AUTOMATION CLASSES NOTES

SELENIUM WITH JAVA

TIME TABLE

Session 1	: Class 1 & 2	: 16th and 17th Aug	: 8am to 9am
Session 2	: Class 3 & 4	: 18th and 19th Aug	: 8am to 9am
Session 3	: Class 5 & 6	: 20th Aug	: 10:30 am to 12:30 am
Session 4	: Class 7 & 8	: 21th Aug	: 6pm to 8pm
Session 5	: Class 9 & 10	: 22th Aug	: 7am to 9am
Session 6	: Class 11 & 12	: 23th Aug	: 7am to 9am
Session 7	: Class 13 & 14	: 24th Aug	: 7am to 9am
Session 8	: Class 15 & 16	: 25th Aug	: 7am to 9am
Session 9	: Class 17 & 18	: 26th Aug	: 7am to 9am
Session 10	: Class 19 & 20	: 27th Aug	: 10:30 am to 12:30 am
Session 11	: Class 21 & 22	: 28th Aug	: 5pm to 7pm
Session 12	: Class 23 & 24	: 29th Aug	: 7am to 9am
Session 13	: Class 25 & 26	: 30th Aug	: 7am to 9am
Session 14	: Class 27 & 28	: 31st Aug	: 7am to 9am
Session 15	: Class 29 & 30	: 01st Sep	: 7am to 9am

Class:1

Topics:

1) Objective of Training
2) Training Topics to be Covered
3) Additional Benefit from Training
4) Introduction to Software Testing & Automation Testing
5) Introduction to Selenium WebDriver
    - Introduction to Automation Testing
    - Introduction to Selenium
    - Selenium Components
    - Advantages
    - Disadvantages
    - Selenium Vs UFT
    - Selenium Vs Silk
    - Introduction to Selenium WebDriver
6) Eclipse and WebDriver Environment Setup

Software Testing Overview

Acceptance testing: Verifying whether the system as a whole works as intended.

Unit testing: Validating that each software unit performs as expected. A unit is the smallest testable component of an application.

Functional testing: Checking functions by emulating business scenarios, based on functional requirements. Black-box testing is a common way to verify functions.

Integration testing: Ensuring that software components or functions operate together

Regression testing: Checking whether new features does noy break or degrade functionality. Sanity testing can be used to verify menus, functions and commands at the surface level, when there is no time for a full regression test.

Performance testing: Testing how the software performs under different workloads. Load testing,

Stress testing: Testing how much strain the system can take before it fails. Considered to be a type of non-functional testing.

Usability testing: Validating how well a customer can use a system or web application to complete a task.

Automation Testing Overview

Automation testing is a Software testing method or technique to test any application and compare the actual result with the expected result. This can be achieved by designing test scripts using any relevant automation testing tool. In manual testing test cases are executed manually without any proper support from relevant tools. However, for automated testing, test cases are executed with the assistance of one or more appropriate tools. Automation Testing is also more reliable as there is less chances of human error. In manual testing we have to run test cases manually every time based on our need, hence it is time consuming. However, in automation testing once scripts are ready we can run the test cases any number of time with different combination of data with minimal time. The main goal of Automation testing is to increase the test efficiency and develop software value.

Automation testing is useful for the below mentioned aspects:

- Functional test suite
- Regression test suite
- Smoke / Sanity test suite
- Build deployment
- Test data creation

Perform Automation testing for the below mentioned scenario of the Software:

- Requirements do not change frequently
- Steady Software with respect to manual testing
- Time crunch/Shortage of time for Testing activities
- Big projects
- Projects which require to test the same functionality often

Automation Testing Lifecycle

A.Test Automation Planning – Analyze Requirement, Analyze the AUT (Application Under Test) in terms of Object Identification, Identify test case for Automation

B.Test Script Design – Create Page Library, Create Reusable Methods, Design Test steps using Element locators

C.Test Script Enhancement – Insert Verification Points using Assert/Verify Commands, use Parameterization in Data driven Testing, Handle expected and unexpected errors

D.Run and debug Test Script - Locating and isolating errors thru Step by Step execution, debug for any wrong output/results

E.Analyze Test Results and Report Defects – Publish Test Results, Identify defect, Log defect

Different Automation Testing Tool

There are lots of free and licensee automation tool are available in the market now a days.

Some of the important tools are listed below:

Selenium: Very powerful free tool for testing of Web Applications. Provides multiple browser support.

QTP/UFT: Very useful tool for non-web applications and comes with a built-in object repository.

RFT: Support automated testing for functional, regression, GUI and data-driven testing.

TOSCA: Powerful automation tool for functional and regression testing of various software products

Silk: Automated functional and regression testing tool for desktop, mobile, and web applications.

Sikuli: Open source tool for GUI testing.

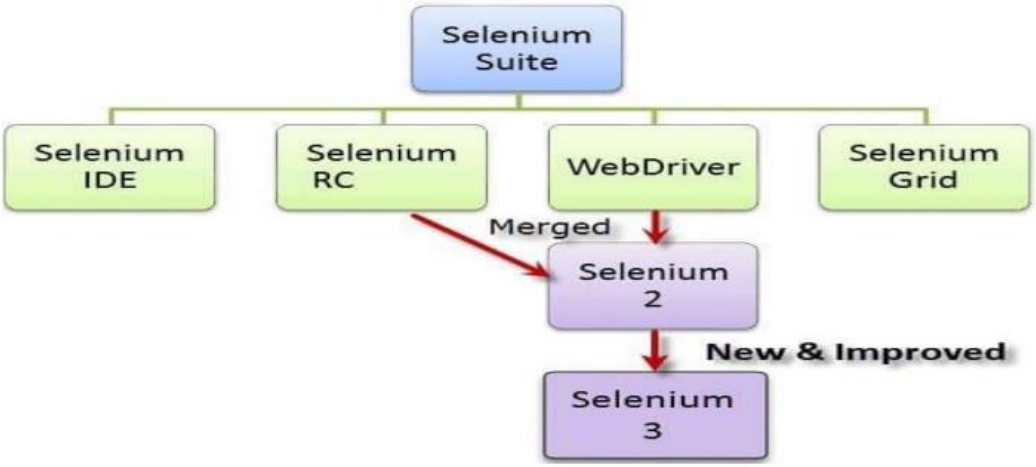
Soap UI: Automation tool for API testing.

Appium: Automation tool that supports mobile testing, native app testing and mobile web application testing.

TestNG: TestNG is not an automation tool in itself, however, it provides great support to automation frameworks built with selenium, appium, rest assured etc.

Junit: Mostly used for Unit testing by the developers

Component of Selenium



Selenium Integrated Development Environment, a Firefox add-on that you can only use in creating relatively simple test cases and test suites.

Selenium Remote Control, also known as Selenium 1, which is the first Selenium tool that allowed users to use programming languages in creating complex tests.

WebDriver, the newer breakthrough that allows your test scripts to communicate directly to the browser, thereby controlling it from the OS level.

