Hi team,

Modified points for Hackathon.

**Maven**

**POM with page factory**

**Cucumber – feature file, step definition and test runner with TestNG**

**TestNG – grouping, rerun failed test cases, multiple XML or Multiple Suite for Smoke and Regression suite**

**Rerun failed testcases with Listeners upto max 2 times**

**Data driven Framework – Excel, Property file and [XML or JSON] 2 file types is mandatory**

**Extent report with screenshot and logs**

**Jenkins with Maven POM xml path**

Hi Team, As questioned about cross browser ... by thilagai, Deva (Cognizant)

thilagai, Deva (Cognizant)

Tuesday 3:22 PM

Hi Team,

As questioned about cross browser testing in cucumber with TestNG, you can refer the below link.

<https://www.numpyninja.com/post/cross-browser-testing-cucumber-with-testng>

Link https://www.softwaretestingmaterial.co... by thilagai, Deva (Cognizant)

thilagai, Deva (Cognizant)

Tuesday 3:26 PM

<https://www.softwaretestingmaterial.com/execute-maven-project-using-jenkins/>

Use the above link can be used for executin... by thilagai, Deva (Cognizant)

thilagai, Deva (Cognizant)

Tuesday 3:28 PM

Use the above link can be used for executing maven POM.xml file in Jenkins

Link https://toolsqa.com/jenkins/jenkins-ma... by thilagai, Deva (Cognizant)

thilagai, Deva (Cognizant)

Tuesday 3:29 PM

<https://toolsqa.com/jenkins/jenkins-maven/>

**What is Cucumber?**

Cucumber is an open-source testing tool that supports the [Behavior Driven Development (BDD)](https://www.lambdatest.com/blog/behaviour-driven-development-by-selenium-testing-with-gherkin/) method. It can be integrated with a variety of testing frameworks and tools, including [Selenium](https://www.lambdatest.com/selenium) and [Appium](https://www.lambdatest.com/appium), and it supports numerous programming languages. Since Cucumber offers a standard language for describing requirements and test scenarios, it facilitates collaboration between developers, testers, and business analysts in Agile software development approaches.

**What language is used by Cucumber?**

**Answer:**[Gherkin](https://github.com/cucumber/cucumber/wiki/Gherkin) is the language that is used by the Cucumber tool. It is a simple English representation of the application behaviour.

Gherkin language uses several keywords to describe the behaviour of applications such as Feature, Scenario, Scenario Outline, Given, When, Then, etc.

**What are the different types of testing that can be performed using Cucumber?**

Cucumber is a testing tool that supports various types of testing, such as acceptance testing, functional testing, and regression testing.

* [Acceptance testing](https://www.lambdatest.com/learning-hub/acceptance-testing) includes testing software's performance and compliance with requirements. Cucumber simplifies verifying that the software complies with the required specifications by allowing teams to define acceptance tests in a human-readable language.
* [Functional testing](https://www.lambdatest.com/learning-hub/functional-testing) includes evaluating certain software features or functions to ensure they function as intended. Teams can build and carry out available tests using Cucumber in a language that technical and non-technical stakeholders can easily comprehend.
* [Regression testing](https://www.lambdatest.com/blog/regression-testing-what-is-and-how-to-do-it/) requires retesting the software following changes to ensure that the previously functional features are still operational.

**What is meant by a feature file?**

**Answer:** A feature file must provide a high-level description of an Application Under Test (AUT). The first line of the feature file must start with the keyword ‘Feature’ followed by the description of the application under test.

A feature file may include multiple scenarios within the same file. A feature file has the extension *.feature.*

**What are the various keywords that are used in Cucumber for writing a scenario?**

Mentioned below are the keywords that are used for writing a scenario:

* Given
* When
* Then
* And
* But

**Write the files needed to implement a Cucumber test scenario.**

The 3 files you need to implement a Cucumber test scenario:

* Features
* Step Definition
* Test Runner

**What is the purpose of a Scenario Outline in Cucumber?**

**Answer:**Scenario outline is a way of parameterization of scenarios. This is ideally used when the same scenario needs to be executed for multiple sets of data, however, the test steps remain the same. Scenario Outline must be followed by the keyword ‘Examples’, which specify the set of values for each parameter.

**What are annotations with respect to Cucumber?**

Annotations are specific types of texts that have been pre-defined and have a specified meaning in Cucumber. It instructs the compiler/interpreter on what to do when the program runs. There is the following set of annotations that are defined and can be used in Cucumber:

**Given:** ‘Given’ specifies the various requirements for running a test.

Ex - Given I have an account on code studio.

**When:** ‘When’ establishes the beginning/starting point for any test scenario.

Ex - When I log in to code studio.

**Then:** 'Then' contains the expected results of a test to be executed.

Ex - Then registration should be successful.

**And:** 'And' is used Between any two statements to specify the logical AND condition. It can be combined with GIVEN, WHEN, and THEN statements.

Ex - When I enter my account number AND CVV.

**But:** ‘Or’ denotes a logical OR relationship between two propositions. OR can also be combined with GIVEN, WHEN, and THEN statements.

Ex - Then I should be logged in BUT I must enter the OTP.

**What is the purpose of the *Given* step in Cucumber?**

The Given step in Cucumber is used to specify the initial state or the prerequisites necessary for the test scenario to execute. In addition to helping to build up the necessary test data and create the required test objects, it describes the environment in which the test is executed.

**What is the purpose of the *When* step in Cucumber?**

The When step in Cucumber is used to specify the action or event that must occur as part of the test scenario. It outlines the procedures needed to simulate a user's interaction with the system being tested.

**What is the purpose of the *Then* step in Cucumber?**

The Then step in Cucumber defines the desired result or test scenario result. It outlines the necessary validation or verification processes to ascertain whether the system under test is acting by expectations.

**Differentiate between Cucumber and JBehave.**

Even though both are used for the same purpose, the acceptance tests have completely different frameworks:

* Cucumber is Ruby-based, while JBehave is purely Java framework.
* Cucumber is based on features, while JBehave is based on stories.

**Provide an example of a feature file using the Cucumber framework.**

**Answer:** Following is an example of a feature file for the scenario ‘Login into the application’:

**Feature:** Login to the application under test.  
**Scenario:** Login to the application.

* Open the Chrome browser and launch the application.
* When the user enters the username onto the UserName field.
* And User enters the password into the Password field.
* When the user clicks on the Login button.
* Then validate if the user login is successful.

**Provide an example of a Scenario Outline using the Cucumber framework.**

**Answer:** The following is an **example** of a Scenario Outline keyword for the scenario ‘Upload a file’. The number of parameter values to be included in the feature file is based on the tester’s choice.

**Scenario Outline:** Upload a file

Given that the user is on upload file screen.  
When a user clicks on the Browse button.  
And user enters <filename> onto the upload textbox.  
And user clicks on the enter button.  
Then verify that the file upload is successful.

**Example:**

|filename|  
|file1|  
|file2|

**What is a ‘feature’ in Cucumber?**

A 'Feature' in a cucumber project can be described as a stand-alone unit or functionality. A feature frequently includes a specific list of scenarios to test for with it. The Feature File is a file in which we store features, descriptions of features, and situations to be evaluated. For each feature under test, it is recommended that a separate feature file be created. A feature file has a ".*feature*" extension. One can make and add as many feature files as required.

Example of the same:-

For an e-commerce website, one can have the following set of features:-

Users can register/sign up on the website.

Users can sign in to their accounts using their credentials.

Users can add products to their cart.

Users can check out products that they added to their carts.

Users can pay for their items.

Users can sign out from the website.

**What is the purpose of the Behaviour Driven Development (BDD) methodology in the real world?**

**Answer:** BDD is a methodology to understand the functionality of an application in the simple plain text representation.

The main aim of the Behavior Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers, Support Teams understand the application without diving deep into the technical aspects.

**How does Behavioral Driven Development work?**

There are three primary steps involved in Behavioral Driven Development:

* Behavior Description: List the features of the application in the feature file.
* Creating the Step Definition file: Map the scenario between each step to define in the feature file, and a code of the function that needs to be implemented is stored in the steps definition file.
* Testing and running: Run test cases to check if it passes. Usually, many failures are seen before getting to the final code.

**What is the limit for the maximum number of scenarios that can be V**

**Answer:** A feature file can contain a maximum of 10 scenarios, but the number can vary from project to project and from one organization to another. But it is generally advisable to limit the number of scenarios included in the feature file.

**What is the use of Background keyword in Cucumber?**

**Answer:** Background keyword is used to group multiple given statements into a single group. This is generally used when the same set of given statements are repeated in each scenario of the feature file.

**What is debugging in Cucumber?**

In Cucumber, debugging entails finding and fixing problems with your step definitions, feature files, or entire test suite. Typical debugging methods include:

* Printing Debug Information: To understand programme flow and variable values, use print statements or logging to produce information at various places in your step definitions.
* Using an integrated development environment (IDE), add breakpoints in your code to halt execution while debugging and allow you to view variables.
* Cucumber frequently offers a "dry run" option that examines the syntax of your step definitions without running tests, assisting in detecting potential problems.
* Reviewing Error Messages: To determine where errors and exceptions occurred, look over error messages and stack traces.
* Review feature files to ensure that Gherkin syntax and scenario definitions are correct.

**Describe the use of the Options tag in the Cucumber Framework.**

In the Cucumber Framework, the Options tag comes under the TestRunner file. The tag takes the form of an annotation named @CucumberOptions.

It has two parameters:

* Feature: The path to the file is defined using the feature option.
* Glue: This argument provides the step definition file’s location.

**Explain how execution starts in Cucumber.**

Cucumber execution starts at the support level, and there, the first env.rb file is loaded, followed by hooks.rb, and finally, the implementation of feature file scenario steps.

**Should any code be written within the TestRunner class?**

The answer is no. No code needs to be written under the TestRunner class. The tags @RunWith and @CucumberOptions should be included.

**What is the difference between a Background and a Scenario Outline in Cucumber?**

A Background is a set of steps that every scenario in a feature file shares. The same scenario is run several times with varying input values using a scenario outline. A Scenario Outline can be used several times to run the same scenario with different input data sets. However the Background section can only be used once per feature file.

**What is meant by hooks in Cucumber?**

The blocks of code that function before or after each scenario in the Cucumber execution cycle are called hooks.

**What are hooks in Cucumber? How can they be used in Cucumber?**

Hooks are sets of instruction statements that execute before or after every Cucumber scenario in an execution cycle. It enables one to control the development workflow better and decrease code redundancy. Setting up the web driver and terminating the web driver session resembles a test setup. When dealing with different scenarios, it's best to do the setup and clean up only once. Hooks are used for adding optimisations in Cucumber tests.

Certain preconditions, such as executing a program, creating/setting up a database connection, preparing test data, etc., may sometimes be required. Several postconditions could also be needed, such as ending/closing the database connections, closing the browser, refreshing test data, logging out of the program etc. Cucumber handles all of these situations using hooks.

One can use the @Before and @After methods to define hooks anywhere in the project or the step definition layers. One can define hooks to get executed before any other test situations and after all test scenarios have been completed.

**How do you run multiple scenarios in Cucumber?**

Cucumber can be implemented in parallel with the help of TestNG and Maven test execution plugins by setting up the data provider parallel option to true.

**What are data tables in Cucumber?**

A data table is a set of input that needs to be provided for a single tag. This tag can be — GIVEN, WHEN, or THEN.

**What are tags in Cucumber, and why are they important?**

It appears to be simple when one only has one, two, or maybe five situations in a feature file. Practically, however, this does not occur. In a single feature file, one may have 10, 20, or even more scenarios for each feature under test. Tags could reflect various purposes (smoke test/regression test), perspectives (developer/QA/BA), and statuses (ready for execution/work in progress).

Tags in Cucumber provide one with a way to run scenarios in a specific sequence from a runner file. One can label each situation with a useful tag. Later, in the runner file, we may specify which tag (and hence which scenario(s)) Cucumber should run. "@" is the first character in a tag. One can use any relevant content after "@" to define your tag. Example - '@InitialTest'

**How do you generate HTML reports in Cucumber?**

To create HTML reports, Cucumber comes with a built-in plugin called HTML Formatter. To create reports with more specific information, you can also use third-party plugins like ExtentReports or Allure Reports.

**What is a cucumber report?**

A report in Cucumber is a document that shows the outcomes of the tests that have been run. After a test run is over, cucumber reports are generated. They offer essential details about the test execution, including the state of each test scenario and the overall state of the test suite.

**What symbol is used for parameterization in Cucumber?**

**Answer:** Pipe symbol (|) is used to specify one or more parameter values in a feature file.

**What is the purpose of Examples keyword in Cucumber?**

**Ans:** Examples keyword is used to specify values for each parameter used in the scenario. Scenario Outline keyword must always be followed by the keyword Examples.

What is the purpose of the Cucumber Options... by thilagai, Deva (Cognizant)

thilagai, Deva (Cognizant)

Friday 11:08 PM

**What is the purpose of the Cucumber Options tag?**

Cucumber Options tag is used to provide a link between the feature files and step definition files. Each step of the feature file is mapped to a corresponding method on the step definition file.

**Below is the syntax of Cucumber Options tag:**

@CucumberOptions(features="Features",glue={"StepDefinition"})

**When is Cucumber used in real-time?**

**Answer:** Cucumber tool is generally used in real-time to write acceptance tests for an application. It is generally used by non-technical people such as Business Analysts, Functional Testers, etc.

**What is the use of Behavior Driven Development in Agile methodology?**

**Answer:** The advantages of Behavior Driven Development are best realized when non-technical users such as Business Analysts use BDD to draft requirements and provide the same to the developers for implementation.

In Agile methodology, user stories can be written in the format of feature file and the same can be taken up for implementation by the developers.

**What is a cucumber scenario outline example?**

A cucumber scenario outline example is a template for creating various scenarios from a collection of input values. The scenario outline is a framework for building a collection of connected scenarios with similar actions but distinct input values. The different input values that will be used to produce the methods are represented by placeholders referred to as "sample tables." Consider the following example of a login function that accepts various usernames and passwords:

Scenario Outline: User logs in with valid credentials  Given the user is on the login page  When they enter "<username>" and "<password>"  And they click the login button  Then they should be redirected to the dashboard Examples:  
 | username | password |  
 | user1 | pass1 |  
 | user2 | pass2 |  
 | user3 | pass3 |

In this illustration, the scenario outline offers a framework for building a variety of login scenarios using various identities and passwords. The input values for each situation are listed in the example table. Each row in the sample table receives its scenario when Cucumber executes this scenario description.

**What are Cucumber parameters?**

You can provide dynamic values to your step definitions using cucumber parameters, which are also known as placeholders or variables in Gherkin scenarios. They are denoted by angle brackets (< >) and serve as placeholders for actual data when running tests. Parameters enhance reusability and flexibility in your scenarios. For example, <username> and <password> can be parameters for logging into different accounts. In step definitions, you can capture and use these values for testing.

**Name a few files that serve as a data source for various Selenium frameworks.**

They can be an XML, Excel, CSV, property file or even a Text file.

