ACCENTURE

What is selenium?

Selenium is one of the most popular automated testing suites. Selenium is designed in a way to support and encourage automation testing of functional aspects of web based applications and a wide range of browsers and platforms. Due to its existence in the open source community, it has become one of the most accepted tools amongst the testing professionals.

Selenium is not just a single tool or a utility, rather a package of several testing tools and for the same reason it is referred to as a Suite. Each of these tools is designed to cater different testing and test environment requirements.

The suite package constitutes of the following sets of tools:

- **Selenium Integrated Development Environment (IDE)** – Selenium IDE is a record and playback tool. It is distributed as a Firefox Plugin.
- **Selenium Remote Control (RC)** – Selenium RC is a server that allows user to create test scripts in a desired programming language. It also allows executing test scripts within the large spectrum of browsers.
- **Selenium WebDriver** – WebDriver is a different tool altogether that has various advantages over Selenium RC. WebDriver directly communicates with the web browser and uses its native compatibility to automate.
- **Selenium Grid** – Selenium Grid is used to distribute your test execution on multiple platforms and environments concurrently.

Can we over load main method in java?

➔ yes we can overload main method. main method must not be static main method.

```
class Sample{
public void main(int a,int b){
System.out.println("The value of a is "  +a);
}
public static void main(String args[]){
System.out.println("We r in main method");
Sample obj=new Sample();
obj.main(5,4);
main(3);
}
```

ANS
We r in main method
The value of a is

Explain about final class?

➔ *final* keyword is used in different contexts. First of all, *final* is a non-access modifier applicable **only to a variable, a method or a class**.Following are different contexts where final is used.

Final variables

When a variable is declared with *final* keyword, its value can't be modified, essentially, a constant.

```
final int THRESHOLD;
```

Final classes
When a class is declared with *final* keyword, it is called a final class. A final class cannot be extended(inherited).

```
final class A
{
     // methods and fields
}
// The following class is illegal.
class B extends A
{
    // COMPILE-ERROR! Can't subclass A
}
```

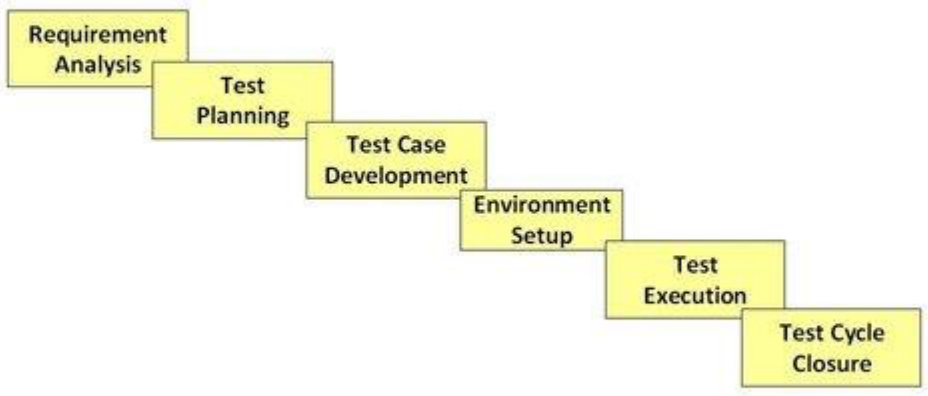
The other use of final with classes is to create an immutable class like the predefined String class.You can not make a class immutable without making it final.

Final methods
When a method is declared with *final* keyword, it is called a final method. A final method cannot be overridden. The Object class does this—a number of its methods are final.

```
final void m1()
    {
        System.out.println("This is a final method.");
    }
```

Software Test life cycle

STLC stands for Software **Testing Life Cycle**. STLC is a sequence of different activities performed by the **testing** team to ensure the quality of the software or the product.
Phases of STLC



How to handle drop down using selenium ?
Value in the drop down can be selected using WebDriver's **Select class**.

