where max() method is defined as static Call the method directly as below:	@Saranyaa @Roopali

methods	***MethodName(arguments);*** *max(2,4); (in case max() method is static)* ☐ ***Outside the class*** Call the method as : ***ClassName.MethodName(arg);*** *Math.max(2,4);*	@Babita
Can we declare local variables as static?	• In Java, a static variable is a ***class variable (for whole class).*** • So if we have static local variable (a variable with scope limited to function), it violates the purpose of static. ***Hence compiler does not allow static local variable.*** `class Test {` `  public static void main(String args[]) {` `   System.out.println(fun());` `  }` `  static int fun()` `  {` `   static int x= 10;  //Error: Static local variables are not allowed` `  return x--;` `  }` `}`	@Amol
Can we declare local variables as final ?	***YES*** `public static void main(String[] args) {` `System.out.println(fun());` `}` `static int fun()` `{` `final int x= 10;` `//allowed But you can't change the value of x once you declare it` `return x;` `}`	@Amol
Ex: WAP program for method that	package MyPrograms;	@anupama

can take input parameter as arrays. Write a function which takes an array of strings and a search string, in this function it should print only those strings of array which have the given search string in it.	public class ArrayString { public void sample(String[] ip, String search) { for (int i = 0; i < ip.length; i++) { if (ip[i].contains(search)) { System.out.println(ip[i]); } } } public static void main(String[] args) { ArrayString a1 = new ArrayString(); String s1[] = { "javay", "python", "Ruby", "Testinngy" }; a1.sample(s1, "y"); } } o/p: javay python Ruby Testinngy	

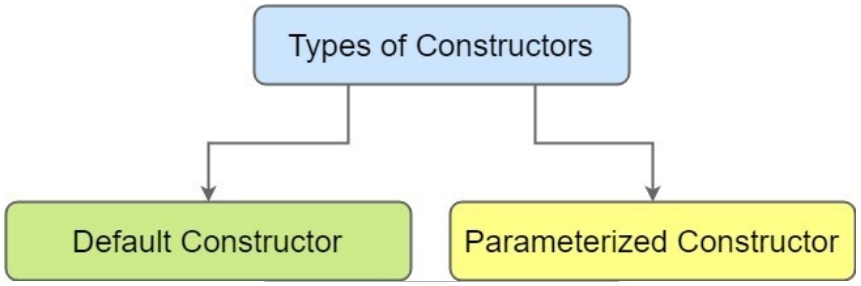
Switch Case	• In Switch case, we can use only string and Integer values, not boolean value. • return and break cannot be written together because return is the last statement in a function.also once return is encountered it will just go back	@Babita @Anuraadha

JavaSession -10

Topic: Constructors_ThisKeyword

What is	• A constructor is a block of codes which is used to initialize the	@Amol

constructor?	object. • It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory. • Every time an object is created using the new() keyword, at least one constructor is called. • It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default. • It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if your class doesn't have any.	
Rules for creating Java constructor	• Constructor name must be the same as its class name. • A Constructor must have no return type. • We can overload the constructor. They are differentiated by the compiler by the number of parameters in the list and their types. ***NOTE*** ☐ We can use access modifiers while declaring a constructor. It controls the object creation. In other words, we can have private, protected, public or default constructor in Java. ☐ ***Rule***: *If there is no constructor in a class, compiler automatically creates a default constructor.*	@Amol

Constructor vs Method	<table><tr><th>Java Constructor</th><th>Java Method</th></tr><tr><td>A constructor is used to initialize the state of an object.</td><td>A method is used to expose the behavior of an object.</td></tr><tr><td>A constructor must not have a return type.</td><td>A method must have a return type.</td></tr><tr><td>The constructor is invoked implicitly.</td><td>The method is invoked explicitly.</td></tr><tr><td>The Java compiler provides a default constructor if you don't have any constructor in a class.</td><td>The method is not provided by the compiler in any case.</td></tr><tr><td>The constructor name must be same as the class name.</td><td>The method name may or may not be same as the class name.</td></tr></table> ☐ The constructor will be called the moment you create an object of the class. ☐ The method will be called when you create the object of the class and use objectReference to call the method. ☐ We never write a business logic inside the constructor ,it's only used to initialise the object that's it.	@Amol
Constructor Types	 1. ***Default Constructor:*** ◦ A constructor is called "Default Constructor" when it doesn't have any parameter. ◦ If there is no constructor in a class, compiler automatically creates a default constructor. 2. ***Parameterised Constructor:*** ◦ A constructor which has a specific number of parameters is called a parameterized constructor. ◦ The parameterized constructor is used to provide different values to distinct objects. However, you can provide the same values also.	@Amol

Constructor program	**public class** Employee { String name; **int** age; String city; //Class variables **double** salary; **boolean** isPerm; **public** Employee(String name,**int** age) { //Local variables System.***out***.println("Employee constructor"); // this keyword used to access current class variables **this**.name=name; **this**.age=age; } **public static void** main(String[] args) { // you have to pass only name and age here ,here its compulsory while creating object Employee e1= **new** Employee("Amol",24); System.***out***.println(e1.name); System.***out***.println(e1.age); } } ***NOTE***: ☐ If you don't assign values to the class variables using 'this' keyword it simply will print default values of class variables e.age=0; e1.name=null.	@Amol

Method Chaining	**public class** Demo { **public void** t1() { System.***out***.println("Method t1"); t2(); } **public void** t2() {	@Amol

```
        System.out.println("Method t2");
        t3();
    }
    public void t3() {
        System.out.println("Method t3");
        t1();
    }
    public static void main(String[] args) {
        Demo d=new Demo();
        d.t1();
        d.t2();
        d.t3();
    }
}
```

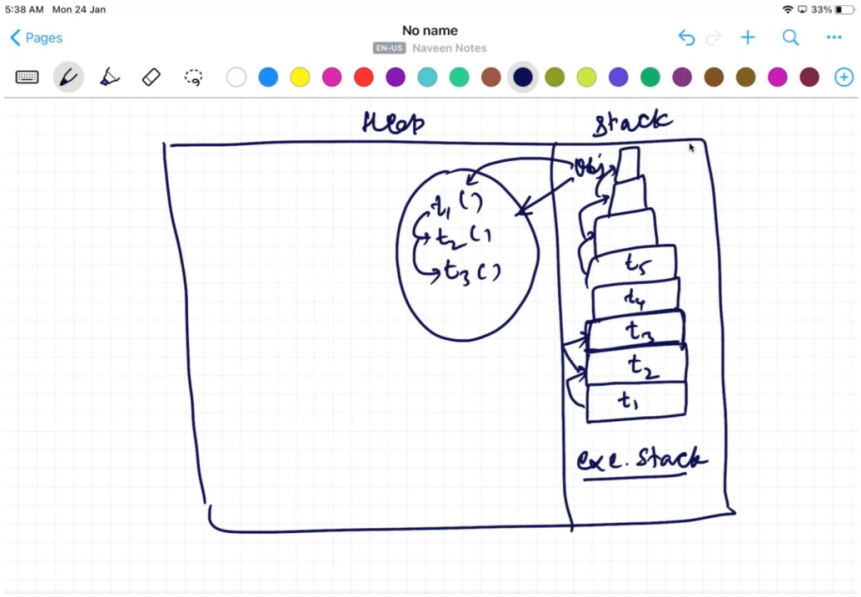
- Method chaining can be defined as if we have an object and we call methods on that object one after another is called method chaining.
- For example,
  obj.method1().method2().method3();
  d.t1().t2().t3();
- In the above statement, we have an object (d) and calling method t1() then method t2(), after that the method t3(). So, calling or invoking methods one after another is known as method chaining.
- The execution of method chaining is happened inside the stack memory.

Stack Overflow Error

Stack releases memory in LIFO - the last function / method called is released, then the previous and the previous once it done processing ,

---

@Babita @Anuradha



- If t1() method is called to t2(),
- t2() method is called to t3() and
- t3() is called t1() again, The stack memory will be overflow and start providing the StackOverflowError.
- Its *not an exception it's an error.*
- This concept is called ***LIFO (Last In First Out)***, Queue works based on FIFO (First In First Out)
- When there is no method chaining eg - t1 () method calls t2() and t2() calls t3(), after t3 processing is done , the t3() is released, followed by release of t2 () and then t1() stack memory in LIFO format.

Advantage of Constructors

- Constructor needs to follow the rules of overloading
- Duplicate constructors are not allowed
- No business logic in constructors
- Constructor should have the same name

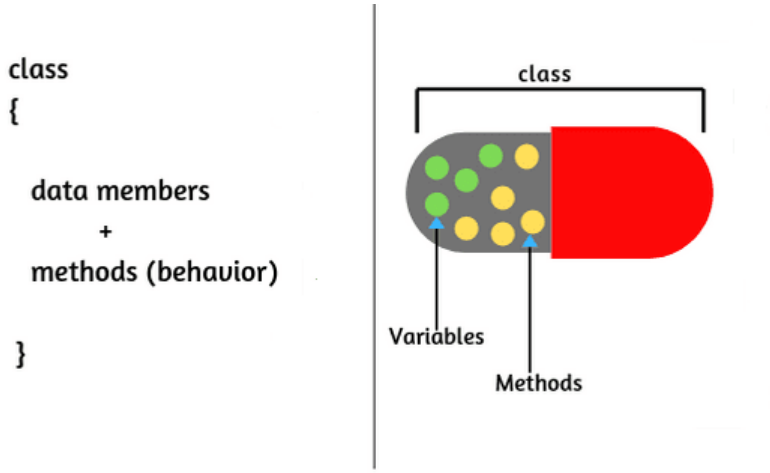
- Constructor restricts creating the unnecessary objects in heap memory .
- Constructor is used to create physical entity in the form of object inside the memory which will help me to initialize the class variables.
- different constructors can be called based on the arguments passed while creating instances of the class
- Constructor is called when the object is created.
- Constructor never returns any value.

@Loknath @Anuradha

as that of the class		

JavaSession -11

Topic: BuilderPattern_MethodChanining_Encapsulation_PrivateMethods

Encapsulation	• Hiding data members of the class is called encapsulation. • We have hidden the data member variables inside a class and have also specified the access modifiers so that they are not accessible to the other classes.Thus encapsulation is also a kind of "***data hiding***". • Data and methods are enclosed in a single unit in encapsulation. • Hiding implementation of the logic / functionalities is called encapsulation. • Encapsulation acts as a protective shield around the data and prevents the data from unauthorized access by the outside world. In other words, it protects the sensitive data of our application.  Ex: ATM Machine ,Laptop(internal parts),capsule For access the private data we can use **setter** and **getter** method.	@Anuradha @Sangeetha @Amol
How to implement Encapsulation?	In Java, there are two steps to implement encapsulation. Following are the steps: 1. *Use the access modifier 'private' to declare the class member*	@Amol

	variables. 2. *To access these private member variables and change their values, we have to provide the public getter and setter methods respectively.* 3. Now we can read the values of the private variables and set new values for these variables using getter and setter methods.	
Why to use Encapsulation?	• It provides you ***more control over the data.*** • It is a way ***to achieve data hiding*** in Java because other class will not be able to access the data through the private data members. • By providing only a setter or getter method, ***you can make the class read-only or write-only.*** • The encapsulated class is ***easy to test***. So, it is better for unit testing. private instance variable Protection | Setter(Mutator) Method | Getter(Accessor) Method No Access | NO Setter | NO Getter method Read-Only Access | NO Setter | Implement Getter method Write-Only Access | Implement Setter method | NO Getter method Read/Write Access | Implement Setter method | Implement Getter method	@Amol
Private keyword	1. If you declare variable/method/class as a private then one can't access properties at the outside the class. 2. The scope of private is within the class only. 3. In order to access those properties indirectly in another class then we are using setters & getters methods. Ex: class emp { String ename; int age; private double sal; } psvm(string[] args)	@Gagan

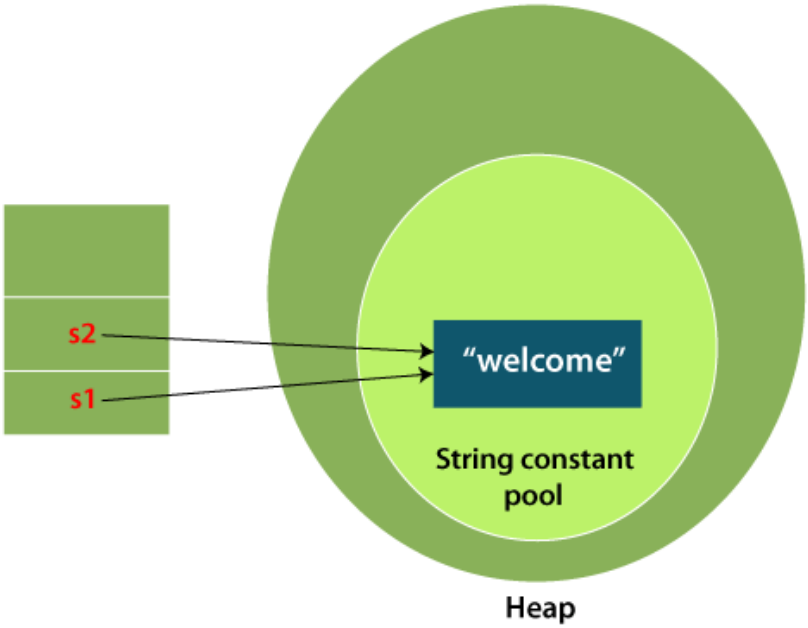
```
{
emp e1=new emp();
e1.ename="peter";
e1.age=25;
e1.sal=30;//Scope :within the class
}
}
class test
{
psvm(String[] args)
{
emp e2= new emp();
e2.name="sam";
e2.age=30;
e2.sal=40.4;//compile time error because this is private variable
the scope is within class only.To overcome this we are using
getters & setters method
}
}
Solution-1
Using setters & Getters method
public Class Employee
{
public String ename;
public int age;
private double esal;
public void setsal(double sal)
{
this.sal=sal;
}
public void getsal()
{
}
}
Class Test
{
psvm(String[] args)
{
Employee e=new Employee();
e.setsal(20.45);
double salary=e.getsal();
```

	syso(salary+300); } } **Note:** 1. private keyword can be applied to variables, methods and inner class in Java. 2. if class variables are public no need to create setters & getters because those are already public in nature and available in outside of the class	
Builder Pattern	"Return this;" Statement is used in all the functions of the class which returns the current class object The return type of the function which has "Return this;" statement should be classname Eg: `public BuilderPattern EnterLogin(String login)` `{` `System.out.println("Entered the login");` `return this;` `}` `public BuilderPattern SearchProduct()` `{` `System.out.println("product searched");` `return this;` `}` `public BuilderPattern Addtocart()` `{` `System.out.println("product added to cart");` `return this;` `}` **Advantage:** **Use of Builder pattern is that Method chain can be created as function returns the object of the class** **E.g:** `public static void main(String[] args) {` `BuilderPattern BP1=new BuilderPattern();` `BP1.EnterLogin("2022class").SearchAroduct().addtocart();` `}`	@Jyothsna

JavaSession -12

Topic: StringManipulationMethods

What is a String	• In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java ◦ **char**[] ch={'j','a','v','a','t','p','o','i','n','t'}; String S=**new** String(ch); //is same as: ◦ String S="javatpoint";	@rajaSekhar @Amol
What is String in Java?	• String is a sequence of characters. But in Java, ***strings are object that represents a sequence of characters.*** • The **java.lang.String** class is used to create a string object. • The Java String is immutable which means it cannot be changed. • Whenever we change any string, a new instance is created.	@Amol
How to create a string object? *There are two ways to create String object:* 1. *By string literal* 2. *By new keyword*	☐ ***1) String Literal*** • Java String literal is created by using double quotes. For Example: String s="Java"; • Each time you create a string literal, the JVM checks the "string constant pool" first. • If the string already exists in the pool, a reference to the pooled instance is returned. • If the string doesn't exist in the pool, a new string instance is created and placed in the pool. For example: a. String s1="Welcome"; b. String s2="Welcome";//It doesn't create a new instance	@Amol



- In the above example, only one object will be created. Firstly, JVM will not find any string object with the value "Welcome" in string constant pool that is why it will create a new object.
- After that it will find the string with the value "Welcome" in the pool, it will not create a new object but will return the reference to the same instance.
- In the above example, only one object will be created. Firstly, JVM will not find any string object with the value "Welcome" in string constant pool that is why it will create a new object.
- After that it will find the string with the value "Welcome" in the pool, it will not create a new object but will return the reference to the same instance.

☐ ***Why Java uses the concept of String literal?***

- To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

☐ ***2) By new keyword*** @Amol

- String s=new String("Welcome");
  //creates two objects and one reference variable

- In such case, JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool.

	• Here The variable s will refer to the object in a heap (non-pool).	
String Manipulation	• String s = "this is my java code"; ◦ Calculate the String Length: ***s.length()*** ◦ Internally string work in index manner i.e. 0th Index : "t" 1st Index: "h" . . . 19th Index : "e" ◦ ***space is also considered as Character.*** ◦ s.length() ==> 20	@Dhrumil @Amol
charAt() => Used to find specific char at specific location.	• Get specific character from the Index ◦ s.charAt(0) ==> "t" ◦ s.charAt(19) ==> "e"	@Dhrumil
StringIndexOutOfBoundsException	• s.charAt(20): String index out of range: 20 • s.charAt(-1): String index out of range: -1	@AMOL
indexOf()=> s.indexOf('char') s.indexOf("String") *indexOf() giving you integer value.*	• indexOf() method returns the position of the first occurrence of the specified character or string in a specified string. • There are four overloaded indexOf() method in Java. The signature of indexOf() methods are given below: **No. / Method / Description** 1 — int indexOf(int ch) — It returns the index position for the given char value 2 — int indexOf(int ch, int fromIndex) — It returns the index position for the given char value and from index 3 — int indexOf(String substring) — It returns the index position for the given substring 4 — int indexOf(String substring, int fromIndex) — It returns the index position for the given substring and from index	@Dhrumil @AMOL

	• s.indexOf("t") ==> 0 ◦ Also, we can check through specific string within String. • *s.indexOf("String")*==> 11	
s.indexOf *s.indexOf('char', from[I])* *s.indexOf("String",from[i])*	• s.indexOf("i") ==> 2 ◦ If there are two "i" then it would give precedence first one. • Syntax: ◦ s.indexOf("i", startIndexwith) ◦ Example: s.indexOf("i", 3) ==> 5 ▪ remove hardcoded problem over here, ▪ **s.indexOf ('i', s.indexOf ('i')+1))** ***NOTE***: ☐ ***If String or Char is not present within the Actual String, s.indexof() will give you "-1"*** s.indexOf("Python") ==> "-1"	@Dhrumil @Amol
Handle validation message in Selenium example:	• String msg="hello admin"; `if(msg.indexOf("admin")>0) {` `System.out.println("correct message");` `}`	@Dhrumil
trim() Method	• The Java String class trim() method eliminates leading and trailing spaces. • The Unicode value of space character is **'\u0020'**. The trim() method in Java string checks this Unicode value before and after the string, if it exists then the method removes the spaces and returns the omitted string. String s1=" Hello Selenium "; System.out.println(s1.trim()); o/p-**Hello Selenium**	@Amol @Dhrumil

	String s2="chrome "; System.*out*.println(s2.trim()); o/p-**chrome** **NOTE:** ☐ The string trim() method doesn't omit middle spaces.	
replace() ***Method*** *replace(arg1, arg2)*	• String s3= "Hello Testing"; • System.*out*.println(s3.replace(" ", "")); ◦ Expected Result: **HelloTesting** • *First Argument representing space and second argument is representing nothing.* • String s4=" Hello Selenium "; System.*out*.println(s4.trim().replace(" ","")); System.*out*.println(s4.replace(" ", "")); ◦ Both gives same output: **HelloSelenium**	@Dhrumil
toUpperCase()	• String str="This is java"; ◦ System.*out*.println(str.toUpperCase()) ◦ => **"THIS IS JAVA"**	@Dhrumil
toLowerCase()	• String str="This is java"; ◦ System.*out*.println(str.toLowerCase()) ◦ => **"this is java"**	@Dhrumil
Compare two String concept *- equals(), returns boolean* *- equasIgnoreCase(), returning boolean*	• String s1= "hello google"; String s2= "hello google"; System.*out*.println(s1.equals(s2));//TRUE String s3="hello Google"; // case sensitive System.*out*.println(s1.equals(s3));//FALSE System.*out*.println(s1.equalsIgnoreCase(s3));//TRUE System.*out*.println(s1==s2); //TRUE	@Dhrumil

	☐ ***Note**: String should be matched exactly char by char, extra spaces also going to be considered.* ◦ *s1 == s3 ==> Do not use this way to comparison of String, instead of that use "equals" method.* ☐ *For primitive data types like Int, we can consider "==" to compare.*	
contains() method => returning boolean value.	• String str="Welcome Testing"; System.***out***.println(str.contains("welcome")); //FALSE---Case sensitive System.***out***.println(str.contains("Welcome")); //TRUE **if**(str.contains("Welcome")) System.***out***.println("This is correct"); **else** System.***out***.println("Not correct");	@Dhrumil @AMOL
Substring() concept ***In case of String:*** • ***startIndex: inclusive*** • ***endIndex: exclusive***	☐ **substring(int startIndex):** ◦ This method returns new String object containing the substring of the given string from specified startIndex (inclusive). The method throws an IndexOutOfBoundException when the startIndex is larger than the length of String or less than zero. ☐ **substring(int startIndex, int endIndex):** ◦ This method returns new String object containing the substring of the given string from specified startIndex to endIndex. ◦ The method throws an IndexOutOfBoundException when the startIndex is less than zero or startIndex is greater than endIndex or endIndex is greater than length of String. ☐ String m= "This is my testing code"; System.***out***.println(m.substring(3)); //is my testing code	@Dhrumil @AMOL

	System.***out***.println(m.substring(5,11)); //is my System.***out***.println(m.substring(m.indexOf("is")+3, 8)); //is	
How to fetch dynamic content from the static string? *Realtime scenario: When Order ID is appended with the static content like* *"Your order id is **12345**"*	• String s5="This is your order id 11234"; System.***out***.println(s5.length()); //27 System.***out***.println(s5.substring (s5.indexOf("id")+3, s5.length())); //11234 • Above can be used with static string and you have to capture dynamic thing.	@Dhrumil
Ex :extract a number from a given input string	**String s="Your order 2312 is generated successfully";** String s1=s.substring(s.indexOf("order")+6, s.indexOf("i")-1); System.out.println(s1);	@anupama
Difference between "==" operator and "equals()" method	☐ The String class ***equals() method*** compares the original content of the string. ***It compares values of string for equality***. String class provides the following two methods: a. **S1.equals(S2)**: compares S1 to S2. b. **S2.equalsIgnoreCase(S2)** compares S1 to S2, ignoring case. ☐ ***The == operator compares references not values.*** • String S1 = **new** String("Hello"); String S2 = "Hello"; ◦ S1==S2 => //FALSE ◦ S1.equals(S2) =>//TRUE • Note: avoid using new keyword while defining string, because it would create two objects - one inside the string pool and	@AMOL @Dhrumil

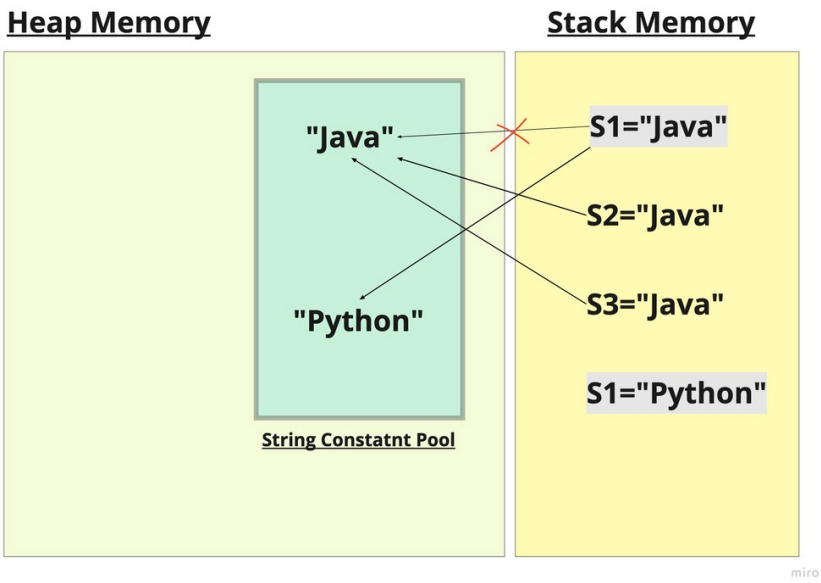
	one inside the Heap. • Always use literals way to define string.(s1)	

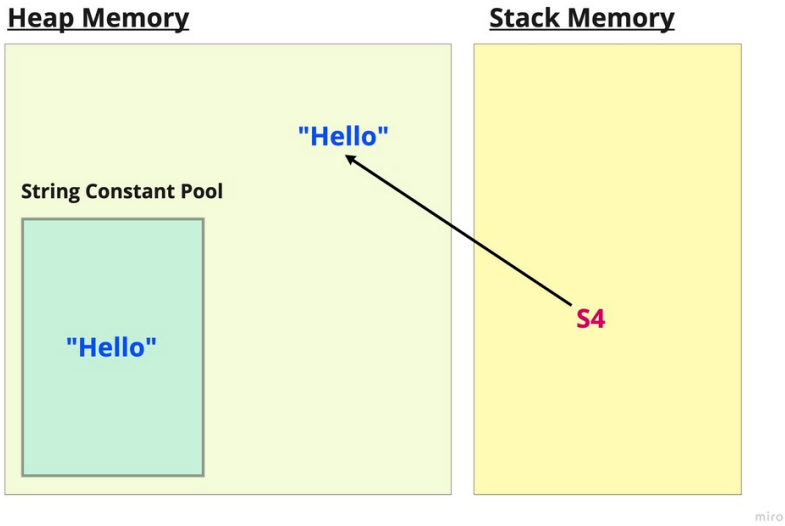
JavaSession -13

Topic:String_Immutability_Split_ConstructorArrayListParameter

- String Constant Pool mechanism
- Constructor- ArrayList and Array passing as Argument to Constructor
- split() method
- call by value vs call/pass by reference understanding
- Data Driven Approach in Selenium using String.

What is String Constant Pool?	• String pool is nothing but a storage area in Java heap where string literals stores. • It is also known as ***String Intern Pool*** or ***String Constant Pool.*** It is just like object allocation. By default, it is empty and privately maintained by the Java String class. • Whenever we create a string the string object occupies some space in the heap memory. Creating a number of strings may increase the cost and memory too which may reduce the performance also. • The JVM performs some steps during the initialization of string literals that increase the performance and decrease the memory load. To decrease the number of String objects created in the JVM the String class keeps a pool of strings. • When we create a string literal, the JVM first check that literal in the String pool. If the literal is already present in the pool, it returns a reference to the pooled instance. If the literal is not present in the pool, a new String object takes place in the String pool.	@AMOL
Creating String in Java ***There are two ways to create a string in Java:*** 1. ***Using String***	• ***Using String literals:*** String S1="Java"; String S2="Java"; String S3="Java"; • How many objects will be created in memory? - 1 object.	@Dhrumil @AMOL

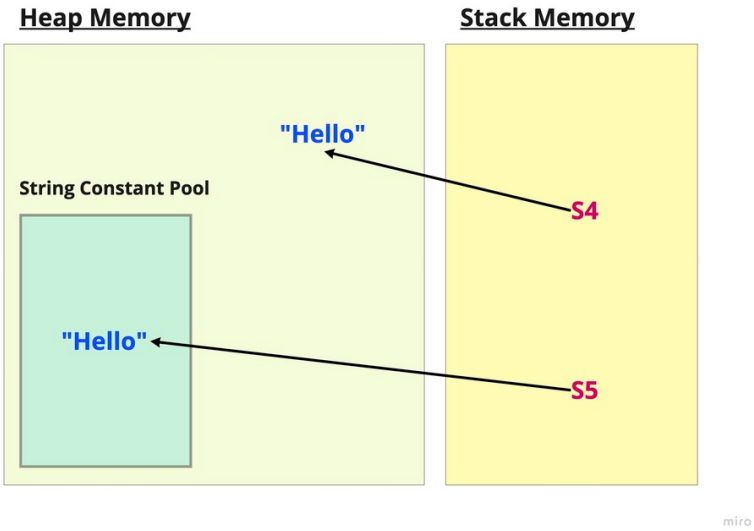
2. ***Literal*** ***Using new Keyword***	• ***Java Memory = Stack Memory + Heap Memory*** • ***String Constant Pool is a part of a Heap Memory.*** System.***out***.println(S1==S2); //true S1="Python"; • Now How many objects will be created in memory? - 2 object. • Now S1 will start pointing to "Python". System.***out***.println(S1==S2);//false 	@AMOL
Creating String Using "new" keyword	• String s4= new String("Hello"); • How many objects will be created? => 2 objects • 1 object is in SCP, 1 object is in Heap memory. • S4 will be pointing to "Hello" inside the heap memory only and the "Hello" object inside SCP will not be referred by s4	@Dhrumil @Manas @Gagan



@AMOL

- String s5= "Hello";
- If "***Hello***" is already present then no new object reference is created inside the SCP.

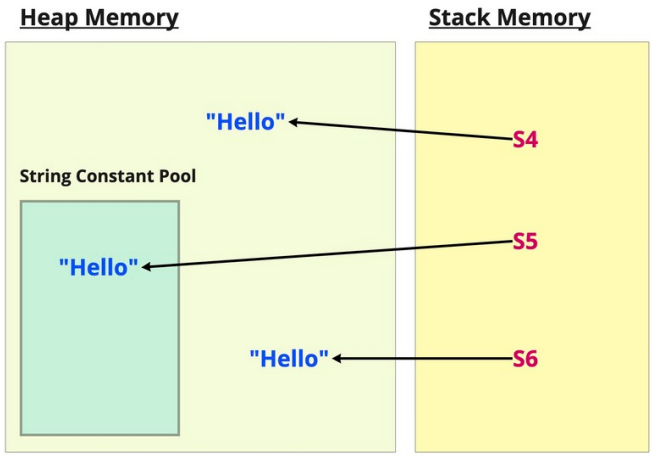
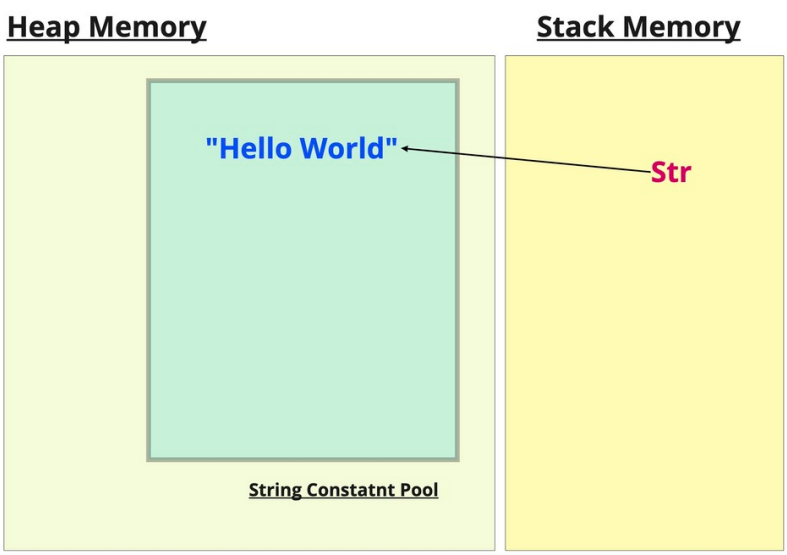
```
String s4= new String("Hello");
String s5= "Hello";
```



@Dhrumil

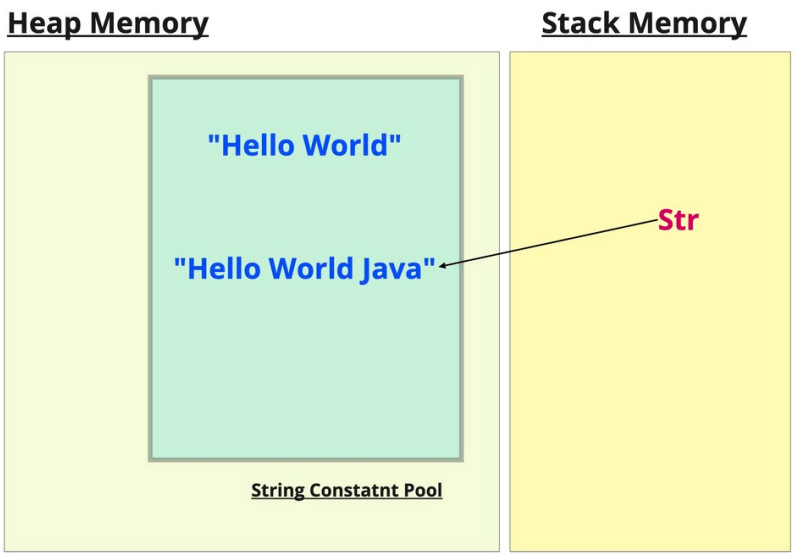
@Amol

- String s6= new String("Hello");
- In above case only one object is created inside the heap as "Hello" reference is already present inside the SCP.

	String s4= new String("Hello"); String s5= "Hello"; String s6= new String("Hello");  • System.*out*.println(s4==s5); //false System.*out*.println(s5==s6); //false System.*out*.println(s4==s6); //false • When we write like this:- String str = new String(); then it will always try to create 2 objects one in heap memory and one in SCP (for the first time).	@Amol
Are Strings Mutable/Immutable?	• String str="Hello World";  String str= "Hello World";	@Dhrumil @AMOL

-

```
str = str+ "Java";
System.out.println(str); //Hello WorldJava
```



String str= "Hello World";

str= str+ " Java";

miro

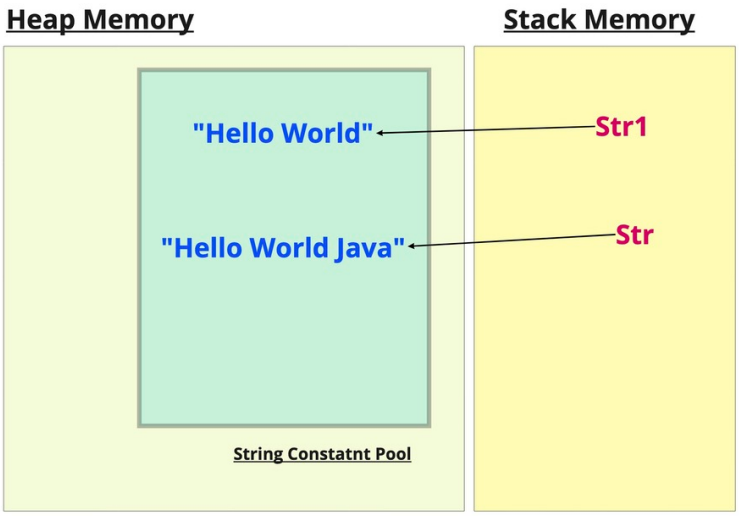
@AMOL

-

- ***String is Immutable object, can't change the value, just pointing of reference is changed.***

- String str1="Hello World";
  System.out.println(str1); //Hello World

@AMOL




```
String str= "Hello World";

str= str+ " Java";

String str1= "Hello World";
```

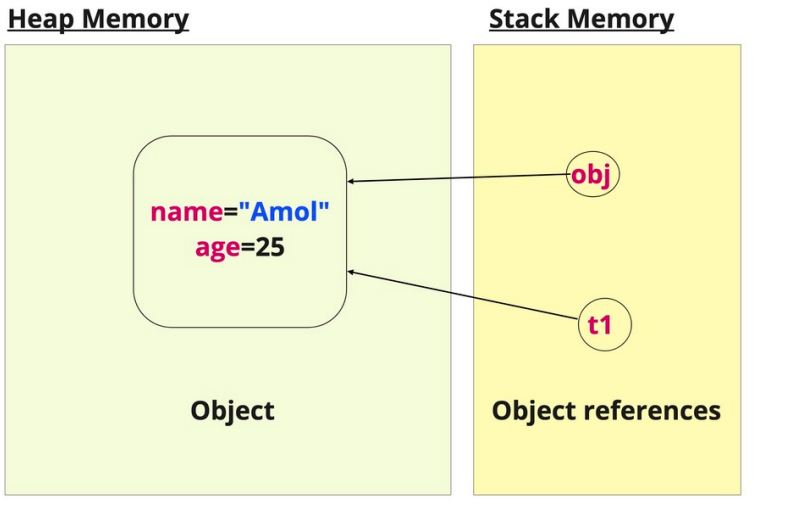
miro

- // No new object is going to be created, as its already present inside the SCP.

Why Strings are immutable in java	1. A String is an unavoidable type of variable while writing any application program. String references are used to store various attributes like username, password, etc. 2. Why only Strings are immutable in java--Because strings are most commonly used data types. 3. ***Memory Optimisation***:The immutability of String helps to minimize the usage in the heap memory. When we try to declare a new String object, the JVM checks whether the value already exists in the String pool or not. If it exists, the same value is assigned to the new object. This feature allows Java to use the heap space efficiently. 4. ***Security***:Consider an example of banking software. The username and password cannot be modified by any intruder because String objects are immutable. This can make the application program more secure.	@Gagan @AMOL
GC mechanism present inside the SCP?	• Do we have GC mechanism with **String Constant pool**? => Yes, it also destroy the unused references from the SCP. • SCP is always having unique values, doesn't contain duplicate	@Dhrumil

	values.	
	• String s1 = "Java" s1+=10; • Two objects present inside the SCP, but pointing of s1 is changed from "Java" to "Java10" in the SCP. • String str1 = "Java" • In this case, no new object is created, it will utilize the already created object which is present in the SCP. • Note: SCP is part of Heap post JDK 1.8 version released.	@Dhrumil
Can we pass ArrayList to Constructor?	• Yes, we can pass ArrayList as a argument to Constructor. • Concept: Need to add ArrayList first and then pass that ArrayList as argument to Constructor. • Example: need to add.	@Dhrumil
Can we pass Static Array to Constructor?	• Yes, we can pass Array as an argument to Constructor. • Concept: Need to define Array first and then pass that Array as argument to Constructor. • Example: need to add. • If you print directly static Array, it will give you memory address of the array • Way 1: Arrays.toString(StaticArray) ==> gives proper output • Way 2: Through For Loop/ Enhanced For Loop	@Dhrumil
Call By value vs Call/Pass By Reference?	• public class JavaCalls { String name; int age;	@Dhrumil @Amol

```
    public void add(int a, int b) {
        System.out.println(a+b);
    }

    public void getDetails(JavaCalls t1) {
        System.out.println(t1.name);
        System.out.println(t1.age);

    }

    public static void main(String[] args) {

        JavaCalls obj= new JavaCalls();
        obj.add(10,20);  //Call by value

        obj.name="Amol";
        obj.age=25;
        obj.getDetails(obj); //Call by reference
    }

}
```



@Amol

- Call/Pass By reference mainly used in Page Object Model Framework.
- One object can have multiple references.

