Difference between following and following-sibling

The tags that are present right below under the same parent is called “following-sibling” whereas for “following” we will be retrieving all the values highlighted with the tag name we specified.

**What are the exceptions in selenium**

1. **ElementNotVisibleException**: Although an element is present in the DOM, it is not visible (cannot be interacted with). E.g. Hidden Elements – defined in HTML using type=”hidden”.
2. **ElementNotSelectableException**: Although an element is present in the DOM, it may be disabled (cannot be clicked/selected).
3. **InvalidSelectorException**: Selector used to find an element does not return a WebElement. Say XPath expression is used which is either syntactically invalid or does not select WebElement.
4. **NoSuchElementException**: WebDriver is unable to identify the elements during run time, i.e. FindBy method can’t find the element.
5. **NoSuchFrameException**: WebDriver is switching to an invalid frame, which is not available.
6. **NoAlertPresentException**: WebDriver is switching to an invalid alert, which is not available.
7. **NoSuchWindowException**: WebDriver is switching to an invalid window, which is not available.
8. **StaleElementReferenceException**: The referenced element is no longer present on the DOM page (reference to an element is now Stale). E.g. The Element belongs to a different frame than the current one OR the user has navigated away to another page.
9. **SessionNotFoundException**: The WebDriver is performing the action immediately after ‘quitting’ the browser.
10. **TimeoutException**: The command did not complete in enough time. E.g. the element didn’t display in the specified time. Encountered when working with waits.
11. **WebDriverException**: The WebDriver is performing the action immediately after ‘closing’ the browser.

------------------------------------------------------------------------------------------

try{

// Some code

}

catch(Exception e){

// Code for Handling the exception

}

Below methods can be used to display Exception information:

* **printStackTrace()**: prints the stack trace , exception name and description.
* **toString()**: returns a text message describing the exception name and description.
* **getMessage()**: displays the description of exception

------------------------------------------------------------------------------------------

**What are the exceptions in java**

**IOException:**

1. Reading a network file and got disconnected.
2. Reading a local file that was no longer available.
3. Using some stream to read data and some other process closed the stream.
4. Trying to read/write a file but don't have permission.
5. Trying to write to a file but disk space was no longer available.
6. **Arithmetic Exception**  
   It is thrown when an exceptional condition has occurred in an arithmetic operation.
7. **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.
8. **ClassNotFoundException**This Exception is raised when we try to access a class whose definition is not found
9. **FileNotFoundException**This Exception is raised when a file is not accessible or does not open.
10. **IOException**It is thrown when an input-output operation failed or interrupted
11. **InterruptedException**It is thrown when a thread is waiting , sleeping , or doing some processing , and it is interrupted.
12. **NoSuchFieldException**It is thrown when a class does not contain the field (or variable) specified
13. **NoSuchMethodException**It is thrown when accessing a method which is not found.
14. **NullPointerException**This exception is raised when referring to the members of a null object. Null represents nothing
15. **NumberFormatException**This exception is raised when a method could not convert a string into a numeric format.
16. **RuntimeException**This represents any exception which occurs during runtime.
17. **StringIndexOutOfBoundsException**It is thrown by String class methods to indicate that an index is either negative than the size of the string

**Use of java script executor in selenium**

JavascriptExecutor js = (JavascriptExecutor)driver;// JavascriptExecutor js = (JavascriptExecutor) driver

js.executeScript("window.scrollBy(0,600)"); // js.executeScript(Script,Arguments);

**example with xpath:**

js.executeScript("document.getElement(By.xpath(\"//div[@class='btnContr']/input[@onclick='return cancel();']\").click()");

DesiredCapabilities is a class

Assertions – In order to check validations we use assertions.

Difference between if and Assert?

An if statement will not raise an exception when the condition is false. Assertion will raise an assertion error when the condition is false. An assertion exception will stop the program unless the exception is handled by a try-except block.

Joins in sql and second max salaries of an employee – Done

**SELECT** **MAX**(Salary) **From** Employee **WHERE** Salary **<** ( **SELECT** **Max**(Salary) **FROM** Employee);

Nth max salary:

SELECT TOP 1 salary FROM (

SELECT TOP 3 salary

FROM employees

ORDER BY salary DESC) AS emp

ORDER BY salary ASC

Read more: <http://www.java67.com/2015/01/second-highest-salary-in-mysql-and-sql-server.html#ixzz5l4RnCDrQ>

Table grid functionality – Done

**<input name="Name Locator" value="selenium">Hello</input>**

Here getText() retrives Hello - driver.findElement(By.name("Name Locator")).getText();

getAttribute() retrives Selenium - driver.findElement(By.name("Name Locator")).getAttribute("value")

**Q) Which of the following command is used to get all the links present on the page?**

**A) findelements**

When webdriver could not find any element it is **ELEMENT NOT VISIBLE EXCEPTION**

