**JAVA**

1. **List :** \*Indexed type. \* Preserves insertion order.  **\***Allows duplicate values \* add(),get()
2. **Set:**
3. **Map:**
4. **Differences:**

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| Implements serializable, clonable and random access interface. Duplicates are allowed. Null values are allowed. Initial size is 10. Retrieving elements is faster and adding or deleting of elements takes time. | Implements serailizable, clonable. Uses doubly linked list data structure. Adds one object at a time. Insertion order is preserved. Duplicates are allowed. Objects are non synchronized. Adding and deleting elements is faster. Retrieving is slower. Implements Queue interface even. |

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| Methods are non synchronized ,cannot be used in multithreaded env. | Methods in vector are synchronized. Hence thread safe therefore useful in multithreaded env. |

1. **Exception Handling:**
2. *Control flow for try catch block:*

\*If no exception , catch block won’t be executed.

\*Within try block if any exception come and its handled thru catch then rest of the code in try wont execute. Therefore keep min risky code in the try.

\* if try has exception but catch cannot catch it then abnormal termination happened.

\*if exception is in catch then it can’t be handled, so abnormal termination.

\*\*Only exception raised in try block can be handled.

\*If exception in finally then it cant be handled so abnormal termination.

\*\*If there an exception in try block cant be handled by any catch block and if there is some statement in finally then finally block executes and terminates abnormally without executing any code outside try -catch even.

1. *Difference between checked and unchecked exception:*

Eg: PrintWriter pw= new PrintWriter(“new.txt”);

Pw.println(“Hello”);

At compile time , compile time error will occur: unreported exception java.io.filenotfound;must be caught or declared to be thrown

Eg: int x= 10/0;

At Compile time: no error , At runtime: exception arithmetic.

1. *Difference between Error and Exception:*

|  |  |
| --- | --- |
| Error | Exception |
| Cannot be recovered. Caused by lack of system resources | Mostly caused by code can be recovered by using try-catch or throws |
| Stack overflow error | Arithmetic or null pointer |

1. **Differences between overloading and overriding:**

|  |  |
| --- | --- |
| OVERLOADING | OVERRIDING |
| Same name but different argument types. | Changing the method implementation in the child class. |
| Argument types must be different ,at least the order | Argument types must be same including order |
| Method signature must be different | Method signature must be same. |
| Can have different return type | Co-variant return type allowed- can override with child class return type |
| Applicable for private , static , final | Not applicable for private static and final |
| No restrictions for overloading for access modifiers | Increasing the visibility of access modifiers is allowed |
| Known as compile time polymorphism | Runtime polymorphism |

1. **Examples of OOPS concepts in Selenium world:**
2. ABSTRACTION: *Page Object Model* *-* Here the locators are present in one class and utilized in test classes. Implementation relating the locators is hidden.
3. INTERFACE: WebDriver is an interface, whose reference is used to create an object of Firefox, chrome or any other browser classes. It has many abstract methods such as close(), quit(),back(),forward()
4. INTERITANCE: WebDriver waits, property files, excel classes are created as base classes and are utilized in the test classes. Different banks with different ROIs.
5. METHOD OVERLOADING: assertEquals method is overloaded.
6. METHOD OVERIDING: equals() and toString() are overridden from object class in string class to give the equality of the contents in two strings and content of the string

get() and navigate() methods of different drivers in selenium are an example.

1. ENCAPSULATION: Every method in selenium is encapsulated. In POM classes we use @FindBy to declare the data members and initialize them using the constructor.

Cola vending machine consists of cola and the operation which Is hidden

**Encapsulation is a way to implement Abstraction.**

1. **Difference between char Array and a String:** String is terminated with a special character ‘\0’.
2. **How to print the address of the String:** getclass() returns the class name of the object. getName() returns the name of the entity(eg: class, interface, primitive type)

Address of the String instance: *String s = new String(“ABC”);*

*s.getClass().getName+’@’+Integer.toHexString(s.hashCode());*

1. **Methods in String class:**  **a.** equals() || equalsIgnoreCase()

**b.** compareTo() : converts string to Unicode value and compares the String. S1> s2 positive values is returned.

S1= S2 gives 0

S2> S1 neg value

**c.** split(regex)|| spilt(regex,limit)

**d.** contains() e. length()

**e.** replace(char old , char new) || replace(charseq target, charseq repl)

replaceAll(String regex, String repl) || replaceFirst(String regex,String replace).