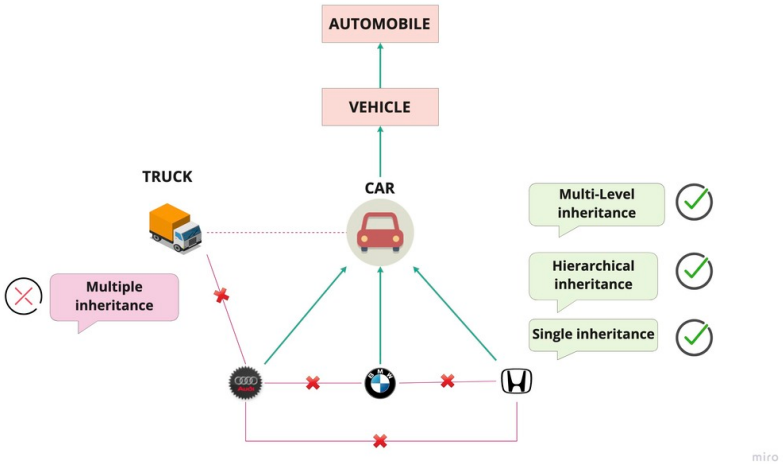
split(): function	• String m="Java_Python_Ruby_C#";	@Dhru

Returns a String Array *The split() method of String class can be used to extract a substring from a sentence.* ***It accepts arguments in the form of a regular expression.***	String test[]= m.split("_"); System.*out*.println(test[0]);//Java System.*out*.println(test[1]);//Python System.*out*.println(test[2]);//Ruby System.*out*.println(test[3]);//C# System.*out*.println(test[4]);//ArrayIndexOutOfBoundsExcept • for(String e:test) { System.*out*.print(e+" "); } ○ Java Python Ruby C# • String demo="xXJavaxXXPythonXxXRuby"; String[] top=demo.split("xX"); System.*out*.println(top[0]);//blank System.*out*.println(top[1]);//Java System.*out*.println(top[2]);//XPythonX System.*out*.println(top[3]);//Ruby	mil @Amol
Data driven Approach using String	• String empData = "tom; peter; 30; LA; USA; 909090"; String emp[] = empData.split(";"); for(String e:emp) { System.*out*.print(e); } • o/p: tom peter 30 LA USA 909090	@Dhru mil
	• Note: Instead of using split every time, always use split method once, store it in Array and then perform operations directly on Array.	@Dhru mil

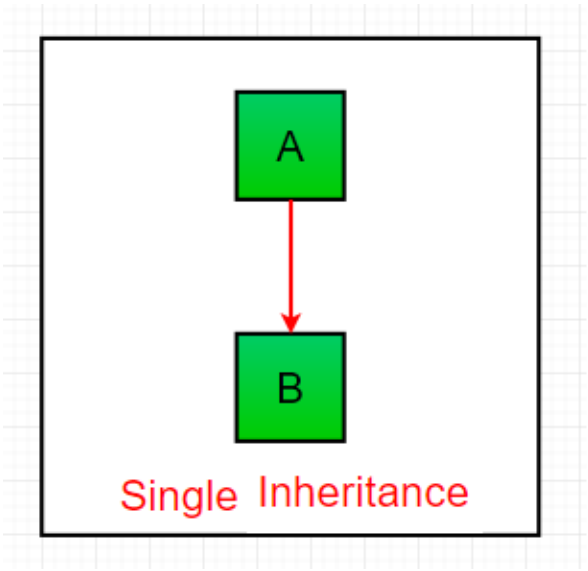
JavaSession -14

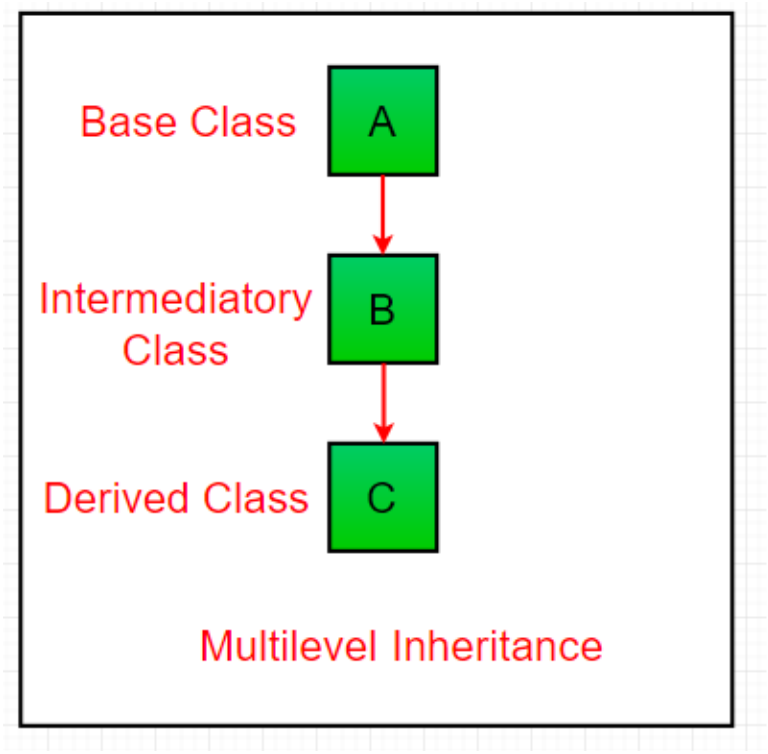
Topic:Inheritance_TopDown_Casting_MultiLevelInheritance_MethodOverriding

- Parent(Super) class and Child(Sub) class
- 'extends' keyword usage
- Method Overriding in Inheritance
  - Static and Private method can't be overridden, only public method can be overridden.
  - Static and Private methods can be overloaded
- @Override annotation- Optional usage but Good practice
- Compile time Polymorphism- Static
- Run time Polymorphism - Dynamic
- *Top casting vs Down casting*
- *reference type check*
- *ClassCastException*
- *Multiple inheritance is not possible. It will create diamond problem*
- *Method hiding*

Diagrammatic view of Inheritance	 ***NOTE:*** ☐ Inheritance represents the **IS-A relationship** which is also known as a ***parent-child*** **relationship.** ☐ Java supports ***Single/MultiLevel/ Hierarchical inheritances*** only. ☐ Java doesn't support Multiple and hybrid inheritance, They are achieved through interface only.	@AMOL @Gagan
What is Inheritance?	• Its the mechanism in Java through which one class acquires all the properties and behaviors OR inherits the methods and fields of another class. • you can create new classes that are built upon existing	@Dhrumil @AMOL

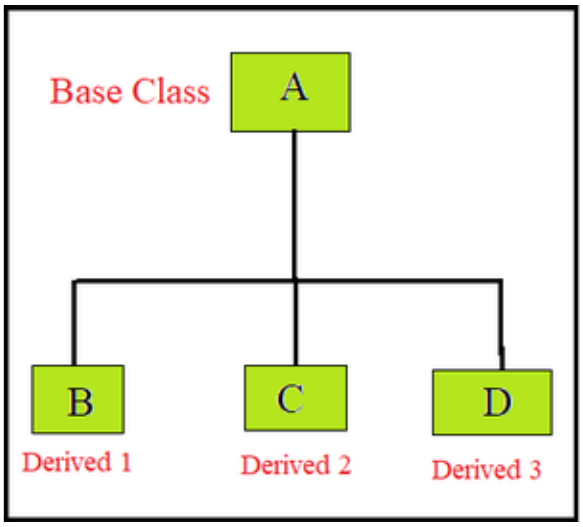
	classes. • When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.	
Why we are using Inheritance?	• **For Method Overriding** (so runtime polymorphism can be achieved). • **Code Reusability** : is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.	@Dhrumil @AMOL
The syntax of Java Inheritance ***"extends" keyword:*** *The extends keyword indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.*	``` class Subclass-name extends Superclass-name { //methods and fields } ``` • ***Super Class/Base Class/Parent Class***: is the class from where a subclass inherits the features. • ***Sub/Child/Derived/Extended Class:*** is a class which inherits the other class.	@Dhrumil @AMOL
Inheritance Rules Parent-Child Relationships	***NOTE:*** ☐ Child classes/siblings cannot inherit properties of each other. ☐ Child can inherit properties from its parent but parent cant inherit childs properties. ☐ Parent class can inherit from its parent class and so a child can inherit from its grandparent class. ☐ One parent class can have mutiple child classes but one child class cant have multiple parent classes.	@AMOL

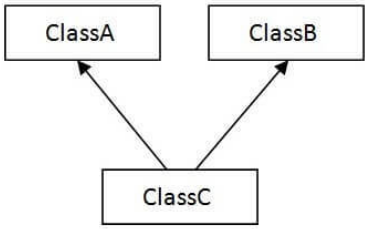
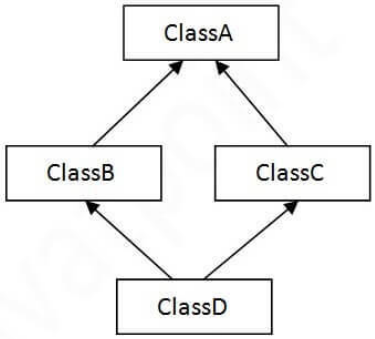
Single Level Inheritance	• When a single class inherits another class, it is known as a *single inheritance*.  • **Single Level inheritance template** Class Car{ } Class BMW extends Car{ }	@Sathish @Dhrumil
Multi Level Inheritance If method test() in class a is overridden in class b , but not overridden in class c , Class C referencing the method test - will be actually from the overidden test method in Class B	• When there is a chain of inheritance, it is known as *multilevel inheritance*. • class c inherits method from both the base class as well as the intermediary class.	@Sathish @Dhrumil @Anuradha

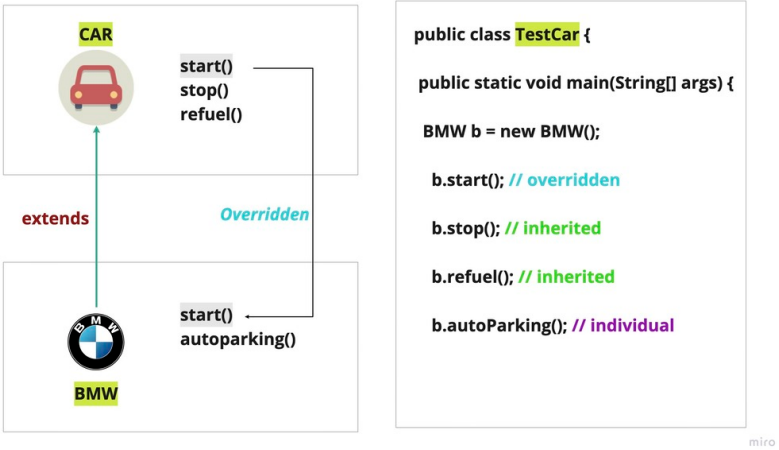
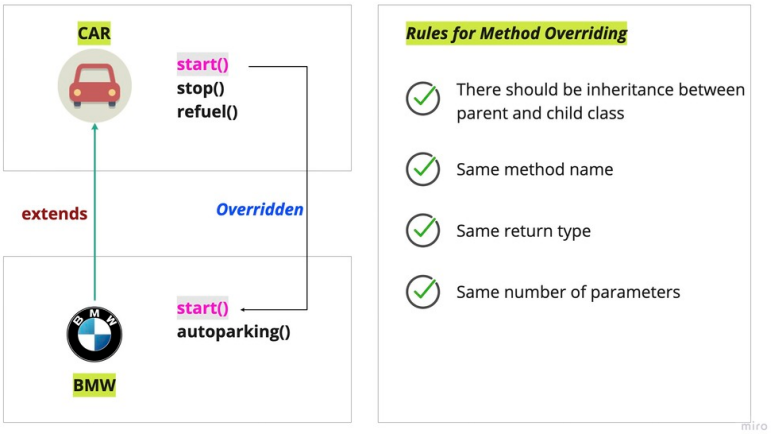


- **Multi level inheritance template**

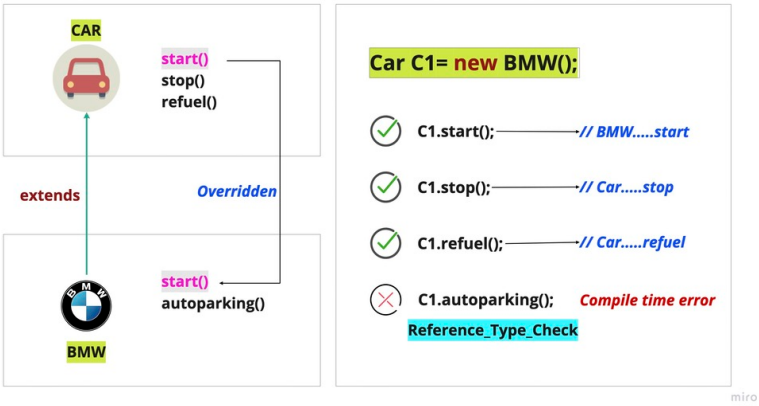
```
Class GrandParent{
 }
 Class Parent extends GrandParent{
 }
Class Child extends Parent{
 }
```

Hierarchical Inheritance Or Linear Inheritance	• When two or more classes inherits a single class, it is known as *hierarchical inheritance*. 	@Dhrumil

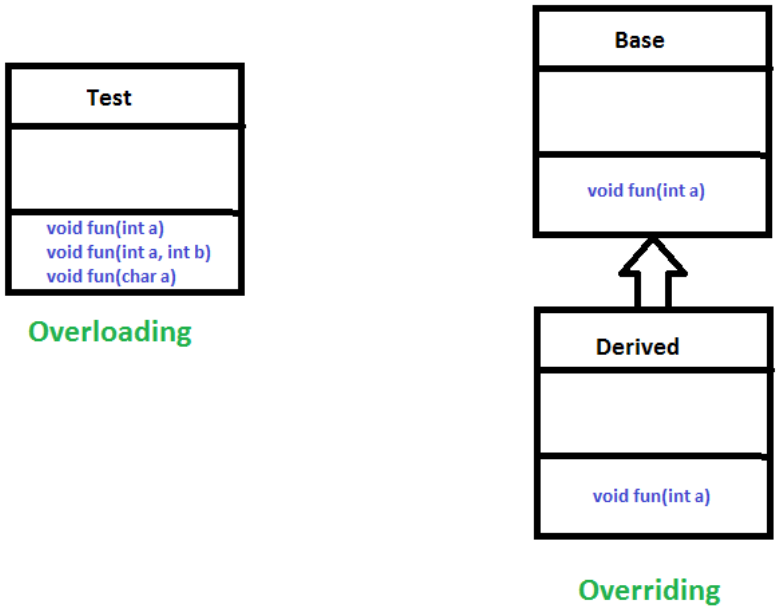
Multiple inheritance Hybrid inheritance	• Multiple inheritance and hybrid inehritance is not allowed in java through Class (Diamond Problem)  4) Multiple  5) Hybrid **Template** Class A{ } Class B{ } Class C extends A, B{ } // Not Allowed	@Sathish @AMOL
Q) Why multiple inheritance is not supported in java?	• To reduce the complexity and simplify the language, multiple inheritance is not supported in java. • In above example where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class. • Since compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error.	@AMOL
Method overriding Concept	• If subclass (child class) ***has the same method as declared in the parent class***, it is known as method overriding in Java. • In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.	@AMOL

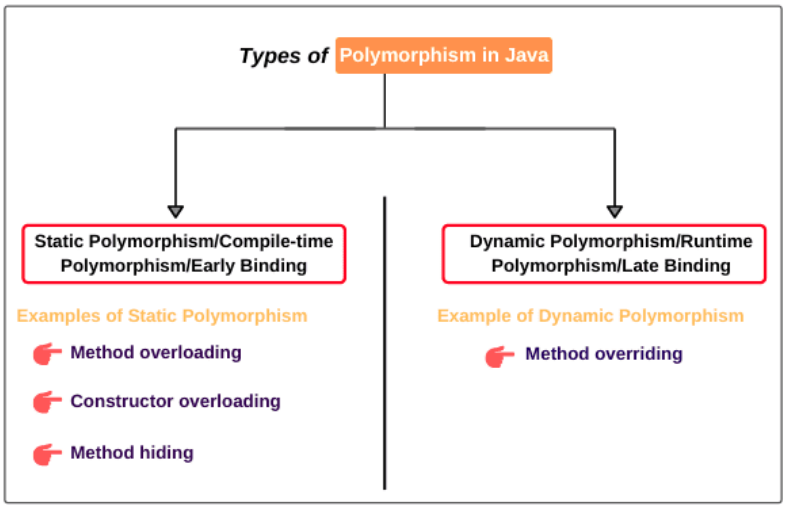
Usage of Java Method Overriding	• Method overriding is used ***to provide the specific implementation*** of a method which is already provided by its superclass. • Method overriding is used for ***runtime polymorphism.***	@AMOL
Method overriding Exaample	• When you have a method in a parent class as well as in child class with the same name and same number of arguments is called **Method Overriding**.	@Sathish @Dhrumil
		@AMOL
Rules for method overriding		@AMOL
Can parent class access methods from child class?	☐ Parent class /grand parent cannot access any property or method from child class.	@Dhrumil

<table>
<tr><td>IMP: Private & Static methods can not be overridden in java</td><td>☐ We can not override private & static methods in java
☐ We can not override methods with final keyword in java
☐ We can overload private, static methods in java</td><td>@Gagan
@anupama
@AMOL</td></tr>
<tr><td>Why can we not override static method?</td><td>• It is because the static method is bound with class whereas instance method is bound with an object.
• Static belongs to the class area, and an instance belongs to the heap area.
• So,Can we override java main method?
◦ No, because the main is a static method.</td><td>@AMOL</td></tr>
<tr><td>Why can we not override private method?</td><td>• No, we cannot override private methods in Java.
• Because Private methods in Java are not visible to any other class which limits their scope to the class in which they are declared.</td><td>@AMOL</td></tr>
<tr><td>Method Overloading vs Method Overriding</td><td><table>
<tr><th>No.</th><th>Method Overloading</th><th>Method Overriding</th></tr>
<tr><td>1)</td><td>Method overloading is used to increase the readability of the program.</td><td>Method overriding is used to provide the specific implementation of the method that is already provided by its super class.</td></tr>
<tr><td>2)</td><td>Method overloading is performed within class.</td><td>Method overriding occurs in two classes that have IS-A (inheritance) relationship.</td></tr>
<tr><td>3)</td><td>In case of method overloading, parameter must be different.</td><td>In case of method overriding, parameter must be same.</td></tr>
<tr><td>4)</td><td>Method overloading is the example of compile time polymorphism.</td><td>Method overriding is the example of run time polymorphism.</td></tr>
<tr><td>5)</td><td>In java, method overloading can't be performed by changing return type of the method only. Return type can be same or different in method overloading. But you must have to change the parameter.</td><td>Return type must be same or covariant in method overriding.</td></tr>
</table></td><td>@AMOL</td></tr>
<tr><td>Method Hiding concept</td><td>• Method hiding can be defined as, "if a subclass defines a static method with the same signature as a static method in the super class, in such a case, the method in the subclass hides the one in the superclass." In that case the method of superclass is hidden by the subclass.
• It signifies that : The version of a method that is executed will NOT be determined by the object that is used to invoke it, since its static and is decided by compiler at compile time itself</td><td>@Gagan
@vibha
@AMOL</td></tr>
</table>

	NOTE: ☐ Static methods are hidden, non-static methods are overriden.	
Method Overriding Vs. Method Hiding	<table><tr><th>Method Hiding</th><th>Method Overriding</th></tr><tr><td>Both methods must be static.</td><td>Both methods must be non-static.</td></tr><tr><td>Method resolution takes care by the compiler based on the reference type.</td><td>Method resolution takes care by JVM based on runtime object.</td></tr><tr><td>It is considered as compile-time polymorphism or static polymorphism or early binding.</td><td>It is considered as runtime polymorphism or dynamic polymorphism or late binding.</td></tr></table>	@Vishnu priy @AMOL
TopCasting	• A child class object referred by **parent/super parent class** reference variable this concept is called top casting. • Casting does not change the actual object type. Only the reference type gets changed. • **Car c1= new BMW();**  **Note:** ☐ By using top-casting, ***we can inherit ONLY Overridden/Inherited methods but not individual methods.***	@Anupama @AMOL
Why do we need Upcasting in Java?	• We need up casting when we want to write code that deals with only the parent class.	@AMOL
DownCasting	• Parent class object referred by any child class reference this concept is called as downcasting. • **BMW b1= new Car();**	@Anupama

	• **Honda h1=(Honda)new Car();** • Here compiler doesn't throw any error compile time but run time **we can get ClassCastException**	
Class Cast Exception	• When we hold the object of parent class into reference variable of child class and then try to access parent class properties then it always throw **"Class cast Exception" at runtime.** • (down casting; putting big box into small box, it can be done by tearing big box into small pieces and inserting, so there will be no compile time error, but @ runtime we will not be able to access any thing properly)	@Gagan @vibha
Reference type check concept	• whenever we try to access individual method of child class with the reference variable of parent class, java will always try to match the ref type, this concept is called **"ref type check."**	@Gagan
Polymorphism	• Polymorphism is a concept by which we can perform a single task in different ways. • The word “poly” means many and “morphs” means forms, So it means many forms. • Method Overloading & Method Overriding both will comes under polymorphism. • **Method Overloading is static/compile time polymorphism. Here compiler takes decision at compile time.** • **Method overriding always comes under runtime/dynamic polymorphism. Here decision has taken at run time only.**	@anupama @AMOL



Types of Polymorphism		@AMOL

Compile time Polymorphism vs Run time Polymorphism

Sr.No	Compile Time Polymorphism	Run time Polymorphism
1	In Compile time Polymorphism, the call is resolved by the compiler.	In Run time Polymorphism, the call is not resolved by the compiler.
2	It is also known as Static binding, Early binding and overloading as well.	It is also known as Dynamic binding, Late binding and overriding as well.
3	Method overloading is the compile-time polymorphism where more than one methods share the same name with different parameters or signature and different return type.	Method overriding is the runtime polymorphism having same method with same parameters or signature, but associated in different classes.
4	It is achieved by function overloading and operator overloading.	It is achieved by virtual functions and pointers.
5	It provides fast execution because the method that needs to be executed is known early at the compile time.	It provides slow execution as compare to early binding because the method that needs to be executed is known at the runtime.

@AMOL

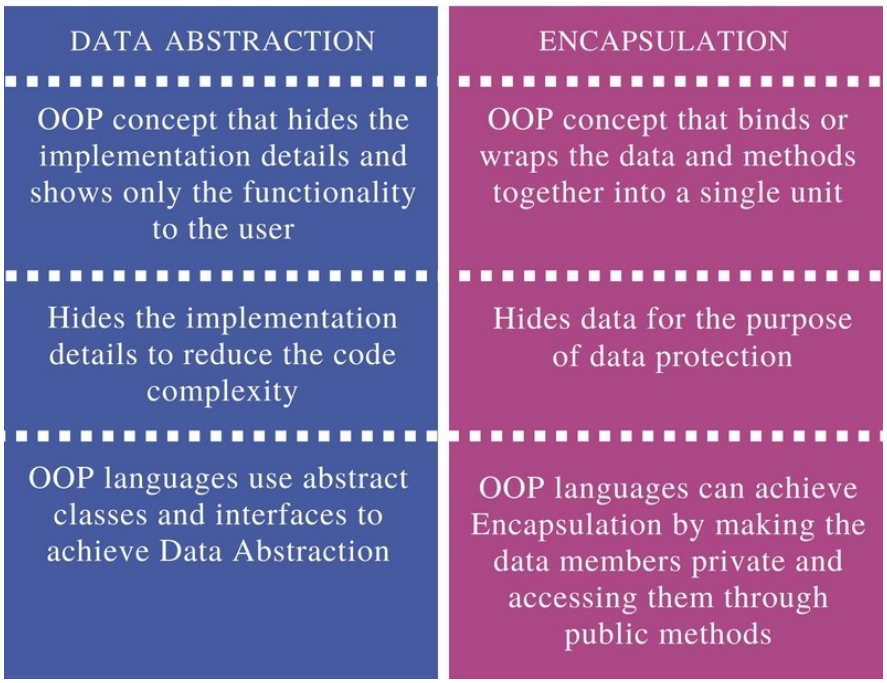
JavaSession -15

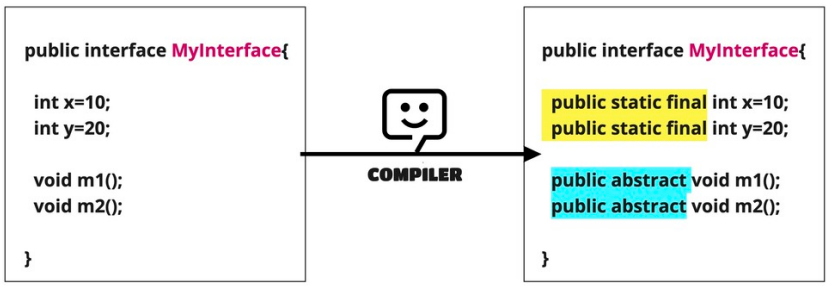
Topic:Abstraction_InterfaceConcept_MultipleInheritance_DiamondProblem

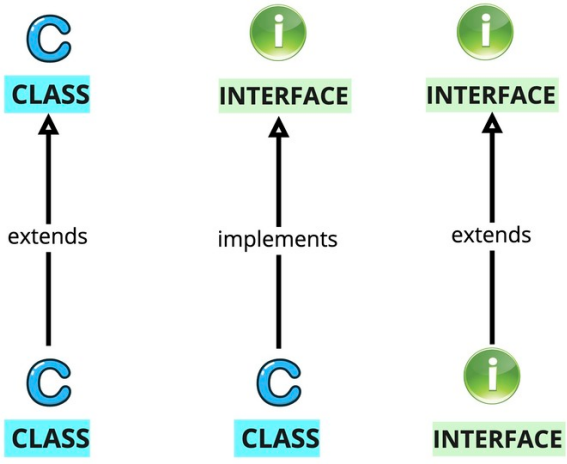
- Multiple Inheritance through Interface.
- Interface
- implements,extend keywords
- Abstract Method concept
  - No Method body
  - Only Method Declaration
- Method implementation completed inside the class only.
- Class can implements multiple interfaces together and can be done by "Comma" separated.
  - Interface A
  - Interface B
  - Interface C
  - Class D implements A, B, C
- Can't create object of Interface
- **Down Casting is not allowed in interfaces,because we can't create object for Interface.**
- Abstract method can't be static & final in nature.
- Abstract method can't be declare as private
- Interface variable by default Static and final
- in nature.

- After JDK 1.8 release,interface can have
  - Static method with method body
  - Default method with method body
- Default methods can be overridden.
- Interface can only have parent interface, it doesn't have parent class.
- Final method can't be overriden.
- Final keyword used to provide constant values, to prevent Inheritance & as well as method overriding
- final variables are allowed in interface but final methods are not allowed in interface

Abstraction ***in Java***	• Abstraction in Java is another OOPs principle that manages complexity. • It is a process of hiding complex internal implementation details from the user and providing only necessary functionality to the users. • In other words, abstraction in Java is a technique by which we can hide the data that is not required to a user. • It hides all unwanted data so that users can work only with the required data. It removes all non-essential things and shows only important things to users.	@AMOL
Realtime Examples of Abstraction in Java	• We all use an ***ATM machine*** for cash withdrawal, money transfer, retrieve min-statement, etc in our daily life.But we don't know internally what things are happening inside ATM machine when you insert an ATM card for performing any kind of operation.	@AMOL
Why do we need Abstraction?	• It reduces the complexity of viewing the things. • Avoids code duplication and increases reusability. • Helps to increase the security of an application or program as only important details are provided to the user.	@AMOL
How to achieve Abstraction in Java?	• There are two ways to achieve or implement abstraction in java program. • They are as follows: 1. **Abstract class (0 to 100%)** 2. **Interface (100%)**	@AMOL @vibha

Difference between Abstraction and Encapsulation		@AMOL
## *Interface* ***What is an interface?***	• An interface is a collection of **abstract methods and constants (i.e. static and final fields).** It is used to achieve complete abstraction. • **Every interface in java is abstract by default.** So, it is not compulsory to write abstract keyword with an interface. • Once an interface is defined, we can create any number of separate classes and can provide their own implementation for all the abstract methods defined by an interface. • A class that implements an interface is called **implementation class**. A class can implement any number of interfaces in Java • interfaces cannot have business logic. • A class can implement more than one interface. • Interface cannot have parent class but it can have parent interface. • A class can extend and implement as well, but extend keyword should be followed by implements.	@vibha @Simran @AMOL
Why do we use interface?	1. It is used to achieve ***100% full abstraction IS-A relationship.*** 2. By interface, we can ***support the functionality of multiple inheritance.***	@AMOL @vibha
Interface Syntax	• accessModifier **interface** interfaceName { // declare constant fields.	@AMOL

	// declare methods that abstract by default. } ***NOTE***: ☐ **Interface fields are public, static and final by default.** ☐ **Interface methods are public and abstract by default.** • The Java compiler adds **public** and **abstract** keywords before the interface method. Moreover, it adds **public, static** and **final** keywords before data members. 	@AMOL
Interface Variables	• **All the variables declared in an interface are considered as *public, static, and final* by default and acts like constant. We cannot change their value once they initialized.** • The variables will be available to any classes that implement interface because it is by default public, static, and final. • The values can also be used in any method as part of the variable declaration or anywhere in the class. • How to call directly : **InterfaceName.variableName**	@AMOL
Java JDK-8 onwards...	• We can also declare default methods and static methods with method bodies inside interfaces. • An interface can also declare its private methods.	@AMOL
Interface KeyPoints/ Features	1. Interface provides pure abstraction in java. It also represents the Is-A relationship. 2. It can contain three types of methods: abstract, default, and static methods. 3. All the (non-default) methods declared in the interface are by default abstract and public. So, there is no need to write abstract or public modifiers before them.	@AMOL

	4. The fields (data members) declared in an interface are by default public, static, and final. Therefore, they are just public constants. So, we cannot change their value by implementing class once they are initialized. 5. **Interface cannot have constructors.** 6. The interface is the only mechanism that allows achieving multiple inheritance in java. 7. A Java class can implement any number of interfaces by using keyword implements. 8. Interface can extend an interface and can also extend multiple interfaces.	
Interface Rules--	• We can't create an object of an interface But You can provide reference to the child class object with parent interface reference variable. ◦ **USMedical us= new FortisHospital();** //TOP-CASTING IS ALLOWED ◦ *By using **'us'** object reference you can access the methods from USMedical interface only.* • Down Casting is not allowed in interfaces (not even @compile time) because you can't create an object of an interface. ◦ **FortisHospital fh = new USMedical();** //DOWN-CASTING IS NOT ALLOWED  	@AMOL @AMOL

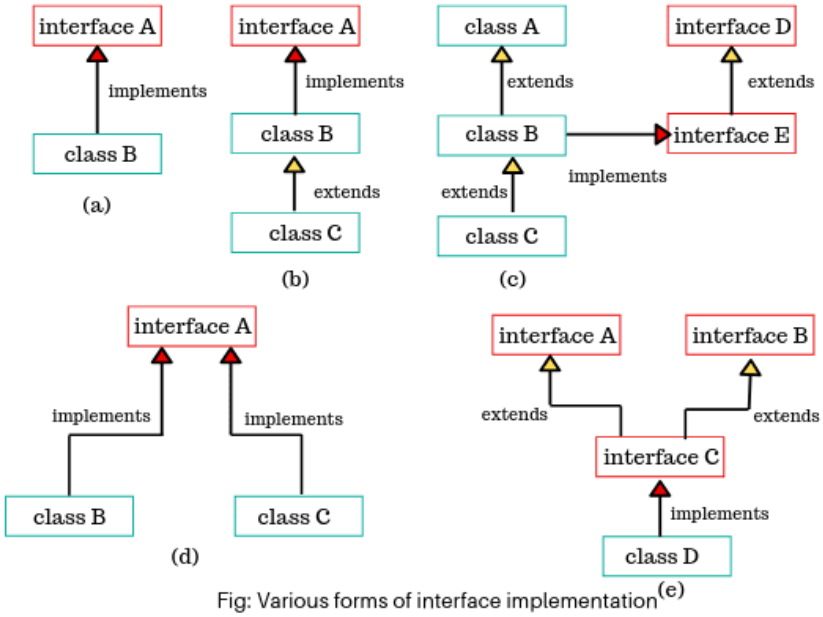
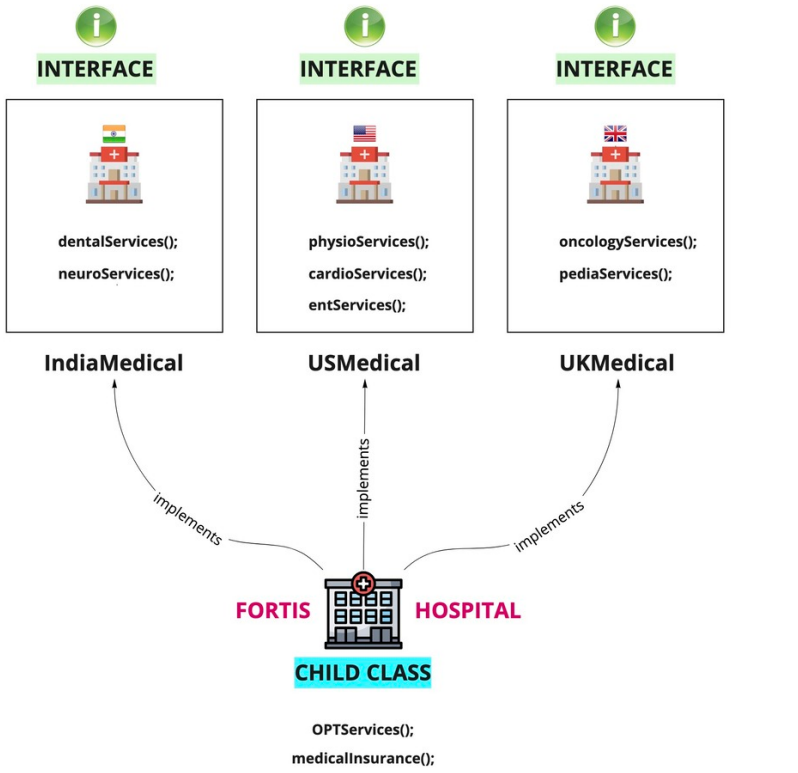


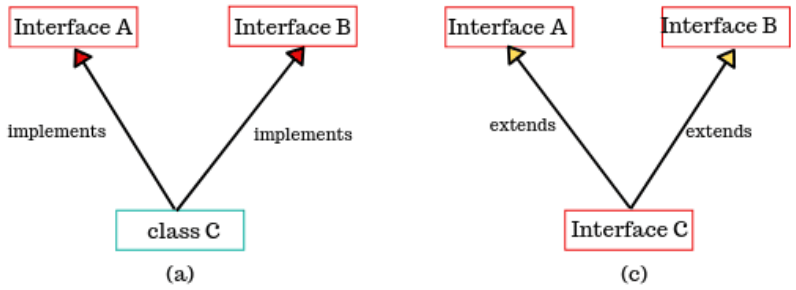
Fig: Various forms of interface implementation

- If we declare a variable in an interface, it must be initialized
- at the time of declaration. It cannot have instance variables.
- A class that implements an interface, must provide its own ***implementations of all the methods defined in the interface.***

@AMOL

- You should not declare interfaces with private/protected/final keywords or it will generate compile time error.
- If you add any new method in interface, all the classes which implement that interface must provide implementations for newly added method because all methods in interface are by default abstract.

Interface illustration		@AMOL
Can we have one common method for above 3 interfaces ?	• YES we can have it ,But it will be implemented only once in the child class which is implementation class of all 3 interfaces.	@AMOL
Can we declare interface's abstract methods as static?	• NO we can't declare abstract methods of interfaces as static because we can't override the static methods. • If we declare abstract methods as static we won't be able to provide its implementation in the child class so abstract methods can never be static.	@AMOL
Can we declare interface's abstract methods as private?	• NO we can't. • If we declare abstract methods as private then we won't be able to access them in implementation class as private methods can't be accessed outside of the class	@AMOL
Multiple Inheritance in Java	• Multiple inheritance in java is achieved through Interfaces using **implements** key word.	@vibha @AMOL

by Interface	• Interfaces specify what a class must do and not how. It is the blueprint of the class. • When a class implements more than one interface, or an interface extends more than one interface, it is called multiple inheritance.  Fig: Various forms of Multiple inheritance by Interface in Java	
Java Class vs Interface	1. **Keyword**: a. A class is declared by using a keyword called class. b. An interface is declared by using a keyword interface. 2. **Instantiate**: a. A class can be instantiated. The keyword new can only create an instance of a class. b. An interface can never be instantiated. That is an object of interface can never be created. 3. **Variables**: a. Variables in a class can be declared using any access modifiers such as public, protected, default, and private. They can also be static, final, or neither. b. All variables in an interface are always public, static, and final. 4. **Methods**: a. Methods declared in a class are implemented. i.e, methods have a body. Methods that have a body is called concrete method. b. Methods declared in an interface cannot be implemented. i.e, In an interface, methods have no body. It contains only abstract method. c. ***Note***: Java 8 introduces default method, which allows us to give the body of a method in an interface. 5. **Access modifiers**: a. The members of a class can be declared with any access modifiers such as private, default, protected, and public.	@AMOL

	b. The members of an interface are always public by default. 6. **Constructors**: a. A class can have constructors to initialize instance variables. b. An interface cannot have any constructors. 7. **Inheritance**: a. Class allows only multilevel and hierarchical inheritances but do not support multiple inheritance. b. Interface supports all types of inheritance such as multilevel, hierarchical, and multiple inheritance. 8. **Implement**: a. A class can implement any number of the interface. b. Interface cannot implement any class. 9. **Extends**: a. A class can extend only one class at a time. b. An interface can extend multiple interfaces at a time. 10. **Use**: a. Interface is used for defining behavior that can be implemented by any class anywhere in the class hierarchy. b. A class is used to define the attributes and behaviors of an object.11. 11. **Final:** a. The method in an interface cannot be final because if you declare a method as final in interface, the method cannot be modified by any of its subclasses. b. A method in a class can be declared as final. 12. **Main method:** a. A class may contain main() method. b. Interface cannot have main() method.	
In Java, Multiple Inheritance is not supported through Class but it is possible by Interface, why?	• In multiple inheritance, subclasses are derived from multiple superclasses.If two superclasses have the same method name then which method is inherited into subclass is the main confusion in multiple inheritance.That's why Java does not support multiple inheritance in case of class. • But, it is supported through an interface because there is no confusion. This is because its implementation is provided by the implementation class.	@AMOL

JavaSession -16

Topic:AbstractClass_WebDriver_With_Interface_RealTimeExample

- **abstract class explanation with real time example for webpage**
- **class to abstract class we can use extends keyword**
- **class to class ->extends**
- **class to interface->implements**
- **When to use abstract class & interface**
- **Difference between abstract class and interface**
- **abstract class allows abstract + non abstract methods**
- **we can't create objects for abstract class but we can create constructor for abstract class.**
- **constructor will call when you create object for child class**
- **default constructor created by JVM if you don't create a constructor in the child class**
- **example for interface concept is WebDriver implementation with different browsers**
- **static is not part of OOPS because we nevkker create objects for static properties**
- **abstract class contains once constructor & child class contains no constructor then sequence always first execute parent class constructor and then child class constructor**

Abstract Class	• An abstract class must be declared with an **abstract** keyword. • It can have **abstract and non-abstract** methods.It can be abstract even without any abstract method. • Abstract class allows to define private, final, static and concrete methods. Everything is possible to define in an abstract class as per application requirements. • It cannot be instantiated.You Cannot create object of abstract class. • It can have constructors and static methods also. • It can have final methods which will force the subclass not to change the body of the method. • Can create constructor of abstract class, This constructor will be called when object will be created of child class. • It can implement one or more interfaces in java.	@Abhay @AMOL

	Abstract class	**Interface**
	1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
	2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
	3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
	4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
	5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.
	6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
	7) An **abstract class** can be extended using keyword "extends".	An **interface** can be implemented using keyword "implements".
	8) A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

@AMOL

--

NOTE:

- ☐ *You can't instantiate/create the object of both abstract class and interface.*
- ☐ *But you can have the constructor inside the abstract class while interface doesn't have any.*
- ☐ *The constructor of an abstract class will be called when you create an object of its child class.*

Abstract class vs Interface 12 points comparision	**1. Keyword(s) used:** a. Two keywords abstract and class are used to define an abstract class. b. Only one keyword interface is used to define an interface. **2. Keyword used by implementing class:** a. To inherit the abstract class, we use the extends keyword. b. To implement an interface, we can use the implements keyword. **3. Variables:** a. Abstract class can have final, non-final, static, and non-static variables. b. Interface cannot have any instance variables. It can have only static variables.	@AMOL

4. **Initialisation:**
    a. The abstract class variable does not require performing initialization at the time of declaration.
    b. Interface variable must be initialized at the time of declaration otherwise we will get compile-time error.

5. **Method:**
    a. Every method present inside an interface is always public and abstract whether we are declaring or not. That's why interface is also known as pure (100%) abstract class.
    b. An abstract class can have both abstract and non-abstract (concrete) methods.

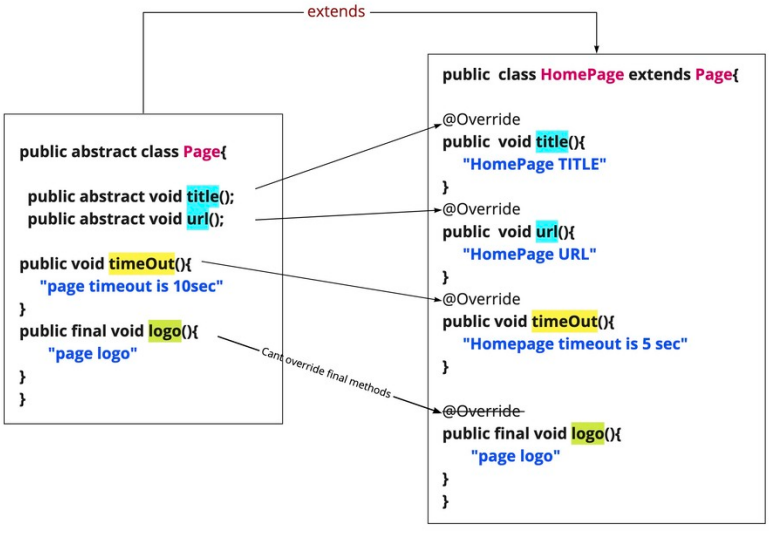
6. **Constructors:**
    a. Inside an interface we cannot declare/define a constructor because the purpose of constructor is to perform initialization of instance variable but inside interface every variable is always static.
    b. Therefore, inside the interface, the constructor concept is not applicable and does not require.
    c. Since an abstract class can have instance variables. Therefore, we can define constructors within the abstract class to initialize instance variables.

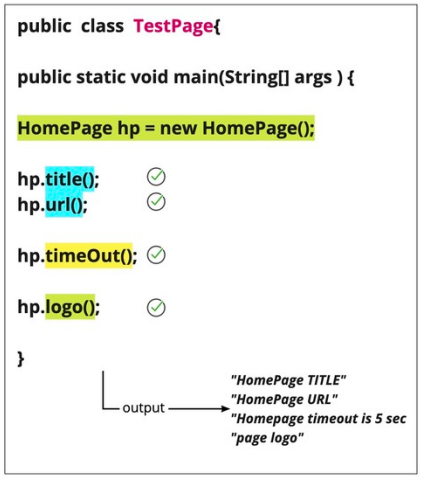
7. **Static and Instance blocks:**
    a. We cannot declare instance and static blocks inside an interface. If you declare them, you will get compile time error.
    b. We can declare instance and static blocks inside abstract class.

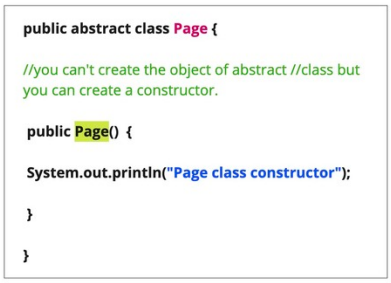
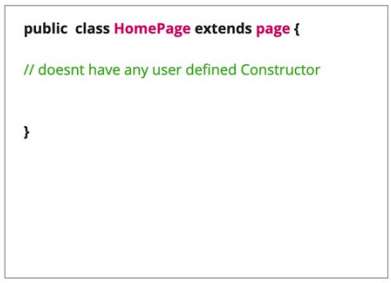
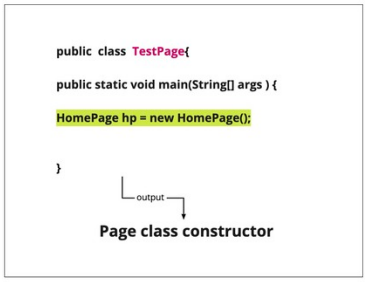
8. **Access modifiers:**
    a. We cannot define any private or protected members in an interface. All members are public by default.
        i. There is no restriction in declaring private or protected members inside an abstract class.

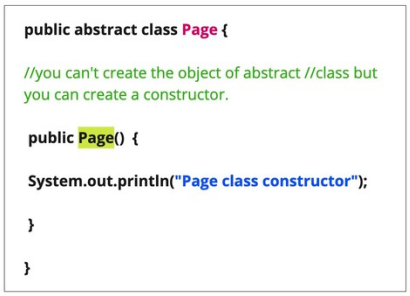
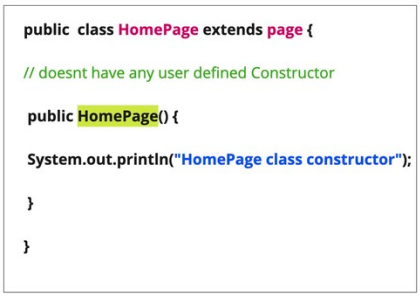
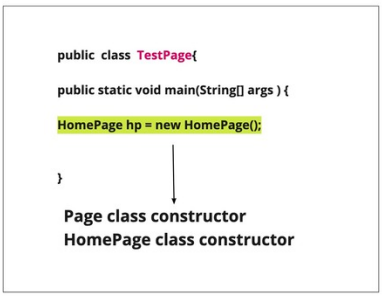
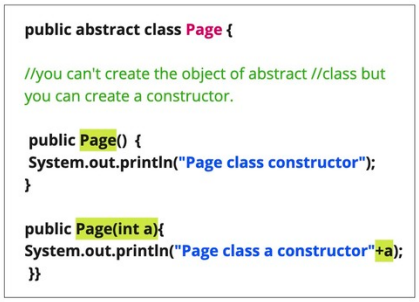
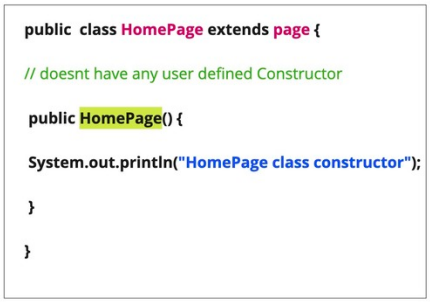
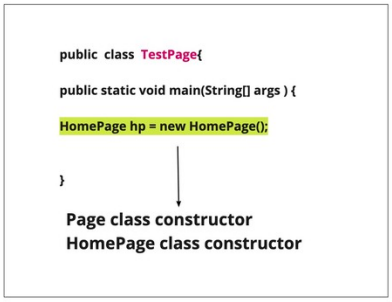
9. **Single vs Multiple inheritance:**
    a. A class can extend only one class (which can be either abstract or concrete class).
    b. A class can implement any number of interfaces.