**Action** is an interface, and **Actions** is the class.

**Q) What will be returned for an Element not found by findElements method in selenium?**

**A)** A list of all WebElements, or an empty list if nothing matches.

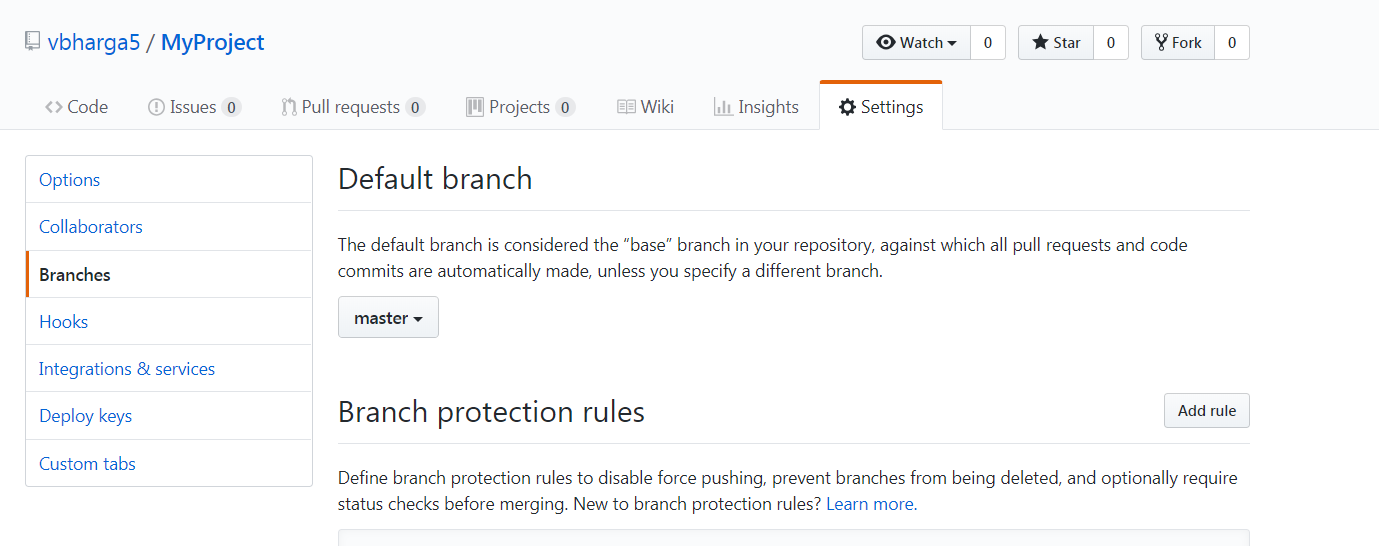
**Close() is used to close the current window, quitting the browser if it's the last window currently open.**

**In webdriver, which method closes the open browser? Close() method**

**Q) To find no of elements size..we use driver.findElements(By.id(“”)).size();**

**/html/body/input is relative xpath**

**Can we delete master branch** – Yes, we can delete master branch when we change the master branch to any other branch in default branch section under github🡪Repo🡪Settings🡪Branches🡪default branch



White box testing and black box testing

The Differences Between [Black Box Testing](http://softwaretestingfundamentals.com/black-box-testing/) and [White Box Testing](http://softwaretestingfundamentals.com/white-box-testing/) are listed below.

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Black Box Testing** | **White Box Testing** |
| *Definition* | Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester | White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester. |
| *Levels Applicable To* | Mainly applicable to higher levels of testing:[Acceptance Testing](http://softwaretestingfundamentals.com/acceptance-testing/)  [System Testing](http://softwaretestingfundamentals.com/system-testing/) | Mainly applicable to lower levels of testing:[Unit Testing](http://softwaretestingfundamentals.com/unit-testing/)  [Integration Testing](http://softwaretestingfundamentals.com/integration-testing/) |
| *Responsibility* | Generally, independent Software Testers | Generally, Software Developers |
| *Programming Knowledge* | Not Required | Required |
| *Implementation Knowledge* | Not Required | Required |
| *Basis for Test Cases* | Requirement Specifications | Detail Design |

Write sauce labs and selenium grid code

**Smoke and sanity testing**

* Smoke testing is a wide approach where all areas of the software application are tested without getting into too deep. However, a sanity software testing is a narrow regression testing with a focus on one or a small set of areas of functionality of the software application.
* Smoke testing of the software application is done to check whether the build can be accepted for through software testing. Sanity testing of the software is to ensure whether the requirements are met or not.
* **Sanity**: a subset of regression test cases are executed that to check whether it rectified the software bugs or issues and no other software bug is introduced by the changes.

**Offline testing** – mirror of production just like stage

* Allow merge commits 

Add all commits from the head branch to the base branch with a merge commit.

* Allow squash merging 

Combine all commits from the head branch into a single commit in the base branch.

* Allow rebase merging 

Add all commits from the head branch onto the base branch individually.

Rebase: This moves the entire feature branch to begin on the tip of the master branch, effectively incorporating all of the new commits in master

Merge commits: retains all of the commits in your branch and creates an extra commit with commits on the base branch.