Syntax:

selectByValue:
Select selectByValue = ***new***
Select(driver.findElement(By.id("SelectID_One")));
selectByValue.selectByValue("greenvalue");

selectByVisibleText:

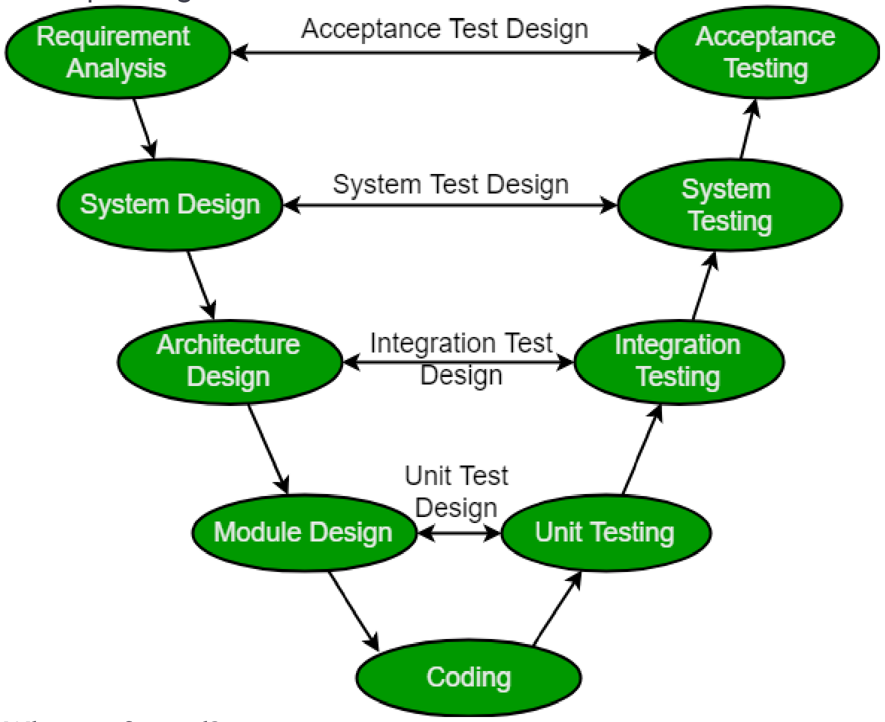
Select selectByVisibleText = ***new*** *Select (driver.findElement(By.id("SelectID_Two")));*
selectByVisibleText.selectByVisibleText("Lime");

selectByIndex:

Select selectByIndex = ***new*** *Select(driver.findElement(By.id("SelectID_Three")));*
selectByIndex.selectByIndex(2);

Using selenium can we automate desktop application's ?
➔ Selenium does not have the capability to automate the desktop applications. The simple answer is no. Selenium is designed to automate web applications, not desktop applications. To automate desktop applications, you will need a different automation tool like WinAppDriver which is designed for desktop automation.
WinAppDriver (short for Windows Application Driver) is a free test automation tool for Windows desktop apps developed by Microsoft.

About vmodel?
The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.



Why preferred?
It is easy to manage due to the rigidity of the model. Each phase of V-Model has specific deliverables and a review process.
Proactive defect tracking – that is defects are found at early stage.

Selenium, Appium, and java which version used in the project.
=>Selenium version=>3.141.59 & java version 1.8

Waits in selenium?

There are primarily two types of waits available in Selenium WebDriver.