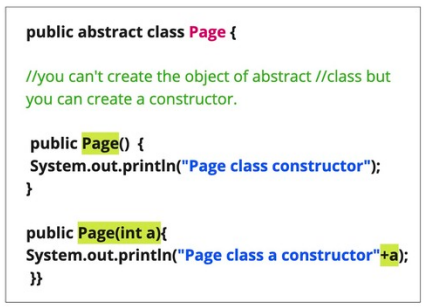
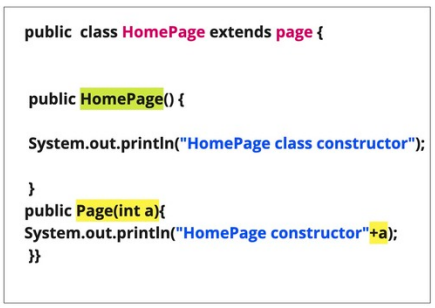
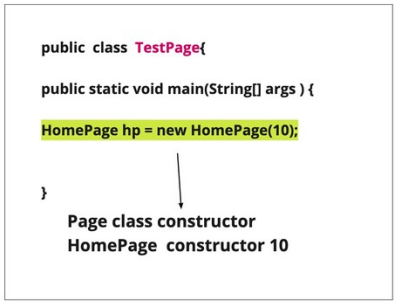
	10. Default Implementation: a. An abstract class can provide a default implementation of a method. So, subclasses of an abstract class can just use that definition but subclasses cannot define that method. b. An interface can only declare a method. All classes implementing interface must define that method. **11. Difficulty in making changes:** a. It is not difficult to make changes to the implementation of the abstract class. For example, we can add a method with default implementation and the existing subclass cannot define it. b. It is not easy to make changes in an interface if many classes already implementing that interface. For example, suppose you declare a new method in interface, all classes implementing that interface will stop compiling because they do not define that method. **12. Uses:** a. If you do not know anything about the implementation. You have just requirement specification then you should go for using interface. b. If you know about implementation but not completely (i.e, partial implementation) then you should go for using abstract class.	
What is abstract method in java?	• A method which is declared as abstract and does not have implementation is known as an abstract method. • If there is an abstract method in a class, that class must be abstract. • If you are extending an abstract class that has an abstract method, you must either provide the implementation of the method or make this class abstract. ***NOTE:*** ☐ *It is mandatory to write 'abstract' keyword before abstract methods in abstract class ,but in case of interfaces it's not compulsory to write 'abstract' keyword before abstract methods because all the methods of an interface are abstract by default.*	@AMOL
Rules of Abstract method in Java	• Abstract method can only be declared in an abstract class. • A non-abstract class cannot have an abstract method whether it is	@AMOL

	inherited or declared in Java. • It must not provide a method body/implementation in the abstract class for which it is defined. • Method name and signature must be the same as in the abstract class. • Abstract method cannot be static or final. • It cannot be private because the abstract method must be implemented in the subclass. If we declare it private, we cannot implement it from outside the class.	
NOTE:	☐ If a new abstract method is added in the abstract class, all non-abstract subclass which extends that abstract class, must implement the newly added abstract method. If it does not implement all the abstract method, the class must be declared as abstract. ☐ If a new instance method is added in the abstract class, all non-abstract subclass which extends that abstract class, is not necessary to implement newly added instance method.	@AMOL
Abstract class demonstration		@AMOL @AMOL



Why abstract class has constructor even though we cannot create object of it?	• We cannot create an object of abstract class but we can create an object of subclass of abstract class. When we create an object of subclass of an abstract class, it calls the constructor of subclass. • This subclass constructor has super in the first line that calls constructor of an abstract class. Thus, the constructors of an abstract class are used from constructor of its subclass. • If the abstract class doesn't have a constructor, a class that extends that abstract class will not get compiled.	@AMOL
How Constructor is called in abstract class? ***CASE 1:*** *When child class doesn't have user-defined constructor and abstract class has its own constructor*	  	@AMOL

CASE 2: *When child class has one user-defined constructor + abstract class has one its own constructor*	   miro ☐ In the first line of constructor, internally super will call the constructor of an abstract class. The control of execution will be immediately transferred to the constructor of abstract class.Therefore, the first output is “Page Class constructor”. ☐ After executing abstract class constructor, control of execution again comes back to execute subclass constructor. The second output is “HomePage class constructor”.	@AMOL
CASE 3: *Overloading the abstract class constructor with same scenarios as in CASE 2*	   miro	@AMOL

CASE 4: *Overloading both abstract class constructor and child class constructor*	   ***NOTE:*** ☐ *It's mandatory to have a parent abstract class default constructor if you want to call child class constructor .* ☐ *If user doesn't define the constructor inside abstract class jvm will create a default constructor.*	@AMOL
	• Abstract class provides Abstraction • When there is constructor in child and parent class preference will be given to parent abstract class constructor • It is compulsory to create default constructor in parent class.	@Simran
	• Zero Abstract /No abstract methods-0% Abstraction • only abstract methods-100% Abstraction • abstract methods+ non abstract methods-Partial abstraction	@Simran
When to use abstract class ?	• When we talk about implementation but not completely then we should go for abstract class.	@Abhay
When to use Interface ?	• If we don't know anything about implementation just we have requirements/specification then we should go for interfaces	@Abhay

JavaSession -17

Topic: ExceptionHandling

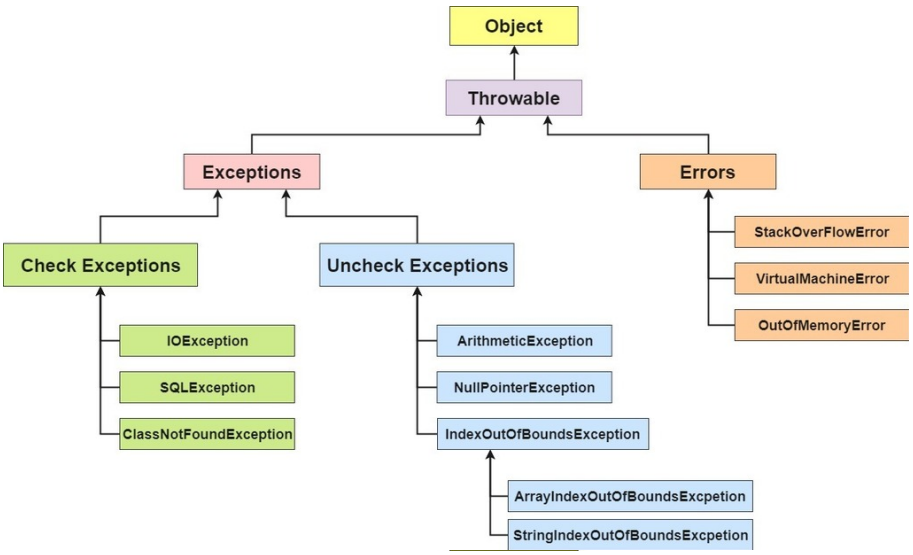
- Exception handling - Try Catch Mechanism
- Throws keyword
- Throw
- Difference between throw and throws keyword
- When to use throw and throws keyword.
- Difference between exception and error
- Error concept
- How to handle errors in the code?
- Good practice is to handle exception in the same method when you use throws keyword

Exception	• An unwanted or unexpected event that occurs during the execution of the program and interrupts the normal flow of program instructions. • These are the errors that occur at compile time and run time. • It occurs in the code written by the developers. It can be recovered by using the try-catch block and throws keyword.	@Abhay
Errors	• Errors are problems that mainly occur due to the lack of system resources. • It cannot be caught or handled. It indicates a serious problem. • It occurs at run time. These are always unchecked	@AMOL

Exception vs Error		@AMOL @vibha

Basis of Comparison	Exception	Error
Recoverable/ Irrecoverable	Exception can be recovered by using the try-catch block. An error cannot be recovered.	
Type	It can be classified into two categories i.e. checked and unchecked.	All errors in Java are unchecked.
Occurrence	It occurs at compile time or run time.	It occurs at run time.
Package	It belongs to java.lang.Exception package.	It belongs to java.lang.Error package.
Known or unknown	Only checked exceptions are known to the compiler.	Errors will not be known to the compiler.
Causes	It is mainly caused by the application itself.	It is mostly caused by the environment in which the application is running.
Example	**Checked Exceptions:** SQLException, IOException **Unchecked Exceptions:** ArrayIndexOutOfBoundException, NullPointerException, ArithmaticException	Java.lang.StackOverFlow, java.lang.OutOfMemoryError

What is Throwable is java?	• Throwable is a class which is derived from Object class, is a top of exception hierarchy from which all exception classes are derived directly or indirectly. It is the root of all exception classes. • Throwable is parent class of Exception class. Error ,Exception both are children of Object class.	@Dhrumil @AMOL



Checked Exceptions	• The exceptions that are ***checked by Java compiler at compilation time*** is called checked exception in Java. • If a method throws a checked exception in a program, the method must either handle the exception or pass it to a caller method.	@AMOL

	• Checked exceptions must be handled either by using try and catch block or by using throws clause in the method declaration. If not handles properly, it will give a compile-time error. • List of Checked Exceptions in JavaA list of some important checked exceptions are given below: ◦ ***ClassNotFoundException*** ◦ ***InterruptedException*** ◦ ***InstantiationException*** ◦ ***IOException*** ◦ ***SQLException*** ◦ ***IllegalAccessException*** ◦ ***FileNotFoundException, etc*** ***NOTE:*** ☐ *Every subclass of Error and RuntimeException is an unchecked exception in Java. A checked exception is everything else under the Throwable class.* ☐ *All exceptions always occur at runtime only but some exceptions are detected at compile-time and some other at runtime.* ☐ *We should not handle the exception with the main method. The method which is responsible for the exception should handle the exception.*	
Unchecked Exceptions	• **Unchecked Exceptions (Runtime Exceptions)** are checked by JVM, not by java compiler. • They occur during the runtime of a program. • All exceptions under runtime exception class are called unchecked exceptions or runtime exceptions in Java. • We can write a Java program and compile it. But we cannot see the effect of unchecked exceptions and errors until we run the program. • This is because Java compiler allows us to write a Java program without handling unchecked exceptions and errors.Java compiler does not check runtime exception at compile time whether programmer handles them or not. • If a runtime exception occurs in a method and programmer does not handle it, JVM terminates the program without the execution of rest of the code. • List of Unchecked Exceptions in JavaSome important examples of runtime exceptions are given below:	@AMOL

	○ ***ArithmeticException*** ○ ***ClassCastException*** ○ ***NullPointerException*** ○ ***ArrayIndexOutOfBoundsException*** ○ ***NegativeArraySizeException***	
Java Exception keywords	<table><tr><th>Keyword</th><th>Description</th></tr><tr><td>try</td><td>The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.</td></tr><tr><td>catch</td><td>The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.</td></tr><tr><td>finally</td><td>The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.</td></tr><tr><td>throw</td><td>The "throw" keyword is used to throw an exception.</td></tr><tr><td>throws</td><td>The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.</td></tr></table>	@AMOL
Exception Handling	• It's technique that allows us to handle runtime errors in a program so that the normal flow of the program can be maintained.	@Abhay
Try-Catch Block	• The Java code that may generate an exception during the execution of program, must be placed within a try block. • We should place risky code that may generate exception inside try block. We should not keep normal code inside try block. • When there is exception , the method in which exception occurs will create object and that object will store three things : a. **Exception Name** >> Name of the exception b. **Description** >> Cause of the exception c. **Stack Trace** >> Line no. where issue is coming • Try >> We write Risky Code in Try Catch >> In Catch we writes the Handling code. • Catch is block of code that handles the exception thrown by the try block. That's why it is also known as ***exception handler block***. • A catch block that catches an exception, must be followed by try block that generates an exception.	@Abhay @AMOL

Try Catch Mechanism

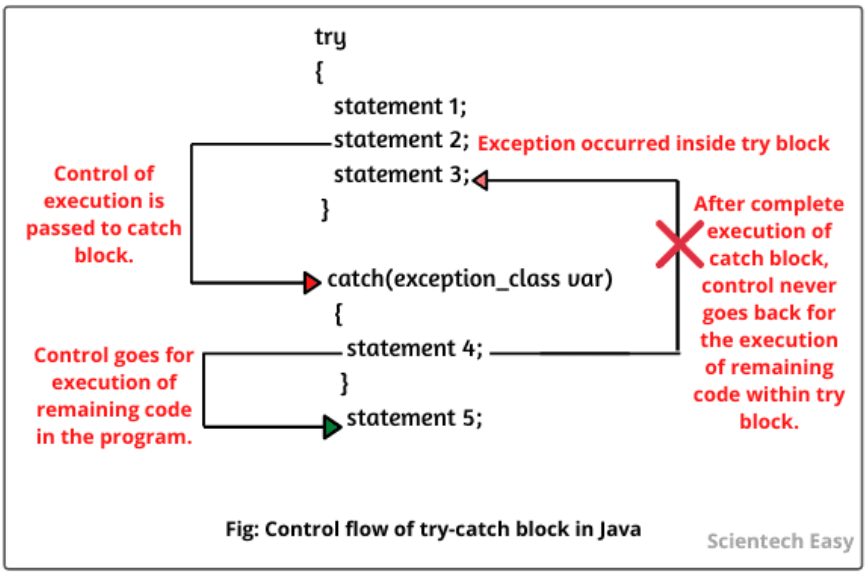


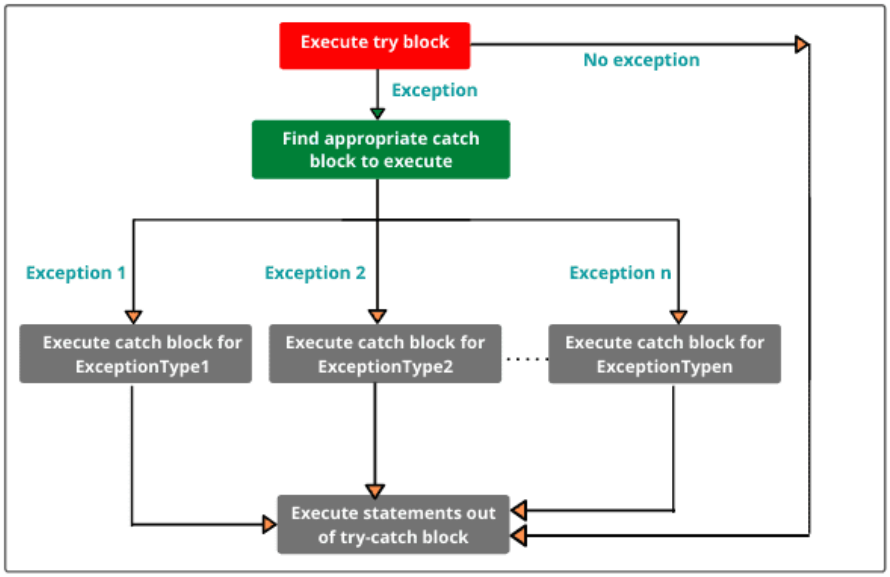
Fig: Control flow of try-catch block in Java

```
public class TryCatchEx1 {

    public static void main(String[] args) {

        System.out.println("11");
        System.out.println("Before divide");

        try {
            int x = 1 / 0;
            System.out.println("After divide");
        }

        catch (ArithmeticException e)
        // Here, e is a reference variable of exception object.
        {
            System.out.println("A number cannot be divided by zero");
        }

        System.out.println("22");
    }
}
```

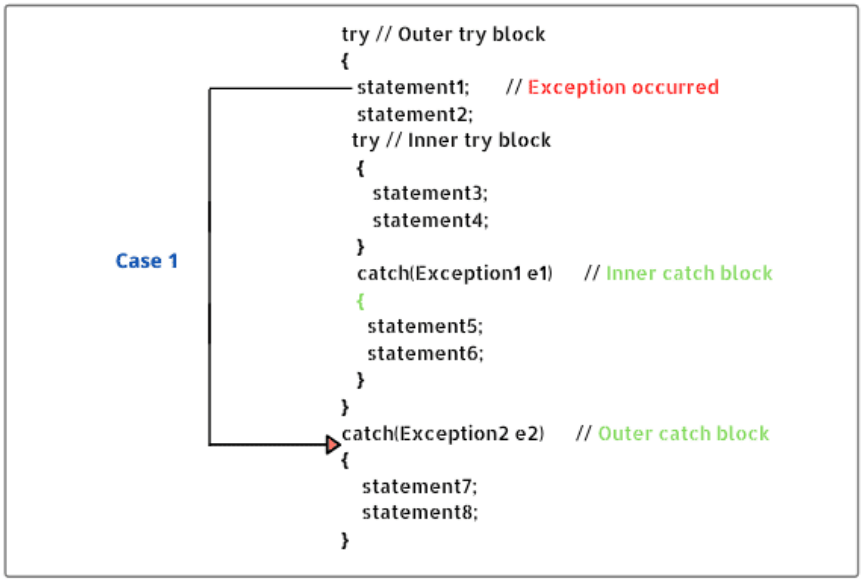
Output:

11

@Dhrumil @vibha @AMOL

	Before divide **A number cannot be divided by zero** **22**	
Multiple Catch Blocks	• A single try block can have multiple catch blocks. When statements in a single try block generate multiple exceptions, we require multiple catch blocks to handle different types of exceptions. This mechanism is called multi-catch block in java. • Each catch block is capable of catching a different exception. That is ***each catch block must contain a different exception handler.*** 	@AMOL
Nested Try Catch Java	• When a try block is defined within another try, it is called nested try block in java. • The try block which encloses another try block is called outer try block and the enclosed try block is called inner try block. ☐ **CASE1**: • If an exception occurs within outer try block, the control of execution is transferred from the outer try block to outer catch block that will handle the exception thrown by outer try block. • After handling the exception by catch block, the execution continues with the statement following the outer catch block. The inner try block and its corresponding catch blocks are skipped in this case.	@AMOL

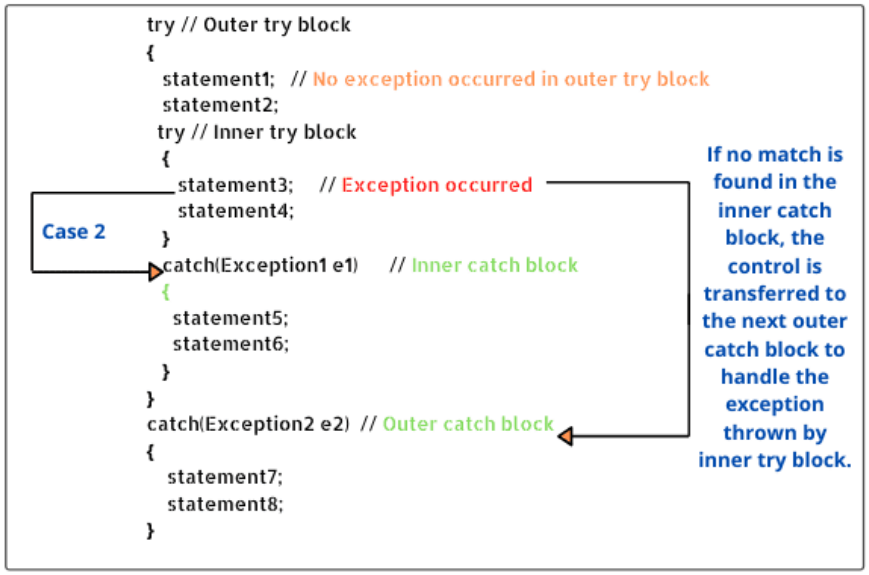
@AMOL



- [ ] **CASE 2**:

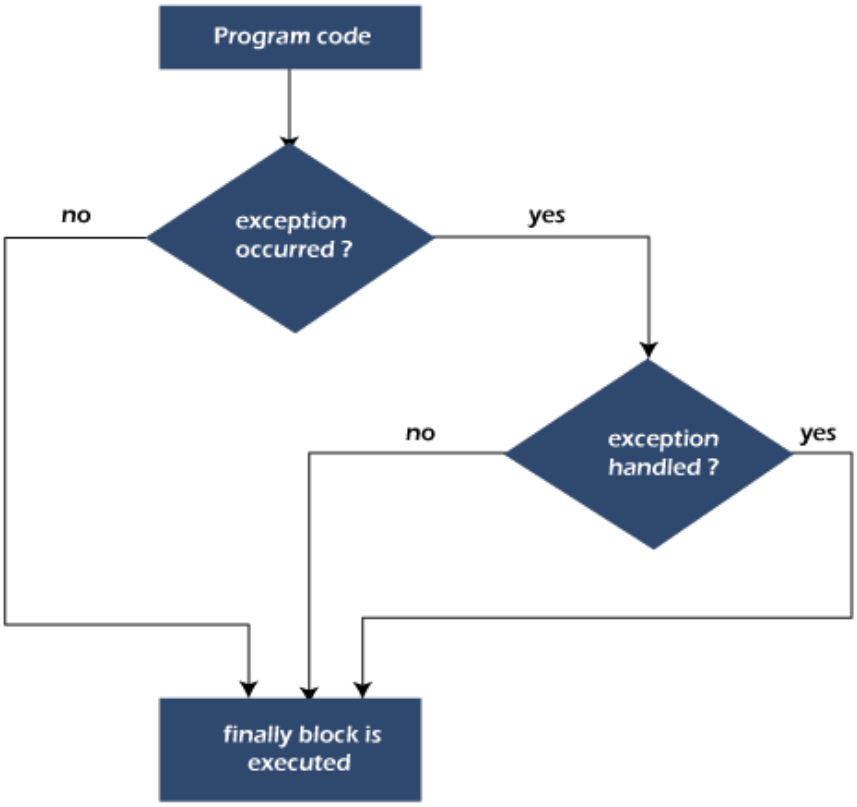
- If an exception does not occur inside outer try block, the control of execution enters into the inner try block. If an exception occurs inside inner try block, the catch block associated with this inner try block is searched for a proper match.
- If a match is found then the execution of outer catch block is skipped and the execution continues with statements following outer catch block.
- If no match is found, the control of execution is transferred to the next outer try-catch block to handle the exception of inner try block. This process continues until and unless an appropriate match is found.
- If no match is found then Java runtime system will handle the exception at runtime and the program terminates abnormally.

@AMOL



Key Points to Remember:	• An exception can be handled using try, catch, and finally blocks. • We can handle multiple exceptions using multiple catch blocks. • There can be a possibility for several exceptions inside the try block but at a time only one exception will be raised. • A single try block in Java can be followed by several catch blocks. • A catch block cannot be without try block but a try block can have without catch block. • We cannot write any statement between try and catch blocks.	@AMOL
Good Practice	• Multiple Catch Block instead of writing generic Exception way. • If there are multiple exceptions and multiple catch blocks, the first matching catch is executed and then finally block is executed. • we can have try without catch, but in this case there has to be finally block. • we can have nested try catch blocks also • We can have multiple try blocks • For each try block there can be zero or more catch blocks, but **only one** finally block. • It is not a **good programming practice to handle exceptions using Object/throwable/Exception** class • We can write specific exceptions followed by Exception class • We can use throwable class for catching Exceptions, but not Object.	@Dhrumil @vibha
Can we write try	• Yes we can use try block without catch block by using finally{} block	@AMOL

block without catch block?	• *try* *{* *statement1;* *statement2;* *}* *finally // finally block* *{* *statement3;* *}* ☐ **Rule**: For each try block there can be zero or more catch blocks, but only one finally block. ☐ **Rule:** Finally block must be defined at the end of last catch block. If finally block is defined before a catch block, the program will not compile successfully. ☐ **Rule:** Unlike catch, multiple finally blocks cannot be declared with a single try block. That is there can be only one finally block with a single try block.	
Finally Block	• Java finally block is a block used to execute important code such as closing the connection, etc. • Java finally block is always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not. • The finally block follows the try-catch block.	@AMOL



- ☐ If you don't handle the exception, before terminating the program, JVM executes finally block (if any).
- ☐ The finally block will not be executed if the program exits (either by calling ***System.exit()*** or by causing a fatal error that causes the process to abort).

Why use Java finally block?	• Generally, finally block or clause is used for freeing up resources, cleaning up code, db closing connection, io stream, etc. • The important statements to be printed can be placed in the finally block.	@AMOL

Finally block	• Ex: when System.exit(1) is encountered and executed , then finally code block is not executed • For each try block there can be zero or more catch blocks, but only one finally block for a try-catch block • The finally block is optional. It always gets executed whether an exception occurred in try block or not . • And if exception does not occur then it will be executed after the try block. • The finally block in java is used to put important codes such as clean up code e.g. closing the file or closing the connection.	@vibha @Anuradha
Conditions where finally block does not execute	• When System.exit() method is invoked before executing finally block. • When an exception happens in the finally block. • When the return statement is declared in the finally block, the control is transferred to the calling routine, and statements after return statement inside finally block will not be executed. ***NOTE:*** ☐ ***The return statement in finally block always overrides the return statements from try and catch blocks.***	@AMOL
Return statement in try block and finally block	**public class** FinallyReturn1 { **public int** m1() { **try** { System.***out***.println("I am in try block"); **return** 30; } **finally** { System.***out***.println("I am in finally block"); **return** 50; } } **public static void** main(String[] args) {	@AMOL

output	FinallyReturn1 obj = new FinallyReturn1(); System.out.println(obj.m1()); } } **I am in try block** **I am in finally block** **50**	
Return statement in catch block and finally block	public class FinallyReturn2 { public int m1() { try { System.out.println("I am in try block"); int x = 10 / 0; System.out.println("Result: " + x); } catch (ArithmeticException ae) { System.out.println("I am in catch block"); return 40; } finally { System.out.println("I am in finally block"); return 50; } } public static void main(String[] args) { FinallyReturn2 obj = new FinallyReturn2(); System.out.println(obj.m1()); } } **I am in try block** **I am in catch block** **I am in finally block** **50**	@AMOL
Return statement in	public class FinallyReturn3 { int m1() {	@AMOL

catch block and finally block but a statement after finally block	`int a = 20, b = 0;` `try {` `System.out.println("I am in try block");` `int c = a / b;` `System.out.println("Result: " + c);` `} catch (ArithmeticException ae) {` `System.out.println("I am in catch block");` `return 40;` `} finally {` `System.out.println("I am in finally block");` `return 50;` `}` `System.out.println("Statement after finally block");` `// This is unreachable code because its executing after return statement.you can't return anything after return statement` `}` `public static void main(String[] args) {` `FinallyReturn3 obj = new FinallyReturn3();` `System.out.println(obj.m1());` `}` `}`	
output	Exception in thread "main" java.lang.Error: Unresolved compilation problem: Unreachable code at FinallyReturn3.m1(FinallyReturn3.java:21) at FinallyReturn3.main(FinallyReturn3.java:26)	
Throw Keyword	• The Java throw keyword is used to throw an exception explicitly.If we want to throw an exception manually, for this, Java provides a keyword throw. • Throw in Java is a keyword that is used to throw a built-in exception or a custom exception explicitly. • Using throw keyword, we can throw either checked or unchecked exceptions in java programming.	@AMOL

☐ **Examples**:

- **throw new** *exception_name*("error message");
  - where *exception_name* is a reference to an object of Throwable class or its subclass.
  - Instances of classes other than Throwable class or its subclasses cannot be used as exception objects.
- **throw new** IOException("sorry device error");
- **throw new** NumberFormatException();

1. Only one object of exception type can be thrown by using throw keyword at a time. Throw keyword can be used inside a method or static block provided that exception handling is present.
2. Using throw keyword, we can throw either checked or unchecked exceptions in java programming.When an exception occurs in the try block, throw keyword transfers the control of execution to the caller by throwing an object of exception.

```
public class TestThrow1 {
    // function to check if person is eligible to vote or not
    public static void validate(int age) {
      if (age < 18) {
              // throw Arithmetic exception if not eligible to vote
       throw new ArithmeticException("Person is not eligible to vote");
       }
           else {
              System.out.println("Person is eligible to vote!!");
            }
    }

    // main method
    public static void main(String args[]) {
          // calling the function
          validate(13);
          System.out.println("rest of the code...");
    }
}
```

Exception in thread "main" java.lang.ArithmeticException: **Person is not eligible to vote**

Throws keyword	• Throws keyword in Java is used in the method declaration. It provides information to the caller method about exceptions being thrown and the caller method has to take the responsibility of handling the exception. • Throws keyword is used in case of checked exception only. • **access_specifie** ***return_type*** method_name(parameter_list) **throws** exception1, exception2, ... exceptionN { // body of the method. } • Which exception should be declared?Ans: **Checked exception only**, because: ◦ **unchecked exception:** under our control so we can correct our code. ◦ **error:** beyond our control. For example, we are unable to do anything if there occurs VirtualMachineError or StackOverflowError.	@AMOL
Throw Vs Throws	• Throw is used for customised exceptions. • Throws keyword is used to delegate the responsibility of exception handling to the caller (It may be a method or JVM),in which case the caller method is responsible to handle that exception. • Exception should be handled immediately in the same method where its prone to failure, which makes easy for maintenance/debugging and for checking logs and fix bugs. • JVM will not handle exceptions if the exceptions are not addressed/handled. • We should not handle exceptions using throws inside TESTNG/main method.(difficult to trace incase of issues).	@vibha @AMOL

Final vs Finally vs Finalize keyword

Sr. no.	Key	final	finally	finalize
1.	Definition	final is the keyword and access modifier which is used to apply restrictions on a class, method or variable.	finally is the block in Java Exception Handling to execute the important code whether the exception occurs or not.	finalize is the method in Java which is used to perform clean up processing just before object is garbage collected.
2.	Applicable to	Final keyword is used with the classes, methods and variables.	Finally block is always related to the try and catch block in exception handling.	finalize() method is used with the objects.
3.	Functionality	(1) Once declared, final variable becomes constant and cannot be modified. (2) final method cannot be overridden by sub class. (3) final class cannot be inherited.	(1) finally block runs the important code even if exception occurs or not. (2) finally block cleans up all the resources used in try block	finalize method performs the cleaning activities with respect to the object before its destruction.
4.	Execution	Final method is executed only when we call it.	Finally block is executed as soon as the try-catch block is executed. It's execution is not dependant on the exception.	finalize method is executed just before the object is destroyed.

@AMOL

Finalize Method

- Finalize() is the method of Object class.
- If any garbage collection activity is happening in your program before GC destroying the objects the finalize() method will be executed.
- This method is called just before an object is garbage collected.
- Finalize() method overrides to dispose system resources, perform clean-up activities and minimize memory leaks.

```
class Test{
      String name;


  public static void main(String args[]){
        Test t= new Test();
        t=null;
        System.gc();
  }

  public void finalize() {
        System.out.println("finally hello.....");   // OP
  }
}
```

@AMOL

Finalize Method IMP Points

Note:

- Object class is superclass of all class, its not require to write extends

@Gagan @Dhru

	keyword when we are using Object class reference. • we can directly override the methods of Object class. • The Java finalize() method of Object class is **a method that the Garbage Collector always calls just before the deletion/destroying the object which is eligible for Garbage Collection to perform clean-up activity**. ... This process is known as Finalization in Java. • Finalize is a method of Object class and we can overide this method in any class • GC is only called when Finalize() is overriden • The finalize method, which is present in the Object class, has an empty implementation. In our class, for clean-up activities, we have to override this method to define our clean-up activities.	mil

JavaSession_18

Topic: AccessModifiers_WrapperClass_Finally_Finalize

- **Data type conversion (by using wrapper classes)**
- **NumberFormatException example**
- **Find Max & minimum values for primitive data types by using wrapper classes**
- **Escape character example**
- **Access Modifiers (Default, Private, Protected, Public)**
- **finally** block example (Some interesting scenarios)
- **System.exit(1)** => JVM shut down.
- **finalize** Concept

Data type conversion example Very Imp while performing Selenium Automation Execution as Most of data we get from UI is in the form of String.	**String to Int:-** String s = "100"; System.out.println(s+20);//10020 int a = Integer.parseInt(s); System.out.println(a+20);//120 **String to Double** String s="100.00"; double d1=Double.parseDouble(s); System.out.println(d1+20);//120.00 **String to boolean**	@Gagan @Dhru mil

- Integer.parseInt(String)
- Double.parseDouble(String)
- Boolean.parseBoolean(String)
- String.valueOf(Int/Double/Boolean)

Note: **String to Int/Double conversion is only applicable to Numeric type of Strings, it doesn't work with Alphanumeric content.**

Example: "AA100"

```
String s="true";
boolean b=Boolean.parseBoolean(s);
if(b)
{
System.out.println("Hi");
}
else
{
System.out.println("Hello");
}
```

Int to String:-

```
int i = 200;
      System.out.println(i+10);//210
      String t = String.valueOf(i);
   System.out.println(t+10);//20010
```

Boolean to String

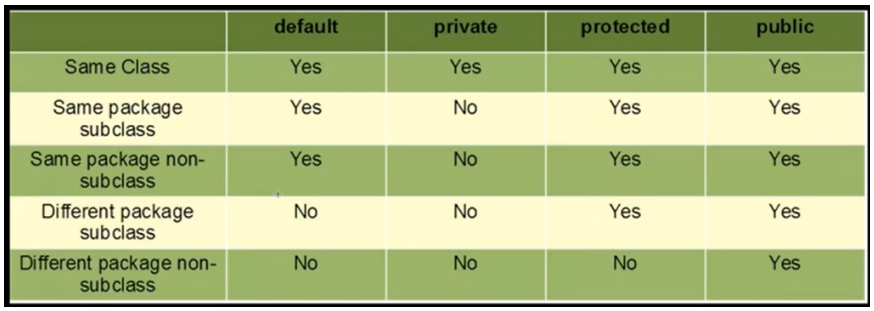
```
boolean b=true;
String b1=String.valueOf(b);
if(b1.equalsIgnoreCase("true"))
{
sopln("bye");
}
else
{
sopln("hi");
}
```

Note:

- parseInt,
- parsedouble,
- parseCharacter,
- parseByte,
- parseShort,
- parseBoolean,
- valueOf
  - all these methods are **static** methods. so we are accessing these methods with class name.

Integer,Double,Boolean,Character,Short,Long all these are

	predefined classes in java.	
Max & Minimum Values for Primitive data types.	System.out.println(Byte.MAX_VALUE);//127 System.out.println(Byte.MIN_VALUE);//-128 System.out.println(Short.MAX_VALUE);//3277 System.out.println(Short.MIN_VALUE);//-3278 System.out.println(Integer.MAX_VALUE);//2147483647 System.out.println(Integer.MIN_VALUE);//-2147483648 System.out.println(Long.MAX_VALUE);//9223372036854775807 System.out.println(Long.MIN_VALUE);//-9223372036854775808 System.out.println(Float.MAX_VALUE);//3.4028235E38 System.out.println(Float.MIN_VALUE);//1.4E-45 System.out.println(Character.MAX_VALUE);//? System.out.println(Character.MIN_VALUE);//blank space System.out.println(Double.MIN_VALUE);//4.9E-324 System.out.println(Double.MAX_VALUE);//1.7976931348623157E308	@anupama @Subhan
NumberFormatException Example:	String p = "AA100"; int r = Integer.parseInt(p);//**NumberFormatException** System.out.println(r); Here we have to extract the numeric string and then parse and use it	@Gagan
Escape character ("\")	String j = "Hi \"this\" is java"; System.out.println(j);//**Hi "this" is java** System.out.println("this is a \'test\&%'") throws "java.lang.Error: Unresolved compilation problem" Invalid escape sequence (valid ones are \b \t \n \f \r \" \' \\	@Gagan @Anuradha

Access Modifiers: Used in Java to restrict the scope of variable, method or class. **Note**: These behavior is applicable to Variables and Methods in similar manner.	 • Public --always YES • Private--always NO except from the same class • Protected--always YES except for different package and non subclass • Default --works only for same package but not from different package	@rajaSekhar @Dhrumil @Vibha
Scope of different access modifiers.	• When no access modifier is defined for any class, method or variable, by default its "default" access modifier. • Keyword "private" and private means “only visible within the enclosing class”. • Keyword "public", and Classes, methods, or data members that are declared as public are **accessible from everywhere** in the program. There is no restriction on the scope of public data members. • Keyword "protected", Anything declared as protected can be accessed by classes in the same package and subclasses in the other packages.	@Dhrumil
Interview Questions on Access Modifiers	1. least restrictive access modifier in Java => **public** 2. most restrictive access modifier in Java? => **private** 3. access modifier is also known as Universal access modifier? **=> public**	@Dhrumil
Finally block	Java finally block is **a block used to execute important code such as closing the connection**, etc. Java finally block is always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not. - **Note 1**: We can write **finally** block without catch block also. finally won't catch exception. - **Note 2:** At any case program will return only ONE value. - **Note 3**: **finally** block without **try** block is not allowed.	@Gagan @Dhrumil

	- **Note 4:** There are few cases where finally block will not execute, for example JVM is down. Application stopped abruptly...etc Ex: System.exit(1) //we won't use it - **Note 5**: If there is return statement condition satisfied and also there is finally block, then it will hold the return value and return value always executed lastly post finally block execution.	
Real Time Example of Finally block. • Close DB Connections • Close Excel files • Driver.quit() • Driver.close() • close streaming objects	`//db connection -- pass` `//pass sql string -- pass` `//try{` `//results from db -- exceptions // no excep` `//}` `catch() {` `some sql exception is coming` `}` `finally {` `//close db connection` `}` `//print the result from db` Reference: https://alvinalexander.com/blog/post/jdbc/-decent-example-of-using-try-catch-finally-with-jdbc/	@Dhrumil

JavaSession_19

Topic: SuperKeyword_HashMapConcept

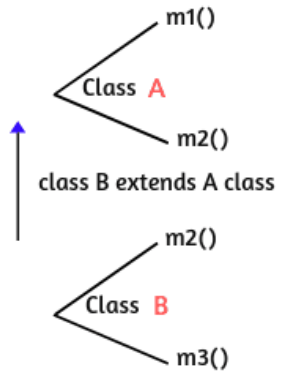
- Parent variable access
- Parent Non-static method access
- Parent Static method access
- final variables/methods access
- Constructor chaining
- Only access parent class properties, cannot access properties of Interface.
  - Interface property can be accessed through normal way i.e interfaceName.variableName

- ***HashMapConcept***
  - <key,value> --> pair/segment
  - Orderless collection - doesn't maintain index
  - put and get concept through HashMap.
  - Special case:
    - Can assign Null as key also.
    - Can we have two Null as key?

- Yes, but will get latest value as output for get method.
- Can we have Null as value? ==> Yes
- Can we have multiple Null as value? ==> Yes
- Can we have duplicate Keys? ==> Yes, but when we access through Key, get latest value i.e override.
- Can we have null as Key and Value? ==> Yes

- Note: Make sure to use wrapper class inside the HashMap declaration.
- HashMap Internal Logic: Initial Virtual Capacity is **16**
- First the hash code is calculated and then the index is calculated.
- At each index calculated to the same value - a new segment or node is added , each segment or node holds the
- node, hash code, value and the next node details. Search in the linked list is slower and O(n) .
- Binary tree logic introduced post java 1.8 to enhance the processing time to O(Log N) , after collision of index values 8 times - the linked list is converted to a binary tree
- Binary tree - lesser than equal value stored to the LHS and the higher value is stored to the RHS , making the search faster

Super Keyword	• Super keyword in Java is a reference variable that refers to an immediate super/parent class object. • The keyword 'super' comes into the picture with the concept of inheritance in Java. • Java super keyword always represents a superclass object. Whenever we create an object of subclass, an object of superclass is created implicitly, which is referred by super reference variable.	@AMOL
Uses of Super Keyword	• We can use Java super keyword in three ways: i. We can use super to call the immediate parent class's instance variable. ii. To call immediate parent class constructor. iii. To invoke the immediate superclass method. • We can apply super keyword with variables, methods, constructors of parent class. Hence, we can call immediate data members or member functions of the parent class.	@AMOL
Super keyword Demonstration	`public class Person` `{` `int age = 50;`	

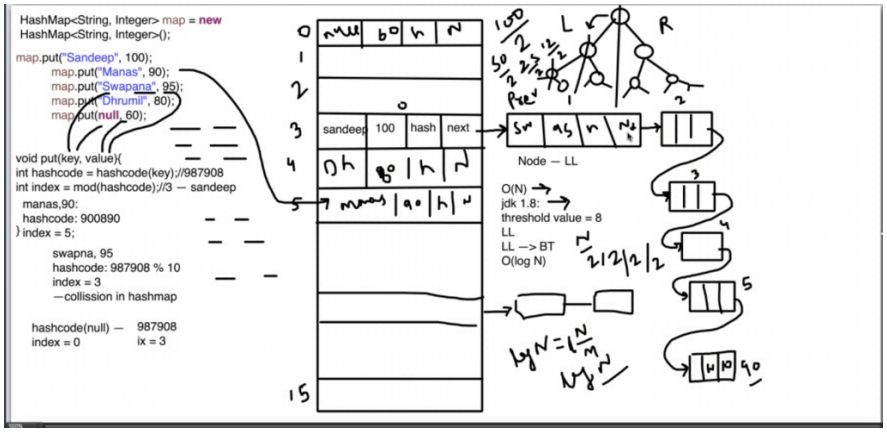
	} **public class** Employee **extends** Person { **int** age = 30; **void** insertStudentAge() { **int** age = 20; // Here, we have two ways to call instance variable 'age' of the person. // 1st way: Person p = **new** Person(); System.***out***.println(p.age); // 50 // 2nd way: System.***out***.println(**super**.age); // 50 // Calling Local variable. System.***out***.println(age); // 20 // Calling instance variable of the same class. System.***out***.println(**this**.age); // 30 } }	
Why to use Super keyword ????	 Scientech Easy There are two ways to call m2() method of superclass A. A a=new A(); // 1st way a.m2(); super.m2(); // 2nd way 1st way: In this case, we bring all the members of class A into RAM for just calling m2() method. So, unnecessary memory will be allocated for other member. 2nd way: In this way, we will just call m2() method using super keyword without wasting of unnecessary memory of RAM. Fig: Ways to call superclass variable or method	@AMOL
How to Call Superclass Instance variable	**public class** SuperDemo { // Declare an instance variable and initialize value of the variable.	@AMOL

using Super keyword?	`int x = 100;` `}` `public class Sub extends SuperDemo {` `// Declare an instance variable with the same name as provided the name of an instance variable in the superclass.` `int x = 200;` `void display() {` `// Call superclass variable x. But, it will call variable x of class Sub because of the same name.` `System.out.println("Value of variable of Sub: " + x);` `// Here, we have created an object of class Sub.` `// Therefore, it will print the value of the variable of the class Sub.` `// To call superclass instance variable, we will use the super keyword as a reference variable.` `System.out.println("Value of variable of SuperDemo: " + super.x);` `// x of class SuperDemo will call.` `}` `public static void main(String[] args) {` `Sub s = new Sub();` `s.display();` `}` `}1` **Value of variable of Sub: 200** **Value of variable of SuperDemo: 100**	
OUTPUT		
How to Call Super Constructor in Java	• A constructor creates an instance of a class. The constructor of superclass does not inherit into the subclass. Therefore, it can only be called from the constructor of subclass using the keyword super. 1. `public class A {`	@AMOL

```
    public A() { // Default constructor put by JVM at runtime.
        // invisible super(); present here.
    }
}

2.  public class X {
    public X() {
        System.out.println("Hello Java");
        super(); // Error because super must be at first line of
constructor.
    }
  }

3.   void msg() {
       super(); // Error as this is method where we tried to add
super();
    }

4.    X(int a) {
        super(); // No error.
        super(10); /// Error.
    }
}
```

- In the above syntax, the statement ***super()*** calls no-argument constructor of its superclass.
- The statement ***super(arguments)*** calls parent class constructor that matches arguments. In other words, the constructor of subclass passes arguments to superclass constructor using the super keyword.
- ***The statement super() or super(arguments) must be the first line of child class constructor***. Calling a parent class constructor's name in the child class causes syntax error.
- A constructor allows to create an object or instance of the class. Unlike the properties and methods, *constructors of the parent class do not inherit in a child class.They can only be called from the constructor of child class using the keyword super.*
- When we create an instance of any class, implicitly, the constructor of the same object gets called. But internally, constructor calls superclass constructor with the help of super keyword. This process refers to ***constructor chaining in Java.***

How to Call Superclass Method in Java	• We can also use the reserved word “super” to reference a method besides the constructor in the superclass. If a method of the subclass overrides one method of its superclass, the overridden method can be called through the use of a ‘super’ keyword. • In other words, the super should use in the case of method overriding. `class Animal {` `  void eat() {` `  System.out.println("eating...");` `  }` `}` `class Dog extends Animal {` `  void eat() {` `  System.out.println("eating bread...");` `  }` `  void bark() {` `  System.out.println("barking...");` `  }` `  void work() {` `  super.eat();` `  bark();` `  }` `}` `class TestSuper2 {` `  public static void main(String args[]) {` `  Dog d = new Dog();` `  d.work();` `  }` `}` **eating...** **barking...** OUTPUT	@AMOL
This Keyword		

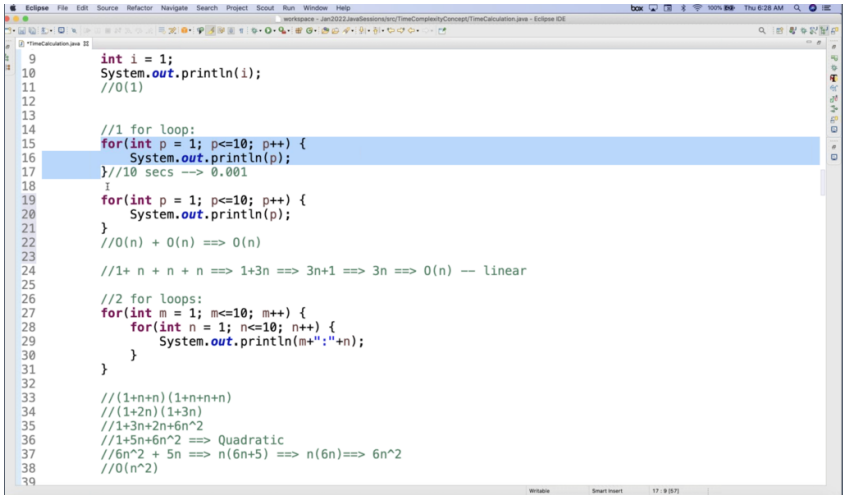
This vs Super Keywords	<table><tr><th>this</th><th>super</th></tr><tr><td>The current instance of the class is represented by this keyword.</td><td>The current instance of the parent class is represented by the super keyword.</td></tr><tr><td>In order to call the default constructor of the current class, we can use this keyword.</td><td>In order to call the default constructor of the parent class, we can use the super keyword.</td></tr><tr><td>It can be referred to from a static context. It means it can be invoked from the static context.</td><td>It can't be referred to from a static context. It means it cannot be invoked from a static context.</td></tr><tr><td>We can use it to access only the current class data members and member functions.</td><td>We can use it to access the data members and member functions of the parent class.</td></tr></table> Difference Between this() and super() Constructor <table><tr><th>this()</th><th>super()</th></tr><tr><td>The this() constructor refers to the current class object.</td><td>The super() constructor refers immediate parent class object.</td></tr><tr><td>It is used for invoking the current class method.</td><td>It is used for invoking parent class methods.</td></tr><tr><td>It can be used anywhere in the parameterized constructor.</td><td>It is always the first line in the child class constructor.</td></tr><tr><td>It is used for invoking a super-class version of an overridden method.</td><td>It is used for invoking a super-class version of an overridden method.</td></tr></table>	@AMOL
HashMap Concept		
HashMap diagram		
Hashmap use cases -	user - RBAC role based access control //customer , admin, seller, partner, vendor, distributor, same login , but user experience will be different	@Anuradha

JavaSession_20

Topic: HashMap_Iteration_Examples_TimeComplexity

HasMapIteration using foreach loop
Time Complexity Calculation for single & nested for loops
Time Complexity Calculation for single statement
Time Complexity Calculation for individual for loop
Map is parent interface for HashMap class

Time Complexity Calculation	• for (int i =1;i<=10;i++){System.out.println("test " + i)} -> time complexity will be O(n) as each step will be 1+n+n+n = 1+3n = 0(n) • for nested loop eg - for within a for - it will be a quadratic equation - O(n to power 2) -> O(n^2) eg	@Anuradha



```
int i = 1;
System.out.println(i);
//O(1)


//1 for loop:
for(int p = 1; p<=10; p++) {
    System.out.println(p);
}//10 secs --> 0.001
 I
for(int p = 1; p<=10; p++) {
    System.out.println(p);
}
//O(n) + O(n) ==> O(n)

//1+ n + n + n ==> 1+3n ==> 3n+1 ==> 3n ==> O(n) -- linear

//2 for loops:
for(int m = 1; m<=10; m++) {
    for(int n = 1; n<=10; n++) {
        System.out.println(m+":"+n);
    }
}

//(1+n+n)(1+n+n+n)
//(1+2n)(1+3n)
//1+3n+2n+6n^2
//1+5n+6n^2 ==> Quadratic
//6n^2 + 5n ==> n(6n+5) ==> n(6n)==> 6n^2
//O(n^2)
```

for (int i =1;i<=10;i++){
for (int j=1;j<=10;j++){
System.out.println("test " + j)}
}
for binary search it will -n/2->n/4->n/8->n/16->n/m
x=n/m -> log x= log n/m
log x=log n
O(log n)

int i=10;
System.out.println(i);
time complexity is O(1)
individual for loops
for (int i=1;i<=10;i++)
{
System.out.println(i);
}

for (int j=1;j<=10;i++)
{
System.out.println(j);
}
Timecomplexity is
O(n)+O(n)=O(2n)
here 2 is constant so Timecomplexity will be O(n) only if you write 100 individual for loops in this case also Timecomplexity should be O(n).