Merge Squash: retains the changes but omits the individual commits from history

git pull = git fetch + git merge

**What’s New and What’s Changed with Selenium 3**

Selenium 1.0 (IDE) + RC = Selenium 2.0

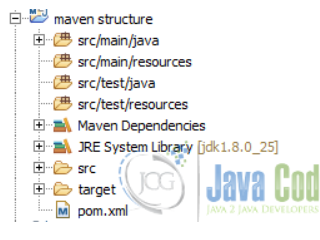
[**Selenium**](https://www.youtube.com/playlist?list=PLyGqUe6Oa_5FRM-L27FnNe8b279Z2txhJ) 3.0 was developed in this year with the new features. It is a combination of Selenium 2.0 – Selenium 1.0, which means it supports Selenium 2.0 features but doesn’t have support for Selenium 1.0 i.e. Selenium Core.

Selenium 3.0 removed Selenium Core but supports Selenium RC indirectly through back-end Webdriver.

* Java 8 is required.
* The original Selenium Core will be retired, which has a significant impact for teams.
* The Selenium RC APIs have been moved to a “legacy” package, a.k.a. leg-rc
* The support for browser drivers has changed.

|  |  |
| --- | --- |
| **Google** | * Will provide their own [chromedriver](https://sites.google.com/a/chromium.org/chromedriver/) binary |
| **Mozilla** | * Upgrading to Selenium 3 no longer supports the default Mozilla Firefox browser—now onwards to Mozilla's [geckodriver](https://github.com/mozilla/geckodriver/releases) |
| **Apple** | * Safari browser is provided on macOS (Sierra 10.12 or later), and Apple owns the [safaridriver](https://developer.apple.com/library/prerelease/content/releasenotes/General/WhatsNewInSafari/Articles/Safari_10_0.html). * Selenium project will no longer maintain the OSS SafariDriver once Safari 10 ships. |
| **Microsoft** | * Edge browser is provided by Microsoft from their [webdriver server](https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/). * IE versions 9 or above are supported. |

Structure of maven project:



**Integration of testng in maven:**

<plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-surefire-plugin</artifactId>

                <version>2.14.1</version>

                <configuration>

                    <!-- Suite testng xml file to consider for test execution -->

                    <suiteXmlFiles>

                        <suiteXmlFile>testng.xml</suiteXmlFile>

                        <suiteXmlFile>suites-test-testng.xml</suiteXmlFile>

                    </suiteXmlFiles>

                </configuration>

            </plugin>

**You can configure maven to run in offline mode. Add this entry to your settings.xml**

**<offline>true</offline>**

See here for further information:

<http://maven.apache.org/settings.html>

Before you can use offline mode, you have to install all necessary third party jars to your local maven repository.

mvn install:install-file

-Dfile=filename.jar

-DgroupId=com.stackoverflow

-DartifactId=artifact

-Dversion=1.0.0

-Dpackaging=jar

-DcreateChecksum=true

-DgeneratePom=true

### What is a Collection?

A collection is an object that represents a group of individual objects represented as a single unit. Collection is a dynamic array where we can add elements and reduce elements.

Return type of driver.findElements is List<WebElement>

driver.getWindowHandle return type is String

driver.getWindowHandles return type is Set<String>

hasNext() return type is boolean

next() return type is string

**Window handles sample code:**

public class WindowHandle\_Demo {

public static void main(String[] args) throws InterruptedException {

WebDriver driver=new FirefoxDriver();

//Launching the site.

driver.get("http://demo.guru99.com/popup.php");

driver.manage().window().maximize();

driver.findElement(By.xpath("//\*[contains(@href,'popup.php')]")).click();

String MainWindow=driver.getWindowHandle();

// To handle all new opened window.

Set<String> s1=driver.getWindowHandles();

Iterator<String> i1=s1.iterator();

while(i1.hasNext())

{

String ChildWindow=i1.next();

if(!MainWindow.equalsIgnoreCase(ChildWindow))

{

// Switching to Child window

driver.switchTo().window(ChildWindow);

driver.findElement(By.name("emailid"))

.sendKeys("[gaurav.3n@gmail.com](mailto:gaurav.3n@gmail.com)");

driver.findElement(By.name("btnLogin")).click();

driver.close(); // Closing the Child Window

}

}

// Switching to Parent window i.e Main Window.

driver.switchTo().window(MainWindow);

}

}

**:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::**

**To convert string value to integer we use Integer.parseInt();**

Use of finally block and in which cases it will come into picture – To clean up the code like closing browser and closing DB connections

Reverse a number – done

ChromeOptions is a class or interface – It’s a class

Basically, we can switch over the elements in frames using 3 ways.

