==========

Tutor: **Amuthan Sakthivel**

Reference: **Testing Mini Bytes**

Course: **Selenium - Java with Docker, Git, and Jenkins**

==========

1. Course URL: https://www.testingminibytes.com/courses/selenium-java-with-docker-git-and-jenkins
2. Document prepared by: **Rajat Verma**
    a. https://www.linkedin.com/in/rajat-v-3b0685128/
    b. https://github.com/rajatt95
    c. https://rajatt95.github.io/

---

1. **Learnings from Course**
    a. Web Application:
        i. https://opensource-demo.orangehrmlive.com/
    b. Why
        i. Automation?
        ii. Selenium?
        iii. Java?
    c. IDE:
        i. IntelliJ
            1. Plugin
                a. Rainbow brackets
                b. Sonarlint
                c. Create TestNG XML
            2. Templates
                a. Pre-Defined:
                    i. main - - > main() method
                    ii. sout - - > System.out.println()
                b. Custom:
                    i. Settings -> Live Templates -> Java
                    ii. dfbi - - > driver.findElement(By.id(""))
    d. Java basics:
        i. JDK vs JRE
        ii. Compiler - javac
        iii. Why do we set environment variables?
        iv. Human readable (.java files) code and Machine-readable code (.class files)
        v. Data Types
        vi. BODMAS

vii. if, if-else, if-else if-if
viii. &&, ||
ix. For Loop
x. Method calling
xi. Array
xii. ArrayList
    1. Dynamic
xiii. String
    1. length()
    2. substring()
    3. chartAt()
    4. toLowerCase()
    5. toUpperCase()
    6. replace()
xiv. Variables:
    1. Local
        a. Present inside method
        b. scope is within the method
    2. Global
        a. present outside method
        b. scope is within the method

e. XPaths:
i. Absolute
ii. Relative
    1. XPath axes
        a. Parent (**/..**)
        b. Ancestor
        c. Preceding-sibling
        d. Following-sibling
        e. Bi-Traverse
    2. XPath functions
        a. text()
        b. contains()
iii. Dynamic XPaths:
    1.

```
String oldXpath="//*[text()='TEST']";
System.out.println(oldXpath);
System.out.println(oldXpath.replace( target: "TEST", replacement: "Laptop"));
System.out.println(oldXpath.replace( target: "TEST", replacement: "Mobiles"));
```

f. Maven:
i. Build and Dependency Management Tool

g. WebDriverManager
    i. To manage the Drivers for different browsers and different platforms
h. **Java**:
    i. Class and Objects
    ii. Constructor and Constructor Overloading
    iii. Method Overloading
    iv. Keywords
        1. super
        2. this
        3. final
        4. static
        5. try
        6. catch
        7. finally
    v. OOPS concepts
        1. Abstraction
        2. Encapsulation
        3. Inheritance
            a. Single Level
            b. Multi-Level
        4. Polymorphism
            a. Static - Method Overloading
            b. Dynamic - Method Overriding
        5. Composition
    vi. Access Modifiers
    vii. Abstract classes
    viii. Interface
        1. From Java 8,
            a. We can have default methods (has a body) in Interfaces
            b. These methods can be overridden as well
            c. All the sub-classes will not be forced to implement these methods.
            d. But, sub-classes have to implement all the abstract methods
    ix. Typecasting
        1. UpCasting
        2. DownCasting
    x. Static Block
    xi. Scanner class
    xii. String Arguments in main() method
    xiii. Exception Handling
    xiv. Array
    xv. Reflection

xvi. Loops
    1. For
    2. ForEach
xvii. List, Set, Map
xviii. Static Imports
xix. Enums
xx. POJO

i. Selenium:

i. WebElements
    1. Dropdown:
        a. Bootstrap dropdown
        b. Select dropdown
ii. Locators:
    1. id
        a. Recommended
        b. Works fine with MultiLingual sites
    2. name
    3. xpath
iii. Actions class
iv. Frames Handling
v. File Upload
vi. JavascriptExecutor
vii. TakesScreenshot
viii. WebDriver Hierarchy
ix. Waits:
    1. Thread.sleep() -> not recommended
    2. Implicit
    3. Explicit
    4. Fluent
x. ChromeOptions
    1. Headless execution

j. **TestNG**

i. Reporting Capability
ii. Annotations
    1. @BeforeSuite
    2. @BeforeTest
    3. @BeforeClass
    4. @Test
    5. @AfterMethod
    6. @AfterClass
    7. @AfterTest
    8. @AfterSuite

Suite - - > Test - - > Class - - > Method

Suite is a collection of multiple Tests
Test is a collection of multiple Classes
Class is a collection of multiple Methods

9. DataProviders
10. Listeners
    a. ITestListener

iii. Annotations Properties
1. priority
2. description
3. invocationCount (default =1)
4. threadPoolSize (Parallel execution)
5. timeOut (max. Time for Test case)
6. dependsOnMethods
7. groups
8. dependsOnGroups

iv. Assert class and its methods

k. Maven:
i. Dependencies management
ii. Profile management in Maven

l. Apache POI
m. Owner library
n. To read values from the .properties file
o. Page Object Model
i. Method Chaining
ii. Page Chaining
iii. Why no to Page Factories?

p. Faker library
q. ExtentReports
i. Custom Annotation

r. Github
s. Docker
t. Amazon AWS EC2
u. Git
v. Jenkins

w. Automation Framework:

i. Automation Testing Tool: **Selenium Webdriver API**
ii. Programming language: **Java (Version 11)**
iii. Build ad Dependency Management Tool: **Maven**
iv. Testing framework: **TestNG**
v. Design Pattern**: Page Object Model** (No Page Factories)
vi. Reporting: **ExtentReports (Latest version - 5.0.9)**
vii. To read property files:
1. Owner library

https://mvnrepository.com/artifact/org.aeonbits.owner/owner

viii. Instead of DataProvider, we are going to use
   1. **Test Data Supplier**
      https://mvnrepository.com/artifact/io.github.sskorol/test-data-supplier

ix. To read values from the Excel file:
   1. **Test Data Supplier**

x. Custom Annotation
   1. For ExtentReport:
      a. Author
      b. Category

xi. Custom Enums

xii. OOPS concepts:
   1. Abstraction
   2. Encapsulation
   3. Inheritance
      a. is-a
   4. Polymorphism
   5. Composition
      a. has-a

xiii. Supports Parallel Cross Browser Testing
   1. Using ThreadLocal class

xiv. Other implementations:
   1. Icons for Browser added on Test level
   2. OS and Browser with version info added on Test level
   3. Zip all the ExtentReports
   4. Users can do customizations:
      a. Send Email to n no. of Recipients with n no. of Reports as attachments
      b. Re-try failed test cases
      c. Screenshots on Test step level

Softwares:

1. Java JDK
2. IntelliJ IDE
   a. Plugin -
      i. Rainbow Brackets
      ii. Sonarlint

========1_Java Installation | JDK and JRE | IntelliJ Installation==========

---

1. **Why Automation?**
    a. Reduce Manual Efforts
    b. Reduce Testing Timeline
    c. Reduce Manual Error
    d. Reduce Regression Testing Efforts
    e. Reduced Cost
    f. High Pay

---

2. **Why Selenium?**
    a. Open-source (Free)
    b. Wide User base
    c. Supports multiple languages (Java, C#, Python, Ruby)
    d. Supports multiple browsers (Chrome, Safari, Edge, Firefox, Opera)
        i. Testing of a web application across different browsers is called ***Browser compatibility Testing***.
    e. Supports multiple OS (WIN, MAC, Linux)
    f. Supports Parallel Execution
    g. A lot of High Paying jobs
    h. Acts like base if you want to learn Appium
    i. Selenium - Web Applications, Appium - Mobile Applications

---

3. **Why Java?**
    a. Open-source
    b. Widely used
    c. Matured language existing over 20 years
    d. Used in multiple companies
    e. A lot of libraries available
    f. Based on OOPS
    g. Selenium with Java have most of the jobs in the market
    h. Web Automation - Selenium with Java
    Mobile Automation - Appium with Java
    API Automation - RestAssured with Java
    i. To automate the web application, we are going to use Java as a programming language and Selenium WebDriver API
    j. **Download Java**
        i. https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html
        ii. WIN (64-bit): **jdk-8u202-windows-x64.exe**

iii. MAC: **jdk-8u202-macosx-x64.dmg**

k. Install Java

i. Install

ii. Set the environment variables

1. Why do we need to set this?
    a. Because we want to use Java globally/from any location of my machine

l. **Java compiler** is responsible for conversion of

i. Human readable code -> Machine Readable Code (0's and 1's)
(.java file) -> (.class file)

ii. Location:

1. WIN:
    a. C://Program Files//Java//jdk1.8.0_281/bin/**javac**

m. **JDK**:

i. Java Development Kit

ii. We can develop Java apps and run/execute Java apps

iii. We can both

1. develop Human Readable Code (.java files)
2. run/execute Machine Readable code (.class files)

n. **JRE**:

i. Java Runtime Environment

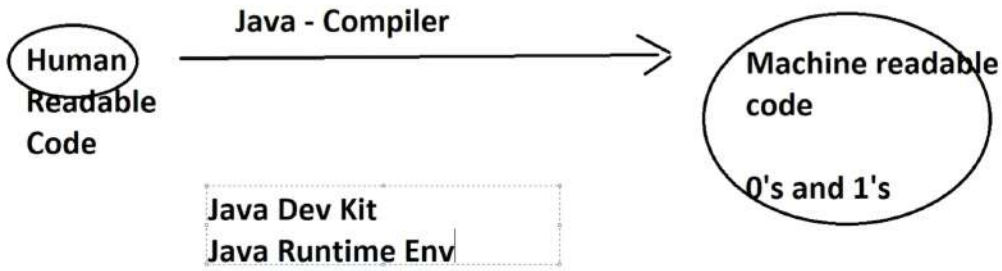
ii. We can only run/execute Java applications

iii. We can only run/execute Machine Readable code (.class files)

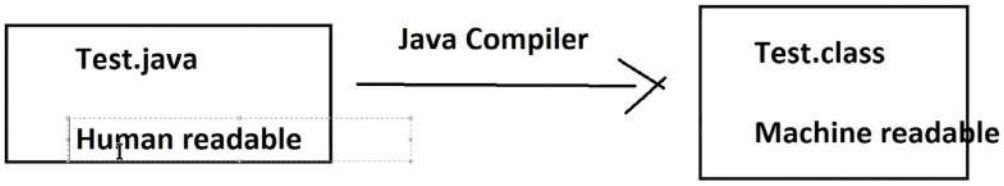
1. **JRE does not have a javac compiler**
2. WIN:
    a. C://Program Files//Java//jre1.8.0_281/bin/**java**
    b. Only **java** is present inside JRE, no **javac** is available

---

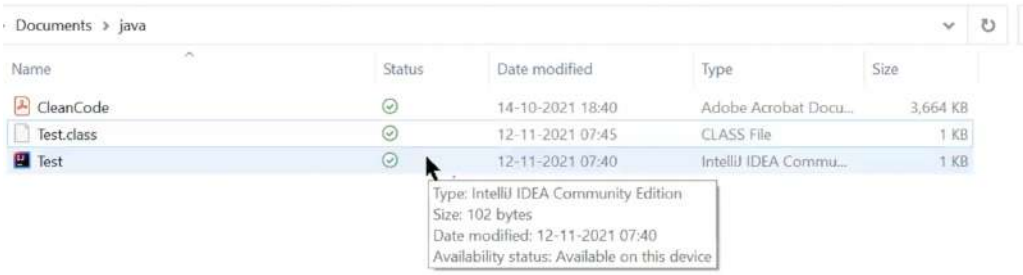
4. Java Compiler:



a.



b.