Selenium Grid is a tool used together with Selenium RC to run parallel tests across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once

Advantages of Selenium

- Selenium is open source and free software and hence there is no licensing cost for its usage.
- Scripting can be done in most of the widely used programming languages like Java, C#, Ruby, Perl, PHP and Python
- Automation using selenium can be done in many OS platform like MS Windows, Macintosh and Linux
- It supports most of the popular browsers like Chrome, Firefox, Internet Explorer, Opera and Safari.
- Selenium Grid helps in parallel and distributed testing
- It uses less Hardware resources

Disadvantages of Selenium

- Web Services - REST or SOAP cannot be automated using selenium.
- Selenium WebDriver requires programming language requirement for script creation.
- No vendor support for tool compared to commercial tools like HP UFT
- As there is no object repository concept in Selenium, maintainability of objects becomes difficult
- For reading-writing to external files, reporting and other customization we have to rely on external libraries/Framework.

Advantages of Selenium WebDriver

Selenium WebDriver supports multiple browsers in multiple platforms

- Google Chrome 12.0.712.0+
- Internet Explorer 6+
- Firefox 3.0+
- Opera 11.5+
- Android – 2.3+ for phones and tablets
- iOS 3+ for phones
- iOS 3.2+ for tablets

• WebDriver is designed to provide a simpler and uniformed programming interface

- Same WebDriver script runs for different OS platforms like MS Windows, Mac, Linux etc.

• Support multiple programming language:

- Java, C#, Python, Ruby, PHP, Perl etc.

• It's efficient

- Faster than other tools of Selenium suite (IDE, RC)
- Control the browser by programming
- Supports Data driven Testing and Cross browser testing

- Supports Parallel test execution with the help of either JUnit or TestN

Selenium Vs UFT

Selenium	UFT
1) Open Source tool	Tool from HP, License is required
2) Supports various OS Environments like MS Windows, Macintosh and Linux	MS Windows only
3) Supports various Programming Environments like Java, Python, Perl etc.	VBScript only
4) No inbuilt Object Repositories	Local and Shared object Repositories
5) No built-in Reporting feature	Built-in reporting feature
6) Supports Web Applications only	Supports Desktop and Web Applications
7) Uses less Hardware resources	Uses more Hardware resources
8) Difficult to setup environment and use	Easy to setup and use.
9) Parameterization can be done via programming but is difficult to implement.	Parameterization Support is built
10) No Reliable support from any specific group	Permanent Support from HP

Selenium Vs Silk

Selenium	Silk
1) Open Source tool	Licensed tool
2) Supports various OS Environments like MS Windows, Macintosh and Linux	MS Windows only
3) Lot of reference, book and tutorial available for selenium	There is not much reference and tutorial available for siik
4) Selenium support a broad range of browsers like IE, Chrome, Firefox, Opera, Safari etc.	Silk Only Support IE and Firefox
5) Supports various Programming Environments like Java, Python, Perl, PHP, C# etc.	Silk uses 4Test scripting language similar to C++
6) Supports Web Applications only	Supports mainly Client server Applications

Pre-requisites for Selenium WebDriver

Refer selenium web driver Eclipse document (Installation details will be in this document).

Class 2

1. Create first Test Case using Selenium WebDriver (Chrome and Gecko Driver)
2. Web Elements and Element Locators
3. Operations on Web Page
4. Operations on Browser
5. Operations on Web Edit Box
6. Operations on Link

Setup eclipse in your System for Selenium WebDriver

Download and install Eclipse IDE. Launch eclipse.exe

- Set your workspace to any location (a workspace is a physical location where we store our project or group of related projects).
- Now create a new project- File->New->Project...->Java->Java Project
- Name you project and click Finish
- Now you will see an src folder under your project. Under this we need to create a package-Right Click src->New->Package (Basically these packages are used to group together related classes). Name your package e.g. 'com.myTestPackage'
- Inside this package create a new class and name it e.g. Test, your Test.java class will get created
- Right Click your project on the left and click on properties. A "Properties for {project name}" dialog box will appear. Click on "Java Build Path" on the left and then click on Libraries tab on the right. In this tab click on "Add External Jars.." button.
- You can verify the same in the "Referenced Libraries" section under your project in the "Package Explorer" section on the left

Some important naming convention

- Class name should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
- Interface name should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.
- Method name should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
- Variable name should start with lowercase letter e.g. firstName, orderNumber etc.
- Package name should be in lowercase letter e.g. java, lang, sql, util etc.
- Constants name should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.*

Run your First Test Case for Chrome Browser

```
package selenium;

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

public class Day1 {

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

	System.setProperty("webdriver.chrome.driver","C:\\Browserdrivers\\chromedriver
.exe");
		driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://itera-qa.azurewebsites.net/");
		driver.findElement(By.xpath("//a[text()=\"Test Automation\"]")).click();


	}

}
```

Run your First Test Case for Firefox Browser

```
package selenium;

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

public class Day1 {

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

	System.setProperty("\"webdriver.gecko.driver","C:\\Browserdrivers\\geckodriver
.exe");
		driver = new FirefoxDriver();
		driver.manage().window().maximize();
		driver.get("https://itera-qa.azurewebsites.net/");
		driver.findElement(By.xpath("//a[text()=\"Test Automation\"]")).click();


	}}
```

What is locator?

The locator can be termed as an address that identifies a web element uniquely within the webpage.

Locators are the HTML properties of a web element.

- Element Locators are common for all Browsers.
- There is a diverse range of web elements available in any webpage. The most common web elements are:

- Edit Box
- Text Box
- Check Box
- Drop Down Box
- Combo box
- Button
- Radio Button
- Link/Hyperlink
- Image/Image Link
- Web table
- Frame

Elements Inspection

1. Open any webpage using Internet Explorer or Chrome browser
2. Click on F12 button from keyboard to inspect element.
3. Click on the Inspector Icon.
4. Move your mouse around the page, the element under your mouse will be highlighted and an annotation displays its HTML tag

Different Elelements locator

i. ID – driver.findElement(By.id("IdName"))

ii. Name – driver.findElement(By.name("Name"))

iii. Class Name – driver.findElement(By.className("Element Class"))

iv. Tag Name – driver.findElement(By.tagName("HTML Tag Name"))

v. Link Text – driver.findElement(By.linkText("LinkText"))

vi. Partial Link Text – driver.findElement(By.partialLinkText("partialLinkText"))

vii. CSS Selector – driver.findElement(By.cssSelector("value"))

viii. XPath – driver.findElement(By.xpath("XPath")

driver – Object/Instance of WebDriver

findElement – WebDriver method

By – Pre-defined Class in Selenium

id – Element locater/attribute

IdName – Locator value

Webpage operations in Selenium WebDriver

1) To return title of the current Browser - getTitle()

Syntax: String variable = driver.getTitle();

Example:

```
driver.get("http://www.rediff.com/");

String pageTitle = driver.getTitle();

System.out.println(pageTitle);
```

2) To return HTML page source of the current focused Browser - getPageSource()