* **By Index**
* **By Name or Id**
* **By Web Element**

**we can select drop downs by**

Select by visible text,

value and

index

Return type of window handles will be **String**

If we know the window name, we can pass this: **driver.switchTo().window("windowName");**

**Scenario outline using excel sheet in cucumber:**

Feature: User is using an excel spreadsheet with cucumber driving it

Scenario Outline: Data Driven with excel and data sets

When I am on the amps mainscreen

Then I input username and passwords with excel row"<row\_index>" dataset

Examples:

| row\_index |

| 1 |

| 2 |

| 3 |

| 4 |

Step File:

//Excel Steps

@When("^I am on the amps mainscreen$")

public void i\_am\_on\_the\_amps\_mainscreen() {

System.out.println("Im loading");

}

//Excel Steps

@Then("^I input username and passwords with excel row\"([^\"]\*)\" dataset$")

public void i\_input\_username\_and\_passwords\_with\_excel\_row\_dataset(int rownum) throws IOException {

login.readExcel(rownum);

}

Actual Code:

public void readExcel (int row) throws IOException{

File src=new File("src/test/resources/username.xlsx");

FileInputStream fis=new FileInputStream(src);

XSSFWorkbook srcBook= new XSSFWorkbook(fis);

XSSFSheet sourceSheet = srcBook.getSheetAt(0);

XSSFRow sourceRow = sourceSheet.getRow(row);

XSSFCell username=sourceRow.getCell(0);

XSSFCell password=sourceRow.getCell(1);

String userExcel = username.getStringCellValue();

String pwExcel = password.getStringCellValue();

System.out.println("The username is" +userExcel);

System.out.println("The password is" +pwExcel);

log.info("The username on " +row + " is: "+userExcel);

log.info("The password on "+row+ " is: "+pwExcel);

driver.findElement(txtbox\_username).sendKeys(userExcel);

driver.findElement(txtbox\_password).sendKeys(pwExcel);

driver.findElement(btn\_logon).click();

}

**Usage of hashmap example:**

Test Steps:

1. Launch Calendar in Month View
2. Capture the current month (eg - March) and save in String curMonth
3. Now swipe(right to left) for Month to Change
4. Capture the value of the Month displayed on screen (Now comes April) and save in String nextMonth
5. Verify the value nextMonth is actually the value next to curMonth

Below code sample which I am trying to use. read the comments

HashMap<String, Integer> Table = new HashMap<String, Integer>();

Table.put("JAN", 1);

Table.put("FEB", 2);

Table.put("MAR", 3);

Table.put("APR", 4);

Table.put("MAY", 5);

Table.put("JUN", 6);

String testdata="MAR";

System.out.println("------------------------------------------------");

Integer curMon = Table.get(testdata);

System.out.println(curMon);

//Here output will be 3 which it will fetch from the hashmap table

//Now I am adding 1 from the return value which is 3. So 3+1=4

Integer newMonthValue=curMon+1;

//Now how to associate the integer value which is 4 stored in newMonthValue and map

**getText() and getAttribute() :**

<input attr1='a' attr2='b' attr3='c'>foo</input>

getAttribute(attr1) you get 'a'

getAttribute(attr2) you get 'b'

getAttribute(attr3) you get 'c'

getText() with no parameter you can only get 'foo'

If there are any elements in the drop down without select class and with div tag then how - select drop down using bootstrap – we use same method like **driver.findElements similar to table concept – Example for findelements basically we will be using it as – driver.findElements(By.Xpath(“”)//ui//li) – so it will return all the li tags under ui tag**

How we pass dynamic xpaths in run time - Dynamic xpaths are the one which use **contains** in xpath.

Can we change the string which are already declared - Done

What are generics - done

What is finalize – done – check example

**Why stale element exception occurs:**

Possible causes for an stale element exception-

1 -The element has been deleted entirely.

2- The element is no longer attached to the DOM.

**What we need to do if stale element exception occurs**

**Solution-1:**

Most easy easy out to handle this is to refresh the page, On refreshing it, most of the time driver found the element, But its not the perfect solution-

Driver.navigate().refersh();  
Driver.findElement(By.id(“property”)).click();  
Driver.navigate().refersh();  
Driver.findElement(By.id(“property”)).click();

**Solution-2:**

**for**(**int** i=0;i<=3;i++)

{

**try**

{

driver.findElement("Property").click();

**break**;

}

}

**catch**(Exception e)

{

System.***out***.println("Exception-Element not found");

}

}

**Solution 3-**

Use a do-While loop on an element which is causing stale element exception-Initialize a counter with Zero and iterate and identify webelement, once its enabled and displayed perform action and exit from loop.

Using a do-while loop is most generic and perfect solution for such stale element exceptions, it should be able to find element even if it appear a bit late after loading a web page.

int Counter=0

do  
{  
try  
{  
String str ;

if(driver.findelement(By.id()).isenabled() && driver.findelement(By.id()).isdisplayed() )  
{  
Counter=Counter+1;  
driver.findelement(By.id()).click();  
break;  
}  
}  
catch(Exception ex)  
{

}  
)  
while(Counter == 0)

Solution:4

**wait.until(ExpectedConditions.presenceOfElementLocated(By.id("table")));**

StaleElementReferenceException and retrieve the element again. This method updates the element by redrawing it and we can access the referenced element.