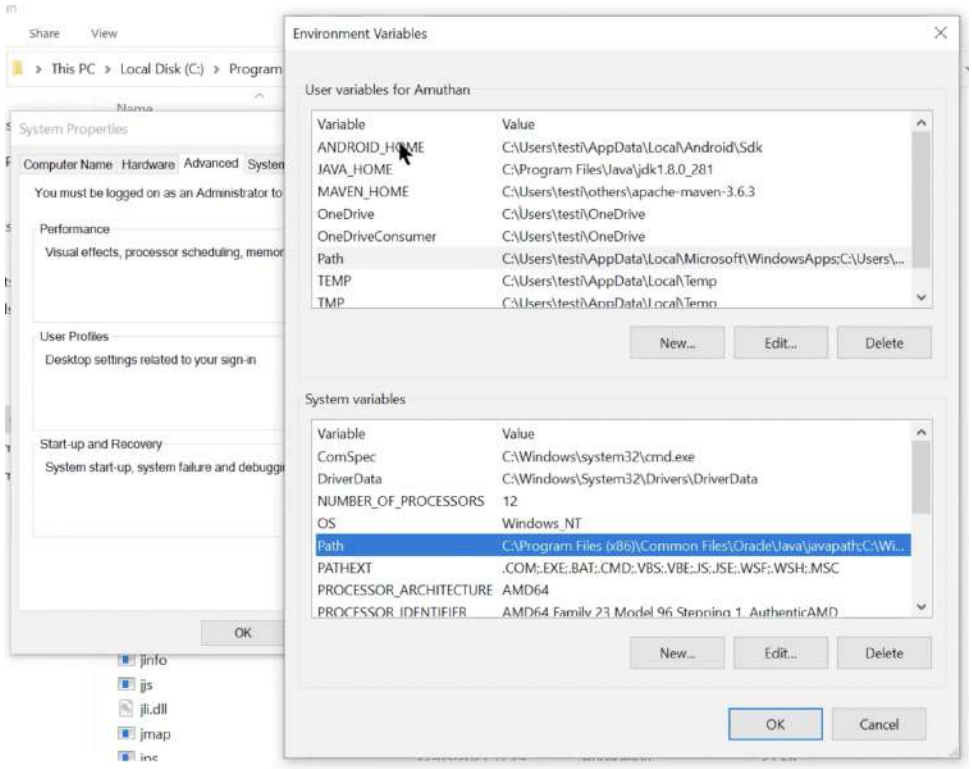
c.

d. Test.java -> Human Readable Code
e. Test.class ->
    i. Machine Readable Code
    ii. Platform Independent
f. Commands:
    i. To compile the Test.java file
        1. **javac Test.java**
            a. This file(Test.java) will get compiled and a new file (Test.class) will be generated.
            b. This file (Test.class) is platform-independent.
    ii. To execute/run Test.class file
        1. **java Test**

---

5. **Variables:**
    a. WIN:
        i. User variables - works only for specific User
        ii. System variables - works for all the User who logs into the machine

iii.



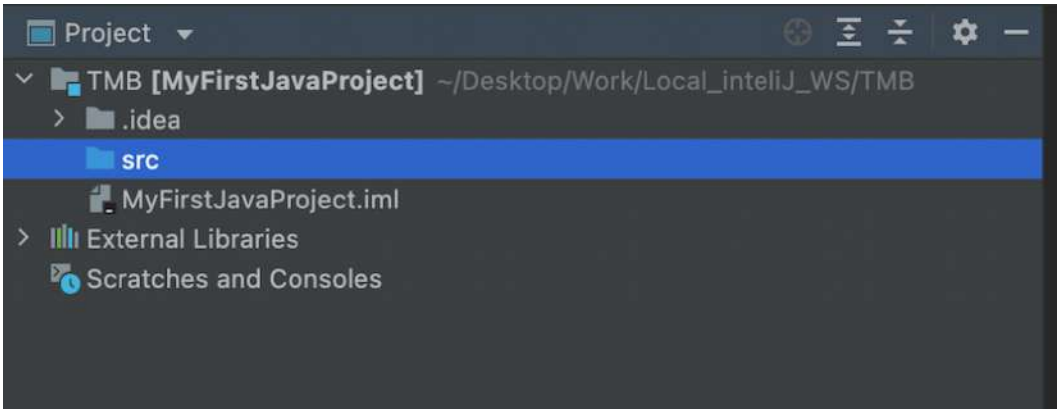
---

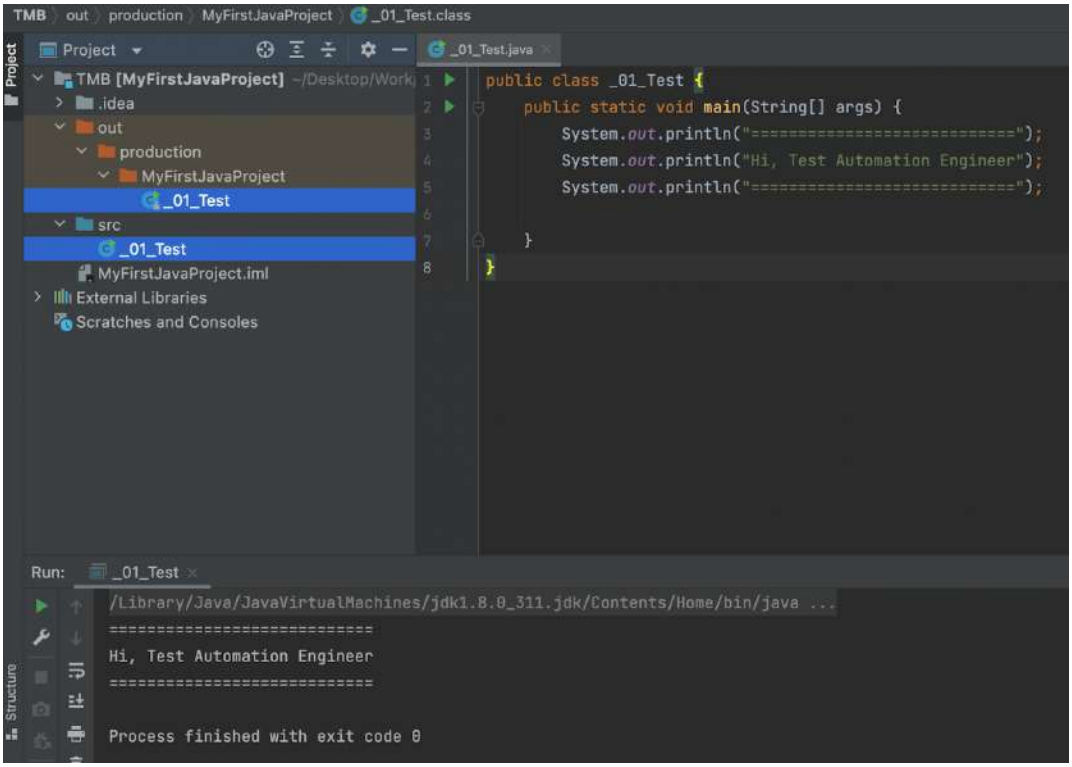
6. **IDE:**
    a. Integrated Development Environment
        i. It comes with many features
            1. Highlights the Erro before you compile the code
            2. Can integrate other tools as plugins
                a. TestNG
                b. Cucumber
    b. Example:
        i. Eclipse
        ii. IntelliJ
    c. Why IntelliJ?
        i. It has really cool features in comparison of Eclipse
    d. Download IntelliJ:
        i. https://www.jetbrains.com/idea/download/#section=mac
        ii. You can download the Community version

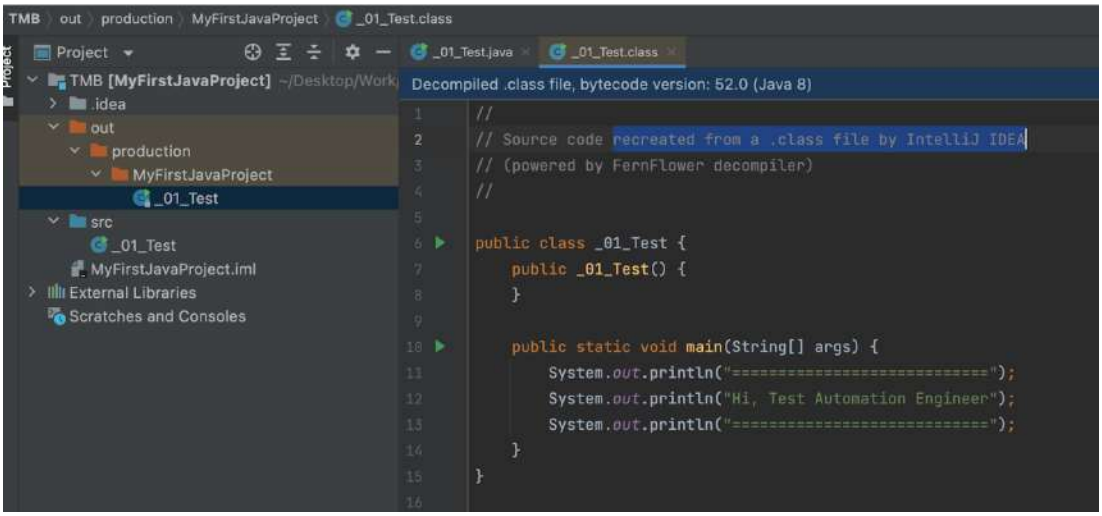
=====2_First Java Code in IntelliJ | int | String | Arithmetic Operators=======

---

1. **Create a new Java Project in intelliJ IDE:**

a. 

b. 

c. 

d. 

e. Program exit with code 0 is **OK**

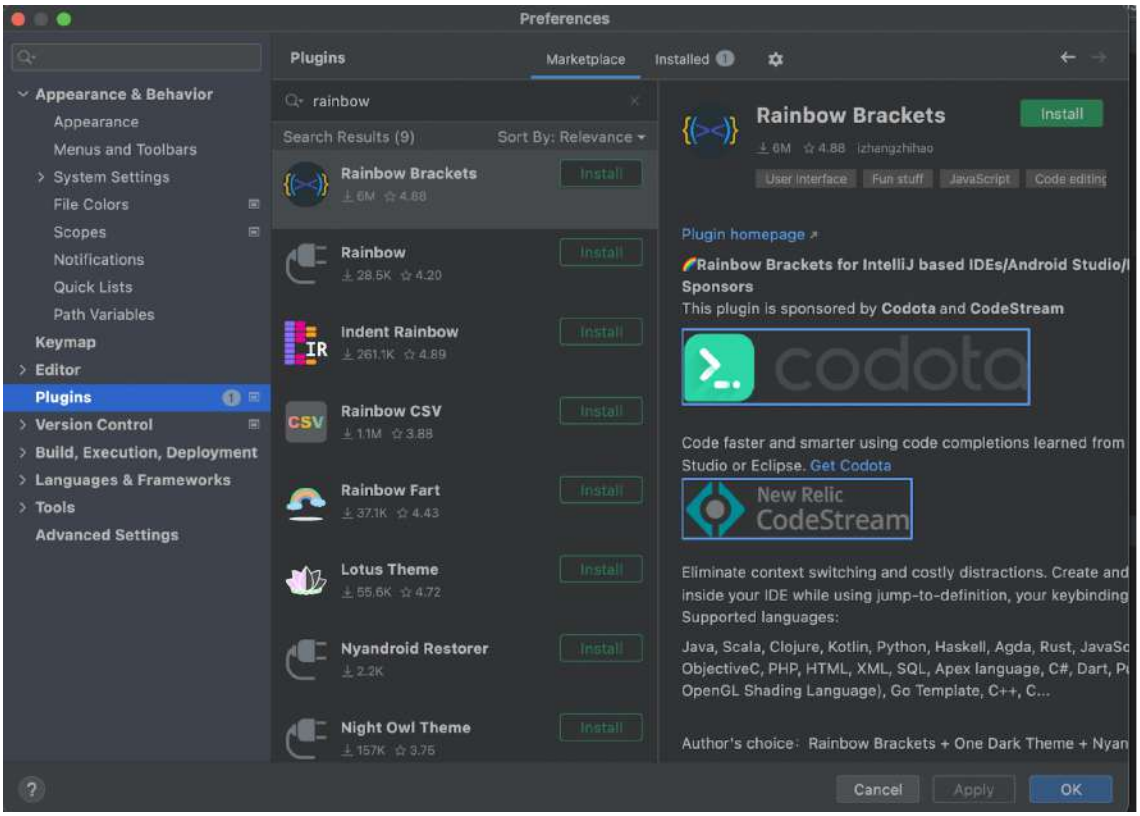
========3_If | If Else | For loop | BODMAS | AND - OR Operator==========

---

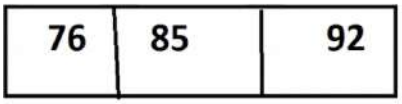
1. BODMAS
    a. BODMAS is an acronym to help children remember the **order of mathematical operations** – the correct order in which to solve maths problems. Bodmas stands for B-Brackets, O-Orders (powers/indices or roots), D-Division, M-Multiplication, A-Addition, S-Subtraction.

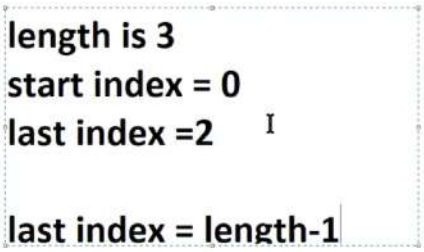
========4_Assignment Solutions | Methods | Arrays==========

---

1. Plugin:
    a. Rainbow Brackets
    b. 

2. Array:
    a. Collection of similar data type
    b. Contiguous memory location
    c. Issue -> Fixed-size





    d.