**What is the difference between a feature and a scenario in Cucumber?**

|  |  |
| --- | --- |
| **Feature in Cucumber** | **Scenario in Cucumber** |
| A feature in Cucumber is a high-level description of a software need or feature. It often has one or more situations and is defined in a feature file. | A scenario is a particular test case or illustration that confirms a specific behavior or result of the feature. |
| The "Feature" keyword is used at the start of a feature in the Gherkin language. | The "Scenario" keyword is used at the start of scenarios written in the Gherkin language. |
| Specific test cases can be described by one or more scenarios for a feature. | One or more steps in a scenario can specify the behaviour that will be tested. |

**What is the difference between a scenario and a step in Cucumber?**

|  |  |
| --- | --- |
| **Scenario in Cucumber** | **Step in Cucumber** |
| A scenario is a single test case that describes one particular aspect of an application's functionality or behavior. | A step is a distinctive assertion or action that is carried out as a part of a scenario. |
| Consists of one or more steps. | Consists of only one step. |
| Can have a variety of stages with various contexts. | Can have various scenarios with various steps. |
| Used to verify and validate the feature's acceptance criteria. | Used to specify particular test procedures to verify the system's operation. |
| Aids in understanding the system's overall behavior. | Aids in understanding a particular behavior of the system. |

**What is the difference between JUnit and Cucumber?**

|  |  |
| --- | --- |
| **JUnit** | **Cucumber** |
| JUnit is a unit testing framework that is generally used to test isolated code modules or units. It offers a collection of annotations and assertions for writing and running tests for Java applications. | Cucumber is a framework for behavior-driven testing that focuses on evaluating a system's overall behavior by specifying scenarios and actions using Gherkin syntax in natural language. |
| JUnit uses annotations to specify test methods (@Test, @Before, @After). | Cucumber defines scenarios using the "given-when-then" syntax (Gherkin language) |
| The reading and understanding of tests can be difficult, particularly for non-technical stakeholders. | Scenarios are stated in simple terms so that non-technical stakeholders can understand them. |
| Can integrate with testing tools like Mockito and Selenium. | Can integrate with testing tools like Selenium and RestAssured |
| JUnit delivers basic test reporting | Cucumber provides a detailed test reporting in JSON, HTML and other formats |

**What does one mean by the Test Driven Development (TDD) approach to testing?**

Test-Driven Development (TDD) is a development practice in which the test cases are created first, followed by the program-based implementations of the business requirements that underpins the test cases. TDD may also be used in automation testing. The TDD approach takes longer to develop due to the fact that it generates fewer flaws in the developed implementation. The TDD development practice results in increased quality of code, which is also much more reusable and flexible. TDD also aids developers in achieving high test coverage, ranging from 90% to 100%. The sole overhead of the TDD approach of testing for developers is that they must build the test cases before producing code.

**Why do we need to use Cucumber with Selenium?**

Both Cucumber and Selenium are two of the most widely used testing frameworks and technologies. Selenium is widely used for functional testing by a lot of organisations. They use Cucumber in conjunction with Selenium as that makes the application flow much simpler to read and comprehend. The most crucial advantage of combining Cucumber and Selenium is that it allows developers to build test cases in simple feature files that the managers, non-technical stakeholders, and business analysts can easily understand. It allows one to develop tests in Gherkin, a human-readable and comprehensible programming language. Java,.NET, PHP, Python, Perl, and other programming languages are also supported by the Selenium-Cucumber framework.

**How can you handle data-driven testing in Cucumber? Explain with an example.**

You can use data tables or scenario outlines to handle data-driven testing in Cucumber. With the help of examples, we can see how to use data-driven testing in Cucumber.

Scenario Outline: User Registration  
Given the user is on the registration page  
When you enter the "<username>", "<email>", and "<password>"  
And click the register button  
Then you should see a "<successMessage>"  
Examples:  
| username | email          | password | success message      |  
| user1    | user1@cn.com   | pass1    | Welcome, User 1!     |  
| user2    | user2@cn.com   | pass2    | Welcome, User 2!     |  
| user3    | user3@ecn.com  | pass3    | Registration failed. |

**You can also try this code with Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

In this example, the scenario outline captures the steps for the user registration process. The placeholders "<username>," "<email>," "<password>," and "<successMessage>" represent the variables that will be replaced with specific values.

The table contains multiple rows, each representing a different user with their respective username, email, password, and the expected success or failure message upon registration. The cucumber will execute the scenario for each row and substitute the placeholders with the corresponding values. By utilizing data-driven testing, you can efficiently test the registration process with different user details within a single scenario. This approach helps validate the registration functionality across various user inputs. It ensures a robust and reliable system.

**What is the meaning of the TestRunner class in Cucumber?**

**Answer:** TestRunner class is used to provide the link between the feature file and the step definition file. The next question provides a sample representation of how the TestRunner class will look like. A TestRunner class is generally an empty class with no class definition.

**What is the difference between TestNG and Cucumber?**

|  |  |
| --- | --- |
| **Cucumber** | **TestNG** |
| Cucumber is used for acceptability testing | TestNG is mostly used for unit testing. |
| Cucumber supports a variety of programming languages | TestNG is focused on Java-based testing. |
| Cucumber provides a more natural language approach using Gherkin syntax for defining tests | TestNG is more developer-centric and relies on annotations. |
| Cucumber offers a behavior-driven development (BDD) strategy that stresses cooperation between developers, testers, and business stakeholders. | TestNG does not have a behavior-driven development (BDD) focus |

**What is the difference between a hook and a step definition in Cucumber?**

|  |  |
| --- | --- |
| **Hook in Cucumber** | **Step Definition in Cucumber** |
| A hook in Cucumber is a section of code that runs before or after a scenario or feature. | A step definition is a piece of code that specifies a step's behavior in a feature file. |
| Suitable for setup and takedown operations like starting a browser or cutting off a database connection. | Can be used to carry out any automation job, including communicating with UI elements and executing API calls. |
| Using context objects or global variables, data can be shared between scenarios or steps. | Can leverage Cucumber's built-in data sharing features, such as scenario outline tables or background steps, but cannot share data between steps or scenarios. |
| Can be used to conditionally change or skip scenarios depending on specific circumstances. | Can be used to construct conditional logic within a step's implementation, but cannot be used to skip or change scenarios. |

**What is the difference between a scenario and a feature file in Cucumber?**

The crucial Cucumber components Scenario and Feature both have a specific function. The key variations between them are as follows:

|  |  |
| --- | --- |
| **Feature File** | **Scenario** |
| A text file called a feature file contains a detailed description of a feature that needs to be tested. | A single test case in the Gherkin language is known as a scenario. |
| It outlines the software's functioning, which needs to be tested. | It outlines the specific steps to be taken to test a specific functionality. |
| It can involve a variety of scenarios, each of which tests a distinct function. | It usually comprises of a number of steps that are carried out in a particular order. |
| The scenarios are placed within a context, and they are arranged into logical groups. | To organise and group scenarios, it can be connected to one or more tags. |

**What is the difference between a tag and a data table in Cucumber?**

|  |  |
| --- | --- |
| **Tag** | **Data Table** |
| Tag begins with the "@" symbol | Data Tablr starts with " |
| Tags organize and filter scenarios | Data Table passes numerous parameters to a scenario |
| A scenario may contain several tags. | A scenario can contain only one data table. |
| The step definitions don't use tags. | With step definitions, data tables are used as arguments. |

**What is the difference between a test scenario and a test case in Cucumber?**

|  |  |
| --- | --- |
| **Test Scenario** | **Test Case** |
| A test scenario is a specific instance of a feature. | A test case is a set of conditions or variables that a tester will use to verify whether or not a system is functioning properly. |
| Test Scenario is defined using Gherkin syntax and consists of a collection of stages that specify the activities and expected outcomes for specific system behavior. | Test Case is often defined by a QA engineer and contains comprehensive instructions on how to test a particular system feature or behavior. |

**What is the difference between a soft assertion and a hard assertion in Cucumber?**

|  |  |
| --- | --- |
| **Soft Assertion** | **Hard Assertion** |
| continues running tests even after a failure | stops testing after a failure |
| Logs the error message before moving on to the next assertion. | logs the error message and stops the test. |
| can be helpful in situations when several claims need to be verified before the test is stopped. | useful in situations where the failure of a single assertion signals the failure of a test |

**How do you use Cucumber hooks?**

A Cucumber Hook is executed at specific times during the Cucumber test cycle. Hooks are used to perform post-actions after a scenario or step has been completed and to set up preconditions before they are met. Many hook annotations are provided by Cucumber, which can be used to specify the intended behavior. A few are listed below.

* @Before: This hook is executed before each scenario or scenario step. It can be used to do any required setup tasks or to set up the test environment.
* @After: This hook is used to determine whether a scenario or a step in a scenario is to be executed. It can be used to teardown the test environment or perform cleanup operations.
* @BeforeStep - It is used before each step in a scenario. It can be used to carry out particular tasks before each stage is carried out.
* @AfterStep: In a scenario, this hook is used after each step. It can be used to carry out particular tasks upon the completion of each stage.

**How do you integrate Cucumber with Jenkins?**

To integrate Cucumber with Jenkins, you can follow these steps:

* Install the Jenkins Cucumber plugin.
* For your Cucumber project, create a Jenkins job.
* Set up the Jenkins task to use a build script or the command line to run the Cucumber tests.
* Set up the Cucumber plugin in the Jenkins job to generate Cucumber reports.
* You can either manually start the Jenkins task or schedule it to execute at a particular time.

pipeline {  agent any  stages {    stage('Build') {      steps {        sh 'mvn clean install'      }    }    stage('Test') {      steps {        sh 'mvn test'      }  
      post {  
        always { cucumber htmlReports: true, jsonReports: true }  
 }  
 }  
 }  
}

This Jenkinsfile creates and tests a Maven-based Cucumber project and uses the Cucumber plugin to produce HTML and JSON reports. This file can be modified to suit the needs of your project.

**How do you define a background in Cucumber?**

A background in Cucumber is a set of functions that each scenario in a feature file uses. It is employed to specify the prerequisites that must be satisfied before the scenarios can be carried out.

In Cucumber, you may define a background by using the “Background” keyword followed by the steps you want to describe. Here's an illustration:

Feature: User registration  Background:    Given the user is on the registration page  Scenario: Valid registration    When the user enters valid information    Then the user should be registered successfully Scenario: Invalid registration  
 When the user enters invalid information  
 Then the user should see an error message

The Background keyword is used to indicate a single step in the example above, namely that the user is now on the registration page. Before both scenarios in the feature file, this step will be carried out.

**How do you handle dynamic input in Cucumber?**

Cucumber can handle dynamic input by using a scenario outline with an Examples table. Here is a brief sample of code:

Scenario Outline: Login with dynamic credentials  Given User is on login page  When User enters "<username>" and "<password>"  And User clicks on login button  Then User should be logged in successfully  Examples:  
 | username | password |  
 | user1 | pass123 |  
 | user2 | pass456 |

In this example, the steps of the scenario were defined using the scenario outline, and dynamic input was then added using the Examples table. The table contains the actual values for the placeholders "username" and "password." For each entry in the table, Cucumber will automatically construct a scenario, replacing the placeholders in the scenario outline with values from the table.

**How do you write a comment in a feature file in Cucumber?**

Cucumber allows you to add comments to feature files by using the "#" sign and then the comment text. Here's an illustration:

Feature: Login  As a registered user  I want to login to my account  So that I can access my dashboard  
  
  # This is a comment Scenario: Valid login  
 Given I am on the login page  
 When I enter valid credentials  
 And click the login button  
 Then I should be redirected to the dashboard page

The comment is denoted in this example by the hash mark ("#") and is typed on a separate line before the term "Scenario."

**How do you run Cucumber tests in parallel?**

Using a test runner like JUnit or TestNG and configuring it to execute tests concurrently will allow us to run Cucumber tests in parallel. Here is an illustration of coding using TestNG:

import cucumber.api.CucumberOptions;import cucumber.api.testng.AbstractTestNGCucumberTests;@CucumberOptions(features = {"src/test/resources/features"},                 glue = {"com.example.steps"})public class TestRunner extends AbstractTestNGCucumberTests {    // nothing to do here, TestNG will run the Cucumber tests in parallel}

You can define the location of the feature files and step definitions in the @CucumberOptions annotation. Then, develop a TestNG test runner by extending AbstractTestNGCucumberTests. In accordance with the settings in the testng.xml configuration file, TestNG will automatically run the Cucumber tests in parallel.

How do you organize your Cucumber feature f... by thilagai, Deva (Cognizant)

thilagai, Deva (Cognizant)

Friday 11:28 PM

**How do you organize your Cucumber feature files?**

Organizing your feature files according to the functionality or module they cover is a good idea. For each module, you can make a directory and place the feature files that go with it. Finding and managing the files are made simpler as a result.

- features/  - authentication/  
    - login.feature  
    - registration.feature  
  - user\_management/  
    - create\_user.feature  
    - delete\_user.feature  
  - orders/  
    - create\_order.feature  
    - view\_order.feature

The feature files in this example are organized into functional groups and stored in subdirectories. The feature files are kept small and focused on a particular functionality, which makes them simpler to execute and maintain.

### 1. What is Java?

Java is a high-level, object-oriented programming language that was developed by Sun Microsystems in 1995. It is platform-independent, meaning that programs written in Java can run on any platform that has a Java Virtual Machine (JVM) installed. [Visit Oracle’s official website to download JVM.](https://www.oracle.com/in/java/technologies/downloads/#jdk20-windows)

### 2. What are the features of Java?

Java has several features that make it a popular programming language. Some of these features include platform independence, object-oriented programming, automatic memory management, robustness, and security.

### 3. What is JVM?

JVM stands for Java Virtual Machine. It is an abstract machine that provides the runtime environment in which Java programs are executed. The JVM interprets Java bytecode and translates it into machine-specific code.

### 4. What is the difference between JDK, JRE, and JVM?

JDK stands for Java Development Kit. It is a software development kit that includes tools for developing, compiling, and debugging Java programs. JRE stands for Java Runtime Environment. It is a software environment that provides the necessary runtime libraries and components for running Java programs. JVM is the virtual machine that executes the Java bytecode.

### 5. What is the difference between a class and an object?

A class is a blueprint or template for creating objects, while an object is an instance of a class. In other words, a class defines the properties and behaviours of an object, while an object is an instance of those properties and behaviours.

### 6. What is object-oriented programming?

Object-oriented programming (OOP) is a programming paradigm that is based on the concept of objects. In OOP, programs are organized around objects, which have properties (attributes) and behaviors (methods). OOP emphasizes encapsulation, inheritance, polymorphism, and abstraction.

### 7. What are the four principles of OOP?

The four principles of OOP are encapsulation, inheritance, polymorphism, and abstraction.

### 8. What is inheritance?

Inheritance is a mechanism in which one class inherits properties and methods from another class. The class that inherits is called the subclass or derived class, and the class that is inherited from is called the superclass or base class.

### 9. What is encapsulation?

Encapsulation is a mechanism in which data and methods are combined into a single unit called a class. Encapsulation provides data hiding, which means that the internal implementation details of a class are hidden from the outside world.

### 10. What is polymorphism?

Polymorphism is a mechanism in which a single method can have different forms or implementations. Polymorphism can be achieved through method overloading or method overriding.

### 11. What is abstraction?

Abstraction is a mechanism in which complex systems can be simplified by hiding unnecessary details. In Java, abstraction can be achieved through abstract classes and interfaces.