**f.**Concat() || +

**g.** substring(strt ind, end indix): gives the String starting from start index till end index-1.

1. **Methods in StringBuffer:** append(), insert(),delete()
2. All the wrapper classes have by default overridden hashcode() and equals() therefore duplicate values cannot be added in hashmap, even written it will override that value and make it as one
3. **Foreach loop**: is an advanced for loop usually used to traverse array and collections.

For(datatype variable: array/collection)

{

}

1. Map: Set keyset(), Set valueSet()
2. Differences between Iterator and Listiterator:

|  |  |
| --- | --- |
| Iterator | ListIterator |
| Can be used across all collection interfaces and the classes. | Can only be used in List interface and its implementing classes. |
| Can be used to traverse only in forward direction. | Can traverse both in forward and reverse direction. |
| Has basic next and hasNext(). Cannot get index | Has additional methods like add() and replace(), nextIndex(), previousIndex() |

1. **HashSet does not allow get():** HashSet is all about having unique values if element is not present in Set then we can add it. Here we can also check the presence of object not get that object. We can use contains() method instead to check if its present or if not present add it.
2. **Difference between remove() of collection and remove() of iterator:** remove(obj) of collection removes the current element in the collection classes as specified in the method. Remove() of iterator removes that element which is pointed by the next() of the iterator.
3. **Collection interfaces:**

|  |  |  |  |
| --- | --- | --- | --- |
| LIST | QUEUE | SET | MAP |
| \*Indexed type  \*duplicates are allowed  \*Insertion order preserved  \*add(),get()  \*collections.reverse(list) | \*not indexed type  \*insertion order not maintained.  \*Cannot insert null values.  \*Duplicates are allowed.  \*add(),peek(),poll() | \*Duplicate not allowed  \*null values allowed  \*does not maintain insertion order. | \*key value pairs |

1. **In detail:**

*ArrayList:* null values and duplicate values. Accessing faster, insertion and deletion slower.

*LinkedList:* Duplicates allowe , insertion order preserved and non -synchronized. Adding and removing data is easier and fetching is difficult.

*PriorityQueue:* Sorted in required way, Using comapreTo().

*HashMap:* Null allowed, not synchronized, no insertion order preserved.

*HashTable:* Insertion order preserved.

*LinkedHasMap:* Allows duplicate, insertion order preserved.

*TreeMap:* Does not allow null values, values sorted using comparator interface.

*HashSet*: duplicates not allowed, null values allowed, does not preserve insertion.

*LinkedHashSet:* Duplicates allowed, preserves insertion.

*TreeSet*: Elements are sorted, Also because it implemets navigable set has many extra methods like method to find value higher or lower than current element eg: higher(Object), lower(Object obj)

1. **System.out.println();** System is a class in java.lang package, out is a static variable in the System class of the type printStream and println() is a method present in the PrintStream class.

**Eg:** class Test

{

Static String s= “abc”;

}

Test.s.lenght();

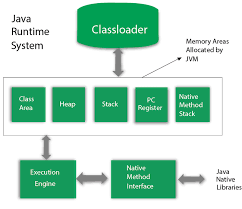
1. **Difference between Abstraction and Encapsulation:**

|  |  |
| --- | --- |
| ABSTRACTION | ENCAPSULATION |
| Hiding implementation and providing only utilization of the object | It is the way of binding the data and code acting on that data (Methods) together. |
| Abstract class-0 to 100% abstraction, interface – 100% abstraction | Private data, public getter setter eg: HashMap protecting inner working of object from outside world. |
| Hiding unwanted details by giving out most important details to the user. | Protecting inner working of objects from outside world. |
| Focuses on what object does instead of how it does | Hiding internal details of how an object works |
| Based on design level | Based on implementation level |
| Focuses on what abject does. | Hiding the mechanism of how the object does something. |
| Mobile phone’s display screen , buttons etc. | No need to understand how keypad is connected to display |
| Focuses on outer lookout | Focuses on inner lookout |

1. **Difference between Interfaces and Abstract class.**

|  |  |
| --- | --- |
| INTERFACE | ABSTRACT CLASS |
| When we don’t know what to implement at all | When we partially know what to implement |
| Methods are public and abstract | Methods can be concrete or abstract |
| 100% pure abstraction | Not pure abstraction. |
| Modifiers can’t be the following: public: so no protected, private. Abstract: private , static , final | Can be declared any how |
| Public static final data members | If a class has at least one abstract method then class should be declared as abstract |
| Always initialize at time of declaration | No such rule |
| Constructors , static and non-static block not allowed | All allowed |
| Cannot create and object, can have only a reference variable. | Can create and object |
|  | Public void test()  {  }  This is not an abstract method this is a concrete method with blank implementation |

1. **Architecture of JVM:**



*Class Loader*: loads the class files .Its the subcomponent of JVM.