- O(N) will be getting for array iterations/search alogorithms
- O(N^2) will be getting for 2D arrays
- O(N^3) will be getting for 3D arrays
- O(logN) will be getting for binary search tree.

Note:

- For linear equation time complexity is O(N)
- For quardatic equations time complexity is O(N^2)
- For cubic time complexity is O(N^3)
- For single statements time complexity is O(1)

HashMap iterate & Practical UseCases

class HashMapIterate
{
psvm(string[] args)
{
HashMap<String,Integer> map=new HasMap<String,Integer>();
map.put("test",100);
map.put("peter",200);
map.put("sam",300);
map.foreach(k,v->System.out.println(k+":"+v));
->declare any two varaibles inside foreach loop and nxt this pointing to System.out.println statement.inside syso write variables
(->) this symbol is called lambda
}
}

UseCases for HashMap

1. **Any ecommerce Login applictaion for multiple username & pwd**

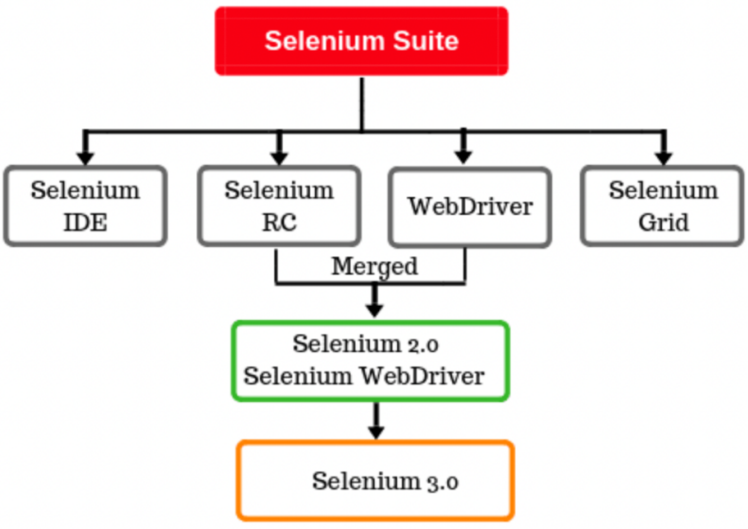
	EX: RBAC **2.** ProductMetaData	
TopCasting	Map<Interger,Interger> test=new HasMap<<Interger,Interger>();	

****
****
********** JAVA END
****
****
****

SeleniumSession -01 Date 11/02/22

Topic : WebDriver_Intro_Basics/ Selenium Basics

Selenium	• Its **just a library** which automates the UI browsers. • It only performs user actions on browsers - click, send keys, launch app, url, capture values from URL etc. • Selenium is - ○ Test automation Tool - **NO** ○ Testing framework- **NO** ○ Library- **YES**	@Amol
Selenium Features	• Supports **multiple languages**-Java, Python, C#, Ruby, JS ,PHP, GO • Supports **multiple browsers**: FireFox, Chrome, Safari, Opera, Edge, IE • Support **multiple platforms**: MAC, Linux, Windows. • **Open source:** source code is public. You can customise src code and can add extra features to it. • **Free license**: no billing is required, it's free.	@Amol
Selenium Limitations	• **Can't automate desktop applications.** (WinApp , QTP) • **Can't automate mobile apps r**equires third party integration Apium, Test Project, WebdriverIO. • Performance testing: not recommended	@Amol

	• Security testing: NO • Testing: We can't verify/test anything it just helps us to perform certain actions on webUI . We need other libraries to test the application : testNG/ JUNIT, Chai, Mocha, Jasmine, Pytest, UnitTest, PHP UNIT, NUNIT etc. • **No test report generation**: we use third parties Allure, Extent, TestNG report. • No logs generation: It generates Only UI logs, For custom logs we need (Log4j) • **No hardware testing**: like bluetooth, wifi etc. • **NO API testing:** we use third party tools like postman,REST Assured for UI+API. • **No support for image, captcha and code scanning** • Cant upload file using selenium.	
Note	• *Never compare Selenium with TestNG : TestNG helps in validation and report generation ,you can't automate user actions with testNG.*	@Amol
Selenium Components	• *Selenium is a suite of tools which consists of various components - IDE, RC, WebDriver, GRID.*  **Fig: Components of Selenium** ◦ **Latest version of Selenium is 4.0(W3C complaint)** ◦ **selenium3.0 used JSON wire protocol.**	@Amol
Selenium IDE	• *Selenium IDE* is an extension available for both Firefox and Chrome, which has the record and replay functionality	@Amol

	available.	
Selenium RC *(Selenium 1.0)*	• Selenium RC is a *server* that acts as a middle man between the user and the browser that needs to interact. • RC uses *Javascript* to work with browsers while allowing the users to write code in the language of their choice.	@Amol
Selenium WebDriver *(Selenium 2.0)*	• Selenium RC + WebDriver • Selenium WebDriver is the most commonly used component of Selenium. WebDriver allows users to write custom code in their language of choice and interact with the browser of their choice, through browser-specific drivers. • WebDriver works on the OS level and uses a Protocol called JSONWireProtocol(*Now W3C)* to communicate with browsers.	@Amol
Selenium GRID: *Infrastructure*	• Selenium GRID allows users to run tests on different machines, with different browsers and OS simultaneously, which gives the ability to run tests in parallel, as such saving a lot of time and resources of testing on several machines. • Custom Grids: Selenoid, Zalenium, Sauce Labs, DOCKERIZED GRID • GRID vendors: BrowserStack, SauceLabs ,LambdaTest • **Selenium Grid does not performs Parallel testing, parallel testing is the feature of TestNG**	@Amol @Simran
Configuration of Selenium Jar inside pom.xml Reference: https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java/4.1.2	<dependencies> <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java --> <dependency> <groupId>org.seleniumhq.selenium</groupId> <artifactId>selenium-java</artifactId> <version>4.1.2</version> </dependency> </dependencies> Note: All the configurable Jars can be downloaded from the Maven Central Repository.	@Dhrumil
Configuration of Java compiler inside	<properties> <project.build.sourceEncoding>UTF-	@Dhrumil

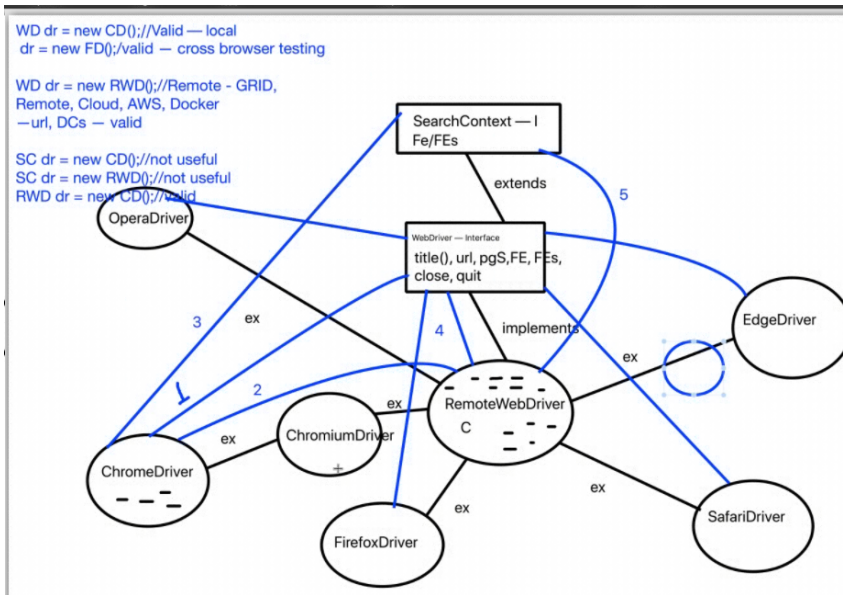
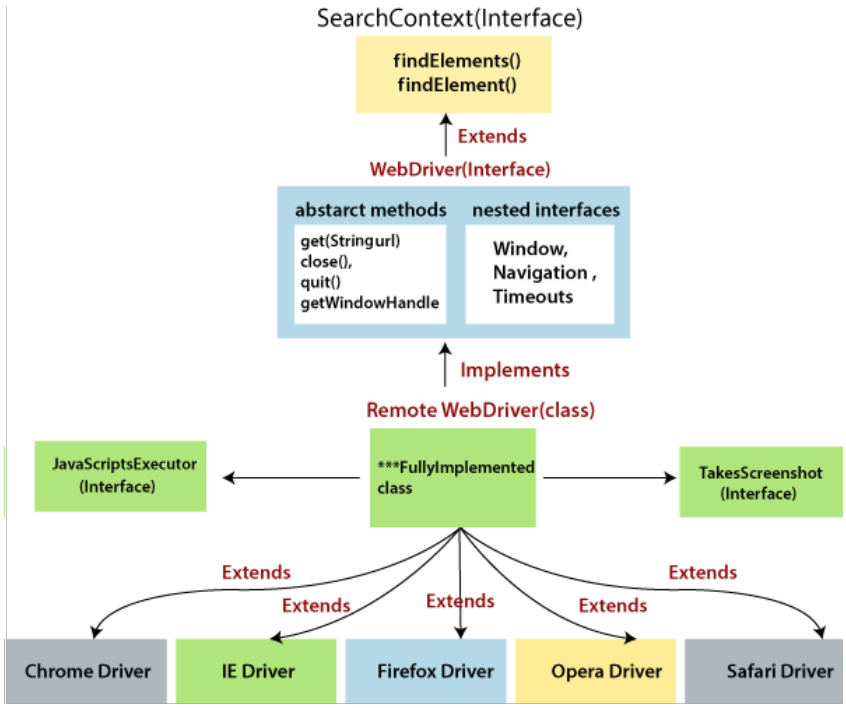
pom.xml Note: Make sure to use 1.8 version and if its not reflect inside the Maven project folder structure then update Maven project.	8</project.build.sourceEncoding> <maven.compiler.source>1.8</maven.compiler.source> <maven.compiler.target>1.8</maven.compiler.target> </properties>	
Communication flow between Test Script and Browser through Selenium Server.	 • To automate tests for a specific browser, we need to download the browser-specific drivers as browsers do not have built-in servers for the test automation. • Direct interaction with client to Browser is not possible. • Every request from client to browser is done through server(selenium server). • Server is based on the type of browser. • These drivers/servers act as a bridge between test scripts and browsers for test automation in order for the client to communicate with browser. • Once the driver is downloaded for a specific browser, the setProperty() method needs to be defined specifying the path for that driver so that test script can communicate with the browser • This helps the Selenium test script identify the browser on which tests are to be executed. • **System.setProperty("webdriver.chrome.driver", "C:\chromedriver.exe");**	@vibha @Dhrumil @Amol

	Chrome browser: download chromdriver.exe from the https://chromedriver.chromium.org/downloads Firefox browser: download geckodriver.exe from the https://github.com/mozilla/geckodriver/releases	

SeleniumSession -02 Date 14/02/22

Topic : WebDriver_TopCastingOptions

Protocol is imp while launching URL	• ***driver.get("www.google.com")*** This will not work as we need to provide http(s) protocol with the url to launch it. It will give : **org.openqa.selenium.InvalidArgumentException: invalid argument**	@Dhrumil
driver.getTitle() getCurrentUrl() getPageSource()	• **getTitle()** is used to fetch/capture the current webpage title. It returns a String. Syntax: String title=driver.getTitle(); Systm.out.println(title); **getCurrentUrl ()** is used to get the current url of the page and return type is String. Syntax: String url=driver.getCurrentUrl(); Systm.out.println(url); **getPageSource()** It gives complete source code of the page. Return type of this method is String. Syntax: Systm.out.println(driver.getPageSource());	@Manas @anupam a

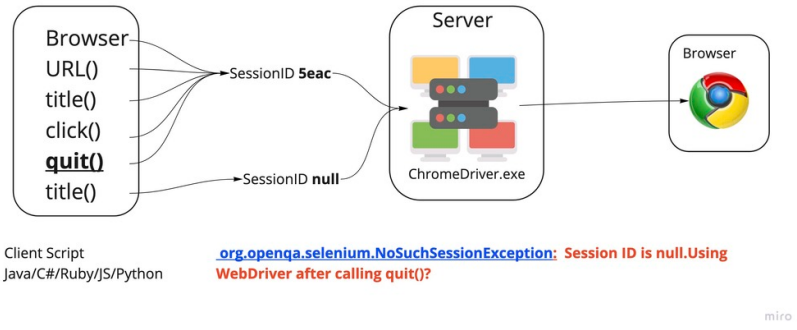
Diagramatic representation of different Top Casting options available for WebDriver		@Anupama
Github link for Selenium code:	https://github.com/SeleniumHQ/selenium/blob/trunk/java/src/org/openqa/selenium/WebDriver.java	@Anuradha
	 ***WebDriver* driver= new *ChromeDriver*(); ----------> TOP-CASTING** • Here WebDriver is an interface and ChromeDriver is a class. • ***ChromeDriver*** class object is referred by the parent ***WebDriver*** interface reference variable named here ***driver***.	@Amol

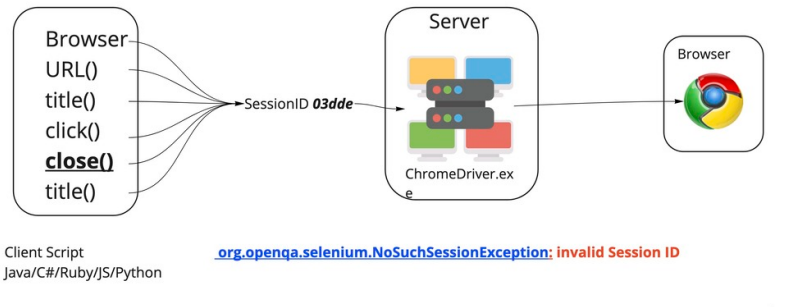
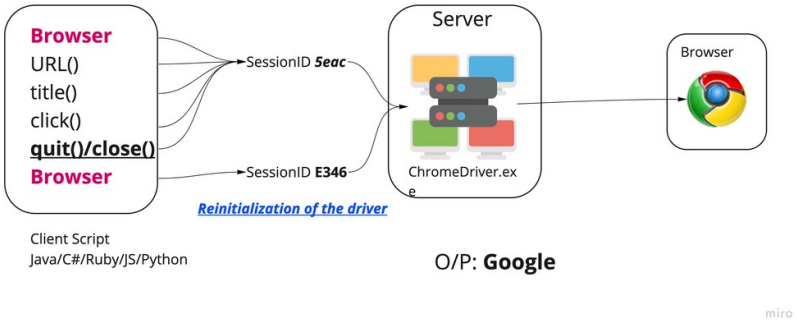
	• Its ***RemoteWebdriver*** (fully implemented) class which is implementing all the abstract (unimplemented) methods of the WebDriver interface. ---------- -- ***ALL POSSIBLE TOP-CASTINGS :*** ☐ WebDriver driver= **new** ChromeDriver(); driver= **new** SafariDriver(); ◦ This can be used for Cross Browser testing. ☐ ChromeDriver driver = **new** ChromeDriver(); ◦ In case if we need only one browser for testing. ☐ WebDriver driver = **new** RemoteWebDriver(); ◦ Remote execution using SeleniumGRID on remote machine :cloud ,AWS, Docker ☐ RemoteWebDriver driver= **new** ChromeDriver(); ◦ Valid, but Java recommends to use top casting with parent interface instead of using class. ◦ Class to Class top casting ☐ SearchContext driver= **new** ChromeDriver(); ◦ Valid but not useful, you will only get findElement() and findElements() methods. ☐ SearchContext driver= **new** RemoteWebDriver(); ◦ Valid but not useful, you will only get findElement() and findElements() methods. All this can be verified by checking selenium source code: https://github.com/SeleniumHQ/selenium/tree/trunk/java/src/org/openqa/selenium	@Amol @Dhrumil
Interview Question : Can we write *WebDriver* driver =	No, as WebDriver is an interface and we cannot create object of an interface.	@Dhrumil

new *WebDriver*()?		
Interview Question: What do you mean by below line: **WebDriver driver = new ChromeDriver();**	• webdriver is an interface. • ChromeDriver is a class. • ***ChromeDriver*** class object is referred by the parent ***WebDriver*** interface reference variable named here ***driver***. • ChromeDriver class is implementing all the methods defined inside the webdriver interface.	@Dhrumil
Interview Question: Is this valid? ChromiumDriver driver = new ChromiumDriver();	From diagram its valid, but when you check actual implementation inside ChromiumDriver class, https://github.com/SeleniumHQ/selenium/blob/trunk/java/src/org/openqa/selenium/chromium/ChromiumDriver.java Constructor is defined as Protected, and thats why we can't create object for the same.	@Dhrumil

SeleniumSession -03 Date 15/02/22

Topic : Quit_vs_Close_BrowserUtils

quit() method		@AMOL

close() method		@AMOL
Reinitialisation of WebDriver		@AMOL
	Safari driver does not have a exe , allow remote automation to be enabled in develop in order to launch safari browser	@Anuradha
Key & value for different browsers	System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver"); System.setProperty("webdriver.gecko.driver", "/path/to/geckodriver"); System.setProperty("webdriver.edge.driver", "/path/to/msedgedriver"); System.setProperty("webdriver.opera.driver", "/path/to/operadriver"); System.setProperty("webdriver.ie.driver", "C:/path/to/IEDriverServer.exe");	@anupama

SeleniumSession -04 Date 16/02/22

Topic : NavigationMethods_BackForward_WebDriverManager

Traditional Way vs WebDriverManager	**Traditional way:** Download the binaries for Chrome, Firefox browsers as per your current browser version and store locally and provide path of executable binary files inside your test scripts.	@Dhrumil

	This way is bit cumbersome as every time when you update your browser, you have to download respective and corresponding binary files.- Manual, tedious and time consuming task. **WebDriverManager:** WebDriverManager is an open-source Java library that carries out the management (i.e., download, setup, and maintenance) of the drivers required by Selenium WebDriver (e.g., chromedriver, geckodriver, msedgedriver, etc.) in a fully automated manner. It • automates the management of WebDriver binaries. • downloads the appropriate driver binaries and stored inside local cache(.m2 folder of maven local repo), if its not present already. • downloads the latest version of browser binary. **Advantage:** • Eliminates the need of storage of browser binary of different versions for different browsers. • No need to set and write "System.set.properties("webdriver.chrome.driver", "path of exe") inside test scripts.	
Github link for WebDriverManager	https://github.com/bonigarcia/webdrivermanager	@Dhrumil
Documentation of WebDriverManager	https://bonigarcia.dev/webdrivermanager/	@Dhrumil
WebDriverManager Dependency to add inside pom.xml **Note:** Remove <scope>test</scope> to use it globally across the project.	<dependency> <groupId>io.github.bonigarcia</groupId> <artifactId>webdrivermanager</artifactId> <version>5.0.3</version> <scope>test</scope> </dependency> Reference: https://bonigarcia.dev/webdrivermanager/#setup	@Dhrumil
If came across ERROR with latest	If you are getting ERROR, then kindly check this thread: https://github.com/bonigarcia/webdrivermanager/issues/576	@Dhrumil

version: ERROR: Exception in thread "main" java.lang.NoSuchMethodError: 'com.google.common.collect.ImmutableMap	Temporary solution: Check and use 4.4.3 version and build project again. Refer: https://stackoverflow.com/questions/71095560/java-lang-nosuchmethoderror-com-google-common-collect-immutablemap-error-when	
WebDriverManager provides a set of *managers* for Chrome, Firefox, Edge, Opera, Chromium, and Internet Explorer.	• WebDriverManager.chromedriver().setup(); • WebDriverManager.firefoxdriver().setup(); • WebDriverManager.edgedriver().setup(); • WebDriverManager.operadriver().setup(); • WebDriverManager.chromiumdriver().setup() • WebDriverManager.iedriver().setup();	@Dhrumil
How to instantiate a specific browser version?	WebDriverManager.chromedriver().browserVersion("85.0.4183.38").setup();	@Dhrumil
How to instantiate a platform version(x32 or x64)?	• WebDriverManager.chromedriver().arch32().setup() • WebDriverManager.chromedriver().arch64().setup()	@Dhrumil
driver.get() vs driver.navigate().to(url) ?	• driver.get() used to launch a particular website, we can't use forward and backward button. • driver.navigate().to() is also used to launch the particular website by passing the URL but we can use forward and backward button to navigate between the pages during test case writing. both are exactly same, both are synonyms of each other. Both are maintained history of the page.Only difference is syntax. driver.get("https://www.google.com"); driver.navigate().to()-->to() is overloaded method 1.driver.navigate().to("https://www.google.com")-->used to launch the url 2.driver.navigate().to(new URL("https://www.google.com"))-->used to launch the url 3. to() internally calling get() only	@Dhrumil

	4. If you are working in xyz application & suddenly you want to move to some third party application inthis case use navigate().to() method **Note: driver.get() and driver.navigate.to() both are same, not driver.get() and driver.navigate().**	
driver.navigate().forward() driver.navigate().back()	• This method enables the web browser to click on the forward button in the existing browser window, it takes you forward by one page on the browser's history. It neither accepts nor returns anything. • This method enables the web browser to click on the **back** button in the existing browser window. it takes you backwards by one page on the browser's history. It neither accepts nor returns anything.	@Dhrumil
driver.navigate().refresh()	• This method refreshes the current page. It neither accepts nor returns anything.	@Dhrumil
driver.navigate().to(new URL("www.google.com")	• This method launch www.google.com in the existing browser window. It internally call the get () method to load the google page. Both navigate().to() and get() are synonyms of each other and exactly does the same work.	@Prabha
Interview Question: 1. What are the various options are available in navigate	1. to() 2. forward() 3. back() 4. refresh() All above methods are abstract methods & implemented by RemoteNaviagtion class. RemoteNaviagtion class inner class of RemoteWebDriver & it is declared as private.	@anupama

SeleniumSession -05 Date 17/02/22

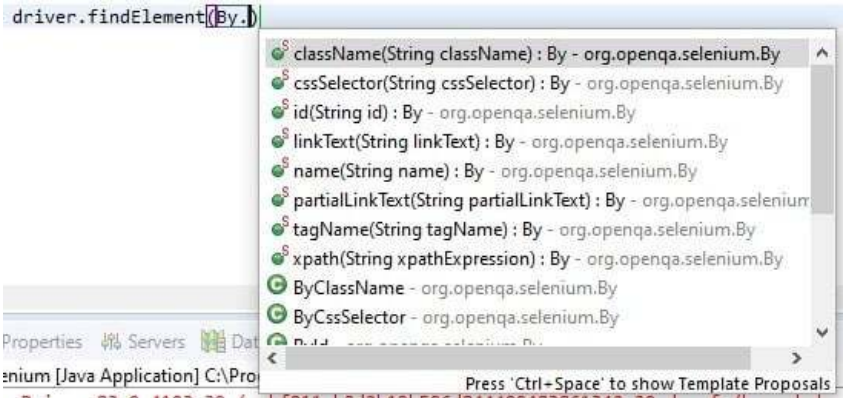
Topic : Different_Locator_Mechanism_Through_Utilities

Practice Link: https://demo.opencart.com/index.php?route=account/register

WebElement	• A WebElement is sometimes called an element. It symbolizes	@Dhrumi

	an HTML element within an HTML document. • **HTML** stands for **HyperText Markup Language** which instructs the browser how to display content. • The HTML element contains a start tag, end tag and content between both tags. • The HTML element is everything from the start tag to the end tag: *My first HTML paragraph.* • Basically whatever you see on the webpage is a WebElement. Example: NaveenAutomationLabs <input type="text" name="email" value="sdf" placeholder="E- Mail Address" id="input-email" class="form-control"> • **TagName** : Input • **Attribute Name:** type, name, value, placeholder, id, class • **Attribute values**: text, email, ,E-Mail Address, etc --- • ***DOM***: *Document Object Model* ◦ When a web page is loaded, the browser creates a Document Object Model of the page. ◦ DOM is a way to represent the webpage in a structured hierarchical way so that it will become easier for programmers and users to glide through the document. ◦ With DOM, we can easily access and manipulate tags, IDs, classes, Attributes, or Elements using commands or methods provided by the Document object.	I @AMOL
***findElement*()** method	• Most Important webElement *method ,*Because if we want to perform any action on the webElement, we have first locate/find that element. • Defined inside the *SearchContext* interface. • Overridden inside the *WebDriver* interface. • Actual implementation found inside the *RemoteWebDriver class.* • **Return type** of this method is "**WebElement**" WebElement emailID= **driver**.findElement(By.*id*("**input-email**"))	@Dhrumi l @AMOL

	○ Then after storing WebElement in emailID reference variable we have to import that WebElement. • Find WebElement + Perform various actions(Click, sendKeys, isDisplayed, getText) • WebElement can be found by using findElement() method.	
Different Locators to find/locate webElements	• To access all these locators, Selenium provides the **"*By*" class**, which helps in locating elements within the DOM. • It offers several different ***methods*** (*some of which are in the image below*) like *className, cssSelector, id, linkText, name, partialLinkText, taName, and xpath, etc.*, which can identify the web elements based on their corresponding locator strategies. • **NOTE**- *All these locator methods are static in nature* *Locator* / *Description* *id* — finds elements by ID attribute. The search value given should match the ID attribute. *name* — Finds or Locates elements based on the NAME attribute. The name attribute is used to match the search value. *class name* — Finds elements that match the class name specified. Note that compound classes are not allowed as strategy names. *tag name* — Finds or Locates elements having tag names that match the search value. *CSS selector* — Matches CSS selector to find the element. *XPath* — Matches XPath expression to the search value and based on that the element is located. *link text* — Here the visible text whose anchor elements are to be found is matched with the search value. *partial link text* — Here also we match the visible text with the search value and find the anchor value. If we are matching multiple elements, only the first entry will be selected. --- You can quickly identify all the supported locators by Selenium by browsing all the visible methods on the "*By*" class,	@AMOL @Dhrumil

		
Interview Question: ***What is By in Selenium?***	https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.html ***By*** is a abstract class in Selenium and below are the static methods defined inside the By class. • id • linkText • partialLinkText • name • tagName • xpath • className • cssSelector	@Dhrumil
Interview Question: What is return type of ***By*** in Selenium?	***By*** class	@Dhrumil
Approach # 1: Basic (Learner)	`//1st Approach: Basic` `driver.findElement(By.id("input-email")).sendKeys("dksoni@gmail.com");` `driver.findElement(By.id("input-password")).sendKeys("password");` ========== This is very initial way of writing the script when you start learning Selenium, but this would not helpful when your requirement is to locate 100 webElements on the webPage and you have to write driver.findElement() for every webElement. Your script looks clumsy and not effective.	@Dhrumil

Approach # 2: Through WebElement Creating WebElement first and perform action on it.	`//2nd Approach: Through WebElement and action later on it.` `WebElement EmailID = driver.findElement(By.id("input-email"));` `WebElement password = driver.findElement(By.id("input-password"));` `EmailID.sendKeys("dksoni@gmail.com");` `password.sendKeys("password");` ========== Locate the webElement through driver.findElement method and store inside a WebElement variable. **Note**: Return type of driver.findElement() method is **WebElement**. **Advantage**: No need to locate the webElement again and again. Just need to pass different data through the WebElement. **Disadvantage**: 100 webElements present on the page and you find it, but for specific scenario you only require 10 webElements. In that case, we still need to find better way to deal.	@Dhrumil
Approach # 3 **By Locator - Object Repository**	☐ ***By Locator Creation:*** By email= By.*id* ("input-email"); By password= By.*id* ("input-password"); • Above is not creation of object ..it takes place when you write driver.findElement() method. ◦ Here at this point we are not interacting with Selenium server ,we are not performing any action on the browser, so we are having only By class reference. --- ☐ ***WebElement Creation:*** • In order to create WebElement we have to pass the By locator to findElement() method then store it in WebElement variable. WebElement **emailID**= *driver*.findElement(email); WebElement **Password**= *driver*.findElement(password); ---	@Dhrumil @Amol

☐ ***Performing Actions on WebElements***

```
emailID.sendKeys("asd@gmail.com");
Password.sendKeys("123sdfsdf");
```

```
// 3rd Approach: By locator: Object Repository

By email = By.id("input-email");
By password = By.id("input-password");

//driver.findElement(email).sendKeys("dksoni@gmail.com");
//driver.findElement(password).sendKeys("password");

WebElement Email = driver.findElement(email);
WebElement pwd = driver.findElement(password);

Email.sendKeys("dksoni@gmail.com");
pwd.sendKeys("password");
```

Note: By.id method returns By class reference only.

Advantage: We are maintaining Object Repository of WebElement and only interacting with the desired webElements as per requirement.

Approach # 4

By using generic utility method

☐ ***By Locator Creation:***

```
By email= By.id ("input-email");
By password= By.id ("input-password");
```

☐ ***Create generic method to find WebElements***

```
public WebElement  getElement(By locator) {
    return driver.findElement(locator);
 }
```

☐ ***Calling getElement() method to perform actions on WebElements***

```
getElement (email).sendKeys("ag@gmail.com");
```

@Dhrumil @Amol

	getElement (password).sendKeys("ag@1243"); --- // 4th Approach: By generic function By email = By.id("input-email"); By password = By.id("input-password"); getElement(email).sendKeys("dksoni@gmail.com"); getElement(password).sendKeys("password"); // driver.close(); } public static WebElement getElement(By locator) { return driver.findElement(locator); } **Advantage:** • Code Reusability • Only written driver.findElement once inside the generic method. Note: Make sure to define driver reference at class level to get all the methods inside the getElement method. **static** WebDriver driver;	
Approach # 5 **By generic function(getElement + doSendKeys) in same class.**	☐ ***Create a By Locator :*** By email= By.*id*("input-email"); By password= By.*id*("input-password"); --- ☐ ***Create generic method to find WebElements*** public WebElement **getElement** (By locator) { return driver.findElement (locator); }	@Dhrumil @Amol

---

- [ ] ***Create generic method to perform actions on the WebElements***

```
public void doSendkeys(By locator, String value) {
    getElement(locator).sendKeys(value);
    }
```

- [ ] ***Performing actions on the WebElements***

```
ElementUtil eleutil= new ElementUtil(driver);
eleutil.doSendkeys (email, "anuradha@gmail.com");
eleutil.doSendkeys (password, "anu@123");
```

```
        //5th Approach: By generic function(getElement + doSendKeys)

        By email = By.id("input-email");
        By password = By.id("input-password");

        doSendKeys(email, "dksoni@gmail.com");
        doSendKeys(password, "password");
        // driver.close();
    }

    public static WebElement getElement(By locator) {
        return driver.findElement(locator);
    }

    public static void doSendKeys(By locator, String value) {
        getElement(locator).sendKeys(value);
    }
```

Approach # 6 **Move getElement and doSendKeys methods in ElementUtil class and use it across all the tests.**		@Dhrumil

```
        //6th Approach: By generic function(getElement + doSendKeys) but through ElementUtils class.

        ElementUtil el = new ElementUtil(driver);

        By email = By.id("input-email");
        By password = By.id("input-password");

        el.doSendKeys(email, "dksoni@gmail.com");
        el.doSendKeys(password, "password");

    }
```

	`` import org.openqa.selenium.By; import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; public class ElementUtil { private WebDriver driver; public ElementUtil(WebDriver driver) { this.driver = driver; } public WebElement getElement(By locator) { return driver.findElement(locator); } public void doSendKeys(By locator, String value) { getElement(locator).sendKeys(value); } } ```	
Please write detailed explanation about why we need to define non-static methods inside the ElementUtil class about getElement and doSendKeys method?	We avoid making the methods as static because if we make the methods as static then the driver should also be static . But in that case we would not be able to achieve parallel execution as the static variables and static methods are stored in Static Memory (Meta Space / Common Memory allocation). for Ex: if there are 3 threads running , then at a time only one thread will be able to make use of Static Driver and another ones will be waiting. It can only be used by another thread once it is free from previous. So never create the WebDriver as static.	@Abhay
Approach # 7 **Using Utility class inside your test class.**	``` public class LoginPageTest { public static void main(String[] args) { BrowserUtils br = new BrowserUtils(); WebDriver driver = br.launchBrowser("chrome"); br.launchUrl("https://demo.opencart.com/index.php?route=account/login"); String title = br.getPageTitle(); System.out.println(title); System.out.println(br.getPageUrl()); By email = By.id("input-email"); By password = By.id("input-password"); ElementUtil el = new ElementUtil(driver); el.doSendKeys(email, "dksoni@gmail.com"); el.doSendKeys(password, "password"); br.closeBrowser(); } ```	@Dhrumil
Approach # 8 **String Locators --> By Locator**		@Dhrumil

```
        //8th Approach - String Locator --> By Locator

        String email_id = "input-email";
        String password_id = "input-password";

        ElementUtil el = new ElementUtil(driver);
        By email = el.getBy("ID",email_id);
        By password = el.getBy("ID", password_id);

        el.doSendKeys(email, "dksoni@gmail.com");
        el.doSendKeys(password, "password");
    }
    public static By getBy(String LocatorType, String LocatorValue) {
        By locator = null;
        switch (LocatorType.toLowerCase()) {
        case "id":
            locator = By.id(LocatorValue);
            break;
        default:
            break;
        }
        return locator;
    }
```

Fill Registration Form Using BrowserUtil and ElementUtil Classes

@Amol

```
package selenium_SeSSIONS;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;

public class RegistrationPageTest {

public static void main(String[] args) throws InterruptedException {

        BrowserUtil brutil = new BrowserUtil();
        WebDriver driver= brutil.launchBrowser("chrome");
        brutil.maximize();

        brutil.launchUrl("https://demo.opencart.com/index.php?
route=account/register");
        System.out.println(brutil.getPageTitle());
        System.out.println(brutil.getPageUrl());


        By firstName= By.id("input-firstname");
        By lastName= By.id("input-lastname");
        By emailID=By.id("input-email");
        By phone=By.id("input-telephone");
        By password=By.id("input-password");
        By passwordConfirm=By.id("input-confirm");
        By agree=By.xpath("//input[@type='checkbox']");
        By button= By.xpath("//input[@type='submit']");
```

```
        ElementUtil eleutil= new ElementUtil(driver);
        eleutil.doSendkeys(firstName, "Amol");
        eleutil.doSendkeys(lastName, "Dhondge");
        eleutil.doSendkeys(emailID, "dhondgeamol@gmail.com");
        eleutil.doSendkeys(phone, "8669064399001");
        eleutil.doSendkeys(password, "B@tytbytais456");
        eleutil.doSendkeys(passwordConfirm, "B@tytbytais456");
        eleutil.doClick(agree);
        eleutil.doClick(button);


        Thread.sleep(3000);
        brutil.closeBrowser();

    }

}
```

SeleniumSession -06 Date 21/02/22

Topic : Different_Locators_Types_GetText_IsDisplayed_IsEnabled

*Selenium supports **8 different types of locators** namely id, name, className, tagName, linkText, partialLinkText, CSS selector and xpath.*

https://www.orangehrm.com/orangehrm-30-day-trial/

Registration Page and Title verification post registration.