### 12. What are access modifiers?

Access modifiers are keywords that are used to specify the accessibility of a class, method, or variable. There are four types of access modifiers in Java: public, private, protected, and default.

### What are the differences between C++ and Java?

### Concept.

C++ is not platform-independent; the principle behind C++ programming is “write once, compile anywhere.”

In contrast, because the byte code generated by the Java compiler is platform-independent, it can run on any machine, Java programs are written once and run everywhere.

**What do you get in the Java download file? How do they differ from one another?**

We get two major things along with the Java Download file.

JDK - [Java Development Kit](https://www.simplilearn.com/tutorials/java-tutorial/jdk-in-java)

JRE - Java Runtime Environment

|  |  |
| --- | --- |
| **JDK** | **JRE** |
| Abbreviation for JavaDevelopment Kit | Abbreviation for Java Runtime Environment |
| JDK is a dedicated kit for solely software development | JRE is a set of software and library designed for executing Java Programs |
| Unlike JVM, JDK is Platform Dependent | Unlike JVM, JRE is also Platform Dependent |
| JDK package is a set of tools for debugging and Developing | JRE Package is one that only supports files and libraries for a runtime environment |
| JDK package will be provided with an installer file | JRE Package does not get an installer but has only a runtime environment |

**What is a ClassLoader?**

A classloader in Java is a subsystem of Java Virtual Machine, dedicated to loading class files when a program is executed; ClassLoader is the first to load the executable file.

[11:35 AM] thilagai, Deva (Cognizant)

### What are static variables and methods?

Static variables and methods belong to the class rather than to an instance of the class. This means that they can be accessed without creating an object of the class.

### What are the final variables and methods?

Final variables and methods cannot be modified or overridden once they are defined. Final variables are constants, while final methods cannot be overridden by subclasses.

### What is a constructor?

A constructor is a special method that is used to initialize objects. It has the same name as the class and does not have a return type.

### What are the different types of variables in Java?

There are three types of variables in Java: local variables, instance variables, and static variables.

**Why is Java a platform independent language?**

Java language was developed so that it does not depend on any hardware or software because the **compiler** compiles the code and then converts it to platform-independent byte code which can be run on multiple systems.

The only condition to run that byte code is for the machine to have a runtime environment (JRE) installed in it.

### ****Why is Java not a pure object oriented language?****

Java supports primitive data types - byte, boolean, char, short, int, float, long, and double and hence it is not a pure [**object oriented language**](https://www.interviewbit.com/oops-interview-questions/).

**Can java be said to be the complete object-oriented programming language?**

It is not wrong if we claim that Java is the complete object-oriented programming language because everything in Java is under the classes and we can access them by creating the objects.

But we can even say that Java is not a completely object-oriented programming language because it has the support of primitive data types like int, float, char, boolean, double, etc.

Now for the question: **Is Java a completely object-oriented programming language?** We can say that - Java is not a pure object-oriented programming language, because it has direct access to primitive data types. And these primitive data types don't directly belong to the Integer classes.

**What do you understand by an instance variable and a local variable?**

**Instance variables** are those variables that are accessible by all the methods in the class. They are declared outside the methods and inside the class. These variables describe the properties of an object and remain bound to it at any cost.

All the objects of the class will have their copy of the variables for utilization. If any modification is done on these variables, then only that instance will be impacted by it, and all other class instances continue to remain unaffected.

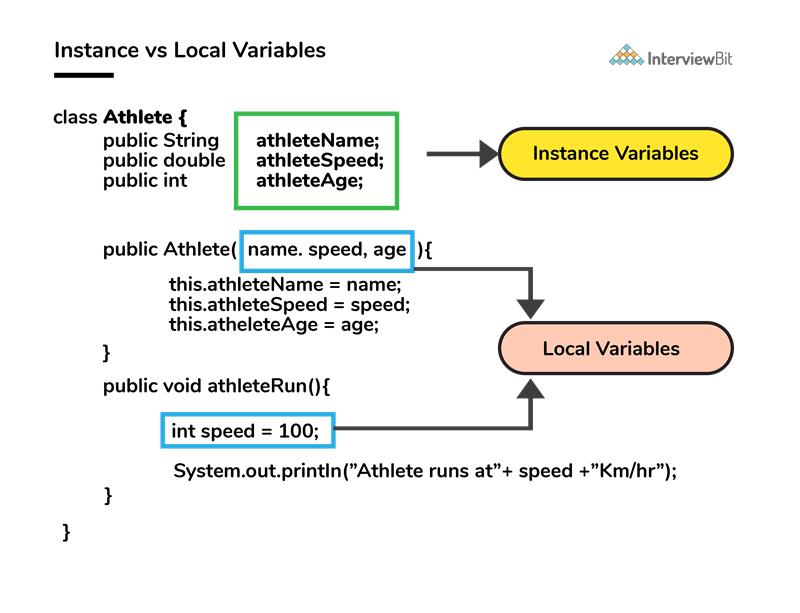
**Example:**

class Athlete {public String athleteName;public double athleteSpeed;public int athleteAge;  
}

**Local variables** are those variables present within a block, function, or constructor and can be accessed only inside them. The utilization of the variable is restricted to the block scope. Whenever a local variable is declared inside a method, the other class methods don’t have any knowledge about the local variable.

**Example:**

public void athlete() {  
String athleteName;double athleteSpeed;int athleteAge;  
}



**What are the default values assigned to variables and instances in java?**

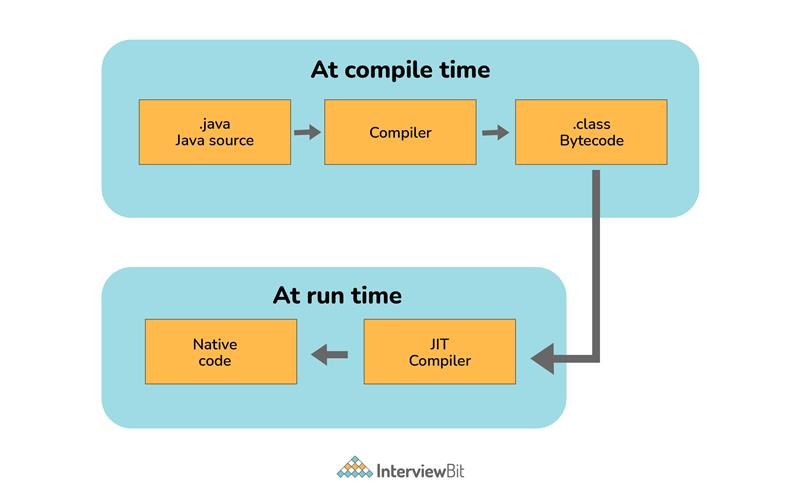
* There are no default values assigned to the variables in java. We need to initialize the value before using it. Otherwise, it will throw a compilation error of (**Variable might not be initialized**).
* But for instance, if we create the object, then the default value will be initialized by the default constructor depending on the data type.
* If it is a reference, then it will be assigned to null.
* If it is numeric, then it will assign to 0.
* If it is a boolean, then it will be assigned to false. Etc.

**What do you mean by data encapsulation?**

* Data Encapsulation is an Object-Oriented Programming concept of hiding the data attributes and their behaviours in a single unit.
* It helps developers to follow modularity while developing software by ensuring that each object is independent of other objects by having its own methods, attributes, and functionalities.
* It is used for the security of the private properties of an object and hence serves the purpose of data hiding.

**Tell us something about JIT compiler.**

* JIT stands for Just-In-Time and it is used for improving the performance during run time. It does the task of compiling parts of byte code having similar functionality at the same time thereby reducing the amount of compilation time for the code to run.
* The compiler is nothing but a translator of source code to machine-executable code. But what is special about the JIT compiler? Let us see how it works:
  + First, the Java source code (.java) conversion to byte code (.class) occurs with the help of the javac compiler.
  + Then, the .class files are loaded at run time by JVM and with the help of an interpreter, these are converted to machine understandable code.
  + JIT compiler is a part of JVM. When the JIT compiler is enabled, the JVM analyzes the method calls in the .class files and compiles them to get more efficient and native code. It also ensures that the prioritized method calls are optimized.
  + Once the above step is done, the JVM executes the optimized code directly instead of interpreting the code again. This increases the performance and speed of the execution.



### ****Can you tell the difference between equals() method and equality operator (==) in Java?****

We are already aware of the **(==) equals** operator. That we have used this to compare the equality of the values. But when we talk about the terms of object-oriented programming, we deal with the values in the form of objects. And this object may contain multiple types of data. So using the **(==) operator** does not work in this case. So we need to go with the .**equals() method.**

Both **[(==)** and **.equals()]** primary functionalities are to compare the values, but the secondary functionality is different.

So in order to understand this better, let’s consider this with the example -

String str1 = "InterviewBit";  
String str2 = "InterviewBit";  
   
System.out.println(str1 == str2);

This code will print true. We know that both strings are equals so it will print true. But here **(==) Operators** don’t compare each character in this case. It compares the memory location. And because the string uses the constant pool for storing the values in the memory, both **str1** and **str2** are stored at the same memory location.

[11:40 AM] thilagai, Deva (Cognizant)

|  |  |
| --- | --- |
| **equals()** | **==** |
| This is a method defined in the Object class. | It is a binary operator in Java. |
| The .equals() Method is present in the Object class, so we can override our custom .equals() method in the custom class, for objects comparison. | It cannot be modified. They always compare the HashCode. |
| This method is used for checking the equality of contents between two objects as per the specified business logic. | This operator is used for comparing addresses (or references), i.e checks if both the objects are pointing to the same memory location. |

 [11:40 AM] thilagai, Deva (Cognizant)

**How is an infinite loop declared in Java?**

Infinite loops are those loops that run infinitely without any breaking conditions. Some examples of consciously declaring infinite loop is:

* Using For Loop:

for (;;)  
{  
   // Business logic// Any break logic  
}

* Using while loop:

while(true){  
   // Business logic// Any break logic  
}

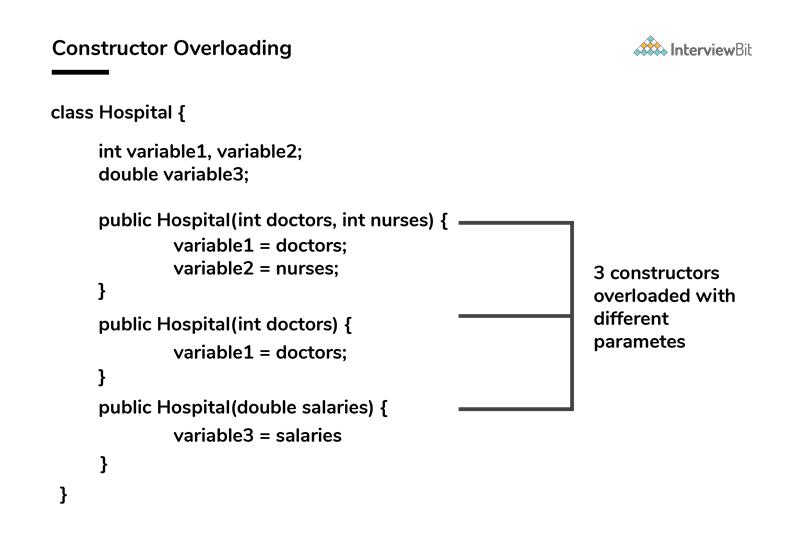
* Using do-while loop:

do{  
   // Business logic// Any break logic}while(true);

### ****Briefly explain the concept of constructor overloading****

Constructor overloading is the process of creating multiple constructors in the class consisting of the same name with a difference in the constructor parameters. Depending upon the number of parameters and their corresponding types, distinguishing of the different types of constructors is done by the compiler.

class Hospital {int variable1, variable2;double variable3;public Hospital(int doctors, int nurses) {  
 variable1 = doctors;  
 variable2 = nurses;  
}public Hospital(int doctors) {  
 variable1 = doctors;  
}public Hospital(double salaries) {  
 variable3 = salaries  
}  
}



Three constructors are defined here but they differ on the basis of parameter type and their numbers.

### ****Define Copy constructor in java.****

Copy Constructor is the constructor used when we want to initialize the value to the new object from the old object of the same class.

class InterviewBit{  
   String department;   String service;   InterviewBit(InterviewBit ib){  
       this.departments = ib.departments;  
 this.services = ib.services;  
 }  
}

Here we are initializing the new object value from the old object value in the constructor.

### ****Can the main method be Overloaded?****

Yes, It is possible to overload the main method. We can create as many overloaded main methods we want. However, JVM has a predefined calling method that JVM will only call the main method with the definition of -

public static void main(string[] args)

Consider the below code snippets:

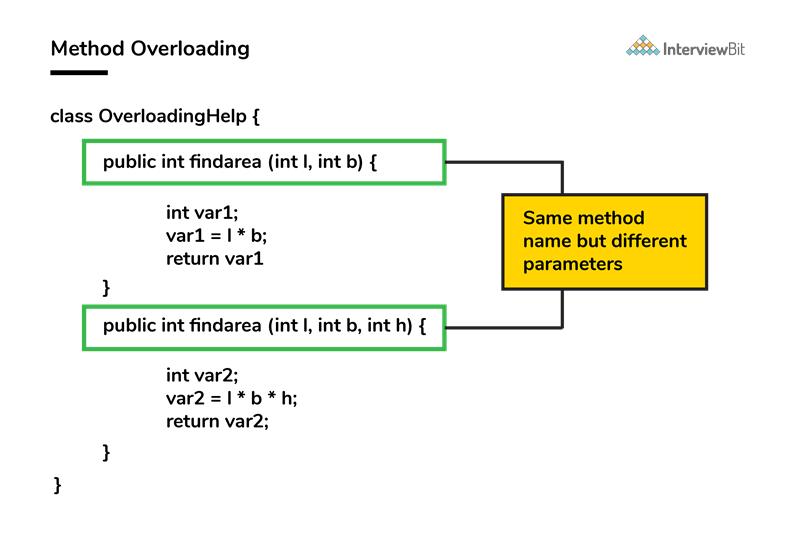
class Main {  
    public static void main(String args[]) {  
        System.out.println(" Main Method");  
    }    public static void main(int[] args){  
        System.out.println("Overloaded Integer array Main Method");  
    }    public static void main(char[] args){  
 System.out.println("Overloaded Character array Main Method");  
 }  
 public static void main(double[] args){  
 System.out.println("Overloaded Double array Main Method");  
 }  
 public static void main(float args){  
 System.out.println("Overloaded float Main Method");  
 }  
}

### ****Comment on method overloading and overriding by citing relevant examples.****

In Java, **method overloading** is made possible by introducing different methods in the same class consisting of the same name. Still, all the functions differ in the number or type of parameters. It takes place inside a class and enhances program readability.

The only difference in the return type of the method does not promote method overloading. The following example will furnish you with a clear picture of it.