*Memory*: has 5 parts:

1. *Method area*: Usually stores the method and the class data.
2. *Heap:* Object created data or instance data.
3. *Stack area*: all the currently running thread information is stored. It holds local variables and partial results in each stack called frame and frame is erased when execution is complete.
4. *Program counter Register*: Stores the address of the JVM instruction surrently being executed.
5. *Native Method Stack*: Contains all the native methods being used in the application.

Eg: getSystemTime() is native method for java.

*Execution Engine*: Executes the java code: contains 3 parts \*virtual processor \* Interpreter \*JIT compiler(just in time). JIT- Enhances execution speed by compiling byte code that have similar functionality. Interpreter: Reads byte code stream then execute the instructions. Eg: if a method is being repeatedly called then JIT compiles it separately and executes the same compiled version to reduce time.

1. **Difference between final, finally, finalize():**

|  |  |  |
| --- | --- | --- |
| *final* | *Finally* | *Finalize()* |
| Final is access modifier can be used with a variable method or a class.  Final variable one time declaration and initialization, methods can be overridden, final class can’t be extended. | Is a block of code used along with try-catch to write the clean up code. | Is a method used by garbage collector to clean up the unaddressed objects. |

1. **File Handling:**

FileInputStream: Read the data onto the file.

FileOutputStream: Write the data into the file.

**SELENIUM**

1. @DataProvider

Publiv object[][] getLoginData()

{

Object data[][]= TestUtil.getTestData(“sheet\_name”);

//TestUtil Is the utility class created by user and getTestData has the logic of apache POI

Return data;

}

@Test(dataProvider=”getLoginData”)

Public void test()

{

}

Upload a file in selenium:

Download AutoIt and AutoIt editor.

ControlFocus(“title”,”text,”controlid””)

controlSetText(“title”,”text,”controlid”,”new text”)

controlClick()

<http://learn-automation.com/upload-file-in-selenium-webdriver-using-autoit/>

RunTime.getRuntime.exec(path of the compile.exe autoit program);//Runtime is a class in java.lang package

Download the file in selenium:

Use AutiIt and use the command Inetget : this command will download the file using http,https protocols.

What is autoit: it’s a freeware programming language used to create automations to the windows based application.

* Write the script in autoit which is already present , just modify the the path where downloaded file needs to be stored with the appropriate extention .zip, .png
* You can also check the size of the downloaded file.

Find the xpath of that link where file downloads

Then RunTime.getRunTime().exec(path of compiled file)

Iterating through the HashMap:

Using lambda function:

1st method:map.foreach((k,v)->syso(key value is +k+ “values is” +v));

2nd method:

Iterator<Map.Entry<integer,String>> it = map.entrySet().iterator();

While(it.hasNext())

{Map.Entry<Ineger,String> ent = it.next();

Syso(ent.getKept())

Syso(ent.getValues);

Barcode and QR code in Selenium:

Add dependencies: Zxing- core and Zxing - javase

XPTAH continued:

<td align=”left”………>

<input type= check1></td>

<td alihn=”left”>

<a href=”https……”> test2 teste2</a>

<td align=………………………>

Xpath= //a[text(),”test2 test2”]//parent::td[@class=””]//preceding-sibling::td[@...]//input[@id=’’]

URL:

Extract the URl of the barcode

* Create an object of URL class and pass the url taken.
* ImageIO.read(url object)
* Decode the binary bitmap
* Get the text of the barcode.

Handling web table using java Script executor:

**publicstaticvoid** clik(WebDriver d,WebElement ele, String dateval)

{

JavascriptExecutor js =((JavascriptExecutor)d);

js.executeScript("arguements[0].setAttribute('value','"+dateval+"');", ele);

}

Here javascript executor checks the value in the DOM and changes it with the enetered data value.

Here u cant handle correct date value…any given date will be selected

**Calendar:**

**For tr and td calendar:**

1. Break the xpath into into before and after and store in two strings
2. Final declaration of the total no of days in a week=7
3. Data will start from sec row as first is just heading row
4. String day=null;

For(int rownum=2; rownum<=7; rownum++)

{

For(colnum=1; colnum<=noofweek; col++)

{

Try

{

day=

d.findElement(By.xpath(beforexpath+rownum+afterxpath+”]”)).getText();

}catch(NoSuchElementException e)

{

Syso(please enter correct date)

Flag=false;//if exception occurred then break from outer loof also therefore flag should be flase.

break

}

if(day.equals(the user input of day))

{

d.findElement(By.xpath(beforexpath+rownum+afterxpath+col+”]”)).click();

flag=true;//declare on flag value above…if element found set it to true

break;

}

}

If(flag)//if flag is true then only break the outer for loop.