Syntax: String stringName = driver.getPageSource();

Example:

driver.get(“https://www.facebook.com/”);

String source = driver.getPageSource();

System.out.println(source);

3) **To return current URL of the focused Browser - getCurrentUrl()**

Syntax: String stringName = driver.getCurrentUrl();

Example:

driver.get(“https://www.google.co.in”);

String url = driver.getCurrentUrl();

System.out.println(url);

Most Useful API commands for a Browser in Selenium Web Driver

Navigate to the intended web page - get()

driverObject.get(“URL”);

Example: driver.get(“https://www.google.co.in”);

Close the focused Browser only - close()

driverObject.close();

Example: driver.close();

Close all Browsers which are opened by WebDriver during test execution - quit()

driverObject.quit();

Example: driver.quit();

Navigate from one URL to another URL - navigate().to()

driverObject.navigate().to("URL");

Navigate back to previous URL - navigate().back()

driverObject.navigate().back();

To move single item forward in the Browser history - navigate().forward()

driverObject.navigate().forward();

Refresh the content of the current web page - navigate().refresh()

driverObject.navigate().refresh();

To maximize the focused Browser - manage().window().maximize()

driverObject.manage().window().maximize();

Operations on WebEdit in Selenium WebDriver

To enter a value in the editable text box - sendkeys()

Syntax: WebElement testelement = driver.findElement(By.ElementLocator("value"));

testelement.sendkeys("Input Data");

To return the object Type - getAttribute()

Syntax: testelement.getAttribute("type");

To return the displayed status - isDisplayed()

Syntax: System.out.println(testelement.isDisplayed());

To return the enabled status - isEnabled()

Syntax: System.out.println(testelement.isEnabled());

To clear already entered value in the editable text box - clear()

Syntax: testelement.clear();

Operations for a link in Selenium WebDriver:

To click a Link - click()

Syntax: driver.findElement(By.ElementLocator("value").click;

To check whether the Element is in enabled status or not - isEnabled()

Syntax: driver.findElement(By.ElementLocator(“value”).isEnabled();

To check if the Element is displayed or not - isDisplayed()

Syntax: driver.findElement(By.ElementLocator(“value”).isDisplayed();

CLASS 3

- Operations on Images
- Operations on Check Box
- Operations on Radio Button

Operations on image

- Image links are the links in web pages represented by an image which when clicked sometimes navigates to a different window or page.
- User cannot use the By.linkText() and By.partialLinkText() methods because image links basically have no link texts at all. It is better to use CSS or XPATH.
- To fetch any attribute in Selenium, user need to use the getAttribute() method. The method takes the attribute name as a parameter.

Operations for a Check box in Selenium WebDriver

To select or unselect a Checkbox - click()

Syntax: driver.findElement(By.ElementLocator("value").click;

To verify whether the Checkbox is Displayed or not - isDisplayed()

Example: boolean a =driver.findElement(By.ElementLocator("value").isDisplayed(); System.out.println(a);

To verify whether the Checkbox is in Enabled status or not - isEnabled()

Example: boolean b = driver.findElement(By.ElementLocator("value").isEnabled();

System.out.println(b);

To verify whether the Checkbox is Selected or not - isSelected()

Example: boolean c = driver.findElement(By.ElementLocator("value").isSelected(); System.out.println(c);

Operations for a Radio Button in Selenium WebDriver

To select a Radio Button - click()

Example: WebElement radio1 = driver.findElement(By.ElementLocator("value");

radio1.click();

To verify whether the Radio Button is Displayed or not - isDisplayed()

Example: boolean a = radio1.isDisplayed(); System.out.println(a);

To verify whether the Radio Button is Selected or not - isSelected()

Example: boolean b = radio1.isSelected(); System.out.println(b);

To verify whether the Radio Button is Enabled or not - isEnabled()

Example: boolean c = radio1.isEnabled (); System.out.println(c);

CLASS:4

- Operations on Dropdown (Select) box
- Complete study of Xpath (XML Path)
- Complete study of CSS Selector

Handle Dropdown list using Selenium WebDriver

Selenium already provides Select class that has some predefined method for handling dropdowns which help a lot while working with Dropdown. To handle Drop Down and Multi Select List in Selenium we use the following types of Select Methods.

Types of Select Methods:

- **selectByVisibleText Method** - selects an option by its displayed text
- **selectByIndex Method -** selects an option by its index
- **selectByValue Method** - selects an option by the value of its "value" attribute

Types of DeSelect Methods

- **deselectByVisibleText Method -** deselects an option by its displayed text
- **deselectByIndex Method -** deselects an option by its index
- **deselectByValue Method** - deselects an option by the value of its "value" attribute
- **deselectAll Method** - deselects all previously selected options

Syntax: Select month = new Select(driver.findElement(By.id(“month")));

month.selectByValue(“4");

month.selectByVisibleText(“June");

What is Xpath and why we use it?

- XPath is defined as XML path. It is a syntax or language for finding any element on the web page using XML path expression. XPath is used to find the location of any element on a webpage using HTML DOM (Document Object Model) structure.
- XPath is a technique in Selenium that allows you to navigate the structure of a webpage's HTML. XPath is a syntax for finding elements on web pages. Using XPath in Selenium helps find elements that are not found by locators such as ID, class, or name. XPath in Selenium can be used on both HTML and XML documents.

Relative, Absolute and Partial Xpath

Relative Xpath is more like starting simply by referencing the element you want and navigate from the particular location. It begins from the current location and is prefixed with a "//".

Example: //span[@class='Email']

Absolute Xpath contains the complete path from the Root Element to the desire element. It begins with a root path and is prefixed with a "/".

Example: /HTML/body/div/div[@id='Email']

Partial Xpath

Example: /div/div[@id='Email']

Simple common expression associated with Xpath

Basic Xpath - XPath expression select nodes or list of nodes on the basis of attributes like ID , Name, Classname, etc. from the XML document.

Contains() - Contains() is a method used in XPath expression. It is used when the value of any attribute changes dynamically.

Using OR & AND - In OR expression, two conditions are used, whether 1st condition OR 2nd condition should be true. It is also applicable if any one condition is true or maybe both. Means any one condition should be true to find the element.

Text() - This mechanism is used to locate an element based on the text available on a webpage.

Starts-with() - It is used to identify an element, when we are familiar with the attributes value (starting with the specified text) of an element

Few Simple Basic Xpath Symbol available for Selenium WebDriver

Xpath Symbols ($x("") or xpath)

- //tagname[@attribute-name='value1']
- //*[@attribute-name='value1']

Xpath Generation using different other methods

- Xpath=//*[contains(@type,'sub')]
- Xpath=//*[@type='submit' or @name='btnReset']
- Xpath=//input[@type='submit' and @name='btnLogin']
- Xpath=//td[text()='textname']
- Xpath=//tagname[starts-with(@attribute, 'value')]

CSS Selector

CSS Selectors are string patterns used to identify an element based on a combination of HTML tag, id, class, and attributes. Locating by CSS Selector is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name.