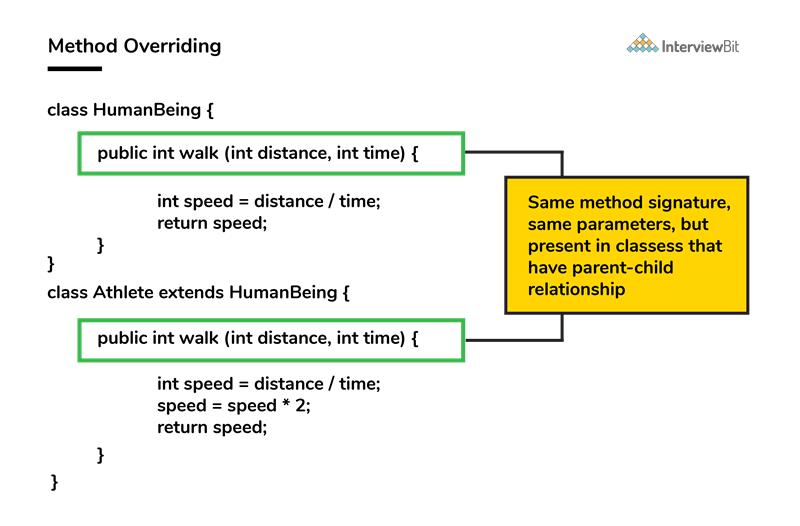
class OverloadingHelp {  
   public int findarea (int l, int b) {  
           int var1;  
           var1 = l \* b;           return var1;  
   }  
   public int findarea (int l, int b, int h) {  
 int var2;  
 var2 = l \* b \* h;  
 return var2;  
 }  
}



Both the functions have the same name but differ in the number of arguments. The first method calculates the area of the rectangle, whereas the second method calculates the area of a cuboid.

**Method overriding** is the concept in which two methods having the same method signature are present in two different classes in which an inheritance relationship is present. A particular method implementation (already present in the base class) is possible for the derived class by using method overriding.  
Let’s give a look at this example:

class HumanBeing {  
       public int walk (int distance, int time) {  
               int speed = distance / time;  
               return speed;  
       }  
}class Athlete extends HumanBeing {  
       public int walk(int distance, int time) {  
               int speed = distance / time;  
 speed = speed \* 2;  
 return speed;  
 }  
}



Both class methods have the name walk and the same parameters, distance, and time. If the derived class method is called, then the base class method walk gets overridden by that of the derived class.

### ****A single try block and multiple catch blocks can co-exist in a Java Program. Explain.****

Yes, multiple catch blocks can exist but specific approaches should come prior to the general approach because only the first catch block satisfying the catch condition is executed. The given code illustrates the same:

public class MultipleCatch {public static void main(String args[]) {  
 try {  
  int n = 1000, x = 0;  
  int arr[] = new int[n];  
  for (int i = 0; i <= n; i++) {  
   arr[i] = i / x;  }  
 }  
 catch (ArrayIndexOutOfBoundsException exception) {  
  System.out.println("1st block = ArrayIndexOutOfBoundsException");  
 }  
 catch (ArithmeticException exception) {  
 System.out.println("2nd block = ArithmeticException");  
 }  
 catch (Exception exception) {  
 System.out.println("3rd block = Exception");  
 }  
}  
}

Here, the second catch block will be executed because of division by 0 (i / x). In case x was greater than 0 then the first catch block will execute because for loop runs till i = n and array index are till n-1.

**Explain the use of final keyword in variable, method and class.**

In Java, the final keyword is used as defining something as constant /final and represents the non-access modifier.

* **final variable:**
  + When a variable is declared as final in Java, the value can’t be modified once it has been assigned.
  + If any value has not been assigned to that variable, then it can be assigned only by the constructor of the class.
* **final method:**
  + A method declared as final cannot be overridden by its children's classes.
  + A constructor cannot be marked as final because whenever a class is inherited, the constructors are not inherited. Hence, marking it final doesn't make sense. Java throws compilation error saying - modifier final not allowed here
* **final class:**
  + No classes can be inherited from the class declared as final. But that final class can extend other classes for its usage.

**19. Do final, finally and finalize keywords have the same function?**

All three keywords have their own utility while programming.

**Final:** If any restriction is required for classes, variables, or methods, the final keyword comes in handy. Inheritance of a final class and overriding of a final method is restricted by the use of the final keyword. The variable value becomes fixed after incorporating the final keyword. Example:

final int a=100;  
a = 0;  // error

The second statement will throw an error.

**Finally:** It is the block present in a program where all the codes written inside it get executed irrespective of handling of exceptions. Example:

try {int variable = 5;  
}catch (Exception exception) {  
System.out.println("Exception occurred");  
}finally {  
System.out.println("Execution of finally block");  
}

**Finalize:** Prior to the garbage collection of an object, the finalize method is called so that the clean-up activity is implemented. Example:

public static void main(String[] args) {  
String example = new String("InterviewBit");  
example = null;  
System.gc(); // Garbage collector called}public void finalize() {// Finalize called}

**Is it possible that the ‘finally’ block will not be executed? If yes then list the case.**

 Yes. It is possible that the ‘finally’ block will not be executed. The cases are-

* Suppose we use **System.exit()** in the above statement.
* If there are fatal errors like Stack overflow, Memory access error, etc.

[11:49 AM] thilagai, Deva (Cognizant)

**Identify the output of the java program and state the reason.**

1. public class InterviewBit2. {3.   public static void main(String[] args) {4.      final int i;5.     i = 20;6.     int j = i+20;7.     i = j+30;8.       System.out.println(i + " " + j);9.   }10. }

The above code will generate a compile-time error at Line 7 saying - **[error: variable i might already have been initialized]**. It is because variable ‘i’ is the final variable. And final variables are allowed to be initialized only once, and that was already done on line no 5.

**When can you use super keyword?**

* The super keyword is used to access hidden fields and overridden methods or attributes of the parent class.
* Following are the cases when this keyword can be used:
  + Accessing data members of parent class when the member names of the class and its child subclasses are same.
  + To call the default and parameterized constructor of the parent class inside the child class.
  + Accessing the parent class methods when the child classes have overridden them.
* The following example demonstrates all 3 cases when a super keyword is used.

class Parent{  
       protected int num = 1;  
              Parent(){           System.out.println("Parent class default constructor.");  
       }              Parent(String x){           System.out.println("Parent class parameterised constructor.");  
       }              public void foo(){  
           System.out.println("Parent class foo!");  
       }   }class Child extends Parent{  
       private int num = 2;  
              Child(){           //super constructor call should always be in the first line// super();              // Either call default super() to call default parent constructor ORsuper("Call Parent"); // call parameterised super to call parameterised parent constructor.  
 System.out.println("Child class default Constructor");  
 }  
   
 void printNum(){  
 System.out.println(num);  
 System.out.println(super.num); //prints the value of num of parent class  
 }  
   
 @Overridepublic void foo(){  
 System.out.println("Child class foo!");  
 super.foo(); //Calls foo method of Parent class inside the Overriden foo method of Child class.  
 }  
 }  
  
public class DemoClass {  
 public static void main(String args[]) {  
 Child demoObject=new Child();  
 demoObject.foo();  
 /\*  
 This would print -   
 Parent class parameterised constructor.  
 Child class default Constructor  
 Child class foo!  
 Parent class foo!  
 \*/  
 }  
}

**Can the static methods be overloaded?**

Yes! There can be two or more static methods in a class with the same name but differing input parameters.

**24. Why is the main method static in Java?**

The main method is always static because static members are those methods that belong to the classes, not to an individual object. So if the main method will not be static then for every object, It is available. And that is not acceptable by JVM. JVM calls the main method based on the class name itself. Not by creating the object.

Because there must be only 1 main method in the java program as the execution starts from the main method. So for this reason the main method is static.

**25. Can the static methods be overridden?**

* No! Declaration of static methods having the same signature can be done in the subclass but run time polymorphism can not take place in such cases.
* Overriding or dynamic polymorphism occurs during the runtime, but the static methods are loaded and looked up at the compile time statically. Hence, these methods cant be overridden.

**26. Difference between static methods, static variables, and static classes in java.**

* **Static Methods and Static variables** are those methods and variables that belong to the class of the java program, not to the object of the class. This gets memory where the class is loaded. And these can directly be called with the help of class names.
  + For example - We have used mathematical functions in the java program like - max(), min(), sqrt(), pow(), etc. And if we notice that, then we will find that we call it directly with the class name. Like - Math.max(), Math.min(), etc. So that is a static method.  And Similarly static variables we have used like (length) for the array to get the length. So that is the static method.
* **Static classes** - A class in the java program cannot be static except if it is the inner class. If it is an inner static class, then it exactly works like other static members of the class.

**What is the main objective of garbage collection?**

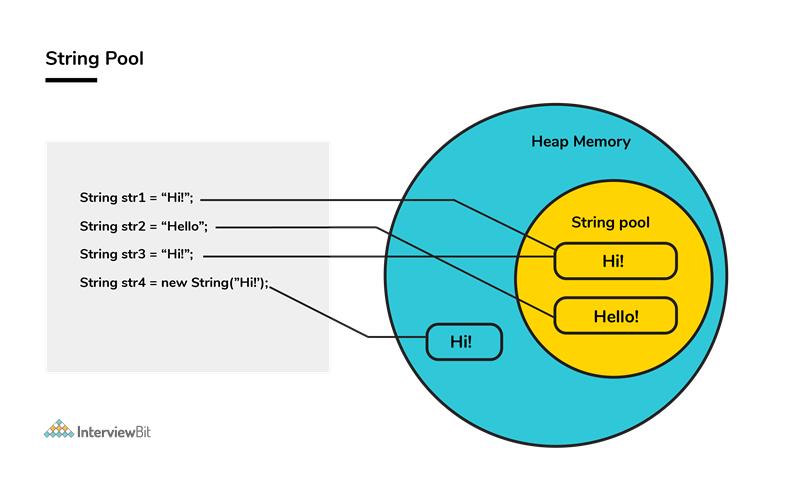
The main objective of this process is to free up the memory space occupied by the unnecessary and unreachable objects during the Java program execution by deleting those unreachable objects.

* This ensures that the memory resource is used efficiently, but it provides no guarantee that there would be sufficient memory for the program execution.

**Apart from the security aspect, what are the reasons behind making strings immutable in Java?**

A String is made immutable due to the following reasons:

* **String Pool:** Designers of Java were aware of the fact that String data type is going to be majorly used by the programmers and developers. Thus, they wanted optimization from the beginning. They came up with the notion of using the String pool (a storage area in Java heap) to store the String literals. They intended to decrease the temporary String object with the help of sharing. An immutable class is needed to facilitate sharing. The sharing of the mutable structures between two unknown parties is not possible. Thus, immutable Java String helps in executing the concept of String Pool.



* [**Multithreading**](https://www.interviewbit.com/multithreading-interview-questions/)**:** The safety of threads regarding the String objects is an important aspect in Java. No external synchronization is required if the String objects are immutable. Thus, a cleaner code can be written for sharing the String objects across different threads. The complex process of concurrency is facilitated by this method.
* [**Collections**](https://www.interviewbit.com/java-collections-interview-questions/)**:** In the case of Hashtables and HashMaps, keys are String objects. If the String objects are not immutable, then it can get modified during the period when it resides in the HashMaps. Consequently, the retrieval of the desired data is not possible. Such changing states pose a lot of risks. Therefore, it is quite safe to make the string immutable.

**How would you differentiate between a String, StringBuffer, and a StringBuilder?**

* **Storage area:** In string, the String pool serves as the storage area. For StringBuilder and StringBuffer, heap memory is the storage area.
* **Mutability:** A String is immutable, whereas both the StringBuilder and StringBuffer are mutable.
* **Efficiency:** It is quite slow to work with a String. However, StringBuilder is the fastest in performing operations. The speed of a StringBuffer is more than a String and less than a StringBuilder. (For example appending a character is fastest in StringBuilder and very slow in String because a new memory is required for the new String with appended character.)
* **Thread-safe:** In the case of a threaded environment, StringBuilder and StringBuffer are used whereas a String is not used. However, StringBuilder is suitable for an environment with a single thread, and a StringBuffer is suitable for multiple threads.  
  **Syntax:**

// StringString first = "InterviewBit";  
String second = new String("InterviewBit");// StringBufferStringBuffer third = new StringBuffer("InterviewBit");// StringBuilderStringBuilder fourth = new StringBuilder("InterviewBit");

[11:53 AM] thilagai, Deva (Cognizant)

**Using relevant properties highlight the differences between interfaces and abstract classes.**

* **Availability of methods:** Only abstract methods are available in interfaces, whereas non-abstract methods can be present along with abstract methods in abstract classes.
* **Variable types**: Static and final variables can only be declared in the case of interfaces, whereas abstract classes can also have non-static and non-final variables.
* **Inheritance:** Multiple inheritances are facilitated by interfaces, whereas abstract classes do not promote multiple inheritances.
* **Data member accessibility:** By default, the class data members of interfaces are of the public- type. Conversely, the class members for an abstract class can be protected or private also.
* **Implementation:** With the help of an abstract class, the implementation of an interface is easily possible. However, the converse is not true;

**Abstract class example:**

public abstract class Athlete {public abstract void walk();  
}

**Interface example:**

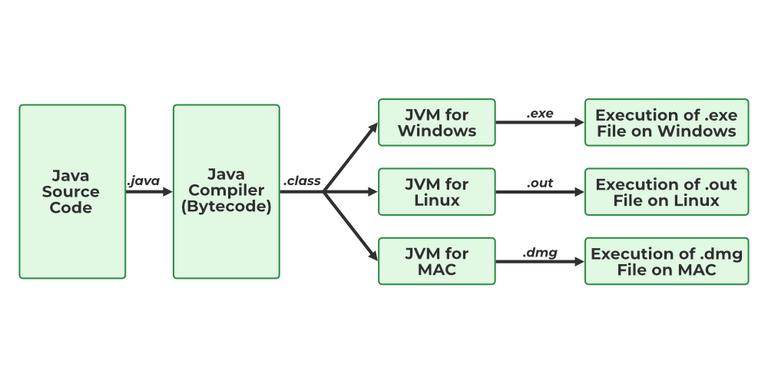
public interface Walkable {void walk();  
}

[12:31 PM] thilagai, Deva (Cognizant)

**What are the differences between JVM, JRE and JDK in Java?**

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **JDK** | **JRE** | **JVM** |
| **Abbreviation** | Java Development Kit | Java Runtime Environment | Java Virtual Machine |
| **Definition** | JDK is a complete software development kit for developing Java applications. It comprises JRE, JavaDoc, compiler, debuggers, etc. | JRE is a software package providing Java class libraries, JVM and all the required components to run the Java applications. | JVM is a platform-dependent, abstract machine comprising of 3 specifications - document describing the JVM implementation requirements, computer program meeting the JVM requirements and instance object for executing the Java byte code and provide the runtime environment for execution. |
| **Main Purpose** | JDK is mainly used for code development and execution. | JRE is mainly used for environment creation to execute the code. | JVM provides specifications for all the implementations to JRE. |
| **Tools provided** | JDK provides tools like compiler, debuggers, etc for code development | JRE provides libraries and classes required by JVM to run the program. | JVM does not include any tools, but instead, it provides the specification for implementation. |
| **Summary** | JDK = (JRE) + Development tools | JRE = (JVM) + Libraries to execute the application | JVM = Runtime environment to execute Java byte code. |

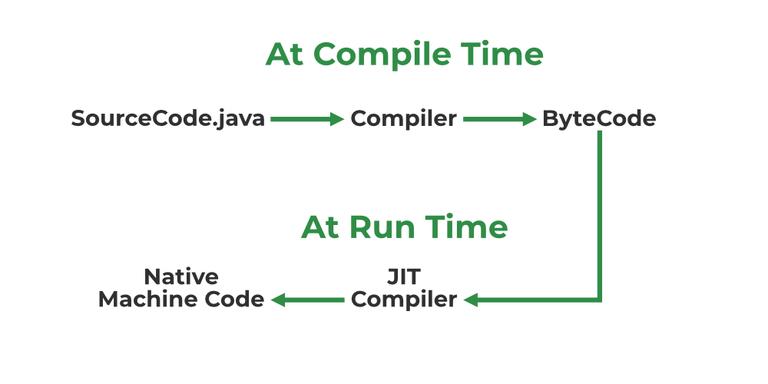
**What is JVM?**



JVM stands for Java Virtual Machine it is a Java interpreter. It is responsible for loading, verifying, and executing the bytecode created in Java.