**wait.until(ExpectedConditions.refreshed(ExpectedConditions.stalenessOf("table")));**

**----------------------------------------------------------------------------------------------------------------------------**

**Broken links code in selenium:**

**public class VerifyLinks {**

**public static void main(String[] args)**

**{**

**WebDriver driver=new FirefoxDriver();**

**driver.manage().window().maximize();**

**driver.get("http://www.google.co.in/");**

**List<WebElement> links=driver.findElements(By.tagName("a"));**

**System.out.println("Total links are "+links.size());**

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

**{**

**WebElement ele= links.get(i);**

**String url=ele.getAttribute("href");**

**verifyLinkActive(url);**

**}**

**}**

**public static void verifyLinkActive(String linkUrl)**

**{**

**try**

**{**

**URL url = new URL(linkUrl);**

**HttpURLConnection httpURLConnect=(HttpURLConnection)url.openConnection();**

**httpURLConnect.setConnectTimeout(3000);**

**httpURLConnect.connect();**

**if(httpURLConnect.getResponseCode()==200)**

**{**

**System.out.println(linkUrl+" - "+httpURLConnect.getResponseMessage());**

**}**

**if(httpURLConnect.getResponseCode()==HttpURLConnection.HTTP\_NOT\_FOUND)**

**{**

**System.out.println(linkUrl+" - "+httpURLConnect.getResponseMessage() + " - "+ HttpURLConnection.HTTP\_NOT\_FOUND);**

**}**

**} catch (Exception e) { }**

**}**

How we write xpaths n IE and highlight them : MRI is the add on which supports from IE 6+.

What are immutables in java - Firstly, all of the java.lang package wrapper classes are immutable: Boolean, Byte, Character, Double, Float, Integer, Long, Short, String.

e.printstacktrace meaning and why we use it and what error it displays

Can interface implements an interface : No, interface extends another Interface

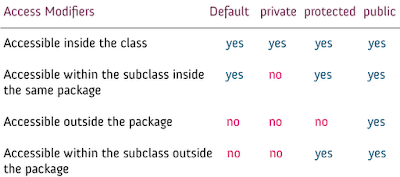
Javascript executor examples - done

Use of java script executor to drag an element to particular webelement – done

Method overloading is an **example** of static **polymorphism**, while method overriding is an **example** of dynamic **polymorphism.**

[Encapsulation](http://contribute.geeksforgeeks.org/encapsulation-in-java/) is data hiding(information hiding) while Abstraction is detail hiding(implementation hiding).

Encapsulation examples are access specifiers and abstraction examples are abstract class.

We look at these Access Specifiers in more detail.  
  
[](http://2.bp.blogspot.com/-LTO8bwD3c6o/Tw1GoeAlHeI/AAAAAAAABlE/0EX24ENt9uY/s1600/access+specifiers.png)

The protected access specifier cannot be applied to class and interfaces.

**Class and** [**Interface**](https://beginnersbook.com/2013/05/java-interface/) **cannot be declared as private.**

Protected data member and method are only accessible by the classes of the same package and the subclasses present in any package.

Junit:

* **@Test:**Annotation lets the system know that the method annotated as @Test is a test method. There can be multiple test methods in a single test script.
* **@Before:**Method annotated as @Before lets the system know that this method shall be executed every time before each of the test method.
* **@After:**Method annotated as @After lets the system know that this method shall be executed every time after each of the test method.
* **@BeforeClass:**Method annotated as @BeforeClass lets the system know that this method shall be executed once before any of the test method.
* **@AfterClass:**Method annotated as @AfterClass lets the system know that this method shall be executed once after any of the test method.
* **@Ignore:**Method annotated as @Ignore lets the system know that this method shall not be executed.

**Backlog:** prioritizing the user stories or features

**Sprint Backlog:**

* The **sprint backlog** is a list of tasks identified by the Scrum team to be completed during the Scrum **sprint**.

**Spillover:**

#### Incomplete user stories in current sprint

#### Lack of communication and collaboration

**Product Backlog Refinement**, also referred to as Product **Backlog** Grooming, is a method for keeping the **backlog** updated, clean and orderly. It is a basic process in Scrum. PBR is a collaborative discussion process which starts at the end of one sprint to confirm whether the **backlog** is ready for the next sprint

**Code Refactoring** is the process of clarifying and simplifying the design of existing code, without changing its behavior. Agile teams are maintaining and extending their code a lot from iteration to iteration, and without continuous refactoring, this is hard to do

**Static:**

**Static variables**:

* We can create static variables at class-level only
* static block and static variables are executed in order they are present in a program

## Static Block

Static block is used for initializing the static variables

class JavaExample{

static int num;

static String mystr;

static{

num = 97;//value is initialized here for declared variable

mystr = "Static keyword in Java";

}

public static void main(String args[])

{

System.out.println("Value of num: "+num);

System.out.println("Value of mystr: "+mystr);

}

}

Output:

Value of num: 97

Value of mystr: Static keyword in Java

Static methods can be directly called without using object.

**Static methods can**'t **be** overriden as it is part of a class rather than an object.

class JavaExample3{

static int var1;

static String var2;

//This is a Static Method

static void disp(){

System.out.println("Var1 is: "+var1);

System.out.println("Var2 is: "+var2);

}

public static void main(String args[])

{

disp();

}

}

If once static variable value is initialized as, suppose static int a =2;

If we again overwrite it as a=4;

Again a=6;

Then value will be the last overwritten one i.e., 6

### Example 2: Static method accessed directly in static and non-static method

class JavaExample{

static int i = 100;

static String s = "Beginnersbook";

//Static method

static void display()

{

System.out.println("i:"+i);

System.out.println("i:"+s);

}

//non-static method

void funcn()

{

//Static method called in non-static method

display();

}

//static method

public static void main(String args[])

{

JavaExample obj = new JavaExample();

//You need to have object to call this non-static method

obj.funcn();

//Static method called in another static method

display();

}}

### Static class Example

class JavaExample{

private static String str = "BeginnersBook";

//Static class

static class MyNestedClass{

//non-static method

public void disp() {

/\* If you make the str variable of outer class

\* non-static then you will get compilation error

\* because: a static class cannot access non-

\* static members of the outer class.