- CSS selector is used by browsers to target specific HTML elements and apply styles. For web browser automation, Selenium webdriver uses CSS selectors to locate and perform an action on an object instead of applying a style.
- Using CSS selectors to locate elements has some benefits:

- It is faster
- More readable
- And used more often

Difference between Xpath and CSS Selector

Using XPath we can traverse both forward and backward whereas CSS selector only moves forward.

- CSS Selector is easy to write as compared to Xpath
- CSS Selector is Faster as compared to Xpath
- CSS Selector does not allow to create a selector based on text content whereas Xpath allows

Basic CSS Selector Symbol available in Selenium WebDriver

css=<HTML tag><#><Value of ID attribute>
css=<HTML tag><.><Value of Class attribute>
css=<HTML tag><[attribute=Value of attribute]>;
Example: input[type=submit]
Starts with (^): css=<HTML tag><[attribute^=prefix of the string]>
End with ($): css=<HTML tag><[attribute$=suffix of the string]>
Contains (*): css=<HTML tag><[attribute*=sub string]>
Example: css=input[id*='id']
Using Multiple Symbols:
css = tagname[attribute1='value1'][attribute2='value2']

Class 5

- Java Introduction
- Comments in Java
- Data Types
- Variables
- Objects and Classes
- Operators
- Java Strings

Features of Java

Java Programming Language was developed by Sun Microsystems in 1995, Now it is subsidiary of Oracle corporation.

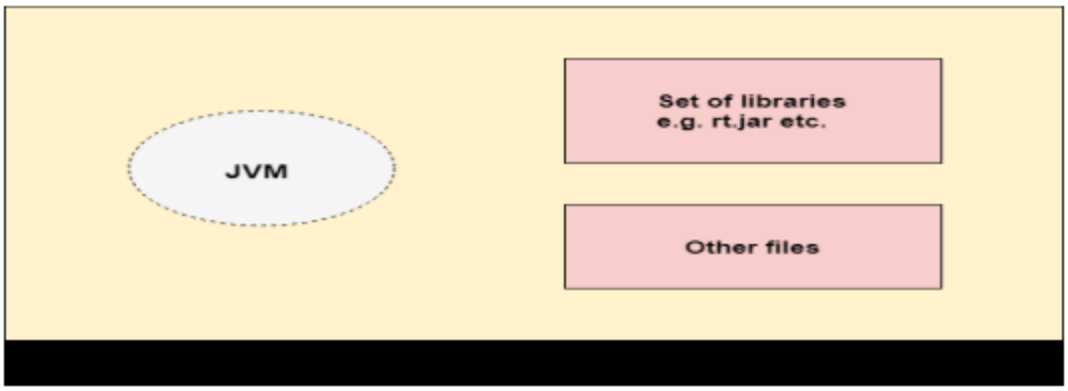
Most important features of Java language are listed below:

- Simple
- Object-Oriented
- Portable
- Secured
- Platform independent
- Architecture-neutral
- High Performance
- Multithreaded
- Robust
- Interpreted

Java Virtual Machine and Java Runtime Environment

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode. At compile time, java file is compiled by Java Compiler (It does not interact with OS) and converts the java code into bytecode.

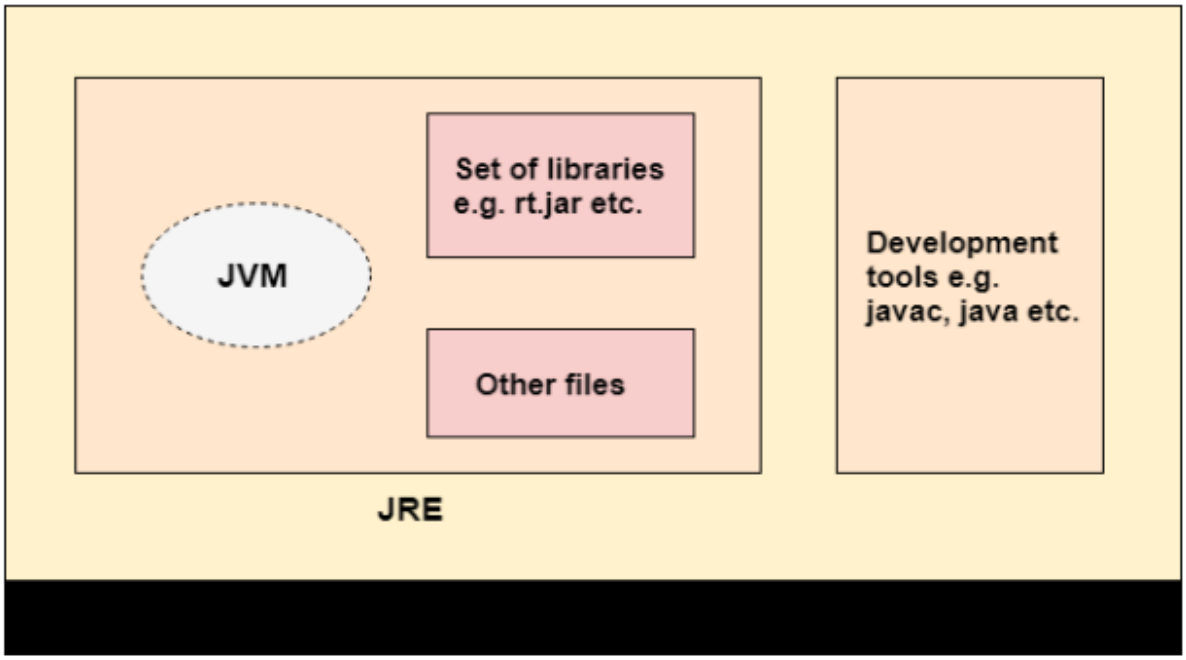
JRE (Java Runtime Environment) is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.



JRE

Java Development Kit

- The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.
- The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.



Importance of Java in Selenium WebDriver

- Selenium written in Java, hence Java is more compatible with Selenium. However, we can use other supported languages also like C#, Python, Ruby etc.
- Good support for Selenium with Java. Anyone can get more help for documentation and code implementations from internet.
- Most of the Selenium Testers are using Java only, so knowledge sharing or clarification resolution is very easy.
- Java is platform independent language, we can use Java on any Operating environment like Mac, Windows, Linux etc

Syntax Rules for Java

- Java is case sensitive language.
- First letter of Class Name should be in Upper case.
- Method names should start with lower case letter.
- Interface name should start with uppercase letter
- Package name should be in lowercase letter.
- Constants name should be in uppercase letter.
- Java program file name should exactly match with class name.
- Java Program execution starts from main method, which is mandatory in every Java program. θ Every Statement should end with semi colon symbol. θ Code blocks enclosed with {}

Create and Run a Java program using Eclipse

- Launch Eclipse IDE
- Create Java Project
- Create Java Package
- Create Java Class
- Write Java code in the Class file and Run

Main Function/Method:

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

- public – Access Modifier
- static – Non-Access Modifier
- void – Return Type (Returns nothing)
- Main - method name

Sample Program (Syso Method):

System.out.println("My First Java Code");

- System – Class (Pre-defined)
- out – Object
- println – method
- "My First Java Code" – Message

Java Comments

- Comments can be used to explain Java code, and to make it more readable.
- Comments are English words used for Code documentation.
- Java supports Single line comment and multiple lines comment.
- Comments in Java make the code Readable. It also make the code disable from execution
- Single-line comments start with two forward slashes (//). Any text between // and the end of the line is ignored by Java (will not be executed). Multi-line comments start with /* and ends with */.