=======5_Selenium Getting Started | Writing First Selenium Script========

---

1. Each website has its DOM
2. Developers had written code for
    a. How web elements will be rendered over Web Page
    b. Web Elements:
        i. TextBoxes
        ii. Buttons
        iii. Links
        iv. Checkboxes
        v. Radio buttons
        vi. Image

========6_Locating Strategies | Mastering XPath==========

---

1. XPath types:
    a. Absolute
        i. Lengthy and Error-prone
        ii. Starts-with /
    b. Relative
        i. Recommended to use
        ii. Starts-with //

```
xpath

absolute xpath and relative xpath

absolute xpath --> lengthy ->error prone

/one level down --> parent --child relationship

//realtive xpath --> starting from root ie)html
```

2. `//tagname[@attribute='value']`

3.
```
//a --> link
//button --> button
//input --> textbox
//table -->web table
//tr --> row in a table
//td --> column in a table
```

========7_Relative XPath | Double Slashes==========

========8_Assigment Solution | FindElements | Arraylist==========

---

1. Amazon

========9_Dynamic XPath | Method Overloading | Manipulating String | Sharing Java Project==========

1.
```
static void click(WebDriver driver, String locatorType, String locatorvalue) throws InterruptedException {
    Thread.sleep( millis: 2000);
    if(locatorType.equalsIgnoreCase( anotherString: "xpath")){
        driver.findElement(By.xpath(locatorvalue)).click();
    } else if(locatorType.equalsIgnoreCase( anotherString: "id")){
        driver.findElement(By.id(locatorvalue)).click();
    } else if(locatorType.equalsIgnoreCase( anotherString: "linkText")){
        driver.findElement(By.linkText(locatorvalue)).click();
    }
}
```

2.

```
static void click(WebDriver driver, By by) throws InterruptedException {
    Thread.sleep( millis: 2000);
   driver.findElement(by).click();
}
static void click(WebElement element) throws InterruptedException {
    Thread.sleep( millis: 2000);
    element.click();
}
```

---

1. **Dynamic XPaths:**
   i.
   ii.

```
String oldXpath="//*[text()='TEST']";
System.out.println(oldXpath);
System.out.println(oldXpath.replace( target: "TEST", replacement: "Laptop"));
System.out.println(oldXpath.replace( target: "TEST", replacement: "Mobiles"));
```

=====10_Select Dropdown | Organising Locators | Maven Introduction======

---

1. **Dropdown:**
   a. Bootstrap dropdown
   b. Select dropdown

```
<select name="searchDirectory[job_title]" id="searchDirectory_job_title"> == $0
    <option value="0">All</option>
    <option value="26">Automation Tester</option>
    <option value="31">Automation Tester 1</option>
    <option value="27">BTest</option>
    <option value="33">CEO 1922164</option>
    <option value="1">Chief Executive Officer</option>
    <option value="2">Chief Financial Officer</option>
    <option value="16">Content Specialist</option>
    <option value="17">Customer Success Manager</option>
    <option value="12">Database Administrator</option>
    <option value="28">EA</option>
    <option value="3">Engineer</option>
    <option value="20">Finance Manager</option>
```

   i.
   ii. To handle this type of Dropdown, we have a class “Select” provided by Selenium WebDriver API
   iii. Methods in Select class
      1. selectByValue()

2. selectByVisibleText()
3. selectByIndex()
4. getOptions()

2. Maven:
    a. Build and Dependency Management tool
    b. Automatically creates a structure for project
        i. Developers:
            1. main java - - > Development code
            2. main resources - - > Configuration files
            3. test java - - > unit tests - - > code written to verify whether the created methods are working fine
            4. test resources - - > Configuration files required to perform Unit testing
        ii. Testers:
            1. main java - - > anything that is not test
            2. test java - - > automated tests
            3. test resources - - > Configuration files - - > Excel (contains Test data)

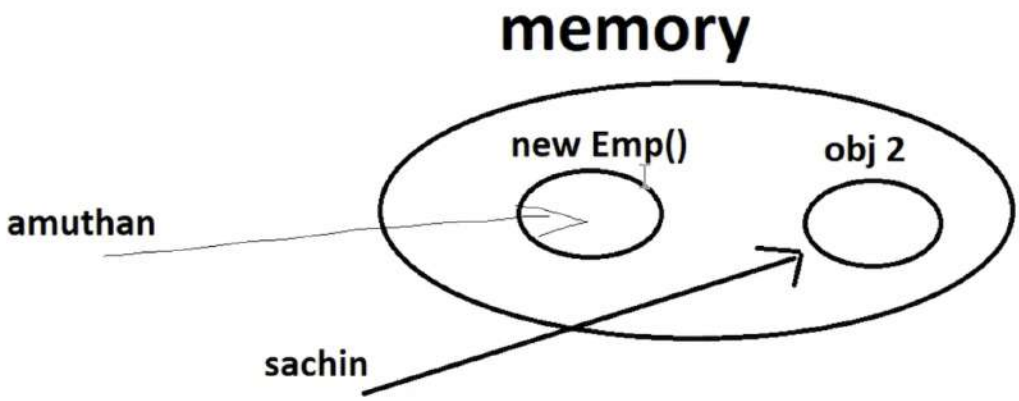
```
//main java  --> development code --> shipped to production
//main resources --> configuration files
//test java  --> unit tests --> code written to verify whether the methods they created are working fine
//test resources -> all the config files needed to perform unit test

//test java --> automated tests
//test resources --> config files, excel --> run your automated tests
//main java - anything that is not tests
//main resources --> not used in test automation
```

========11_WebDriverManager| Classes and Objects==========

---

1. Classes and Objects:
   a.
   ```
   public class Employee {  //ABCD

       int id;
       String firstName;
       String lastName;
       int age;
       String companyName;
   ```
      i. id, firstName, lastName, age, companyName are
         1. Field
         2. Variables
         3. Data Member

   b.
   ```
   public static void main(String[] args) {
       //Create a object --> Create a new employee

       Employee amuthan = new Employee();

       //Employee -> class name
       //amuthan -->reference variable
       //new Employee() --> object

       // new --> keyword --> create a new object
       //Employee() --> constructor
   }
   ```



   c.
   d. Each Object has its own properties
2. Class is like a Blueprint that an Object follows
3. Classes can have:
   a. Variables/Fields/Data Members
      i. What they possess
   b. Methods
      i. What they can do
4. Default values for:
   a. int -> 0
   b. String -> null
   c. Boolean - false
5. 

========12_All about Constrcutors==========

---

1. **Constructor:**
    a. It helps to construct the object of a class
    b. Same name as class name
    c. Special method
    d. It has no explicit return type
    e. Types:
        i. Default
        ii. Parameterized

3. Default Constructor is available in every class by default
    a. You can check this in the .class file (using IntelliJ IDE)
    b.

```
Decompiled .class file, bytecode version: 52.0 (Java 8)
1  //
2  // Source code recreated from a .class file by IntelliJ IDEA
3  // (powered by FernFlower decompiler)
4  //
5
```

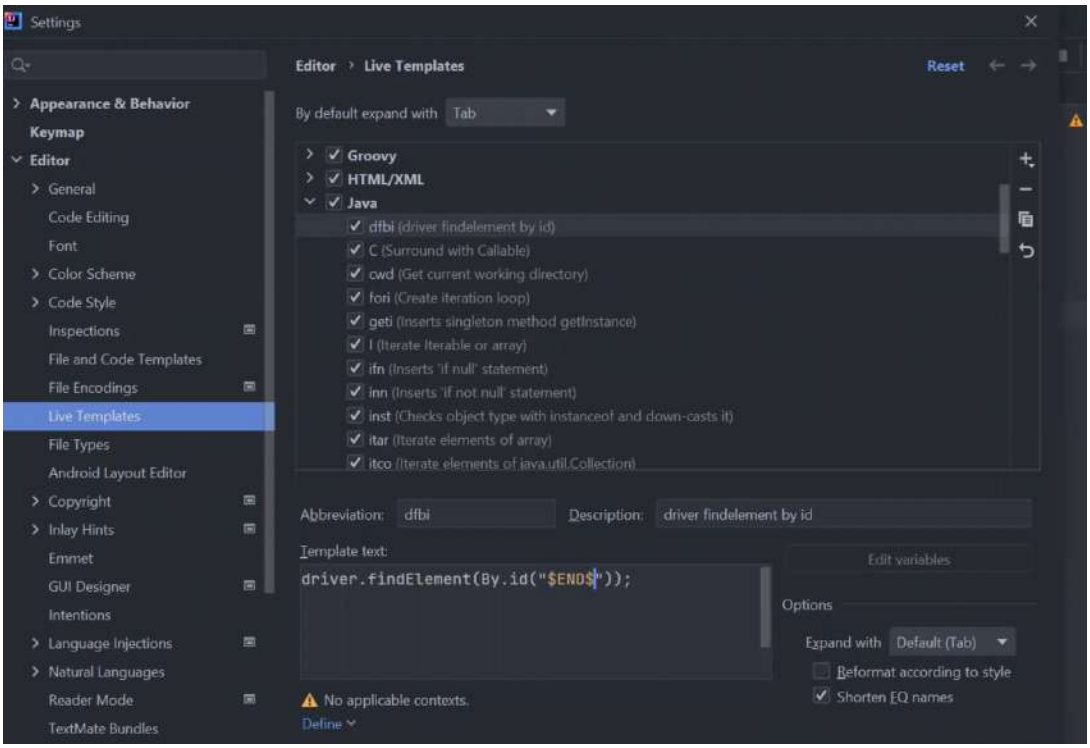
    c.

```
loyee.java × MyFirstMavenTest.java × StringConstants.java × StringConstants.class ×
iled .class file, bytecode version: 52.0 (Java 8)
/.../

package constants;

public class StringConstants {
    public static String url = "https://opensource-demo.orangehrmlive.

    public StringConstants() {
    }
}
```

========13_Actions class | Drag and Drop | Frames==========

---

1. Templates:
    a. Pre-Defined:
        i. main - - > main method
        ii. sout - - > System.out.println
    b. Custom:
        i. Settings -> Live Templates -> Java



        ii. Dfbi - - > driver.findElement(By.id(“”))

========14_File Upload | Inheritance | Encapsulation | Abstration=========

---

1. File Upload:
    a. Check
        i. DOM ->
            1. Tag name is **input**
            2. **type=”file”**

```
▼<div id="uploadmode1" style="display:block">
  ▼<div id="filelist">
    ▼<div id="file_wraper0">
        <input type="file" name="uploadfile_0" class="upload_txt" id="uploadfile_0" size="40"> == $0
        <br>
      ▶<div class="upload_options">…</div>
      </div>
    </div>
  </div>
```

        ii.

2.

```
@Test
public void fileUpload() {
    driver.get("https://demo.guru99.com/test/upload/");
    WebElement btn_FileUpload = driver.findElement(By.id("uploadfile_0"));

    //To upload File (present in Project root location)
    String fileToUpload = System.getProperty("user.dir")+"/FileUpload.txt";
    btn_FileUpload.sendKeys(fileToUpload);
```

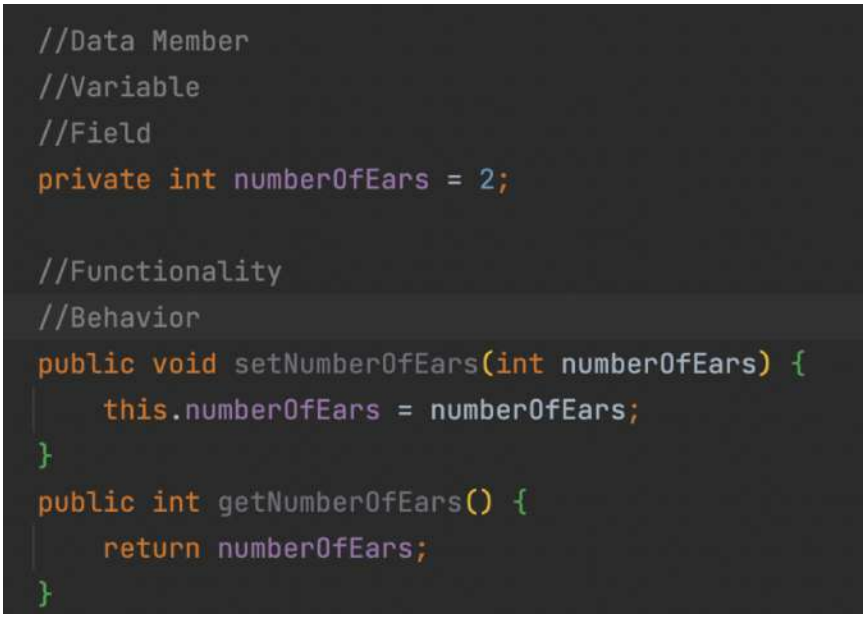
---

1. **OOPS concepts:**
    a. Encapsulation
    b. Inheritance
    c. Abstraction
    d. Polymorphism

---

1. **Encapsulation**
    a. Private Variables/Fields/Data Members
    Public methods
    b. Used to restrict the user to directly access the variable
    c. We get more control over the variables
        i. We can put condition checks before setting the value as well
            1. In Setters
    d. Hiding the implementation details and exposing only the functionality

e.



```
//Data Member
//Variable
//Field
private int numberOfEars = 2;

//Functionality
//Behavior
public void setNumberOfEars(int numberOfEars) {
    this.numberOfEars = numberOfEars;
}
public int getNumberOfEars() {
    return numberOfEars;
}
```