Although it is platform dependent which means the software of JVM is different for different Operating Systems it plays a vital role in making Java platform Independent.

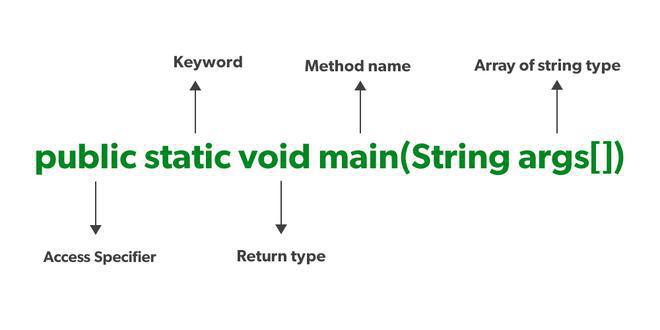
**4. What is JIT?**



JIT stands for (Just-in-Time) compiler is a part of JRE(Java Runtime Environment), it is used for better performance of the Java applications during run-time. The use of JIT is mentioned in step by step process mentioned below:

1. Source code is compiled with **javac** compiler to form bytecode
2. Bytecode is further passed on to JVM
3. JIT is a part of JVM, JIT is responsible for compiling bytecode into native machine code at run time.
4. The JIT compiler is enabled throughout, while it gets activated when a method is invoked. For a compiled method, the JVM directly calls the compiled code, instead of interpreting it.
5. As JVM calls the compiled code that increases the performance and speed of the execution.

**Explain public static void main(String args[]) in Java.**



Unlike any other programming language like C, C++, etc. In Java, we declared the main function as a public static void main (String args[]). The meanings of the terms are mentioned below:

1. **public**: the public is the access modifier responsible for mentioning who can access the element or the method and what is the limit.  It is responsible for making the main function globally available. It is made public so that JVM can invoke it from outside the class as it is not present in the current class.
2. **static**: static is a keyword used so that we can use the element without initiating the class so to avoid the unnecessary allocation of the memory.
3. **void**: void is a keyword and is used to specify that a method doesn’t return anything. As the main function doesn’t return anything we use void.
4. **main**: main represents that the function declared is the main function. It helps JVM to identify that the declared function is the main function.
5. **String args[]**: It stores Java command-line arguments and is an array of type java.lang.String class.

**What will happen if we declare don’t declare the main as static?**

We can declare the main method without using static and without getting any errors. But, the main method will not be treated as the entry point to the application or the program.

**What are Packages in Java?**

Packages in Java can be defined as the grouping of related types of classes, interfaces, etc providing access to protection and namespace management.

**13. Why Packages are used?**

Packages are used in Java in order to prevent naming conflicts, control access, and make searching/locating and usage of classes, interfaces, etc easier.

**14. What are the advantages of Packages in Java?**

There are various advantages of defining packages in Java.

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

**15. How many types of packages are there in Java?**

There are two types of packages in Java

* User-defined packages
* Build In packages

**16. Explain different data types in Java.**

There are 2 types of data types in Java as mentioned below:

1. Primitive Data Type
2. Non-Primitive Data Type or Object Data type

Primitive Data Type: Primitive data are single values with no special capabilities. There are 8 primitive data types:

* **boolean**: stores value true or false
* **byte**: stores an 8-bit signed two’s complement integer
* **char**: stores a single 16-bit Unicode character
* **short**: stores a 16-bit signed two’s complement integer
* **int**: stores a 32-bit signed two’s complement integer
* **long**: stores a 64-bit two’s complement integer
* **float**: stores a single-precision 32-bit IEEE 754 floating-point
* **double**: stores a double-precision 64-bit IEEE 754 floating-point

Non-Primitive Data Type: Reference Data types will contain a memory address of the variable’s values because it is not able to directly store the values in the memory. Types of Non-Primitive are mentioned below:

* Strings
* Array
* Class
* Object
* Interface

### Can we declare Pointer in Java?

No, Java doesn’t provide the support of Pointer. As Java needed to be more secure because which feature of the pointer is not provided in Java.

[12:34 PM] thilagai, Deva (Cognizant)

### What is the default value of byte datatype in Java?

The default value of the byte datatype in Java is 0.

### 

### 20. What is the default value of float and double datatype in Java?

The default value of the float is 0.0f and of double is 0.0d in Java.

**What is the Wrapper class in Java?**

Wrapper, in general, is referred to a larger entity that encapsulates a smaller entity. Here in Java, the wrapper class is an object class that encapsulates the primitive data types.

The primitive data types are the ones from which further data types could be created. For example, integers can further lead to the construction of long, byte, short, etc. On the other hand, the string cannot, hence it is not primitive.

Getting back to the wrapper class, Java contains 8 wrapper classes. They are Boolean, Byte, Short, Integer, Character, Long, Float, and Double. Further, custom wrapper classes can also be created in Java which is similar to the concept of Structure in the C programming language. We create our own wrapper class with the required data types.

**22. Why do we need wrapper classes?**

The wrapper class is an object class that encapsulates the primitive data types, and we need them for the following reasons:

1. Wrapper classes are final and immutable
2. Provides methods like valueOf(), parseInt(), etc.
3. It provides the feature of autoboxing and unboxing.

**23. Differentiate between instance and local variables.**

|  |  |
| --- | --- |
| **Instance Variable** | **Local Variable** |
| Declared outside the method, directly invoked by the method. | Declared within the method. |
| Has a default value. | No default value |
| It can be used throughout the class. | The scope is limited to the method. |

**24. What are the default values assigned to variables and instances in Java?**

In Java When we haven’t initialized the instance variables then the compiler initializes them with default values. The default values for instances and variables depend on their data types. Some common types of default data types are:

* The default value for numeric types (byte, short, int, long, float, and double) is 0.
* The default value for the boolean type is false.
* The default value for object types (classes, interfaces, and arrays) is null.
* The null character, “u0000, ” is the default value for the char type.

**Example:**

Java

// Java Program to demonstrate use of default valuesimport java.io.\*;class GFG {    // static values    static byte b;    static int i;    static long l;    static short s;    static boolean bool;    static char c;    static String str;    static Object object;    static float f;    static double d;    static int[] Arr;    public static void main(String[] args)    {        // byte value        System.out.println("byte value" + b);        // short value        System.out.println("short value" + s);        // int value        System.out.println("int value" + i);        // long value        System.out.println("long value" + l);        System.out.println("boolean value" + bool);        System.out.println("char value" + c);        System.out.println("float value" + f);        System.out.println("double value" + d);        System.out.println("string value" + str);        System.out.println("object value" + object);        System.out.println("Array value" + Arr);    }}

**Output**

byte value0  
short value0  
int value0  
long value0  
boolean valuefalse  
char value  
float value0.0  
double value0.0  
string valuenull  
object valuenull  
Array valuenull

### What is a Class Variable?

In Java, a class variable (also known as a static variable) is a variable that is declared within a class but outside of any method, constructor, or block. Class variables are declared with the static keyword, and they are shared by all instances (objects) of the class as well as by the class itself. No matter how many objects are derived from a class, each class variable would only exist once.

**Example:**

Java

// Java program to demonstrate use of Clas Variableclass GFG {    public static int ctr = 0;    public GFG() { ctr++; }    public static void main(String[] args)    {        GFG obj1 = new GFG();        GFG obj2 = new GFG();        GFG obj3 = new GFG();        System.out.println("Number of objects created are "                           + GFG.ctr);    }}

**Output**

Number of objects created are 3

### 

### 26. What is the default value stored in Local Variables?

There is no default value stored with local variables. Also, primitive variables and objects don’t have any default values.

### 

### 27. Explain the difference between instance variable and a class variable.

**Instance Variable:** A class variable without a static modifier known as an instance variable is typically shared by all instances of the class. These variables can have distinct values among several objects. The contents of an instance variable are completely independent of one object instance from another because they are related to a specific object instance of the class.

**Example:**

Java

// Java Program to demonstrate Instance Variableimport java.io.\*;class GFG {    private String name;    public void setName(String name) { this.name = name; }    public String getName() { return name; }    public static void main(String[] args)    {        GFG obj = new GFG();        obj.setName("John");        System.out.println("Name " + obj.getName());    }}

**Output**

Name John

**Class Variable:** Class Variable variable can be declared anywhere at the class level using the keyword static. These variables can only have one value when applied to various objects. These variables can be shared by all class members since they are not connected to any specific object of the class.

**Example:**

Java

// Java Program to demonstrate Class Variableimport java.io.\*;class GFG {    // class variable    private static final double PI = 3.14159;    private double radius;    public GFG(double radius) { this.radius = radius; }    public double getArea() { return PI \* radius \* radius; }    public static void main(String[] args)    {        GFG obj = new GFG(5.0);        System.out.println("Area of circle: "                           + obj.getArea());    }}

**Output**

Area of circle: 78.53975

### 28. What is a static variable?

The static keyword is used to share the same variable or method of a given class. Static variables are the variables that once declared then a single copy of the variable is created and shared among all objects at the class level.

[12:37 PM] thilagai, Deva (Cognizant)

**What is the difference between System.out, System.err, and System.in?**

**System.out –** It is a PrintStream that is used for writing characters or can be said it can output the data we want to write on the Command Line Interface console/terminal.

**Example:**

Java

// Java Program to implement// System.outimport java.io.\*;// Driver Classclass GFG {    // Main Function    public static void main(String[] args)    {        // Use of System.out        System.out.println("");    }}

**System.err –** It is used to display error messages.

**Example:**

Java

// Java program to demonstrate// System.errimport java.io.\*;// Driver Classclass GFG {    // Main function    public static void main(String[] args)    {        // Printing error        System.err.println(            "This is how we throw error with System.err");    }}

**Output:**

This is how we throw error with System.err

Although, System.err have many similarities both of them have quite a lot of difference also, let us check them.

| **System.out** | **System.err** |
| --- | --- |
| It will print to the standard out of the system. | It will print to the standard error. |
| It is mostly used to display results on the console. | It is mostly used to output error texts. |
| It gives output on the console with the default(black) color. | It also gives output on the console but most of the IDEs give it a red color to differentiate. |

**System.in –** It is an InputStream used to read input from the terminal Window. We can’t use the System.in directly so we use Scanner class for taking input with the system.in.

**Example:**

Java

// Java Program to demonstrate// System.inimport java.util.\*;// Driver Classclass Main {    // Main Function    public static void main(String[] args)    {        // Scanner class with System.in        Scanner sc = new Scanner(System.in);        // Taking input from the user        int x = sc.nextInt();        int y = sc.nextInt();        // Printing the output        System.out.printf("Addition: %d", x + y);    }}

**Output:**

3  
4  
Addition: 7

[12:37 PM] thilagai, Deva (Cognizant)

### What are the FileInputStream and FileOutputStream?

To read and write data, Java offers I/O Streams. A Stream represents an input source or an output destination, which could be a file, an i/o device, another program, etc. [**FileInputStream**](https://www.geeksforgeeks.org/java-io-fileinputstream-class-java/) in Java is used to read data from a file as a stream of bytes. It is mostly used for reading binary data such as images, audio files, or serialized objects.

**Example:**

File file = new File("path\_of\_the\_file");  
FileInputStream inputStream = new FileInputStream(file);

In Java, the [**FileOutputStream**](https://www.geeksforgeeks.org/fileoutputstream-in-java/) function is used to write data byte by byte into a given file or file descriptor. Usually, raw byte data, such as pictures, is written into a file using FileOutputStream.

**Example:**

File file = new File("path\_of\_the\_file");  
FileOutputStream outputStream = new FileOutputStream(file);

### 

### 34. What is the purpose of using BufferedInputStream and BufferedOutputStream classes?

When we are working with the files or stream then to increase the Input/Output performance of the program we need to use the BufferedInputStream and BufferedOutputStream classes. These both classes provide the capability of buffering which means that the data will be stored in a buffer before writing to a file or reading it from a stream. It also reduces the number of times our OS needs to interact with the network or the disk. Buffering allows programs to write a big amount of data instead of writing it in small chunks. This also reduces the overhead of accessing the network or the disk.

BufferedInputStream(InputStream inp);  
// used to create the bufferinput stream and save the arguments.

BufferedOutputStream(OutputStream output);  
// used to create a new buffer with the default size.

**How many ways you can take input from the console?**

There are two methods to take input from the console in Java mentioned below:

1. Using Command line argument
2. Using Buffered Reader Class
3. Using Console Class
4. Using Scanner Class

The program demonstrating the use of each method is given below.

**Example:**

Java

// Java Program to implement input// using Command line argumentimport java.io.\*;class GFG {    public static void main(String[] args)    {        // check if length of args array is        // greater than 0        if (args.length &gt; 0) {            System.out.println(                "The command line arguments are:");            // iterating the args array and printing            // the command line arguments            for (String val : args)                System.out.println(val);        }        else            System.out.println("No command line "                               + "arguments found.");    }}// Use below commands to run the code// javac GFG.java// java Main GeeksforGeeks

Java

// Java Program to implement// Buffer Reader Classimport java.io.\*;class GFG {    public static void main(String[] args)        throws IOException    {        // Enter data using BufferReader        BufferedReader read = new BufferedReader(            new InputStreamReader(System.in));        // Reading data using readLine        String x = read.readLine();        // Printing the read line        System.out.println(x);    }}

Java

// Java program to implement input// Using Console Classpublic class GfG {    public static void main(String[] args)    {        // Using Console to input data from user        String x = System.console().readLine();        System.out.println("You entered string " + x);    }}

Java

// Java program to demonstrate// working of Scanner in Javaimport java.util.Scanner;class GfG {    public static void main(String args[])    {        // Using Scanner for Getting Input from User        Scanner in = new Scanner(System.in);        String str = in.nextLine();        System.out.println("You entered string " + str);    }}

**Output:**

GeeksforGeeks

**38. Difference in the use of print, println, and printf.**

print, println, and printf all are used for printing the elements but print prints all the elements and the cursor remains in the same line. println shifts the cursor to next line. And with printf we can use format identifiers too.

[12:38 PM] thilagai, Deva (Cognizant)

**What are operators?**

Operators are the special types of symbols used for performing some operations over variables and values.

