Selenium Interview Questions

1. What is Automation Testing?

Automation testing is the process of testing a software or application using an automation testing tool to find the defects. In this process, executing the test scripts and generating the results are performed automatically by automation tools. It is required when we have huge amount of regression test cases.Some most popular tools to do automation testing are HP QTP/UFT, Selenium WebDriver etc.,

2. What are the benefits of Automation Testing?

This is one of the common interview questions in any Automation testing job.

1. Saves time and money. Automation testing is faster in execution.
2. Reusability of code. Create one time and execute multiple times with less or no maintenance.
3. Easy reporting. It generates automatic reports after test execution.
4. Easy for compatibility testing. It enables parallel execution in the combination of different OS and browser environments.
5. Low-cost maintenance. It is cheaper compared to manual testing in a long run.
6. Automated testing is more reliable.
7. Automated testing is more powerful and versatile. Automation tools allow us to integrate with Cross Browser Testing Tools, Jenkins, Github etc.,
8. It is mostly used for regression testing. Supports execution of repeated test cases.
9. Minimal manual intervention. Test scripts can be run unattended.
10. Maximum coverage. It helps to increase the test coverage.

3. What are the challenges and limitations of Selenium WebDriver?

As we all know Selenium WebDriver is a tool which automates the browser to mimic real user actions on the web. Selenium is a free open source testing tool. Some of the challenges with selenium webdriver are as follows

1. We cannot test windows application
2. We cannot test mobile apps
3. Limited reporting
4. Handling dynamic Elements
5. Handling page load
6. Handling pop up windows
7. Handling captcha

4. What type of tests have you automated?

Our main focus is to automate test cases to do *Regression testing, Smoke testing, and Sanity testing.* Sometimes based on the project and the test time estimation, we do focus on End to End testing.

5. How many test cases you have automated per day?

It depends on Test case scenario complexity and length. I did automate 5-15 test scenarios per day when the complexity is limited. Sometimes just 1 or fewer test scenarios in a day when the complexity is high.

6. What is a Framework?

A framework defines a set of rules or best practices which we can follow in a systematic way to achieve the desired results. There are different types of automation frameworks and the most common ones are:

- Keyword Driven Testing Framework
- Modular Framework
- Data Driven Testing Framework
- Hybrid Testing Framework

7. Have you created any Framework?

If you are a beginner: You can say "No, I didn't get a chance to create framework from the scratch. I have used the framework which is already available. My contribution is mostly in creating test cases by using the existing framework."

If you are a beginner but have good knowledge on creating framework: You can say "Yes, I have involved in developing framework along with other automation tester in my company."

If you are an experienced tester: You can say "I have contributed in developing framework." or You can say "Yes, I have created framework from the scratch. I designed the framework from the scratch."

8. Can you explain the Framework which you have used in your Selenium Project?

We have to clearly explained each component of Framework like the following

Explain about folder structure, test data implementation, Reporting, Configurable options for environment, partial or full test suite execution.

9. Why do you prefer Selenium Automation Tool?

1. Free and open source
2. Have large user base and helping communities
3. Cross browser compatibility

4. Platform compatibility
5. Multiple programming languages support such as Java, Perl, Phyton, Php, C# etc.,

10. What is Selenium?

Selenium is an open source (free) automated testing suite to test web applications. It supports different platforms and browsers. It has gained a lot of popularity in terms of web-based automated testing and giving a great competition to the famous commercial tool HP QTP (Quick Test Professional) AKA HP UFT (Unified Functional Testing).

Selenium is a set of different software tools. Each tool has a different approach in supporting web based automation testing.

It has four components namely,

i. Selenium IDE (Selenium Integrated Development Environment)
ii. Selenium RC (Selenium Remote Control)
iii. Selenium WebDriver
iv. Selenium Grid

11. What is Selenium IDE?

Selenium IDE (Integrated Development Environment) is a Firefox plugin. It is the simplest framework in the Selenium Suite. It allows us to record and playback the scripts. Even though we can create scripts using Selenium IDE, we need to use Selenium RC or Selenium WebDriver to write more advanced and robust test cases.

12. What is Selenese?

Selenese is the language which is used to write test scripts in Selenium IDE.

13. Which is the only browser that supports Selenium IDE to be used?

Firefox

14. What is Selenium RC?

Selenium RC AKA Selenium Remote control / Selenium 1. Selenium Remote Control was the main Selenium project for a long time before the WebDriver merge brought up Selenium 2. Selenium 1 is still actively supported (in maintenance mode). It relies on JavaScript for automation. It supports Java, Javascript, Ruby, PHP, Python, Perl and C#. It supports almost every browser out there.

15. What is Selenium WebDriver?

Selenium WebDriver AKA Selenium 2 is a browser automation framework that accepts commands and sends them to a browser. It is implemented through a browser-specific driver. It controls the browser by directly communicating with it. Selenium WebDriver supports Java, C#, PHP, Python, Perl, Ruby.

16. What is Selenium Grid?

Selenium Grid is a tool used to run tests on different machines against different browsers in parallel. That is, running multiple tests at the same time against different machines running different browsers and operating systems.

In simple words, it is used to distribute your test execution on multiple platforms and environments concurrently.

17. When do you use Selenium Grid?

Selenium Grid can be used to execute same or different test scripts on multiple platforms and browsers concurrently so as to achieve distributed test execution

18. What are the advantages of Selenium Grid?

It allows running test cases in parallel thereby saving test execution time.
It allows multi-browser testing
It allows us to execute test cases on multi-platform

19. What is a hub in Selenium Grid?

A hub is a server or a central point that controls the test executions on different machines.

20. What is a node in Selenium Grid?

Node is the machine which is attached to the hub. There can be multiple nodes in Selenium Grid.

21. What are the types of WebDriver APIs available in Selenium?

- Firefox Driver
- Gecko Driver
- InternetExplorer Driver
- Chrome Driver
- HTMLUnit Driver
- Opera Driver
- Safari Driver
- Android Driver
- iPhone Driver
- EventFiringWebDriver

22. Which WebDriver implementation claims to be the fastest?

The fastest implementation of WebDriver is the HTMLUnitDriver or any headless driver

Headless browser testing using Selenium WebDriver is done to test the application without any visual interruption.

Two widely used headless drivers with practical examples.

1. HTMLUnitDriver
2. PhantomJSDriver

Other Headless Drivers (Chrome, Firefox etc.) can also be used

Advantages of Performing Headless Browser Testing In Selenium:

1. It's faster. The performance is better compared to browser automation. Automated testing is to automate a browser to ensure that the application is running as expected. Running regression scripts take lots of time. With headless browsers, we could save time.
2. It enables you to run the scripts on a system which doesn't have a browser.
3. Imagine you have a situation to run some tests on a version of google chrome and there is no such version of google chrome on your local system. In this case, you could use the headless browser, most of the headless browsers support browser versions.

Disadvantages of Headless Browsers Testing:

1. Debugging is a bit difficult using headless browsers. Here browser is not visible, the only way is to capture a screenshot.

23. What are the Programming Languages supported by Selenium WebDiver?

- C#
- Java
- Python
- Ruby
- Perl
- PHP

24. What are the Operating Systems supported by Selenium WebDriver?

- Windows
- Linux
- Apple

25. What are the Open-source Frameworks supported by Selenium WebDriver?

- JUnit
- TestNG

26. What are the Locators available in Selenium?

In Selenium WebDriver, there are 8 different types of locators:

1. ID
2. ClassName
3. Name
4. TagName
5. LinkText
6. PartialLinkText
7. XPath
8. CSS Selector

27. What is an XPath?

XPath is used to locate the elements. Using XPath, we could navigate through elements and attributes in an XML document to locate web elements such as textbox, button, checkbox, Image etc., in a web page.

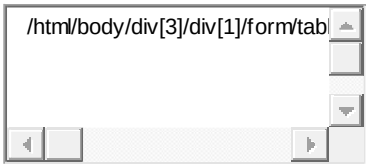
28. What is the difference between "/" and "//"

Single Slash "/" – Single slash is used to create XPath with absolute path i.e. the XPath would be created to start selection from the document node/start node.

Double Slash "//" – Double slash is used to create XPath with relative path i.e. the XPath would be created to start selection from anywhere within the document.

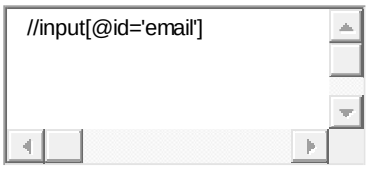
29. What is the difference between Absolute Path and Relative Path?

Absolute XPath starts from the root node and ends with desired descendant element's node. It starts with top HTML node and ends with input node. It starts with a single forward slash(/) as shown below.



1 /html/body/div[3]/div[1]/form/table/tbody/tr[1]/td/input

Relative XPath starts from any node in between the HTML page to the current element's node(last node of the element). It starts with a double forward slash(//) as shown below.



1 //input[@id='email']

30. What is the difference between Assert and Verify in Selenium?

Assert: In simple words, if the assert condition is true then the program control will execute the next test step but if the condition is false, the execution will stop and further test step will not be executed.

TestNg Assert Class will be used

Verify: In simple words, there won't be any halt in the test execution even though the verify condition is true or false.

If else conditions can be used

31. What are Soft Assert and Hard Assert in Selenium?

Soft Assert: Soft Assert collects errors during *@Test* Soft Assert does not throw an exception when an assert fails and would continue with the next step after the assert statement.

Use SoftAssert Object of TestNG needs to be created

Hard Assert: Hard Assert throws an *AssertException* immediately when an assert statement fails and test suite continues with next *@Test*

Use Assert class of TestNg with out creating object.

32. What are the verification points available in Selenium?

In Selenium IDE, we use Selenese Verify and Assert Commands as Verification points
In Selenium WebDriver, there is no built-in features for verification points. It totally depends on our coding style. some of the Verification points are

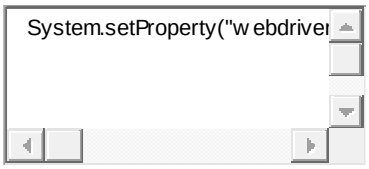
- To check for page title
- To check for certain text
- To check for certain element (text box, button, drop down, etc.)

33. How to launch a browser using Selenium WebDriver?

WebDriver is an Interface. We create an Object of a required driver class such as FirefoxDriver, ChromeDriver, InternetExplorerDriver etc.,

To launch Firefox Driver:
WebDriver driver = new FirefoxDriver();

Note: *If you use geckodriver with Selenium, you must upgrade to Selenium 3.3. Here we have to set the property as follows*




```
1 System.setProperty("webdriver.gecko.driver", "D:\\Selenium
  Environment\\Drivers\\geckodriver.exe");
```

To launch Chrome Driver:
WebDriver driver = new ChromeDriver();

To launch Internet Explorer Driver:
WebDriver driver = new InternetExplorerDriver();

To launch Safari Driver:
WebDriver driver = new SafariDriver();

34. Is the FirefoxDriver a Class or an Interface?

FirefoxDriver is a Java class, and it implements the *WebDriver* interface.

35. What is the super interface of WebDriver?

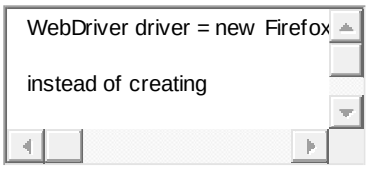
SearchContext.

36. Explain the line of code *Webdriver driver = new FirefoxDriver();* ?

1 Webdriver driver = new FirefoxDriver();

'*WebDriver*' is an interface and we are creating an object of type WebDriver instantiating an object of FirefoxDriver class.

37. We do create a reference variable 'driver' of type WebDriver




```
1 WebDriver driver = new FirefoxDriver();
2
3 instead of creating
4
5 FirefoxDriver driver = new FirefoxDriver();
```

What is the purpose of doing this way?

If we create a reference variable driver of type WebDriver then we could use the same driver variable to work with any browser of our choice such as IEDriver, SafariDriver etc.,

38. What are the different exceptions you have faced in Selenium WebDriver?

Some of the exceptions I have faced in my current project are

1. ElementNotVisibleException
2. StaleElementReferenceException

Element Not visible Exception:

This exception will be thrown when you are trying to locate a particular element on webpage that is not currently visible eventhough it is present in the DOM. Also sometimes, if you are trying to locate an element with the xpath which associates with two or more element.

Stale Element Reference Exception:

A Stale Element Reference Exception is thrown in one of two cases, the first being more common than the second.

The two reasons for Stale element reference are

1. The element has been deleted entirely.
2. The element is no longer attached to the DOM.

We face this stale element reference exception when the element we are interacting is destroyed and then recreated again. When this happens the reference of the element in the DOM becomes stale. Hence we are not able to get the reference to the element.

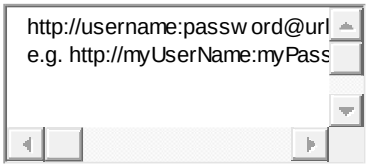
Some other exceptions we usually face are as follows:

- WebDriverException

- IllegalStateException
- TimeoutException
- NoAlertPresentException
- NoSuchWindowException
- NoSuchElementException etc

39. How to Login into any site if it is showing an Authentication Pop-Up for Username and Password?

To do this we pass username and password with the URL



1 http://username:password@url
2 e.g. http://myUserName:myPassword@softwaretestingmaterial.com

40. What are the types of waits available in Selenium WebDriver?

In Selenium we could see three types of waits such as Implicit Waits, Explicit Waits and Fluent Waits.

- Implicit Waits
- Explicit Waits
- Fluent Waits

41. What is Implicit Wait In Selenium WebDriver?

Implicit waits tell to the WebDriver to wait for a certain amount of time before it throws an exception. Once we set the time, WebDriver will wait for the element based on the time we set before it throws an exception. The default setting is 0 (zero). We need to set some wait time to make WebDriver to wait for the required time.

42. What is WebDriver Wait In Selenium WebDriver?

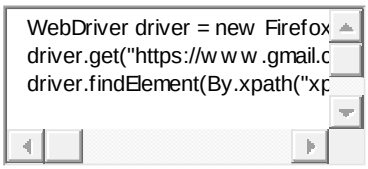
WebDriverWait is applied on a certain element with defined *expected condition* and *time*. This wait is only applied to the specified element. This wait can also throw an exception when an element is not found.

43. What is Fluent Wait In Selenium WebDriver?

FluentWait can define the maximum amount of time to wait for a specific condition and frequency with which to check the condition before throwing an "*ElementNotVisibleException*" exception.

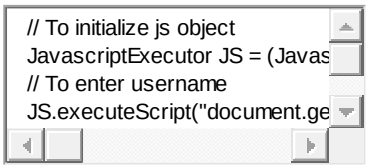
44. How to input text in the text box using Selenium WebDriver?