---

2. **Inheritance**
    a. Base and Derived
       Super and Sub
       Parent and Child
    b. **"is-a"** relationship
    c. Dog extends Animal
       Cat extends Animal
        i. Dog and Cat are child classes
        ii. Animal is parent class
        iii. Dog **is-a** Animal
             Cat **is-a** Animal
    d. Parent class will have similarities which are common in Child class
    e. NOTE:
        i. Animal animal1 = new Dog();
            1. The object for the Dog
            2. A reference type is now Parent class (Animal)
        ii. List list1 = new ArrayList();
        This type of implementation is required when we want to change the implementation at runtime.

    f. We have 2 concepts
        i. Interface
            1. Used for 100% Abstraction
            2. Methods will have no body/implementation
            3. If any class_A **implements** Interface_B

        a. class_A has to **override** the methods present in Interface_B
    ii. Abstract classes
        1. Used for partial Abstraction
        2. Abstract methods will not have any body/implementation
        3. If any class_A **extends** Abstract class_B
            a. class_A has to **override** the abstract methods present in Abstract class_B

g. Summary for Inheritance:

- i. We use Inheritance - - >
    1. Similarities in the class
    2. Code Re-usability
    3. is-a relationship
    4. Object with Parent Reference type - - > Liberty to create a required child object
    5. Method in parent class - - > Do not want this behavior - - > Child can implement its own behavior
    6. This is Method Overriding - - > Dynamic Polymorphism
- ii. Method Overloading - - > Static Polymorphism

======15_Multi Level Inheritance | Final keywords | Constructor calls======

---

1. Typecasting
    a. UpCasting
        i. Happens implicitly
        ii. Example:
            1. Animal animal = new Dog();
    b. DownCasting
        i. Has to be done explicitly
            1. Animal animal = new Dog();
               (Dog(animal)).bark();

---

1. **Multi-Level Inheritance:**
    a. Dog extends DomesticAnimal
    b. DomesticAnimal extends Animal

c.

```
Dog dog = new Dog();
dog.isDomestic();

 DomesticAnimal dog1 = new Dog();
 dog1.isDomestic();
 ((Dog) dog1).bark();

 Animal dog2 = new Dog();
 dog2.isDomestic();
 ((Dog)dog2).bark();
```

---

1. **Final Keyword**
    a. Variable - - > value can not be changed
    b. Method - - > can not be overridden
    c. Class - - > can not be inherited
        i. If any class tries to extend final class, then,
            1. Message: Cannot inherit from final class

---

1. **Constructor calling Order:**
    a. When we create the object of Child class,
        i. Constructor of parent class gets called first
        ii. Grandfather -> Father -> Son

========16_Super Keyword, Problems with Multiple Inheritance==========

---

1. Abstract class:
    a. We can not create instance/object of Abstract class
        i. These classes are not Concrete
        ii. Message: Abstract class can not be instantiated
    b. If we have even 1 abstract method, then, that class has to be declared as an Abstract class.
    c. Using @Override annotation is not mandatory
    This is juts for our understanding.

d. Abstract methods defines the Skeleton

e.
```
public abstract class RBI {

    public abstract void withdrawl();
    public abstract void deposit();

    //defines the skeleton

}
```

---

1. **Access Modifiers:**

a.

Modifier	Class	Package	Subclass	Global
Public	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	No
Default	Yes	Yes	No	No
Private	Yes	No	No	No

b.
```
//public --> accessed by any class
//private --> it can be accessed only within the class
//protected --> it can be accessed only by the class and its child classes
//default or package protected
```

========17_Static Variable and Method | Interface | Static Blocks =========

---

1. **Multiple Inheritance:**
    a. Not possible with classes
    b. Possible with Interface
        i. ClassA **implements** InterfaceA, InterfaceB, InterfaceC

---

1. **Static**:
    a. Keyword
    b. Can be used with variable and method
    c. Used for **Memory Management**
    d. Static variables are **common/shared for all the Objects**
    e. They are not specific to particular objects.
    f. You can access static variables and methods directly with class name
        i. No need to create an object and call
    g.

```
public class Student {

    public String name; //when each object have diff value
    public int rollNum;
    public static String schoolName="abcd"; //common for all objects
```

    h. Static methods deal with static variables only

---

1. **Interface:**
    a. Only have abstract methods
        i. No body/implementations
    b. Used to define the Skeleton/Rules
    c. Methods -> public static
    Variables -> public static final
    d. Interface does not have any Constructor
    e. InterfaceA **extends** InterfaceB
    ClassA **extends** ClassB
    ClassA **implements** InterfaceB
    ClassA **implements** InterfaceA, InterfaceB

---

1. **Static Block:**
    a. Executes even before main() method

======18_Scanner Class, String Arguments in Main, TestNG basics========

---

1. String Arguments in main() method:
2.

```
for (int i = 0; i < args.length; i++) {
    System.out.println(args[i]);
}
```

---

1. **TestNG:**
    a. Open source Testing framework
    b. Reporting Capability
        i. Tests execution
            1. Project location ->
                a. test-output ->
                    i. Emailable-reports.html
                    ii. Index.html
    c. Annotations:
        i. Test
    d. Properties with Annotation:
        i. priority

========19_TestNG Annotations in Details | TestNG XML | Groups=========

---

1. @BeforeSuite // create DB connections, Reports initialization
2. @BeforeTest
3. @BeforeClass
4. @BeforeMethod //launch the Browser
5. @Test
6. @AfterMethod // to quit the Browser
7. @AfterClass
8. @AfterTest
9. @AfterSuite // close DB connection, Reports flush

Suite - - > Test - - > Class - - > Method

1. **Suite is a collection of multiple Tests**
2. **Test is a collection of multiple Classes**
3. **Class is a collection of multiple Methods**

---

1. All tests need to be independent.

======20_JavascriptExecutor | TakesScreenshot | Variable Arguments=====

---

1. **JavascriptExecutor:**
    a. It is an interface with the help of which we can execute Javascript code using Selenium WebDriver.
    b. Methods:
        i. executeScript()
2. JavascriptExecutor jse = (JavascriptExecutor) driver;
   This is an example of **Downcasting**.

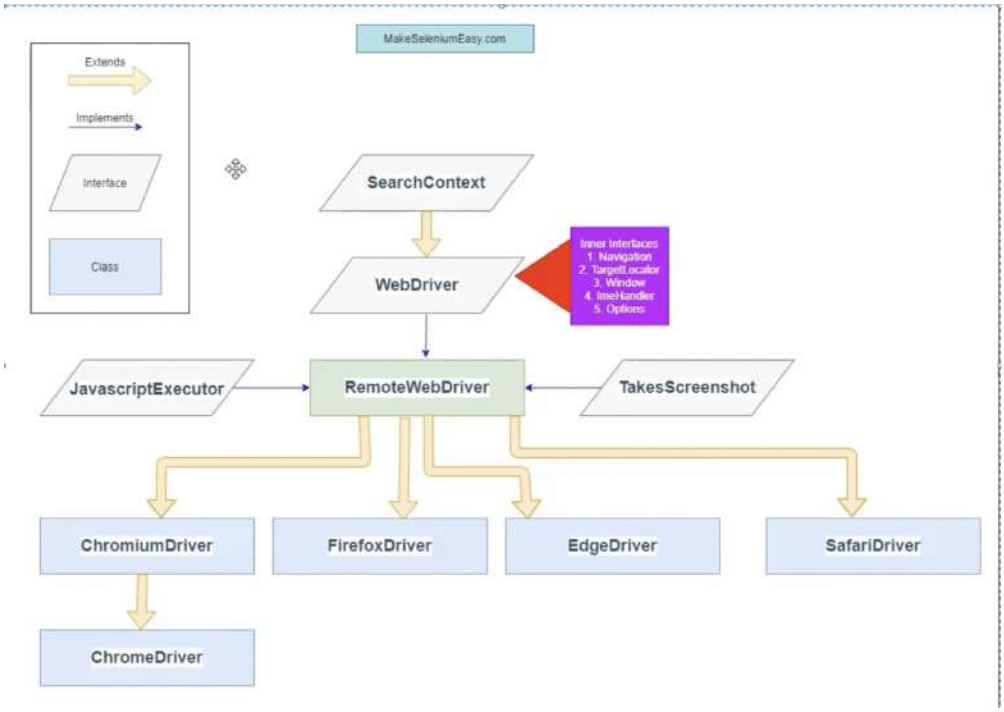
---

1. **TakesScreenshot:**
    a. It is an interface
    b. It is used to take the screenshot of the page
    c. Method:
        i. getScreenshotAs()

---

1. **WeDriver Hierarchy:**

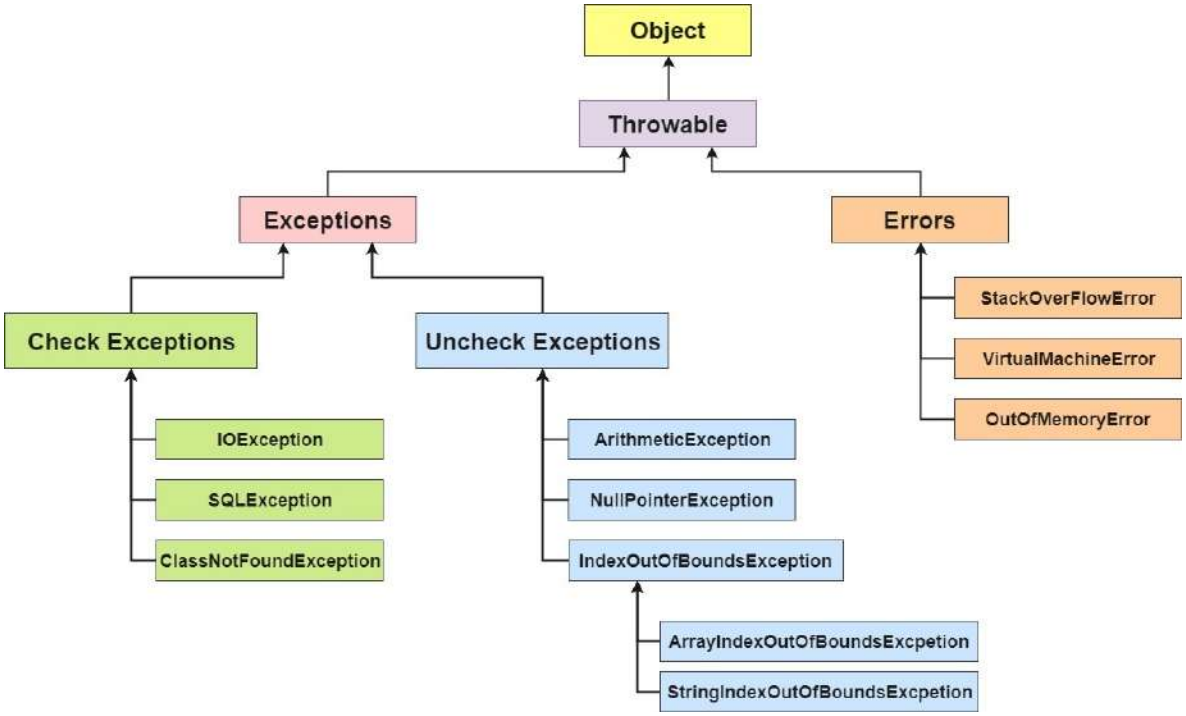
2.



========21_Exception Handling in Java | All you need to know==========

---

1. **Exceptions:**
    a. Checked Exception
        i. Compile-time Exception
        ii. InterruptedException, IOException,
    b. Unchecked Exception
        i. Run time Exception
        ii. ArithmeticException, IndexOutOfBoundsException, FileNotFoundException
        iii. Selenium Exceptions:
            1. NoSuchElementException



2. Exception Chaining

========22_2D Array | Data Provider | Extent Reports==========

---

1. **2-D Array:**

```java
public class _08_TwoDimArray {
    public static void main(String[] args) {

        int[][] twoDimArray = new int[2][2]; //[Row][Column]
        twoDimArray[0][0] = 10;
        twoDimArray[0][1] = 20;
        twoDimArray[1][0] = 30;
        twoDimArray[1][1] = 40;

        //Row
        for (int i = 0; i < twoDimArray.length; i++) {
            //Column
            for (int j = 0; j < twoDimArray.length; j++) {
                System.out.println("twoDimArray[" + i + "][" + j + "] = " + twoDimArray[i][j]);
            }
        }
```

2. All the classes are sub-classes of the Object class in Java

---

1. **Data Providers:**

a.

```java
@DataProvider
public static Object[][] getData() { //2 , 2
    return new Object[][]{
            {"Admin","admin123"},
            {"Admin123","admin12345"}
    };

    //number of rows --> number of times it is going to run your test
    //number of columns --> number of parameters to your test method
}
```

---

1. **ExtentReports:**

a.