Implicit Wait
Explicit Wait
Fluent Wait

Implicit Wait

Implicit waits are used to provide the latency within each and every test step of the test script. Thus, the program control would wait for the specified time before moving the execution control to the next step. Thus the implicit waits are applied to all the test step of the testscript. In other words we can say that the system would wait for the specified period of time in order to load the desired object into the DOM.

```
//Create WebDriver instance variable WebDriver driver;
        //Launch browser
        driver=new
        FirefoxDriver();
        //Apply implicit wait
        driver.manage().timeouts().implicitlyWait(20,
        TimeUnit.SECONDS);
```

Explicit Wait

Explicit waits are smarter waits than implicit waits. Explicit waits can be applied at certain instances instead of applying on every web element within the test script. Suppose if we are creating a login script for Gmail. We know that we are most likely to wait for a few seconds to let the home page load successfully. Thus, in such cases explicit waits can be used. Another important benefit that we get from explicit wait is that it waits maximum for the specified period of time or a condition to be met. Thus, if our condition (Let's say an element should be visible before we click on it) is met before the specified time has elapsed, then the system control would move forward rather than waiting for the complete time to elapse in order to save our execution time.

Code Sample

```
public void explicitWait(WebDriver driver)
{
WebDriverWait wait = new WebDriverWait(driver, 20);
wait.until(ExpectedConditions.elementToBeClickable(driver.findElement(By
.id ("element id"))));
}
```

In the above method WebDriver would wait for the expected condition (elementToBeClickable) to be met or for the timeout (20 seconds) to occur. As soon as the condition is met, the WebDriver would execute the next test step.

Fluent Wait in Selenium

The Fluent Wait command defines the maximum amount of time for Selenium WebDriver to wait for a certain condition to appear. It also defines the frequency with which WebDriver will check if the condition appears before throwing the “ElementNotVisibleException”.

Fluent Wait commands are most useful when interacting with web elements that can sometimes take more time than usual to load. This is largely something that occurs in Ajax applications.

While using Fluent Wait, it is possible to set a default polling period as needed. The user can configure the wait to ignore any exceptions during the polling period.

Syntax

```
Wait wait = new FluentWait(WebDriver reference)

.withTimeout(timeout, SECONDS)

.pollingEvery(timeout, SECONDS)

.ignoring(Exception.class);

WebElement foo=wait.until(new Function<WebDriver, WebElement>() {

public WebElement applyy(WebDriver driver) {

return driver.findElement(By.id("foo"));
```

```
    }

    });
```

Thread. sleep in selenium

=> Thread. sleep() method can be used to pause the execution of current thread for specified time in milliseconds. The argument value for milliseconds can't be negative, else it throws IllegalArgumentException .
Thread. sleep (1000)

Explain excel data-driven testing?
=>A Data Driven Framework in Selenium is a technique of separating the “data set” from the actual “test case” (code).
WebDriver does not directly support data reading of excel files. Therefore, one needs to use a plugin such as **Apache POI** for reading/writing on any Microsoft office document.

```
try {
File s = new File(excelfilePath);
FileInputStream stream = new FileInputStream(s);
work_book = new XSSFWorkbook(stream);
}
```

Action and select class methods
=> 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. As Select is an ordinary class, its object is created by the keyword New and also specifies the location of the web element.
Syntax:

```
Select objSelect = new Select();
```

Different Select Methods
The following are the most commonly used methods to deal with a drop-down list:
1. **selectByVisibleText:** *selectByVisibleText(String arg0): void*
This method is used to select one of the options in a drop-down box or an option among multiple selection boxes. It takes a parameter of String which is one of the values of Select element and it returns nothing.

Syntax:

```
obj.Select.selectByVisibleText(“text”);
```

Example:

```
Select objSelect =new Select(driver.findElement(By.id("search-box")));
objSelect.selectByVisibleText("Automation");
```

2. selectByIndex: *selectByIndex(int arg0) : void*
This method is similar to ‘*selectByVisibleText*’, but the difference here is that the user has to provide the index number for the option rather than text. It takes the integer parameter which is the index value of Select element and it returns nothing.
Syntax:

```
oSelect.selectByIndex(int);
```

Example:

```
Select objSelect = new Select(driver.findElement(By.id("Seacrch-box")));
Select.selectByIndex(4);
```

3. selectByValue: *selectByValue(String arg0) : void*
This method asks for the value of the desired option rather than the option text or an index. It takes a String parameter which is one of the values of Select element and it does not return anything.
Syntax:

```
oSelect.selectByValue("text");
```