Break;

}

Types of exceptions in Selenium:

1.Element click intercepted exception: when click request did not happen properly.

2. Element not interactable exception: element is not interactable ,hidden , not maximized etc.

3. Element not selectable exception:

4. Element not visisble element; available on DOM but not visible on page

5.Error in resposnse exception: thrown when error occurs on server side.

6. InsecureCeritficateException:

7. Invalid argument exception: instead of int ur passing string.

8. Invalid elementstate exception:

9.Invalid sessionID;

10.Inavild switchto target

11.Javascript exception.

12. Noalertpresent.

Challenges in IE browser:

1. Protacter setting;session not found.
2. Zooming level
3. Xpath not matching.
4. SendKeys performance.

Challenges faced in executing seleniu webdriver:

1. Ajax call: explain country ans state dropdown example.
2. Handling duplicate elements: finding xpath using child, sibling etc.
3. Handling dynamically changing element on webpage.
4. Handling alerts and popups multiple at one window.
5. Multiple windows more than two, session id needs to be known
6. Dynamic web table.
7. Iframes in webpage.
8. Ie browser script execution.
9. Window based popups : autoit
10. Creating object repo by POM etc.
11. Reporting mechanism.
12. sendKeys and click javascript executor.
13. Ajax calls : element cannot be clicked as it should be clicked somehwre and then that element will be enabled.
14. NullPointerException.

Important points:

* To refresh the page: d.navigate().refresh(), d.getUrl(), d,getCurrentURL(),d.navigate.to(d.navigate().currentURL()), KEYS.F5
* getWindowHandle:returns unique window id(like class name unique string), getWindowHandles: retruns in Set of all windows popping up
* d.close():close the existing current webdriver instance, d.quit(): closes all the opned webdriver instances.
* D.findElements(): if multiple links, or dropdown elements etc
* For screenshot: TakesScreeshot interface and getScreeshotAs()
* Mousehover: Actions class d.moveToElement(ele).build().perform()
* Webbased popup: use window handler API
* Window based pop: use send keys and write the path of the file to upload it or use third party tools
* Hidden element: use javaScriptExecutorjs.executeScript(document.getElementByclasName(ele).click());
* How to read excel data: apache POI API
* List some scenarios which are not poss by sel: captcha, barcode, window based popups, bitmap comparison(like comparing pdfs or images),third party calendars.
* OR: no particular format. We can create onw by using properties file.POM also is a object repository
* How to use recovery scenario in sel: use try catch block
* Upload a file in sel: sendKeys() – here there should be <type=”file”>or autoIT
* Download a file: using autoIT
* Running sel webdriver test by command line: go to code written path🡺 javac filename🡺java filename
* Testing in command line: java org.testng.Testng <file path>
* D.switchto.frame(id,name,index,webelement)
* Connect to db: using jdbc , pass query will get result set object
* Resize window: ude Dimension class, d.windo.max(d)
* For right click we have contextClick() in actions class
* Drag and drop: create actions class object: clickAndHold(ele).moveToElement(ele1).release.build.perform()
* How to hightlight a element: using javascriptExecutor(style.backgroud)

Broken Elements in Selenium:**package** com.sel.assign;

**import** java.net.HttpURLConnection;

**import** java.net.URL;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.xml.ws.Response;

**import** org.apache.http.HttpClientConnection;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**publicclass** DemoBrokenLinks {

**publicstaticvoid** main(String[] args) {

System.*setProperty*("webdriver.chrome.driver","D:/chromedriver.exe");

WebDriver d= **new** ChromeDriver();

d.get("https://freecrm.com");

d.manage().window().maximize();

WebElement ele= d.findElement(By.*xpath*("//a[@href='https://ui.freecrm.com']"));

ele.click();

WebElement user = d.findElement(By.*xpath*("//\*[@id='ui']/div/div/form/div/div[1]/div/input"));

user.sendKeys("poojaarballi@gmail.com");

WebElement pass= d.findElement(By.*xpath*("//\*[@id='ui']/div/div/form/div/div[2]/div/input"));

pass.sendKeys("Danamma@24");

WebElement loginbtn=d.findElement(By.*xpath*("//\*[@id='ui']/div/div/form/div/div[3]"));

loginbtn.click();

List<WebElement>l=d.findElements(By.*tagName*("a"));

l.addAll(d.findElements(By.*tagName*("img")));