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

        ExtentReports extent = new ExtentReports();
        ExtentSparkReporter spark = new ExtentSparkReporter( path: "Extent_Spark.html");
        extent.attachReporter(spark);
        extent.createTest( name: "MyFirstTest")
                .log(Status.PASS,  details: "This is a logging event for MyFirstTest, and it passed!");
        extent.flush();
    }
}
```

=23_TestNG and Extent Reports Integration | Excel Reading using Apache POI=

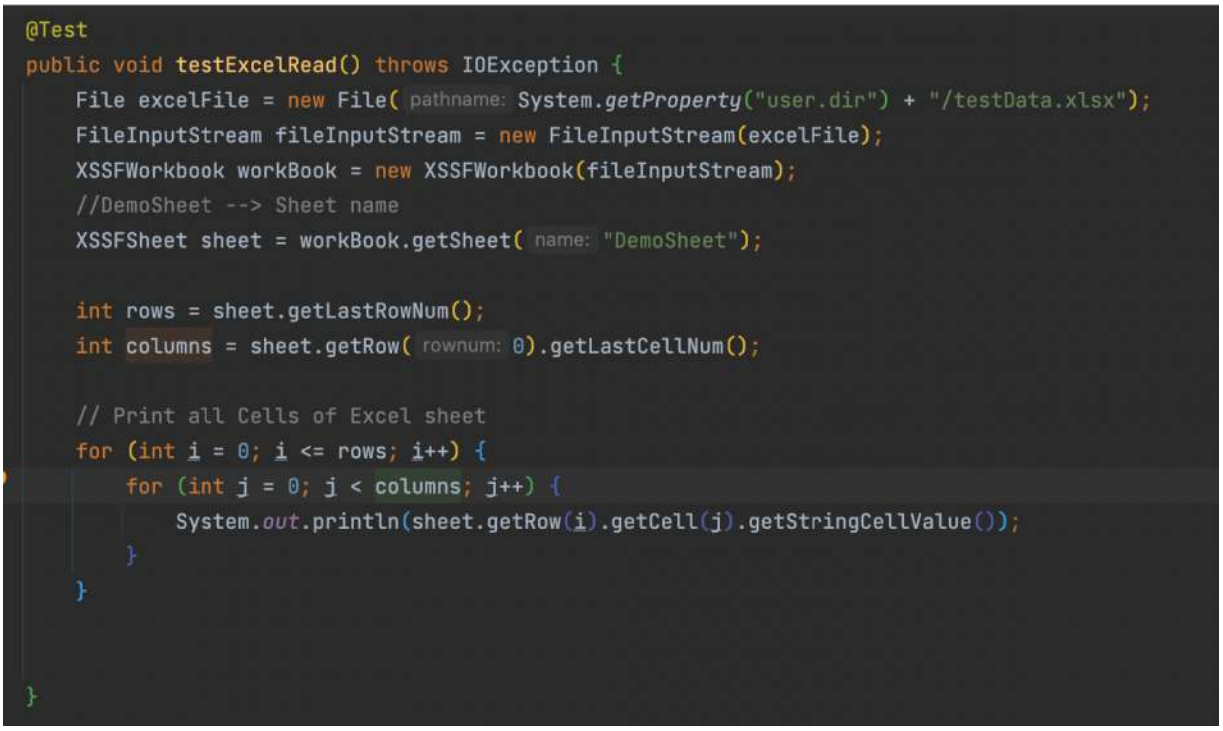
---

1. Apache POI

a.

```xml
<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>5.1.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.1.0</version>
</dependency>
```

b.



```
@Test
public void testExcelRead() throws IOException {
    File excelFile = new File( pathname: System.getProperty("user.dir") + "/testData.xlsx");
    FileInputStream fileInputStream = new FileInputStream(excelFile);
    XSSFWorkbook workBook = new XSSFWorkbook(fileInputStream);
    //DemoSheet --> Sheet name
    XSSFSheet sheet = workBook.getSheet( name: "DemoSheet");

    int rows = sheet.getLastRowNum();
    int columns = sheet.getRow( rownum: 0).getLastCellNum();

    // Print all Cells of Excel sheet
    for (int i = 0; i <= rows; i++) {
        for (int j = 0; j < columns; j++) {
            System.out.println(sheet.getRow(i).getCell(j).getStringCellValue());
        }
    }


}
```

======24_Excel-Dataprovider Integration | Listeners | Set and Map========

---

1. **Apache POI and DataProvider:**

a.

```
@Test(dataProvider = "loginTestData")
public void loginTest(String username, String password) {
    System.out.println("-----------");
    System.out.println("Username: " + username);
    System.out.println("password: " + password);
}
```

b.

```java
@DataProvider(name = "loginTestData")
public Object[][] getTestDataForLogin() throws IOException {
//      return new Object[][]{
//              {"uname1", "pwd1", 1234},
//              {"uname2", "pwd2", 3456},
//              {"uname3", "pwd3", 4876},
//              {"uname4", "pwd4", 9087}
//      };
    File excelFile = new File( pathname: System.getProperty("user.dir") + "/testData.xlsx");
    FileInputStream fileInputStream = new FileInputStream(excelFile);
    XSSFWorkbook workBook = new XSSFWorkbook(fileInputStream);
    //DemoSheet --> Sheet name
    XSSFSheet sheet = workBook.getSheet( name: "DemoSheet");

    int rows = sheet.getLastRowNum();
    int columns = sheet.getRow( rownum: 0).getLastCellNum();

    Object[][] object = new Object[rows][columns];

    // Print all Cells of Excel sheet
    //i=1 --> we want to skip the 1st row
    for (int i = 1; i <= rows; i++) {
        for (int j = 0; j < columns; j++) {
          //  System.out.println(sheet.getRow(i).getCell(j).getStringCellValue());
            object[i - 1][j] = sheet.getRow(i).getCell(j).getStringCellValue();
        }
    }
    return object;
}
```

---

1. Listeners
    a. ITestListener
        i. In latest version of TestNG,
            1. Methods inside the ITestListener interface are default methods
            2. It is not mandatory for sub-classes to implement those methods

```java
package org.testng;

/**
 * A listener for test running.
 * Author: Cedric Beust, Alexandru Popescu, Hani Suleiman
 */
public interface ITestListener extends ITestNGListener {

  /**
   * Invoked each time before a test will be invoked. The ITestResult is only partially filled with the
   * references to class, method, start millis and status.
   * Params: result – the partially filled ITestResult
   * See Also: ITestResult.STARTED
   */
  default void onTestStart(ITestResult result) {
    // not implemented
  }

  /**
   * Invoked each time a test succeeds.
   * Params: result – ITestResult containing information about the run test
   * See Also: ITestResult.SUCCESS
   */
  default void onTestSuccess(ITestResult result) {
    // not implemented
  }

  /**
   * Invoked each time a test fails.
   * Params: result – ITestResult containing information about the run test
   * See Also: ITestResult.FAILURE
   */
  default void onTestFailure(ITestResult result) {
    // not implemented
  }
```

ii. In the older version of TestNG,
    1. Methods inside the ITestListener interface were abstract methods
    2. It was mandatory for sub-classes to implement those methods

---

1. **How can you remove the duplicates from the list?**
    a. Add elements inside to a Set.
    b. Set does not allow duplicates.
        i. Set<String> set = new HashSet<>(list);
        ii. Set does not maintain the Insertion order.

---

1. **Map:**
    a. Key-Values

b. No duplicate keys are allowed
c. If we have 2 values with the same Key, then, the new value replaces the old value
d. Map also does not maintain the Insertion order
e.

```
Map<String,String> countriesCapital = new HashMap<>();
countriesCapital.put("India","New Delhi");
countriesCapital.put("China","Beijing");

System.out.println(countriesCapital.get("USA"));

Set<String> countryNames = countriesCapital.keySet();

for(String temp: countryNames){
    System.out.println(temp +":"+countriesCapital.get(temp));
}
```

========25_Dataprovider with Hashmap | Property File Reading==========

---

1. **DataProvider with HashMap**

a.

username	password	firstname	lastname		
Admin123	admin	sjdfs	sdjkln		map1
username	password	firstname	lastname		
Admin123	1234	sjdkfns	sdlkjf		map2
username	password	firstname	lastname		
Admin123	sdkfjs	skdjfns	skdjfsw		map3

b.

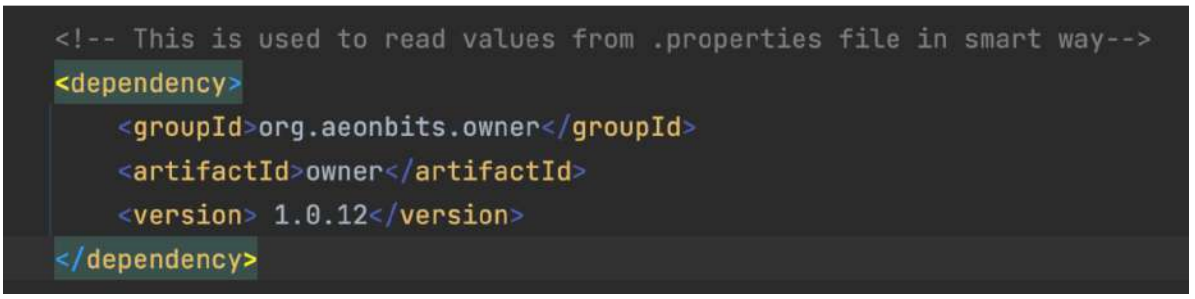
```java
@DataProvider(name = "loginTestData_Smart")
public Object[][] getTestDataForLogin_Smart() throws IOException {
    File excelFile = new File( pathname: System.getProperty("user.dir") + "/testData.xlsx");
    FileInputStream fileInputStream = new FileInputStream(excelFile);
    XSSFWorkbook workBook = new XSSFWorkbook(fileInputStream);
    //DemoSheet --> Sheet name
    XSSFSheet sheet = workBook.getSheet( name: "DemoSheet_2");

    int rows = sheet.getLastRowNum();
    int columns = sheet.getRow( rownum: 0).getLastCellNum();

    //Smart way
    Object[][] object = new Object[rows][1];
    Map<String, String> map;

    // Print all Cells of Excel sheet
    //i=1 --> we want to skip the 1st row
    for (int i = 1; i <= rows; i++) {
        map = new HashMap<>();
        for (int j = 0; j < columns; j++) {
            String key = sheet.getRow( rownum: 0).getCell(j).getStringCellValue();
            String value = sheet.getRow(i).getCell(j).getStringCellValue();
            map.put(key, value);
        }
        object[i - 1][0] = map;
    }
    return object;
}
```

c.

```java
@Test(dataProvider = "loginTestData_Smart")
public void loginTest_Smart(HashMap<String, String> testData) {

    //Username  Password    FirstName   LastName    Address City -> COLUMNS in .xlsx file
    String username = testData.get("Username");
    String password = testData.get("Password");
    String firstName = testData.get("FirstName");
    String lastName = testData.get("LastName");
    String address = testData.get("Address");
    String city = testData.get("City");