Example:

```
Select objSelect = new Select(driver.findElement(By.id("Search-box")));
objSelect.selectByValue("Automation Testing");
```

4. getOptions: *getOptions( ) : List<WebElement>*
This method gets all the options belonging to the Select tag. It takes no parameter and returns List<WebElements>.
Syntax:

```
oSelect.getOptions();
```

Example:

```
Select objSelect = new Select(driver.findElement(By.id("Search-box")));
List <WebElement> elementCount = oSelect.getOptions();
System.out.println(elementCount.size());
```

5. deselectAll()
This method clears all the selected entries. This is only valid when the drop-down element supports multiple selections.
Syntax:

```
objSelect.deselectAll();
```

Actions
Actions class is an ability provided by Selenium for handling keyboard and mouse events.
In Selenium WebDriver, handling these events includes operations such as drag and drop, clicking on multiple elements with the control key, among others.
Action class is defined and invoked using the following syntax:

```
Actions action = new Actions(driver);
action.moveToElement(element).click().perform();
```

Mouse Actions:
doubleClick(): Performs double click on the element
clickAndHold(): Performs long click on the mouse without releasing it
dragAndDrop(): Drags the element from one point and drops to another
moveToElement(): Shifts the mouse pointer to the center of the element
contextClick(): Performs right-click on the mouse
Keyboard Actions:
sendKeys(): Sends a series of keys to the element
keyUp(): Performs key release
keyDown(): Performs keypress without release

How to right-click in selenium?
=> contextClick() method for right click to an element after moving the //mouse to with the moveToElement()

```
Actions a = new Actions(driver);
a.moveToElement(driver.findElement(By.xpath("input[@type='text']"))).
contextClick().
build().perform();
```

TestNG annotations.
=> Hierarchy of the TestNG Annotations:
@BeforeSuite
@BeforeTest
@BeforeClass
@BeforeMethod
@Test
@AfterMethod
@AfterClass
@AfterTest
@AfterSuite

TestNG supports many different annotations to configure Selenium WebDriver test.

@Test

@Test annotation describes method as a test method or part of your test.

@Test (Priority = 1)

@ to the priority of the function

@Test(group = {“smoke”})

@ set the method name with group, when testNG will run “smoke” group only those methods will run which are only belongs to the “smoke” group

@Parameters

When you wants to pass parameters in your test methods, you
need to use @Parameters annotation.
In the .xml file
(<parameter name="browser" value="FFX" />

In the class, use just above the
methods @Test
@Parameters ({"browser"})

@BeforeMethod

Any method which is marked with @BeforeMethod annotation will be executed before each and every @test annotated method.

@AfterMethod

Same as @BeforeMethod, If any method is annotated with @AfterMethod annotation then it will be executed after execution of each and every @test annotated method.

@BeforeClass
Method annotated using @BeforeClass will be executed before first @Test method execution. @BeforeClass annotated method will be executed once only per class so don't be confused.

@AfterClass
Same as @BeforeClass, Method annotated with @AfterClass annotation will be executed once only per class after execution of all @Test annotated methods of that class.

@BeforeTest
@BeforeTest annotated method will be executed before the any @Test annotated method of those classes which are inside <test> tag in testng.xml file.

@AfterTest
@AfterTest annotated method will be executed when all @Test annotated methods completes its execution of those classes which are inside <test> tag in testng.xml file.

@BeforeSuite
Method marked with @BeforeSuite annotation will run before the all suites from test.

@AfterSuite
@AfterSuite annotated method will start running when execution of all tests executed from current test suite.

@DataProvider
When you use @DataProvider annotation for any method that means you are using that method as a data supplier. Configuration of @DataProvider annotated method must be like it always return Object[][] which we can use in @Test annotated method.
If you want to provide the test data, the DataProvider way, then we need to declare a method that returns the data set in the form of two **dimensional object array Object[][]. The first array represents a data set whereas the second array contains the parameter values**.
There are two way to provide data using @DataProvideer annotation

Dynamic way: passing data through another method:

We have provided the DataProvider method getData within the test class itself.