By using sendKeys() method




```
1 WebDriver driver = new FirefoxDriver();
2 driver.get("https://www.gmail.com");
3 driver.findElement(By.xpath("xpath")).sendKeys("Software Testing Material Website");
```

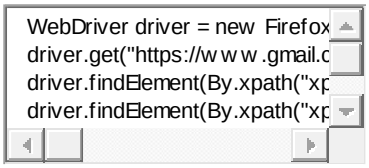
45. How to input text in the text box without calling the sendKeys()?




```
1 // To initialize js object
2 JavascriptExecutor JS = (JavascriptExecutor)webdriver;
3 // To enter username
4 JS.executeScript("document.getElementById('User').value='SoftwareTestingMaterial.com'");
5 // To enter password
6 JS.executeScript("document.getElementById('Pass').value='tester'");
```

46. How to clear the text in the text box using Selenium WebDriver?

By using clear() method




```
1 WebDriver driver = new FirefoxDriver();
2 driver.get("https://www.gmail.com");
3 driver.findElement(By.xpath("xpath_of_element1")).sendKeys("Software Testing Material
  Website");
4 driver.findElement(By.xpath("xpath_of_element1")).clear();
```

47. How to get a text of a web element?

By using getText() method 

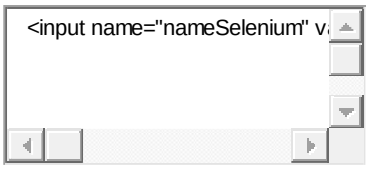
```
 @Test
 public void testmethod(){
 System.setProperty("webdriver.chrome.driver", "D:\\Selenium
 Environment\\Drivers\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.google.com");
    String availableText =
 driver.findElement(By.xpath("//*[@id='gbw']/div/div/div[1]/div[1]/a")).getText();
    System.out.println("Text Available is :"+availableText);
 }
 }
```

48. How to get an attribute value using Selenium WebDriver?

By using getAttribute(value);

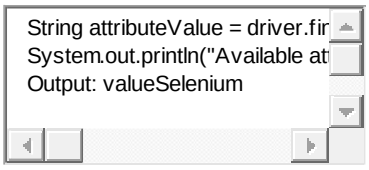
It returns the value of the attribute passed as a parameter.

HTML:



1 <input name="nameSelenium" value="valueSelenium">SoftwareTestingMaterial</input>

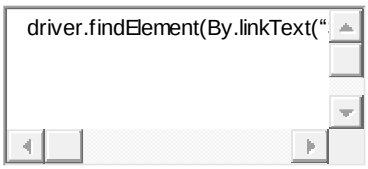
Selenium Code:



1 String attributeValue = driver.findElement(By.name("nameSelenium")).getAttribute("value");
2 System.out.println("Available attribute value is :"+attributeValue);
3 Output: valueSelenium

49. How to click on a hyperlink using Selenium WebDriver?

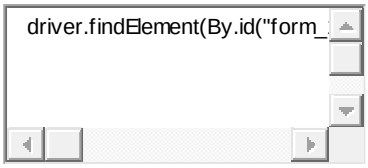
We use click() method in Selenium to click on the hyperlink



1 driver.findElement(By.linkText("Software Testing Material Website")).click();

50. How to submit a form using Selenium WebDriver?

We use "submit" method on element to submit a form

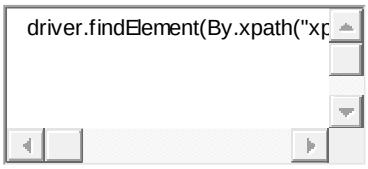


1 driver.findElement(By.id("form_1")).submit();

Alternatively, you can use click method on the element which does form submission

51. How to press ENTER key on text box In Selenium WebDriver?

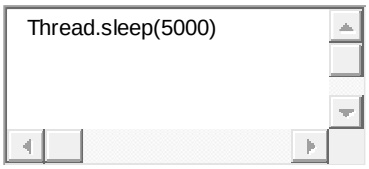
To press ENTER key using Selenium WebDriver, We need to use Selenium Enum Keys with its constant ENTER.



1 driver.findElement(By.xpath("xpath")).sendKeys(Keys.ENTER);

52. How to pause a test execution for 5 seconds at a specific point?

By using **java.lang.Thread.sleep***(long milliseconds)* method we could pause the execution for a specific time. To pause 5 seconds, we need to pass parameter as 5000 (5 seconds)



1 Thread.sleep(5000)

53. Is Selenium Server needed to run Selenium WebDriver Scripts?

When we are distributing our Selenium WebDriver scripts to execute using Selenium Grid, we need to use Selenium Server.

54. What happens if I run this command. *driver.get("www.softwaretestingmaterial.com") ;*

An exception is thrown. We need to pass HTTP protocol within driver.get() method.

1 driver.get("https://www.softwaretestingmaterial.com");

55. What is the alternative to *driver.get()* method to open an URL using Selenium WebDriver?

Alternative method to *driver.get("url")* method is *driver.navigate.to("url")*

56. What is the difference between *driver.get()* and *driver.navigate.to("url")*?

driver.get(): To open an URL and it will wait till the whole page gets loaded
driver.navigate.to(): To navigate to an URL and It will not wait till the whole page gets loaded

57. Can I navigate back and forth in a browser in Selenium WebDriver?

We use Navigate interface to do navigate back and forth in a browser. It has methods to move back, forward as well as to refresh a page.

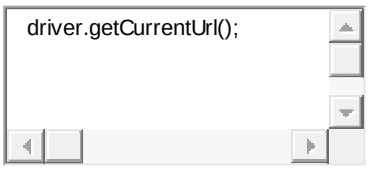
driver.navigate().forward(); – to navigate to the next web page with reference to the browser's history
driver.navigate().back(); – takes back to the previous webpage with reference to the browser's history
driver.navigate().refresh(); – to refresh the current web page thereby reloading all the web elements
driver.navigate().to("url"); – to launch a new web browser window and navigate to the specified URL

58. What are the different types of navigation commands?

Forward(), back(), to()

59. How to fetch the current page URL in Selenium?

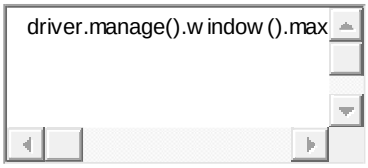
To fetch the current page URL, we use *getCurrentURL()*




```
1 driver.getCurrentUrl();
```

60. How can we maximize browser window in Selenium?

To maximize browser window in selenium we use *maximize()* method. This method maximizes the current window if it is not already maximized




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

61. How to delete cookies in Selenium?

To delete cookies we use *deleteAllCookies()* method


```
1 driver.manage().deleteAllCookies();
```

62. What are the ways to refresh a browser using Selenium WebDriver?

There are multiple ways to refresh a page in selenium

- Using *driver.navigate().refresh()* command as mentioned in the question 45
- Using driver.get("URL") on the current URL or using *driver.getCurrentUrl()*
- Using driver.navigate().to("URL") on the current URL or *driver.navigate().to(driver.getCurrentUrl());*
- Using *sendKeys(Keys.F5)* on any textbox on the webpage

63. What is the difference between driver.getWindowHandle() and driver.getWindowHandles() in Selenium WebDriver?

driver.getWindowHandle() – It returns a handle of the current page (a unique identifier)
driver.getWindowHandles() – It returns a set of handles of the all the pages available.

64. What is the difference between driver.close() and driver.quit() methods?

Purpose of these two methods (driver.close and driver.quit) is almost same. Both allow us to close a browser but still, there is a difference.

driver.close(): To close current WebDriver instance
driver.quit(): To close all the opened WebDriver instances

65. What is the difference between driver.findElement() and driver.findElements() commands?

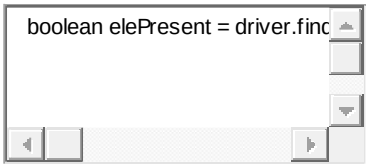
The difference between driver.findElement() and driver.findElements() commands is-

- findElement() returns a single WebElement (found first) based on the locator passed as parameter. Whereas findElements() returns a list of WebElements, all satisfying the locator value passed.
- Syntax of findElement()-
  WebElement textbox = driver.findElement(By.id("textBoxLocator"));
  Syntax of findElements()-
  List <WebElement> elements = element.findElements(By.id("value"));
- Another difference between the two is- if no element is found then findElement() throws NoSuchElementException whereas findElements() returns a list of 0 elements.

66. How to find whether an element is displayed on the web page?

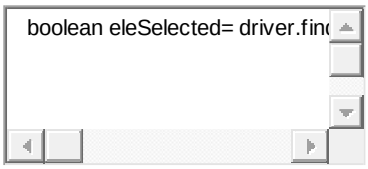
WebDriver facilitates the user with the following methods to check the visibility of the web elements. These web elements can be buttons, drop boxes, checkboxes, radio buttons, labels etc.

1. isDisplayed()



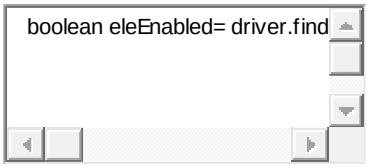
1 boolean elePresent = driver.findElement(By.xpath("xpath")).isDisplayed();

2. isSelected()



1 boolean eleSelected= driver.findElement(By.xpath("xpath")).isSelected();

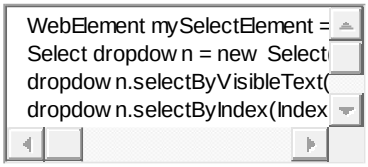
3. isEnabled()



1 boolean eleEnabled= driver.findElement(By.xpath("xpath")).isEnabled();

67. How to select a value in a dropdown?

By using *Select* class



1 WebElement mySelectElement = driver.findElement(By.name("dropdown"));
2 Select dropdown = new Select(mySelectElement);
3 dropdown.selectByVisibleText(Text);
4 dropdown.selectByIndex(Index);
5 dropdown.selectByValue(Value);

68. How to capture Screenshot in Selenium WebDriver?

Test cases may fail while executing the test scripts. While we are executing the test cases manually we just take a screenshot and place in a result repository. The same can be done by using Selenium WebDriver.

Some of the scenarios we may need to capture a screenshot using Selenium WebDriver are

i. Application issues
ii. Assertion Failure
iii. Difficulty to find Webelements on the web page
iv. Timeout to find Webelements on the web page

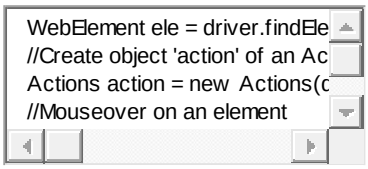
Selenium provides an interface called *TakesScreenshot* which has a method *getScreenShotAs* which can be used to take a screenshot of the application under test.

In Selenium 3, to handle issues while capturing Screenshots. we use aShot utility (additional jar file) .

```
@Test
public static void captureScreenMethod() throws Exception{
System.setProperty("webdriver.gecko.driver","D://Selenium
Environment//Drivers//geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.manage().window().maximize();
driver.get("http://www.girmiti.com");
File screenshotFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(screenshotFile, new File("D:\\SoftwareTesting.png"));
driver.close();
driver.quit();
}
```

69. How to mouse hover on a web element using WebDriver?

By using Actions class




```
1 WebElement ele = driver.findElement(By.xpath("xpath"));
2 //Create object 'action' of an Actions class
3 Actions action = new Actions(driver);
4 //Mouseover on an element
5 action.moveToElement(ele).perform();
```

70. How can we handle web based pop-up?

To handle alerts popups we need to do switch to the alert window and call Selenium WebDriver Alert API methods.

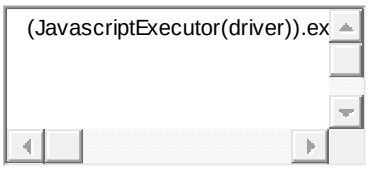
71. How can we handle windows based pop up?

Selenium doesn’t support windows based applications. It is an automation testing tool which supports only web application testing. We could handle windows based popups in Selenium using some third party tools such as AutoIT, Sikuli, Robot class etc.

72. How to handle hidden elements in Selenium WebDriver?

It is one of the most important selenium interview questions.

We can handle hidden elements by using javaScript executor

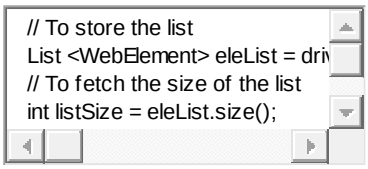



```
1 (JavascriptExecutor(driver)).executeScript("document.getElementsByClassName(ElementLoca
  tor).click();");
```

73. How can you find Broken Links in a page using Selenium WebDriver?

```
@Test

public void brokenLinks(){

System.setProperty("webdriver.chrome.driver", "C:\\resources\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("https://www.google.com/");

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

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

                URL u = new URL(allLinks.get(i).getAttribute("href"));

                        HttpURLConnection httpcon = (HttpURLConnection)
                        u.openConnection();

                        httpcon.setConnectTimeout(2000);

                        httpcon.connect();

// 200 is successful. 400 is broken link

                        System.out.println(httpcon.getResponseCode());

                }

}
```

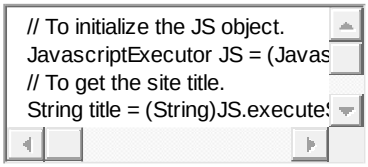
74. How to find more than one web element in the list?




```
1  // To store the list
2  List <WebElement> eleList = driver.findElements(By.xpath("xpath"));
3  // To fetch the size of the list
4  int listSize = eleList.size();
5  //for loop
6  for (int i=0; i<listSize; i++)
7  {
8  // Clicking on each link
9  links.get(i).click();
10 // Navigating back to the previous page that stores the links
11 driver.navigate().back();
12 }
```

75. How to read a JavaScript variable in Selenium WebDriver?

By using JavascriptExecutor

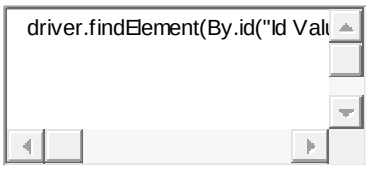



```
1 // To initialize the JS object.
2 JavascriptExecutor JS = (JavascriptExecutor) webdriver;
3 // To get the site title.
4 String title = (String)JS.executeScript("return document.title");
5 System.out.println("Title of the webpage : " + title);
```

76. What is JavaScriptExecutor and in which cases JavaScriptExecutor will help in Selenium automation?

In general, we click on an element using click() method in Selenium.

For example:



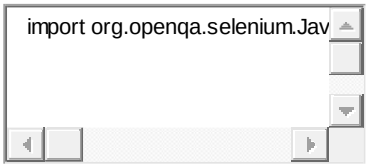
```
1 driver.findElement(By.id("Id Value")).click();
```

Sometimes web controls don't react well against selenium commands and we may face issues with the above statement (click()). To overcome such kind of situation, we use JavaScriptExecutor interface.

It provides a mechanism to execute Javascript through Selenium driver. It provides "executescript" & "executeAsyncScript" methods, to run JavaScript in the context of the currently selected frame or window.

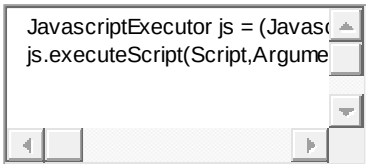
There is no need to write a separate script to execute JavaScript within the browser using Selenium WebDriver script. Just we use predefined interface named 'Java Script Executor'. We need to import the JavascriptExecutor package in the script.