    System.out.println("-----------");
    System.out.println("Username: " + username);
    System.out.println("password: " + password);
    System.out.println("firstName: " + firstName);
    System.out.println("lastName: " + lastName);
    System.out.println("address: " + address);
    System.out.println("city: " + city);
}
```

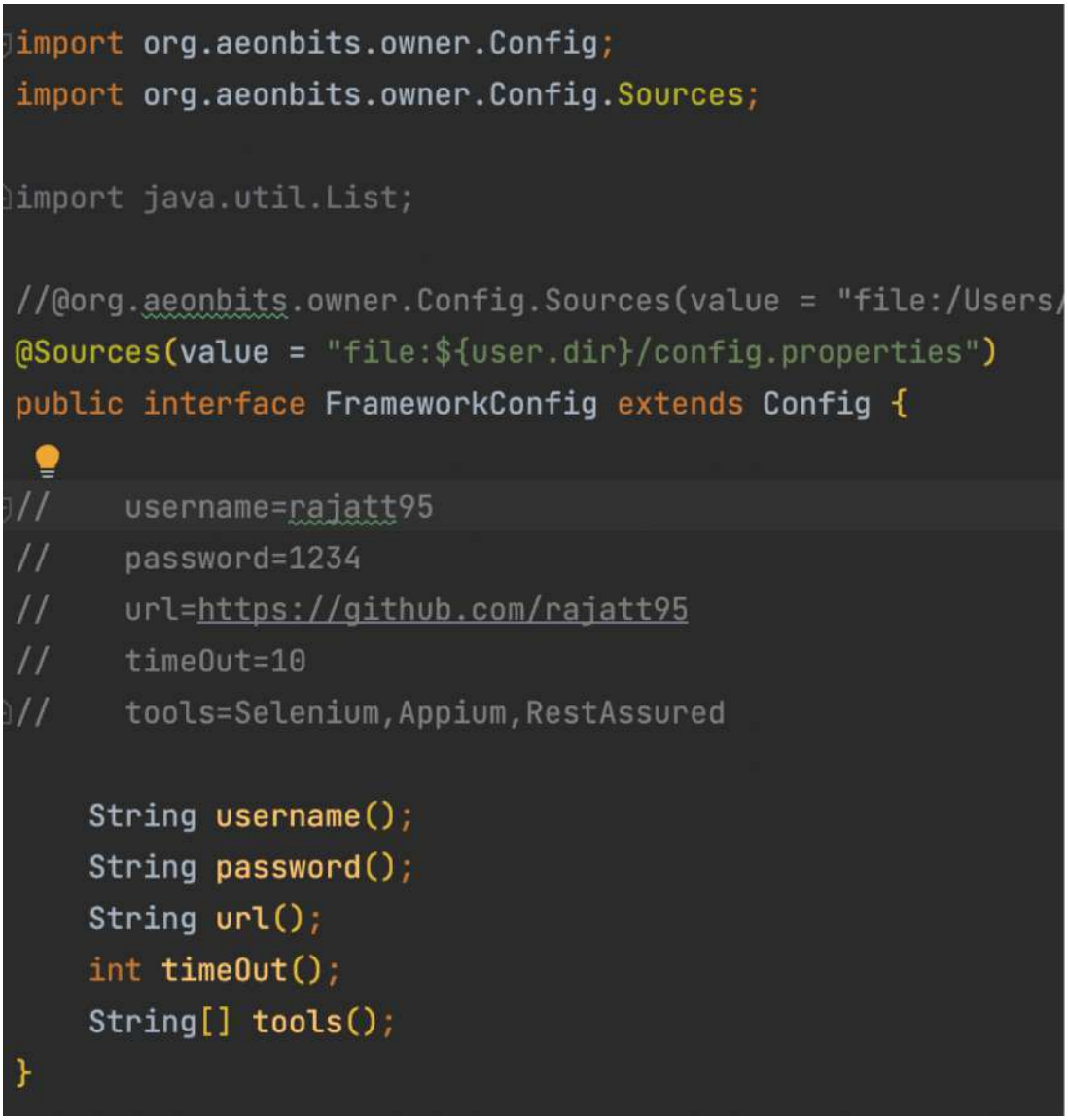
---

1. **Owner library:**
    a. To read values from the .properties file in a smart way
    b.



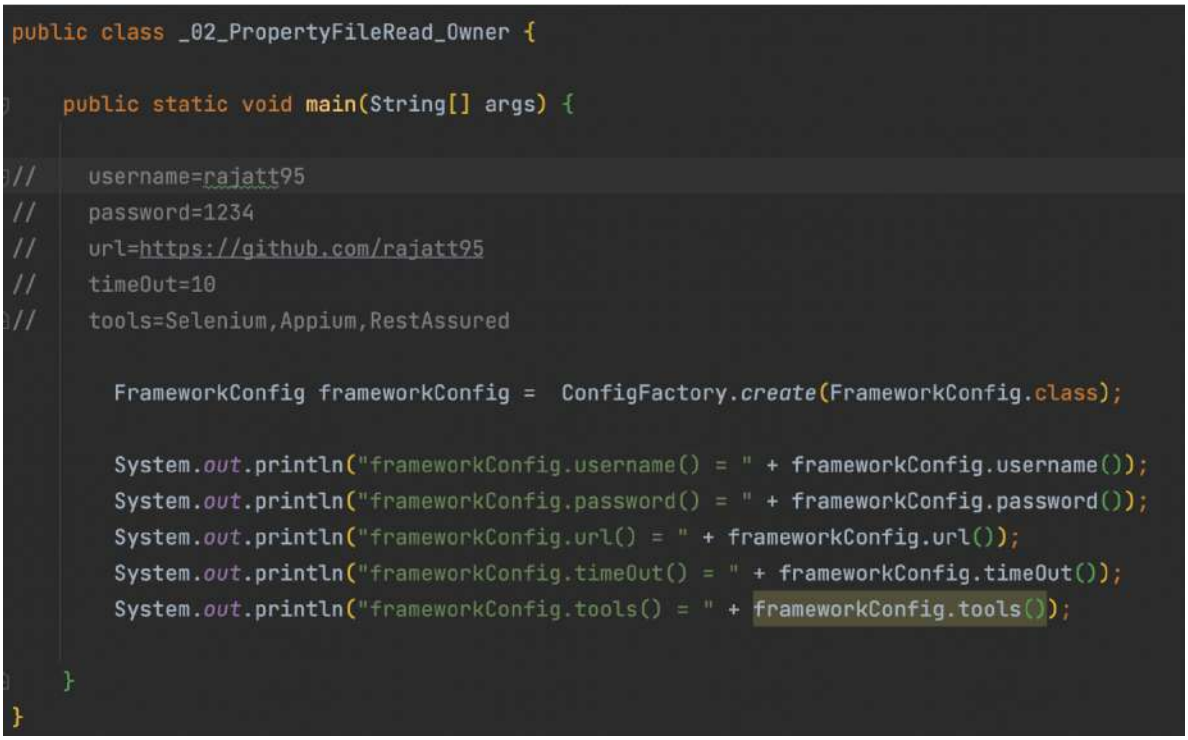
```
<!-- This is used to read values from .properties file in smart way-->
<dependency>
    <groupId>org.aeonbits.owner</groupId>
    <artifactId>owner</artifactId>
    <version> 1.0.12</version>
</dependency>
```

    c.



```
import org.aeonbits.owner.Config;
import org.aeonbits.owner.Config.Sources;

import java.util.List;

//@org.aeonbits.owner.Config.Sources(value = "file:/Users/
@Sources(value = "file:${user.dir}/config.properties")
public interface FrameworkConfig extends Config {

//    username=rajatt95
//    password=1234
//    url=https://github.com/rajatt95
//    timeOut=10
//    tools=Selenium,Appium,RestAssured

    String username();
    String password();
    String url();
    int timeOut();
    String[] tools();
}
```

d.



```java
public class _02_PropertyFileRead_Owner {

    public static void main(String[] args) {

//    username=rajatt95
//    password=1234
//    url=https://github.com/rajatt95
//    timeOut=10
//    tools=Selenium,Appium,RestAssured

        FrameworkConfig frameworkConfig =  ConfigFactory.create(FrameworkConfig.class);

        System.out.println("frameworkConfig.username() = " + frameworkConfig.username());
        System.out.println("frameworkConfig.password() = " + frameworkConfig.password());
        System.out.println("frameworkConfig.url() = " + frameworkConfig.url());
        System.out.println("frameworkConfig.timeOut() = " + frameworkConfig.timeOut());
        System.out.println("frameworkConfig.tools() = " + frameworkConfig.tools());

    }
}
```

====26_Waits in Selenium | ChromeOptions and Headless | Docker Intro===

---

1. Waits
    a. Implicit
    b. Explicit
        i. Polling - 500 ms
    c. Fluent

    Do not mix implicit and explicit waits.
    Go with Explicit. It is recommended to use.

---

1. **Headless:**

a.

```
ChromeOptions chromeOptions = new ChromeOptions();
//chromeOptions.addArguments("headless");
chromeOptions.addArguments("--headless");

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

---

1. **Docker:**
    a. Can execute test scripts on a browser with any specific version
        i. For Browser Compatibility testing
    b. Download it:
        i. https://www.docker.com/products/docker-desktop
    c. Terminal:
        i. docker
            This command must give other options
    d. Using Docker, we can create multiple **Containers**
    e. We'll set up a Selenium Hub and attach all the containers with it
    f. Containers are like
        i. Taking a PG and start living there

Some common problems

1. Script worked yesterday and not today?
2. Script working in my machine not in colleague/client's machine?
3. I have a requirement that I need to test my application in all the latest browser versions of chrome(80,81,82,..,89) and firefox(78,79,80,..84)
4. Less budget to onboard cloud providers.
5. Local machine configuration.

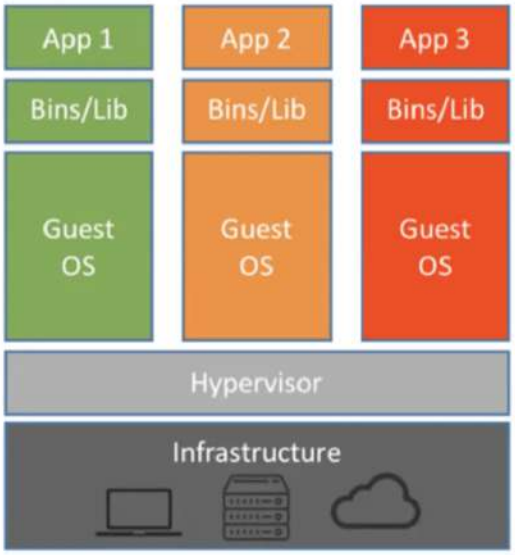
Answer is **Docker** !!

========27_Execute Test in Docker | Framework Creation Intro==========

1.

Virtualization

1. Importing a guest OS on top of Host OS.
2. Run multiple OS in different virtual machines all running on the same host.



Machine Virtualization

2.

Virtualization Advantages

•Multiple operating systems can run on the same machine

•Maintenance and Recovery were easy in case of failure conditions as image management gets easier

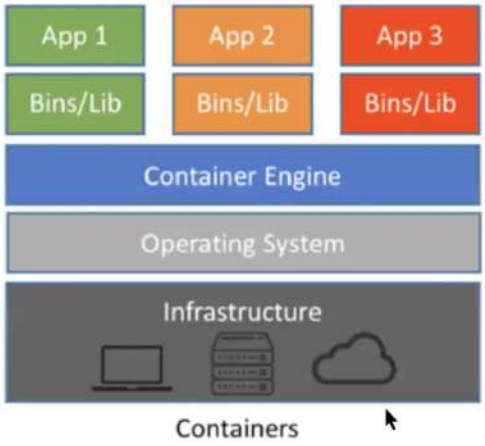
•Total cost of ownership was also less due to the reduced need for infrastructure

---

Containerization

•Containers are a method of **operating system virtualization** that allow you to run an application and its dependencies in resource-isolated processes.

•Containers can run on top of VMs



1.

Advantages of Containerization

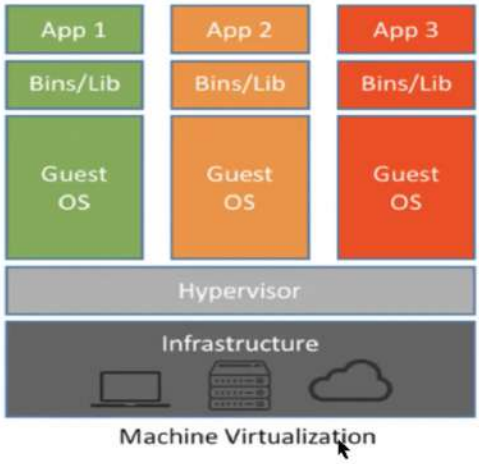
No guest OS overhead and utilizes a host's operating system.

Share relevant **libraries & resources** as and when needed unlike virtual machines.

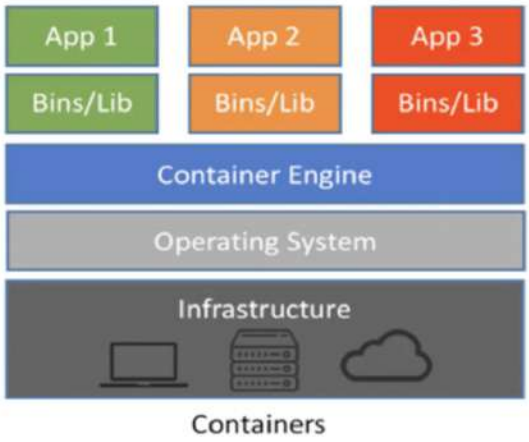
Lightweight and Faster than Virtual Machines

2.

1.



RENT A ROOM



PG (OR) AIRBNB

2.

How easy to use Docker in Test Automation

1. How easy is the setup?
2. How to execute our tests in Docker?
3. Changes to be made in our framework to execute the tests in docker
4. How easy to clean up the infra?

3.