Comments Syntax in Java:

Use // for Single line comments

Use /* …… ……. ……. */ for Multiple lines comments

Data Types Java

- A data type is a classification of the type of data that a variable or object can hold in computer programming.
- Java supports two categories of Data types:

a) Primitive Data Types (8 data types):

i) Integer Types:

1) byte (8 bits);

Example: byte a =10;

2) short (16 bits);

Example: short a =1000;

3) Integer (32 bits);

Example: int i = 10000;

4) long (64 bits);

Example: long l =100000000000L;

ii) Relational types (Numbers with decimal places)

5) float (32 bits);

Example: float f = 1.23f or (float) 1.23;

6) double (64 bits);

Example: double d =123.4567890;

iii) Characters

7) Character (16 bits);

Example: char c ='Z'

iv) Conditional

8) Boolean (1 bit);

Example: boolean b = true;

b) Non-primitive Data Types / Reference Data Types: Non-primitive or Reference data types in Java are Objects, Class, Interface, String and Arrays. Example: String str = "Java World"

What is Variables

a. A variable is a container which holds the value while the java program is executed. A variable is assigned with a datatype. Variable is a name of memory location.

b. Java variables are case sensitive.

c. Variable names should not match with Java keywords/Reserved words.

d. Must be unique in the scope of declaration.

e. Variable names Must not exceed 255 characters.

f. In Java, all the variables must be declared before they can be used.

g. The value stored in a variable can be changed during program execution

Types of Variables in Java

Three types of variables are available in Java

a) Local variable: Local variable is declared in methods or blocks.

b)Instance variable: Instance variables are declared inside the class but outside the body of the method.

It is called instance variable because its value is instance specific and is not shared among instances.

c)Class/Static variable: A Variable that is declared as static, It cannot be local. User can create a single

copy of static variable and share among all the instances of the class.

Example:

```
class Test{ int i1=10; //instance variable

static int i2=20; //static/class variable

void testmethod()

{

int i3=30; //local variable

} }
```

Objects and Classes

- An object in Java is the physical as well as logical entity whereas a class in Java is a logical entity only.
- A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- An object is an element of an application, representing an instance of a class.
- An entity that has state and behavior is known as an object e.g. chair, table, ball, cycle,, bike car etc. **Example:** Class: Animal; Object: Tiger, Elephant, Lion, Human, Cow, Dog etc.

Syntax to declare a class:

class { field; method; }

//Defining a Student class.

class Student{

//defining fields

int rollno= 1;

String name = “Raj";

//creating main method inside the Student class

public static void main(String args[]){

//Creating an object or instance

Student student1=new Student();

//creating an object of Student //Printing values of the object

System.out.println(student1.rollno);

//accessing member through reference variable

System.out.println(student1.name); } }

Java Operators

Operator in java is a symbol that is used to perform operations.

For example: +, -, *, / etc.

Important Categories of Operators:

a) Arithmetic Operators

1) Addition + (for Addition, String concatenation)
2) Subtraction – (for Subtraction, Negation)
3) Multiplication *
4) Division /
5) Modules %
6) Increment ++
7) Decrement —

b) Relational Operators

1) ==
2) !=
3) >
4) >=
5) <
6) <=

c) Assignment Operators

1) Assignment Operator: =
2) Add and Assign: +=
3) Subtract and assign: -=
4) Multiple and assign: *=

d) Logical Operators

1) Logical Not Operator !
2) Logical And Operator &&
3) Logical Or Operators ||

Java String

- String is a sequence of characters written in double quotes.
- String may have Alphabets, Numbers and Special characters.
- The java.lang.String class provides many useful methods to perform operations on sequence of char values.
- An array of characters works same as Java string.
- String can be created using new keyword.

String Example:

```
String s1 = "Welcome to Java"

char[] ch={'j','a','v','a'};
String s2=new String(ch);

String s3=new String("Java String");
```

String class function - Contd

toLowerCase(): Returns string with all uppercase characters changed to lowercase

```
String str4 = "ABCDEFGHIJ";
System.out.println(str4.toLowerCase());  //Output: abcdefghij
```

toUpperCase(): Returns string with all lowercase character changed to uppercase

```
String str5 = "selenium";
System.out.println(str5.toUpperCase());  //Output: SELENIUM
```

trim(): Returns a string from which any leading and trailing whitespaces are being removed

```
String str6 = "    hellojava     ";
System.out.println(str6.trim());  //Output: hellojava
```

contains(): Searches the sequence of characters in this string. It returns true if sequence of char values are found in this string or else returns false.

```
String name="do you know about selenium webdriver?";
System.out.println(name.contains("you know")); //Output: true
System.out.println(name.contains("hello")); //Output: false
```

String class function - Contd

isEmpty(): Checks if this string is empty or not. It returns true, if length of string is 0 or else false.

```
String s1="";
String s2="javaprogramming";
System.out.println(s1.isEmpty());   //Output: true
System.out.println(s2.isEmpty());   //Output: false
```

startsWith(): Checks if this string starts with given prefix. It returns true if this string starts with given prefix or else returns false.

```
String s3="welcome to java and selenium training session";
System.out.println(s3.startsWith("we"));   // Output: true
```

endsWith(): Checks if this string starts with given prefix. It returns true if this string starts with given prefix else returns false.

```
String s4="welcome to java and selenium training session";
System.out.println(s4.endsWith("session"));   // Output: true
System.out.println(s4.endsWith("java"));   // Output: false
```

substring(): Returns a part of the string. User pass begin index and end index number position in the java substring method where start index is inclusive and end index is exclusive.

```
String s5="javatraining";
System.out.println(s5.substring(2,4)); // Output: va
System.out.println(s5.substring(2));   // Output: vatraining
```

Class 6

- Array
- For loop
- While loop
- Do While loop
- Break Statement
- Continue Statement

Array in Java

- Array is a collection of similar type of elements that have a contiguous memory location.
- An array is a very common type of data structure wherein all elements must be of the same data type. Once defined, the size of an array is fixed and cannot increase to accommodate more elements, index starts from zero to n-1.
- We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.
- There are two types of array
    - Single Dimensional Array – Example: int [] array1 = {1, 2, 3, 4};
    - Multidimensional Array – Example: int [] [] array2 = {{1, 3, 5}, {2, 4, 6}};

Using an array in any program is a 3 step process:

1) Declaring the Array -

Example: int intArray[];

2) Constructing the Array -

Example: intArray = new int[2];

3) Initialize the Array -

Example:

intArray[0]=1;

intArray[1]=2;

Java Arrays class provides few utility methods. One of the utility method

Arrays.sort() helps us to sort an Array of objects.