Note that it is annotated with @DataProvider. Since it doesn't have the name attribute, its name by default will be getData. It returns two sets of data, each set of which contains two values, an integer and a string value.

```
Public class

InstanceDataProviderExample {

@Test(dataProvider="getData")
public void instanceDbProvider(int p1, String p2) {
System.out.println("Instance DataProvider Example: Data(" + p1 + ", " + p2 +")");}
@DataProvider
        public Object[][] getData() {
                return new Object[][]{{5, "five"}, {6, "six"}};
        }}
```

```
OUTPUT:
[TestNG] Running:
 C:\javacodegeeks_ws\testNgDataProvider\test\com\javacodegeeks\testng\testn
 g.xml

Instance DataProvider Example: Data(5,
five) Instance DataProvider Example:
Data(6, six)
```

Static Data Provider: DataProvider method can also be defined in a separate class as a static method, in which case, the test method using it has to specify both the DataProvider name and its class in the @Test attributes dataProvider and dataProviderClass.

```
public class StaticDataProviderExample {

@Test(dataProvider="client1",dataProviderClass=DataProviderSource.c
        lass)
public void client1Test(Integer p) {
                System.out.println("Client1 testing: Data(" + p + ")");
        }
@Test(dataProvider="client2", dataProviderClass=DataProviderSource.class)
 public void client2Test(Integer p) {
                System.out.println("Client2 testing: Data(" + p + ")");
        }
}
```

```
OUTPUT:
C:\javacodegeeks_ws\testNgDataProvider\test\com\javacodegeeks\testng\staticD
ataPr oviderTestng.xml
```

Client1 testing:
Data(1) Client2
testing: Data(2)

@BeforeGroups

@BeforeGroups annotated method will run before the first test run of that specific group.

@AfterGroups

@AfterGroups annotated method will run after all test methods of that group completes its execution.

@Factory

When you wants to execute specific group of test cases with different values, you need to use @Factory annotation. An array of class objects is returned by @Factory annotated method and those TestNG will those objects as test classes.

@Listeners

@Listeners are used to with test class. It is helpful for logging purpose.

How you will give priority in TestNG

=> Using the priority attribute

```
@Test(priority = 3)
        public void testThree() {
        }

        @Test(priority = 1)
        public void testOne() {
        }
```

How you will define in testng if any other methods depend on any method.

Using dependencies

```
public class TestNGOrderedTests {

        @Test(dependsOnMethods = {"parentTest"})
        public void childTest() {
        }

        @Test
        public void parentTest() {
        }
}
```

Types of Xpath

=>XPath is a technique in Selenium to navigate through the HTML structure of a page. XPath enables testers to navigate through the XML structure of any document, and this can be used on both HTML and XML documents
Xpath=//tagname[@attribute='value']

XPath Locators	Find different elements on web page
ID	To find the element by ID of the element
Classname	To find the element by Classname of the element
Name	To find the element by name of the element
Link text	To find the element by text of the link
XPath	XPath required for finding the dynamic element and traverse between various elements of the web page
CSS path	CSS path also locates elements having no name, class or ID.

How to do Parent child traversing in XPath?
With selenium xpath parent child traverse relationship method, you identify a unique parent or grand parent and from that point carefully traverse to your desired object with the help of tags and back slashes
//div[@class='region region-navigation']/section/form/div/div/div/input

Explain window handling in selenium.
=> **What is a window handle?**
It is a unique identifier that holds the address of all the windows. Think of it as a pointer to a window, which returns the string value. It is assumed that each browser will have a unique window handle. This window handle function helps to retrieve the handles of all windows.
Syntax
get.windowhandle(): This method helps to get the window handle of the current window
get.windowhandles(): This method helps to get the handles of all the windows opened
set: This method helps to set the window handles in the form of a string. *set<string> set= driver.get.windowhandles()*
switch to: This method helps to switch between the windows
action: This method helps to perform certain actions on the windows

Explain Java modifiers
=> What are java modifiers?
1. Private access modifier.
2. Role of private constructor.
3. Default access modifier.
4. Protected access modifier.
5. Public access modifier.