Basics of Docker

Software World	Docker World
Software	Images
Install and run application	Containers
Github Repository	Docker registry(docker hub)

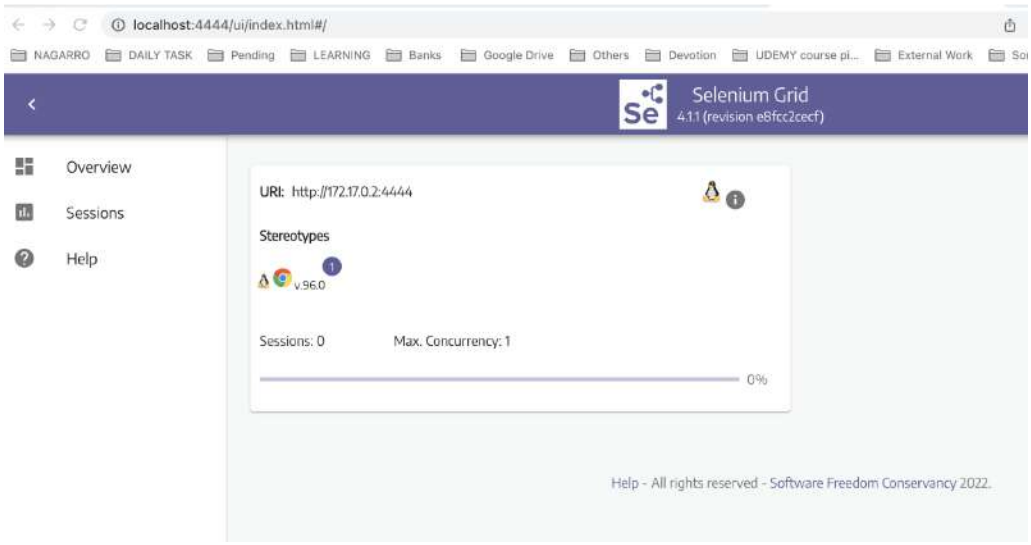
FAQ:

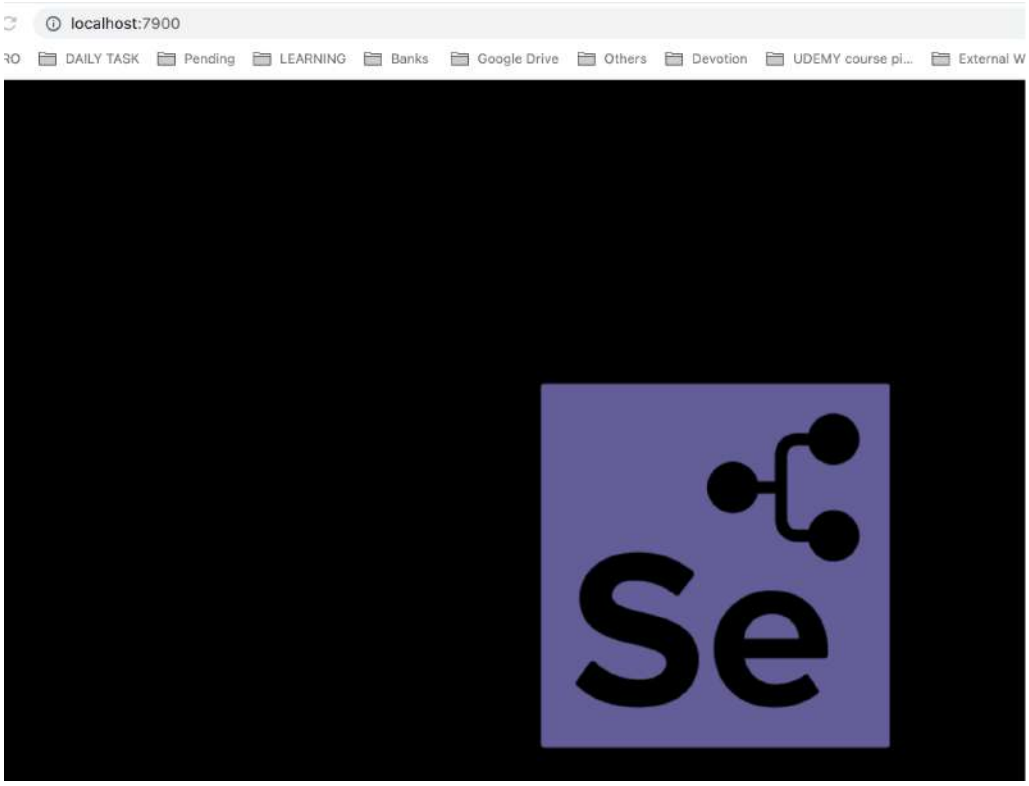
1. Linux Vs Windows containers.
2. Multiple containers from one image is possible.
3. Multiple images in a container is not possible.

---

1. Terminal:
    a. **docker ps**
        This command lists down all the Docker containers which are in running state.
2. Go to
    a. https://hub.docker.com/
    b. Search for: selenium/standalone-chrome
        https://hub.docker.com/r/selenium/standalone-chrome
        https://hub.docker.com/r/selenium/standalone-chrome/tags
3. Terminal:
    a. **docker pull selenium/standalone-chrome:96.0**
        This command will download/pull the image from Docker Hub to your local machine.
    b. **docker images**
        This command will list out all the available images in your local machine.
    c. **docker run -p 4444:4444 -p 7900:7900 selenium/standalone-chrome:96.0**
        i. Left **4444** -> **Local machine** port
        ii. Right **4444** -> **Container image** port
        This command will start/run the container.
    d. **http://localhost:4444/**
        This URL is to see the Docker container status
    e. **http://localhost:7900/**
        This URL is to see the live execution in the Docker container
        Password: secret
        Check this:
            i. https://github.com/SeleniumHQ/docker-selenium
    f. **docker start name/ID**
        This command will start the container.
    g. **docker rm name/ID**
        This command will remove the container.
    h. **docker stop name/ID**
        This command will stop the running container.
    i.

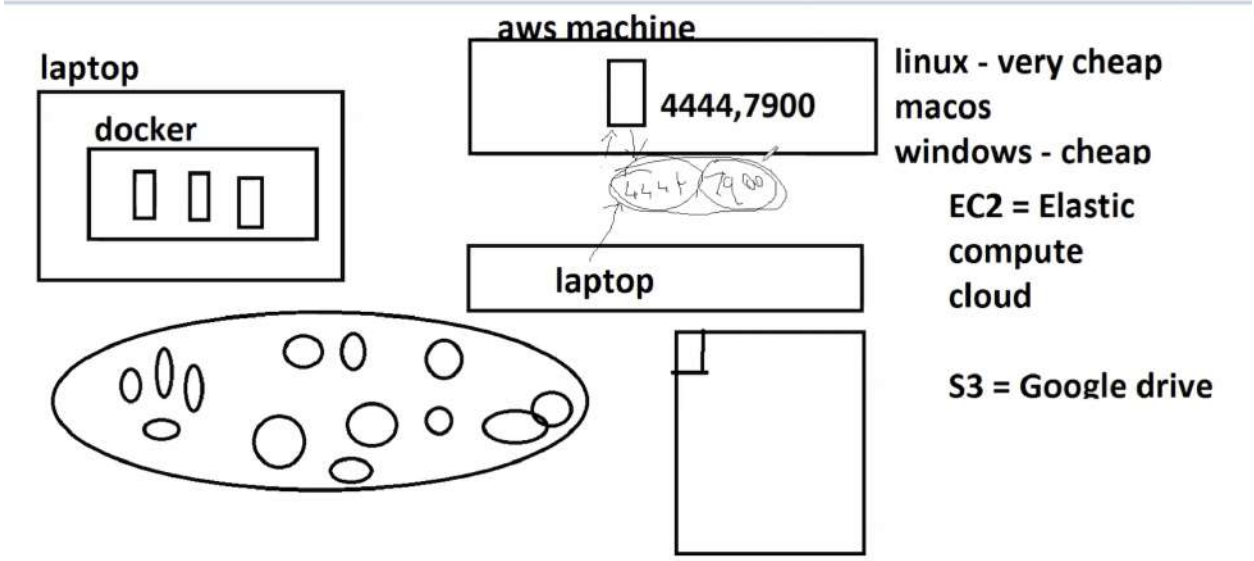
1.
```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setBrowserName("chrome");

// http://localhost:4444/ ->
//4444 - >Local machine port
WebDriver driver = new RemoteWebDriver(new URL( spec: "http://localhost:4444/"), capabilities);
```

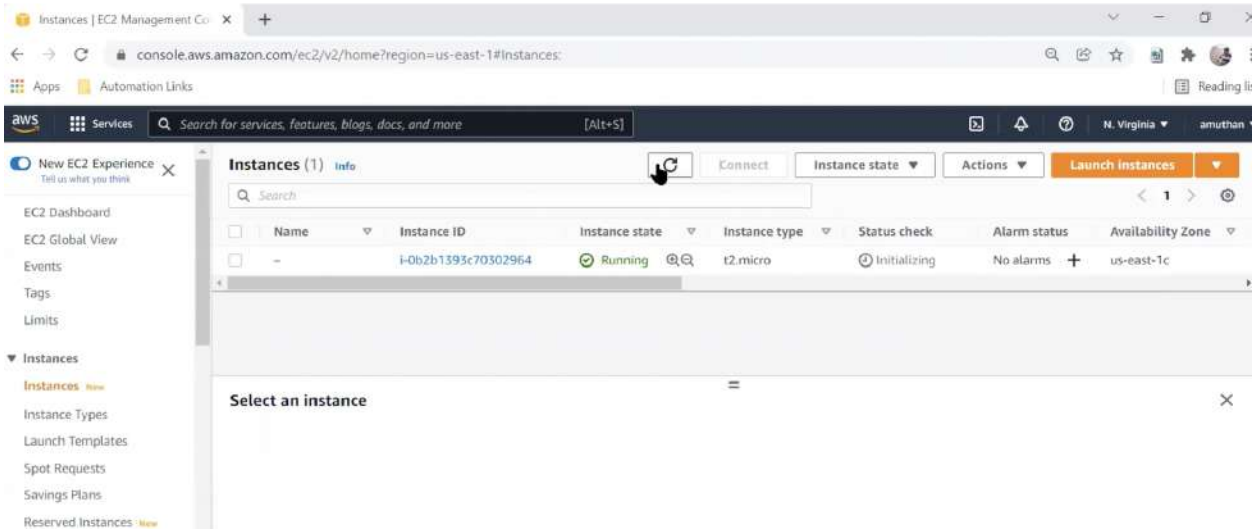
2. 
localhost:4444/ui/index.html#/
NAGARRO
DAILY TASK
Pending
LEARNING
Banks
Google Drive
Others
Devotion
UDEMY course pi...
External Work
Selenium Grid
4.1.1 (revision e8fcc2cecf)
Overview
Sessions
Help
URI: http://172.17.0.2:4444
Stereotypes
v.96.0
Sessions: 0
Max. Concurrency: 1
0%
Help - All rights reserved - Software Freedom Conservancy 2022.

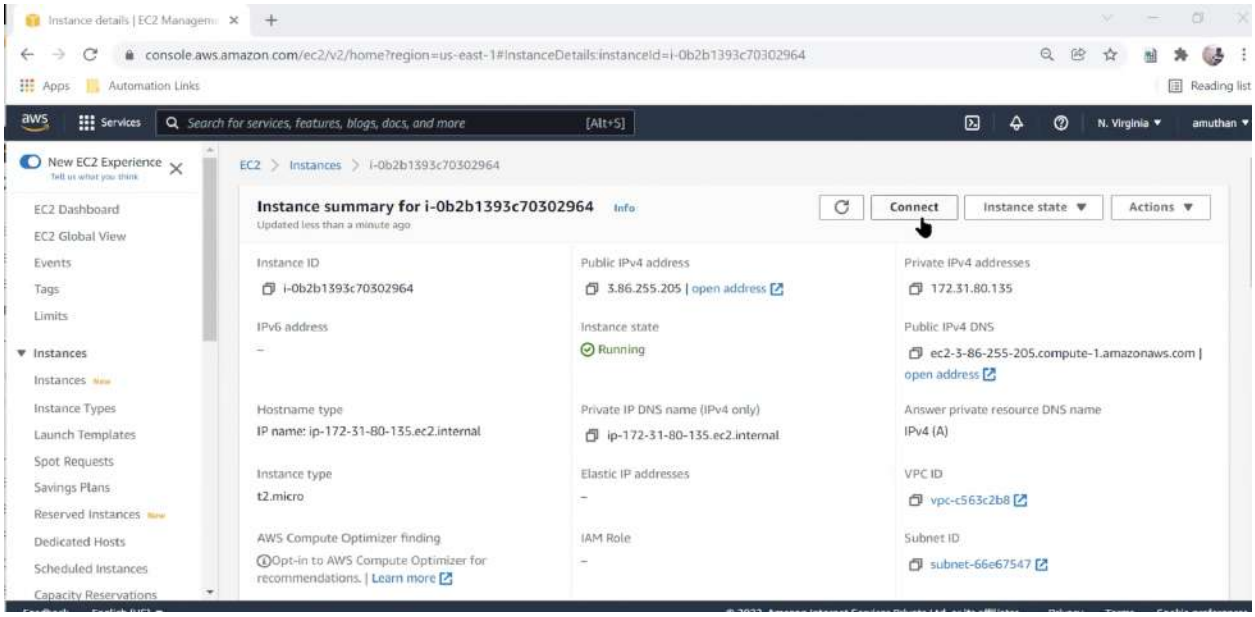
3. 
localhost:7900
DAILY TASK
Pending
LEARNING
Banks
Google Drive
Others
Devotion
UDEMY course pi...
External W

========28_Run Selenium Tests in AWS EC2==========

---

1. If our laptop is not that good
    a. Which we can use for Docker setup
2. Then, we can use
    a. Amazon EC2 (Elastic Compute Cloud)
    b. Go to https://aws.amazon.com/
    c. Do Signup
    d. Create a Free tier instance
3. 

4. 

5. 