By.id()	• An ID for a web element always **needs to be unique.** • The ID is one of the fastest and unique ways of locating web elements. • ID is one of the most reliable methods for locating an element. • But in case of dynamic webpages where "*IDs*" are generated dynamically and generally not the reliable way to locate a web element, as they change for different users.	@Amol

By.name()	• ***Can be duplicate*** .A web page can have multiple elements with the same "***name***" attribute. • If multiple elements have the same value of the '***name***' attribute, then, Selenium will just select the first values in the page which matches the search criteria. • So always make sure that the value of the ***name*** attribute should be unique when we select it for locating a web element.	@Amol
By.className()	• ***Can be same for different elements.***(not recommended, highly risky) • To identify it successfully, we need to make sure that the *class name* value that we are using for locating the web element is unique, and any other class doesn't have the same value. • If any other class contains the same value as this, then Selenium will select whichever element comes first while scrapping through the web page.	@Amol
By.xpath()	• ***It's not an attribute***, its an address for the element inside the html-dom. • It locates an element based on an XPath value. • X-path is very useful in case of dynamic pages where ID is not reliable. X-path can access any element present in the webpages even when they have dynamic properties. • *XPath* uses the *XML* expression to locate an element on the webpage.	@Amol
By.cssSelector()	• ***Its not an attribute,*** locates elements based on a CSS value. • ***CSS*** *or* ***Cascading style sheets*** are used to style webpages and hence can be used to locate various web elements.	@Amol
	NOTE ☐ *CSS selector and XPath can identify dynamic elements on a web page. A combination of different attributes and tag names can be used with CSS and xpath to identify any given element.*	
By.linkText()	• Only works for links (Anchor tags) <a> Downloads <a> • locates an element based on the value of the ***exact text*** in a	@Amol

	web page.	
By.partialLinkText()	• locates an element by using *part of the hyperlink text.*	@Amol
	NOTE ☐ ***linkText*** *and* ***partialLinkText*** *both are quite similar in functionality and locate web elements by using the hyperlink texts.* ☐ *We can only use them for elements containing* ***anchor(<a>)*** *tags.*	@Amol
By.tagName	• Its a tag not an attribute. • This locator type uses *tag names* to identify elements in a web page. The *tag name* is the *HTML* tag, such as ***input, div, anchor tag, button,* etc. • The *tagName* locator returns all the elements from the page that contains a specified tag. Example: By.tagName("a") will return all the links present on the webpage.	@Amol
Preferences: ***when web page doesn't have any dynamic Elements.***	1. id 2. name 3. className (not recommended, highly risky) • *ID and name attributes take precedence over other locators if your web page is not dynamic and contains unique ID and name, then it's always advisable to use them instead of others.*	@Amol
Preferences: ***when web page contains dynamic Elements.***	1. xpath 2. cssSelector	@Amol
getText()	Extracts the text of the webElement	@Dhrumil
isDisplayed()	• The **isDisplayed** method in Selenium verifies if a certain element is present and displayed. • This accepts nothing as a parameter but returns boolean value(*true/false*). • If the element is displayed, then the value returned is true. If not, then the value returned is false.	@Dhrumil

	○ boolean eleSelected= driver.findElement(By.xpath("xpath")).isDisplayed();	
isEnabled()	• The **isEnabled** method verifies if an element is enabled. • This accepts nothing as a parameter but returns boolean value(*true/false*). • If the element is enabled, it returns a true value. If not, it returns a false value. ○ boolean eleEnabled= driver.findElement(By.xpath("xpath")).isEnabled();	@Dhrumil
NoSuchElementException: no such element: Unable to locate element: ---------- ~~***ElementNotFoundException***~~	• It will be thrown when WebElement is not found on the webPage. • Its thrown by ***driver.findElement()*** method. **When we are using driver.findElements() method, if elements are not located then it would not give any exception, but it will give 0 elements i.e Empty list.** ---------- ☐ ***ElementNotFoundException*** is not a Selenium Exception. ☐ We can not compare this exception with NoSuchElementException.	@Gagan @Dhrumil

SeleniumSession -07 Date 22/02/22

Topic :

TotalLinks_Images_GetAttribute_findElements_usecases_SelectDropDown_basics

https://qa.max.com/shop/create-account/
https://www.orangehrm.com/orangehrm-30-day-trial/

Beginners

- https://itera-qa.azurewebsites.net/home/automation
- https://www.globalsqa.com/demo-site/

- https://testautomationpractice.blogspot.com/
- https://www.saucedemo.com/

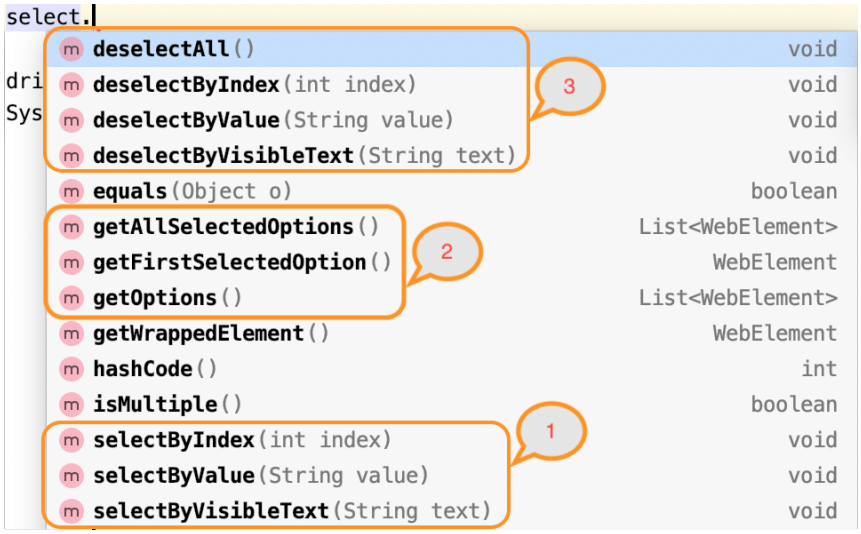
Intermediate & Advanced

- https://opensource-demo.orangehrmlive.com/
- http://demo.nopcommerce.com/
- http://admin-demo.nopcommerce.com/
- https://demo.opencart.com/
- https://demo.opencart.com/admin/
- http://automationpractice.com/
- http://live.demoguru99.com/
- https://itera-qa.azurewebsites.net/home/automation

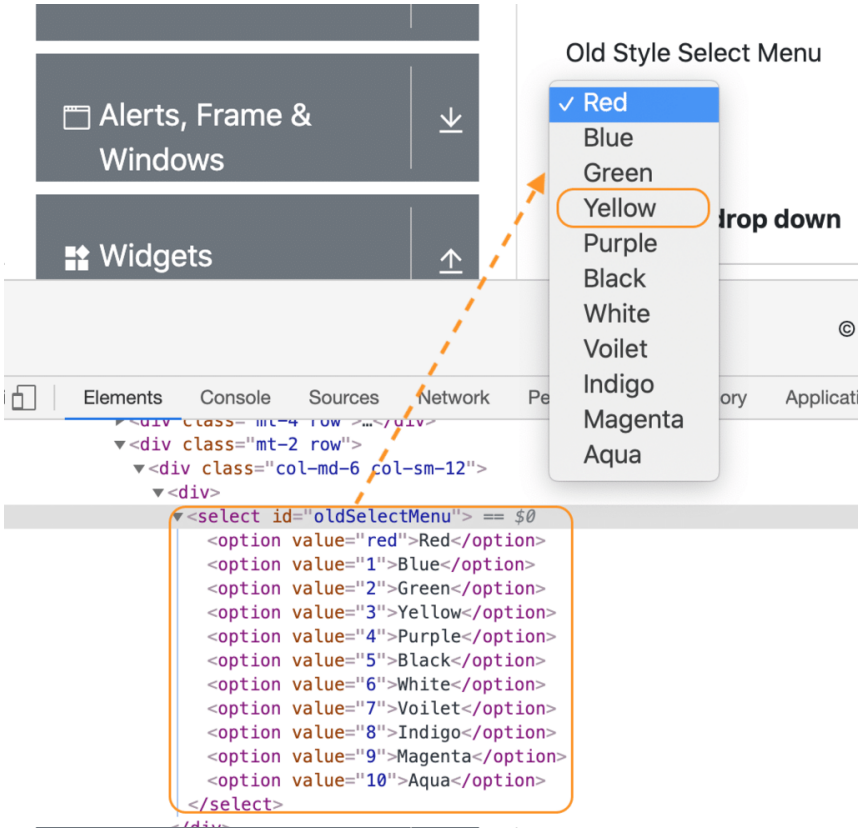
driver.findElements( )	• find a group of WebElements matching a provided criterion. **Syntax:** List<WebElement> elementName = driver.findElements(By.LocatorStrategy("LocatorValue")); **Example:** List<WebElement> listOfElements = driver.findElements(By.xpath("//div"));	@Dhrumil
Interview Question: Difference between findElement vs findElements method	Difference between findElement and findElements **findElement** — Returns the first matching web element if multiple web elements are discovered by the locator; Throws **NoSuchElementException** if the element is not found **findElements** — Returns a list of multiple matching web elements; Returns an empty list if no matching element is found	@Dhrumil
Total No of Links present on the webPage	List<WebElement> linksList= driver.findElements(By.*tagName*("a")); System.***out***.println("total links on the page:"+ linksList.size());	@Dhrumil
Get text of each link present on the page	List<WebElement> linksList= driver.findElements(By.*tagName*("a")); System.***out***.println("total links on the page:"+ linksList.size());	@Dhrumil

	// Enhanced For Loop to iterate through each link and extract text through it. for(WebElement e: linksList) { String text=e.getText(); if(!text.isEmpty()) { // condition becomes false if text is empty System.out.println(text); } --- **Alternative way:** for(int i=0;i<linksList.size();i++) { String textlink=linksList.get(i).getText(); if(!textlink.isEmpty()) { System.out.println(textlink); } }	
isEmpty()	The **Java String class isEmpty()** method checks if the input string is empty or not. Note that here empty means the number of characters contained in a string is zero.	@Dhrumil
GetAttribute()	The getAttribute() method is declared in the WebElement interface, and it returns the value of the web element's attribute as a ***string***. For attributes having boolean values, the getAttribute() method will return either true or null. **Real time example where it can be used.** *"Consider an air ticket booking application. The color of booked and available seats are different. Red represents the booked seats, and available seats are represented by green. So, for verifying whether a seat is booked or available, QAs need to fetch the attribute (color) value through the test script. Once the status of the seat is verified,*	@Dhrumil

	only then can QAs verify further test scenarios."	
NOTE	☐ In ***HTML,*** the *dropdowns* are generally implemented either using the <***select***>tag or the <***input***> tag.	@Amol
SelectClass DropDowns	• To perform certain operations on the *dropdowns,* which are declared using the <***select***> HTML tag, *Selenium WebDrivers* provides a class called *"**Select** "* class. • There are various types of *dropdowns* available : ***single-select** (which allows selecting only one value)* or ***multi-select** (which allows selecting multiple values).* • *Selenium WebDriver* provides a class named *"**Select**"*, which provides various methods to handle the *dropdowns,* be it single-select or multi-select *dropdowns.*	*@Amol*
SelectClass Object Creation	• *"Select "* class is provided by *"**org.openqa.selenium.support.ui** "*package of *Selenium WebDriver.* • You can create an object of the Select class, by-passing the object of the *"WebElement"*class, which shows the object returned by the corresponding *locator* of the *WebElement.* 	@Amol
Select class- Used for Static drop-down content selection.	• In Selenium, the Select class provides the implementation of the HTML SELECT tag. A Select tag provides the helper methods with select and deselect options. Syntax: **Select objSelect = new Select(WebElement);** • The *Select class constructor* accepts one parameter: the WebElement object returned by the locators of the specified element.	@Dhrumil @Hemanth @Amol
All the methods of SelectClass		@Amol



How to select a value from a dropdown ? • *selectByIndex()* • *selectByVisibleText()* • *selectByValue()*	☐ ***selectByIndex()*** ***selectByIndex(int arg0) : void*** User has to provide the index number for the option. It takes the integer parameter which is the index value of Select element and it returns nothing. Example: select.selectByIndex(5); --- ☐ ***selectByVisibleText()*** **selectByVisibleText:** ***selectByVisibleText(String arg0): void*** This method is used to select one of the options in a drop-down box It takes a parameter of String which is one of the values of Select element and it returns nothing. Example: select.selectByVisibleText("India"); --- ☐ ***selectByValue()*** **selectByValue:** ***selectByValue(String arg0) : void*** It takes a String parameter which is one of the values of Select element and it does not return anything. Example: // value of the "value" attribute of option tag select.selectByValue("Algeria"); ---	@Dhrumil

			@Amol

SeleniumSession -08 Date 23/02/22

Topic:

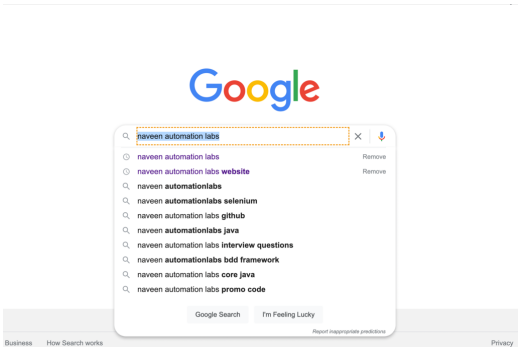
DropDown_WithSelect_WithoutSelect_GoogleSearch_SuggList

Utilities for • ***selectByIndex()*** • ***selectByVisibleText()*** • ***selectByValue()***		**public void** doSelectByVisibleText(By locator, String text) { //doSelectByVisibleText Select select=**new** Select(getElement(locator)); select.selectByVisibleText(text); } --- ---		@Dhrumil

		public void doSelectByIndex(By locator, **int** index) { //doSelectByIndex Select select=**new** Select(getElement(locator)); select.selectByIndex(index); } ---------- **public void** doSelectByValue(By locator, String value) { //doSelectByValue Select select=**new** Select(getElement(locator)); select.selectByValue(value); }		
getOptions () method		• We can extract all the options in a dropdown in Selenium with the help of Select class which has the getOptions() method. • This retrieves all the options on a Select tag and returns a list of web elements. • This method does not accept any arguments. Select select = **new** Select(*getElement*(locator)); List<WebElement> OptionsList = select.getOptions(); • Using this method, we can retrieve all the options of a *dropdown (be it single-select or multi-select).*		@Amol

How to select value from the drop-down without using • ***selectByIndex()*** • ***selectByVisibleText()*** • ***selectByValue()***		By country=By.*id*("Form_submitForm_Country"); **public static** WebElement getElement(By locator) { **return** *driver*.findElement(locator); } **public static void** doSelectValue(By locator, String value) { Select select = **new** Select(*getElement*(locator)); List<WebElement> OptionsList = select.getOptions(); **for**(WebElement e:OptionsList) { String text=e.getText(); **if**(text.equals("India")) { e.click(); **break**; } } }		@Amol
Interview Question: ***Select the value from the dropdown without using*** • ***Select class*** • ***SendKeys***		//Instead of using Select class // we will use a generic Xpath to locate all 232 options List<WebElement> CountryOptions = *driver*.findElements(By.xpath("//select[@id='Form_submitForm_Country']/option")); **for**(WebElement e:CountryOptions) { String text=e.getText();		@Amol

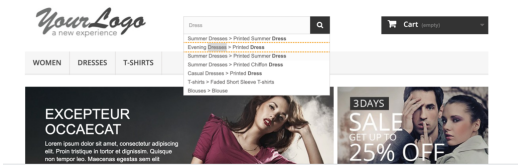
• ***Select class methods***		``` if(text.equals("India")) { e.click(); break; } } ```		
How to deselect a value from a dropdown ?		• *The Select Class also provides deselection methods to deselect certain values from a dropdown.* • Just like we select values in a *DropDown & Multi-Select,* we can deselect the values too. • But the deselect method works only for *Multi-Select.* • You can deselect pre-selected options from a *Multi-select* element using the different *deselect*methods : • Select class provides the following methods to deselect values of a *dropdown:* a. ***deselectAll()*** b. ***deselectByIndex()*** c. ***deselectByValue()*** d. ***deselectByVisibleText()***		@Amol
isMultiple()		Whether this select element supports multiple options at the same time? Return type is boolean Exp- By country=By.id("Form_submitForm_Country"); Select select=new Select(driver.findElement(country));		@Manas

		System.out.println(select.isMultiple()); // false		
getAllSelectedOptions()		It gives all the selected options belongs to this select tag Exp- By country=By.id("Form_submitForm_Country"); Select select=new Select(driver.findElement(country)); select.getAllSelectedOptions();		@Manas
Interview Question: Suggestions based drop-down handling ***Google Search: AutoSuggestion Using for loop & List<WebElement>***		 **public class** GoogleSearch_Autosuggestion { **static** WebDriver *driver*; **public static void** main(String[] args) **throws** InterruptedException { WebDriverManager.*chromedriver*().setup(); *driver*=**new** ChromeDriver(); *driver*.get("https://www.google.com/"); *driver*.findElement(By.*name*("q")).sendKeys ("naveen automation labs"); Thread.*sleep*(3000); List<WebElement>		@Amol

```
options=driver.findElements

(By.xpath("//ul[@class='G43f7e']/li"));

        for(WebElement e: options) {

            String option=e.getText();
            System.out.println(option);


if(option.contains("interview questions"))
{
                e.click();
                break;
            }

        }
        driver.close();

    }

}
```

Auto-Suggestions based drop-down handling without using *for* loop

Single Xpath- http://automationpractice.com/index.php




```
class Auto_Sugeestion_ListHandle {

static WebDriver driver;
public static void main(String[] args)
throws InterruptedException {


WebDriverManager.chromedriver().setup();
        driver= new ChromeDriver();
```

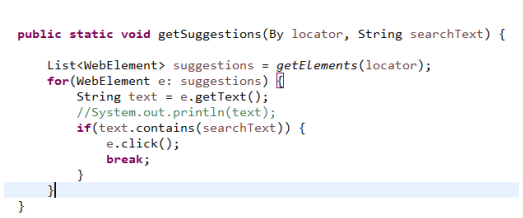
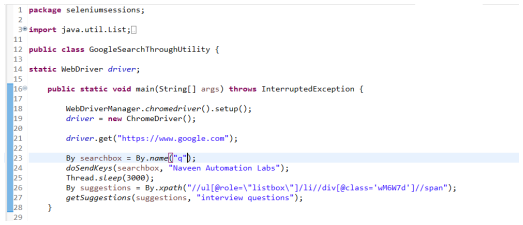
@Amol

		driver.get("http://automationpractice.com/index.php"); *driver*.findElement(By.*id*("search_query_top")). sendKeys("Dress"); Thread.*sleep*(4000); *driver*.findElement(By.*xpath*("//div[@class='ac_results']//li[contains(.,'Evening Dresses > Printed Dress')]")).click(); } }		
Assignment : Google Search using single Xpath		**public class** GoogleSearch_Autosuggestion { **static** WebDriver *driver*; **public static void** main(String[] args) **throws** InterruptedException { WebDriverManager.*chromedriver*().setup(); *driver*=**new** ChromeDriver(); *driver*.get("https://www.google.com/"); *driver*.findElement(By.*name*("q")).sendKeys("naveen automation labs"); Thread.*sleep*(3000); // Google Search using single Xpath // In below x-path "selenium" single word is enough *driver*.findElement(By.*xpath*("//ul[@class='G43f7e']//li[contains(.,'selenium')]")).click();		@Amol

Assignment: Automation Practice website- http://automationpractice.com/index.php **Using suggestion list with for Loop.**		**public class** Auto_Sugeestion_ListHandle { **static** WebDriver *driver*; **public static void** main(String[] args) **throws** InterruptedException { WebDriverManager.*chromedriver*().setup(); *driver*= **new** ChromeDriver(); *driver*.get("http://automationpractice.com/index.php"); *driver*.findElement(By.*id*("search_query_top")).sendKeys("Dress"); Thread.*sleep*(4000); List<WebElement>DressOptions= *driver*.findElements(By.*xpath*("//div.[@class='ac_results']//li")); **for**(WebElement e:DressOptions) { String txt=e.getText(); **if**(txt.contains("Chiffon")) { e.click(); **break**; } } *driver*.close(); }		@Amol

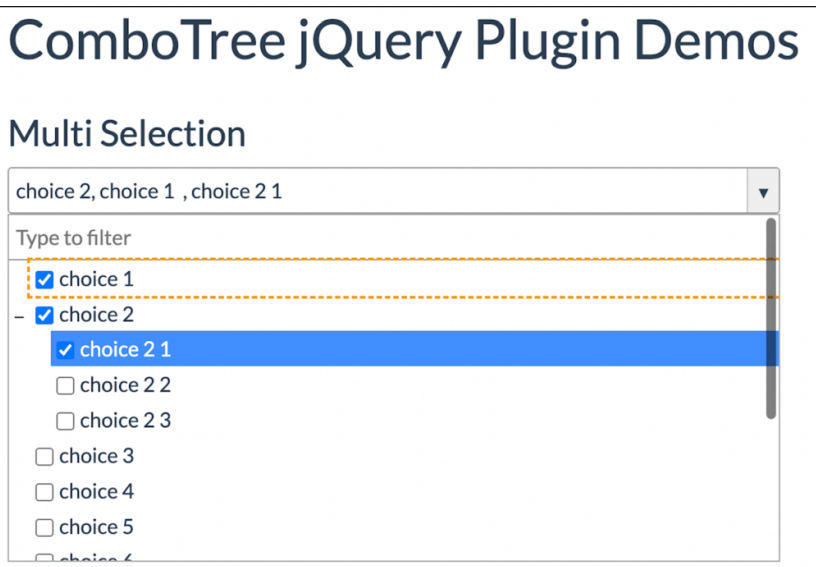
		}		
Create a Generic Utility that returns List<String > with all suggestions we will pass I/p : any search key in search box		public class GoogleSearch_Autosuggestion { static WebDriver *driver*; public static void **main**(String[] args) throws InterruptedException { WebDriverManager.*chromedriver*().setup(); *driver*=new ChromeDriver(); *driver*.get("https://www.google.com/"); By textField= By.*name*("q"); By suggestList= By.*xpath*("//ul[@role='listbox']//li"); ***doSendkeys***(textField,"naveen automation labs"); List<String> suggestionList=***getSuggestionList***(suggestList); System.***out***.println(suggestionList); *driver*.close(); } public static List<WebElement> ***getElements***(By locator){ //getElements() return *driver*.findElements(locator);		@ A m ol

```
	}

	public static WebElement getElement(By locator)
{			//getElement()
		return driver.findElement(locator);

		}
	public static void doSendkeys(By locator, String value) {
//doSendKeys()


getElement(locator).sendKeys(value);
	}

	public static List<String> getSuggestionList(By locator)
	{
//getSuggestionList
		List<WebElement> suggestOptionsList= getElements(locator);
		List<String> suggestionList= new ArrayList<String>();

		for(WebElement e: suggestOptionsList) {

			String txt=e.getText();
			System.out.println(txt);
			suggestionList.add(txt);

		}
		return suggestionList;

	}
```

Utility for the getSuggestions:				@Dhrumil
Google Search Through Utility				@Dhrumil

SeleniumSession -09 Date 24/02/22

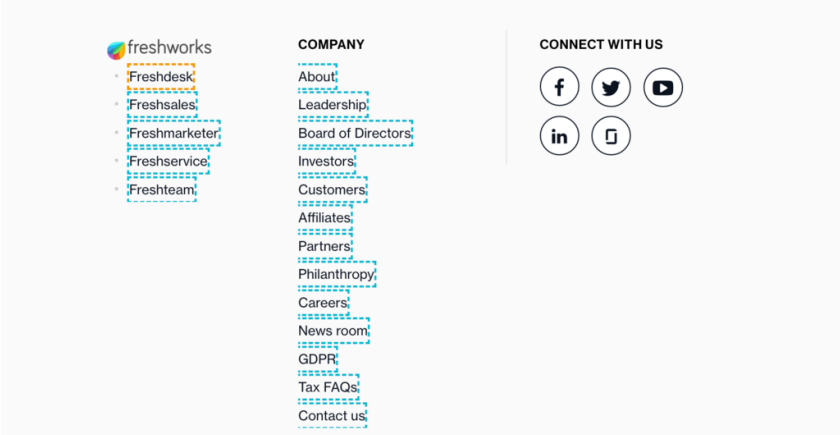
Topic: JqueryDropDownHandle_FooterList_VariousExceptions

JqueryDropDown Handle Without SelectClass *//Single Choice Selection* *// Multiple Choices selection* *//All choices Selection*	 **public class** JQueryDropDownHandle { **static** WebDriver *driver*; **public static void** main(String[] args) **throws** InterruptedException { WebDriverManager.*chromedriver*().setup();	@Amol

```
		driver=new ChromeDriver();
		driver.get("https://www.jqueryscript.net/demo/Drop-Down-Combo-Tree/");


driver.findElement(By.cssSelector("#justAnInputBox")).click();
		Thread.sleep(2000);

		By choices= By.xpath("//span[@class='comboTreeItemTitle']");


		//Single Choice Selection
		selectDropDown(choices,"choice 1");

		// Multiple Choices selection
		selectDropDown(choices,"choice 1","choice 2 1","choice 6 2 1");

		//All choices Selection
		selectDropDown(choices,"all");
	}

	public static void selectDropDown(By locator,String... value) {
	List<WebElement> choiceList=driver.findElements(locator);


	 if(!value[0].equalsIgnoreCase("all")) {
		for(WebElement e:choiceList) {
			String txt=e.getText();

			for(int i=0;i<value.length;i++) {
				if(txt.equals(value[i])) {
					e.click();
					break;
				}
			}
			}

		}
	 else {
```

```
            try {
              for(WebElement e:choiceList) {
                  e.click();
                  }
              }
              catch(ElementNotInteractableException e) {
                    System.out.println("All choices are over.....");
              }
            }  }
}
```

How to capture all the elements from FooterList?

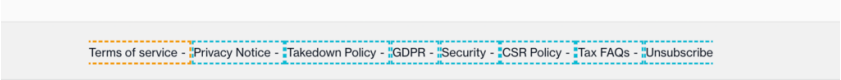


@Amol

```
public class FooterList {

      static WebDriver driver;
      public static void main(String[] args) {


            WebDriverManager.chromedriver().setup();
            driver= new ChromeDriver();
            driver.get("https://www.freshworks.com/");


            By footer1 = By.xpath("//div[@class='col-sm-6']//a");
            By footer2 = By.xpath("//div[@class='footer-nav
copyrights-nav hide-in-mobile']//a");


            getFooterList(footer1);

      }
```

```
    public static List<WebElement> getElements(By
locator){            //getElements()

        return driver.findElements(locator);
    }

    public static void getFooterList(By locator) {

        List<WebElement> footerList=getElements(locator);
        for(WebElement e : footerList) {

            String txt=e.getText();
            System.out.println(txt);
        }

    }
}
```

Assignment: ***Capture the elements from FooterList***



- In above program just change the X-path to -"//div[@class='footer-nav copyrights-nav hide-in-mobile']/a"
- By footer2 = By.*xpath*("//div[@class='footer-nav copyrights-nav hide-in-mobile']//a");
- Just call the ***getFooterList(locator)*** method in main method
  getFooterList(footer2);

@Amol

Revision: ***findElement vs findElements***	<table><tr><th></th><th>findElement</th><th>findElements</th></tr><tr><td>When multiple elements are present on webpage</td><td>Returns the first element</td><td>Returns all the elements</td></tr><tr><td>When single element is present on webpage</td><td>Returns a single element</td><td>Returns a single element</td></tr><tr><td>When no element is present on webpage</td><td>Throws an Exception i.e., NoSuchElementException</td><td>Doesn't throw any Exception, return zero elements</td></tr><tr><td>Return type</td><td>WebElement</td><td>List<WebElement></td></tr><tr><td>How to access</td><td>Can be accessed directly</td><td>Iterate the list and access each item using for/foreach</td></tr></table>	@Amol
Interview Question: *Which exception thrown with driver.findElement() and driver.findElements() method when no element located by Selenium?*	driver.findElement() ==> NoSuchElementException driver.findElements() ==> Empty list	@Dhrumil
isElementPresent() utility using findElements() method	• We can use driver.findElements() for single element. • **if**(*getElements*.size()>0) System.***out***.println("The element is present on the page"); • **public boolean** *isElementPresent*(By locator) { **if**(getElements(locator).size()>0) { **return true**; } **else** { **return false**; } }	@Amol
Interview Question:	1. Through isDisplayed() method 2. Locate the element through driver.findElements() method and	@Dhrumil

Tell me different ways to check whether element is present on the page?	capture inside the list and validate whether size is greater than 0.	
Interview Question: ***What would be output of the below code?*** *Note: Consider that provided xpath/locator is wrong.*	boolean flag = driver.findElement(By.linkText("Customers111")).isDisplayed(); System.out.println(flag); ========== **Answer: NoSuchElementException**	@Dhrumil
What if you pass the wrong X-path or any other locator syntax	Exception in thread "main" org.openqa.selenium.**InvalidSelectorException: invalid selector:** Unable to locate an element	@Amol
List Down all the exception which we have learned till the time? Update this list with necessary explanation, when and with which situation it occurs.	1. **ElementNotInteractableException-** **Element is displayed on the webpage but it is not interacting so in this case we will get ElementNotInteractableException.** **Ex:** 2. **InvalidArgumentException --->if you pass url without https/http in this case we will see InvalidArgumentException.** Ex: driver.get("www.amazon.in"); 3. **NoSuchElementException-->if you pass wrong locator then driver.findElemnt throws NoSuchElementException** 4. **InvalidSelectorException-->x-path/css selctor syntax is wrong then will get InvalidSelectorException** **5. illegalStateException-->If you miss System.setProperty then we will get illegalStateException** **6. NoAlertPresentException** **7. NoFrameException** **8. NoSuchSessionException-->Call webdriver after quiting/closing the browser**	

Anyone tried De-Selection Logic through utility creation?		

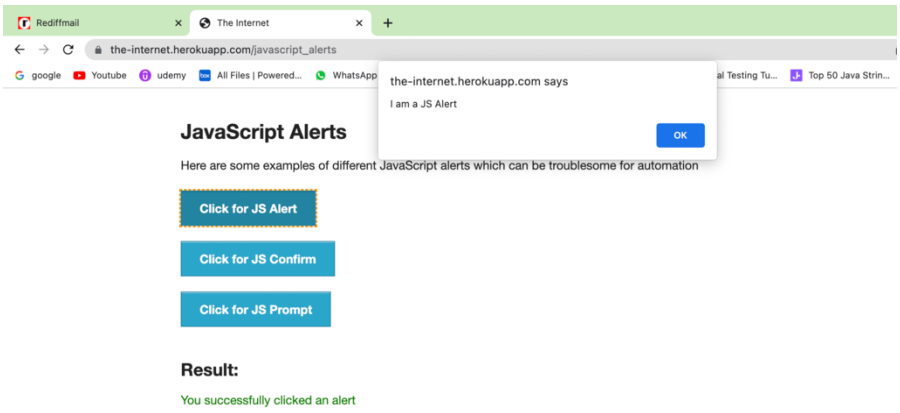
SeleniumSession -10 Date 28/02/22

Topic: Alert_PopUp_AuthPopUp_Frame_FileUpload

1. https://the-internet.herokuapp.com/javascript_alerts
2. https://the-internet.herokuapp.com (Authentication PopUps)
3. http://www.londonfreelance.org/courses/frames/index.html
4. https://mail.rediff.com/cgi-bin/login.cgi
5. https://cgi-lib.berkeley.edu/ex/fup.html -(File Upload PopUp)
6. https://www.globalsqa.com/demo-site/frames-and-windows/#iFrame-(iframe handling)

Web/Javascript /Browser-based Alerts	• Web/Browser based alerts are primarily called Javascript alerts and are those alerts that are browser dependent. These alerts are majorly called Popups. • *Web-based alerts can further bifurcate into :* a. ***Simple alerts*** b. ***Prompt alerts*** c. ***confirmation alerts***	@Amol
JavaScript Alerts ***driver.switchTo().alert();*** **Alert interface provides only 4 abstract methods in selenium** ***alert.getText();*** ***alert.accept();*** ***alert.dismiss();*** ***alert.sendKeys();*** ***alert.contains();***	**public class** JsAlertPopUp { **public static void** main(String[] args) { WebDriverManager.*chromedriver*().setup(); WebDriver driver=**new** ChromeDriver(); driver.get("https://the-internet.herokuapp.com/javascript_alerts"); ***1) Just a JS-Simple Alert popup*** *Simple alert: These alerts are just informational alerts and have an OK button on them. Users can click on the OK button after reading the message displayed on the alert box. A simple alert box looks like below:*	@Amol

-As per selenium official document we don't have contains method in Alert interface.




```
driver.findElement(By.cssSelector("button[onclick='jsAlert()']")).click();
```

@Amol

			@Amol

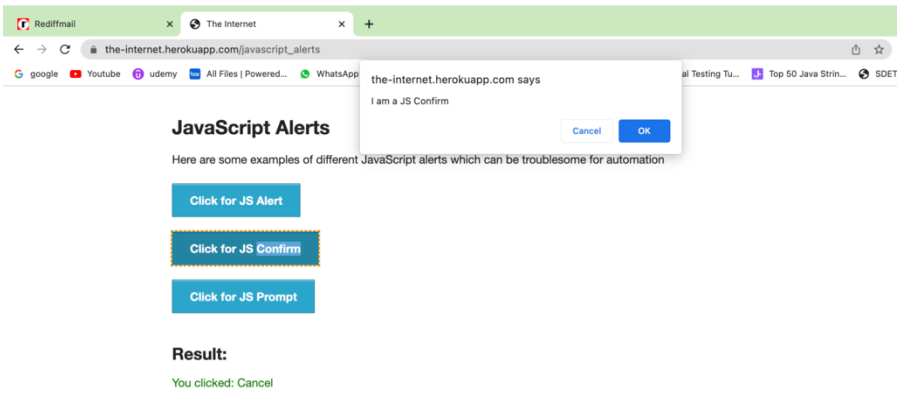
```
Alert alert=driver.switchTo().alert();

String alertTxt=alert.getText();
System.out.println(alertTxt);

alert.accept();              //It clicks on OK button
```

System.***out***.println(driver.findElement(By.*xpath*("//p[@id='result']")).getText());
//Result msg---"*You successfully clicked an alert*"

---

2) JS-Alert with confirmation popup : OK/CANCEL

These alerts get some confirmation from the user in the form of accepting or dismissing the message box. They are different from prompt alerts in a way that the user cannot enter anything as there is no text-box available. Users can only read the message and provide the inputs by pressing the OK/Cancel button.



driver.findElement(By.*cssSelector*("button[onclick='jsAlert()']")).click();

Alert alert2=driver.switchTo().alert();

String alertTxt2=alert2.getText();
System.***out***.println(alertTxt2);

alert2.accept(); //It clicks on OK
//alert2.dismiss(); // It clicks on Cancel

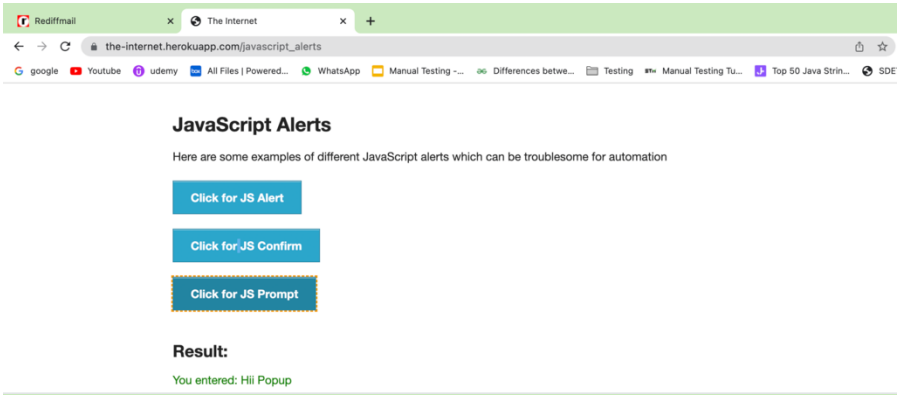
System.***out***.println(driver.findElement(By.*xpath*("//p[@id='result']")).getText());
//Result msg---"*You successfully clicked an alert*"

---

3) JS-alert with Prompt -

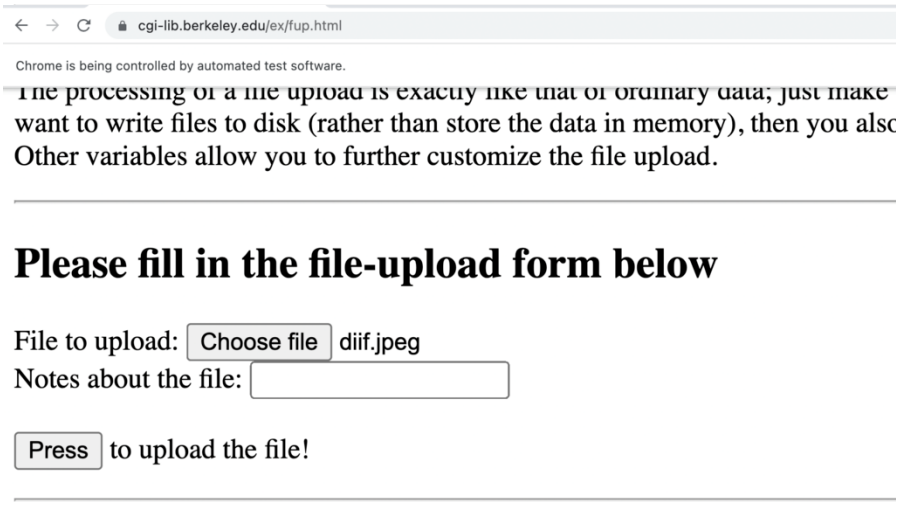
In Prompt alerts, some input requirement is there from the user in the form of text needs to enter in the alert box. A prompt alert box is displayed like below, where the user can enter his/her username and

press the OK button or Cancel the alert box without entering any details.



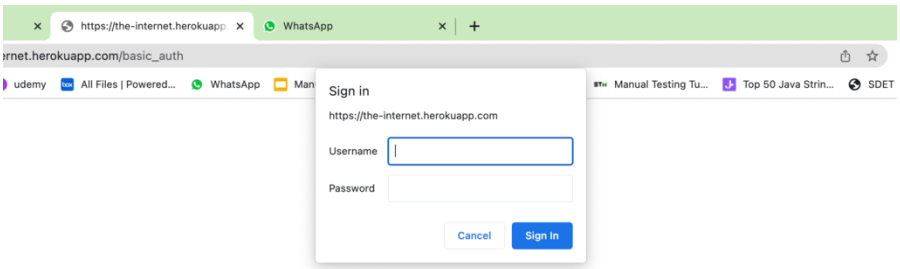

```
driver.findElement(By.xpath("//button[contains(text(),'Click for JS Prompt')]")).click();

        Alert alert3= driver.switchTo().alert();
        alert3.getText();
        alert3.sendKeys("Hii popup");
        alert3.accept();       //It clicks on OK
        alert3.dismiss();    // It clicks on Cancel

System.out.println(driver.findElement(By.xpath("//p[@id='result']")).getText());
//Result msg---"You entered: Hii popup"

driver.close();
    }
}
```

NoAlertPresentException	***Why does NoAlertPresentException occur in Selenium?*** • Generally, the Selenium program driver tries to switch to an alert box or pop-up that is displayed on top of the webpage when we click on the button. • However, if there is no alert box present at the time of switching, then Selenium raises NoAlertPresentException instead. • NoAlertPresentException: no such alert • In some cases, this may happen because the website you want to automate may not have loaded fully. It's also possible that the alert box element is not loaded into DOM yet.	@Amol

	• When we try to run the code below, it raises NoAlertPresentException. • This is because the code expects an alert button to be clicked, which it will eventually switch to. • However, in the following code, we try to switch to the alert box without clicking the button. driver.get("https://the-internet.herokuapp.com/javascript_alerts"); //driver.findElement(By.cssSelector("button[onclick='jsAlert()']")).click(); Alert alert=driver.switchTo().alert(); String alertTxt=alert.getText(); System.*out*.println(alertTxt); alert.accept();	
File Upload PopUps *where type="file" attribute is mandatory.*	 driver.get("https://cgi-lib.berkeley.edu/ex/fup.html"); driver.findElement(By.*xpath*("//input[@type='file']")).sendKeys("/Users/amoldhondge/Downloads/diif.jpeg"); • No need to click on the '**choose file**' just give the path of a file through .sendKeys() method.	@Amol
Authentication PopUps		@Amol

(Non-JS popups)

These are not JavaScript popups



- Here you have to pass the username and password along with the URL in following format:

https://**Username:Password@**the-internet.herokuapp.com/basic_auth

- **public class** AuthenticationPopUp {

public static void main(String[] args) {

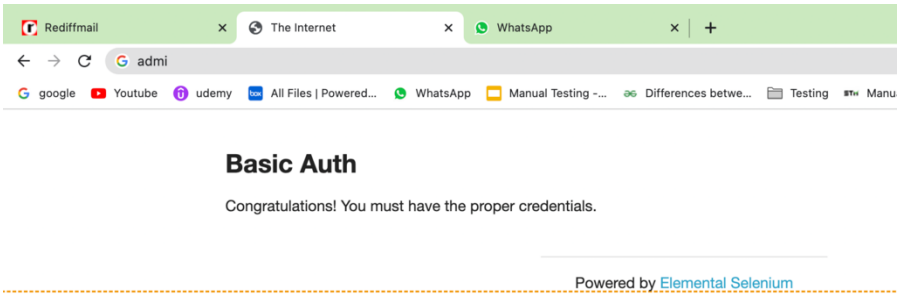
WebDriverManager.*chromedriver*().setup();
WebDriver driver= **new** ChromeDriver();

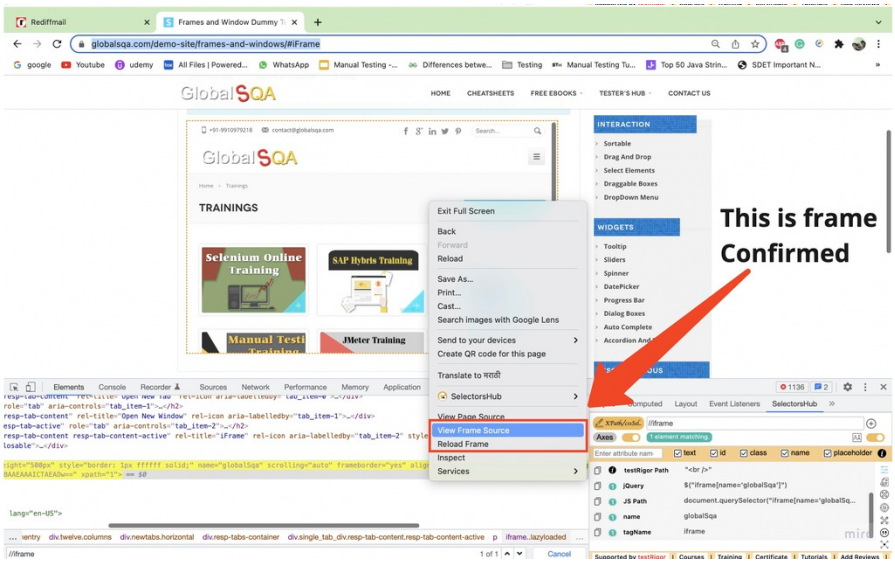
driver.get("https://admin:admin@the-internet.herokuapp.com/basic_auth");

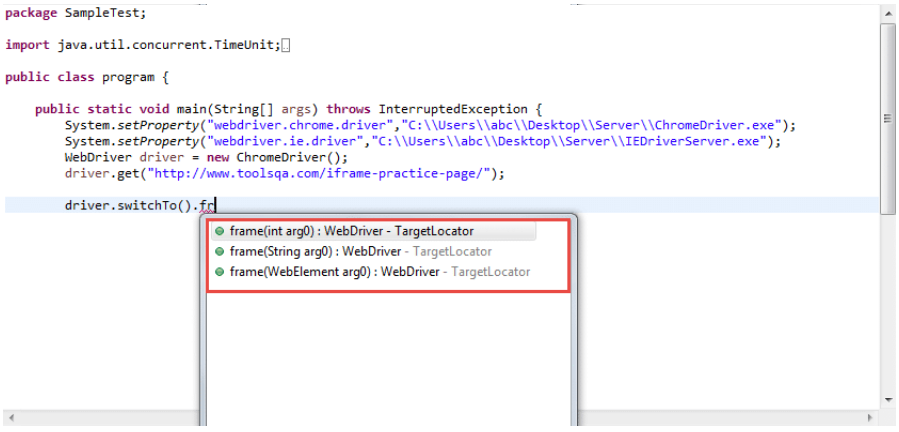
}

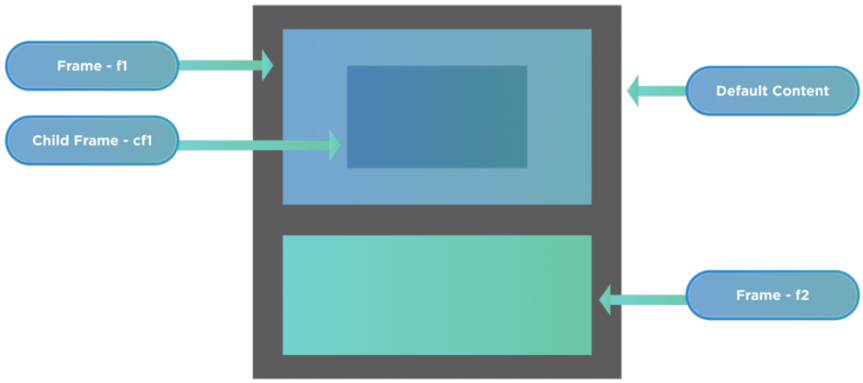
}

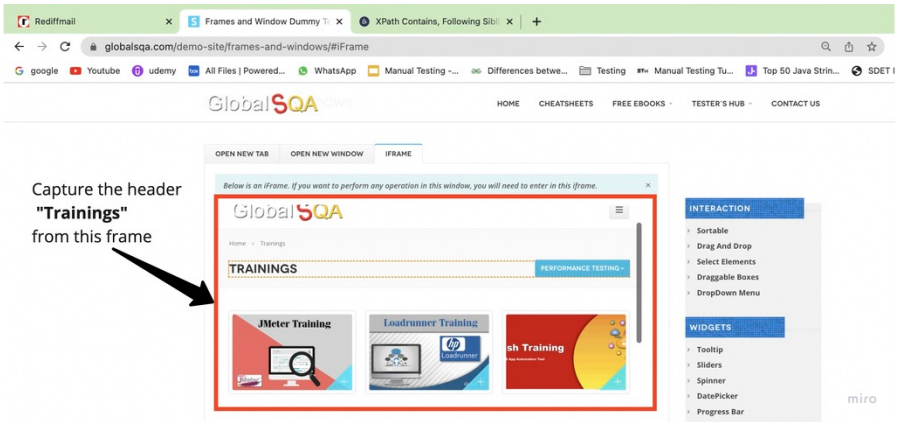
- After successful login it will directly give login status as below, you will not see any prompt.



	# *Frame Handling*	
What are frames/iframes ?	• An ***iframe*** is also known as the ***inline frame***. • It is a tag used in HTML5 to embed an HTML document within a parent HTML document. An iframe tag is defined using **<iframe> </iframe>** tags. • The iframes are mainly ***used to insert content from external sources.*** For example, ***an advertisement displayed on a web page.*** They can float within the webpage, which means one can position an iframe at a specific position on a web page. • Every frame is a WebElement and has separate html DOM structure with #document. • The frame can have both type of HTML tags frame and iframe.	@Amol
How to Identify a Frame on a Page?	☐ Right-click on the specific element and check all the options. If you find an option like **This Frame**, **view Frame source** or **Reload Frame,** it means the page includes frames.  ☐ Right click on the page and click 'View Page Source' and Search with the 'iframe', if you can find any tag name with the 'iframe' then it is meaning to say the page consisting an iframe. ☐ ***NOTE*** ◦ We can even identify total number of iframes by using ◦ **//By finding all the web elements using iframe tag:**	@Amol

	List<WebElement> iframeElements= driver.findElements(By.tagName("iframeResult")); System.***out***.println("Total number of iframes are " + iframeElements.size());	
How to Handle iframe with Selenium WebDriver: *Using the SwitchTo().frame function*	☐ ***Switch to the frame by index:*** ○ driver.switchTo().frame(*0*); ○ driver.switchTo().frame(*1*); ☐ ***Switch to the frame by Name or ID:*** ○ driver.switchTo().frame("*name* of the element"); ○ driver.switchTo().frame("*id* of the element"); ☐ ***Switch to the frame by Web Element:*** //First find the element using any of locator strategy WebElement *iframeElement* = driver.findElement(By.id("IF1")); //now use the switch command driver.switchTo().frame(*iframeElement*) We can switch between two frames by using these three methods here the method ***frame*** is being overloaded.  	@Amol
How to switch back to the parent iframe from the child iframe?	• ***driver.switchTo().parentFrame():*** *This will pass the control to the immediate parent frame of the current frame.* ○ for ex. **switch from cf1---->f1**	@Amol



How to switch the context back to the main web page from the parent/child iframe?	• ***driver.switchTo().defaultContent():*** *This will pass the control to the main document which contains the iframes.* • *It switches the context back to the main page, no matter how deep the current context of the WebDriver is.* ○ *Here Ex.* **switch from cf1---->Default content main web page**	
NoSuchFrame Exception	• When frame is not present	
Program to demonstrate how to switch from webPage to frame. ***URL:*** ***https://www.globalsqa.com/demo-site/frames-and-windows/#iFrame***	 **public class** FrameHandle { **public static void** main(String[] args) { WebDriverManager.*chromedriver*().setup(); WebDriver driver= **new** ChromeDriver(); driver.get("https://www.globalsqa.com/demo-site/frames-and-windows/#iFrame"); //Counting the number of frames no the page	@Amol

```
            List<WebElement> iframeElements =
driver.findElements(By.tagName("iframe"));
            System.out.println("The number of frames present are
:"+iframeElements.size());


        //Switching to frame by passing the name attribute of the
iframe
            driver.switchTo().frame("globalSqa");

        //Capturing the header text and storing it in String
            String headerName=driver.findElement
             (By.xpath("//div/h1[contains(text(),'Trainings')]")).getText();


       //Printing the headerText of the frame
            System.out.println(headerName);


  //Now switching back to the main page and clicking on "open new
tab"
            driver.switchTo().defaultContent();
            Thread.sleep(3000);
            driver.findElement(By.xpath("//li[@id='Open New
Tab']")).click();
      }
}
```

Selenium 4.x feature	driver.switchTo().parentFrame(); - This is the newly added feature in selenium 4.x	

SeleniumSession -11 Date 01/03/22

Topic: BrowserWindowHandle_newWindow_Selenium4.x_Changes

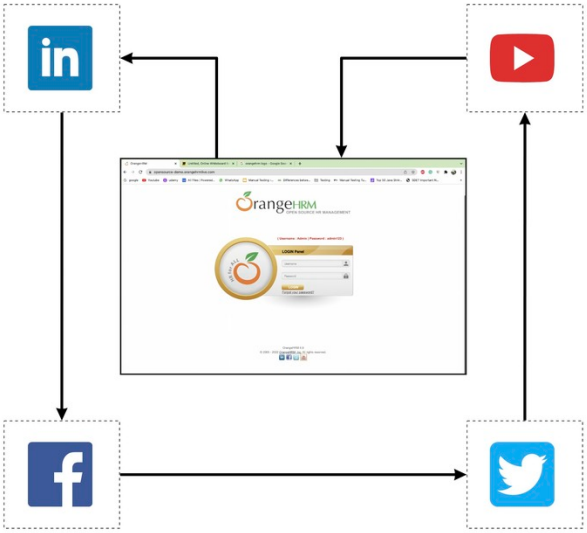
Selenium 4.x feature:	*driver*.switchTo().**newWindow**(WindowType.***WINDOW***);	@Amol

Switch to new Window/Tab	*driver*.switchTo().**newWindow**(WindowType.***TAB***);	
What is windowHandle?	• A window handle stores the unique address of the browser windows. • It is just a pointer to a window, whose return type is *alphanumeric*.	@Amol
How to handle multiple browser windows by using getWindowHandles() method?	☐ In order to switch from one window to other we use : ***driver*.switchTo().window(WindowID/Name);** ☐ Get the list of all the handles which are currently open by : **Set<String> handles=*driver*.getWindowHandles();** ▪ Set<String> is non-ordered list. ☐ How to get Window ID ? ◦ There are 2 ways u can get the windowID ◦ Iterate through the handles list using iterator interface: **Iterator<String> it=handles.iterator();** **String ParentWindowID=it.next();** ◦ OR Use ArrayList<String> instead: **List<String> handlesList= new ArrayList<>(handles);** **String ParentID= handlesList.get(0);** **String ChildID= handlesList.get(1);**	@Amol
getWindowHandle() vs getWindowHandles()	Selenium WebDriver provides you two methods ***getWindowHandle***() and ***getWindowHandles***() which are used to get window handle/s. Differences between both are given below:- 1. ***getWindowHandle***() returns the window handle of currently focused window/tab. ***getWindowHandles***() returns all windows/tabs handles launched/opened by the same driver instance including all parent and child window. 2. Return type of ***getWindowHandle***() is String while return type of ***getWindowHandles***() is Set<String>. The return type is Set as window handle is always unique. 3. ***getWindowHandles***() internally uses LinkedHashSet. So	@Amol

whatever Set it returns, it will give window handles in the order it is opened.