Package:




```
1 import org.openqa.selenium.JavascriptExecutor;
```

Syntax:




```
1 JavascriptExecutor js = (JavascriptExecutor) driver;
2 js.executeScript(Script,Arguments);
```

Script *– The JavaScript to execute*
Arguments *– The arguments to the script(Optional). May be empty.*
Returns *– One of Boolean, Long, String, List, WebElement, or null.*

Let's see some scenarios we could handle using this Interface:

1. To type Text in Selenium WebDriver without using sendKeys() method
2. To click a Button in Selenium WebDriver using JavaScript
3. To handle Checkbox
4. To generate Alert Pop window in selenium
5. To refresh browser window using Javascript
6. To get innertext of the entire webpage in Selenium

7. To get the Title of our webpage
8. To get the domain
9. To get the URL of a webpage
10. To perform Scroll on an application using Selenium
11. To click on a SubMenu which is only visible on mouse hover on Menu
12. To navigate to different page using Javascript

77. How do you read test data from excels?

Test data can efficiently be read from excel using JXL or POI API. POI API has many advantages than JXL.

78. Is it possible to automate the captcha using Selenium?

No, It's not possible to automate captcha and bar code reader.

79. How to handle Ajax calls in Selenium WebDriver?

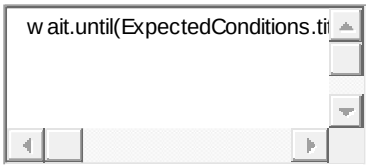
Handling AJAX calls is one of the common issues when using Selenium WebDriver. We wouldn't know when the AJAX call would get completed and the page has been updated. In this post, we see how to handle AJAX calls using Selenium.

AJAX stands for Asynchronous JavaScript and XML. AJAX allows the web page to retrieve small amounts of data from the server without reloading the entire page. AJAX sends HTTP requests from the client to server and then process the server's response without reloading the entire page. To handle AJAX controls, wait commands may not work. It's just because the actual page is not going to refresh.

When you click on a submit button, the required information may appear on the web page without refreshing the browser. Sometimes it may load in a second and sometimes it may take longer. We have no control over loading time. The best approach to handle this kind of situations in selenium is to use dynamic waits (i.e. WebDriverWait in combination with ExpectedCondition)

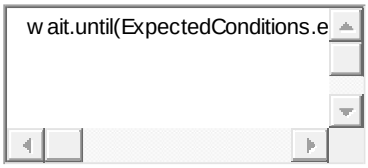
Some of the methods which are available are as follows:

1. titleIs() – The expected condition waits for a page with a specific title.



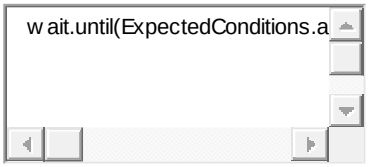
1 wait.until(ExpectedConditions.titleIs("Deal of the Day"));

2. elementToBeClickable() – The expected condition waits for an element to be clickable i.e. it should be present/displayed/visible on the screen as well as enabled.



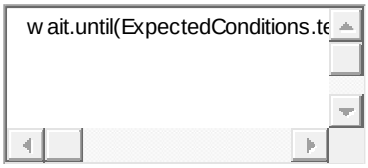
1 wait.until(ExpectedConditions.elementToBeClickable(By.xpath("xpath")));

3. alertIsPresent() – The expected condition waits for an alert box to appear.



1 wait.until(ExpectedConditions.alertIsPresent()) !=null);

4. textToBePresentInElement() – The expected condition waits for an element having a certain string pattern.



1 wait.until(ExpectedConditions.textToBePresentInElement(By.id(“title’”), “text to be found”));

80. List some scenarios which we cannot automate using Selenium WebDriver?

1. Bitmap / image comparison is not possible using Selenium WebDriver
2. Automating Captcha is not possible using Selenium WebDriver
3. We can not read bar code using Selenium WebDriver

81. What is Object Repository in Selenium WebDriver?

Object Repository is used to store element locator values in a centralized location instead of hard coding them within the scripts. We do create a property file (*.properties*) to store all the element locators and these property files act as an object repository in Selenium WebDriver.

82. How you build Object Repository in your project?

In Selenium, we call objects as locators (such as ID, Name, Class Name, Tag Name, Link Text, Partial Link Text, XPath, and CSS). Object repository is a collection of objects. One of the ways

to create Object Repository is to place all the locators in a separate file (i.e., properties file). But the best way is to use Page Object Model. In the Page Object Model Design Pattern, each web page is represented as a class. All the objects related to a particular page of a web application are stored in a class.

83. What is Page Object Model in Selenium?

Page Object Model (POM) is a Design Pattern which has become popular in Selenium Test Automation. It is widely used design pattern in Selenium for enhancing test maintenance and reducing code duplication. Page object model (POM) can be used in any kind of framework such as modular, data-driven,keyword driven, hybrid framework etc. A page object is an object-oriented class that serves as an interface to a page of your Application Under Test(AUT). The tests then use the methods of this page object class whenever they need to interact with the User Interface (UI) of that page. The benefit is that if the UI changes for the page, the tests themselves don't need to change, only the code within the page object needs to change. Subsequently, all changes to support that new UI is located in one place.

84. What is Page Factory?

We have seen that 'Page Object Model' is a way of representing an application in a test framework. For every 'page' in the application, we create a Page Object to reference the 'page' whereas a 'Page Factory' is one way of implementing the 'Page Object Model'.

85. What is the difference between Page Object Model (POM) and Page Factory?

Page Object is a class that represents a web page and hold the functionality and members.
Page Factory is a way to initialize the web elements you want to interact with within the page object when you create an instance of it.

86. What are the advantages of Page Object Model Framework?

Code reusability – We could achieve code reusability by writing the code once and use it in different tests.

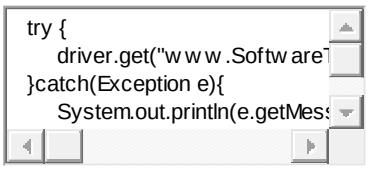
Code maintainability – There is a clean separation between test code and page specific code such as locators and layout which becomes very easy to maintain code. Code changes only on Page Object Classes when a UI change occurs. It enhances test maintenance and reduces code duplication.

Object Repository – Each page will be defined as a java class. All the fields in the page will be defined in an interface as members. The class will then implement the interface.

Readability – Improves readability due to clean separation between test code and page specific code

87. How can you use the Recovery Scenario in Selenium WebDriver?

By using "*Try Catch Block"* within Selenium WebDriver Java tests.




```
1 try {
2      driver.get("www.SoftwareTestingMaterial.com");
3 }catch(Exception e){
4      System.out.println(e.getMessage());
5 }
```

88. How to Upload a file in Selenium WebDriver?

There are two cases which are majorly used to upload a file in Selenium WebDriver such as using *SendKeys* Method and using *AutoIT* Script.

Robot Class, Sikuli are also options

89. How to Download a file in Selenium WebDriver?

By using *AutoIT* script, we could download a file in Selenium WebDriver.

90. How to run Selenium WebDriver Test from the command line?

There are two ways

1. Run testng.xml file from command line with the following commands

Set classpath for project

- set projectLocation=C:\Users\Admin\Desktop\STMSeleniumTutorial\workspace\SoftwareTestingMaterial
- cd %projectLocation%
- set classpath=%projectLocation%\bin;%projectLocation%\lib*

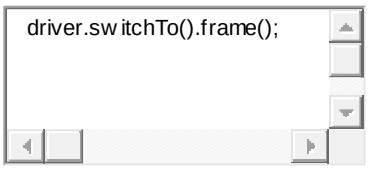
Run testng.xml

- java org.testng.TestNG %projectLocation%\testng.xml

2. Run testng.xml from pom.xml using failsafe or surefire plugin, if it is maven project

91. How to switch between frames in Selenium?

By using the following code, we could switch between frames using frame id or frame name



driver.switchTo().frame(int); - Using Frame id
driver.switchto().frame(string); - Using Frame Name

92. How to connect a Database in selenium?

As we all know Selenium WebDriver is a tool to automate User Interface. We could only interact with Browser using Selenium WebDriver.

We use JDBC Driver to connect the Database in Selenium (While using Java Programming Language).

93. How To Resize Browser Window Using Selenium WebDriver?

To resize the browser window to particular dimensions, we use 'Dimension' class to resize the browser window.

```
Dimension d = new Dimension(500,620);

driver.manage().window().setSize(d);
```

94. How To Scroll Web Page Down Or UP Using Selenium WebDriver?

JavaScript **scrollBy**() method scrolls the document by the specified number of pixels.

```
JavascriptExecutor js = (JavascriptExecutor) driver;

 js.executeScript("window.scrollBy(0,250)", "");
```

95. How To Perform Right Click Action (Context Click) In Selenium WebDriver?

We use Actions class in Selenium WebDriver to do Right-Click (Context Click) action.

```
Actions action = new Actions(driver);

action.contextClick(WebElement).build().perform();
```

96. How To Perform Double Click Action In Selenium WebDriver?

We use Actions class to do Double click action in selenium.

action.doubleClick(webelement).build().perform();

97. How To Perform Drag And Drop Action in Selenium WebDriver?

To achieve this, we use Actions class in Selenium WebDriver.

action.dragAndDrop(sourceLocator, targetLocator).build().perform();

98. How To Highlight Element Using Selenium WebDriver?

By using JavascriptExecutor interface, we could highlight the specified element

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("arguments[0].setAttribute('style', 'background: yellow; border: 2px solid red;');", ele);

99. Have you used any crossbrowsertesting tool to run selenium scripts on cloud?

If you want to say YES.

You have the following options

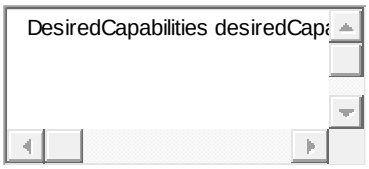
Browserstack, Sauce labs etc

100. What is desired capabilities?

In Selenium we use desired capabilities to define the driver capability

handle SSL certificates in chrome browser

We need to create an instance of DesiredCapabilities



DesiredCapabilities desiredCapability = DesiredCapabilities.chrome();

101. What is Continuous Integration?

Continuous Integration is abbreviated as CI. Continuous Integration is a development practice which aims to make sure the correctness of a software. After each commit, a suite of tests run automatically and test the software to ensure whether the software is running without any breaks. If any test fails, we will get immediate feedback say “build is broken”.

In simple words, continuous integration is a process of verifying the correctness of a software.

Some of the continuous integration tools are Jenkins, TeamCity, Bamboo, Travis, Circle Ci, Bitbucket.

We can schedule the test suite execution using these CI Tools.

102. How to achieve Database testing in Selenium?

As we all know Selenium WebDriver is a tool to automate User Interface. We could only interact with Browser using Selenium WebDriver.

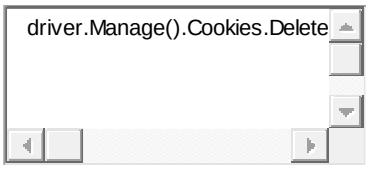
Sometimes, we may face a situation to get the data from the Database or to modify (update/delete) the data from the Database. If we plan to automate anything outside the vicinity of a browser, then we need to use other tools to achieve our task. To achieve the Database connection and work on it, we need to use JDBC API Driver.

The Java Database Connectivity (JDBC) API provides universal data access from the Java programming language. Using the JDBC API, you can access virtually any data source, from relational databases to spreadsheets and flat files. It lets the user connect and interact with the Database and fetch the data based on the queries we use in the automation script. JDBC is a SQL level API that allows us to execute SQL statements. It creates a connectivity between Java Programming Language and the database.

Using JDBC Driver we could do the following

i. Establish a Database connection
ii. Send SQL Queries to the Database
iii. Process the results

103. How to delete Browser Cookies with Selenium Web Driver?



1 driver.Manage().Cookies.DeleteAllCookies();

TestNG – Interview Questions:

104. What is TestNG?

TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing to integration testing. TestNG will give flexibility to the user to develop and execute automated tests.

What are the annotations available in TestNG?

@BeforeTest
@AfterTest
@BeforeClass
@AfterClass
@BeforeMethod
@AfterMethod
@BeforeSuite
@AfterSuite
@BeforeGroups
@AfterGroups
@Test

106. What is TestNG Assert and list out some common Assertions supported by TestNG?

TestNG Asserts help us to verify the condition of the test in the middle of the test run. Based on the TestNG Assertions, we will consider a successful test only if it is completed the test run without throwing any exception.

Some of the common assertions supported by TestNG are

- assertEqual(String actual,String expected)
- assertEqual(String actual,String expected, String message)
- assertEquals(boolean actual,boolean expected)
- assertTrue(condition)
- assertTrue(condition, message)
- assertFalse(condition)
- assertFalse(condition, message)

107. How to create and run TestNG.xml?

In TestNG framework, we need to create T**estNG XML** file to create and handle multiple test classes. We do configure our test run, set test dependency, include or exclude any test, method, class or package and set priority etc in the XML file.

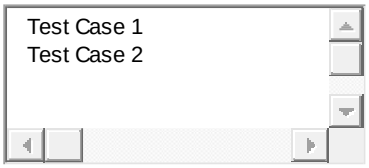
108. How to set test case priority in TestNG?

We use *priority* attribute to the *@Test* annotations. In case priority is not set then the test scripts execute in alphabetical order.

```
package TestNG;
import org.testng.annotations.*;
public class PriorityTestCase{
        @Test(priority=0)
```

```
1  package TestNG;
2  import org.testng.annotations.*;
3  public class PriorityTestCase{
4  @Test(priority=0)
5  public void testCase1() {
6  system.out.println("Test Case 1");
7  }
8  @Test(priority=1)
9  public void testCase2() {
10 system.out.println("Test Case 2");
11 }
12 }
```

Output:



1 Test Case 1
2 Test Case 2

109. What is Parameterized testing in TestNG?

Parameterized tests allow developers to run the same test over and over again using different values.

There are two ways to set these parameters:

- *with testng.xml*

```
@Test
@Parameters("browser")
 public void parameterizedTest(String browser){
 if(browser.equals("firefox")){
 System.out.println("Open Firefox Driver");
 }else if(browser.equals("chrome")){
 System.out.println("Open Chrome Driver");
 }
 }
```

- *with Data Providers*

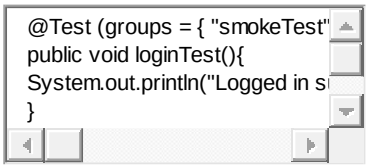
```
@Test (dataProvider="getData")
```

```
    // Number of columns should match the number of input parameters
 public void loginTest(String Uid, String Pwd){
 System.out.println("UserName is "+ Uid);
 System.out.println("Password is "+ Pwd);
 }

 //If the name is not supplied, the data provider's name automatically defaults to the method's name.
 //A data provider returns an array of objects.
 @DataProvider(name="getData")
 public Object[][] getData(){
 //Object [][] data = new Object [rowCount][colCount];
 Object [][] data = new Object [2][2];

 data [0][0] = "FirstUid";
data [0][1] = "FirstPWD";

 data[1][0] = "SecondUid";
 data[1][1] = "SecondPWD";

return data;

 }
```

110. How to run a group of test cases using TestNG?

TestNG allows you to perform sophisticated groupings of test methods. Not only can you declare that methods belong to groups, but you can also specify groups that contain other groups. Then TestNG can be invoked and asked to include a certain set of groups (or regular expressions) while excluding another set. This gives you maximum flexibility in how you partition your tests and doesn't require you to recompile anything if you want to run two different sets of tests back to back.