Java Loops

- There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.
- Loop statements are used for repetitive execution.
- There are three types of loops in java. a) for loop b) while loop c) do while loop

Java For Loop

Java for loop is a control flow statement that iterates a part of the program multiple times.

For loop repeats a block of statements for a specified number of times.

If the number of iteration is fixed, it is recommended to use for loop.

Syntax:

```
for (startValue; endValue; increment/decrement){
Statement(s)
}
```

Example: Print 0 to 9 Numbers

```
for(int i=0; i<=9; i++){
System.out.println(i);
}
```

While Loops

A while loop statement in Java programming language repeatedly executes a target statement as long as a given condition is true.

Syntax:

```
Initialization;
while (Condition){
Statement(s);
increment/decrement;
}
```

Do while Loop

- The Java do-while loop is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.
- It executes a block of statements at least once irrespective of the condition.
- First, the statements inside loop execute and then the condition gets evaluated, if the condition returns true then the control gets transferred to the “do” else it jumps to the next statement after do-while.

Syntax:

```
Initialization;
do
{
Statement(s);
increment/decrement;
} while (Condition);
```

Break Statement in Java

- Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.
- Java supports break and continue control statements.
- When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

- The Java break is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

```
public class Main {
  public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
      if (i == 4) {
        break;
      }
      System.out.println(i);
    }
  }
}
```

Continue Statement in Java

- The continue keyword can be used in any of the loop control structures. It causes the loop to immediately jump to the next iteration of the loop.
- The purpose of writing a continue statement in java code is to skip the current iteration of a loop, say for, while and do-while.
- The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
public class Main {
  public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
      if (i == 4) {
        continue;
      }
      System.out.println(i);
    }
  }
}
```

CLASS: 7

- If Statement
- If-else Statement
- Switch Statement
- Methods
- Constructor

Conditional Statement in Java

Conditional Statements are used to control the flow of the program and insert verification points & error handling.

Five types of Conditional statements in Java

1) if statement

2) if else statement

3) else if statement/if else if ladder

4) nested if else Statement

5) switch Statement

if statement is used only to specify a block of Java code to be executed if a condition is met (true).

```
if(condition)
{
Statement(s);
}
```

An if statement can be followed by an optional else statement, else statement is used to specify a block of code to be executed if the condition is not met (false).

```
if(condition)
{
Statement(s);
} else
{
Statement(s);
}
```

if else if ladder in Java

else if statement is used to specify a new condition when first condition is false.

```
if(condition_1) {
   //execute this statement in case condition_1 is true
   Statement(s);
}
else if(condition_2) {
   //execute this statement in case condition_1 is not met but condition_2 is true
   Statement(s);
}
else if(condition_3) {
   //execute this statement in case condition_1 and condition_2 are not met but condition_3 is true
   Statement(s);
}
.....
else {
  //execute this statement in case none of the above conditions are true
   Statement(s);
}
```

Switch Conditional Statement in Java

- A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.
- The switch statement works with char, short, int, long data types.
- The case values must be unique. In case of duplicate value, it will throw compile-time error.
- Each case statement can have a break statement which is optional. When control reaches to the break statement, it jumps the control after the switch expression. If a break statement is not found, it executes the next case.

Switch Statement Example

```
char grade= 'B';
switch  (grade){
case 'A':
System.out.println("Excellent");
break;
case 'B':
System.out.println("Very Good");
break;
case 'C':
System.out.println("Good");
break;
case 'D':
System.out.println("Average");
break;
default:
System.out.println("Invalid Grade");
}
```

Java Method

- A Java method is a set of statements that are grouped together to perform an operation.
- Whenever we want to perform any operation multiple times then we choose methods.
- Methods are also known as Functions. Basically, two types of Methods are available in Java.

a) Built in Methods:

String Methods

Date & Time Methods etc.

b) User defined Methods:

Method with returning value

Method without returning any value

//Create a Method (User defined)

```
public int add(int a, int b)
{
int result = a + b ;
return result;
}
```

//Create Object and access Method

```
Sample obj = new Sample();
int x = obj.add(10, 25);
System.out.println(x);
```

Constructor in Java

- Constructor is a block of codes similar to the method. It is called when an instance of the object is created, and memory is allocated for the object.
- The purpose of constructor is to initialize the object of a class while the purpose of a method is to perform a task by executing java code. In other words, constructor provide memory to an object. Without initializing an object, we can't use its properties.
- A constructor eliminates placing the default values.
- Every time an object is created using new() keyword, at least one constructor is called. It calls a default constructor. Constructor can be overloaded but can not be overridden.
- Constructor(s) of a class must has name as the class name in which it resides.
- A constructor in Java can not be abstract, final and static.
- A Constructor must have no explicit return type.
- Java compiler distinguish between a method and a constructor by its name and return type. In Java, a constructor has same name as that of the class and doesn't return any value.
- If a class doesn't have a constructor, Java compiler automatically creates a default constructor during run-time. The default constructor initialize instance variables with default values. **For example:** int variable will be initialized to 0.

Constructor Example

Example:

```
class Test {
   Test() {
      // constructor body
   }
}  //Here, Test() is a constructor; it has same name as that of the class and
doesn't have a return type.
class Test1 {
   void Test1() {
      // method body
   }
}  //Here, Test1() has same name as that of the class. However, it has a
return type void. Hence, it's a method not a constructor.
```

CLASS:8

- Wrapper Class
- Date and Calendar function
- Try Catch in Java: Exception Handling

Java Wrapper Class

- A Wrapper class is a class whose object wraps or contains a primitive data types.
- Wrapper classes are used to convert any data type into an object. The primitive data types are not objects; they do not belong to any class; they are defined in the language itself. Sometimes, it is required to convert data types into objects in Java language.
- Wrapper class provide a mechanism to 'wrap' primitive values in an object so that primitives can do activities reserved for the objects like being added to ArrayList, Hashset, HashMap etc. collection.
- The eight classes of java.lang package are known as wrapper classes in java

Example:

```
int b = 10; // int data type
Integer intobj = new Integer(b); //wrapping around Integer object
double d = 250.5; // double data type
Double doubleobj = new Double(d); // Wrapping around Double object
System.out.println("Integer object intobj: " + intobj);
System.out.println("Double object doubleobj:  " + doubleobj);
String baseS = "12345";
System.out.println (baseS + 50);
int baseI = Integer.parseInt (baseS);
System.out.println (baseI);
System.out.println (baseI + 50);
```

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

Date & Calendar class function

- The class Date represents a specific instant in time, with millisecond precision. The Date class of java.util package implements Serializable, Cloneable and Comparable interface. It provides constructors and methods to deal with date and time with java.
- Java Calendar class is an abstract class that provides methods for converting date between a specific instant in time and a set of calendar fields such as MONTH, YEAR, HOUR, etc. It inherits Object class and implements the Comparable interface.

EXAMPLE:

```
import java.util.Calendar;
public class CalendarExample1 {
public static void main(String[] args) {
Calendar calendar = Calendar.getInstance();
System.out.println("The current date is : " + calendar.getTime());
System.out.println("At present Calendar's Year: " + calendar.get(Calendar.YEAR));
System.out.println("At present Calendar's Day: " + calendar.get(Calendar.DATE));
System.out.println("Current MINUTE: " + calendar.get(Calendar.MINUTE));
System.out.println("Current SECOND: " + calendar.get(Calendar.SECOND));
int maximum = calendar.getMaximum(Calendar.DAY_OF_WEEK);
System.out.println("Maximum number of days in week: " + maximum);
calendar.add(Calendar.DATE, -10);
System.out.println("10 days ago: " + calendar.getTime());
calendar.add(Calendar.MONTH, 4);
System.out.println("4 months later: " + calendar.getTime());
calendar.add(Calendar.YEAR, 2);
System.out.println("2 years later: " + calendar.getTime());
calendar.set(2018, 10, 15);
System.out.println(calendar.getTime());
}
}
```

Java Exception Handling:

An exception (or exceptional event) is a problem that arises during the execution of a program. When

- an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.
- An exception can occur for many different reasons. Following are some scenarios where an exception occurs. A user has entered an invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications. There are mainly three type of exception available in Java.

Checked Exception:

The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions

e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

Unchecked Exception:

The classes which inherit RuntimeException are known as unchecked exceptions

e.g. ArithmeticException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

Error:

Error is irrecoverable

e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

5 keywords is available in Java to handle exceptions

Keyword	Description
try	The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

Different Type of Exceptions in Java

Java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.

Built-in exceptions are the exceptions which are available in Java libraries. These exceptions are suitable to explain certain error situations. Below is the list of important built-in exceptions in Java.

Arithmetic Exception

It is thrown when an exceptional condition has occurred in an arithmetic operation.

ArrayIndexOutOfBoundException

It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.

ClassNotFoundException

This Exception is raised when we try to access a class whose definition is not found

FileNotFoundException

This Exception is raised when a file is not accessible or does not open.

IOException

It is thrown when an input-output operation failed or interrupted

InterruptedException

It is thrown when a thread is waiting , sleeping , or doing some processing , and it is interrupted.

NoSuchFieldException

It is thrown when a class does not contain the field (or variable) specified

NoSuchMethodException

It is thrown when accessing a method which is not found.

NullPointerException

This exception is raised when referring to the members of a null object.

NumberFormatException

This exception is raised when a method could not convert a string into a numeric format.

RuntimeException

This represents any exception which occurs during runtime.

StringIndexOutOfBoundsException

It is thrown by String class methods to indicate that an index is either negative than the size of the string

Try Catch Block

- The try block contains set of statements where an exception can occur. A try block is always followed by a catch block, which handles the exception that occurs in associated try block. A try block must be followed by catch blocks or finally block or both.
- A single try block can have any number of catch blocks.
- In case no exception occurs in try block then the catch blocks are completely ignored

Syntax:

```
try{
Statements
--
}
catch (Exception name){
Exception handling code
}
```

Example:

```
int a =10;
int b = 0;
try {
int result = a/b;
System.out.println(result);
}
catch (ArithmeticException e){
System.out.println("Divided by Zero Error");
}
```

CLASS:9

- Inheritance
- Method Overloading
- Method Overriding
- super keyword
- Abstraction/Abstract Class

Inheritance in Java

- Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.
- It is an important part of OOPS (Object Oriented Programming System).
- 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 Parent class and we can add new methods and fields.
- The class where the class members are getting Inherited is called as Super class/Parent class/Base class.
- The class to which the class members are getting Inherited is called as Sub class/Child class/Derived class.
- The Inheritance between Super class and Sub class is achieved using “extends” keyword.

Usage of inheritance in java ?

- For Method Overriding
  (so runtime polymorphism
  can be achieved).
- For Code Reusability.

Terms used in Inheritance

- **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

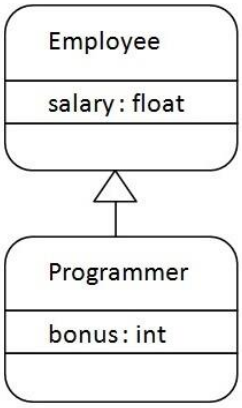
The syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
  //methods and fields
}
```

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called a parent or superclass, and the new class is called child or subclass.

Java Inheritance Example



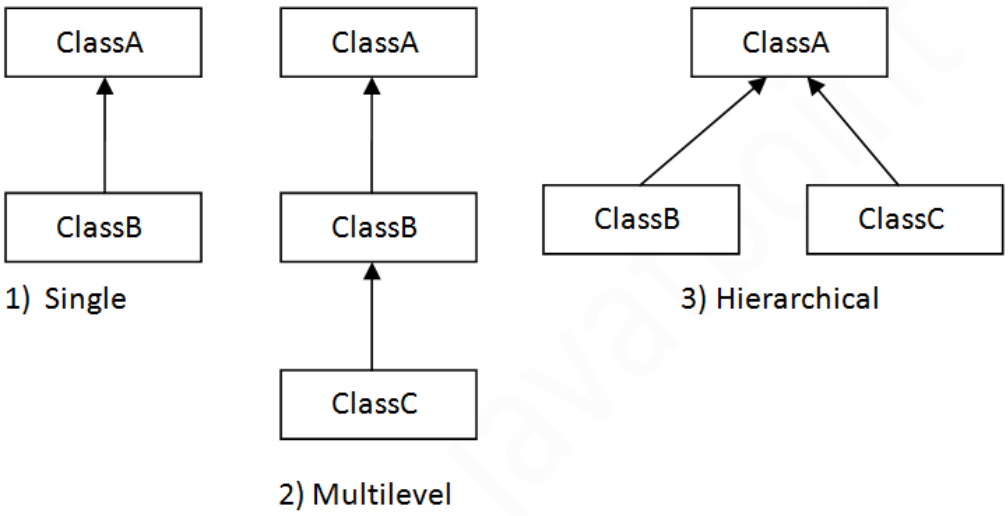
As displayed in the above figure, Programmer is the subclass and Employee is the superclass. The relationship between the two classes is **Programmer IS-A Employee**. It means that Programmer is a type of Employee.

```
class Employee{
 float salary=40000;
}
class Programmer extends Employee{
 int bonus=10000;
 public static void main(String args[]){
   Programmer p=new Programmer();
   System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus of Programmer is:"+p.bonus);
}
}
```

Types of inheritance in java

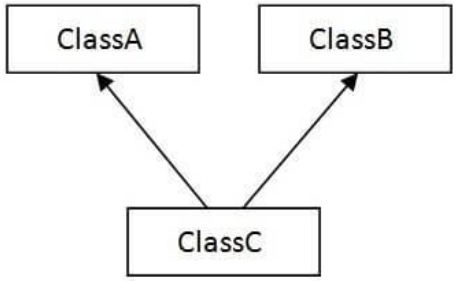
On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.