4. SYNTAX:
   - String handle= *driver*.getWindowHandle();
   - List<String> Handles=*driver*.getWindowHandles();

Assignment **- *Iterating all 4 child windows - closing them and returning to Parent window.***



```
public class BrowserWindowPopUp {
	static WebDriver driver;
	public static void main(String[] args) throws InterruptedException
{

		WebDriverManager.chromedriver().setup();
		 driver= new ChromeDriver();
		driver.get("https://opensource-demo.orangehrmlive.com/");

	 By LinkedIn=By.xpath("//img[@alt='LinkedIn OrangeHRM
group']");
	 By Facebook=By.xpath("//img[@alt='OrangeHRM on Facebook']");
	 By twitter=By.xpath("//img[@alt='OrangeHRM on twitter']");
	 By youtube=By.xpath("//img[@alt='OrangeHRM on youtube']");

		ElementUtil eleUtil=new ElementUtil(driver);
		eleUtil.doClick(LinkedIn);
		eleUtil.doClick(Facebook);
```

@Amol @hemanthSibba

```
            eleUtil.doClick(twitter);
            eleUtil.doClick(YouTube);

            Thread.sleep(3000);

            Set<String> handles=driver.getWindowHandles();
            Iterator<String> it=handles.iterator();

            String parentWindowID=it.next();
            String YoutubeWindowID=it.next();
            String TwitterWindowID=it.next();
            String FacebookWindowID=it.next();
            String LinkedInWindowID=it.next();

 String
LinkedInWindowTitle=getChildWindowTitle(LinkedInWindowID);
    System.out.println("LinkedInWindowTitle: "+LinkedInWindowTitle);
    driver.close();

String FacebookWindowTitle= getChildWindowTitle
(FacebookWindow ID);
    System.out.println("FacebookWindowTitle:
"+FacebookWindowTitle);
    driver.close();

String TwitterWindowTitle= getChildWindowTitle(TwitterWindowID);
     System.out.println("TwitterWindowTitle: "+TwitterWindowTitle);
     driver.close();

String YoutubeWindowTitle=
getChildWindowTitle(YoutubeWindowID);
      System.out.println("YoutubeWindowTitle: "+YoutubeWindowTitle);
       driver.close();

String parentWindowTitle= getChildWindowTitle(parentWindowID);
      System.out.println("parentWindowTitle: "+parentWindowTitle);
      driver.quit();
}

      public static String getChildWindowTitle(String ChildWindowID) {
```

```
        driver.switchTo().window(ChildWindowID);
        String ChildWindowTitle=driver.getTitle();
        return ChildWindowTitle;
    }

}
```

Assignment

In above program close all the child windows except Parent Window

@Amol

```
public class BrowserWindowIterator_WhileLoop {

    static WebDriver driver;
    public static void main(String[] args) {

        WebDriverManager.chromedriver().setup();
        driver= new ChromeDriver();

        driver.get("https://opensource-demo.orangehrmlive.com/");
        String ParentWindowTitle=driver.getTitle();

        ElementUtil eleUtil= new ElementUtil(driver);

    By LinkedIn= By.xpath("//img[@alt='LinkedIn OrangeHRM group']");
    By Facebook= By.xpath("//img[@alt='OrangeHRM on Facebook']");
    By twitter= By.xpath("//img[@alt='OrangeHRM on twitter']");
    By youtube= By.xpath("//img[@alt='OrangeHRM on youtube']");

            eleUtil.doClick(LinkedIn);
            eleUtil.doClick(Facebook);
            eleUtil.doClick(twitter);
            eleUtil.doClick(youtube);


            Set<String> handles=driver.getWindowHandles();

            Iterator<String> it=handles.iterator();

            while(it.hasNext()) {

            String windowID= it.next();
            driver.switchTo().window(windowID);
```

```
                String WindowTitle=driver.getTitle();
                System.out.println(WindowTitle);

               if( !WindowTitle.equals(ParentWindowTitle))
                driver.close();

                }
        }

}
```

Assignment:

1) Click on the LinkedIn link

2) GetTheTitle of childWindow (LinkedIn)

3) Close the childWindow

4) Get back to the parentWindow

5) Capture the title of parentWindow

---

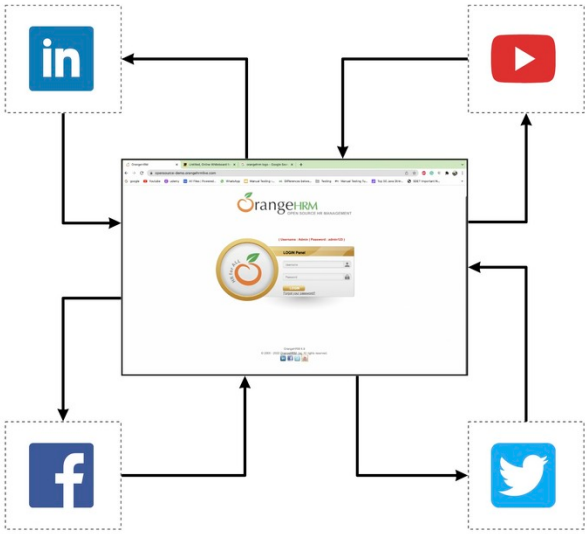
1) Click on the faceBook link

2) GetTheTitle of childWindow (faceBook)

3) Get back to the parentWindow

4) Capture the title of parentWindow

@Amol




```
public class BrowserWindowPopUp_2 {

 static WebDriver driver;
public static void main(String[] args) throws InterruptedException {


                WebDriverManager.chromedriver().setup();
                driver= new ChromeDriver();

        driver.get("https://opensource-demo.orangehrmlive.com/");

        By LinkedIn=By.xpath("//img[@alt='LinkedIn OrangeHRM group']");
        By Facebook=By.xpath("//img[@alt='OrangeHRM on Facebook']");
        By twitter=By.xpath("//img[@alt='OrangeHRM on twitter']");
        By youtube=By.xpath("//img[@alt='OrangeHRM on youtube']");
```

So on.....

.

.

.

.

.

.

.

.

.

.

```
        getWindowsTitles(LinkedIn);
        getWindowsTitles(Facebook);
        getWindowsTitles(twitter);
        getWindowsTitles(youtube);
        driver.quit();

}
    public  static void getWindowsTitles(By locator) {

    //1) Click on the childTab link
        doClick(locator);

        Set<String> handles=driver.getWindowHandles();
        Iterator<String> it=handles.iterator();

    //2) Capture the title of parentWindow
        String ParentWindowID=it.next();
        System.out.println(getWindowTitle(ParentWindowID));

    //3) Capture the title of childWindow
        String ChildWindowID=it.next();
        System.out.println(getWindowTitle(ChildWindowID));

        driver.close();
        driver.switchTo().window(ParentWindowID);

    }
    public  static String getWindowTitle(String WindowID) {

        //getWindowTitle
        driver.switchTo().window(WindowID);
        String WindowTitle=driver.getTitle();
        return WindowTitle;
    }

    public static void doClick(By locator) {
        //doClick()
        getElement(locator).click();
    }
```

	public static WebElement *getElement*(By locator) { //getElement() return *driver*.findElement(locator); } }	
introduce in selenium 4.0	//Switch to a diffrent domain String wid=driver.getWindowHandle(); //Used to get current window session id driver.switchTo().newWindow(WindowType.WINDOW);//Used to open URL in new window driver.switchTo().newWindow(WindowType.TAB);//Used to open URL in new Tab i	@June d

SeleniumSession -12 Date 02/03/22

Topic:ActionsClass_DragnDrop_MouseOver_ContextClick_Click_SendKeys

1. https://www.bigbasket.com/?nc=logo (MouseOver)
2. https://www.udemy.com/(MouseOver)
3. https://www.spicejet.com/(MouseOver)
4. https://demo.guru99.com/test/simple_context_menu.html(Double/context click)

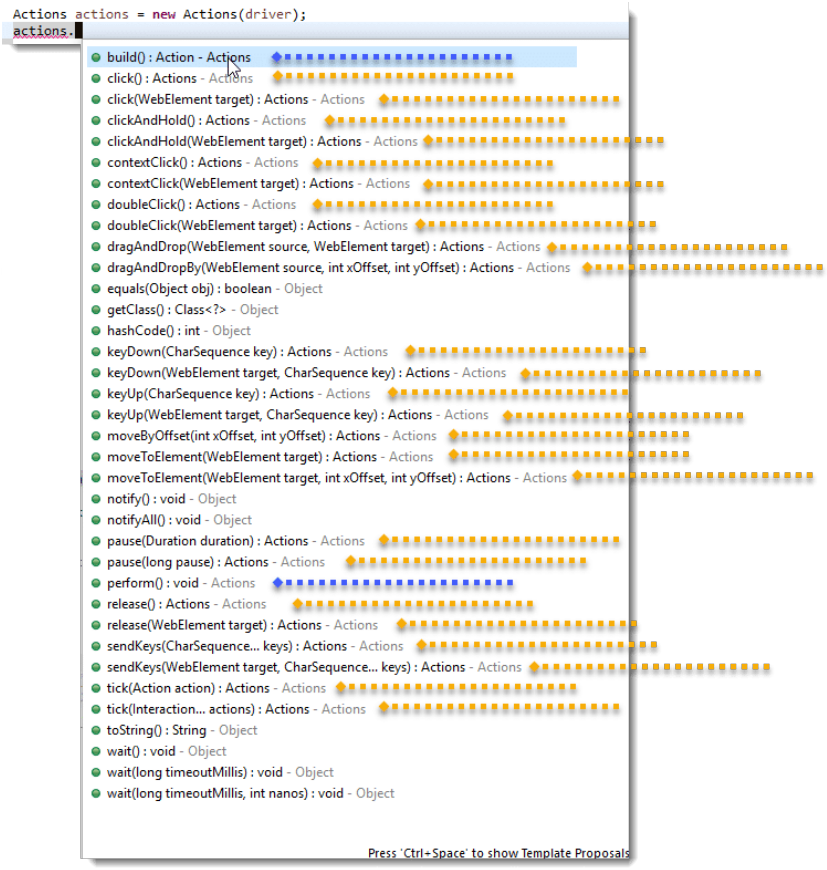
Refer below Java_Class_Files for this Session:

- DragAndDropConcept.java
- MouseOverConcept.java
- RightClickConcept.java
- ActionsClickAndSendKeys.java

Action vs ActionsClass	**Actions Class** • It is a ***class*** and the package is ***org.openqa.selenium.interactions.Actions*** • It represents collection of individual Action that you want to perform. • Using this class we can handle keyboard and mouse events.	@Amol

i.e.,

- Keyboard interface methods
- Mouse interface methods



Action Interface

- Action is an ***interface***
- It represents ***single user interaction.***
- Using this interface, on the Actions object we perform series of actions.
- Most widely and must use method is **perform()** after creating series of actions and storing in Action

How to Use Actions class in Selenium?	☐ ***Instantiate Actions class:*** • Actions class object is needed to invoke to use its methods. ○ ***Actions act = new Actions();*** • It needs the WebDriver object to initiate its class. ○ ***Actions actions = new Actions(webdriver object);*** ☐ ***Import package:*** • *Actions class & Action class* reside in	@Amol

	org.openqa.selenium.Interactions package of *WebDriver API.* • To consume these, import their packages: ***import org.openqa.selenium.interactions.Actions;*** ☐ ***Generate actions sequence:*** • Complex action is a sequence of multiple actions ◦ ***Actions a = new Actions(driver);*** ***a.moveToElement(WebElement).*** ☐ ***Build the actions sequence:*** • Now, build this sequence using the *build() method* of Actions class and get the composite action. • Build method generates a composite action containing all actions so far which are ready to be performed. ◦ ***a.moveToElement(WebElement).build().*** ☐ ***Perform actions sequence:*** • And finally, perform the actions sequence using *perform()* method of Action Interface.*action.perform();* ◦ ***a.moveToElement(WebElement).build().perform();***	
DragAndDrop functionality ***Note:*** • *If only one user action need to perform, then perform() working properly.* • *If Multiple user actions need to perform, then have to write build() and perform() requires to complete*	Element to Element drag and drop functionality. https://jqueryui.com/resources/demos/droppable/default.html **Approach#1:** • Locate Source WebElement • Locate Target WebElement • Create object of Actions class • Perform various actions like ◦ clickAndHold(source) ◦ moveToElement(target) ◦ release ◦ build ◦ perform ========== **Code Implementation** WebElement sourceElement =	@Dhrumil

desired work.	driver.findElement(By.id("draggable")); WebElement targetElement = driver.findElement(By.id("droppable")); Actions act = new Actions(driver); //Object creation of Actions class, make sure to pass driver parameter act.**clickAndHold**(sourceElement).**moveToElement**(targetElement) .**release**().**build**().**perform**(); ========== Mandotary to write build() and perform() to complete action. ========== **Alternative Approach:** act.dragAndDrop(sourceElement, targetElement).perform();	
MouseHover Functionality ***Assignment**: SpiceJet Application >> Move to Add-Ons >> Select any option* ***Assignment**: BigBasket > Menu and Submenu >>*	**Approach#** • Locate the Menu • Define Action class • moveToElement and perform • wait for 2 sec • Locate desired element and click it. ========== **Code Implementation:** **public class** **MouseHoverConcept** { **static** WebDriver *driver*; **public static void** main(String[] args) **throws** InterruptedException { WebDriverManager.*chromedriver*().setup(); *driver*= **new** ChromeDriver(); *driver*.get("http://mrbool.com/course/"); WebElement CONTENT=*driver*.findElement (By.*xpath*("//a[@class='menulink']"));	@Dhrumil @Amol

	Actions a = new Actions(*driver*); a.moveToElement(CONTENT).perform(); Thread.*sleep*(3000); WebElement Courses= *driver*.findElement (By.*linkText*("COURSES")); Courses.click(); *driver*.close(); } } ========== **NOTE:** ☐ While finding element by LinkText make sure its case(upper/lower) is matching the visible linkText on the webPage . ☐ If the linkText visible on the page is in capital case so pass the Capital text, otherwise it will throw **NoSuchElementException.**	
SelectSubMenu Utility	selectSubMenu(By parentMenu, By childMenu) utility ========== **public void *selectSubMenu***(By parentMenu, By child1Menu) **throws** InterruptedException { //selectSubMenu Actions a= new Actions(driver); a.moveToElement(getElement(parentMenu)).perform(); Thread.*sleep*(2000); getElement(child1Menu).click(); }	@Dhrumil
selectSubMenu Level3 Utility	selectSubMenu(By parentMenu, By childMenu, By subChildMenu) utility ========== **public void *selectSubMenu***(By parentMenu, By child1Menu, By	@Dhrumil

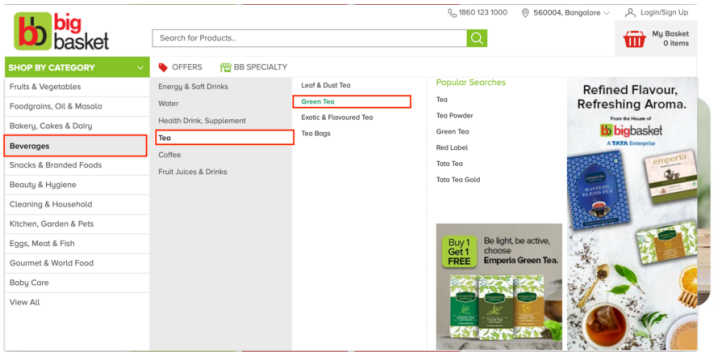
```
child2Menu) throws InterruptedException {

        Actions a= new Actions(driver);
        a.moveToElement(getElement(parentMenu)).perform();
        Thread.sleep(2000);

        a.moveToElement(getElement(child1Menu)).perform();
        Thread.sleep(2000);

        getElement(child2Menu).click();
    }
```

Assignment:

BigBasket

Select SubMenu LEVEL 4

1. ***SHOP BY CATEGORY***
2. ***Beverages***
3. ***Tea***
4. ***Green Tea***



@Amol

```
public class MouseHoverBigBasket {

    static WebDriver driver;
    public static void main(String[] args) throws
InterruptedException {

        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();

        driver.get("https://www.bigbasket.com/");
        ElementUtil eleUtil = new ElementUtil(driver);

  By shopByCat = By.xpath("//li[@class='dropdown full-wid hvr-
drop']");
  By beverages= By.linkText("Beverages");
  By tea = By.linkText("Tea");
  By greenTea= By.linkText("Green Tea");

        eleUtil.selectSubMenu(shopByCat, beverages, tea,
greenTea);
```

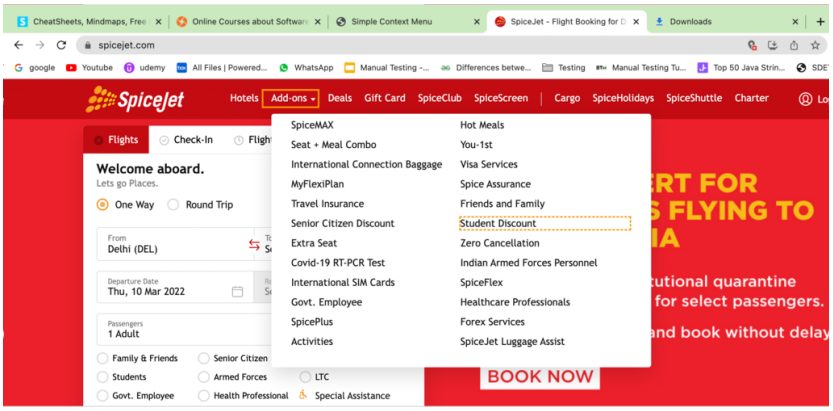
```
            Thread.sleep(2000);
            driver.close();
        }

}
```

Assignment:

SpiceJet

Select SubMenu LEVEL 2

1. ***Add-ons***
2. ***Student Discount***




```
public class MouseHoverSpiceJet {
    static WebDriver driver;

    public static void main(String[] args) throws
InterruptedException {

        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();

        driver.get("https://www.spicejet.com/");
        ElementUtil eleUtil = new ElementUtil(driver);

        By addon= By.xpath("//div[text()='Add-ons']");
        By student= By.linkText("Student Discount");

        eleUtil.selectSubMenu(addon, student);
        Thread.sleep(2000);
        driver.close();
    }

}
```

Right Click concept OR

Approach#

- locate the WebElement on which you have to perform right

@Amol

@Dhrumil

Context Click concept

click

- Define Actions Class
  - Actions act = new Actions(driver)
- act.contextClick(WebElement).perform()

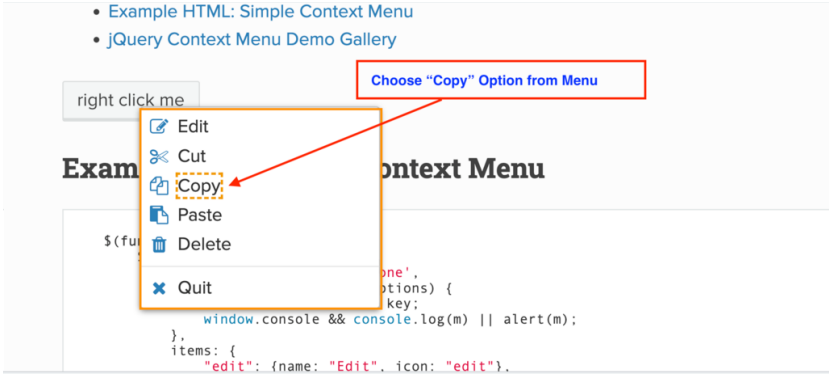
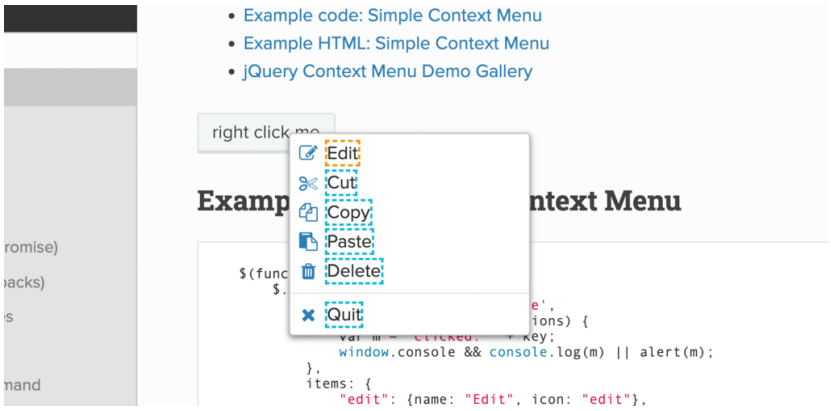
// No of Options available with the right click

- ItemList through findElements
- Iterate through Enhanced For Loop
- Print it.

==========

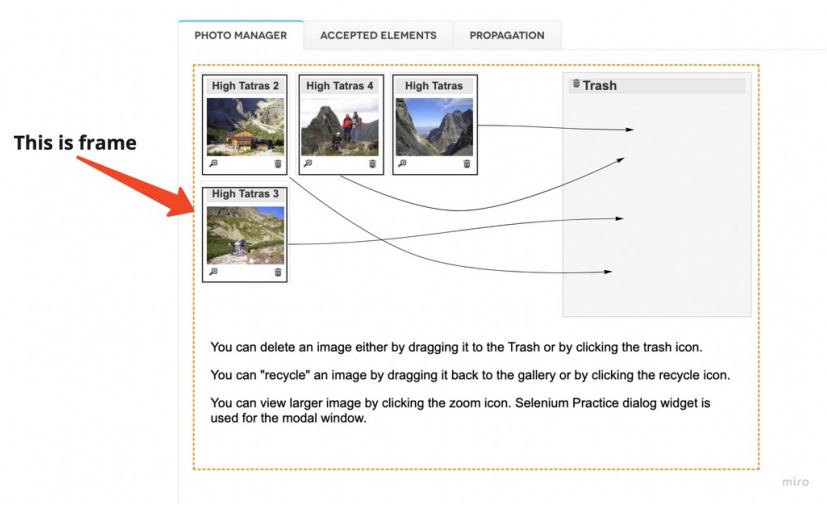
Code Implementation:

```
public class RightClick_OR_ContextClick {

      static WebDriver driver;
  public static void main(String[] args) throws
InterruptedException {

            WebDriverManager.chromedriver().setup();
            driver= new ChromeDriver();
            driver.get("https://swisnl.github.io/jQuerycontextMenu
/demo.html");

   By rightClickButton= By.xpath("//span[text()='right click me']");
   By rightClickOptions=By.xpath("//ul[@class='context-menu-list
context-menu-root']//span");

       Actions a= new Actions(driver);
       WebElement rightBtn=driver.findElement(rightClickButton);
       a.contextClick(rightBtn).perform();

        List<WebElement> itemsList = driver. findElements
                                              (rightClick Options);

          System.out.println(itemsList.size());

          for(WebElement e: itemsList) {
                String txt=e.getText();
                System.out.println(txt);
         }
```

	} 	
Utility for Right ClickMenu OR Context ClickMenu	 **public void** ***getRightClickMenu***(By rightClickTarget, By rightClickOptions , String value) { ***doContextClick***(rightClickTarget); List<WebElement> itemsList = ***getElements***(rightClickOptions); System.***out***.println(itemsList.size()); **for**(WebElement e: itemsList) { String txt=e.getText(); System.***out***.println(txt); **if**(txt.equals(value)) { e.click(); **break**; } }	@Dhrumil @Amol
Utility for getRightClickOptions List	 **public** List<String> ***getRightClickOptionsList***(By rightClickTarget, By rightClickOptions){	@Dhrumil @Amol

	doContextClick(rightClickTarget); List<String> rightClickItems= **new** ArrayList<String>(); List<WebElement> itemsList = ***getElements***(rightClickOptions); System.***out***.println(itemsList.size()); **for**(WebElement e: itemsList) { String txt=e.getText(); rightClickItems.add(txt); } **return** rightClickItems;	
Utility for getRightClickOptionsCount	**public int *getRightClickOptionsCount***(By rightClickTarget, By rightClickOptions){ **return *getRightClickOptionsList***(rightClickTarget, rightClickOptions).size(); }	@Dhrumil
Utility for doContextClick	**public void *doContextClick***(By locator) { Actions a = **new** Actions(driver); a.contextClick(***getElement***(locator)).perform();; }	@Dhrumil
ActionsClick() And ActionsSendKeys()	• You can perform sendKeys and Click action through Actions class also. ========== • Actions a = **new** Actions(*driver*); a.sendKeys(WebElement,"amol@gmail.com").perform(); a.sendKeys(WebElement,"amol@123").perform(); a.click(WebElement).perform();	@Dhrumil
Difference between click and actions.click()	• ***Clicks in the middle of the given element***. • Equivalent to: *Actions.moveToElement(onElement).click()*	@Dhrumil

method?	• In normal click() method, it click on the WebElement directly.	
Difference between sendKeys() and actions.sendKeys() method?	• Equivalent to calling: *Actions.click(element).sendKeys(keysToSend).* This method is different from WebElement.sendKeys(CharSequence) • Actions.sendKeys == Actions.click(WebElement).sendKeys("char");	@Dhrumil
ElementNotInteractableException	• When to use Actions.sendKeys() and Actions.click() method instead of normal click() and sendKeys() method, when you are getting ElementNotInteractableException.	@Dhrumil
doActionsClick() *utility*	/* * Clicks in the middle of the given element. Equivalent to: Actions.moveToElement(onElement).click() * @param locator */ public static void doActionsClick(By locator) { Actions a= new Actions(*driver*); a.click(*getElement*(locator)).perform();; }	@Dhrumil
doActionsSendKeys() *utility*	/* * Equivalent to calling: Actions.click(element).sendKeys(keysToSend). * @param locator * @param value */ public static void doActionsSendKeys(By locator, String value) { Actions a = new Actions(*driver*); a.sendKeys(*getElement*(locator), value).perform();; }	@Dhrumil

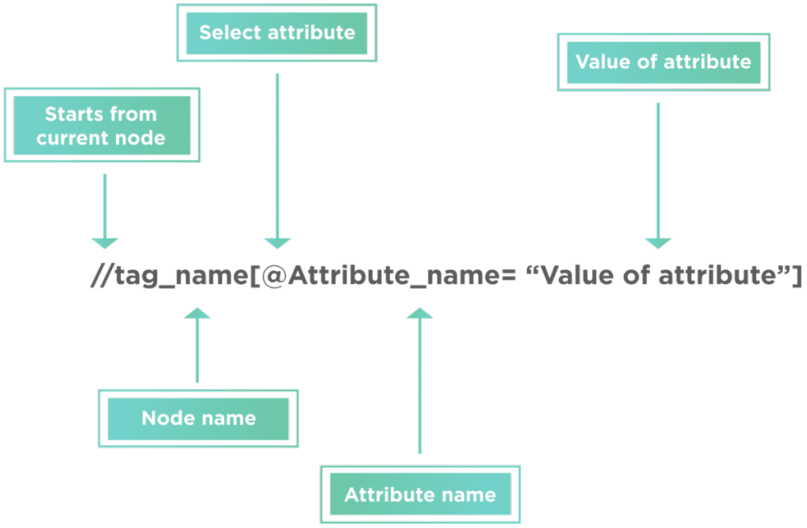
Drag_and_Drop Practice: *Drag all the 4 images inside a frame and drop in trash region:* ***URL*** *https://www.globalsqa.com/demo-site/draganddrop/*		@Amol

SeleniumSession -13 Date 03/03/22

Topic: Xpath_Basics_part_01

- Java_Class_File_Name: **Custom_Xpath_1.java**

What is Xpath?	• XPath, also known as XML Path, is one of the most commonly used locators in Selenium WebDriver that can help you navigate through the HTML structure of a page. • *X-Path is a query language* used for HTML and XML documents to locate any element in a web page using HTML DOM structure. ***NOTE:*** ☐ All the *HTML* web pages are internally represented as an *XML* document. Additionally, the XML document has a *tree-like structure*, where we have different tags and attributes.	@Dhrumil @Amol
What is DOM? W3C Standards: World Wide Web Consortium	• ***HTML/XML DOM*** *is a kind of platform/language neutral interface in the form of API's that allows programs and scripts to dynamically access and update the content, structure, and style of a document."* • DOM defines a standard way for accessing and manipulating XML documents. It presents an XML document as a tree-	@Amol

	structure. • JavaScript can access all the elements in a webpage making use of Document Object Model (DOM). • HTML DOM is provided by the browser and each web browser creates a DOM of the webpage when the page is loaded	
Xpath- *Its not attribute, it's a address of the webElement on the webpage.* *URL:* *https://demo.opencart.com /index.php?route=account/login*	☐ **Syntax:** **//tagName[@Attribute='Value']**  Example: //input[@id='input-email']	@Dhrumil @Amol

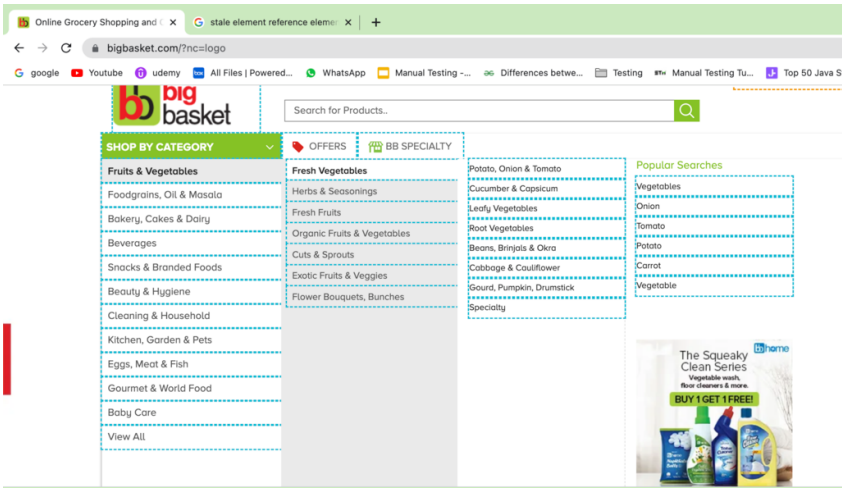
Meaning of different syntaxes in X-Path	<table><tr><th>*Syntax Element*</th><th>*Details*</th><th>*Example*</th><th>*Example Details*</th></tr><tr><td>***Single slash*** *"/"*</td><td>It selects the node from the root. In other words, if you want to select the first available node, this expression is used.</td><td>/html</td><td>It will look for the *HTML* element at the start of the document.</td></tr><tr><td>***Double slash*** *"//"*</td><td>It selects any element in the DOM that matches the selection. Additionally, it doesn't have to be the exact next node and can be present anywhere in the DOM.</td><td>//input</td><td>It will select the input node present anywhere in the *DOM*.</td></tr><tr><td>***Address sign*** *"@"*</td><td>It selects a particular ***attribute*** of the node</td><td>//@text</td><td>It will select all the elements which have text attribute.</td></tr><tr><td>***Dot*** *"."*</td><td>It selects the ***current*** node.</td><td>//h3/.</td><td>It will give the currently selected node, i.e., ***h3***.</td></tr><tr><td>***Double dot*** *".."*</td><td>It selects the ***parent*** of the current node.</td><td>//div/input/..</td><td>It will select the parent of the current node. The current node is input so that it will select the parent, i.e., ***"div"***.</td></tr><tr><td>***Asterisk*** *'"'*</td><td>It selects ***any element*** present in the node</td><td>//div/*</td><td>This matches with any of the child nodes of the ***"div"***.</td></tr><tr><td>***Address and Asterisk*** *"@"*</td><td>It selects ***any attribute*** of the given node.</td><td>//div[@*]</td><td>It matches any of the ***div*** nodes that contain at least one attribute of any type.</td></tr><tr><td>***Pipe*** *"|"*</td><td>This expression is used to select a ***different*** path.</td><td>//div/h5</td><td>//div/form</td></tr></table>	@Amol
Xpath using Logical operator AND *URL: https://demo.opencart.com /index.php?route=account/login*	☐ Syntax: **//tagname[@attribute1=**'value1' ***and*** **@attribute2=**'value2'] Example: //input[@type='submit' **and** @value='login']	@Dhrumil @Amol
Xpath using Logical Operator OR	☐ Syntax: **//tag[@attribute1=**'value1' ***or*** **@attribute2=**'value2']	@Dhrumil

URL: *https://demo.opencart.com /index.php?route=account/login*	Example: //input[@placeholder='E-Mail Address' or @name='Email']	
//*[@id] ***Note: Try to avoid *, instead use specific tag- Improves performance.***	☐ Returns all the webElements of all the tags in DOM which having id as attribute.	@Dhrumil
Xpath using contains() function *URL:* *https://demo.opencart.com /index.php?route=account/login*	☐ Syntax: **//tag[contains(@attribute, 'value')]** Example: ○ //input[contains(@id, 'Email')] ○ //input[contains(@placeholder,'E-Mail')] • Contains() is a very useful method in XPath. • It can be used for all such web elements whose value can change dynamically.	@Dhrumil
***contains()** with **Multiple** Attributes* *URL:* *https://demo.opencart.com /index.php?route=account/login*	☐ ***contains()** with **Multiple** Attributes* Example: ○ //input[contains(@id,'Email') ***and*** contains(@name, 'Email')] ○ //input[contains(@id,'Email') ***and*** (@placeholder,'E-Mail') ***and*** contains(@name,'email')]	@Dhrumil
Xpath- One Attribute with contains() method and Other attribute without contains()	☐ **Example:** ○ //input[contains(@id,'Email') **and** @type='text'] ○ //input[contains(@placeholder,'E-Mail') **and** @name='email']	@Dhrumil

URL: *https://demo.opencart.com /index.php?route=account/login*		
text() function usage in Xpath *URL:* *https://demo.opencart.com /index.php?route=account/login* Mainly we are using it for , <a> and <h> tags.	☐ Syntax: **//tag[text()="value"]** **Example:** ○ //a[text()="My Account"] ○ //a[text()='Address Book']	@Dhrumil
text() and @attr together in Xpath *URL:* *https://demo.opencart.com /index.php?route=account/login*	☐ **Syntax:** **//tag[text()="value" and @attribute='value']** **Example**: ○ //a[text()='Address Book' and @class='list-group-item']	@Dhrumil
Important: contains() with text() *URL:* *https://www.amazon.com/*	☐ **Syntax:** **//tag[contains(text(),'value')]** **Example:** ○ //span[contains(text(),'internationally')] Make sure, when you are using contains and text together, we are using "," , not "=".	@Dhrumil
Multiple contains() with text() in Xpath *URL:*	☐ **Syntax:** **//tag[contains(text(),'value') *and* contains(@attribute,'value')]** **Example:**	@Dhrumil

https://www.amazon.com/	◦ //a[contains(text(),'Gift') and contains(@href,'gift-cards')]	
Xpath using starts-with() *URL:* *https://www.amazon.com/*	☐ **Syntax:** **//tag[starts-with(text(),'value')]** • **XPath starts-with()** is a function used for finding the web element whose attribute value gets changed on refresh or by other dynamic operations on the webpage. **Example:** ◦ //a[starts-with(text(),'Registry')]	@Dhrumil
Xpath using starts-with()- and attribute value *URL:* *https://demo.opencart.com/index.php?route=account/login*	☐ **Syntax:** **//tag[starts-with(@attribute, 'value')]** **Example:** ◦ //input[starts-with(@id,'Email')] ◦ //input[starts-with(@class,'form-')]	@Dhrumil
//ends-with is deprecated	• Latest Xpath engine, it's deprecated.	@Dhrumil
Indexing/Positioning in Xpath	☐ **Indexing:** ◦ (//input[@class='form-control'])[2] ◦ **(X-path)[i]**	@Dhrumil
position() function in Xpath	☐ **Positioning:** ◦ (//input[@class='form-control'])[position()=2] ◦ **(X-path)[position()=i]** Either to use Indexing OR position function with Xpath.	@Dhrumil

last() function in Xpath	☐ **Syntax:** ○ **(X-path)[last()]** ○ **(X-path)[last()-1]** | *Predicate*	*Details*	*Example*	*Example Details* | |---|---|---|---| | ***Get the last node*** | We can get the last node using the function "***last()***" inside the square bracket. | //div/input[last()] | It will give us the last input node, which is the child of the div node. | | ***Get the node with specified Position*** | We can get the node from specific positions by using "***position()***" inside the square bracket. | //div/input[position()='2'] | It will provide us with the child node of div. In other words, input present at the second position in the hierarchy. | **Real Time Example of last() function:** Check on Amazon site, and check whether help is available with the last footer section. **Xpath:** ((//div[@class='navFooterLinkCol navAccessibility'])[last()]//a)[last()]	@AMOL @Dhrumil
Parent - Child Relationships	☐ Parent to Child-----***Forward Traversing*** ○ Parent X-path **/ child :: tagName** ○ Parent X-path **// child :: tagName** ☐ Child to Parent-----***Backward Traversing*** ○ Child X-path **/../../../..** ○ Child X-path **/parent:: tagName** • *Here /.. refers to immediate parent of a child.* • *In backward traversing you will always get immediate parent of a child no matter what you write / **or** //* **Backward Traversing is only possible with Xpath, not with CSS Selector.**	@AMOL

	☐ Child to GrandParents----*Backward Traversing* ○ Child X-path **/ancestor:: tagName**	
Child -Siblings Relationships	☐ Child to Following Sibling------*Forward Traversing* ○ Child X-path **//following-sibling :: tagName** ☐ Child to Preceding Sibling------*Backward Traversing* ○ Child X-path **//preceding-sibling :: tagName**	@AMOL
Assignment: BigBasket Category Iteration using advanced for:each loop	 **public class** BigBasket_iteration { **static** WebDriver *driver*; **public static void** main(String[] args) **throws** InterruptedException { WebDriverManager.*chromedriver*().setup(); *driver*= **new** ChromeDriver(); *driver*.get("https://www.bigbasket.com/?nc=logo"); WebElement SBC= *driver*.findElement(By.*xpath*("//li[@class='dropdown full-wid hvr-drop']")); Actions a= **new** Actions(*driver*);	@AMOL

```
        a.moveToElement(SBC).perform();
        Thread.sleep(3000);

List<WebElement>List1=driver.findElements
                        (By.xpath("//ul[@id='navBarMegaNav']//a"));

        for(WebElement e1:List1) {

            a.moveToElement(e1).perform();
            System.out.println(e1.getText());

List<WebElement> List2 =driver.findElements
          (By.xpath("(//ul[@class='nav nav-pills nav-stacked'])
[2]//a"));

            for(WebElement e2:List2) {
                a.moveToElement(e2).perform();
                System.out.println(e2.getText());

List<WebElement> List3 =driver.findElements
                        (By.xpath("(//div[@class='box'])[3]/ul/li/a"));

                for(WebElement e3:List3) {
                    a.moveToElement(e3).perform();
                    System.out.println(e3.getText());
                }
            }
        }

        driver.close();
      }

}
```

Syntax Xpath function:	.//*[@id] -->all attribute contains();--> Dynamic id .//htmltag[contains(@attrb,value)]	@Juned Shah

.//htmltag[contains(@attrb,value) and contains(@attrb,value)]

.//html[text()='value']

.//htmltag[contains(text(),'value')]

.//htmltag[contains(text(),'value') and contains(@attr,'value')]

.//a[start-with(text(),value)]

.//htmltag[start-with(text(),value)]

.//htmltag[start-with(@class,value)]

//indexing

(.//htmltag[@attr=value])[2] ----> Pointing secound element

(.//htmltag[@attr=value])[position()=2] ----> Pointing secound element

(//htmltag[@attr='value' or contains(@type='value')])[last()]

(//htmltag[@attr='value' or contains(@type='value')])[last()-1]

@ Direct child element '/'
(.//div[@attr,value])[2]/input[@class='value'] --> Parent to child

.//htmltag[@attr='']/child::input

@ Direct+Indirect Child element '//'

(.//div[@attr,value])[2]//input[@class='value'] --> Parent to child

.//htmltag[@attr='']//child::input

	Child to parent .//htmltag[@attr='value']/../../../../../ .//htmltag[@attr='value']/parent::div ---->Parent .//htmltag[@attr='value']/ancestor::div ---> Grand parent	

SeleniumSession -14 Date 04/03/22

Topic: WebTableHandling_Xpath_Axes

Java_Class_File_Names:

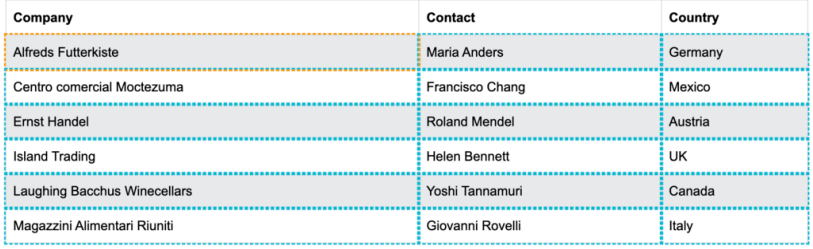
- **WebTableStaticTraverse.java**
- **WebTableCheckbox.java**
- **WebTableHandle.java**
- **BigBasketCategoryIteration.java**

Topics: WebTable Handing

- concept of following-sibling
- concept of preceding-sibling
- concept of moving left side of Element through Xpath traversing i.e. parent, preceding-sibling, child etc

Important Note:

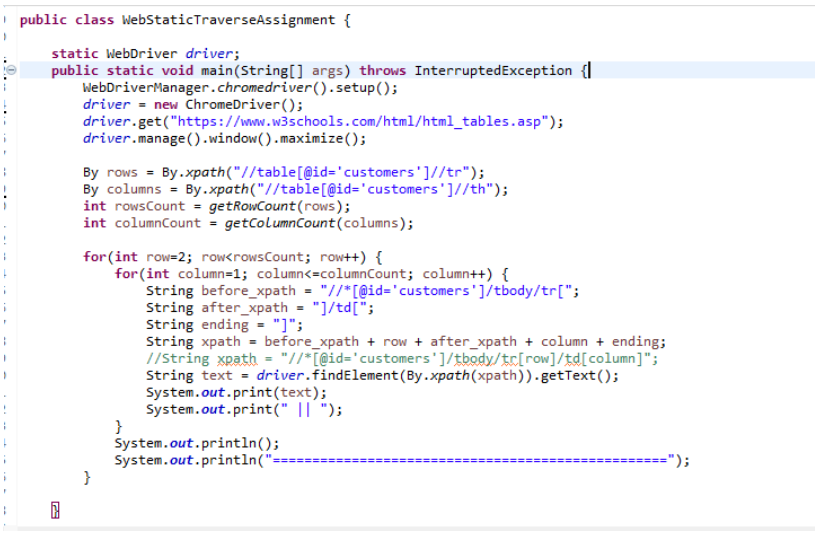
If you are trying to locate element on webpage, but on hovering and trying to locate in Elements window- if you are unable to find it in Elements section, then this way you can proceed.
Go to Source window of Browser > Press Function + F8 key > "Paused in debugger mode" > Enjoy Locating Xpaths.....