Groups are specified in your testng.xml file and can be found either under the <test> or <suite> tag. Groups specified in the <suite> tag apply to all the <test> tags underneath.



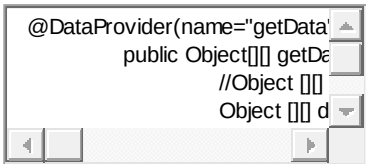

```
1 @Test (groups = { "smokeTest", "functionalTest" })
2 public void loginTest(){
3 System.out.println("Logged in successfully");
4 }
```

111. What is the use of @Listener annotation in TestNG?

Ans. TestNG listeners are used to configure reports and logging. One of the most widely used listeners in TestNG is *ITestListener* interface. It has methods like *onTestStart, onTestSuccess, onTestFailure, onTestSkipped* etc. We should implement this interface creating a listener class of our own. Next, we should add the listeners annotation (*@Listeners*) in the Class which was created.

112. How can we create a data-driven framework using TestNG?

By using *@DataProvider* annotation, we can implement data driven framework

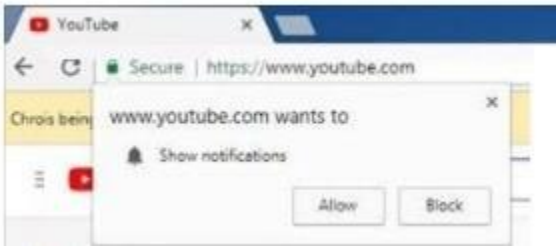



```
@DataProvider(name="getData")
public Object[][] getData(){
//Object [][] data = new Object [rowCount][colCount];
Object [][] data = new Object [2][2];
data [0][0] = "FirstUid";
data [0][1] = "FirstPWD";
data[1][0] = "SecondUid";
```

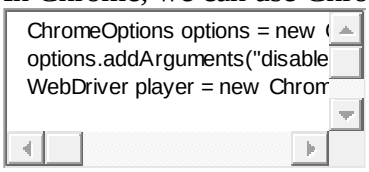
```
data[1][1] = "SecondPWD";
return data;
}
```

113. Where you have applied OOPS in Automation Framework?
Question can be answered based on the framework.
Class
Object
Abstraction – Implementing POM in different class
Interface – Webdriver driver = new ChromeDriver();
Inheritance
Polymorphism - Method Overriding, Method Overloading
Encapsulation – Implementing method in POM

114. How to handle browser (chrome) notifications in Selenium?



In Chrome, we can use ChromeOptions as shown below.



1 ChromeOptions options = new ChromeOptions();
2 options.addArguments("disable-infobars");
3 WebDriver player = new ChromeDriver(options);

Core Java Interview Questions

1) What is Java?

Java is the high-level, object-oriented, robust, secure programming language, platform-independent, high performance, Multithreaded, and portable programming language. It was developed by **James Gosling** in June 1991. It can also be known as the platform as it provides its own JRE and API.

---

2) What are the differences between C++ and Java?

The differences between C++ and Java are given in the following table.

Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Design Goal	C++ was designed for systems and applications programming. It was an extension of C	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces.
Operator Overloading	C++ supports operator overloading	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.
Compiler and Interpreter	C++ uses compiler only. C++ is compiled and run using the	Java uses compiler and interpreter both. Java source code is converted into bytecode

	compiler which converts source code into machine code so, C++ is platform dependent.	at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.
Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.
Documentation comment	C++ doesn't support documentation comment.	Java supports documentation comment (/** ... */) to create documentation for java source code.
Virtual Keyword	C++ supports virtual keyword so that we can decide whether or not override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
unsigned right shift >>>	C++ doesn't support >>> operator.	Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator.
Inheritance Tree	C++ creates a new inheritance tree always.	Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the inheritance tree in java.
Hardware	C++ is nearer to hardware.	Java is not so interactive with hardware.
Object-oriented	C++ is an object-oriented language. However, in C language, single root hierarchy is not possible.	Java is also an object-oriented language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object.

---

3) List the features of Java Programming language.

There are the following features in Java Programming Language.

- **Simple:** Java is easy to learn. The syntax of Java is based on C++ which makes easier to write the program in it.
- **Object-Oriented:** Java follows the object-oriented paradigm which allows us to maintain our code as the combination of different type of objects that incorporates both data and behavior.
- **Portable:** Java supports read-once-write-anywhere approach. We can execute the Java program on every machine. Java program (.java) is converted to bytecode (.class) which can be easily run on every machine.
- **Platform Independent:** Java is a platform independent programming language. It is different from other programming languages like C and C++ which needs a platform to be executed. Java comes with its platform on which its code is executed. Java doesn't depend upon the operating system to be executed.
- **Secured:** Java is secured because it doesn't use explicit pointers. Java also provides the concept of ByteCode and Exception handling which makes it more secured.
- **Robust:** Java is a strong programming language as it uses strong memory management. The concepts like Automatic garbage collection, Exception handling, etc. make it more robust.
- **Architecture Neutral:** Java is architectural neutral as it is not dependent on the architecture. In C, the size of data types may vary according to the architecture (32 bit or 64 bit) which doesn't exist in Java.
- **Interpreted:** Java uses the Just-in-time (JIT) interpreter along with the compiler for the program execution.
- **High Performance:** Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++).
- **Multithreaded:** We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.
- **Distributed:** Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.
- **Dynamic:** Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

---

4) What do you understand by Java virtual machine?

Java Virtual Machine is a virtual machine that enables the computer to run the Java program. JVM acts like a run-time engine which calls the main method present in the Java code. JVM is

the specification which must be implemented in the computer system. The Java code is compiled by JVM to be a Bytecode which is machine independent and close to the native code.

---

5) What is the difference between JDK, JRE, and JVM?

JVM

JVM is an acronym for Java Virtual Machine; it is an abstract machine which provides the runtime environment in which Java bytecode can be executed. It is a specification which specifies the working of Java Virtual Machine. Its implementation has been provided by Oracle and other companies. Its implementation is known as JRE.

JVMs are available for many hardware and software platforms (so JVM is platform dependent). It is a runtime instance which is created when we run the Java class. There are three notions of the JVM: specification, implementation, and instance.

JRE

JRE stands for Java Runtime Environment. It is the implementation of JVM. The 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.

JDK

JDK is an acronym for Java Development Kit. It is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools. JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

---

6) How many types of memory areas are allocated by JVM?

Many types:

1. **Class(Method) Area:** Class Area stores per-class structures such as the runtime constant pool, field, method data, and the code for methods.
2. **Heap:** It is the runtime data area in which the memory is allocated to the objects
3. **Stack:** Java Stack stores frames. It holds local variables and partial results, and plays a

part in method invocation and return. Each thread has a private JVM stack, created at the same time as the thread. A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

4. **Program Counter Register:** PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.
5. **Native Method Stack:** It contains all the native methods used in the application.

---

7) What is JIT compiler?

Just-In-Time(JIT) compiler: It is used to improve the performance. JIT compiles parts of the bytecode that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

---

8) What is the platform?

A platform is the hardware or software environment in which a piece of software is executed. There are two types of platforms, software-based and hardware-based. Java provides the software-based platform.

---

9) What are the main differences between the Java platform and other platforms?

There are the following differences between the Java platform and other platforms.

- Java is the software-based platform whereas other platforms may be the hardware platforms or software-based platforms.
- Java is executed on the top of other hardware platforms whereas other platforms can only have the hardware components.

---

10) What gives Java its 'write once and run anywhere' nature?

The bytecode. Java compiler converts the Java programs into the class file (Byte Code) which is the intermediate language between source code and machine code. This bytecode is not platform specific and can be executed on any computer.

---

11) What is classloader?

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

1. **Bootstrap ClassLoader**: This is the first classloader which is the superclass of Extension classloader. It loads the *rt.jar* file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes, etc.
2. **Extension ClassLoader**: This is the child classloader of Bootstrap and parent classloader of System classloader. It loads the jar files located inside *$JAVA_HOME/jre/lib/ext* directory.
3. **System/Application ClassLoader**: This is the child classloader of Extension classloader. It loads the class files from the classpath. By default, the classpath is set to the current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.

---

12) Is Empty .java file name a valid source file name?

Yes, Java allows to save our java file by **.java** only, we need to compile it by **javac .java** and run by **java classname** Let's take a simple example:

```
1. //save by .java only
2. class A{
3. public static void main(String args[]){
4. System.out.println("Hello java");
5. }
6. }
7. //compile by javac .java
8. //run by     java A
```

compile it by **javac .java**

run it by **java A**

---

13) Is delete, next, main, exit or null keyword in java?

No.

---

14) If I don't provide any arguments on the command line, then what will the value stored in the String array passed into the main() method, empty or NULL?

It is empty, but not null.

---

15) What if I write static public void instead of public static void?

The program compiles and runs correctly because the order of specifiers doesn't matter in Java.

---

16) What is the default value of the local variables?

The local variables are not initialized to any default value, neither primitives nor object references.

---

17) What are the various access specifiers in Java?

In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable. In Java, there are four access specifiers given below.

- **Public** The classes, methods, or variables which are defined as public, can be accessed by any class or method.
- **Protected** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.
- **Default** Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.
- **Private** The private class, methods, or variables defined as private can be accessed within the class only.

---

18) What is the purpose of static methods and variables?

The methods or variables defined as static are shared among all the objects of the class. The static is the part of the class and not of the object. The static variables are stored in the class area, and we do not need to create the object to access such variables. Therefore, static is used in the case, where we need to define variables or methods which are common to all the objects of the class.

For example, In the class simulating the collection of the students in a college, the name of the college is the common attribute to all the students. Therefore, the college name will be defined as **static**.

---

19) What are the advantages of Packages in Java?

There are various advantages of defining packages in Java.

- Packages avoid the name clashes.
- The Package provides easier access control.
- We can also have the hidden classes that are not visible outside and used by the package.
- It is easier to locate the related classes.

---

20) What is the output of the following Java program?

```
1. class Test
2. {
3.     public static void main (String args[])
4.     {
5.         System.out.println(10 + 20 + "Javatpoint");
6.         System.out.println("Javatpoint" + 10 + 20);
7.     }
8. }
```

The output of the above code will be

30Javatpoint
Javatpoint1020

Explanation

In the first case, 10 and 20 are treated as numbers and added to be 30. Now, their sum 30 is treated as the string and concatenated with the string **Javatpoint**. Therefore, the output will be **30Javatpoint**.

In the second case, the string Javatpoint is concatenated with 10 to be the string **Javatpoint10** which will then be concatenated with 20 to be **Javatpoint1020**.

---

21) What is the output of the following Java program?

```
1. class Test
2. {
3.     public static void main (String args[])
4.     {
5.         System.out.println(10 * 20 + "Javatpoint");
6.         System.out.println("Javatpoint" + 10 * 20);
```

```
7.      }
8.  }
```

The output of the above code will be

200Javatpoint
Javatpoint200

Explanation

In the first case, The numbers 10 and 20 will be multiplied first and then the result 200 is treated as the string and concatenated with the string **Javatpoint** to produce the output **200Javatpoint**.

In the second case, The numbers 10 and 20 will be multiplied first to be 200 because the precedence of the multiplication is higher than addition. The result 200 will be treated as the string and concatenated with the string **Javatpoint**to produce the output as **Javatpoint200**.

---

22) What is the output of the following Java program?

```
1.  class Test
2.  {
3.      public static void main (String args[])
4.      {
5.          for(int i=0; 0; i++)
6.          {
7.              System.out.println("Hello Javatpoint");
8.          }
9.      }
10. }
```

The above code will give the compile-time error because the for loop demands a boolean value in the second part and we are providing an integer value, i.e., 0.

Core Java - OOPs Concepts: Initial OOPs Interview Questions

---

23) What is object-oriented paradigm?

It is a programming paradigm based on objects having data and methods defined in the class to which it belongs. Object-oriented paradigm aims to incorporate the advantages of modularity and reusability. Objects are the instances of classes which interacts with one another to design applications and programs. There are the following features of the object-oriented paradigm.

- Follows the bottom-up approach in program design.
- Focus on data with methods to operate upon the object?s data
- Includes the concept like Encapsulation and abstraction which hides the complexities from the user and show only functionality.
- Implements the real-time approach like inheritance, abstraction, etc.
- The examples of the object-oriented paradigm are C++, Simula, Smalltalk, Python, C#, etc.

---

24) What is an object?

The Object is the real-time entity having some state and behavior. In Java, Object is an instance of the class having the instance variables as the state of the object and the methods as the behavior of the object. The object of a class can be created by using the **new** keyword.

---

25) What is the difference between an object-oriented programming language and object-based programming language?

There are the following basic differences between the object-oriented language and object-based language.

- Object-oriented languages follow all the concepts of OOPs whereas, the object-based language doesn't follow all the concepts of OOPs like inheritance and polymorphism.
- Object-oriented languages do not have the inbuilt objects whereas Object-based languages have the inbuilt objects, for example, JavaScript has window object.
- Examples of object-oriented programming are Java, C#, Smalltalk, etc. whereas the examples of object-based languages are JavaScript, VBScript, etc.

---

26) What will be the initial value of an object reference which is defined as an instance variable?

All object references are initialized to null in Java.

Core Java - OOPs Concepts: Constructor Interview Questions

---

27) What is the constructor?

The constructor can be defined as the special type of method that is used to initialize the state of

an object. It is invoked when the class is instantiated, and the memory is allocated for the object. Every time, an object is created using the **new** keyword, the default constructor of the class is called. The name of the constructor must be similar to the class name. The constructor must not have an explicit return type.

---

28) How many types of constructors are used in Java?

Based on the parameters passed in the constructors, there are two types of constructors in Java.

- **Default Constructor:** default constructor is the one which does not accept any value. The default constructor is mainly used to initialize the instance variable with the default values. It can also be used for performing some useful task on object creation. A default constructor is invoked implicitly by the compiler if there is no constructor defined in the class.
- **Parameterized Constructor:** The parameterized constructor is the one which can initialize the instance variables with the given values. In other words, we can say that the constructors which can accept the arguments are called parameterized constructors.

---

29) What is the purpose of a default constructor?

The purpose of the default constructor is to assign the default value to the objects. The java compiler creates a default constructor implicitly if there is no constructor in the class.

```
1. class Student3{
2. int id;
3. String name;
4.
5. void display(){System.out.println(id+" "+name);}
6.
7. public static void main(String args[]){
8. Student3 s1=new Student3();
9. Student3 s2=new Student3();
10. s1.display();
11. s2.display();
12. }
13. }
```

Output:

```
0 null
0 null
```

Explanation: In the above class, you are not creating any constructor, so compiler provides you a default constructor. Here 0 and null values are provided by default constructor.