Difference between merge and rebase.
=> Git rebase and merge both integrate changes from one branch into another. Where they differ is how it's done. Git rebase moves a feature branch into a master. Git merge adds a new commit, preserving the history.

How you will select frames from multiple frames?
driver.switchTo().frame(int arg0);
Select a frame by its (zero-based) index. That is, if a page has multiple frames (more than 1), the first frame would be at index "0", the second at index "1" and so on

Can we achieve multiple inheritance?
=> Java does not support multiple inheritance using classes. "A class can extend only one class but it can implement multiple interfaces." For example,
below inheritance using multiple classes is wrong as two classes cannot be extended or inherited. Class C is inheriting class A and B

What is difference between assert and verify commands?

Assert: Assert command checks whether the given condition is true or false. Let's say we assert whether the given element is present on the web page or not. If the condition is true then the program control will execute the next test step but if the condition is false, the execution would stop and no further test would be executed.
Verify: Verify command also checks whether the given condition is true or false. Irrespective of the condition being true or false, the program execution doesn't halts
i.e. any failure during verification would not stop the execution and all the test steps would be executed.

Nosuchelementexception and elementnotfound exception what is the difference between that?
The NotFoundException is a super class which includes the subclass NoSuchElementException. The known direct subclasses of NotFoundException are:
NoAlertPresentException,
NoSuchContextException,
NoSuchElementException,
NoSuchFrameException and
NoSuchWindowException.
So, if I want one catch statement to catch all five exceptions and they will all be handled the same way then I can just handle NotFoundException. But if I want to handle any of these five exceptions differently, I can catch the more specific subclass.
The NoSuchElementException is thrown when the element you are attempting to find is not in the DOM. This can happen for three reasons.
- The first is because the element does not exist and never will. To fix this, change your findElement to be correct.
- The second is that you need to do something on the page to make the element appear.

Who is writing feature files in ur project and who should write it by your understanding?
Feature file is a file of written steps in plane English format ideally in Given when then
Ideally Business Analyst should write this feature file

Suppose there are two elements with the same locators then how you will select the second element.
If you really want to continue testing something that is invalid, and you also want to corrupt your automated tests (which you really don't want to do), you could do the following:

```
driver.findElement(By.xpath("(//img[@id='MoveAllRight'])[2]")).click();
```
It will work, but it is the wrong thing to do.

OR

```
list<webelement>  listele = driver.findelements(by.tageName("Img");
syso(listele.size)

listele.get(0).click;
listele.get(1).click;
```

Can we have multiple catch after single try?
Yes. They will be executed on the specific to generic order

Where u used collection in your framework?
You can mentioned below -
Reading data from Excel
Reading data from DB
Reading dynamic tables

What is hashmap and how it works ?
HashMap works on the principle of hashing, we have put(key, value) and get(key) method for storing and retrieving Objects from HashMap. When we pass Key and Value object to put() method on Java HashMap, HashMap implementation calls hashCode method on Key object and applies returned hashcode into its own hashing function to find a bucket location for storing Entry object, important point to mention is that HashMap in Java stores both key and value object as Map.Entry in a bucket which is essential to understand the retrieving logic.

How to iterate array-list ?