\*/

System.out.println(str);

}

}

public static void main(String args[])

{

/\* To create instance of nested class we didn't need the outer

\* class instance but for a regular nested class you would need

\* to create an instance of outer class first

\*/

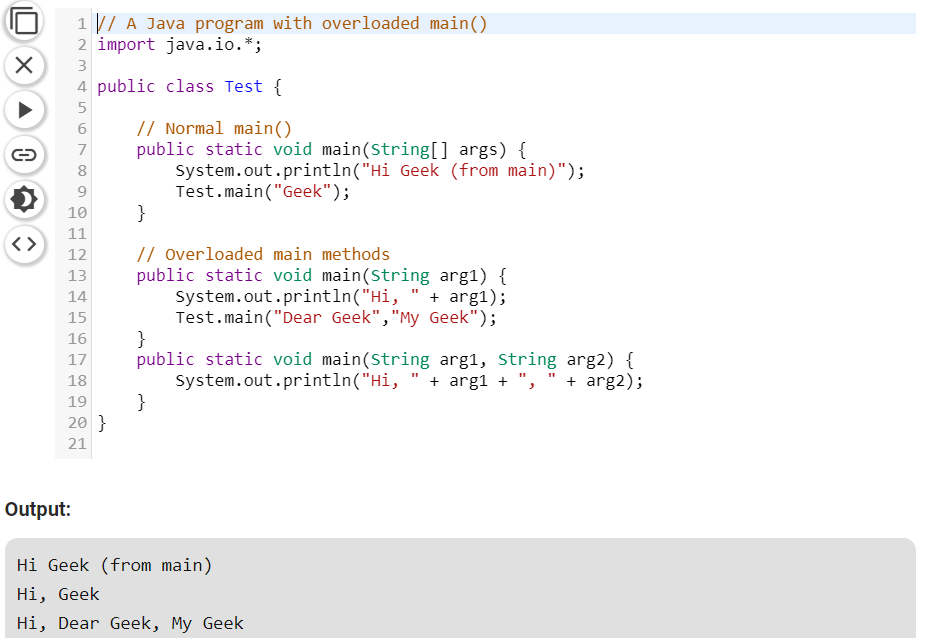
JavaExample.MyNestedClass obj = new JavaExample.MyNestedClass();

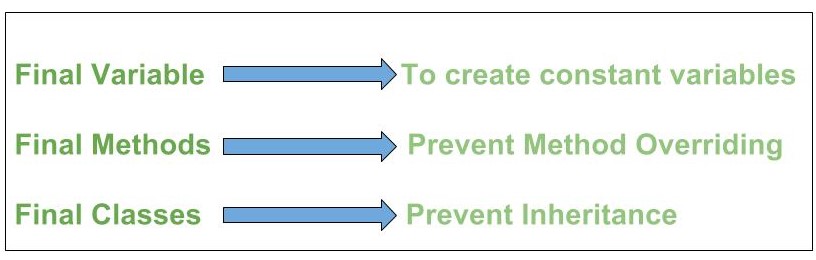
obj.disp();

