Video 2 - Selenium Test Life Cycle, Java for Selenium.

I) Selenium Test Process

1) Test Planning
2) Generate Basic Tests/Test Cases
3) Enhance Test Cases
4) Run & Debug Test Cases
5) Analyze Test Result and Report Defects

1) Test Planning

- Get Environment (UI Design and Database) details from Development team
- Analyze the AUT (Application Under Test) in terms of Object identification/Element identification

Selenium IDE: Use Record and playback some navigations and analyze
Selenium WebDriver: Using Element inspectors analyze the AUT,

Mozilla Firefox - Built-in feature "Page inspector"
Google Chrome - Built-in Developer tools (F12)
MS Edge - Built-in Developer tools (F12)

- Select Test Cases for Automation
  Test cases that can be executed on Initial build and every modified build (Smoke, Sanity)
  Test Cases that can be executed on every modified build (Re & Regression Tests)
  Test Cases that can be executed using multiple sets of Test data (Data Driven Tests)
- Select Selenium Tools and Others for Automated Testing and Configure Eclipse IDE, Java, Selenium WebDriver, TestNG, Maven, Jenkins etc…
- Download Eclipse IDE and Extract -Editor, Syntax Guidance, Context Help and Auto Compilation…
- Download Java (JDK) Software and Install in your computer
- Set Java Environment Variable path in the OS Environment
- Download selenium WebDriver java language binding from www.selniumhq.org and add WebDriver
  jar files to Java Project in Eclipse IDE
- Download & Install TestNG Testing framework in Eclipse IDE

Note: Download browser driver/s (Firefox, Chrome, MS Edge…) and Instantiate in Selenium Test Cases

2) Generate Basic Tests/Test Cases

In UFT:
a) Object Repository based Test Design (Recording, Keyword Driven Methodology)
b) Descriptive Programming (Static Programming and Dynamic Programming)

In Selenium:
a) Selenium IDE
Using Recording feature or Type Test steps using Element Locators and Selenese commands

b) Selenium WebDriver
Using Element Locators and WebDriver API Commands

3) Enhance Test Cases

a) Inserting verification points

In UFT:
Using Check points (UFT Tool feature)
Using VBScript conditional statements

In Selenium IDE: using "verify" or "assert" commands
In Selenium WebDriver:
Using WebDriver verification Commands (limited use)
Using Programming Control Flow Statements
Using TestNG Testing Framework "assert" methods

b) Parameterization

Replacing constant (fixed) values with parameters is called Parameterization
Passing single value or multiple values….

We use Parameterization feature for Data Driven Testing….

In UFT:
i) Using Data Table (UFT tool feature)
ii) Using VBScript Automation Object Models (FileSystemObject, Excel Object….)

In Selenium IDE: No Support
In Selenium WebDriver: Using programming (Flat files/excel files)….

c) Synchronization

In UFT: Using Synchronization Point, or wait command or increase Tool default time

In Selenium IDE: Using Synchronization Timeout feature (Tool feature)
In Selenium WebDriver: Using programming wait feature or WebDriver command

d) Error Handling

Handling expected and unexpected Errors

In UFT: Using Recovery Scenarios (tool feature or VBScript features
In Selenium WebDriver: using programming features (Ex: exception handling in Java)

e) Adding Comments

To make the code readable and to disable the code from execution

In UFT: We can use VBScript Comment syntax
In Selenium IDE: Using Selenium IDE comments syntax
In Selenium WebDriver: using Programming Syntax (if we use Java then use Java Comment syntax)

4) Run & Debug Test Cases

Run > Debug > Run…

Run Test Cases is a mandatory task in Testing

but Debugging Test cases is an optional task in Automated testing

Run Test Scripts / Test Cases

Run a Single Test case
Batch Testing

In UFT:
Use "Test Batch Runner" tool or using AOM Script or ALM/QC etc…

In Selenium IDE: using Test Suite feature

In Selenium WebDriver: Using Programming and using Testing Framework

Debug Test Cases

Note: Debugging feature is NA for Manual Testing and It is only for Automated Testing

What is Debugging?
Locating and isolating errors through step by step execution

Application Life Cycle
Development Testing Production

Error Fault Failure
Mistake Defect
Bug

When debugging is required?

Scenario 1: Test Case is not showing any error and providing desired result - Not required

Scenario 2: Test Case is showing errors - Optional

Scenario 3: Test Case is not showing any error and Not providing desired result - Required

Whenever Test case is not showing any error and not providing desired result there debugging is required.

5) Analyze Test Result and Report Defects

a) Analyze Test Result:

Selenium WebDriver doesn't have built-in Result Report facility, using programming Control Flow statements or Testing Framework verification methods we can generate Test Results

Status of Test Results in Software Testing
1) Pass (if expected == actual)
2) Fail (if expected != actual)
3) Warning (Whenever Test Case is not executing properly)
4) Done (if there is no verification point in the Test case)

b) Reporting Defects:

After analyzing the test Results, if we find any deviation from expected then report defects….

Functional Test Automation Defect Management

Selenium Excel

Selenium Bugzilla or Jira…

II) Java for Selenium (Overview)

Java has three important editions

i) Java Standard Edition / Core Java (Old name J2SE)
ii) Java Enterprise Edition / Advanced Java (Old name J2EE)
iii) Java Micro Edition (Old name J2ME)

Note: Java Standard Edition or Core Java is enough for Automated Testing with Selenium,

Java Standard Edition or Core Java

1) Comments

> To make the code readable
> To make the code disable from execution
Note: Java supports Single line and multiple line comments

Note 1: In Computer Programming 80% concepts are common (in every programming language) and Syntax may vary from one language to another

Example: Data Types, Variables, Operators, Control Flow (Conditional, Loop and Branching) etc…

Note 2: Some features may vary from one language to another
Example: Functions in C Languages, Methods in Java….