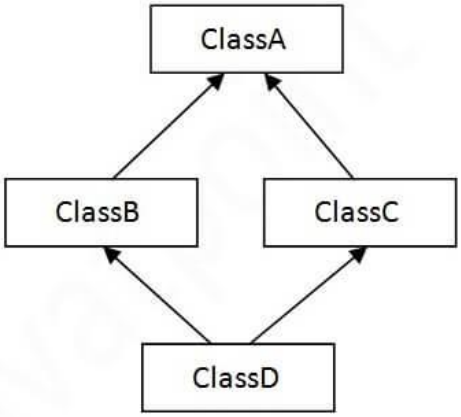


When one class inherits multiple classes, it is known as multiple inheritance. For

Example:



4) Multiple



5) Hybrid

.

Practise example on types of inheritance ?

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark()
{
System.out.println("barking...");
}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

1) Single Inheritance
Example:

```
public Class ClassB extends Class A {
}
```

2) Multi level Inheritance
Example:

```
public Class ClassB extends ClassA {
public Class ClassC extends ClassB {
}
}
```

Compile Time Polymorphism in Java

Polymorphism in Java is a concept by which we can perform a single action in different ways.

Polymorphism derived from two Greek words, Poly-means Many and Morphs - means ways; So polymorphism means many ways.

There are two types of Polymorphism is available in Java.

a) Compile Time Polymorphism (Method Overloading)

b) Run-time Polymorphism (Method Overriding)

Method Overloading: Two are more methods having same name in the same class but they differ in following ways.

a) Number of Arguments

b) Type of Arguments

Example for Method Overloading:

```
public class MethodOverLoading {
public void add(int a, int b){
System.out.println(a+b);
}
public void add(int a, int b, int c){
System.out.println(a+b+c);
}
public void add(double a, double b){
System.out.println(a+b);
}
public static void main(String[] args) {
MethodOverLoading obj = new MethodOverLoading();
```

```
obj.add(100, 200);
obj.add(15, 25, 35);
obj.add(101.234, 23.456);
}
```

Run-time Polymorphism in Java:

- If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.
- When a method in a subclass has the same name, same parameters or signature and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.
- Constructors cannot be overridden.
- A method declared final cannot be overridden.
- Static method cannot be overridden. It is because the static method is bound with class whereas instance method is bound with an object.
- Lets take a simple example to understand this. We have two classes: A child class Girl and a parent class Human. The Girl class extends Human class. Both the classes have a common method void eat(). Girl class is giving its own implementation to the eat() method or in other words it is overriding the eat() method.

Example:

```
class Human{
//Overridden method
public void eat()
{
System.out.println("Human is eating");
}}
class Girl extends Human{
//Overriding method
public void eat(){
System.out.println("Girl is eating");
}
public static void main( String args[]) {
Girl obj = new Girl();
//This will call the child class version of eat()
obj.eat();
}
} //Output: Girl is eating
```

Super keyword

- The super keyword in Java is a reference variable which is used to refer immediate parent class object.
- Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.
- Super keyword can be used at variable, method and constructor level.
- Private methods of the super-class cannot be called. Only public and protected methods can be called by the super keyword.

Example:

```
class Animal{
String Name=“Elephant";
}
class Dog extends Animal{
String Name=“Bullet";
void printName(){
System.out.println(Name);//prints Name of Dog class
System.out.println(super.Name);//prints Name of Animal class
}}
class TestSuper{
public static void main(String args[]){
Dog d=new Dog();
d.printName();
}}
```

Abstraction in Java

- Abstraction is a process of hiding implementation details and showing only functionality to the user.
- In another way it shows important things to the user and hides internal details, for example, sending SMS where you type the text and send the message. One don't know the internal processing about the message delivery.
- Abstraction focuses on what the Object does instead of how it does.
- A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).
- From Class (Concrete class or Abstract Class) to Class we use “extends” keyword

Example:

```
abstract class Animal{
abstract void run();
}
class Tiger extends Animal{
void run()
{System.out.println(“Tiger is running");
}
public static void main(String args[]){
Tiger obj = new Animal();
obj.run();
} }
```

CLASS:10

- Interface
- static keyword
- final keyword
- finally keyword

Interface in Java

- Interface is a Java type definition block which is 100% abstract
- All the Interface methods by default public and abstract.
- static and final modifiers are not allowed for interface methods.
- In Interfaces variables are public, static, and final by default.
- Interface can not be instantiated, and it does not contain any constructors.
- Interface can be used using “implements” keyword for a Class.
- A class that implements an interface must implement all the methods declared in the interface.
- From Interface to Interface, we use “extends” keyword

Example:

```
public interface Bike{
public void engine();
public void wheels();
public void seat();
}
```

Static keyword

- The static keyword in Java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than an instance of the class.

- When a variable is declared as static, then a single copy of variable is created and shared among all objects at class level. Static variables are, essentially, global variables. All instances of the class share the same static variable.

- It is possible to create a class members (variables and methods) that can be accessed and invoked without creating an instance of the class. In order to create such a member, we have to use static keyword while declaring a class member.

Example:

```
public class TestStaticVariable {
static int m = 10; // static variable
int n = 5; // non static variable
public static void main(String[] args) {
TestStaticVariable tsv = new TestStaticVariable();
System.out.println(tsv.n);
System.out.println(m);
}
}
```

Final keyword

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 -> To create constant variables

Final Methods -> Prevent Method Overriding

Final Classes -> Prevent Inheritance

- The only difference between a normal variable and a final variable is that we can re-assign value to a normal variable, but we cannot change the value of a final variable once assigned. Hence final variables must be used only for the values that we want to remain constant throughout the execution of program.
- A final method cannot be overridden by any subclasses. The final modifier prevents a method from being modified in a subclass. The main intention of making a method final would be that the content of the method should not be changed by any outsider.
- The main purpose of using a class being declared as final is to prevent the class from being subclassed.
- If a class is marked as final, then no class can inherit any feature from the final class

Example:

```
class Test
{
public static void main(String args[])
{
final int i; // local final variable
i = 20;
//i=50; It will throw error
System.out.println(i);
}
}
```

Java finally Block

- A finally block contains all the crucial statements that must be executed whether exception occurs or not.
- A finally block must be associated with a try block, you cannot use finally without a try block. You should place those statements in this block that must be executed always.
- In normal case when there is no exception in try block then the finally block is executed after try block. However, when an exception occurs then the catch block is executed before finally block.
- Try block can be used with finally before catch also.

Syntax

```
try {
        //Statements which might cause any kind of exception
        } catch
        { //Exception Handling
        }
        finally
        { //Statements to be executed
        }
```

Example

```
        InputStream in = null;
        try { in = new FileInputStream("D:\\Selenium\\file.txt");
        } finally
        { if (in != null)
        { try { in.close();
        }catch (IOException e)
        { System.out.println(e);
        } } }
```