Assignment: ***Print this static table using two for loops one for row and another for column*** ***URL:***	 WebDriverManager.chromedriver().setup(); driver= new ChromeDriver();	@AMOL

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

https://www.w3schools.com /html/html_tables.asp %22

```
            driver.get("https://www.w3schools.com
/html/html_tables.asp");

            By rows=By.xpath("//table[@id='customers']//tr");
            By columns=By.xpath("//table[@id='customers']//th");


            int rowCount=driver.findElements(rows).size();
            int colCount=driver.findElements(columns).size();


 for(int row=2;row<=rowCount;row++) {

       for(int col=1;col<=colCount;col++) {

String xpathMain="//*[@id=\"customers\"] //tr["+row+"] /td
["+col+"]";

       String
txt=driver.findElement(By.xpath(xpathMain)).getText();
                  System.out.print(txt+" || ");
                   }
                   System.out.println();
System.out.println("-----------------------------------------------")
;
      }

      }

}

}
```

		@Dhrumil

```java
public class WebStaticTraverseAssignment {

    static WebDriver driver;
    public static void main(String[] args) throws InterruptedException {
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
        driver.get("https://www.w3schools.com/html/html_tables.asp");
        driver.manage().window().maximize();

        By rows = By.xpath("//table[@id='customers']//tr");
        By columns = By.xpath("//table[@id='customers']//th");
        int rowsCount = getRowCount(rows);
        int columnCount = getColumnCount(columns);

        for(int row=2; row<rowsCount; row++) {
            for(int column=1; column<=columnCount; column++) {
                String before_xpath = "//*[@id='customers']/tbody/tr[";
                String after_xpath = "]/td[";
                String ending = "]";
                String xpath = before_xpath + row + after_xpath + column + ending;
                //String xpath = "//*[@id='customers']/tbody/tr[row]/td[column]";
                String text = driver.findElement(By.xpath(xpath)).getText();
                System.out.print(text);
                System.out.print(" || ");
            }
            System.out.println();
            System.out.println("================================================");
        }

    }
```

SeleniumSession -15 Date 08/03/22

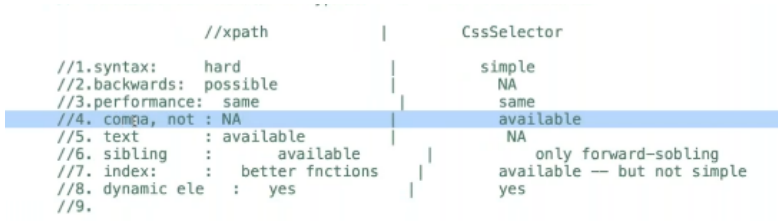
Topic: CssSelector

CSS selector with "**id**" attribute **<input type="text" name="search" value="" placeholder="Search" class="form-control input-lg">**	**Syntax:** • #id • tagname#id **Example:** • #input-email • input#input-email	@Dhrumil
CSS selector with "**class**" attribute	**Syntax:** • .classname • tagname.classname **Example:** • .form-control • input.form-control	@Dhrumil

CSS selector with combination of "**id**" and "**class**"	**Syntax:** • #id.classname • .classname#id **Example:** • #input-email.form-control • .form-control#input-email	@Dhrumil
CSS selector with **multiple classes**	**Syntax:** • .c1.c2.c3.c4..... **Example:** • By search = By.cssSelector(".nav-input.nav-progressive-attribute");	@Dhrumil
Important Note: By.classname only allowed to use single class. Multiple classname only allowed with Xpath and CssSelector.	Contact Sales locator through various ways: ☐ By.className("btn-orange contact-ohrm "); //Invalid ☐ By.xpath("//a[@class='btn-orange contact-ohrm ']"); **//valid** ☐ By.cssSelector("a.btn-orange contact-ohrm "); **//valid** ☐ By.className("contact-ohrm"); **//valid**	@Dhrumil
In Xpath, You have to pass complete value of attribute, and if you want to use single value of attribute then need to use different functions like contains() and starts-with etc.	**Example:** <button class="uiButton private-button private-button--primary private-button--default m-bottom-4 login-submit private-button--non-link" </button> **//valid one** ☐ By.xpath("//button[contains(@class, 'login-submit')]"); ☐ By.xpath("//button[@class='uiButton private-button private-button--primary private-button--default m-bottom-4 login-submit private-button--non-link']") **// Invalid one** ☐ By.xpath("//button[@class='login-submit']")	@Dhrumil

Basic Syntax for CSS Selector when Id and class attributes not available.	**Syntax:** • tagName[Attribute = 'value'] **Example:** • input[type='submit']	@Dhrumil
CSS Selector with multiple attribute value	**Syntax:** • tagName[Attribute1='value'][Attribute2='value']..... **Example:** • input[type='submit'][value='login']	@Dhrumil
CSS Selector with "*" which represents **contains** over here. contains() in Xpath == **"*"** in CSS selector	**Example:** • input[id*='email'] • button[class*='login-submit']	@Dhrumil
CSS Selector with **starts-with("^")** starts-with in Xpath == **"^"** in CSS Selector	**Example:** • input[id^='email']	@Dhrumil
CSS Selector with **ends-with("$")** ends-with in Xpath == "$" in CSS Selector	**Example:** • button[data-test-id$='button']	@Dhrumil
Parent to child traversing with CSS Selector ">"	• **Direct child element** ◦ div.private-form **>** input#username • **Direct + Indirect child element- "Space" between parent and Child.** ◦ div.private-form input#username4	@Dhrumil

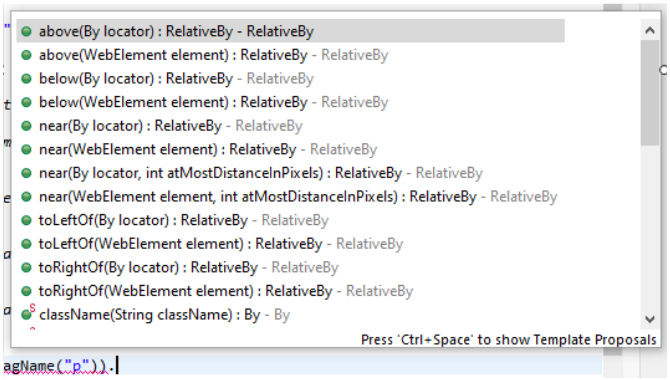
Important Interview Question: Is Backward traversing is possible with CSS Selector?	Not possible.	@Dhrumil
Sibling concept with CSS • **Following-sibling ==> "+"** • Preceding-sibling ==> Not possible	**Example:** • label.control-label + input#input-email	@Dhrumil
"not" in CSS Selector **Note:** Only works with class and ID fields. :not(name='search') >> This will not work	**Example:** • input.form-control.private-form-control:not(#username) This will locate password field.	@Dhrumil
Important feature: Comma in CSS Selector	**Example:** • input#username, input#email, input#loginButton This will locate 3 elements.	@Dhrumil
Interview Question: Can we use text() with CSS Selector?	text() based selectors is deprecated in CSS.	@Dhrumil
Indexing in CSS **nth-of-type(i)** **"i"** represent Index over here.	**Example:** • For 5th Index: ◦ ul.footer-nav li:nth-of-type(5) > a • For all the elements: ◦ ul.footer-nav li:nth-of-type(n) > a	@Dhrumil
Comparison between	1. Syntax	@Dhrumil

Xpath vs CSS	2. Backward traversing 3. Performance 4. Additional features like comma, not 5. text() 6. Sibling mechanism 7. Indexing 8. Dynamic Elements handling capability 	

SeleniumSession -16 Date 09/03/22

Topic: RelativeLocator_JSExecutor_Different_JS_Utilities

Relative Locators Selenium 4.x comes with a new feature which is called “**Relative Locators**” before it is called “**Friendly Locators**“.	**Pre-requisites:** To use relative locators, you have to import below package. ***import static org.openqa.selenium.support.locators.RelativeLocator.with;***	@Dhrumil
Different Relative Locators in Selenium 4.x	• **toLeftOf():** Element located to the left of specified element. • **toRightOf():** Element located to the right of the specified element. • **above():** Element located above with respect to the specified element. • **below():** Element located below with respect to the specified element. • **near():** Element is at most 50 pixels far away	@Dhrumil

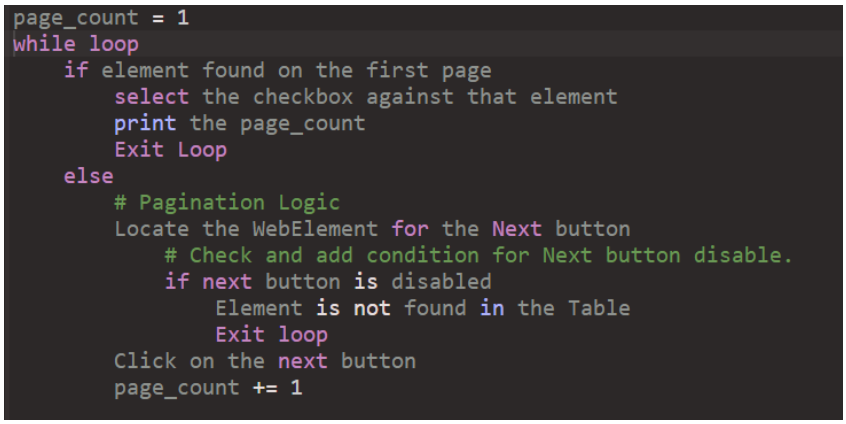
	from the specified element. The pixel value can be modified.	
Different methods available for relative locators.		@Dhrumil
JavaScript Executor		

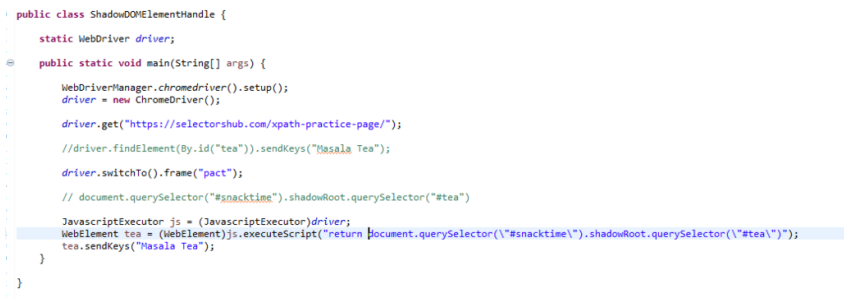
SeleniumSession -17 Date 10/03/22

Topic: WebTablePagination_SVGElement_ShadowDom_handle

Java Files for this Session:

- PaginationTest.java
- SVGElementHandle.java
- ShadowDOMElementHandle.java

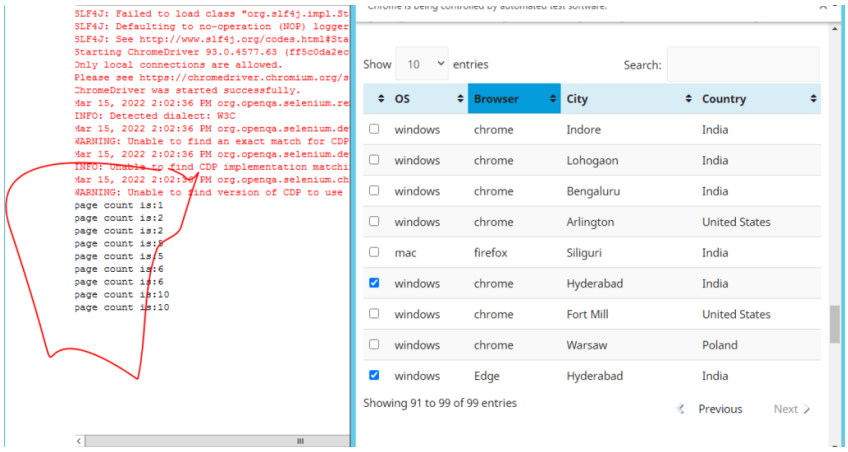
WebTablePagination concept		@Dhrumil
SVG Element - Only xpath works, CSS is not compatible.	local-name() and name() functions are used in SVG Element xpath.	@Gagan
Syntax to locate SVG Element	**//*[local-name()='svg']**.	@Dhrumil
SVG Element Syntax Example: URL: https://petdiseasealerts.org/forecast-map/#/	//*[local-name()='svg' and @id='map-svg']//*[name()='g' and @id='states']//*[name()='g']//*[name()='path']	@Dhrumil
SVG Element handler example		@Dhrumil
Shadow DOM Element handling	For Shadow DOM element, we have to use CSS Selector, xpath will not work.	@Dhrumil

Shadow DOM Element Example:	 	@Dhrumil
	We cannot automate the shadow dom element which having tag as "Close", limitation of Automation. shadow-root (close)	@Dhrumil
Assignment for pagination: Can anyone paste assignment code for pagination?	import java.util.List; import org.openqa.selenium.By; import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.chrome.ChromeDriver; import UtilsFiles.ElementUtils; import io.github.bonigarcia.wdm.WebDriverManager; public class BCC { static WebDriver driver; public static void main(String[] args) throws InterruptedException { WebDriverManager.chromedriver().setup(); driver=new ChromeDriver(); driver.get("https://selectorshub.com/xpath-practice-page/"); Thread.sleep(3000); //JavascriptExecutor js=(JavascriptExecutor)driver; By city=By.xpath("//td[text()='Hyderabad']"); //driver.findElement(By.xpath("//td[text()='Hyderabad']/../td/input[@type='checkbox']")).click(); ElementUtils e1=new ElementUtils(driver);	@anupama

```
int actualSize=e1.getElements(city).size();
//List<WebElement>checkboxlist=e1.getElements(city);
By checkList=By.xpath("//td[text()='Hyderabad']/../td
/input[@type='checkbox']");



int pagecount=1;
while(true)
{
	if(actualSize>0)
	{
		for(int i=0;i<e1.getElements(checkList).size();i++)
		{
			if(!e1.getElements(checkList).get(i).isSelected())
			{
			e1.getElements(checkList).get(i).click();
			}

		}
		System.out.println("page count is:"+pagecount);
		actualSize--;
		//pagecount++;
	//
driver.findElement(By.xpath("//td[text()='Hyderabad']/../td
/input[@type='checkbox']")).click();
	//	break;
	}
	else
	{
	WebElement
next=driver.findElement(By.linkText("Next"));
	if(next.getAttribute("class").contains(" disabled"))
	{

		break;
	}
	next.click();
	Thread.sleep(3000);
	pagecount++;
	actualSize=e1.getElements(city).size();
```

```
            }



        }


        }

        public static WebElement selectCheckBox(String name)
        {

            return
driver.findElement(By.xpath("//td[text()='"+name+"']/../td
/input[@type='checkbox']"));
        }

        public static List<WebElement> getElements(By locator)
        {
            return driver.findElements(locator);
        }
}
```

O/P



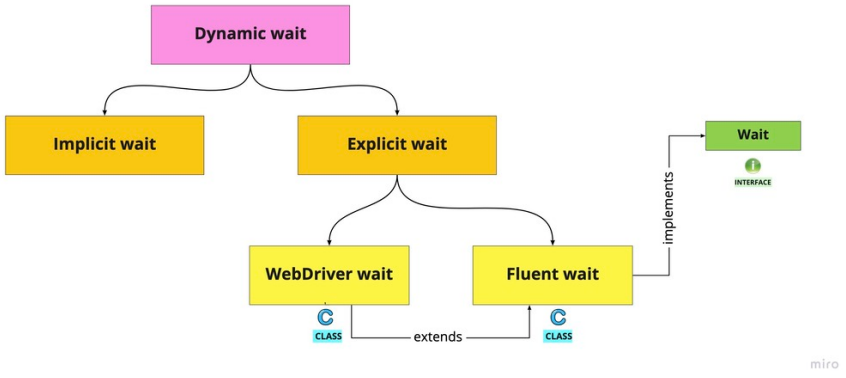
SeleniumSession -18 Date 14/03/22

Topic: Syncronization (Implicit wait/Explicit wait)

Java Class Files:

- ImplicitlyWaitConcept.java
- ExplicitWaitConcept.java
- WebDriverWaitWithPolling.java
- WaitForAlertConcept

Static-wait vs Dynamic-waits	• **Static wait**: ***Thread.sleep(5000);*** ◦ It will wait for total timeout 5 sec, no matter whether element is found or not. ◦ If element is found within 2 seconds, still script will wait for 3 more seconds.. so performance matters over here. • **Dynamic wait**: ***Implicit wait & Explicit waits*** ◦ *Total timeout -10sec---> element found in 2 sec-----* *-->remaining 8 seconds will be ignored.*	@AMOL
Implicit wait	• It is also known as *global wait*. • It will be applied to all the webElements on the page by default. • We should not use implicit wait in our framework. • ***Specifies the amount of time the driver should wait when searching for an element if it is not immediately present.*** ☐ **Syntax:** • **driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));** • Before throwing an exception Implicit wait will let the selenium wait for timeout-10 seconds. If the required element is not found within timeout range(10 sec) then only after that it will throw an exception. NoSuchElementException: no such element: Unable to locate element	@Gagan @Pooja @Dhrumil @AMOL
Implicit Wait limitations	• It works only for webElements. • It doesn't work for non-web elements like **alerts**, **url**, **title** etc. • Thats why we avoid using implicitly wait in our framework.	@Pooja @AMOL

	• If the element is there but it's simply just hidden on the page, the implicit wait will not work and it will fail immediately. • The implicit wait applies to all the future commands utilized in your test. • So it slows down the testing of your automation script, as the driver has to wait for a specific amount of time.	
Implicit Wait can be overridden	• >We can override implicit wait by changing the value - • **driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));** ◦ To Nullify pass 0 • **driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(0));** ◦ Now new timeout is 0 seconds because the above implicit wait has been overridden here. ☐ ***This overriden of Implicit wait increases the script execution time, so better to avoid and try customization of wait.***	@Pooja @Dhrumil
ImplicitWait vs ExplicitWait	<table><tr><th>Implicit Waits</th><th>Explicit Waits</th></tr><tr><td>1. Implicit Wait time is applied to all the elements in the script</td><td>1. Explicit Wait time is applied only to those elements which are specified by the user</td></tr><tr><td>2. In Implicit Wait, we need not specify “ExpectedConditions” on the element to be located</td><td>2. In Explicit Wait, we need to specify “ExpectedConditions” on the element to be located</td></tr><tr><td>3. It is recommended to use when the elements are located with the time frame specified in implicit wait</td><td>3. It is recommended to use when the elements are taking a long time to load and also for verifying the property of the element like (visibilityOfElementLocated, elementToBeClickable,elementToBeSelected)</td></tr></table>	@AMOL
Explicit wait hierarchy		@AMOL
	## WebDriverWait	

Explicit Wait ***byPresence OfElement Located*** ***visibilityOf ElementLocate d***	☐ Creating the object of WebDriver wait class **By emailID= By.*id*("input-email");** **WebDriverWait wait= newWebDriverWait (*driver*, Duration. *of Seconds*(10));** ☐ Calling the method *until* of FluentWait class **wait.until(ExpectedConditions.*presenceOfElementLocated*(emailID));** ◦ FluentWait class is a parent of WebDriverWait class so it can access until method of its parent FluentWait. ◦ **This method will return WebElement.** ◦ This internally takes By locator as an input ☐ WebEelment ele **WebElement ele =wait.until (ExpectedConditions.*presenceOfElement Located* (emailID));** ☐ **Performing actions on WebElement ele** ◦ **ele**.sendKeys("dhondgeamol@gmail.com");	@AMOL
presenceOf ElementLocate d ***Utility***	/*** An expectation for checking that element is present on the dom of a page. * This does not necessarily mean that element is visible. * **@param** locator * **@param** timeout ***@param** pollingTime =500milliSec(0.5 sec) * **@return** WebElement */ **public** WebElement ***waitForElementPresent***(By locator, **int** timeout) { WebDriverWait wait = **new** WebDriverWait (driver, Duration.*ofSeconds*(timeout)); **return** wait.until(ExpectedConditions.***presenceOfElementLocated***	@AMOL

	(locator)); } 	
visibilityOf ElementLocated ***Utility***	/** More powerful for UI testing * An expectation for checking that an element is present on the DOM of a page and visible. * Visibility means that the element is not only displayed * but also has a height and width that is greater than 0. * **@param** locator * **@param** timeout ***@param** pollingTime =500milliSec(0.5 sec) * **@return** WebElement */ **public** WebElement ***waitForElementToBeVisible***(By locator, **int** timeout) { WebDriverWait wait = **new** WebDriverWait(driver, Duration.*ofSeconds*(timeout)); **return** wait.until(ExpectedConditions.***visibilityOfElementLocated***(locator)); }	@AMOL
What is Polling time ?	• In selenium polling time is how many times the selenium server will find the element on webPage with some time interval. • Default polling time = 500 ms If Duration of timeout is 20 sec, and polling time is 500ms, selenium server will check 20/0.5 = 40 times but with 0.5 seconds of interval.	@Gagan @Dhrumil
WebDriverWait with PollingTime/ Interval	• ***Default pollingTime:*** 0.5 seconds(500 milliSeconds) /** * An expectation for checking that element is present on the Dom of a page. * This does not necessarily mean that element is visible. * **@param** locator * **@param** timeout * **@param** pollingTime * **@return** WebElement */	@AMOL

	public WebElement waitForElementPresent(By locator, int timeout, long pollingTime) { WebDriverWait wait = new WebDriverWait(driver, Duration.*ofSeconds*(timeout),Duration.*ofMillis*(pollingTime)); return wait.until(ExpectedConditions.*presenceOfElementLocated*(locator)); }	
NOTE	☐ ExplicitWait can be used for WebElements as well as Non-WebElements such as popups,title,url. ☐ Whereas implicitlyWait is applied for only WebElements.	@AMOL
Selenium 4.x feature	• A small change selenium team has been introduced in WebDriverWait is "Duration" class. • All the Depricated methods have been removed from the selenium library.	@Gagan
Waiting for Alerts • ***alertIsPresent*** *Different Utilities through waitForAlert* • ***acceptAlert*** • ***dismissAlert*** • ***getAlertText***	public Alert ***waitForAlert***(int timeout) { WebDriverWait wait = new WebDriverWait(driver, Duration. *of Seconds*(timeout)); return wait.until(ExpectedConditions.*alertIsPresent*()); } --- public void ***acceptAlert***(int timeOut) { waitForAlert(timeOut).accept(); } --- public void ***dismissAlert***(int timeOut) { waitForAlert(timeOut).dismiss(); } --- public String ***getAlertText***(int timeOut) { return waitForAlert(timeOut).getText();	@AMOL

	}	

SeleniumSession -19 Date 15/03/22

Topic: WaitUtilities_URL_Title_Frame_WebElements

Java Class Files:

- **WaitForUrlConcept.java**
- **WaitForFrameConcept.java**
- **ElementToBeClickableConcept.java**
- **WaitForElements.java**

waitForTitle • ***waitForTitleContains*** • ***waitForTitleIs***	***/***** **** Wait for Title ******** **** waitForTitleContains*** **** waitForTitleIs*** ****/*** **public** String **waitForTitleContains**(**int** timeout, String titleFraction) { WebDriverWait wait= **new** WebDriverWait(driver,Duration.*ofSeconds*(timeout)); **if**(wait.until(ExpectedConditions.*titleContains*(titleFraction))) { **return** driver.getTitle(); } **return null**; } --- /** * An expectation for checking the title of a page. * @param timeOut * @param titleValue	@AMOL

	* @return Title */ public String waitForTitleIs(int timeout, String title) { WebDriverWait wait= new WebDriverWait(driver,Duration.*ofSeconds*(timeout)); if(wait.until(ExpectedConditions.*titleIs*(title))) { return driver.getTitle(); } return null; }	
waitForFrame • ***waitForFrameByLocator*** • ***waitForFrameByIndex*** • ***waitForFrameByString*** • ***waitForFrameByElement***	/** * Wait for frame by * By Locator * By Index * By String * By WebElement */ public WebDriver waitForFrameByLocator(int timeout, By locator) { WebDriverWait wait= new WebDriverWait(driver,Duration.*ofSeconds*(timeout)); return wait.until(ExpectedConditions.*frameToBeAvailableAndSwitchToIt*(locator)); } ---------- -- public WebDriver waitForFrameByIndex(int timeout, int frameIndex) { WebDriverWait wait= new WebDriverWait(driver,Duration.*ofSeconds*(timeout));	@AMOL

	return wait.until(ExpectedConditions.*frameToBeAvailableAndSwitchToIt*(frameIndex)); } ---------- -- public WebDriver waitForFrameByString(int timeout, String frameLocator) { WebDriverWait wait= new WebDriverWait(driver,Duration.*ofSeconds*(timeout)); return wait.until(ExpectedConditions.*frameToBeAvailableAndSwitchToIt*(frameLocator)); } ---------- -- public WebDriver waitForFrameByWebElement(int timeout, WebElement frameElement) { WebDriverWait wait= new WebDriverWait(driver,Duration.*ofSeconds*(timeout)); return wait.until(ExpectedConditions.*frameToBeAvailableAndSwitchToIt*(frameElement)); }	
waitForElement	*/** * An expectation for checking an element is visible and enabled * such that you can click it. * @param locator * @param timeOut */ public void clickWhenReady(By locator, int timeOut) { WebDriverWait wait= new WebDriverWait(driver,Duration.*ofSeconds*(timeOut)); wait.until(ExpectedConditions.*elementToBeClickable*(locator)).click	@AMOL

	(); } ---------- -- public void clickElementWhenReady(By locator, int timeOut) { WebDriverWait wait= new WebDriverWait(driver,Duration.*ofSeconds*(timeOut)); wait.until(ExpectedConditions.*elementToBeClickable*(getElement(locator))).click(); }	
waitForElements	*/*** ** An expectation for checking that all elements present on the web page that match the locator are visible.* ** Visibility means that the elements are not only displayed but also have a height and width that is greater than 0.* ** @param locator* ** @param timeOut* ** @return* **/* public List<WebElement> waitForElementsToBeVisible(By locator,int timeOut) { WebDriverWait wait= new WebDriverWait(driver,Duration.*ofSeconds*(timeOut)); List<WebElement> footerList=wait.until(ExpectedConditions.*visibilityOfAllElementsLocatedBy*(locator)); return footerList; } ---------- -- public void printAllWebElementsText(By locator, int timeOut) {	@AMOL

```
        List<WebElement> eleList= waitForElementsToBeVisible(
locator,timeOut);
                for(WebElement e:eleList) {
                System.out.println(e.getText());
                }

    }
-------------------------------------------------------------------------
--

    public  List<String> getElementsTextListWithWait(By locator,
int timeOut) {
        List<WebElement> eleList= waitForElementsToBeVisible(
locator,timeOut);

        List<String> eleTextList=new ArrayList<String>();

                for(WebElement e:eleList) {
                String text=e.getText();
                eleTextList.add(text);

                }
                return eleTextList;

    }
```

SeleniumSession -20 Date 16/03/22

Topic:

Custom Wait	`public class CustomWait {` `static WebDriver driver;` `public static void main(String[] args) {` `WebDriverManager.chromedriver().setup();` `driver= new ChromeDriver();`	@AMOL

```
		driver.get("https://demo.opencart.com
/index.php?route=account/login");
		By emailID= By.id("input-email12");
		retryingElement(emailID,20,2000);

	}
	public static WebElement getElement(By locator) {

		return driver.findElement(locator);
	}

	public static WebElement retryingElement(By locator, int
timeOut, int intervalTime) {

		WebElement element=null;
		int attempts=0;

		while(attempts<timeOut) {

			try {

				element=getElement(locator);
				break;

			}
			catch(NoSuchElementException e) {
				System.out.println("Element is not found in
attempt:"+attempts+":"+locator);

				try{
					Thread.sleep(intervalTime); //custom
interval time
				}
				catch(InterruptedException e1) {
					e1.printStackTrace();
				}

			}
			attempts++;
		}
```

	if(element == null) { try { throw new Exception("ELEMENT_NOT_FOUND_EXCEPTION"); } catch(Exception e) { System.*out*.println("Element is not found exception....tried for :"+timeOut+":"+"secs"+ "with an interval of"+intervalTime+"ms"); } } return element; } }	
waitForPageLoad using JS Executor	public static void waitForPageLoad(int timeOut) { long endTime= System.*currentTimeMillis*()+timeOut; while(System.*currentTimeMillis*()<endTime) { JavascriptExecutor js=(JavascriptExecutor) *driver*; String state=js.executeScript("return document.readyState").toString(); System.*out*.println(state); if(state.equals("complete")) { break; } } }	@AMOL
	FluentWait	
FluentWait vs WebDriverWait	• **Class WebDriverWait is a child of Class FluentWait and grand child of interface Wait.** • In short, Class FluentWait implements Wait interface and WebDriverWait extends FluentWait.	@AMOL

	• So there are many things common in WebDriverWait and FluentWait instances such as :- a. ***You can set polling interval in both instances.*** b. ***You can ignore any exceptions in both instances.*** c. ***You can use ExpectedConditions methods in both instances.***	
waitForElementPresentWithFluentWait	**public static** WebElement waitForElementPresentWithFluentWait (By locator,**int** timeOut, **int** pollingTime) { Wait<WebDriver> wait= **new** FluentWait<WebDriver>(*driver*).withTimeout(Duration.*ofSeconds*(timeOut)) .pollingEvery(Duration.*ofSeconds*(pollingTime)) .ignoring(NoSuchElementException.**class**,ElementNotInteractableException.**class**); **return** wait.until(ExpectedConditions.*presenceOfElementLocated*(locator)); } 	@AMOL
waitForElementVisibleWithFluentWait	**public static** WebElement waitForElementVisibleWithFluentWait(By locator,**int** timeOut, **int** pollingTime) { Wait<WebDriver> wait= **new** FluentWait<WebDriver>(*driver*) .withTimeout(Duration.*ofSeconds*(timeOut)) .pollingEvery(Duration.*ofSeconds*(pollingTime)) .ignoring(NoSuchElementException.**class**,ElementNotInteractableException.**class**); **return** wait.until(ExpectedConditions.*visibilityOfElementLocated*(locator)); 	@AMOL

	} 	
StaleElement Exception	• Stale Element means an old element or no longer available element. • Assume there is an element that is found on a web page referenced as a WebElement in WebDriver. • If the DOM changes then the WebElement goes stale. If we try to interact with an element which is staled then the **StaleElementReferenceException** is thrown. ☐ **Causes** : 1. ***The referenced web element has been deleted completely.*** 2. ***The referenced element is no longer attached to the DOM.*** ☐ **Solutions to avoid SEE** **Using POM** • We can handle Stale Element Reference Exception by using POM. • We could avoid StaleElementException using POM. In POM, we use getElements() method which loads the element but it won't initialise elements. getElements() takes latest address. It initializes during run time when we try to perform any action on an element.	@AMOL

SeleniumSession -21 Date 17/03/22

Topic: Pseudo_Element_Calendar_handling_Selenium_Quiz

Pseudo Element Handler **Scenario: To check which elements are mandotary on the UI.**	public class PseudoElementHandler { static WebDriver driver; public static void main(String[] args) { WebDriverManager.chromedriver().setup(); driver = new ChromeDriver(); driver.get("https://demo.opencart.com	@Dhrumil

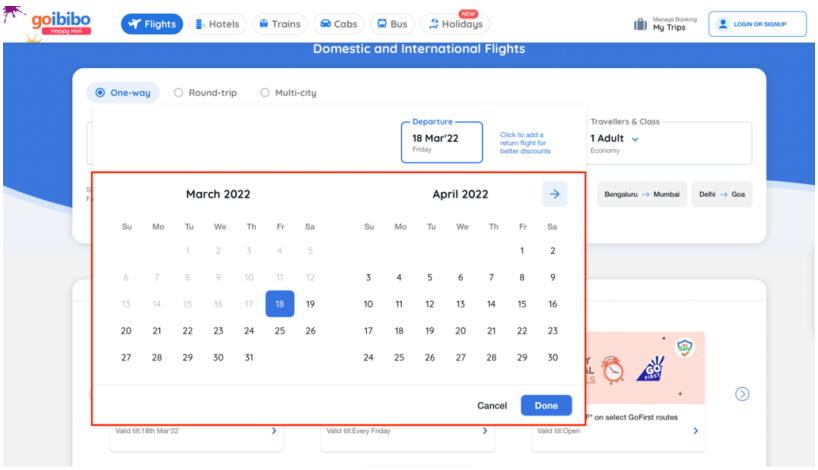
	/index.php?route=account/register"); JavascriptExecutor js =(JavascriptExecutor)driver; String script = "return window.getComputedStyle(document.querySelector(\"label[for='input-firstname']\"), '::before').getPropertyValue('content')"; String mandotary_text = js.executeScript(script).toString(); System.out.println(mandotary_text); } }	
CalenderHandling	**public static void** selectFutureDate(String expMonthYear , String dateNum) { **if**(Integer.*parseInt*(dateNum)>31){ System.***out***.println("please pass the correct date"); **return**; } **if**(expMonthYear.contains("February") && Integer.*parseInt*(dateNum)>29) { System.***out***.println("please pass the correct date"); **return**; } **if**((expMonthYear.contains("April") || expMonthYear.contains("June")||expMonthYear.contains("September")||expMonthYear.contains("November")) && Integer.*parseInt*(dateNum)>30) { System.***out***.println("please pass the correct date"); **return**;	@AMOL

```
        }


        String
currentMonthYear=driver.findElement(By.cssSelector(".ui-
datepicker-title")).getText();


while(!expMonthYear.equalsIgnoreCase(currentMonthYear)) {


driver.findElement(By.xpath("//a[@title='Next']")).click();

currentMonthYear=driver.findElement(By.cssSelector(".ui-
datepicker-title")).getText();
        }

        selectDate(dateNum);

    }

    public static void selectDate(String dateNum) {


driver.findElement(By.xpath("//a[text()='"+dateNum+"']")).click();
    }
```

Assignment: Goibibo Calendar

https://www.goibibo.com

@AMOL




```
public class CalenderGolbibo {
```

```
	static WebDriver driver;
	public static void main(String[] args) throws
InterruptedException {

		WebDriverManager.chromedriver().setup();
		driver= new ChromeDriver();

		driver.get("https://www.goibibo.com/");

		driver.findElement(By.xpath("(//div[@class='sc-bkkeKt
gAqCbJ fswFld ']}[3]")).click();



		selectFutureDate("March 2023","23");

		Thread.sleep(3000);
		driver.quit();
	}
	public static void selectFutureDate(String expMonthYear,
String date) throws InterruptedException {

		if(Integer.parseInt(date)>31) {

			System.out.println("please pass the right date");
			return;
		}

		if(expMonthYear.contains("February") &&
Integer.parseInt(date)>29) {

			System.out.println("please pass the right date");
			return;
		}


if((expMonthYear.contains("April")||expMonthYear.contains("June"
)||expMonthYear.contains("August")||expMonthYear.contains("No
vember")) && Integer.parseInt(date)>30){

			System.out.println("please pass the right date");
			return;
```

```
                }

    List<String> currentMonthYearList= new ArrayList<String>();

List<WebElement>
CurentMonthYear=driver.findElements(By.cssSelector(".DayPicke
r-Caption"));

        for(WebElement e:CurentMonthYear) {
            String txt=e.getText();
            currentMonthYearList.add(txt);

        }


        while(!currentMonthYearList.contains(expMonthYear)) {

            driver.findElement(By.xpath("//span[@aria-
label='Next Month']")).click();

            currentMonthYearList= new ArrayList<String>();


CurentMonthYear=driver.findElements(By.cssSelector(".DayPicke
r-Caption"));

            for(WebElement e:CurentMonthYear) {
                String txt=e.getText();
                currentMonthYearList.add(txt);

            }
        }
        Thread.sleep(3000);

        if(currentMonthYearList.get(0).contains(expMonthYear))
{

            selectDate1(date);
        }
        else
```

```
            {
                selectDate2(date);
            }

driver.findElement(By.xpath("//span[contains(text(),'Done')]")).clic
k();



    }

//Selecting the date from left side calendar table
    public static void selectDate1(String date) {

        driver.findElement(By.xpath("(//div[@class='DayPicker-
Day']/p[text()='"+date+"'])[1]")).click();

    }

//Selecting the date from right side calendar table
    public static void selectDate2(String date) {

        driver.findElement(By.xpath("(//div[@class='DayPicker-
Day']/p[text()='"+date+"'])[2]")).click();



    }

}
```

****

TestNG -01 Date 21/03/22

Topic:TestNGAnnotations | Selenium_Integration | BeforeTest_vs_BeforeMethod |

Java Class Files:

- AppTest.java
- AmazonTest.java
- AmazonTestBM.java

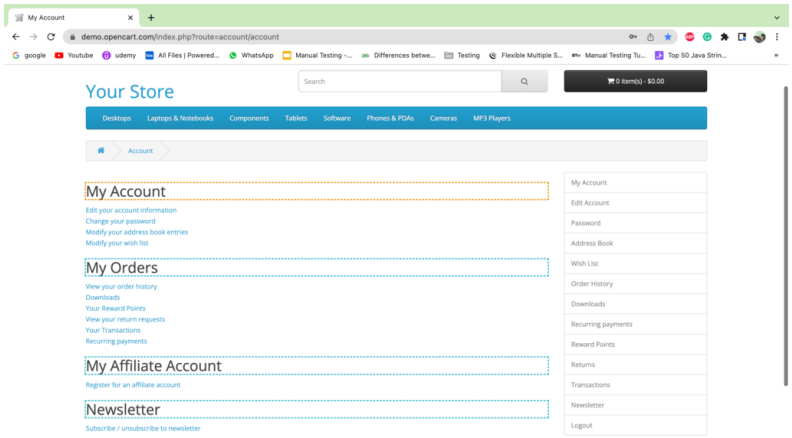
- BaseTest.java
- DemoCartAppTest.java

What is TestNG? **Official Documentation:** https://testng.org/doc/documentation-main.html	TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing (testing a class in isolation of the others) to integration testing (testing entire systems made of several classes, several packages and even several external frameworks, such as application servers).	@Dhrumil
Why we required TestNG?	Till the time, we are writing our logic inside the main method of class and to run that specific functionality we are running that class. Now, assume that we have to automate 100 test scenarios and we developed 100 test scripts using our traditional style i.e A class with main method. Few Problems: • If you want to run all test scripts at once, you need to create a separate runner class where you will call main method of each script class. But it will be very difficult to continue or stop script execution when some scripts fails. • Every scenarios can have same or different prerequisites and post requisites conditions. You may need to write duplicate codes repeatedly. • Establishing relationship or dependencies among test scripts will not be easy task. • If new test scripts need to be added, you might fed up in maintenance and modifications. Do you think its feasible with increasing of requirements and framework development? Answer is NO. Here TestNG is comes into picture.	@Dhrumil
Eclipse Plugin for TestNG: https://github.com	To install it: • Click "Help -> Install New Software..." on top level menu • Paste the url https://testng.org/testng-eclipse-update-site	@Dhrumil

/cbeust/testng-eclipse	to Work with: text field and press enter. • Select the plugins • Click "Next" button and accept the license to complete the installation. • Restart Eclipse	
TestNG configuration in pom.xml https://mvnrepository.com/artifact/org.testng/testng/6.14.3	<!-- https://mvnrepository.com/artifact/org.testng/testng --> <dependency> <groupId>org.testng</groupId> <artifactId>testng</artifactId> <version>6.14.3</version> </dependency>	@Dhrumil
What are the TestNG annotations?	• TestNG Annotations are used to control the next method to be executed in the test script. • TestNG annotations are defined before every method in the test code. • In case any method is not prefixed with annotations, it will be ignored and not be executed as part of the test code. • To define them, methods need to be simply annotated with '@Test'.	@Dhrumil
Different Types of Test Annotations	• **@BeforeMethod:** This will be executed before every @test annotated method. • **@AfterMethod:** This will be executed after every @test annotated method. • **@BeforeClass:** This will be executed before first @Test method execution. It will be executed one only time throughout the test case. • **@AfterClass:** This will be executed after all test methods in the current class have been run • **@BeforeTest:** This will be executed before the first @Test annotated method. It can be executed multiple times before the test case. • **@AfterTest:** A method with this annotation will be executed when all @Test annotated methods complete the execution of those classes inside the <test> tag in the TestNG.xml file. • **@BeforeSuite:** It will run only once, before all tests in the suite are executed. • **@AfterSuite:** A method with this annotation will run once after the execution of all tests in the suite is complete.	@Dhrumil
Imp: Sequence of execution for TestNG annotations **Note:** Only single time: @BeforeSuite, @BeforeTest,	• @BeforeSuite • @BeforeTest • @BeforeClass • @BeforeMethod • @Test • @AfterMethod • @AfterClass • @AfterTest	@Dhrumil

@BeforeClass and @BeforeMethod allowed. @Test can be used multiple times.	• @AfterSuite	
Example: AppTest using different TestNG annotations Test cases are executed in Alphabetical order. **O/P:** BS--Connect with DB BT -- Create User BC --- Launch Browser **BM --- Login to App** Test of Header **AM - User is logged out** **BM --- Login to App** User is Logged in **AM - User is logged out**	public class AppTest { // Global Pre-Conditions // Pre-conditions // test steps + Actual vs Expected Result // post steps @BeforeSuite public void connectWithDB() { System.out.println("BS--Connect with DB"); } @BeforeTest public void createUser() { System.out.println("BT -- Create User"); } @BeforeClass public void launchBrowser() { System.out.println("BC --- Launch Browser"); } @BeforeMethod public void login() { System.out.println("BM --- Login to App"); } @Test public void titleTest() { System.out.println("Test of Title"); } @Test public void headerTest() {	@Dhrumil

BM --- Login to App Test of Title **AM - User is logged out** AC-- Close Browser AT -- Delete User AS -- Close DB Connection	System.out.println("Test of Header"); } @Test public void isUserLoggedInTest() { System.out.println("User is Logged in"); } @AfterMethod public void logout() { System.out.println("AM - User is logged out"); } @AfterClass public void closeBrowser() { System.out.println("AC-- Close Browser"); } @AfterTest public void deleteUser() { System.out.println("AT -- Delete User"); } @AfterSuite public void closeDBConnection() { System.out.println("AS -- Close DB Connection"); }	
BeforeMethod and AfterMethod annotations creating a pair with Test Annotations.	Executed each time with Test Annotations.	@Dhrumil
BeforeMethod & AfterMethod	We can use it in **Regression Test**, as number of testcases are more , so to avoid blockage in Mid of test execution, this pair of annotations we can use it will give maximum percentage of execution	@Jeetendra