2) Data Types:

A Data type is a classification of the type of data that a variable or object can hold in Computer programming

Examples:
Character,
Integer,
Float, Double….
String
Boolean etc…
Java supports two categories of Data Types,
i) Primitive Data Types
ii) Non Primitive Data Types / Reference Data Types

3) Java Modifiers

Modifiers are used to set access levels for classes, variables and methods,

Java supports two categories of Modifies

a) Access Modifiers (default, public, private, and protected)
b) Non Access Modifiers (static, final, abstract, …)

4) Java Variables

A named memory location to store or hold temporary data within a program (RAM
ROM - HDD, CD, DVD, USB Drive etc…)

Java Supports three types of variables,

a) Local Variables
b) Instance Variables
c) Static Variables/Class Variables

Note: Java supports explicit declaration of variables

In Java:
```
int a;
a=100;//Correct
b=200; //Incorrect
int c=a+b; //Incorrect

int a;
int a, b, c;
int d=100;
int e=200, f=300, g=400;
```

In VBScript:
Dim a
a=100 'Correct (Explicit variable)
b=200 'Correct (Implicit variable)
Msgbox a+b '300
Note: VBScript supports Implicit and Explicit declaration of variables

5) Operators in Java

Operators are used to perform Mathematical, Comparison and Logical operations

Categories of Operators in Java,
a) Arithmetic Operators
b) Assignment Operators
c) Relational / Comparison Operators
d) Logical Operators

e) Bitwise Operators
f) Miscellaneous Operators
etc…

6) Java Control Flow Statements

i) Conditional / Decision Making statements (if, switch,)
ii) Loop Statements (for, while, do while and enhanced for)
iii) Branching statements (break, continue and return)

a) Java Conditional Statements / Decision Making Statements

i) Two Types of Conditional Statements

1) if statement
2) switch statement

ii) Three types of Conditions

1) Single Condition (Positive / Negative Condition)

```
if (a>b){
.
}

if (b<a){
}
```

2) Compound / Multi Condition

```
if ((a>b) && (a>c) && (a>d)){
.
}
```

3) Nested Condition

```
if (a>b){
if (a>c){
if (a>d){
…
}
}
}
```

iii) Usage of Conditional statements

1) Execute a block of statements when a condition is true

2) Execute a block of statements when a condition is true otherwise execute another block of statements

3) Execute a block of statements when a compound condition is true otherwise
execute another block of statements

4) Decide among several alternates (else if)

5) Execute a block of statements when more than one condition is true (nested if)

6) Decide among several alternates (using switch statement)

b) Loop Statements

Used for repetitive execution

i) for loop
ii) while loop
iii) do while loop
iv) enhanced for loop (Array)

c) Branching Statements

- Branching statements are used to transfer control from one point to another in the code
- Branching statements are defined by three keywords - break, continue and return

7) String handling in Java

- Sequence of characters written in ""
- String may contain Alpha bytes, Numeric, Alpha-numeric and special characters.

Example:

"SELENIUM"
"selenium"
"India123"
"123"
123 //Integer
"India123*&^"
Etc…

Note: Using String predefined methods we can perform operations on strings

8) Arrays

Generally, Array is a collection of similar type of elements

Dim a, b(3)
b(0)=100
b(1) ="India"
b(2) =10.234

- In Java Array is an Object that contains elements of similar data types
- Java Array is index bases and index starts from zero
- The length of an Array is established when they are is created and Array length is fixed,
- Each item in an Array is called as Element

9) IO Operations (includes File Handling)

- Read Data using Input devices
- Display Output on the Console

- Read data from files, Write data to files

10) Methods

What is Method?

A Set of statements to perform an Operation

Why Methods?
Methods are used for code reusability

When we choose methods?
Whenever we want to perform any operation multiple times then we choose methods

Note: All programming features can be used in methods also

Two Types of Methods in Java,

i) Built-in Methods
Number Methods
Character Methods
String Methods
Array Methods Etc…

ii) User Defined Methods
Method with returns a value (Static and Non Static methods)
Method with returns nothing (Static and Non Static methods)

11) Exception Handling in Java

- In Computer programming exception is an abnormal condition,
- An exception is an event that occurs during the execution of a program that disturbs the nor flow of instructions
- The exception handling in Java is one of the powerful mechanisms to handle run-time errors

int a=10; //No Error

int b=10 //Syntax Error

int c= 10/2; //No Error

int d=10/0; //Run-time Error (* Exception handling is required)

12) Java Object Oriented Programming (OOPS) Fundamentals

a) Inheritance
b) Polymorphism
c) Abstraction
d) Encapsulation

a) Inheritance

- Inheritance is a mechanism in which one object acquires all the properties and behaviors of parent object
- Using Inheritance we can create classes that are built-in upon existing classes
- When we inherit from an existing class then we can reuse methods and fields from the parent class

Java Supports,

i) Single Inheritance
ClassB extends ClassA

ii) Mluti Level Inheritance
ClassB extends ClassA
ClassC extends ClassB

iii) Multiple Inheritance (* Java doesn't support multiple Inheritance)
ClassB extends ClassA
ClassB extends ClassZ

b) Polymorphism

Performing task/s in different ways,

Polymorphism derived from two Greek words,

Poly - Many
Morphs - Forms / ways

So Polymorphism means "Many Ways"

Two types of Polymorphism in Java,
a) Compile Time Polymorphism / Method OverLoading
b) Run-time Polymorphism / Method Overriding