}}

Output: BeginnersBook

**Overloading a main method:**





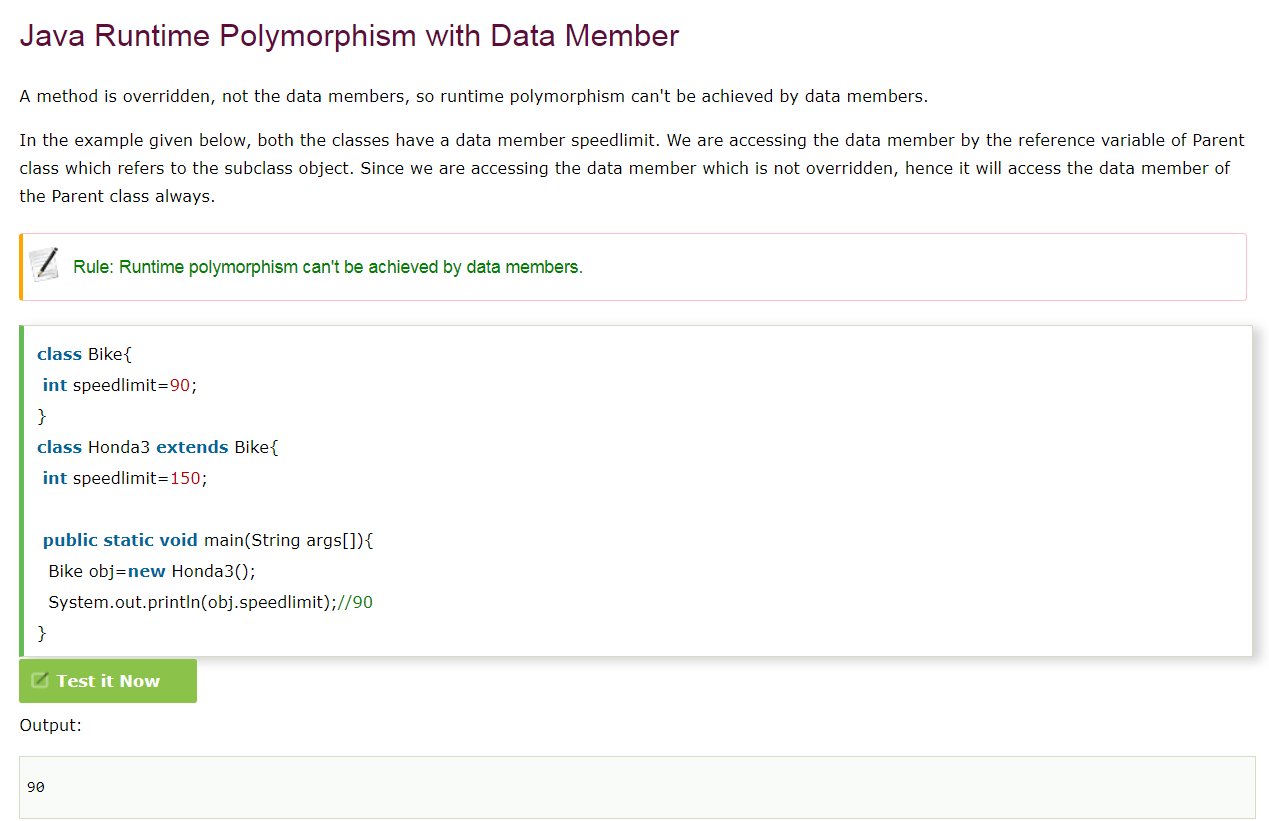
**Final classes**

When a class is declared with final keyword, it is called a final class. A final class cannot be extended(inherited). There are two uses of a final class :

One is definitely to prevent [inheritance](https://www.geeksforgeeks.org/inheritance-in-java/), as final classes cannot be extended. For example, all [Wrapper Classes](https://www.geeksforgeeks.org/wrapper-classes-java/) like [Integer](https://www.geeksforgeeks.org/java-lang-integer-class-java/),[Float](https://www.geeksforgeeks.org/java-lang-float-class-in-java/) etc. are final classes. We can not extend them.

1. final class A
2. {
3. // methods and fields
4. }
5. // The following class is illegal.
6. class B extends A
7. {
8. // COMPILE-ERROR! Can't subclass A
9. }

The other use of final with classes is to [create an immutable class](https://www.geeksforgeeks.org/create-immutable-class-java/) like the predefined [String](https://www.geeksforgeeks.org/string-class-in-java/) class.You can not make a class immutable without making it final. **So, to make class as immutable we use final keyword for class**



### static binding

When type of the object is determined at compiled time(by the compiler), it is known as static binding.

If there is any private, final or static method in a class, there is static binding.

Example:

1. **class** Dog{
2. **private** **void** eat(){System.out.println("dog is eating...");}
4. **public** **static** **void** main(String args[]){
5. Dog d1=**new** Dog();
6. d1.eat();
7. }
8. }



**Why multiple inheritance is possible by interfaces and not classes ?**

A class can implement any number of interfaces. In this case there is no ambiguity even though both the interfaces are having same method. Why? Because methods in an interface are always [abstract](https://beginnersbook.com/2013/05/java-abstract-class-method/) by default, which doesn’t let them give their implementation (or method definition ) in interface itself. So, we will be implementing method body in a separate class or a main class and we call this implemented method.

Example:

interface X

{

public void myMethod();

}

interface Y

{

public void myMethod();

}

class JavaExample implements X, Y

{

public void myMethod()

{

System.out.println("Implementing more than one interfaces");

}

public static void main(String args[]){

JavaExample obj = new JavaExample();

obj.myMethod();

}

}

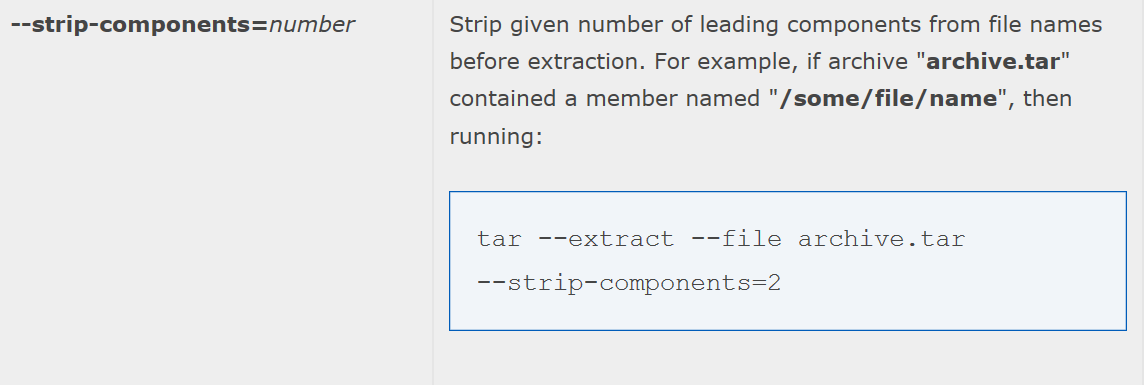
Output:

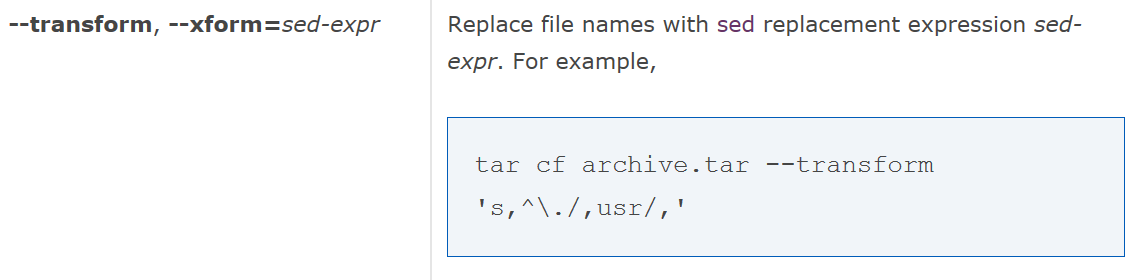
Implementing more than one interfaces

## Java Recursion Example: Finite times

1. **public** **class** RecursionExample2 {
2. **static** **int** count=0;
3. **static** **void** p(){
4. count++;
5. **if**(count<=5){
6. System.out.println("hello "+count);
7. p();
8. }
9. }
10. **public** **static** **void** main(String[] args) {
11. p();
12. }
13. }

|  |  |
| --- | --- |
| zcvf - to make it to tar |  |
| xvf - unzip |  |
| scp - copy from one server to other |  |
| nohup - not to interrupt any batch when we run in server, if we placed the larger file, even when we logout or abort |  |
| wget - to get details from server |  |
| To zip a file: zip <File name> |  |
| To Unzip a file: if unzipping is in same location Unzip <filename> | If unzip is in different location unzip -d <dest-path> <filename> |





**Bootstrap drop downs code:**

public class BootstrapDropDown

{

public static void main(String[] args) throws InterruptedException

{

WebDriver driver=new FirefoxDriver();

driver.manage().window().maximize();

driver.get("G:\\Testing\_Utilities\\Selenium\_Practice\\BootstrapDropdown.html");

driver.manage().timeouts().implicitlyWait(5000, TimeUnit.SECONDS);

driver.findElement(By.id("menu1")).click();

List<WebElement> li=driver.findElements(By.xpath(".//ul[@class='dropdown-menu']/li/a"));

for(WebElement el:li)

{

if(el.getText().equals("CSS"))

{

el.click();

Thread.sleep(3000);

System.out.println("Element Clicked");

break;

}

System.out.println(el.getText());

}

}

|  |
| --- |
| zcvf - to make it to tar |
| xvf - unzip |
| scp - copy from one server to other |
| nohup - not to interrupt any batch when we run in server, if we placed the larger file, even when we logout or abort |
| wget - to get details from server |
|  |

:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::TO DO:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

IO files, enum

Serializable

Cloneable

What are ask ids in sonar

Java interview questions

Selenium interview questions

Git interview questions

Java programs

What are different exceptions in java

Jenkins interview questions

How we are retrieving URL and credentials in our framework from config.properties

How to select future dates in a calendar using selenium

What are wrapper classes

check sikuli

difference between string, string buffer and string builder – Done

declarative and scripted pipelines

cmws and cmis differences

**Difference between SOAP and REST**

**Health care domain – How claims process will be**

Agile principles

Risk management and risk mitigation

:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

Generics examples:

class Test<T>

{

    // An object of type T is declared

  T obj;

    Test(T obj) {  this.obj = obj;  }  // constructor

    public T getObject()  { return this.obj; }

}

// Driver class to test above

class Main

{

    public static void main (String[] args)

    {

        // instance of Integer type

        Test <Integer> iObj = new Test<Integer>(15);

        System.out.println(iObj.getObject());

        // instance of String type

        Test <String> sObj =

                          new Test<String>("GeeksForGeeks");

        System.out.println(sObj.getObject());

    }

}

We can also pass multiple Type parameters in Generic classes.

class Test<T, U> {

 T obj1;  // An object of type T

    U obj2;  // An object of type U

    // constructor

    Test(T obj1, U obj2)

    {

        this.obj1 = obj1;

        this.obj2 = obj2;

    }

    // To print objects of T and U

    public void print()

    {

        System.out.println(obj1);

        System.out.println(obj2);

    }

}

// Driver class to test above

class Main

{

    public static void main (String[] args)

    {

        Test <String, Integer> obj =

            new Test<String, Integer>("GfG", 15);

        obj.print();

    }

}