---

30) Does constructor return any value?

Ans: yes, The constructor implicitly returns the current instance of the class (You can't use an explicit return type with the constructor).

---

31)Is constructor inherited?

No, The constructor is not inherited.

---

32) Can you make a constructor final?

No, the constructor can't be final.

---

33) Can we overload the constructors?

Yes, the constructors can be overloaded by changing the number of arguments accepted by the constructor or by changing the data type of the parameters. Consider the following example.

```
1. class Test
2. {
```

```
3.      int i;
4.      public Test(int k)
5.      {
6.          i=k;
7.      }
8.      public Test(int k, int m)
9.      {
10.         System.out.println("Hi I am assigning the value max(k, m) to i");
11.         if(k>m)
12.         {
13.             i=k;
14.         }
15.         else
16.         {
17.             i=m;
18.         }
19.     }
20. }
21. public class Main
22. {
23.     public static void main (String args[])
24.     {
25.         Test test1 = new Test(10);
26.         Test test2 = new Test(12, 15);
27.         System.out.println(test1.i);
28.         System.out.println(test2.i);
29.     }
30. }
31.
```

In the above program, The constructor Test is overloaded with another constructor. In the first call to the constructor, The constructor with one argument is called, and i will be initialized with the value 10. However, In the second call to the constructor, The constructor with the 2 arguments is called, and i will be initialized with the value 15.

---

34) What do you understand by copy constructor in Java?

There is no copy constructor in java. However, we can copy the values from one object to another like copy constructor in C++.

There are many ways to copy the values of one object into another in java. They are:

- By constructor
- By assigning the values of one object into another

- By clone() method of Object class

In this example, we are going to copy the values of one object into another using java constructor.

```
1. //Java program to initialize the values from one object to another
2. class Student6{
3.     int id;
4.     String name;
5.     //constructor to initialize integer and string
6.     Student6(int i,String n){
7.     id = i;
8.     name = n;
9.     }
10.    //constructor to initialize another object
11.    Student6(Student6 s){
12.    id = s.id;
13.    name =s.name;
14.    }
15.    void display(){System.out.println(id+" "+name);}
16.
17.    public static void main(String args[]){
18.    Student6 s1 = new Student6(111,"Karan");
19.    Student6 s2 = new Student6(s1);
20.    s1.display();
21.    s2.display();
22.   }
23. }
```

Output:

```
111 Karan
111 Karan
```

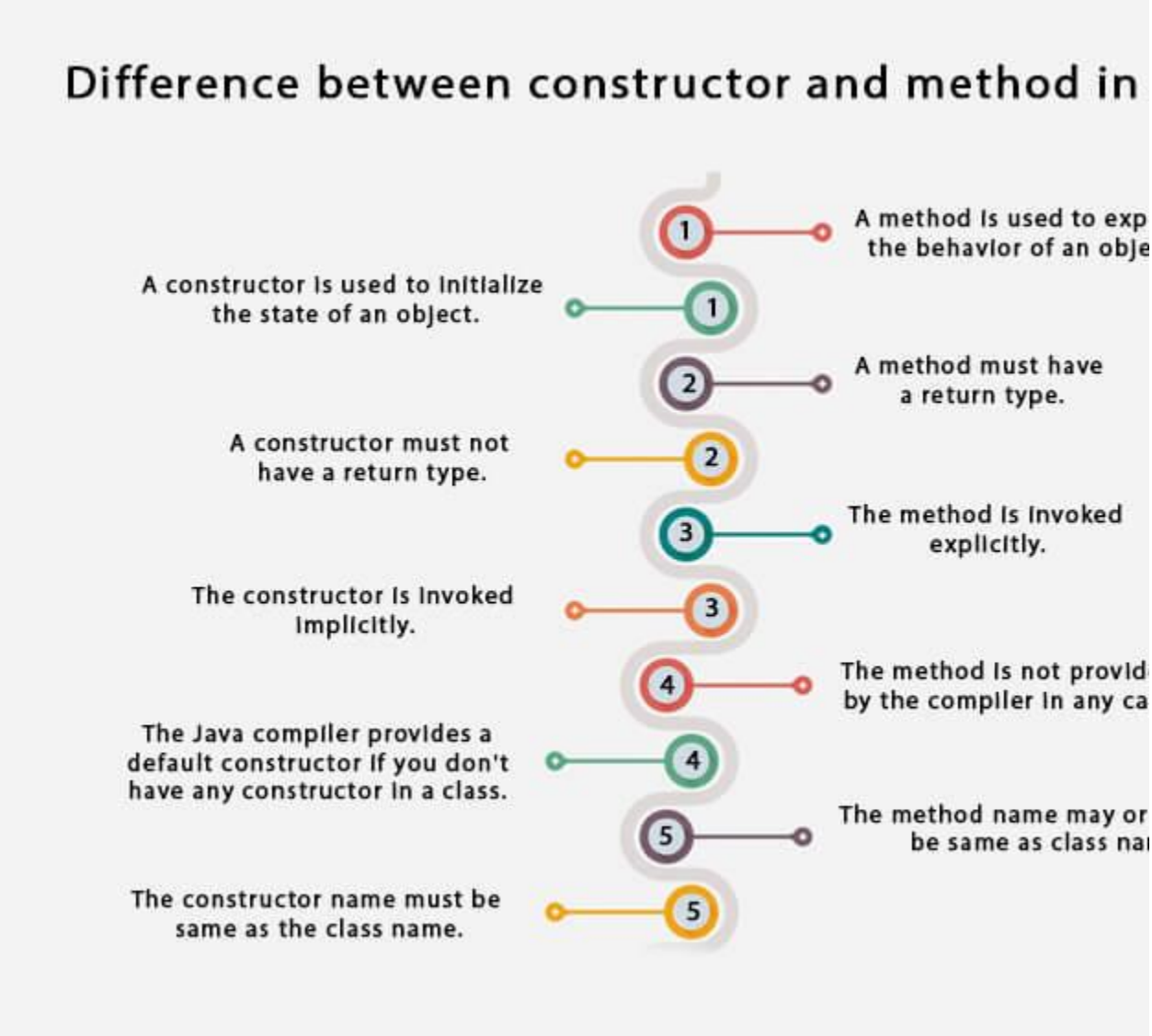
---

35) What are the differences between the constructors and methods?

There are many differences between constructors and methods. They are given below.

Java Constructor	Java Method
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.

The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as class name.



---

36) What is the output of the following Java program?

```
1. public class Test
2. {
```

```
3.      Test(int a, int b)
4.      {
5.          System.out.println("a = "+a+" b = "+b);
6.      }
7.      Test(int a, float b)
8.      {
9.          System.out.println("a = "+a+" b = "+b);
10.     }
11.     public static void main (String args[])
12.     {
13.         byte a = 10;
14.         byte b = 15;
15.         Test test = new Test(a,b);
16.     }
17. }
```

The output of the following program is:

a = 10 b = 15

Here, the data type of the variables a and b, i.e., byte gets promoted to int, and the first parameterized constructor with the two integer parameters is called.

---

37) What is the output of the following Java program?

```
1.  class Test
2.  {
3.      int i;
4.  }
5.  public class Main
6.  {
7.      public static void main (String args[])
8.      {
9.          Test test = new Test();
10.         System.out.println(test.i);
11.     }
12. }
```

The output of the program is 0 because the variable i is initialized to 0 internally. As we know that a default constructor is invoked implicitly if there is no constructor in the class, the variable i is initialized to 0 since there is no constructor in the class.

---

38) What is the output of the following Java program?

```
1.  class Test
2.  {
3.      int test_a, test_b;
4.      Test(int a, int b)
5.      {
6.      test_a = a;
7.      test_b = b;
8.      }
9.      public static void main (String args[])
10.     {
11.         Test test = new Test();
12.         System.out.println(test.test_a+" "+test.test_b);
13.     }
14. }
```

There is a **compiler error** in the program because there is a call to the default constructor in the main method which is not present in the class. However, there is only one parameterized constructor in the class Test. Therefore, no default constructor is invoked by the constructor implicitly.

Core Java - OOPs Concepts: static keyword Interview Questions

---

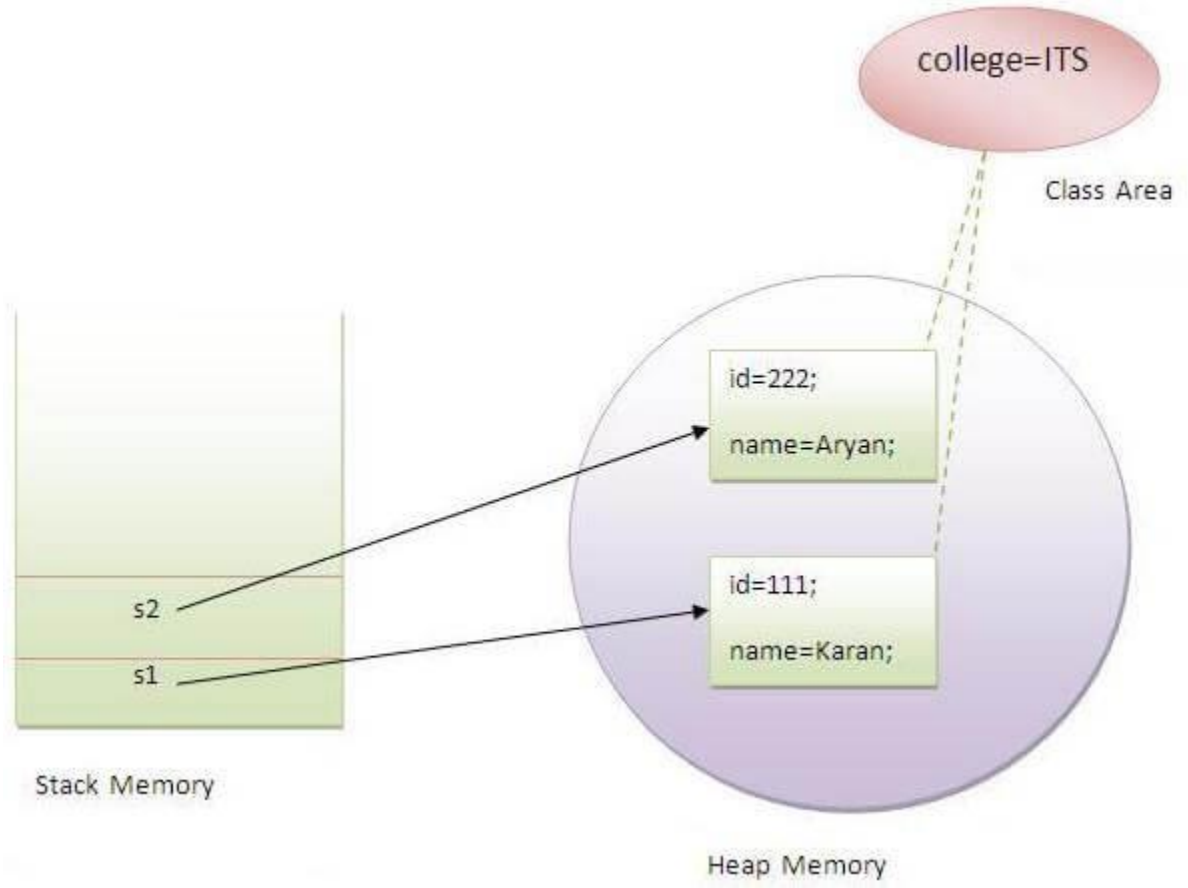
39) What is the static variable?

The static variable is used to refer to the common property of all objects (that is not unique for each object), e.g., The company name of employees, college name of students, etc. Static variable gets memory only once in the class area at the time of class loading. Using a static variable makes your program more memory efficient (it saves memory). Static variable belongs to the class rather than the object.

```
1.  //Program of static variable
2.
3.  class Student8{
4.     int rollno;
5.     String name;
6.     static String college ="ITS";
7.
8.     Student8(int r,String n){
9.     rollno = r;
10.    name = n;
11.    }
```

```
12.  void display (){System.out.println(rollno+" "+name+" "+college);}
13.
14.  public static void main(String args[]){
15.  Student8 s1 = new Student8(111,"Karan");
16.  Student8 s2 = new Student8(222,"Aryan");
17.
18.  s1.display();
19.  s2.display();
20.  }
21. }
```

Output:111 Karan ITS
222 Aryan ITS



More Details.

---

40) What is the static method?

- A static method belongs to the class rather than the object.
- There is no need to create the object to call the static methods.
- A static method can access and change the value of the static variable.

---

41) What are the restrictions that are applied to the Java static methods?

Two main restrictions are applied to the static methods.

- The static method can not use non-static data member or call the non-static method directly.
- this and super cannot be used in static context as they are non-static.

---

42) Why is the main method static?

Because the object is not required to call the static method. If we make the main method non-static, JVM will have to create its object first and then call main() method which will lead to the extra memory allocation.

---

43) Can we override the static methods?

No, we can't override static methods.

44) What is the static block?

Static block is used to initialize the static data member. It is executed before the main method, at the time of classloading.

```
1. class A2{
2.   static{System.out.println("static block is invoked");}
3.   public static void main(String args[]){
4.    System.out.println("Hello main");
5.   }
6. }
```

Output: static block is invoked
Hello main

---

45) Can we execute a program without main() method?

Ans) Yes, one of the ways to execute the program without the main method is using static block.

---

46) What if the static modifier is removed from the signature of the main method?

Program compiles. However, at runtime, It throws an error "NoSuchMethodError."

---

47) What is the difference between static (class) method and instance method?

static or class method	instance method
1)A method that is declared as static is known as the static method.	A method that is not declared as static is known as the instance method.
2)We don't need to create the objects to call the static methods.	The object is required to call the instance methods.
3)Non-static (instance) members cannot be accessed in the static context (static method, static block, and static nested class) directly.	Static and non-static variables both can be accessed in instance methods.
4)For example: public static int cube(int n){ return n*n*n;}	For example: public void msg(){...}.

---

48) Can we make constructors static?

As we know that the static context (method, block, or variable) belongs to the class, not the object. Since Constructors are invoked only when the object is created, there is no sense to make the constructors static. However, if you try to do so, the compiler will show the compiler error.

---

49) Can we make the abstract methods static in Java?

In Java, if we make the abstract methods static, It will become the part of the class, and we can directly call it which is unnecessary. Calling an undefined method is completely useless therefore it is not allowed.

---

50) Can we declare the static variables and methods in an abstract class?