//we will not consider a tag where href is not available

Listactivelinks=**new** ArrayList<WebElement>();

**for** (**int**i = 0; i<l.size(); i++) {

**if**(l.get(i).getAttribute("href")!=**null**&& (!l.get(i).getAttribute("href").contains("javascript"))){

activelinks.add(l.get(i));

}

}

System.***out***.println("total no of links:"+l.size());

System.***out***.println("no of active link:"+activelinks.size());

//check the hrefurl with httpconnectionapi

**for** (**int**j = 0; j<activelinks.size(); j++) {

HttpURLConnection con=

(HttpURLConnection)**new** URL(activelinks.get(j).getAttribute("href")).openConnection();

con.connect();

String response=con.getResponseMessage();//OK or NOt found

con.disconnect();

System.***out***.println(activelinks.get(j).getAttribute("href")+"......>"+response);

}

}

}

Important points for testing:

* Advantages: parallel execution, dependencies, priorities, grouping of testcases, parameters, @dataprovider used to execute same testcases with different test data, assertions,html report, easily integrate with Jenkins, listners.
* We have pre-requisities annotation and post condition annotation.
* XML schema: XML version and doctype
* Testing.xml is called the runner class. Chart for your suite to exectute
* How to parameterize: <parameter name=url value=”http:free crm.com”/> under classes tag
* Assertnull,assert not null,asserttrue etc
* <suite name="Test-method Suite" parallel="methods" thread-count="2" >//here 2 defines no of webdriver instances that are to be opened.
* @Test

Public void test1(expectedException=”elementNotFoundException”)

* Grouping in testing.xml: <define><include>
* How to run testcases in parallel: use parallel attribute = methods,tests etc.
* How to exclude test case: in the class tag🡺<exclude name=”name of test”>
* Enabled=false for disabling test cases.
* Dependencies: dependsonMethod and dependsOnGroup
* Testng report: in the test-output folder: index.html, testing.xmls
* Listners:IReport and ITestListener, purpose is to generate reports and logs mainly
* @test(timeout=1000), timeout for a test.

1. **Architecture of WebDriver:**

WebDriver is an open source API collection used to automate web based applications. WebDriver allows you to work with multiple browsers on multiple platforms.

HTTP over Server

Selenium client library:

Java,C#,Ruby,Python.

Real Browser

Browser driver

Chrome,ie,firefox

Chrome,ie,firefox

Json Wire protocol over HTTP

1. **TestNG**: Testing next generation is an automation framework used to execute webdriver scripts written by developers. 3 types of annotations : precondition annotation, post condition, test annotation.

Eg: @BeforeSuite, @BeforeTest, @BeforeMetod, @BeforeClass, @Test, @after…..

Advantages: open source, easy to maintain the test scripts, lots of annotations, priorities can be assigned, grouping is poosible, generates reports,

* Assign priorities: @Test(priority=1)
* Grouping: @Test(groups=”name”)
* Dependency: @Test(dependsOnMethods=”loginTest”)
* Call a method several times: @Test(invocation=8)
* If any method in the script takes long time to execute then terminate using time out: @Test(timeout=2000)
* Annotation for exception: @Test(expectedException=numberformatException.class)
* Disable a test @Test(enabled=’false’)
* Parameters: @Test({“username,” “password”}) in .xml: <parameter name= “username” value=”para”>

Eg: Assert.assertFalse(ele.isSelected())

Hard assert: program stops after assert stamen fails.

Soft assert: program continues to execute even after assertion fails.

SoftAssert sassert= new SoftAssert();

Why testing.xml? to execute many classes and tests as a single test suite.

1. **Page Object Model:** Page object is a class that represents a webpage which has elements as objects and functionalities to interact with the webpage.

**Page factory:** is a way to initialize the web elements present inside the webpage

XPATH:

1. //tagname[@att name=att-val]
2. Functions in xpath: text(), contains().starts-with()
3. Contains: //tagname[conatains(@att-val,’part of the value’)]
4. Starts-with: //tagname[starts-with(@att-valu,’part of value’)]
5. Text(): //tagname[text(),’full text inner html’]
6. Contains and text: //tagname[conatins(text(),’full inner html’)]

Page Object Model:

EventFiringWebDriver class will give the detailed logs og each line execution of the test scripts.

*e\_driver* =**new** EventFiringWebDriver(*d*);

*eventListner*= **new** WebEventListener();

*d*= *e\_driver*.register(*eventListner*);

Extent Report: by implementing IReporter interface.

In testng.xml there should be a listener tag to specify the location of extent report class