Method 1 : Using for loop :
```
import java.util.*;
class GFG {
   public static void main(String[] args)
   {
      // initializing ArrayList
      List<Integer> numbers = Arrays.asList(1, 2, 3,
                              4, 5, 6, 7, 8);
      // For Loop for iterating ArrayList
      for (int i = 0; i < numbers.size(); i++)
         System.out.print(numbers.get(i) + " ");
   }
}
```
Method 2 : Using Iterator
```
// Java program to iterate over an arraylist
// using Iterator
import java.util.*;
class GFG {
   public static void main(String[] args)
   {
      // initializing ArrayList
      List<Integer> numbers = Arrays.asList(1, 2, 3,
                                    4, 5, 6, 7, 8);
      // Looping ArrayList using Iterator
      Iterator it = numbers.iterator();
      while (it.hasNext())
         System.out.print(it.next() + " ");
   }
}
```

What and all commands u used it git?

Command	Description
git push origin --delete [branchName]	Delete a remote branch
git checkout -b [branch name]	Create a new branch and switch to it
git checkout -b [branch name] origin/[branch name]	Clone a remote branch and switch to it
git checkout [branch name]	Switch to a branch

SQL query - find the second largest salary?

```
SELECT name, MAX(salary) as salary FROM employee
```

How will you connect the db to java.

```
Database database;

@Before
public void createDatabase() {
  database = Databases.createFrom("com.mysql.jdbc.Driver",
"jdbc:mysql://localhost/test");
}

@After
public void shutdownDatabase() {
  database.shutdown();
}
```

Java program to find duplicates?

Here are two versions. One **using *ArrayList*** and the other using ***HashSet***
Compare them and create your own version from this, until you get what you need.

```
public class Repeated {
        public static void main( String  [] args ) {
            Collection listOne = new ArrayList(Arrays.asList("milan","dingo", "elpha", "hafil",
     "meat", "iga", "neeta.peeta"));
           Collection listTwo = new ArrayList(Arrays.asList("hafil", "iga", "binga", "mike",
     "dingo"));
```

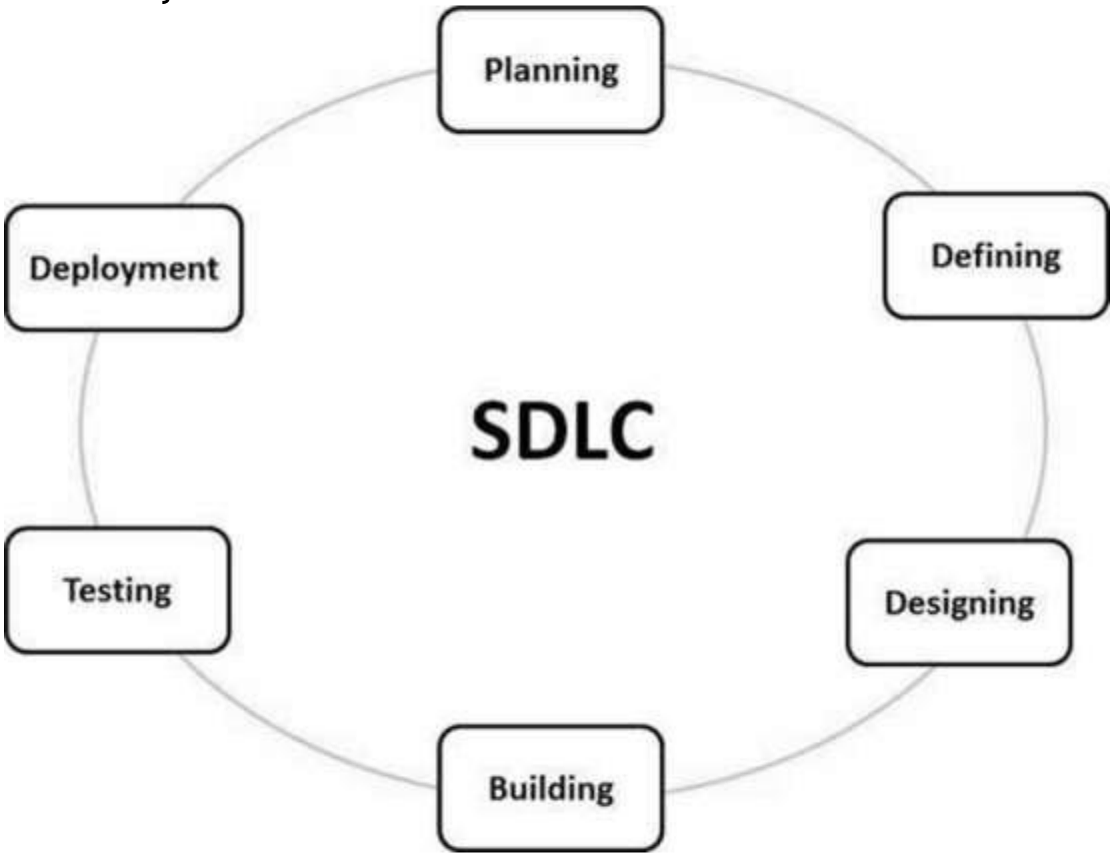
```
        listOne.retainAll( listTwo );
        System.out.println( listOne );
    }
}
```