**40. How many types of operators are available in Java?**

All types of operators in Java are mentioned below:

1. [Arithmetic Operators](https://www.geeksforgeeks.org/java-arithmetic-operators-with-examples/)
2. [Unary Operators](https://www.geeksforgeeks.org/java-unary-operator-with-examples/)
3. [Assignment Operator](https://www.geeksforgeeks.org/java-assignment-operator-with-examples/)
4. [Relational Operators](https://www.geeksforgeeks.org/java-relational-operators-with-examples/)
5. [Logical Operators](https://www.geeksforgeeks.org/java-logical-operators-with-examples/)
6. [Ternary Operator](https://www.geeksforgeeks.org/java-ternary-operator-with-examples/)
7. [Bitwise Operators](https://www.geeksforgeeks.org/operators-in-java/)
8. [Shift Operators](https://www.geeksforgeeks.org/operators-in-java/)
9. [instance of operator](https://www.geeksforgeeks.org/java-instanceof-and-its-applications/)

Postfix operators are considered as the highest precedence according to Java operator precedence.

[12:38 PM] thilagai, Deva (Cognizant)

**What is dot operator?**

The Dot operator in Java is used to access the instance variables and methods of class objects. It is also used to access classes and sub-packages from the package.

[12:39 PM] thilagai, Deva (Cognizant)

**What’s the difference between the methods sleep() and wait()?**

|  |  |
| --- | --- |
| **Sleep()** | **Wait()** |
| The sleep() method belongs to the thread class. | Wait() method belongs to the object class. |
| Sleep does not release the lock that the current thread holds. | wait() release the lock which allows other threads to acquire it. |
| This method is a static method. | This method is not a static method. |
| Sleep() does not throw an InterruptedException. | InterruptedException is shown if the thread is interrupted while waiting. |
| Mainly used to delay a thread for some specific time duration. | Mainly used to pause a thread until notified by another thread. |
| Sleep() Has Two Overloaded Methods:   * sleep(long millis)millis: milliseconds * sleep(long millis, int nanos) nanos: Nanoseconds | Wait() Has Three Overloaded Methods:   * wait() * wait(long timeout) * wait(long timeout, int nanos) |

**47. What are the differences between String and StringBuffer?**

|  |  |
| --- | --- |
| **String** | **StringBuffer** |
| Store of a sequence of characters. | Provides functionality to work with the strings. |
| It is immutable. | It is mutable (can be modified and other string operations could be performed on them.) |
| No thread operations in a string. | It is thread-safe (two threads can’t call the methods of StringBuffer simultaneously) |

**48. What are the differences between StringBuffer and StringBuilder?**

|  |  |
| --- | --- |
| **StringBuffer** | **StringBuilder** |
| StringBuffer provides functionality to work with the strings. | StringBuilder is a class used to build a mutable string. |
| It is thread-safe (two threads can’t call the methods of StringBuffer simultaneously) | It is not thread-safe (two threads can call the methods concurrently) |
| Comparatively slow as it is synchronized. | Being non-synchronized, implementation is faster |

**49. Which among String or String Buffer should be preferred when there are a lot of updates required to be done in the data?**

The string is preferred over StringBuffer as StringBuilder is faster than StringBuffer, but StringBuffer objects are the preferred over as it provides more thread safety.

**50. Why is StringBuffer called mutable?**

StringBuffer class in Java is used to represent a changeable string of characters. It offers an alternative to the immutable String class by enabling you to change a string’s contents without constantly creating new objects. Mutable (modifiable) strings are created with the help of the StringBuffer class. The StringBuffer class in Java is identical to the String class except that it is changeable.

**Example:**

Java

// Java Program to demonstrate use of stringbufferpublic class StringBufferExample {    public static void main(String[] args)    {        StringBuffer s = new StringBuffer();        s.append("Geeks");        s.append("for");        s.append("Geeks");        String message = s.toString();        System.out.println(message);    }}

**Output**

GeeksforGeeks

[12:40 PM] thilagai, Deva (Cognizant)

**What is an array in Java?**

An Array in Java is a data structure that is used to store a fixed-size sequence of elements of the same type. Elements of an array can be accessed by their index, which starts from 0 and goes up to a length of minus 1. Array declaration in Java is done with the help of square brackets and size is also specified during the declaration.

**Syntax:**

int[] Arr = new int[5];

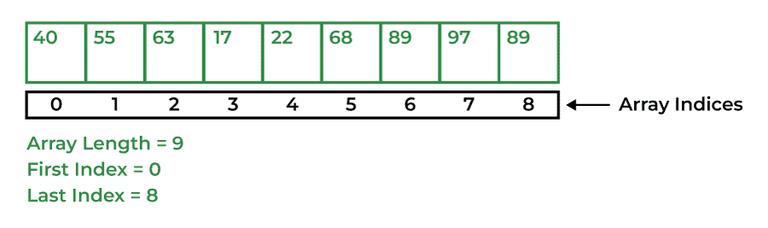
**53. On which memory arrays are created in Java?**

Arrays in Java are created in heap memory. When an array is created with the help of a new keyword, memory is allocated in the heap to store the elements of the array. In Java, the heap memory is managed by the Java Virtual Machine(JVM) and it is also shared between all threads of the Java Program. The memory which is no longer in use by the program, JVM uses a garbage collector to reclaim the memory. Arrays in Java are created dynamically which means the size of the array is determined during the runtime of the program. The size of the array is specified during the declaration of the array and it cannot be changed once the array is created.

**54. What are the types of an array?**

There are two types of arrays i.e., Primitive arrays and References Arrays.

* **Single-Dimensional Arrays:** Arrays that have only one dimension i.e., an array of integers or an array of strings are known as single-dimensional arrays.



**Syntax:**

data\_type[] Array\_Name = new data\_type[ArraySize];

* **Multi-Dimensional Arrays:** Arrays that have two or more dimensions such as two-dimensional or three-dimensional arrays.

**55. Why does the Java array index start with 0?**

The index of an array signifies the distance from the start of the array. So, the first element has 0 distance therefore the starting index is 0.

**Syntax:**

[Base Address + (index \* no\_of\_bytes)]

**56. What is the difference between int array[] and int[] array?**

Both int array[] and int[] array are used to declare an array of integers in java. The only difference between them is on their syntax no functionality difference is present between them.

int arr[] is a C-Style syntax to declare an Array.

int[] arr is a Java-Style syntax to declare an Array.

However, it is generally recommended to use Java-style syntax to declare an Array. As it is easy to read and understand also it is more consistent with other Java language constructs.

[12:40 PM] thilagai, Deva (Cognizant)

**How to copy an array in Java?**

In Java there are multiple ways to copy an Array based on the requirements.

* **clone() method in Java:** This method in Java is used to create a shallow copy of the given array which means that the new array will share the same memory as the original array.

int[] Arr = { 1, 2, 3, 5, 0};  
int[] tempArr = Arr.clone();

* **arraycopy() method:** To create a deep copy of the array we can use this method which creates a new array with the same values as the original array.

int[] Arr = {1, 2, 7, 9, 8};  
int[] tempArr = new int[Arr.length];  
System.arraycopy(Arr, 0, tempArr, 0, Arr.length);

* **copyOf() method:** This method is used to create a new array with a specific length and copies the contents of the original array to the new array.

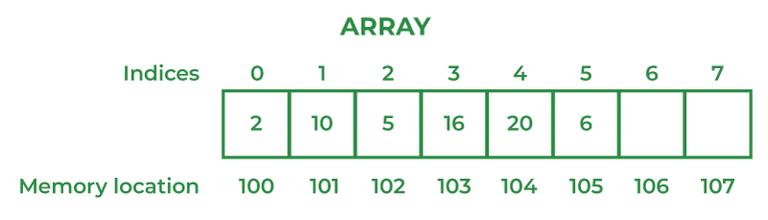
int[] Arr = {1, 2, 4, 8};  
int[] tempArr = Arrays.copyOf(Arr, Arr.length);

* **copyOfRange() method:** This method is very similar to the copyOf() method in Java, but this method also allows us to specify the range of the elements to copy from the original array.

int[] Arr = {1, 2, 4, 8};  
int[] temArr = Arrays.copyOfRange(Arr, 0, Arr.length);

[12:41 PM] thilagai, Deva (Cognizant)

**What are the advantages and disadvantages of an array?**



**The advantages of Arrays are:**

* Direct and effective access to any element in the collection is made possible by arrays. An array’s elements can be accessed using an O(1) operation, which means that the amount of time needed to do so is constant and independent of the array’s size.
* Data can be stored effectively in memory using arrays. The size of an array is known at compile time since its elements are stored in contiguous memory regions.
* Due to the fact that the data is stored in contiguous memory areas, arrays provide quick data retrieval.
* Arrays are easy to implement and understand, making them an ideal choice for beginners learning computer programming.

**Disadvantages of Arrays are:**

* Arrays are created with a predetermined size that is chosen at that moment. This means that if the array’s size needs to be extended, a new array will need to be made, and the data will need to be copied from the old array to the new array, which can take a lot of time and memory.
* There may be unused memory space in an array’s memory space if the array is not completely occupied. If you have poor recall, this can be a problem.
* Compared to other data structures like linked lists and trees, arrays might be rigid due to their fixed size and limited support for sophisticated data types.
* Because an array’s elements must all be of the same data type, it does not support complex data types like objects and structures.

[12:41 PM] thilagai, Deva (Cognizant)

**What are the main concepts of OOPs in Java?**

The main concepts of OOPs in Java are mentioned below:

* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

**63. What is the difference between an object-oriented programming language and an object-based programming language?**

| **Object-Oriented Programming Language** | **Object-Based Programming Language** |
| --- | --- |
| Object-oriented programming language covers larger concepts like inheritance, polymorphism, abstraction, etc. | The scope of object-based programming is limited to the usage of objects and encapsulation. |
| It supports all the built-in objects | It doesn’t support all the built-in objects |
| Examples: Java, C#, etc. | Examples: Java script, visual basics, etc. |

[2:17 PM] thilagai, Deva (Cognizant)

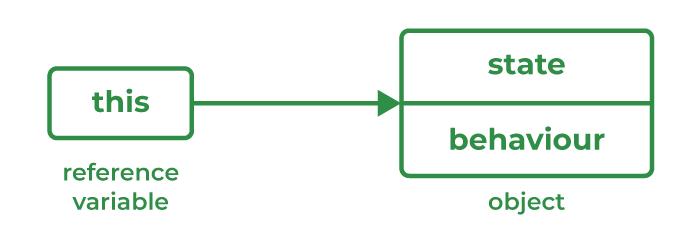
**What are Classes in Java?**

In Java, Classes are the collection of objects sharing similar characteristics and attributes. Classes represent the blueprint or template from which objects are created.  Classes are not real-world entities but help us to create objects which are real-world entities.

**66. What is the difference between static (class) method and instance method?**

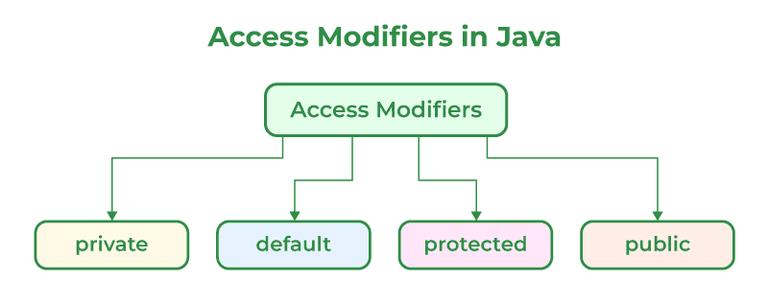
| **Static(Class) method** | **Instance method** |
| --- | --- |
| Static method is associated with a class rather than an object. | The instance method is associated with an object rather than a class. |
| Static methods can be called using the class name only without creating an instance of a class. | The instance method can be called on a specific instance of a class using the object reference. |
| Static methods do not have access to **this** keyword**.** | Instance methods have access to **this** keyword**.** |
| This method can access only static members of the class | This method can access both static and non-static methods of the class. |

**67. What is this keyword in Java?**



‘this’ is a keyword used to reference a variable that refers to the current object.

**68. What are Brief Access Specifiers and Types of Access Specifiers?**



Access Specifiers in Java help to restrict the scope of a class, constructor, variable, method, or data member. There are four types of Access Specifiers in Java mentioned below:

1. Public
2. Private
3. Protected
4. Default

**69. What will be the initial value of an object reference which is defined as an instance variable?**

The initial value of an object reference which is defined as an instance variable is a NULL value.

**70. What is an object?**

The object is a real-life entity that has certain properties and methods associated with it. The object is also defined as the instance of a class. An object can be declared using a new keyword.

[2:18 PM] thilagai, Deva (Cognizant)

**What is the constructor?**

Constructor is a special method that is used to initialize objects. Constructor is called when a object is created. The name of constructor is same as of the class.

**Example:**

// Class Created  
class XYZ{  
      private int val;  
        
      // Constructor  
      XYZ(){  
            val=0;  
      }  
};

**75. What happens if you don’t provide a constructor in a class?**

If you don’t provide a constructor in a class in Java, the compiler automatically generates a default constructor with no arguments and no operation which is a default constructor.

**76. How many types of constructors are used in Java?**

There are two types of constructors in Java as mentioned below:

1. Default Constructor
2. Parameterized Constructor

Default Constructor: It is the type that does not accept any parameter value. It is used to set initial values for object attributes.

class\_Name();  
// Default constructor called

Parameterized Constructor: It is the type of constructor that accepts parameters as arguments. These are used to assign values to instance variables during the initialization of objects.

class\_Name(parameter1, parameter2......);  
// All the values passed as parameter will be  
// allocated accordingly

**77. What is the purpose of a default constructor?**

Constructors help to create instances of a class or can be said to create objects of a class. Constructor is called during the initialization of objects. A default constructor is a type of constructor which do not accept any parameter, So whatever value is assigned to properties of the objects are considered default values.

[2:18 PM] thilagai, Deva (Cognizant)

**What are the differences between the constructors and methods?**

Java constructors are used for initializing objects. During creation, constructors are called to set attributes for objects apart from this few basic differences between them are:

1. Constructors are only called when the object is created but other methods can be called multiple times during the life of an object.
2. Constructors do not return anything, whereas other methods can return anything.
3. Constructors are used to setting up the initial state but methods are used to perform specific actions.

**81. What is an Interface?**

An interface in Java is a collection of static final variables and abstract methods that define the contract or agreement for a set of linked classes. Any class that implements an interface is required to implement a specific set of methods. It specifies the behavior that a class must exhibit but not the specifics of how it should be implemented.

**Syntax:**

interface   
{  
    // constant fields  
    // methds that are abstract by default  
}

**Example:**

Java

// Java Program to demonstrate Interfaceimport java.io.\*;interface Shape {    double getArea();    double getPerimeter();}class Circle implements Shape {    private double radius;    public Circle(double radius) { this.radius = radius; }    public double getArea()    {        return Math.PI \* radius \* radius;    }    public double getPerimeter()    {        return 2 \* Math.PI \* radius;    }}class GFG {    public static void main(String[] args)    {        Circle circle = new Circle(5.0);        System.out.println("Area of circle is "                           + circle.getArea());        System.out.println("Perimeter of circle is"                           + circle.getPerimeter());    }}

**Output**

Area of circle is 78.53981633974483  
Perimeter of circle is31.41592653589793

**82. Give some features of the Interface.**

An Interface in Java programming language is defined as an abstract type used to specify the behavior of a class. An interface in Java is a blueprint of a behavior. A Java interface contains static constants and abstract methods.