==29_Driver Factory | Config Factory | Parallel execution with ThreadLocal ===

---

1. ThreadLocal
    a. Class to handle Multi-Threading
    b. This will be helpful for Parallel execution
    c. It helps to create objects that are read and written by the same thread

---

========30_Creating Page Layers | Page Components | Composition | Understanding Page Object Model==========

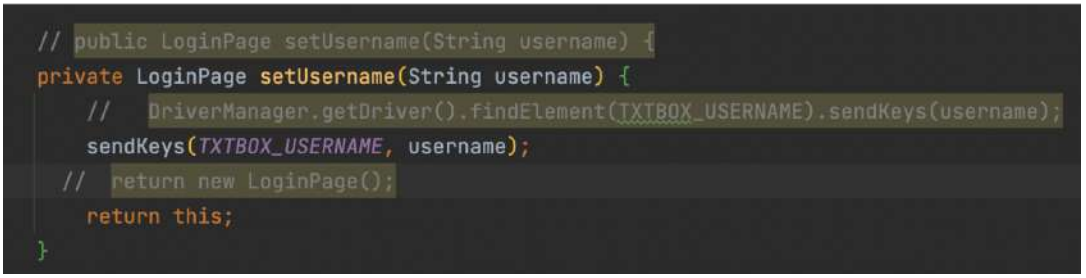
---

1. Page Object Model
    a. Method Chaining
        i.

```
//Wrapper method for performing 3 operations
public HomePage loginToApplication(String username, String password) {
   //Method Chaining
    return setUsername(username)
            .setPassword(password)
            .clickLogin();
}
```

b. Page Chaining

i.



```
// public LoginPage setUsername(String username) {
private LoginPage setUsername(String username) {
    //   DriverManager.getDriver().findElement(TXTBOX_USERNAME).sendKeys(username);
    sendKeys(TXTBOX_USERNAME, username);
  //  return new LoginPage();
    return this;
}
```

ii.

```
private HomePage clickLogin() {
    //DriverManager.getDriver().findElement(BTN_LOGIN).click();
    click(BTN_LOGIN);
    return new HomePage(); //Page Chaining
}
```

========31_Method Chaining | SeleniumUtils | Static Imports | Dynamic Locators and Java Faker ==========

---

1. Why no to Page Factories?
    a. Dynamic Locators are not possible
    b. In every Page class,
        i. PageFactory.initElements(driver, this);
            is mandatory

---

1. Java Faker API

a.

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

        Faker faker = new Faker();
        for (int i = 0; i < 2; i++) {
            System.out.println("----------------------");
            System.out.println("faker.address().cityName() = " + faker.address().cityName());
            System.out.println("faker.superhero().name() = " + faker.superhero().name());
            System.out.println("faker.address().fullAddress() = " + faker.address().fullAddress());
            System.out.println("faker.animal().name() = " + faker.animal().name());
            System.out.println("faker.number().randomNumber() = " + faker.number().randomNumber());
            faker.number().numberBetween(5,500);
        }

    }
}
```

========32_Enums | Extent Report Integration with Framework==========

=33_Listeners Integration with Framework | Annotation, and Usage in Framework =

========34_Excel and Test Data Supplier - Github and Git==========

---

1. Instead of DataProvider, we are going to use
    a. **Test Data Supplier**
        https://mvnrepository.com/artifact/io.github.sskorol/test-data-supplier
        Version - `1.9.7`
2. To read values from the Excel file:
    a. **Test Data Supplier**

---

1. **Test Data Supplier:**
    a. This library is built over Ownercell
    b. This can read the values from JSON, YAML, xlsx, CSV files.
        i. NOTE:

1. Make sure, you are using **Java 11 i**n Build path
2. Otherwise, it may result as:

```
/Users/rajatverma/Desktop/Work/Local_inteliJ_WS/TMB_SeleniumFramework2/src/main/java/com/learning/rough
 /DataSupplierTest.java:3:30
java: cannot access io.github.sskorol.core.DataSupplier
  bad class file: /Users/rajatverma/.m2/repository/io/github/sskorol/test-data-supplier/1.9
   .7/test-data-supplier-1.9.7.jar!/io/github/sskorol/core/DataSupplier.class
    class file has wrong version 55.0, should be 52.0
    Please remove or make sure it appears in the correct subdirectory of the classpath.
```

1.
```java
import com.beust.jcommander.converters.IntegerConverter;
import com.creditdatamw.zerocell.annotation.Column;

public class TestData {

    //  @Column(name="testcasename",index = 0) ->
    //Column name is testcasename and it is at 0th index in Excel sheet
    @Column(name="testcasename",index = 0)
    public String testCaseCame;

    @Column(name="username",index = 1)
    public String username;

    @Column(name="password",index = 2)
    public String password;

    //converterClass = IntegerConverter.class -> This will do the conversion from String to int
    @Column(name="age",index = 3, converterClass = IntegerConverter.class)
    public int age;

}
```

2.

testcasename	username	password	age
dummyTest	rajat	rajatt95	27
titleValidationTest	Admin	admin123	30
titleValidationTest	Admin	admin1234	35

3.

```
    @Test(dataProvider = "getData")
    public void test1(TestData testData) {
        System.out.println("testData.testCaseCame = " + testData.testCaseCame);
        System.out.println("testData.username = " + testData.username);
        System.out.println("testData.password = " + testData.password);
        System.out.println("testData.age = " + testData.age);

    }
    //@DataProvider --> Return type -> Object[][] or Object[]
    @DataSupplier //--> It can read any file (CSV, xlsx, JSON, YAMLDataSupplierTest)
    public StreamEx<TestData> getData() {

//        return Arrays.asList("Selenium", "Appium", "RestAssured");

        return  TestDataReader.use(XlsxReader.class) TestDataReader<XlsxReader>
                .withTarget(TestData.class) TestDataReader<...>.DataBuilder<...>
                //By default, it looks for files in src/test/resources directory
                .withSource("testdata/testData.xlsx")
                .read();
    }

}
```

---

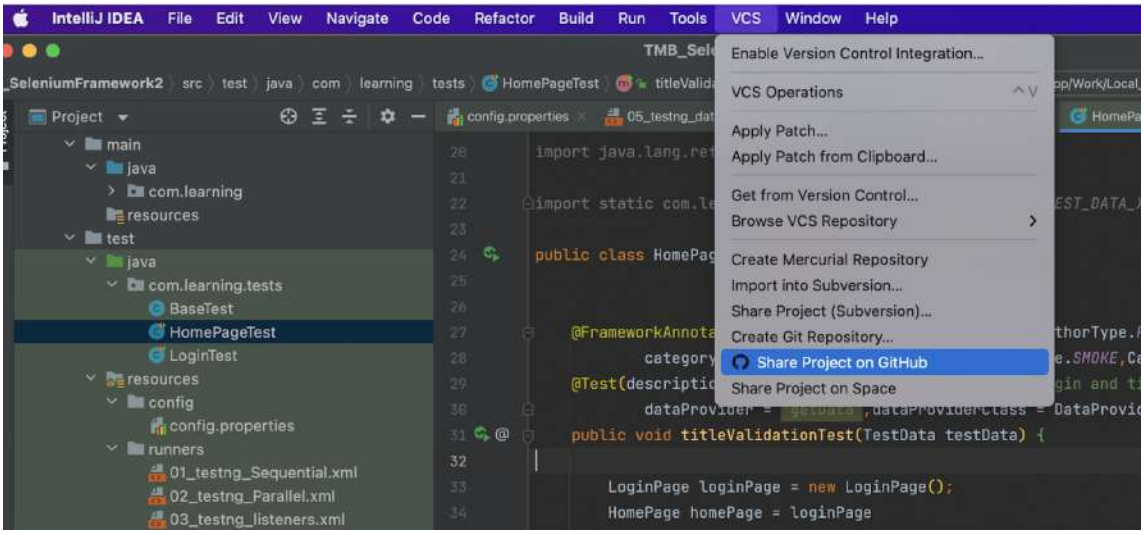
1. Now, if you want to filter for specific test cases, you need specific data, then,

a.

```
//@DataProvider --> Return type -> Object[][] or Object[]
//@DataSupplier //--> It can read any file (CSV, xlsx, JSON, YAMLDataSupplierTest)
@DataSupplier(runInParallel = true)
public StreamEx<TestData> getData(Method method) {

    return TestDataReader.use(XlsxReader.class) TestDataReader<XlsxReader>
            .withTarget(TestData.class) TestDataReader<...>.DataBuilder<...>
            //By default, it looks for files in src/test/resources directory
            .withSource(TEST_DATA_XLSX_FILE)
            .read() StreamEx<TestData>
            //  .filter(testData -> testData.getTestCaseName().equalsIgnoreCase("titleValidationTest"));

            // Using Java reflection -> Getting method name and use it for filtering
            .filter(testData -> testData.getTestCaseName().equalsIgnoreCase(method.getName()));
}
```

---

1. **Git:**
    a. Open IntelliJ
    b. VCS -> Share Project on Github

c. 

d. Code -> Commit -> Push to specific branch

e. Pull

f. Pull Request (to merge develop branch into master)

g. Merge-Conflicts handle

========35_Local and Remote Driver Factories | Git and Jenkins Integration==========

---

1. **Profile management in Maven:**

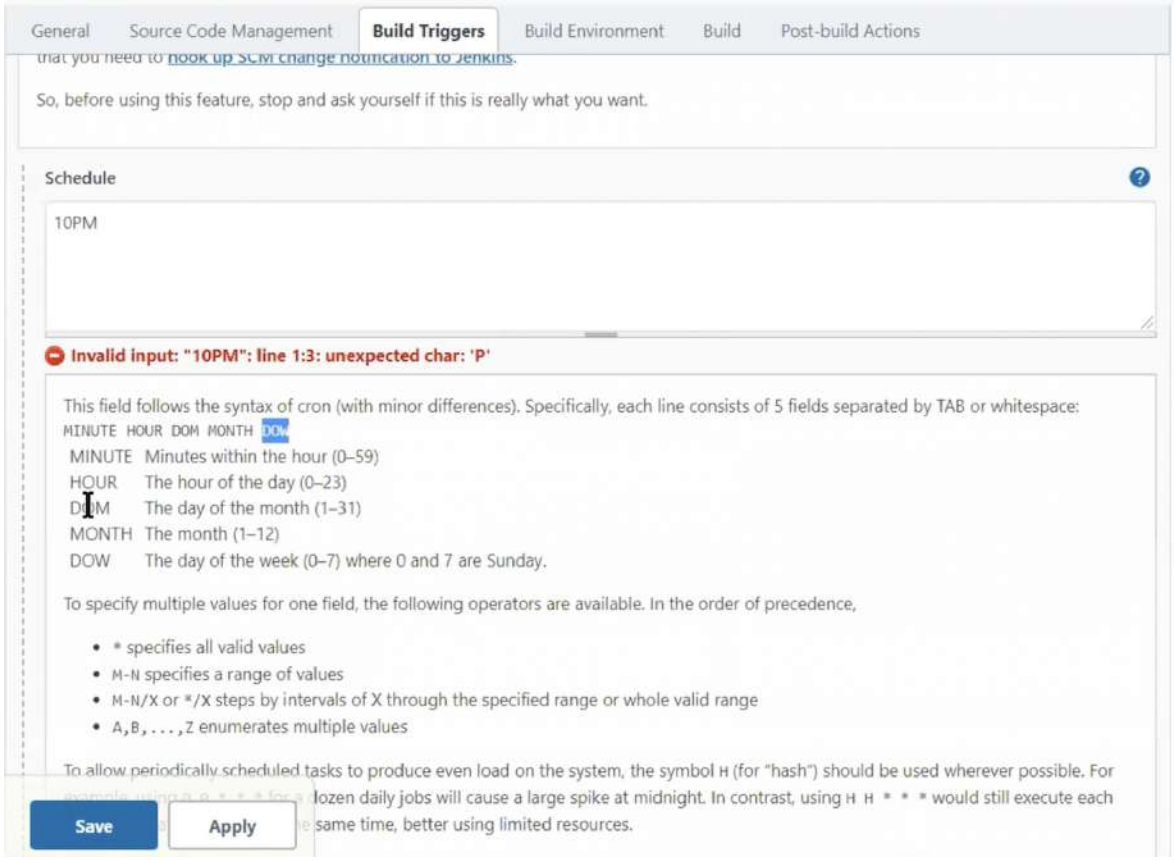
a.
```
<profiles>
    <profile>
        <id>smoke</id>
        <build>
            <pluginManagement>
                <plugins>
                    <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-surefire-plugin</artifactId>
                        <version>3.0.0-M5</version>
                        <configuration>
                            <suiteXmlFiles>
                                <suiteXmlFile>testng.xml</suiteXmlFile>
                            </suiteXmlFiles>
                        </configuration>
                    </plugin>
                </plugins>
            </pluginManagement>
        </build>
    </profile>
</profiles>
```

b. mvn clean test -Psmoke

i. mvn -> We are using the Maven command
ii. clean -> Whatever is present in the target folder will be removed
iii. test -> Maven Goal
iv. -Pall -> We are specifically executing **smoke** Profile

---

1. **Jenkins**
    a. CI/CD Tool
    b. Create job as Freestyle Project
    c. Execute from
        i. Local machine
        ii. Remote (Github/Bitbucket)
    d. Build Parameters
    e. Scheduling



==========