Java program to find product without using multiplication.

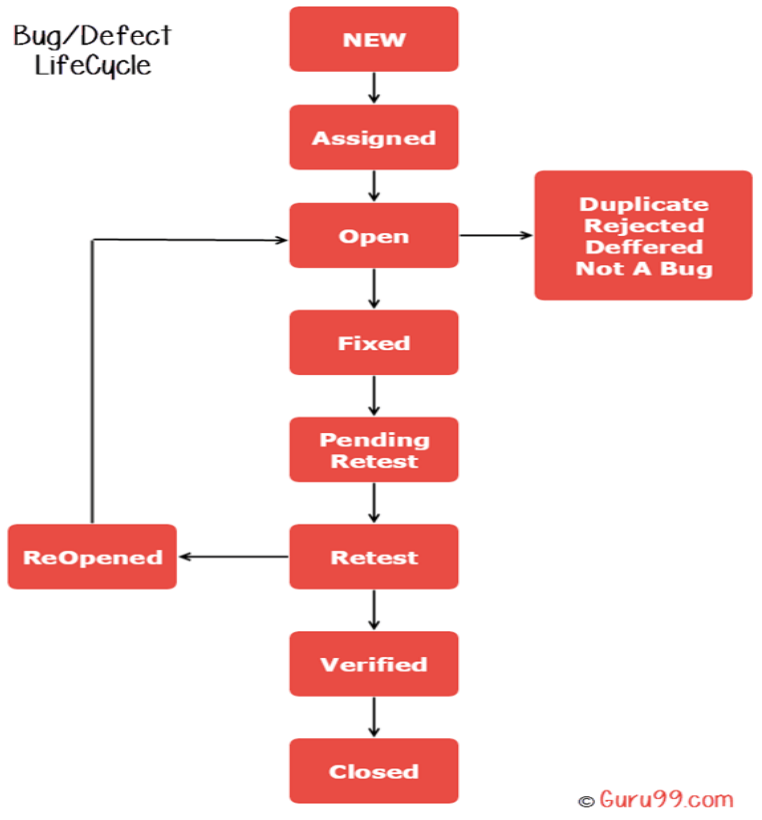
```
public static int multiplierLoop(int a, int b) {
    int resultat = 0;
    for (int i = 0; i < a; i++) {
        resultat += b;
    }

    return resultat;
}
```

SDLC Life Cycle



Defect/Bug Life Cycle



1. *Which java modifiers you used in your project- for main class file which modifier you have used?*
2. *If there is a textbox, which accepts 100 characters only when you start typing the tooltip should be there displaying how many characters are left. How will you validate how many counts are left?*
3. *Explain the framework used in the project.*
4. *Which external plugin is required to upload files in selenium?*
5. *How do you decide which testcase is feasible for automation or not*
6. *Questions on agile - process, scrum framework, retrospective meeting, team*
7. *Questions on Api testing - different status code, indempotent method , some scenarios as well.*