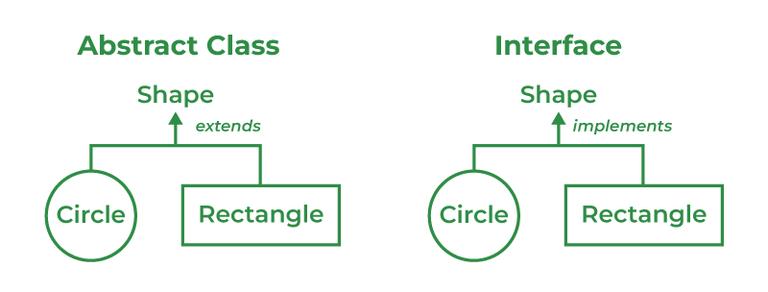
Features of the Interface are mentioned below:

* The interface can help to achieve total abstraction.
* Allows us to use multiple inheritances in Java.
* Any class can implement multiple interfaces even when one class can extend only one class.
* It is also used to achieve loose coupling.

**83. What is a marker interface?**

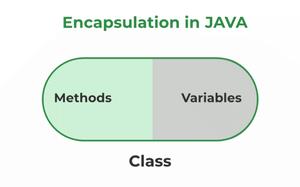
An Interface is recognized as an empty interface (no field or methods) it is called a marker interface. Examples of marker interfaces are Serializable, Cloneable, and Remote interfaces.

**84. What are the differences between abstract class and interface?**



| **Abstract Class** | **Interface Class** |
| --- | --- |
| Both abstract and non-abstract methods may be found in an abstract class. | The interface contains only abstract methods. |
| Abstract Class supports Final methods. | The interface class does not support Final methods. |
| Multiple inheritance is not supported by the Abstract class. | Multiple inheritances is supported by Interface Class. |
| Abstract Keyword is used to declare Abstract class. | Interface Keyword is used to declare the interface class. |
| **extend** keyword is used to extend an Abstract Class. | **implements** Keyword is used to implement the interface. |
| Abstract Class has members like protected, private, etc. | All class members are public by default. |

**85. What do you mean by data encapsulation?**



Data Encapsulation is the concept of OOPS properties and characteristics of the classes that The interface is binded together. Basically, it bundles data and methods that operate on that data within a single unit. Encapsulation is achieved by declaring the instance variables of a class as private, which means they can only be accessed within the class.

**86. What are the advantages of Encapsulation in Java?**

The advantages of Encapsulation in Java are mentioned below:

1. Data Hiding:  it is a way of restricting the access of our data members by hiding the implementation details. Encapsulation also provides a way for data hiding. The user will have no idea about the inner implementation of the class.
2. Increased Flexibility: We can make the variables of the class read-only or write-only depending on our requirements.
3. Reusability: Encapsulation also improves the re-usability and is easy to change with new requirements.
4. Testing code is easy: Code is made easy to test for unit testing.

**87. What is the primary benefit of Encapsulation?**

The main advantage of Encapsulation in Java is its ability to protect the internal state of an object from external modification or access. It is the is a way of hiding the implementation details of a class from outside access and only exposing a public interface that can be used to interact with the class. The main benefit is of providing a way to control and manage the state and the behavior of an object and also protecting it from modification and unauthorized access at the same time.

**Example:**

Java

// Java Program to demonstrate use of Encapsulationimport java.io.\*;class Person {    private String Name;    private int age;    public String getName() { return Name; }    public void setName(String Name) { this.Name = Name; }    public int getAge() { return age; }    public void setAge(int age) { this.age = age; }}// Driver classclass GFG {    // main function    public static void main(String[] args)    {        Person p = new Person();        p.setName("Rohan");        p.setAge(29);        System.out.println("Name is " + p.getName());        System.out.println("Age is " + p.getAge());    }}

**Output**

Name is Rohan  
Age is 29

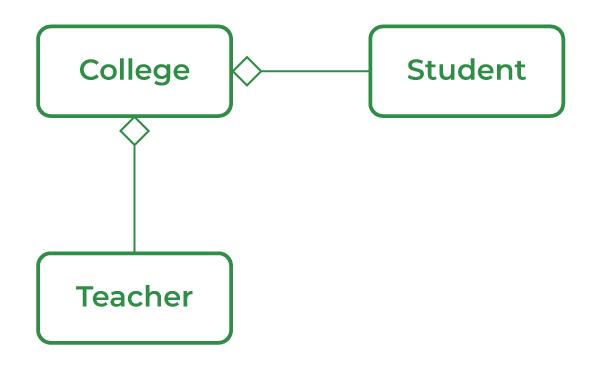
[2:20 PM] thilagai, Deva (Cognizant)

**when to use Abrstract class and interface?**

[2:20 PM] thilagai, Deva (Cognizant)

Do the test of "IS A" or "HAS A" relationship

[2:21 PM] thilagai, Deva (Cognizant)



Aggregation is a term related to the relationship between two classes best described as a “has-a” relationship. This kind is the most specialized version of association. It is a unidirectional association means it is a one-way relationship. It contains the reference to another class and is said to have ownership of that class.

### 

### 89. What is the ‘IS-A ‘ relationship in OOPs Java?

‘IS-A’ is a type of relationship in OOPs Java where one class inherits another class.

[2:21 PM] thilagai, Deva (Cognizant)

**What are the different types of inheritance in Java?**

Inheritance is the method by which the Child class can inherit the features of the Super or Parent class. In Java, Inheritance is of four types:

* **Single Inheritance:** When a child or subclass extends only one superclass, it is known to be single inheritance. Single-parent class properties are passed down to the child class.
* **Multilevel Inheritance:** When a child or subclass extends any other subclass a hierarchy of inheritance is created which is known as multilevel inheritance. In other words, one subclass becomes the parent class of another.
* **Hierarchical Inheritance:** When multiple subclasses derive from the same parent class is known as Hierarchical Inheritance. In other words, a class that has a single parent has many subclasses.
* **Multiple Inheritance:** When a child class inherits from multiple parent classes is known as Multiple Inheritance. In Java, it only supports multiple inheritance of interfaces, not classes.

**92. What is multiple inheritance? Is it supported by Java?**

A component of the object-oriented notion known as multiple inheritances allows a class to inherit properties from many parent classes. When methods with the same signature are present in both superclasses and subclasses, an issue arises. The method’s caller cannot specify to the compiler which class method should be called or even which class method should be given precedence.

***Note:*** *Java doesn’t support Multiple Inheritance*

**Example:**

Java

// Java Program to show multiple Inheritanceimport java.io.\*;interface Animal {    void eat();}interface Mammal {    void drink();}class Dog implements Animal, Mammal {    public void eat() { System.out.println("Eating"); }    public void drink() { System.out.println("Drinking"); }    void bark() { System.out.println("Barking"); }}class GFG {    public static void main(String[] args)    {        Dog d = new Dog();        d.eat();        d.drink();        d.bark();    }}

**Output**

Eating  
Drinking  
Barking

### What is runtime polymorphism or dynamic method dispatch?

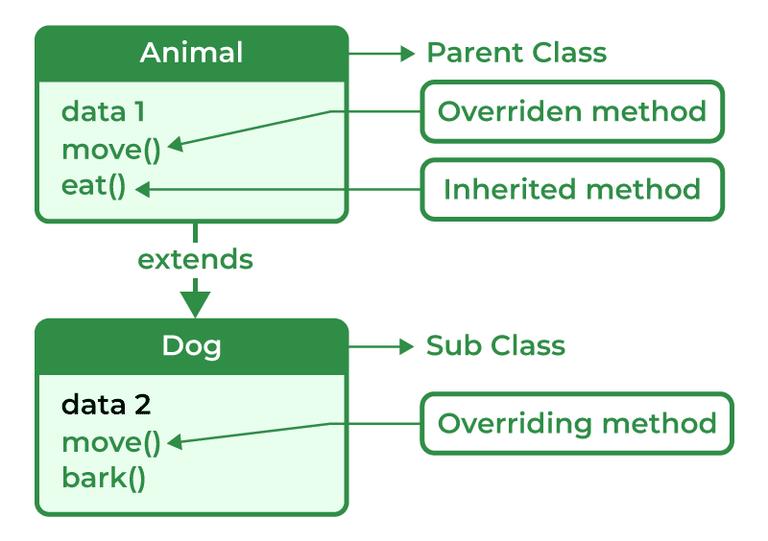
Dynamic method dispatch is a resolving mechanism for method overriding during the run time. Method overriding is the one where the method in a subclass has the same name, parameters, and return type as a method in the superclass. When the over-ridden method is called through a superclass reference, java determines which version (superclass or subclass) of that method is to be executed based upon the type of an object being referred to at the time the call occurs. Thus the decision is made at run time. This is referred to as dynamic method dispatch.

### 103. What is method overriding?

Method overriding, also known as run time polymorphism is one where the child class contains the same method as the parent class. For instance, we have a method named ‘gfg()’ in the parent class. A method gfg() is again defined in the sub-class. Thus when gfg() is called in the subclass, the method within the class id executed. Here, gfg() within the class overridden the method outside.

### 

### 104. What is method overloading?



Method overriding is a method to achieve Run-time polymorphism in Java. Method overriding is a feature that allows a child class to provide a specific implementation of a method that is already provided by one of its parent classes. When a method in a child class has the same name, the same parameters or signature, and the same return type(or sub-type) as a method in its parent class, then the method in the subclass is said to override the method in the superclass.

[2:23 PM] thilagai, Deva (Cognizant)

**What are method overloading and method overriding?**

**Method Overloading:** It is also known as Compile Time Polymorphism. In method overloading two or more methods are shared in the same class with a different signature.

**Example:**

Java

// Java Program to demonstrate use of Method Overloadingimport java.io.\*;class GFG {    static int multiply(int a, int b) { return a \* b; }    static int multiply(int a, int b, int c)    {        return a \* b \* c;    }    static int multiply(int a, int b, int c, int d)    {        return a \* b \* c \* d;    }    public static void main(String[] args)    {        System.out.println("multiply() with 2 parameters");        System.out.println(multiply(4, 5));        System.out.println("multiply() with 3 parameters");        System.out.println(multiply(2, 3, 4));        System.out.println("multiply() with 4 parameters");        System.out.println(multiply(2, 3, 4, 1));    }}

**Output**

multiply() with 2 parameters  
20  
multiply() with 3 parameters  
24  
multiply() with 4 parameters  
24

**Method Overriding:** Method Overriding occurs when a subclass can provide the implementation of a method which is already defined in the parent class or superclass. The return type, name and arguments must be similar to the methods in superclass.

**Example:**

Java

// Java Program to demonstrate use of Method Overridingimport java.io.\*;class Vehicle {    void drive()    {        System.out.println("drive() method of base class");        System.out.println("driving the Car.");    }}class Car extends Vehicle {    void drive()    {        System.out.println(            "drive() method of derived class");        System.out.println("Car is driving.");    }}class GFG {    public static void main(String[] args)    {        Car c1 = new Car();        Vehicle v1 = new Vehicle();        c1.drive();        v1.drive();        Vehicle vehicle = new Car();        // drive() method of Vehicle class is overridden by        // Car class drive()        vehicle.drive();    }}

**Output**

drive() method of derived class  
Car is driving.  
drive() method of base class  
driving the Car.  
drive() method of derived class  
Car is driving.

| **Method Overloading** | **Method Overriding** |
| --- | --- |
| When two or multiple methods are in the same class with different parameters but the same name. | When a subclass provides its own implementation of a method that is already defined in the parent class. |
| Method overloading can only happen in the same class or between a subclass or parent class. | Method overriding can only happen in Subclass. |
| When an error occurs it is caught at the compile time of the program. | When an error occurs it is caught at Runtime of the program. |
| Example of Compile Time Polymorphism. | Example of Run Time Polymorphism. |
| Method Overloading may or may not require Inheritance. | Method overriding always needs Inheritance. |
| It occurs within the class. | It is performed in two classes with an inheritance relationship. |

**What is Abstraction?**

Abstraction refers to the act of representing essential features without including background details. The detailed information or the implementation is hidden. The most common example of abstraction is a car, we know how to turn on the engine, accelerate and move, however, the way engine works, and its internal components are complex logic hidden from the general users. This is usually done to handle the complexity.

**114. What is Abstract class?**

A class declared as abstract, cannot be instantiated i.e., the object cannot be created. It may or may not contain abstract methods but if a class has at least one abstract method, it must be declared abstract.

***Example of an abstract class with abstract method:***

Java

// Java Program to implement// abstract methodimport java.io.\*;// Abstract classabstract class Fruits {    abstract void run();}// Driver Classclass Apple extends Fruits {    void run()    {        System.out.println("Abstract class example");    }    // main method    public static void main(String args[])    {        Fruits obj = new Apple();        obj.run();    }}

**When Abstract methods are used?**

An abstract method is used when we want to use a method but want to child classes to decide the implementation in that case we use Abstract methods with the parent classes.

[2:26 PM] thilagai, Deva (Cognizant)

**What is Collection Framework in Java?**

Collections are units of objects in Java. The collection framework is a set of interfaces and classes in Java that are used to represent and manipulate collections of objects in a variety of ways. The collection framework contains classes(ArrayList, Vector, LinkedList, PriorityQueue, TreeSet) and multiple interfaces (Set, List, Queue, Deque) where every interface is used to store a specific type of data.

**118. Explain various interfaces used in the Collection framework.**

Collection framework implements

1. Collection Interface
2. List Interface
3. Set Interface
4. Queue Interface
5. Deque Interface
6. Map Interface

[2:26 PM] thilagai, Deva (Cognizant)

**Collection interface:** Collection is the primary interface available that can be imported using java.util.Collection.

**Syntax:**

public interface Collection<E> extends iterable

[2:27 PM] thilagai, Deva (Cognizant)

**Why do we need a synchronized ArrayList when we have Vectors (which are synchronized) in Java?**

ArrayList is in need even when we have Vectors because of certain reasons:

1. ArrayList is faster than Vectors.
2. ArrayList supports multithreading whereas Vectors only supports single-thread use.
3. ArrayList is safer to use, as Vectors supports single threads and individual operations are less safe and take longer to synchronize.
4. Vectors are considered outdated in Java because of their synchronized nature.
5. [2:28 PM] thilagai, Deva (Cognizant)

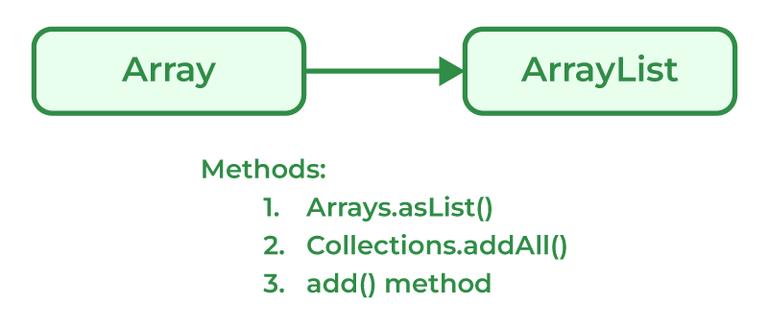
**Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.**

The elements of an array are stored in contiguous memory locations, which means that each element is stored in a separate block based on it located within the array. Since the elements of the array are stored in contiguous locations, it can be relatively easy to access any element by its index, as the element address can be calculated based on the location of the element. But Java implements ArrayLists as dynamic arrays, which means that the size can change as elements are removed or added. ArrayList elements are not stored in contiguous memory locations in order to accommodate this dynamic nature. Instead, the ArrayList makes use of a method known as an expandable array in which the underlying array is expanded to a larger size as needed and the elements are then copied to the new location. In contrast to an ArrayList, which has a dynamic size and does not store its elements in contiguous memory locations, an array has a fixed size and its elements are stored there.