Yes, we can declare static variables and methods in an abstract method. As we know that there is no requirement to make the object to access the static context, therefore, we can access the static

context declared inside the abstract class by using the name of the abstract class. Consider the following example.

```
1.  abstract class Test
2.  {
3.      static int i = 102;
4.      static void TestMethod()
5.      {
6.          System.out.println("hi !! I am good !!");
7.      }
8.  }
9.  public class TestClass extends Test
10. {
11.     public static void main (String args[])
12.     {
13.         Test.TestMethod();
14.         System.out.println("i = "+Test.i);
15.     }
16. }
```

Output

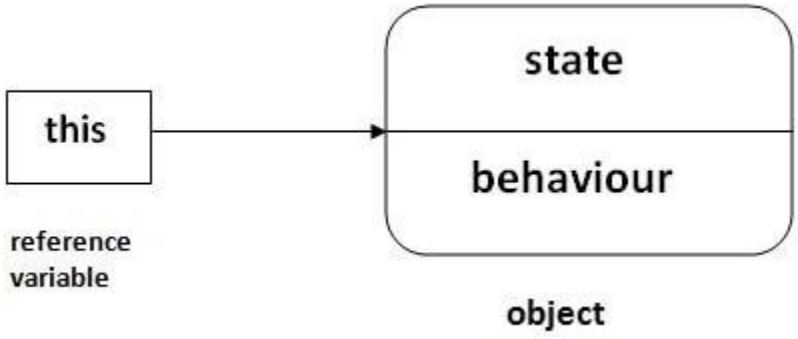
```
hi !! I am good !!
i = 102
```

Core Java - OOPs Concepts: Inheritance Interview Questions

---

51) What is this keyword in java?

The **this** keyword is a reference variable that refers to the current object. There are the various uses of this keyword in Java. It can be used to refer to current class properties such as instance methods, variable, constructors, etc. It can also be passed as an argument into the methods or constructors. It can also be returned from the method as the current class instance.



---

52) What are the main uses of this keyword?

There are the following uses of **this** keyword.

- **this** can be used to refer to the current class instance variable.
- **this** can be used to invoke current class method (implicitly)
- **this()** can be used to invoke the current class constructor.
- **this** can be passed as an argument in the method call.
-
- **this** can be passed as an argument in the constructor call.
- **this** can be used to return the current class instance from the method.

---

53) Can we assign the reference to this variable?

No, this cannot be assigned to any value because it always points to the current class object and this is the final reference in Java. However, if we try to do so, the compiler error will be shown. Consider the following example.

```
1. public class Test
2. {
3.     public Test()
4.     {
5.         this = null;
6.         System.out.println("Test class constructor called");
7.     }
8.     public static void main (String args[])
9.     {
10.        Test t = new Test();
11.    }
12. }
```

Output

```
Test.java:5: error: cannot assign a value to final variable this
      this = null;
      ^
1 error
```

---

54) Can this keyword be used to refer static members?

Yes, It is possible to use this keyword to refer static members because this is just a reference

variable which refers to the current class object. However, as we know that, it is unnecessary to access static variables through objects, therefore, it is not the best practice to use this to refer static members. Consider the following example.

```
1.  public class Test
2.  {
3.      static int i = 10;
4.      public Test ()
5.      {
6.          System.out.println(this.i);
7.      }
8.      public static void main (String args[])
9.      {
10.         Test t = new Test();
11.     }
12. }
```

Output

10

---

55) How can constructor chaining be done using this keyword?

Constructor chaining enables us to call one constructor from another constructor of the class with respect to the current class object. We can use this keyword to perform constructor chaining within the same class. Consider the following example which illustrates how can we use this keyword to achieve constructor chaining.

```
1.  public class Employee
2.  {
3.      int id,age;
4.      String name, address;
5.      public Employee (int age)
6.      {
7.          this.age = age;
8.      }
9.      public Employee(int id, int age)
10.     {
11.         this(age);
12.         this.id = id;
13.     }
14.     public Employee(int id, int age, String name, String address)
15.     {
16.         this(id, age);
17.         this.name = name;
```

```
18.         this.address = address;
19.     }
20.     public static void main (String args[])
21.     {
22.         Employee emp = new Employee(105, 22, "Vikas", "Delhi");
23.         System.out.println("ID: "+emp.id+" Name:"+emp.name+" age:"+emp.age+" address
    : "+emp.address);
24.     }
25.
26. }
```

Output

ID: 105 Name:Vikas age:22 address: Delhi

---

56) What are the advantages of passing this into a method instead of the current class object itself?

As we know, that this refers to the current class object, therefore, it must be similar to the current class object. However, there can be two main advantages of passing this into a method instead of the current class object.

- this is a final variable. Therefore, this cannot be assigned to any new value whereas the current class object might not be final and can be changed.
- this can be used in the synchronized block.

---

57) What is the Inheritance?

Inheritance is a mechanism by which one object acquires all the properties and behavior of another object of another class. It is used for Code Reusability and Method Overriding. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also. Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

There are five types of inheritance in Java.

- Single-level inheritance
- Multi-level inheritance
- Multiple Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

Multiple inheritance is not supported in Java through class.

---

58) Why is Inheritance used in Java?

There are various advantages of using inheritance in Java that is given below.

- Inheritance provides code reusability. The derived class does not need to redefine the method of base class unless it needs to provide the specific implementation of the method.
- Runtime polymorphism cannot be achieved without using inheritance.
- We can simulate the inheritance of classes with the real-time objects which makes OOPs more realistic.
- Inheritance provides data hiding. The base class can hide some data from the derived class by making it private.
- Method overriding cannot be achieved without inheritance. By method overriding, we can give a specific implementation of some basic method contained by the base class.

---

59) Which class is the superclass for all the classes?

The object class is the superclass of all other classes in Java.

---

60) Why is multiple inheritance not supported in java?

To reduce the complexity and simplify the language, multiple inheritance is not supported in java. Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since the compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have the same method or different, there will be a compile time error.

```
1. class A{
2. void msg(){System.out.println("Hello");}
3. }
4. class B{
5. void msg(){System.out.println("Welcome");}
6. }
7. class C extends A,B{//suppose if it were
8.
```

```
9.    Public Static void main(String args[]){
10.    C obj=new C();
11.    obj.msg();//Now which msg() method would be invoked?
12. }
13. }
```

Compile Time Error

---

61) What is aggregation?

Aggregation can be defined as the relationship between two classes where the aggregate class contains a reference to the class it owns. Aggregation is best described as a **has-a** relationship. For example, The aggregate class Employee having various fields such as age, name, and salary also contains an object of Address class having various fields such as Address-Line 1, City, State, and pin-code. In other words, we can say that Employee (class) has an object of Address class. Consider the following example.

Address.java

```
1.  public class Address {
2.  String city,state,country;
3.
4.  public Address(String city, String state, String country) {
5.      this.city = city;
6.      this.state = state;
7.      this.country = country;
8.  }
9.
10. }
```

Employee.java

```
1.  public class Emp {
2.  int id;
3.  String name;
4.  Address address;
5.
6.  public Emp(int id, String name,Address address) {
7.      this.id = id;
8.      this.name = name;
9.      this.address=address;
10. }
11.
12. void display(){
13. System.out.println(id+" "+name);
```

```
14. System.out.println(address.city+" "+address.state+" "+address.country);
15. }
16.
17. public static void main(String[] args) {
18. Address address1=new Address("gzb","UP","india");
19. Address address2=new Address("gno","UP","india");
20.
21. Emp e=new Emp(111,"varun",address1);
22. Emp e2=new Emp(112,"arun",address2);
23.
24. e.display();
25. e2.display();
26.
27. }
28. }
```

Output

```
111 varun
gzb UP india
112 arun
gno UP india
```

---

62) What is composition?

Holding the reference of a class within some other class is known as composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition. In other words, we can say that composition is the particular case of aggregation which represents a stronger relationship between two objects. Example: A class contains students. A student cannot exist without a class. There exists composition between class and students.

---

63) What is the difference between aggregation and composition?

Aggregation represents the weak relationship whereas composition represents the strong relationship. For example, the bike has an indicator (aggregation), but the bike has an engine (composition).

---

64) Why does Java not support pointers?

The pointer is a variable that refers to the memory address. They are not used in Java because

they are unsafe(unsecured) and complex to understand.

---

65) What is super in java?

The **super** keyword in Java is a reference variable that is used to refer to the immediate parent class object. Whenever you create the instance of the subclass, an instance of the parent class is created implicitly which is referred by super reference variable. The super() is called in the class constructor implicitly by the compiler if there is no super or this.

```
1. class Animal{
2. Animal(){System.out.println("animal is created");}
3. }
4. class Dog extends Animal{
5. Dog(){
6. System.out.println("dog is created");
7. }
8. }
9. class TestSuper4{
10. public static void main(String args[]){
11. Dog d=new Dog();
12. }
13. }
```

Output:

animal is created
dog is created

---

66) How can constructor chaining be done by using the super keyword?

```
1. class Person
2. {
3.     String name,address;
4.     int age;
5.     public Person(int age, String name, String address)
6.     {
7.         this.age = age;
8.         this.name = name;
9.         this.address = address;
10.    }
11. }
12. class Employee extends Person
13. {
```

```
    float salary;
    public Employee(int age, String name, String address, float salary)
    {
        super(age,name,address);
        this.salary = salary;
    }
}
public class Test
{
    public static void main (String args[])
    {
        Employee e = new Employee(22, "Mukesh", "Delhi", 90000);
        System.out.println("Name: "+e.name+" Salary: "+e.salary+" Age: "+e.age+" Addres
    s: "+e.address);
    }
}
```

Output

Name: Mukesh Salary: 90000.0 Age: 22 Address: Delhi

---

67) What are the main uses of the super keyword?

There are the following uses of super keyword.

- super can be used to refer to the immediate parent class instance variable.
- super can be used to invoke the immediate parent class method.
- super() can be used to invoke immediate parent class constructor.

---

68) What are the differences between this and super keyword?

There are the following differences between this and super keyword.

- The super keyword always points to the parent class contexts whereas this keyword always points to the current class context.
- The super keyword is primarily used for initializing the base class variables within the derived class constructor whereas this keyword primarily used to differentiate between local and instance variables when passed in the class constructor.
- The super and this must be the first statement inside constructor otherwise the compiler will throw an error.

---

69) What is the output of the following Java program?

```
1.  class Person
2.  {
3.      public Person()
4.      {
5.          System.out.println("Person class constructor called");
6.      }
7.  }
8.  public class Employee extends Person
9.  {
10.     public Employee()
11.     {
12.         System.out.println("Employee class constructor called");
13.     }
14.     public static void main (String args[])
15.     {
16.         Employee e = new Employee();
17.     }
18. }
```

Output

Person class constructor called
Employee class constructor called

Explanation

The super() is implicitly invoked by the compiler if no super() or this() is included explicitly within the derived class constructor. Therefore, in this case, The Person class constructor is called first and then the Employee class constructor is called.

---

70) Can you use this() and super() both in a constructor?

No, because this() and super() must be the first statement in the class constructor.

Example:

```
1.  public class Test{
2.      Test()
3.       {
4.          super();
5.          this();
```

```
6.          System.out.println("Test class object is created");
7.      }
8.      public static void main(String []args){
9.      Test t = new Test();
10.     }
11. }
```

Output:

Test.java:5: error: call to this must be first statement in constructor

---

71)What is object cloning?

The object cloning is used to create the exact copy of an object. The clone() method of the Object class is used to clone an object. The **java.lang.Cloneable** interface must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates CloneNotSupportedException.

```
1. protected Object clone() throws CloneNotSupportedException
```

Core Java - OOPs Concepts: Method Overloading Interview Questions

---

72) What is method overloading?

Method overloading is the polymorphism technique which allows us to create multiple methods with the same name but different signature. We can achieve method overloading in two ways.

- Changing the number of arguments
- Changing the return type

Method overloading increases the readability of the program. Method overloading is performed to figure out the program quickly.

---

73) Why is method overloading not possible by changing the return type in java?

In Java, method overloading is not possible by changing the return type of the program due to avoid the ambiguity.

```
1. class Adder{
2. static int add(int a,int b){return a+b;}
```

```
3. static double add(int a,int b){return a+b;}
4. }
5. class TestOverloading3{
6. public static void main(String[] args){
7. System.out.println(Adder.add(11,11));//ambiguity
8. }}
```

Output:

Compile Time Error: method add(int, int) is already defined in class Adder

---

74) Can we overload the methods by making them static?

No, We cannot overload the methods by just applying the static keyword to them(number of parameters and types are the same). Consider the following example.

```
1. public class Animal
2. {
3.     void consume(int a)
4.     {
5.         System.out.println(a+" consumed!!");
6.     }
7.     static void consume(int a)
8.     {
9.         System.out.println("consumed static "+a);
10.    }
11.    public static void main (String args[])
12.    {
13.        Animal a = new Animal();
14.        a.consume(10);
15.        Animal.consume(20);
16.    }
17. }
```

Output

```
Animal.java:7: error: method consume(int) is already defined in class Animal
   static void consume(int a)
               ^
Animal.java:15: error: non-static method consume(int) cannot be referenced from a static context
       Animal.consume(20);
           ^
2 errors
```

---

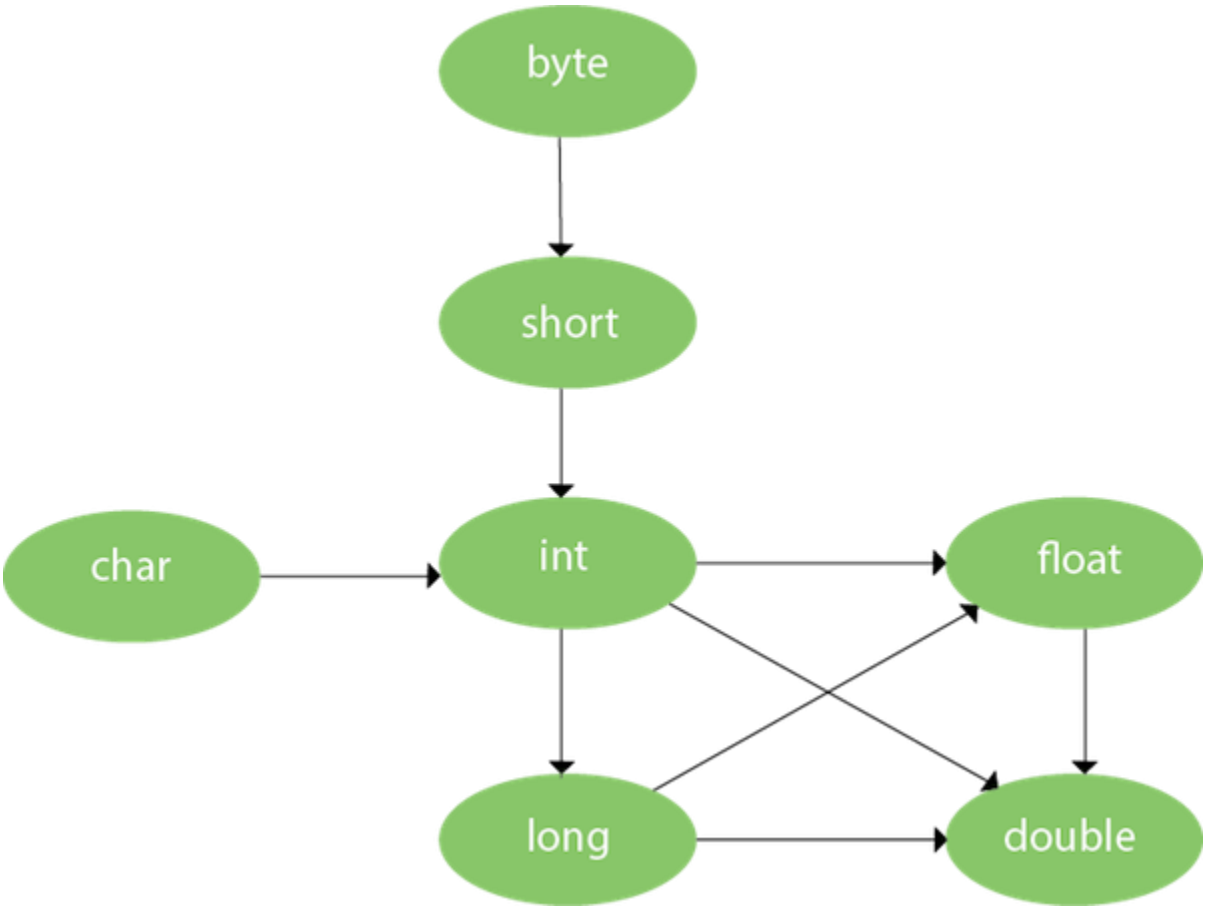
75) Can we overload the main() method?