BeforeTest & AfterTest	We can use it in **Sanity Test**, as we can choose or customise our test cases in limited numbers (specific functions), to get quick execution. here if in middle of executions if there any blockage occurs then further execution will be stopped	@Jeetendra
How to define Priority of test inside test class using @Test Annotations?	@Test(priority = 1) @Test(priority = 2) @Test(priority = 3)	@Dhrumil
Assignment 1. Login to the MyAccount 2. Validate if the required element is displayed or not using assertions. 3. Capture the required header's text & validate by matching the actual result with expected result. 4. Logout	 **public class *BaseTest*** { WebDriver driver; @BeforeMethod **public void** setup() { WebDriverManager.*chromedriver*().setup(); driver= **new** ChromeDriver(); driver.manage().deleteAllCookies(); driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(10)); driver.get("https://demo.opencart.com/index.php?route=account/login"); driver.findElement(By.*id*("input-	@AMOL

```
email")).sendKeys("dhondgeamol@gmail.com");
		driver.findElement(By.id("input-
password")).sendKeys("Btgbygai456");

driver.findElement(By.xpath("//input[@type='submit']")).click();
	}

	@AfterMethod
	public void tearDown() {

		driver.findElement(By.linkText("Logout")).click();
		driver.quit();

}
	}

-----------------------------------------------------------------------
---------------------
public class DemoCartPage_2 extends BaseTest {



	@Test
	public void myAccountHeaderTest() {

   WebElement
myAccount=driver.findElement(By.xpath("//h2[contains(text(),'
My Account')]"));

		Assert.assertTrue(myAccount.getText().contains("My
Account"));
		Assert.assertTrue(myAccount.isDisplayed());

	}

	@Test
	public void myOrderstHeaderTest() {

		WebElement
myOrders=driver.findElement(By.xpath("//h2[contains(text(),'M
y Orders')]"));
```

@AMOL

```
        Assert.assertTrue(myOrders.getText().contains("My Orders"));
        Assert.assertTrue(myOrders.isDisplayed());

    }
    @Test
    public void myAffiliateAccountHeaderTest() {

        WebElement MyAffiliateAccount=driver.findElement(By.xpath("//h2[contains(text(),'My Affiliate Account')]"));


Assert.assertTrue(MyAffiliateAccount.getText().contains("My Affiliate Account"));

Assert.assertTrue(MyAffiliateAccount.isDisplayed());

    }

    @Test
    public void newsletterHeaderTest() {

        WebElement Newsletter=driver.findElement(By.xpath("//h2[contains(text(),'Newsletter')]"));


Assert.assertTrue(Newsletter.getText().contains("Newsletter"));
        Assert.assertTrue(Newsletter.isDisplayed());

    }
```

Output

```
}


PASSED: myAccountHeaderTest
PASSED: myAffiliateAccountHeaderTest
PASSED: myOrderstHeaderTest
```

	PASSED: newsletterHeaderTest ========== Default test Tests run: 4, Failures: 0, Skips: 0 ========== ========== Default suite Total tests run: 4, Failures: 0, Skips: 0 ==========	

TestNG -02 Date 22/03/22

Topic: Different_TestNG_Features | TestNG_XML | TestRunner

Java Class Files:

- AmazonTest.java
- BaseTest.java
- DemoCartAppTest.java
- InvocationCountConcept.java
- ExpectedExceptionConcept.java
- DependsOnMethodConcept.java
- PriorityTest.java
- Test Runner - testng.xml

Priority feature	• By default, when priority is not defined- execution happened in alphabetical order for different test methods. • priority = 0 also can be given to @Test • priority = 0 and priority = -1, -1 method will be executed before 0 priority method. • If test priority is not defined while, running multiple test cases, TestNG assigns all @Test a priority as zero(0) and execution happened in alphabetical manner. • If two tests having same priority, then execution happened in alphabetical manner with respect to test method name. • When there is a combination of priority and non-priority	@Dhrumil

	based tests, non-priority based test case execution happens first(Alphabetically) before priority based test case execution.	
InvocationCount feature *The number of times this method should be invoked.* *Above feature is useful while performing API test automation.* *We can club multiple feature together. e.g. - priority and invocationCount.* ***Output***	• TestNG supports multi-invocation of a test method, i.e., a @Test method can be invoked multiple times sequentially or in parallel. • If we want to run single @Test 10 times at a single thread, then **invocationCount** can be used. • To invoke a method multiple times, the keyword **invocationCount** = **<int>** is required. • **Example:** `public class InvocationCountConcept {` `@Test(invocationCount=10)` `public void loginTest() {` `System.out.println("loginTest");` `}` `}` loginTest loginTest loginTest loginTest loginTest loginTest loginTest loginTest loginTest loginTest PASSED: loginTest PASSED: loginTest PASSED: loginTest PASSED: loginTest PASSED: loginTest PASSED: loginTest PASSED: loginTest PASSED: loginTest	@Dhrumil @AMOL

	PASSED: loginTest PASSED: loginTest === Default test Tests run: 10, Failures: 0, Skips: 0 === === Default suite Total tests run: 10, Failures: 0, Skips: 0 ===	
***dependsOnMethods** - Single Dependent Test Methods in TestNG* *Note: Try to avoid such implementation while implementing framework and test cases should be work independent.*	• A single dependent test in TestNG is declared when a single test depends on another test. **Example:** public class DependsOnTest { @Test (dependsOnMethods = { "OpenBrowser" }) public void SignIn() { System.out.println("User has signed in successfully"); } @Test public void OpenBrowser() { System.out.println("The browser is opened"); } @Test (dependsOnMethods = { "SignIn" }) public void LogOut() { System.out.println("The user logged out successfully"); } }	@Dhrumil
dependsOnMethods - Multiple Dependent Tests in TestNG	public class DependsOnTest { @Test public void OpenBrowser() {	@Dhrumil

	System.out.println("Opening The Browser"); } @Test(**dependsOnMethods = { "SignIn", "OpenBrowser" }**) public void LogOut() { System.out.println("Logging Out"); } @Test public void SignIn() { System.out.println("Signing In"); } }	
expectedException *feature*	• *Within @Test annotation, TestNG supports multiple exceptions being provided for verification using attribute* ***expectedExceptions****. If the exception thrown by the test is not part of the user entered list of exceptions, the test will be marked as failed.* **public class** ExpectedException { String name; @Test(expectedExceptions= {ArithmeticException.**class**,NullPointerException.**class**}) **public void** loginTest() { System.***out***.println("login is successful"); **int** i=9/0; ExpectedException ep= **new** ExpectedException(); ep=**null**; ep.name="amol"; } } • *Note: The test will failed if the exception thrown by the method does not match the exception list provided in the* ***expectedExceptions*** *list.*	@Dhrumil @AMOL
testng.xml purpose	TestNG.xml file is a configuration file that helps in organizing our	@Dhrumil

	tests. It allows testers to create and handle multiple test classes, define test suites and tests. It makes a tester's job easier by controlling the execution of tests by putting all the test cases together and run it under one XML file.	
How to create and use testng.xml	• Refer:https://www.softwaretestinghelp.com/testng-example-to-create-testng-xml/#:~:text=xml%3F-,TestNG.,it%20under%20one%20XML%20file.	@Dhrumil
TestNG.xml demo code	<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd"> <suite name="Regression Test Suite" verbose="4"> <test name="Amazon Test"> <classes> <class name="***testNG_Sessions.AmazonTestBM***" /> </classes> </test> <test name="Demo Cart Test"> <classes> <class name="***testNG_Sessions.DemoCartPage_2***" /> </classes> </test> </suite>	@AMOL
Disable Test Case *By default, all the test*	• **@Test(enabled = false)**	@Dhrumil

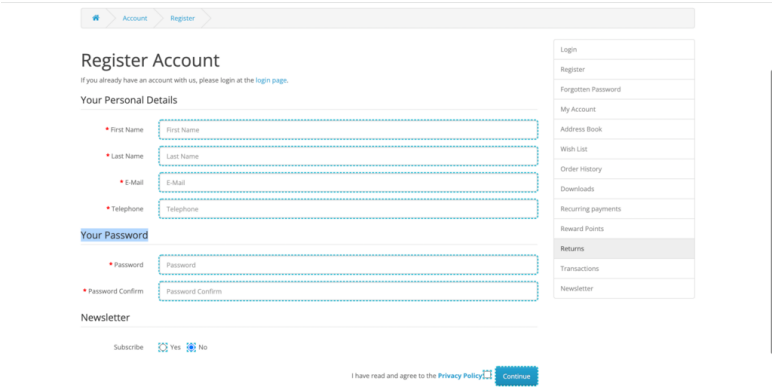
cases are enabled = true.		
Run specific test case	• Right Click on Test > Run As > TestNG test	@Dhrumil
Imp Files to consider under test-output folder	1. index.html 2. emailable-report.html 3. test-result.xml 4. testng-failed.xml	@Dhrumil
Description *usage for testcase*	• *This description is printed along with the testname in console as well inside index.html reports.* --- • *titleTest* • *verifying title test of demo cart application....*Examples: ◦ @Test(description = "verifying title test of demo cart application....") ◦ @Test(enabled = true, description = "url test....")	@Dhrumil
Note: In testng.xml, can't give the same test name to two test blocks.		@Dhrumil
verbose usage in testng.xml Ideal case : 4 or 5. <suite name="Regression Test Suite" verbose="4">	minimum = 1 maximum = 10 mainly used for the TestNG logs in descriptive manner in the console window. · Default value of Verbose is 1	@Dhrumil

****

TestNG -03 Date 23/03/22

Topic:

ASSIGNMENT




```
public class RegisterAccount {

	private WebDriver driver;
	private ElementUtil eleUtil;

// 1* Private By locators: to achieve encapsulation

	private By header=By.xpath("//div[@id='content']/h1");
	private By personalDetails=By.xpath("//legend[text()='Your Personal Details']");
	private By firstName=By.id("input-firstname");
	private By lastName=By.id("input-lastname");
	private By email=By.id("input-email");
	private By telephone=By.id("input-telephone");
	private By yourPassword=By.xpath("//legend[text()='Your Password']");
	private By password=By.xpath("//input[@id='input-password']");
	private By passwordConfirm=By.xpath("//input[@id='input-confirm']");
	private By checkBox=By.xpath("//input[@type='checkbox']");
	private By button=By.xpath("//input[@type='submit']");



// 2* Initialize the WebDriver using constructor: public pageClass constructor

	public RegisterAccount(WebDriver driver) {
		this.driver=driver;
```

@AMOL

```
        eleUtil=new ElementUtil(driver);
    }


// 3* public Page Actions:

    public String registerAccountPageTitle() {
        return
eleUtil.waitForTitleContains(Constants.DEFAULT_TIME_OUT
, Constants.REGISTER_ACCOUNT_PAGE_TITLE);

    }

    public boolean isRegisterAccountHeaderExist() {
        return eleUtil.doIsDisplayed(header);
    }

    public boolean isPersonalDetailsHeaderExist() {
        return
eleUtil.waitForElementToBeVisible(personalDetails,
5).getText().contains(Constants.PERSONAL_DETAILS_HEAD
ER);
    }

    public void fillPersonalDetails(String fName,String
lName ,String mail,String phoneNumber) {

        eleUtil.doSendkeys(firstName, fName);
        eleUtil.doSendkeys(lastName, lName);
        eleUtil.doSendkeys(email, mail);
        eleUtil.doSendkeys(telephone, phoneNumber);

    }
    public boolean isYourPasswordHeaderExist() {
        return
eleUtil.waitForElementToBeVisible(yourPassword,
5).getText().contains(Constants.YOUR_PASSWORD_HEADE
R);
    }

    public void fillPassword(String pwd, String pwdConfirm)
```

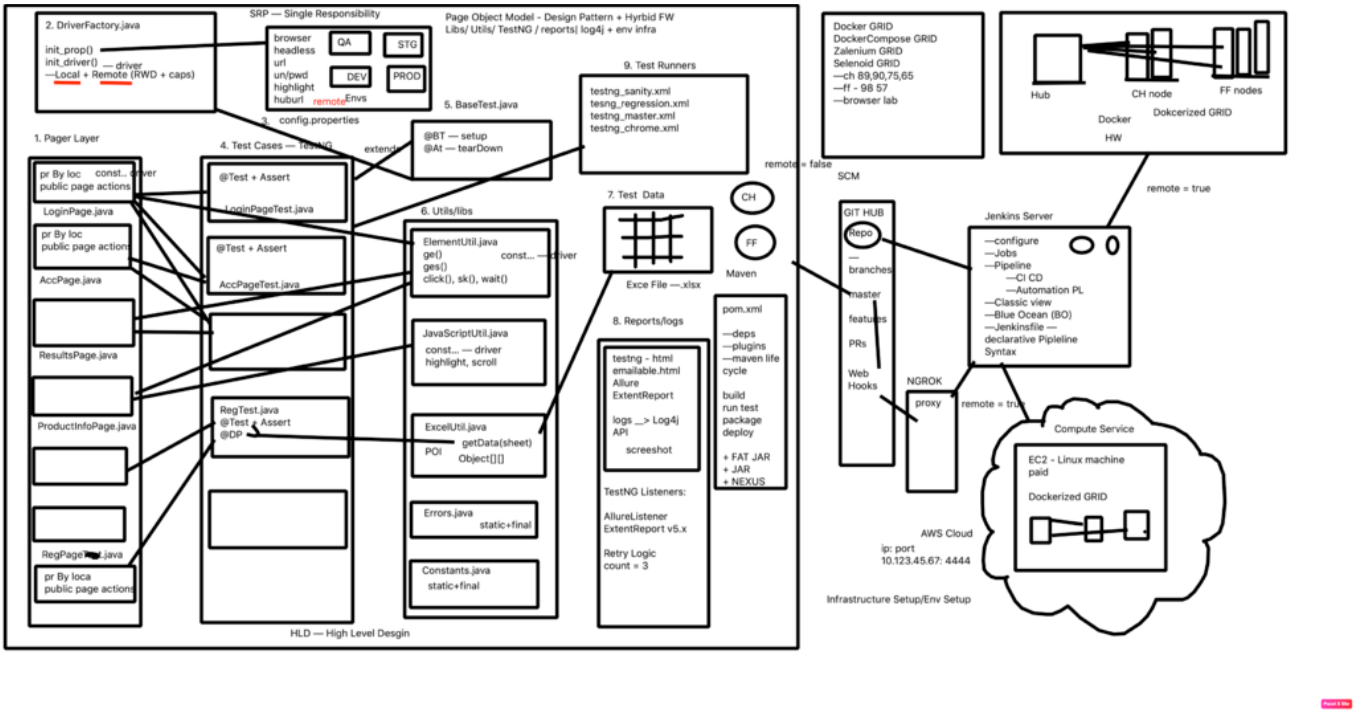
```
{

		eleUtil.doSendkeys(password, pwd);
		eleUtil.doSendkeys(passwordConfirm, pwdConfirm);
		eleUtil.doClick(checkBox);
		eleUtil.doClick(button);
	}
}
```

@AMOL

```
-----------------------------------------------------------------------
----------------------

public class RegisterAccountPageTest extends BaseTest {


	@BeforeClass
	public void RegisterAccountPageSetUp() {
		regAccount=loginPage.navigateToRegisterPage();
	}

	@Test
	public void registerAccountPageTitleTest() {

		String
actTitle=regAccount.registerAccountPageTitle();
		System.out.println("Register Account page title is
:"+actTitle);
		Assert.assertEquals(actTitle,
Constants.REGISTER_ACCOUNT_PAGE_TITLE);
	}

	@Test
	public void registerAccountHeaderTest() {

Assert.assertTrue(regAccount.isRegisterAccountHeaderExist()
);
	}

	@Test
	public void personalDetailsHeaderTest() {

Assert.assertTrue(regAccount.isPersonalDetailsHeaderExist());
```

```
        }

        @Test
        public void registerAccountTest() {

regAccount.fillPersonalDetails(prop.getProperty("firstName"),
prop.getProperty("lastName"), prop.getProperty("email"),
prop.getProperty("telephone"));

regAccount.fillPassword(prop.getProperty("password"),
prop.getProperty("password"));
        }
}
```

Best Practices for Framework Implementation



1. TestClass - Make sure to use Assertion inside the TestClass, and No driver reference should be present in TestClass.

2. PageClass - Driver reference should be present inside page class and all the methods which we are going to use inside TestClass should be defined in PageClass.
3. 3 Things to keep in mind while implementing respective PageClass
    i. Initialize private WebDriver driver
    ii. Maintain all the locator of the page and make it Private.
    iii. Create public page constructor and page actions.
4. BaseTest is very Imp with reference to maintain BeforeTest(setup) and AfterTest(tearDown) actions.
5. All the desired results which is known already should be maintained inside the Constants.
6. Use ElementUtil methods as much as possible while creating Framework, this will give you better readability and clean code visual.

Soft Assert vs Hard Assert

Headless Concept: @Dhrumil

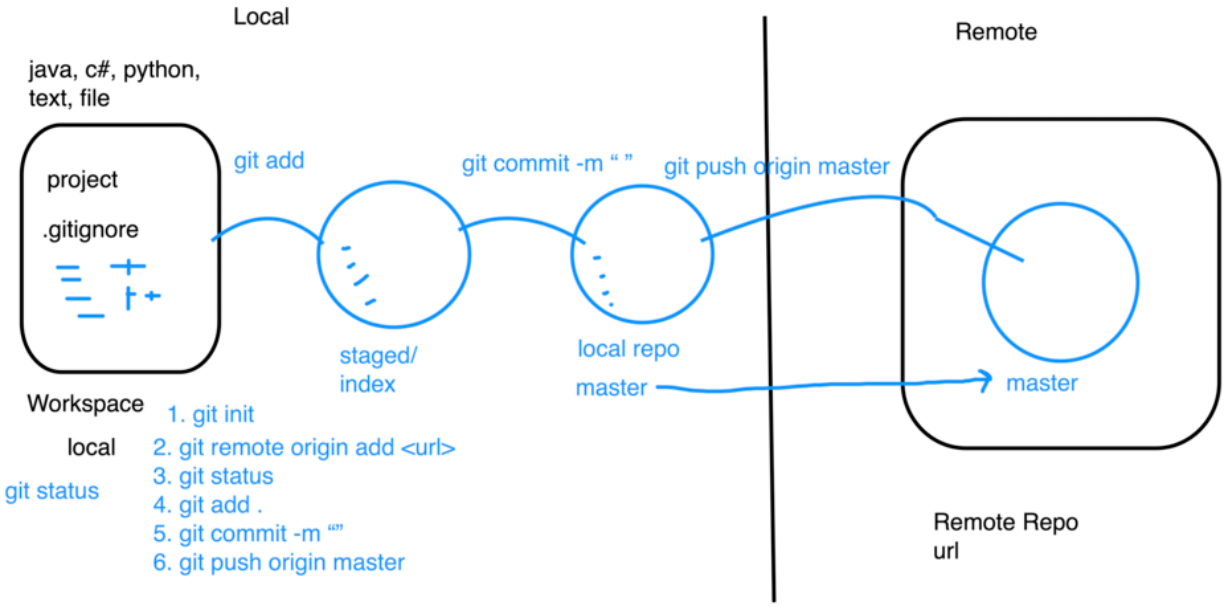
ChromeOptions class is used for execution in headless browser and Incognito mode.

```
ChromeOptions co = new ChromeOptions();
//co.setHeadless(true);
co.addArguments("--headless");
WebDriver driver = new ChromeDriver(co);
```

Allure Report Integration: @Dhrumil

1. Install apectj dependency inside pom.xml
2. Configure maven-surefire and maven-compiler plugin inside pom.xml
3. Imp link to follow: https://docs.qameta.io/allure/#_testng
4. Create TestAllureListener file inside listeners package.
5. Add reference of the TestAllureListerner in the testng.xml file
6. Add different annotations provided by this adapter plugin(Non-mandatory)
    a. Description
    b. Severity
    c. Steps
    d. Epic
    e. Story
7. Install Allure server configuration through following below link: https://docs.qameta.io/allure/#_installing_a_commandline
8. Execute the tests via testng.xml
9. Go to project directory path in command line and execute below command to generate reports from the allure-results folder
    a. allure serve allure-results

Git Architecture:

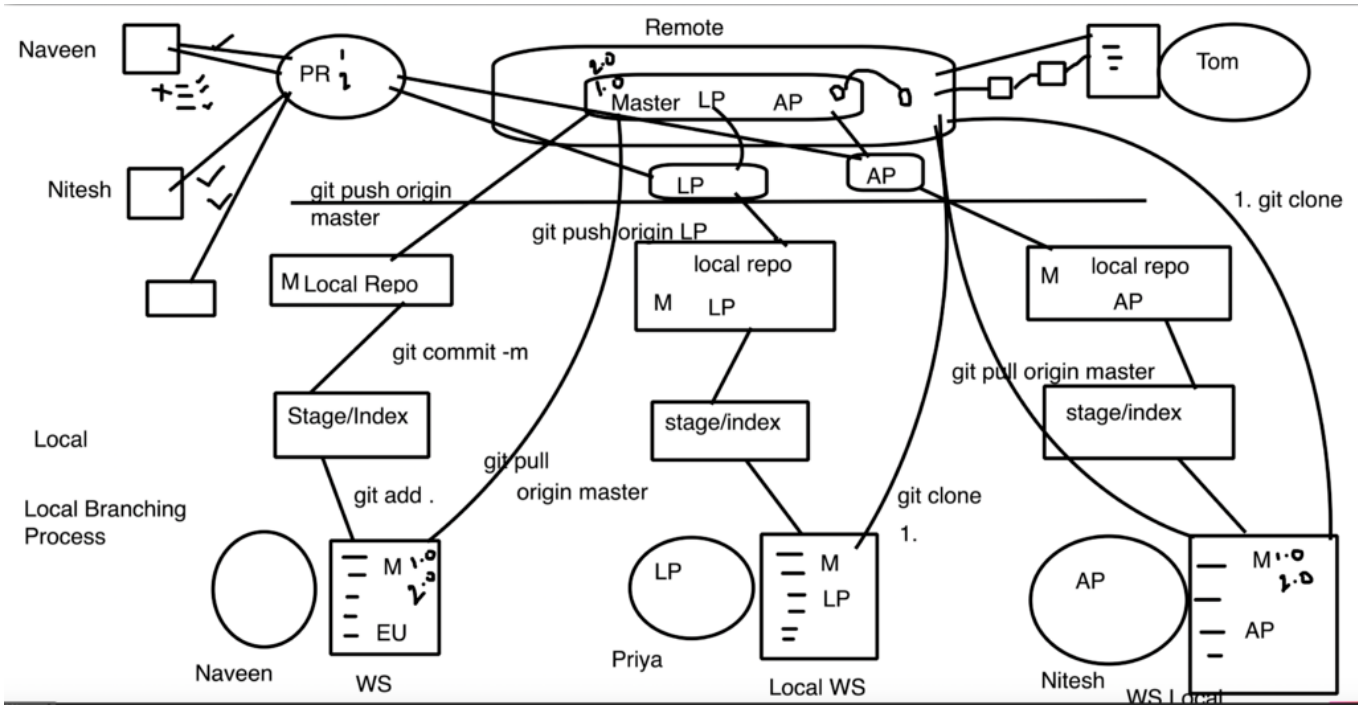


Git Important Commands:

git init	This command is to initialize your project	@Gagan
git remote origin add <git url>	This is to connect your local project to remote git repo.	@Gagan
git status	-This command is to check the current status of the activities on project (if you have made some changes and the changes are needs to be commit or not) - It is a good practice to use git status command every time.	@Gagan
git commit -m"reason to commit"	This command is used to commit the changes on local master branch.	@Gagan

git push origin master	This command is used to push all the changes to Remote master branch	@Gagan
git branch	This command is used to check on which branch the cursor is.. Like Master branch/ Login page brach (* master)	@Gagan

GIT - Local Branching Process : @Naveen AutomationLabs



GIT Real Time Use Cases : @Naveen AutomationLabs

//1. first commit:

1. go to project directory
2. git init

Initialized empty Git repository in /Users/naveenautomationlabs/Documents/workspace /Jan2022POMSeries/.git/

3. one .git (hidden dir) will be created
4. git remote add origin <repo url>
5. git status
6. add .gitignore file
7. git add .
8. git commit -m "reason"

9. git push origin master
10. go to the repo url and check the code at remote side

//2. new team member:
1. get the repo url
2. create a directory in your local : C/D drive
3. clone the repo in your local: git clone <repo url>
4. import this project into Eclipse/IntelliJ
5. start looking into the project: run/debug it/check the code
6. add/update/delete some files
7. git add <files>
8. git commit -m "reason"
9. git push origin master
10. go to the repo url and check the code at remote side

//3. Local branching Process:
1. git branch -- point to master only
2. create/cut the brach from master: git branch cart
3. switch to the cart branch: git checkout cart
4. git branch : it should point to cart
5. start working on cart branch (make sure in eclipse cart is reflected)
6. add/update/delete the files/code in your working copy (eclipse)
7. git status
8. git add <files>
9. git commit -m <reason>
10. git push origin cart
11. changes should be reflected at remote side : a new cart brnach should be created at remote

//4. PR (Pull Request + Merge):
12. Raise a PR to the reviewers with summary
13. Rev will check the code and put the comments and PR is not approved
14. Req will read those comments and will update the code in WC (eclipse)
15. Req will add--> commit --> push (git push origin cart)
16. PR is updated with latest car page changes
17. Rev will again check the latest changes as per the given review comments
18. This time PR is approved
19. Req/Rev will merge the PR to master (Cart --> Master)
20. Check the master branch (remote) : make sure cart changes are refelected
21. git checkout master (point to master branch)
22. Req has to take the latest pull : git pull origin master
23. In local, Master is updated with latest Pull (cartpage)

//5. A new assignment process (with exisiting team members):
1. team member has to take the latest pull from master (whenever there are chnages in master - remote): git master---> git pull origin master
2. cut the branch: git branch <name>
3. follow #3 and #4 processes

//6. Merge Conflict:
1. Naveen is making some changes in local --> demopage.java--> demo() -- master branch
2. Shailesh also making some changes in local --> demopage.java--> demo() -- master branch
3. Shailesh will merge the code to master after PR
4. Naveen will try to take the latest pull:git pull origin master
5. PULL will be aborted
6. Naveen has to move the code to stash: git add <file> : add to stage and then stash
git stash
7. then take the latest pull: git pull origin master
8. Remote changes will be reflect in Naveen's local WC
9. Naveen has to take the stash code back to WC: git stash pop
10. Merge conflict will happen
<<<<<<<<upstream
remote code
=======
>>>>>>>>>downstream stash
local code
11. communicate and resolve it, accept the respective changes (local/remote)
12. Naveen has to push the code to master (if Naveen's local changes are accepted)
13. Shailesh has to take the latest pull

//4. Reset:
1. do some wrong changes at remote (wrong push)
2. git log --oneline --> it will give you the commit history
3. copy the (N-1) commit hash code (ID)
4. and use reset: git reset --hard 66744b1 (ID)
5. finally do the force push to the remote side
git push -f origin master
6. that wrong file should be deleted from remote side(reverted back to the previous commit)

//.gitignore file:

```
allure-results/
screenshots/
screenshot/
test-output/
build/
################################
## Java
################################
.mtj.tmp/
*.class
*.jar
*.war
*.ear
*.nar
hs_err_pid*
activityLog.log
################################
## Maven
################################
target/
pom.xml.tag
pom.xml.releaseBackup
pom.xml.versionsBackup
pom.xml.next
pom.xml.bak
release.properties
dependency-reduced-pom.xml
buildNumber.properties
.mvn/timing.properties
.mvn/wrapper/maven-wrapper.jar

################################
## IntelliJ
################################
out/
.idea/
.idea_modules/
*.iml
*.ipr
```

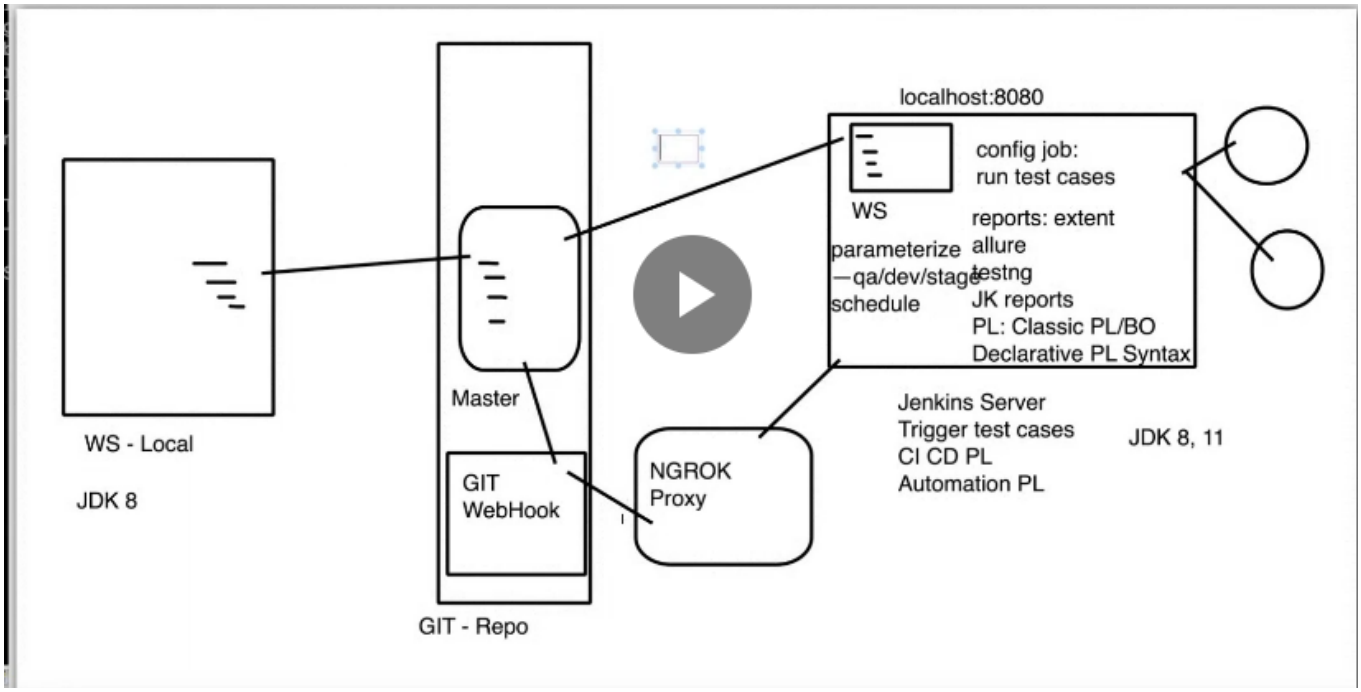
```
*.iws
##############################
## Eclipse
##############################
.settings/
bin/
tmp/
.metadata
.classpath
.project
*.tmp
*.bak
*.swp
*~.nib
local.properties
.loadpath
.factorypath

## OS X
##############################
.DS_Store
```

Jenkins_NGROK proxy configuration (Dhrumil)

NGROK proxy agent is act as bridge between Remote Repository and Jenkins.

"ngrok is a globally distributed reverse proxy fronting your web services running in any cloud or private network, or your machine."

Desired Goal:



Steps:

1. Sign up to ngrok website and it will generate unique auth token associated with your account.
2. Download ngrok binary zip file as per your OS configurations.
3. Unzip it.
4. Command Line > Go to specific ngrok zip folder
5. Run the command : ngrok config add-authtoken "Unique Token"
   a. This will add your auth token to the default ngrok.yml configuration file.
6. To get help with ngrok commands : ngrok help
7. To start a HTTP tunnel which forwarding to your local port on which Jenkins is configured.
   a. ngrok http 8080
      i. By running it will give you below output and details about ngrok configurations.
      ii.

Command Prompt - ngrok http 8080

```
ngrok

Session Status                online
Account                       Dhrumil Soni (Plan: Free)
Version                       3.0.3
Region                        United States (us)
Latency                       334.8136ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://b0ab-72-138-150-22.ngrok.io -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              3       0       0.00    0.00    5.51    6.61

HTTP Requests
-------------

POST /github-webhook/         200 OK
POST /github-webhook/         200 OK
POST /github-webhook/         200 OK
```

8. Now, its time to configure connection establishment between your ngrok agent and Github through webhook.

a. Login to github and select your repository.
b. Settings > WebHooks > Payload URL
c. In Payload URL: add Forwarding URL/github-webhook/
d. Update webhook.

9. Next action is to configure and enabling at Jenkins end, How Jenkins came to know whether code is pushed and I need to build job automatically.
   a. That can be achieved by selecting
      i. Github hook trigger for GitSCM polling checkbox, Save and Apply.
10. Push the small change to Cloud Repo and see the Magic.....

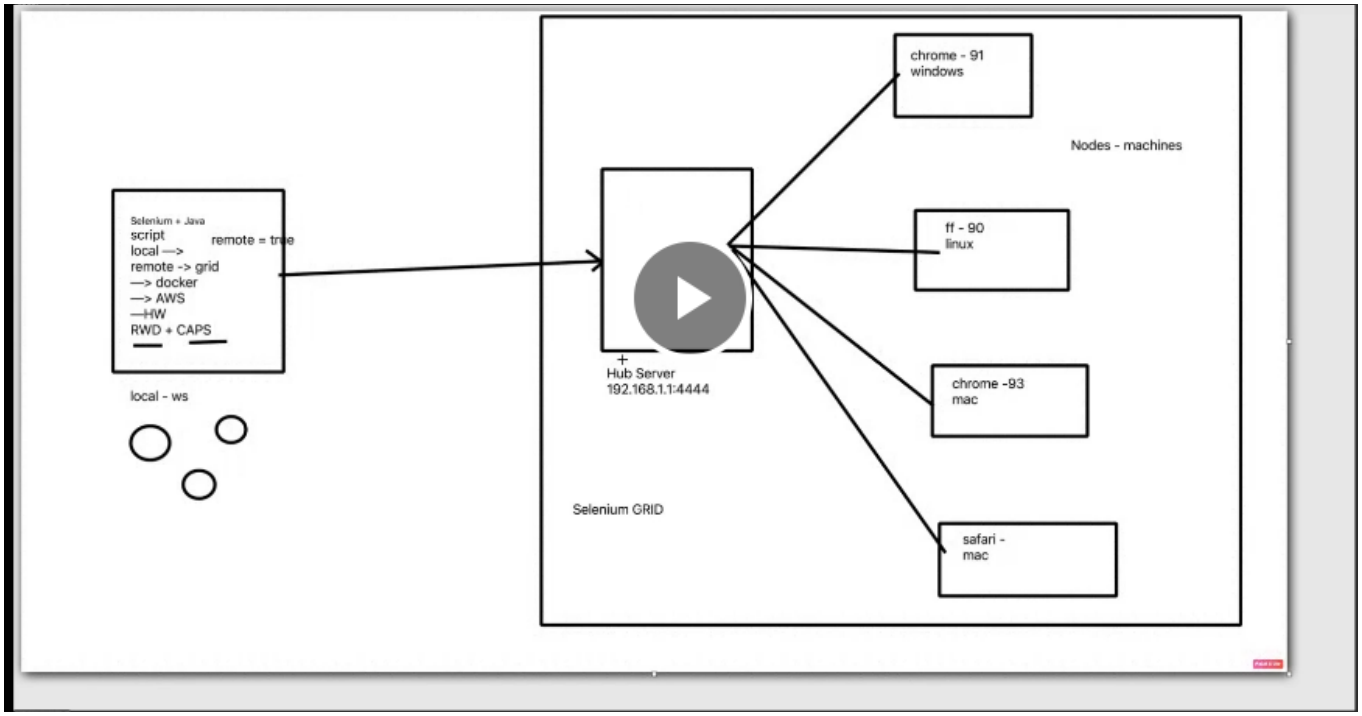
Debugging steps:

- If build is not automatically triggering, check WebHook recent delivery in Github and check logs.
- Try to resend it, this might work.

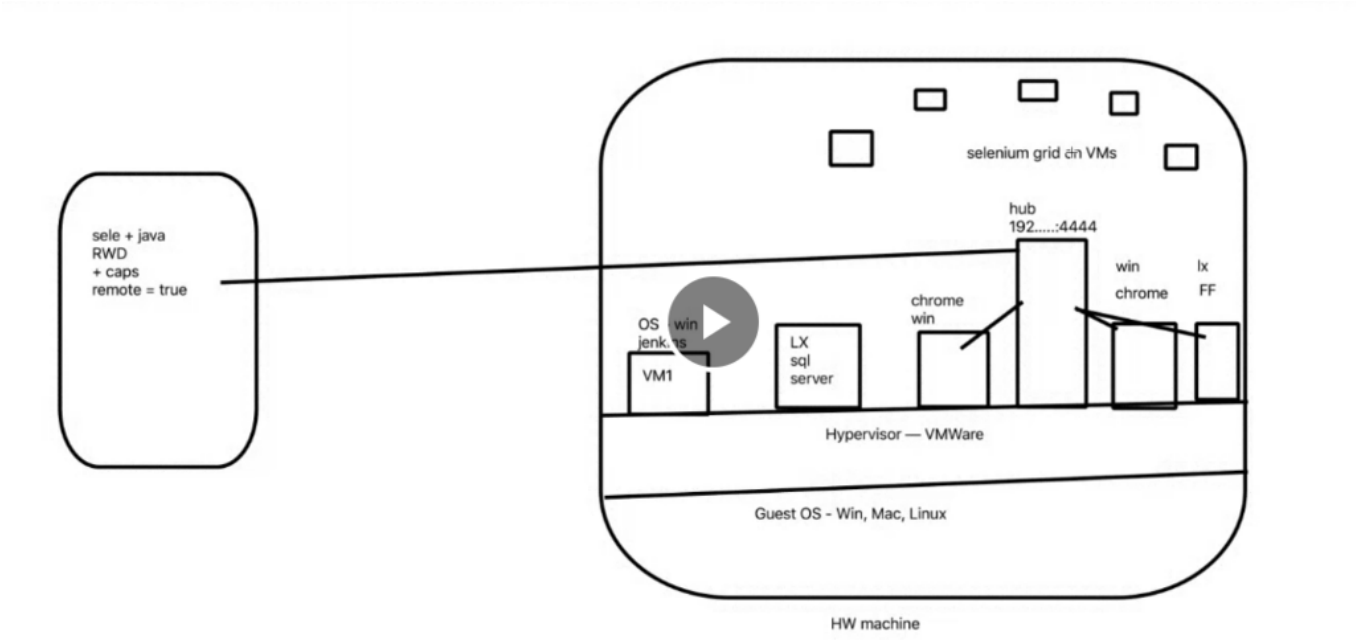
Jenkinsfile debugging Link:

- https://stackoverflow.com/questions/45140614/jenkins-pipeline-sh-fail-with-cannot-run-program-nohup-on-windows

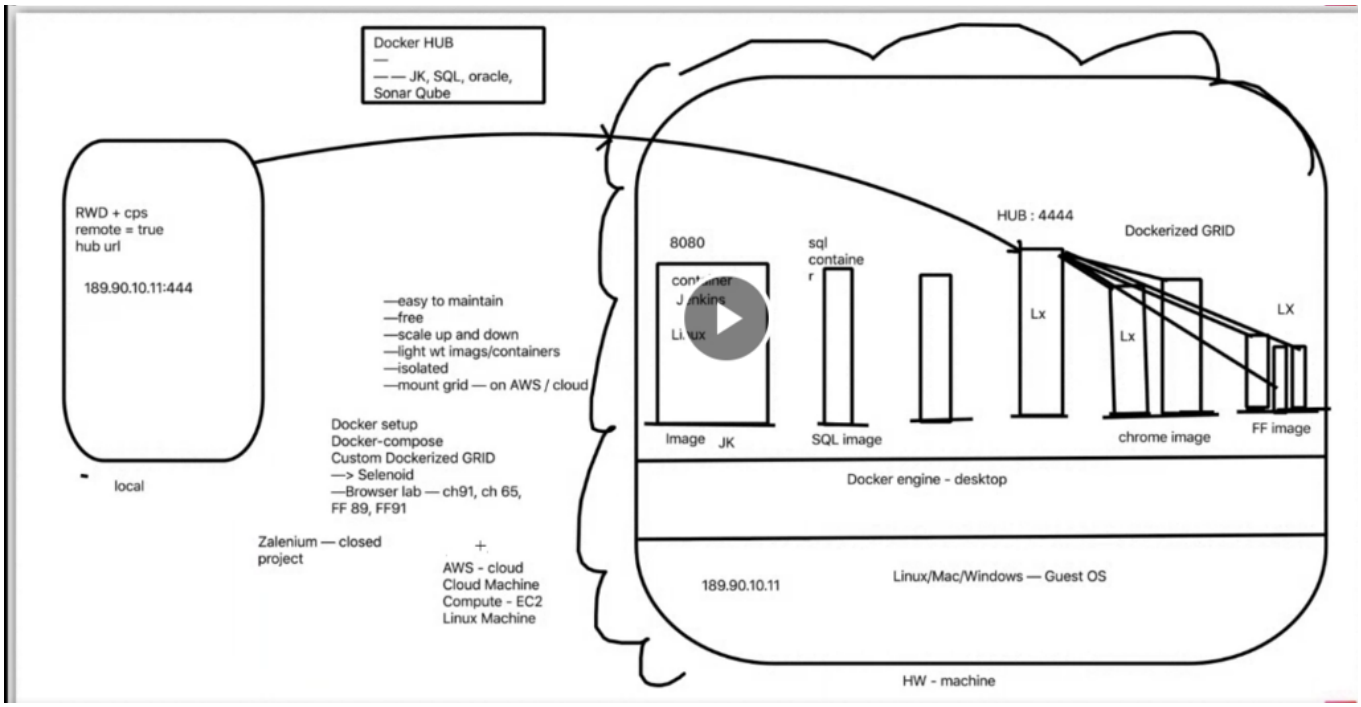
Selenium Grid Concept: (@Dhrumil)



Virtualization: (@Dhrumil)



Dockerized Grid Introduction: (@Dhrumil)



Docker Commands: (@Dhrumil)

- docker -v ==> Current version of docker
- docker-compose -v
- docker images => This command will show all the images

- docker system prune -a => This command will remove all stopped containers, all build cache, all images.
- docker run -d -p 80:80 docker/getting-started
- docker ps -a
- From Docker Hub to Docker Machine- docker pull command is used.
- docker run command will run specific container
- docker run = docker pull + start container

Pull below images

- **selenium/hub -> Tag - 3.141.59**
- **selenium/node-chrome-debug -> Tag - 3.141.59**
- **selenium/node-firefox-debug -> Tag - 3.141.59**

Create container on this images

- Hub Creation
  - docker run -d -p 4444:4444 --name selenium-hub -P selenium/hub:3.141.59
    - -d = detached mode
    - 4444:4444 => Local port is mapped with docker container machine port, Port forwarding.
    - --name => name of the container.
    - -P = name of the image
- Link chrome node to Selenium Hub by using below command
  - docker run -d --link selenium-hub:hub -P selenium/node-chrome-debug:3.141.59
- Link Firefox node to Selenium Hub by using below command
  - docker run -d --link selenium-hub:hub -P selenium/node-firefox-debug:3.141.59

- docker stop "container id" => This will stop container id.
- docker rm 'container id' => This will stop and remove container id.
- docker rmi "image id"
- docker images
- docker-compose -v
- docker-compose up -d
- docker system prune -a => This will remove and delete all the infrastructure setup i.e Images and Containers.

VNC viewer is used to check and visualize the execution inside the docker containers.

docker-compose.yml file
==========
version: "3"
services:

```
 selenium-hub:
   image: selenium/hub:3.141.59-20210929
   container_name: selenium-hub
   ports:
     - "4444:4444"

 chrome:
   image: selenium/node-chrome:3.141.59-20210929
   volumes:
     - /dev/shm:/dev/shm
   depends_on:
     - selenium-hub
   environment:
     - HUB_HOST=selenium-hub
     - HUB_PORT=4444

 firefox:
   image: selenium/node-firefox:3.141.59-20210929
   volumes:
     - /dev/shm:/dev/shm
   depends_on:
     - selenium-hub
   environment:
     - HUB_HOST=selenium-hub
     - HUB_PORT=4444
```

==========

Selenoid Grid Setup

If anyone getting this ERROR with Remote Grid execution:
org.openqa.selenium.SessionNotCreatedException: Could not start a new session. Response code 500. Message: unknown error: Chrome failed to start: crashed. (unknown error: DevToolsActivePort file doesn't exist)

Refer: https://stackoverflow.com/questions/50642308/webdriverexception-unknown-error-devtoolsactiveport-file-doesnt-exist-while-t

Add below lines in the **optionsManager** class in getChromeOptions() method.

```
co = new ChromeOptions();
co.addArguments("--no-sandbox");
```

co.addArguments("--disable-dev-shm-usage");

==========

docker-compose down ==> This will down whole infrastructure down and removed all the running containers.
This command will not remove any images which installed through docker-compose.yml file.

Biggest advantage of using docker-compose is **SCALABILITY**.

One can scale up the environment in easy manner by running below command:

docker-compose up --scale chrome=4 -d

You can scale up and scale down by using above command only.

docker-compose up
docker-compose down

docker network create selenoid