Yes, we can have any number of main methods in a Java program by using method overloading.

---

76) What is method overloading with type promotion?

By Type promotion is method overloading, we mean that one data type can be promoted to another implicitly if no exact matching is found.



As displayed in the above diagram, the byte can be promoted to short, int, long, float or double. The short datatype can be promoted to int, long, float or double. The char datatype can be promoted to int, long, float or double and so on. Consider the following example.

```
1. class OverloadingCalculation1{
2.   void sum(int a,long b){System.out.println(a+b);}
3.   void sum(int a,int b,int c){System.out.println(a+b+c);}
4.
5.   public static void main(String args[]){
```

```
6.    OverloadingCalculation1 obj=new OverloadingCalculation1();
7.    obj.sum(20,20);//now second int literal will be promoted to long
8.    obj.sum(20,20,20);
9.    }
10. }
```

Output

```
40
60
```

---

77) What is the output of the following Java program?

```
1.  class OverloadingCalculation3{
2.    void sum(int a,long b){System.out.println("a method invoked");}
3.    void sum(long a,int b){System.out.println("b method invoked");}
4.
5.    public static void main(String args[]){
6.    OverloadingCalculation3 obj=new OverloadingCalculation3();
7.    obj.sum(20,20);//now ambiguity
8.    }
9.  }
```

Output

```
OverloadingCalculation3.java:7: error: reference to sum is ambiguous
obj.sum(20,20);//now ambiguity
   ^
    both method sum(int,long) in OverloadingCalculation3
    and method sum(long,int) in OverloadingCalculation3 match
1 error
```

Explanation

There are two methods defined with the same name, i.e., sum. The first method accepts the integer and long type whereas the second method accepts long and the integer type. The parameter passed that are a = 20, b = 20. We can not tell that which method will be called as there is no clear differentiation mentioned between integer literal and long literal. This is the case of ambiguity. Therefore, the compiler will throw an error.

Core Java - OOPs Concepts: Method Overriding Interview Questions

---

78) What is method overriding:

If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding. It is used for runtime polymorphism and to implement the interface methods.

Rules for Method overriding

- The method must have the same name as in the parent class.
- The method must have the same signature as in the parent class.
- Two classes must have an IS-A relationship between them.

---

79) Can we override the static method?

No, you can't override the static method because they are the part of the class, not the object.

---

80) Why can we not override static method?

It is because the static method is the part of the class, and it is bound with class whereas instance method is bound with the object, and static gets memory in class area, and instance gets memory in a heap.

---

81) Can we override the overloaded method?

Yes.

---

82) Difference between method Overloading and Overriding.

Method Overloading	Method Overriding
1) Method overloading increases the readability of the program.	Method overriding provides the specific implementation of the method that is already provided by its superclass.
2) Method overloading occurs within the class.	Method overriding occurs in two classes that have IS-A relationship between them.
3) In this case, the parameters must	In this case, the parameters must be the same.

be different.

---

83) Can we override the private methods?

No, we cannot override the private methods because the scope of private methods is limited to the class and we cannot access them outside of the class.

---

84) Can we change the scope of the overridden method in the subclass?

Yes, we can change the scope of the overridden method in the subclass. However, we must notice that we cannot decrease the accessibility of the method. The following point must be taken care of while changing the accessibility of the method.

- The private can be changed to protected, public, or default.
- The protected can be changed to public or default.
- The default can be changed to public.
- The public will always remain public.

---

85) Can we modify the throws clause of the superclass method while overriding it in the subclass?

Yes, we can modify the throws clause of the superclass method while overriding it in the subclass. However, there are some rules which are to be followed while overriding in case of exception handling.

- If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception, but it can declare the unchecked exception.
- If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.

---

86) What is the output of the following Java program?

```
1. class Base
2. {
3.     void method(int a)
4.     {
5.         System.out.println("Base class method called with integer a = "+a);
6.     }
```

```
7.
8.      void method(double d)
9.      {
10.         System.out.println("Base class method called with double d ="+d);
11.     }
12. }
13.
14. class Derived extends Base
15. {
16.     @Override
17.     void method(double d)
18.     {
19.         System.out.println("Derived class method called with double d ="+d);
20.     }
21. }
22.
23. public class Main
24. {
25.     public static void main(String[] args)
26.     {
27.         new Derived().method(10);
28.     }
29. }
```

Output

Base class method called with integer a = 10

Explanation

The method() is overloaded in class Base whereas it is derived in class Derived with the double type as the parameter. In the method call, the integer is passed.

87) Can you have virtual functions in Java?

Yes, all functions in Java are virtual by default.

---

88) What is covariant return type?

Now, since java5, it is possible to override any method by changing the return type if the return type of the subclass overriding method is subclass type. It is known as covariant return type. The covariant return type specifies that the return type may vary in the same direction as the subclass.

```
1. class A{
```

```
2.  A get(){return this;}
3.  }
4.
5.  class B1 extends A{
6.  B1 get(){return this;}
7.  void message(){System.out.println("welcome to covariant return type");}
8.
9.  public static void main(String args[]){
10. new B1().get().message();
11. }
12. }
```

Output: welcome to covariant return type

---

89) What is the output of the following Java program?

```
1.  class Base
2.  {
3.      public void baseMethod()
4.      {
5.          System.out.println("BaseMethod called ...");
6.      }
7.  }
8.  class Derived extends Base
9.  {
10.     public void baseMethod()
11.     {
12.         System.out.println("Derived method called ...");
13.     }
14. }
15. public class Test
16. {
17.     public static void main (String args[])
18.     {
19.         Base b = new Derived();
20.         b.baseMethod();
21.     }
22. }
```

Output

Derived method called ...

Explanation

The method of Base class, i.e., baseMethod() is overridden in Derived class. In Test class, the reference variable b (of type Base class) refers to the instance of the Derived class. Here, Runtime polymorphism is achieved between class Base and Derived. At compile time, the presence of method baseMethod checked in Base class, If it presence then the program compiled otherwise the compiler error will be shown. In this case, baseMethod is present in Base class; therefore, it is compiled successfully. However, at runtime, It checks whether the baseMethod has been overridden by Derived class, if so then the Derived class method is called otherwise Base class method is called. In this case, the Derived class overrides the baseMethod; therefore, the Derived class method is called.

Core Java - OOPs Concepts: final keyword Interview Questions

---

90) What is the final variable?

In Java, the final variable is used to restrict the user from updating it. If we initialize the final variable, we can't change its value. In other words, we can say that the final variable once assigned to a value, can never be changed after that. The final variable which is not assigned to any value can only be assigned through the class constructor.


```
1. class Bike9{
2.  final int speedlimit=90;//final variable
3.  void run(){
4.   speedlimit=400;
5.  }
6.  public static void main(String args[]){
```

```
7.    Bike9 obj=new  Bike9();
8.    obj.run();
9.    }
10. }//end of class
```

Output:Compile Time Error

---

91) What is the final method?

If we change any method to a final method, we can't override it.

```
1.  class Bike{
2.    final void run(){System.out.println("running");}
3.  }
4.
5.  class Honda extends Bike{
6.     void run(){System.out.println("running safely with 100kmph");}
7.
8.     public static void main(String args[]){
9.     Honda honda= new Honda();
10.    honda.run();
11.    }
12. }
```

Output:Compile Time Error

---

92) What is the final class?

If we make any class final, we can't inherit it into any of the subclasses.

```
1.  final class Bike{}
2.
3.  class Honda1 extends Bike{
4.    void run(){System.out.println("running safely with 100kmph");}
5.
6.    public static void main(String args[]){
7.    Honda1 honda= new Honda1();
8.    honda.run();
9.    }
10. }
```

Output:Compile Time Error

---

93) What is the final blank variable?

A final variable, not initialized at the time of declaration, is known as the final blank variable. We can't initialize the final blank variable directly. Instead, we have to initialize it by using the class constructor. It is useful in the case when the user has some data which must not be changed by others, for example, PAN Number. Consider the following example:

```
1. class Student{
2. int id;
3. String name;
4. final String PAN_CARD_NUMBER;
5. ...
6. }
```

---

94) Can we initialize the final blank variable?

Yes, if it is not static, we can initialize it in the constructor. If it is static blank final variable, it can be initialized only in the static block.

---

95) Can you declare the main method as final?

Yes, We can declare the main method as public static final void main(String[] args){}.

---

96) What is the output of the following Java program?

```
1. class Main {
2.  public static void main(String args[]){
3.    final int i;
4.    i = 20;
5.    System.out.println(i);
6.  }
7. }
```

Output

20

Explanation

Since i is the blank final variable. It can be initialized only once. We have initialized it to 20. Therefore, 20 will be printed.

---

97) What is the output of the following Java program?

```
1.  class Base
2.  {
3.      protected final void getInfo()
4.      {
5.          System.out.println("method of Base class");
6.      }
7.  }
8.
9.  public class Derived extends Base
10. {
11.     protected final void getInfo()
12.     {
13.         System.out.println("method of Derived class");
14.     }
15.     public static void main(String[] args)
16.     {
17.         Base obj = new Base();
18.         obj.getInfo();
19.     }
20. }
```

Output

```
        Derived.java:11: error: getInfo() in Derived cannot override getInfo() in Base
   protected final void getInfo()
                        ^
  overridden method is final
1 error
```

Explanation

The getDetails() method is final; therefore it can not be overridden in the subclass.

---

98) Can we declare a constructor as final?

The constructor can never be declared as final because it is never inherited. Constructors are not ordinary methods; therefore, there is no sense to declare constructors as final. However, if you

try to do so, The compiler will throw an error.

---

99) Can we declare an interface as final?

No, we cannot declare an interface as final because the interface must be implemented by some class to provide its definition. Therefore, there is no sense to make an interface final. However, if you try to do so, the compiler will show an error.

---

100) What is the difference between the final method and abstract method?

The main difference between the final method and abstract method is that the abstract method cannot be final as we need to override them in the subclass to give its definition.

---

101) What is the difference between compile-time polymorphism and runtime polymorphism?

There are the following differences between compile-time polymorphism and runtime polymorphism.

SN	compile-time polymorphism	Runtime polymorphism
1	In compile-time polymorphism, call to a method is resolved at compile-time.	In runtime polymorphism, call to an overridden method is resolved at runtime.
2	It is also known as static binding, early binding, or overloading.	It is also known as dynamic binding, late binding, overriding, or dynamic method dispatch.
3	Overloading is a way to achieve compile-time polymorphism in which, we can define multiple methods or constructors with different signatures.	Overriding is a way to achieve runtime polymorphism in which, we can redefine some particular method or variable in the derived class. By using overriding, we can give some specific implementation to the base class properties in the derived class.
4	It provides fast execution because the type of an object is determined at compile-time.	It provides slower execution as compare to compile-time because the type of an object is determined at run-time.
5	Compile-time polymorphism provides less flexibility because all	Run-time polymorphism provides more flexibility

the things are resolved at compile-time.	because all the things are resolved at runtime.

---

102) What is Runtime Polymorphism?

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

```
1. class Bike{
2.   void run(){System.out.println("running");}
3. }
4. class Splendor extends Bike{
5.   void run(){System.out.println("running safely with 60km");}
6.   public static void main(String args[]){
7.     Bike b = new Splendor();//upcasting
8.     b.run();
9.   }
10. }
```

Output:

running safely with 60km.

In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

---

103) Can you achieve Runtime Polymorphism by data members?

No, because method overriding is used to achieve runtime polymorphism and data members cannot be overridden. We can override the member functions but not the data members. Consider the example given below.

```
1. class Bike{
2.  int speedlimit=90;
3. }
4. class Honda3 extends Bike{
5.  int speedlimit=150;
6.  public static void main(String args[]){
7.   Bike obj=new Honda3();
```

```
8.     System.out.println(obj.speedlimit);//90
9.      }
```

Output:

90

---

104) What is the difference between static binding and dynamic binding?

In case of the static binding, the type of the object is determined at compile-time whereas, in the dynamic binding, the type of the object is determined at runtime.

Static Binding

```
1.  class Dog{
2.   private void eat(){System.out.println("dog is eating...");}
3.
4.   public static void main(String args[]){
5.    Dog d1=new Dog();
6.    d1.eat();
7.   }
8.  }
```

Dynamic Binding

```
1.  class Animal{
2.   void eat(){System.out.println("animal is eating...");}
3.  }
4.
5.  class Dog extends Animal{
6.   void eat(){System.out.println("dog is eating...");}
7.
8.   public static void main(String args[]){
9.    Animal a=new Dog();
10.  a.eat();
11.  }
12. }
```

---

105) What is the output of the following Java program?

```
1.  class BaseTest
2.  {
3.    void print()
```

```
4.    {
5.      System.out.println("BaseTest:print() called");
6.    }
7.  }
8.  public class Test extends BaseTest
9.  {
10.   void print()
11.   {
12.     System.out.println("Test:print() called");
13.   }
14.   public static void main (String args[])
15.   {
16.     BaseTest b = new Test();
17.     b.print();
18.   }
19. }
```

Output

```
Test:print() called
```

Explanation

It is an example of Dynamic method dispatch. The type of reference variable b is determined at runtime. At compile-time, it is checked whether that method is present in the Base class. In this case, it is overridden in the child class, therefore, at runtime the derived class method is called.

---

106) What is Java instanceOf operator?

The instanceof in Java is also known as type comparison operator because it compares the instance with type. It returns either true or false. If we apply the instanceof operator with any variable that has a null value, it returns false. Consider the following example.

```
1.  class Simple1{
2.   public static void main(String args[]){
3.   Simple1 s=new Simple1();
4.   System.out.println(s instanceof Simple1);//true
5.   }
6.  }
```

Output

```
true
```

An object of subclass type is also a type of parent class. For example, if Dog extends Animal then object of Dog can be referred by either Dog or Animal class.

Core Java - OOPs Concepts: Abstraction Interview Questions

---

107) What is the abstraction?

Abstraction is a process of hiding the implementation details and showing only functionality to the user. It displays just the essential things to the user and hides the internal information, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery. Abstraction enables you to focus on what the object does instead of how it does it. Abstraction lets you focus on what the object does instead of how it does it.

In Java, there are two ways to achieve the abstraction.

- Abstract Class
- Interface

---

108) What is the difference between abstraction and encapsulation?

Abstraction hides the implementation details whereas encapsulation wraps code and data into a single unit.

---

109) What is the abstract class?

A class that is declared as abstract is known as an abstract class. It needs to be extended and its method implemented. It cannot be instantiated. It can have abstract methods, non-abstract methods, constructors, and static methods. It can also have the final methods which will force the subclass not to change the body of the method. Consider the following example.

```
1. abstract class Bike{
2.   abstract void run();
3. }
4. class Honda4 extends Bike{
5. void run(){System.out.println("running safely");}
6. public static void main(String args[]){
```

```
7.    Bike obj = new Honda4();
8.    obj.run();
9.   }
10. }
```

Output

running safely

---

110) Can there be an abstract method without an abstract class?

No, if there is an abstract method in a class, that class must be abstract.

---

111) Is the following program written correctly? If yes then what will be the output of the program?

```
1.  abstract class Calculate
2.  {
3.      abstract int multiply(int a, int b);
4.  }
5.
6.  public class Main
7.  {
8.      public static void main(String[] args)
9.      {
10.         int result = new Calculate()
11.         {
12.             @Override
13.             int multiply(int a, int b)
14.             {
15.                 return a*b;
16.             }
17.         }.multiply(12,32);
18.         System.out.println("result = "+result);
19.     }
20. }
```

Yes, the program is written correctly. The Main class provides the definition of abstract method multiply declared in abstract class Calculation. The output of the program will be:

Output

384

112) Can you use abstract and final both with a method?

No, because we need to override the abstract method to provide its implementation, whereas we can't override the final method.

---

113) Is it possible to instantiate the abstract class?

No, the abstract class can never be instantiated even if it contains a constructor and all of its methods are implemented.

---

114) What is the interface?

The interface is a blueprint for a class that has static constants and abstract methods. It can be used to achieve full abstraction and multiple inheritance. It is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. Java Interface also represents the IS-A relationship. It cannot be instantiated just like the abstract class. However, we need to implement it to define its methods. Since Java 8, we can have the default, static, and private methods in an interface.

---

115) Can you declare an interface method static?

No, because methods of an interface are abstract by default, and we can not use static and abstract together.

---

116) Can the Interface be final?

No, because an interface needs to be implemented by the other class and if it is final, it can't be implemented by any class.

---

117) What is a marker interface?

A Marker interface can be defined as the interface which has no data member and member

functions. For example, Serializable, Cloneable are marker interfaces. The marker interface can be declared as follows.

```
1. public interface Serializable{
2. }
```

---

118) What are the differences between abstract class and interface?

Abstract class	Interface
An abstract class can have a method body (non-abstract methods).	The interface has only abstract methods.
An abstract class can have instance variables.	An interface cannot have instance variables.
An abstract class can have the constructor.	The interface cannot have the constructor.
An abstract class can have static methods.	The interface cannot have static methods.
You can extend one abstract class.	You can implement multiple interfaces.
The abstract class **can provide the implementation of the interface**.	The Interface **can't provide the implementation of the abstract class**.
The **abstract keyword** is used to declare an abstract class.	The **interface keyword** is used to declare an interface.
An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
An **abstract class** can be extended using keyword **extends**	An **interface class** can be implemented using keyword **implements**
A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

Example:
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

---

119) Can we define private and protected modifiers for the members in interfaces?

No, they are implicitly public.

---

120) When can an object reference be cast to an interface reference?

An object reference can be cast to an interface reference when the object implements the referenced interface.

---

121) How to make a read-only class in Java?

A class can be made read-only by making all of the fields private. The read-only class will have only getter methods which return the private property of the class to the main method. We cannot modify this property because there is no setter method available in the class. Consider the following example.

```
1.  //A Java class which has only getter methods.
2. public class Student{
3. //private data member
4. private String college="AKG";
5. //getter method for college
6. public String getCollege(){
7. return college;
8. }
9. }
```

---

122) How to make a write-only class in Java?

A class can be made write-only by making all of the fields private. The write-only class will have only setter methods which set the value passed from the main method to the private fields. We cannot read the properties of the class because there is no getter method in this class. Consider the following example.

```
1.  //A Java class which has only setter methods.
2. public class Student{
3. //private data member
4. private String college;
5. //getter method for college
6. public void setCollege(String college){
7. this.college=college;
8. }
9. }
```

123) What are the advantages of Encapsulation in Java?

There are the following advantages of Encapsulation in Java?

- By providing only the setter or getter method, you can make the class read-only or write-only. In other words, you can skip the getter or setter methods.
- It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.
- It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members.
- The encapsulate class is easy to test. So, it is better for unit testing.
- The standard IDE's are providing the facility to generate the getters and setters. So, it is easy and fast to create an encapsulated class in Java.

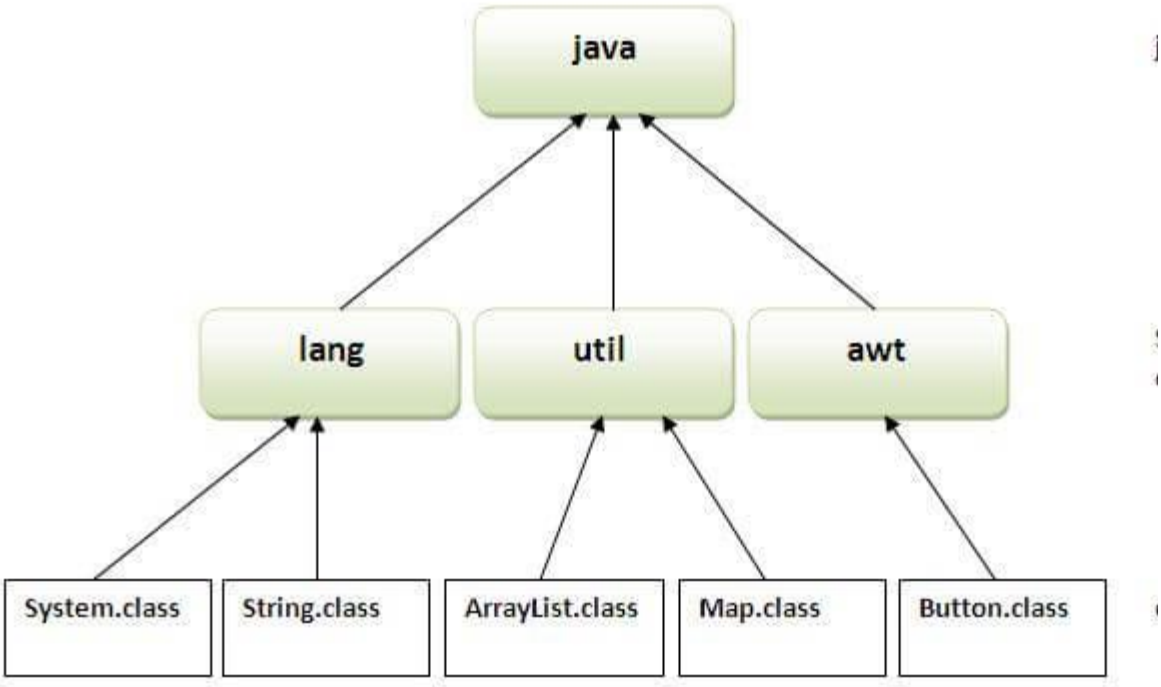
Core Java - OOPs Concepts: Package Interview Questions

---

124) What is the package?

A package is a group of similar type of classes, interfaces, and sub-packages. It provides access protection and removes naming collision. The packages in Java can be categorized into two forms, inbuilt package, and user-defined package. There are many built-in packages such as Java, lang, awt, javax, swing, net, io, util, sql, etc. Consider the following example to create a package in Java.

```
1. //save as Simple.java
2. package mypack;
3. public class Simple{
4.  public static void main(String args[]){
5.     System.out.println("Welcome to package");
6.    }
7. }
```



---

125) What are the advantages of defining packages in Java?

By defining packages, we can avoid the name conflicts between the same class names defined in different packages. Packages also enable the developer to organize the similar classes more effectively. For example, one can clearly understand that the classes present in java.io package are used to perform io related operations.

---

126) How to create packages in Java?

If you are using the programming IDEs like Eclipse, NetBeans, MyEclipse, etc. click on **file->new->project** and eclipse will ask you to enter the name of the package. It will create the project package containing various directories such as src, etc. If you are using an editor like notepad for java programming, use the following steps to create the package.

- Define a package **package_name**. Create the class with the name **class_name** and save this file with **your_class_name.java**.

- Now compile the file by running the following command on the terminal.
    1. javac -d . your_class_name.java

    The above command creates the package with the name **package_name** in the present working directory.

- Now, run the class file by using the absolute class file name, like following.

    1. java package_name.class_name

---

127) How can we access some class in another class in Java?

There are two ways to access a class in another class.

- **By using the fully qualified name:** To access a class in a different package, either we must use the fully qualified name of that class, or we must import the package containing that class.
- **By using the relative path**, We can use the path of the class that is related to the package that contains our class. It can be the same or subpackage.

---

128) Do I need to import java.lang package any time? Why?

No. It is by default loaded internally by the JVM.

---

129) Can I import same package/class twice? Will the JVM load the package twice at runtime?

One can import the same package or the same class multiple times. Neither compiler nor JVM complains about it. However, the JVM will internally load the class only once no matter how many times you import the same class.

---

130) What is the static import?

By static import, we can access the static members of a class directly, and there is no to qualify it with the class name.

Java: Exception Handling Interview Questions

There is given a list of exception handling interview questions with answers. If you know any exception handling interview question, kindly post it in the comment section.

---

131) How many types of exception can occur in a Java program?

There are mainly two types of exceptions: checked and unchecked. Here, an error is considered as the unchecked exception. According to Oracle, there are three types of exceptions:

- **Checked Exception:** Checked exceptions are the one which are checked at compile-time. For example, SQLException, ClassNotFoundException, etc.

- **Unchecked Exception:** Unchecked exceptions are the one which are handled at runtime because they can not be checked at compile-time. For example, ArithmaticException, NullPointerException, ArrayIndexOutOfBoundsException, etc.

- **Error:** Error cause the program to exit since they are not recoverable. For Example, OutOfMemoryError, AssertionError, etc.

---

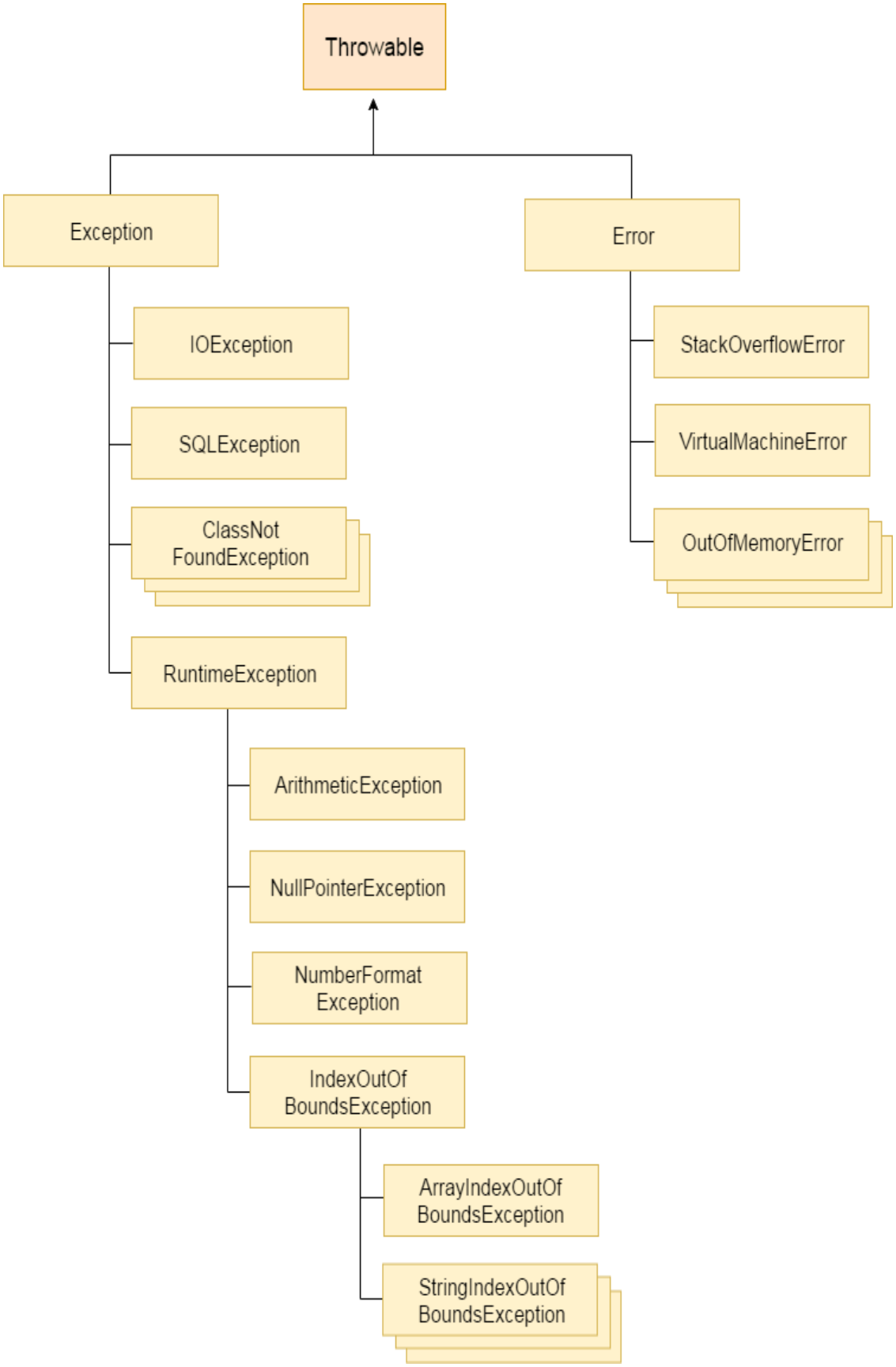
132) What is Exception Handling?

Exception Handling is a mechanism that is used to handle runtime errors. It is used primarily to handle checked exceptions. Exception handling maintains the normal flow of the program. There are mainly two types of exceptions: checked and unchecked. Here, the error is considered as the unchecked exception.

---

133) Explain the hierarchy of Java Exception classes?

The java.lang.Throwable class is the root class of Java Exception hierarchy which is inherited by two subclasses: Exception and Error. A hierarchy of Java Exception classes are given below:



---

134) What is the difference between Checked Exception and Unchecked Exception?

1) Checked Exception