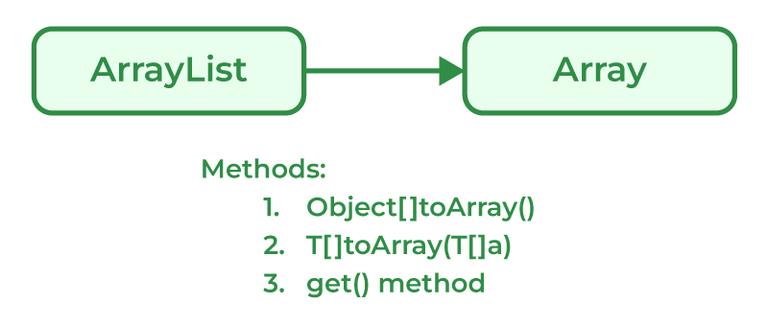
**123. Explain the method to convert ArrayList to Array and Array to ArrayList.**

### Conversion of List to ArrayList

There are multiple methods to convert List into ArrayList

1. 
2. Programmers can convert an Array to ArrayList using asList() method of the Arrays class. It is a static method of the Arrays class that accepts the List object.
3. **Syntax:**
4. Arrays.asList(item)
5. **Example:**
6. Java
8. // Java program to demonstrate conversion of// Array to ArrayList of fixed-size.import java.util.\*;// Driver Classclass GFG {    // Main Function    public static void main(String[] args)    {        String[] temp = { "Abc", "Def", "Ghi", "Jkl" };        // Conversion of array to ArrayList        // using Arrays.asList        List conv = Arrays.asList(temp);        System.out.println(conv);    }}
10. **Output**
11. [Abc, Def, Ghi, Jkl]

### Conversion of ArrayList to Array

1. 
2. Java programmers can convert ArrayList to
3. **Syntax:**
4. List\_object.toArray(new String[List\_object.size()])
5. **Example:**
6. Java
8. // Java program to demonstrate working of// Objectp[] toArray()import java.io.\*;import java.util.List;import java.util.ArrayList;// Driver Classclass GFG {    // Main Function    public static void main(String[] args)    {        // List declared        List&lt;Integer&gt;            arr = new ArrayList&lt;Integer&gt;();        arr.add(1);        arr.add(2);        arr.add(3);        arr.add(2);        arr.add(1);        // Conversion        Object[] objects = arr.toArray();        // Printing array of objects        for (Object obj : objects)            System.out.print(obj + " ");    }}
10. **Output**
11. 1 2 3 2 1
12. [2:28 PM] thilagai, Deva (Cognizant)

### How does the size of ArrayList grow dynamically? And also state how it is implemented internally.

Due to ArrayLists array-based nature, it grows dynamically in size ensuring that there is always enough room for elements. When an ArrayList element is first created, the default capacity is around 10-16 elements which basically depends on the Java version. ArrayList elements are copied over from the original array to the new array when the capacity of the original array is full. As the ArrayList size increases dynamically, the class creates a new array of bigger sizes and it copies all the elements from the old array to the new array. Now, the reference of the new array is used internally. This process of dynamically growing an array is known as resizing.

**Explain the LinkedList class.**

LinkedList class is Java that uses a doubly linked list to store elements. It inherits the AbstractList class and implements List and Deque interfaces. Properties of the LinkedList Class are mentioned below:

1. LinkedList classes are non-synchronized.
2. Maintains insertion order.
3. It can be used as a list, stack, or queue.

Syntax:

LinkedList<class> list\_name=new LinkedList<class>();

[2:29 PM] thilagai, Deva (Cognizant)

**What is Set in the Java Collections framework and list down its various implementations?**

Sets are collections that don’t store duplicate elements. They don’t keep any order of the elements. The Java Collections framework provides several implementations of the Set interface, including:

* **HashSet:** HashSet in Java, stores the elements in a has table which provides faster lookups and faster insertion. HashSet is not ordered.
* **LinkedHashSet:** LinkedHashSet is an implementation of HashSet which maintains the insertion order of the elements.
* **TreeSet:** TreeSet stores the elements in a sorted order that is determined by the natural ordering of the elements or by a custom comparator provided at the time of creation.

**131. What is the HashSet class in Java and how does it store elements?**

The HashSet class implements the Set interface in the Java Collections Framework and is a member of the HashSet class. Unlike duplicate values, it stores a collection of distinct elements. In this implementation, each element is mapped to an index in an array using a hash function, and the index is used to quickly access the element. It produces an index for the element in the array where it is stored based on the input element. Assuming the hash function distributes the elements among the buckets appropriately, the HashSet class provides constant-time performance for basic operations (add, remove, contain, and size).

**132. What is LinkedHashSet in Java Collections Framework?**

The LinkedHashSet is an ordered version of Hashset maintained by a doubly-linked List across all the elements. It is very helpful when iteration order is needed. During Iteration in LinkedHashSet, elements are returned in the same order they are inserted.

**Syntax:**

LinkedHashSet<E> hs = new LinkedHashSet<E>();

**Example:**

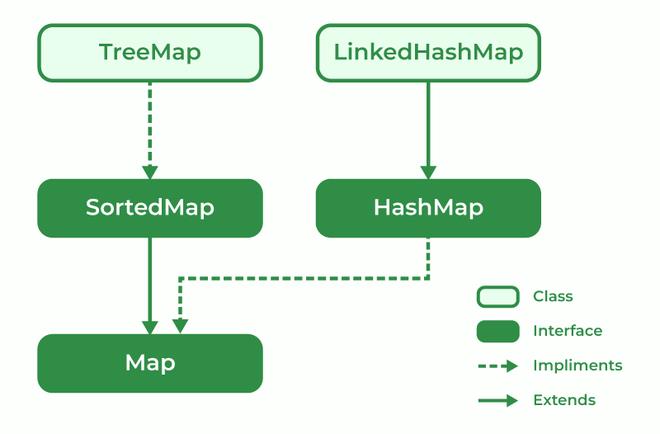
Java

// Java Program to implement// LinkedHashSetimport java.io.\*;import java.util.\*;// Driver Classclass GFG {    // Main Function    public static void main(String[] args)    {        // LinkedHashSet declared        LinkedHashSet&lt;Integer&gt;            hs = new LinkedHashSet&lt;Integer&gt;();        // Add elements in HashSet        hs.add(1);        hs.add(2);        hs.add(5);        hs.add(3);        // Print values        System.out.println("Values:" + hs);    }}

**Output**

Values:[1, 2, 5,

**133. What is a Map interface in Java?**



The map interface is present in the Java collection and can be used with Java.util package. A map interface is used for mapping values in the form of a key-value form. The map contains all unique keys. Also, it provides methods associated with it like containsKey(), contains value (), etc.

There are multiple types of maps in the map interface as mentioned below:

1. SortedMap
2. TreeMap
3. HashMap
4. LinkedHashMap

**134. Explain Treemap in Java**

TreeMap is a type of map that stores data in the form of key-value pair. It is implemented using the red-black tree. Features of TreeMap are :

1. It contains only unique elements.
2. It cannot have a NULL key
3. It can have multiple NULL values.
4. It is non-synchronized.
5. It maintains ascending order.
6. [2:31 PM] thilagai, Deva (Cognizant)
7. **What is an enumeration?**
8. Enumeration is a user-defined data type. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain. The main objective of the enum is to define user-defined data types.
9. **Example:**
10. // A simple enum example where enum is declared   
    // outside any class (Note enum keyword instead of   
    // class keyword)   
    enum Color   
    {   
        RED, GREEN, BLUE;   
    }
11. **141. What is the difference between Collection and Collections?**

|  |  |
| --- | --- |
| **Collection** | **Collections** |
| The Collection is an Interface. | Collections is a class. |
| It provides the standard functionality of data structure. | It is to sort and synchronize the collection elements. |
| It provides the methods that can be used for the data structure. | It provides static methods that can be used for various operations. |

1. **142. Differentiate between Array and ArrayList in Java.**

|  |  |
| --- | --- |
| **Array** | **ArrayList** |
| Single-dimensional or multidimensional | Single-dimensional |
| For and for each used for iteration | Here iterator is used to traverse riverArrayList |
| length keyword returns the size of the array. | size() method is used to compute the size of ArrayList. |
| The array has Fixed-size. | ArrayList size is dynamic and can be increased or decreased in size when required. |
| It is faster as above we see it of fixed size | It is relatively slower because of its dynamic nature |
| Primitive data types can be stored directly in unlikely objects. | Primitive data types are not directly added to unlikely arrays, they are added indirectly with help of autoboxing and unboxing |
| They can not be added here hence the type is in the unsafe. | They can be added here hence makingArrayList type-safe. |
| The assignment operator only serves the purpose | Here a special method is used known as add() method |

1. **143. What is the difference between Array and Collection in Java?**

|  |  |
| --- | --- |
| **Array** | **Collections** |
| Array in Java has a fixed size. | Collections in Java have dynamic sizes. |
| In an Array, Elements are stored in contiguous memory locations. | In Collections, Elements are not necessarily stored in contiguous memory locations. |
| Objects and primitive data types can be stored in an array. | We can only store objects in collections. |
| Manual manipulation is required for resizing the array. | Resizing in collections is handled automatically. |
| The array has basic methods for manipulation. | Collections have advanced methods for manipulation and iteration. |
| The array is available since the beginning of Java. | Collections were introduced in Java 1.2. |

2. **144. Difference between ArrayList and LinkedList.**

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| ArrayList is Implemented as an expandable Array. | LinkedList is Implemented as a doubly-linked list. |
| In ArrayList, Elements are stored in contiguous memory locations | LinkedList Elements are stored in non-contiguous memory locations as each element has a reference to the next and previous elements. |
| ArrayLists are faster for random access. | LinkedLists are faster for insertion and deletion operations |
| ArrayLists are more memory efficient. | LinkedList is less memory efficient |
| ArrayLists Use more memory due to maintaining the array size. | LinkedList Uses less memory as it only has references to elements |
| The search operation is faster in ArrayList. | The search operation is slower in LinkedList |

2. **145. Differentiate between ArrayList and Vector in Java.**

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| ArrayLists are implemented as an expandable array. | Vector is Implemented as a growable array. |
| ArrayList is not synchronized. | The vector is synchronized. |
| ArrayLists are Faster for non-concurrent operations. | Vector is Slower for non-concurrent operations due to added overhead of synchronization. |
| ArrayLists were Introduced in Java 1.2. | Vector was Introduced in JDK 1.0. |
| Recommended for use in a single-threaded environment. | Vectors are Recommended for use in a multi-threaded environment. |
| The default initial capacity of ArrayLists is 10. | In Vectors, the default initial capacity is 10 but the default increment is twice the size. |
| ArrayList performance is high. | Vector performance is low. |

2. **146. What is the difference between Iterator and ListIterator?**

|  |  |
| --- | --- |
| **Iterator** | **ListIterator** |
| Can traverse elements present in Collection only in the forward direction. | Can traverse elements present in Collection both in forward and backward directions. |
| Used to traverse Map, List, and Set. | Can only traverse List and not the other two. |
| Indexes can’t be obtained using Iterator | It has methods like nextIndex() and previousIndex() to obtain indexes of elements at any time while traversing the List. |
| Can’t modify or replace elements present in Collection | Can modify or replace elements with the help of set(E e) |
| Can’t add elements, and also throws ConcurrentModificationException. | Can easily add elements to a collection at any time. |
| Certain methods of Iterator are next(), remove(), and hasNext(). | Certain methods of ListIterator are next(), previous(), hasNext(), hasPrevious(), add(E e). |

2. **147. Differentiate between HashMap and HashTable.**

|  |  |
| --- | --- |
| **HashMap** | **HashTable** |
| HashMap is not synchronized | HashTable is synchronized |
| One key can be a NULL value | NULL values not allowed |
| The iterator is used to traverse HashMap. | Both Iterator and Enumertar can be used |
| HashMap is faster. | HashTable is slower as compared to HashMap. |

2. **148. What is the difference between Iterator and Enumeration?**

|  |  |
| --- | --- |
| **Iterator** | **Enumeration** |
| The Iterator can traverse both legacies as well as non-legacy elements. | Enumeration can traverse only legacy elements. |
| The Iterator is fail-fast. | Enumeration is not fail-fast. |
| The Iterators are slower. | Enumeration is faster. |
| The Iterator can perform a remove operation while traversing the collection. | The Enumeration can perform only traverse operations on the collection. |

2. **149. What is the difference between Comparable and Comparator?**

|  |  |
| --- | --- |
| **Comparable** | **Comparator** |
| The interface is present in java.lang package. | The Interface is present in java.util package. |
| Provides compareTo() method to sort elements. | Provides compare() method to sort elements. |
| It provides single sorting sequences. | It provides multiple sorting sequences. |
| The logic of sorting must be in the same class whose object you are going to sort. | The logic of sorting should be in a separate class to write different sorting based on different attributes of objects. |
| Method sorts the data according to fixed sorting order. | Method sorts the data according to the customized sorting order. |
| It affects the original class. | It doesn’t affect the original class. |
| Implemented frequently in the API by Calendar, Wrapper classes, Date, and String. | It is implemented to sort instances of third-party classes. |

2. **150. What is the difference between Set and Map?**

|  |  |
| --- | --- |
| **Set** | **Map** |
| The Set interface is implemented using java.util package. | The map is implemented using java.util package. |
| It can extend the collection interface. | It does not extend the collection interface. |
| It does not allow duplicate values. | It allows duplicate values. |
| The set can sort only one null value. | The map can sort multiple null values. |



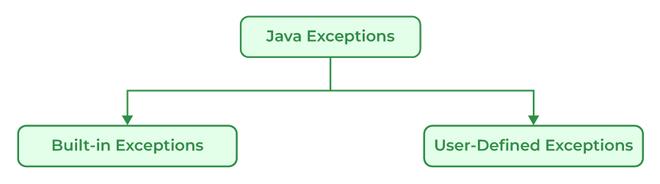
[2:31 PM] thilagai, Deva (Cognizant)

**What is Exception Handling?**

An [Exception](https://www.geeksforgeeks.org/exceptions-in-java/) is an Event that interrupts the normal flow of the program and requires special processing. During the execution of a program, errors and unplanned occurrences can be dealt with by using the Java Exception Handling mechanism. Below are some reasons why Exceptions occur in Java:

* Device failure
* Loss of Network Connection
* Code Errors
* Opening an Unavailable file
* Invalid User Input
* Physical Limitations (out of disk memory)

**153. How many types of exceptions can occur in a Java program?**



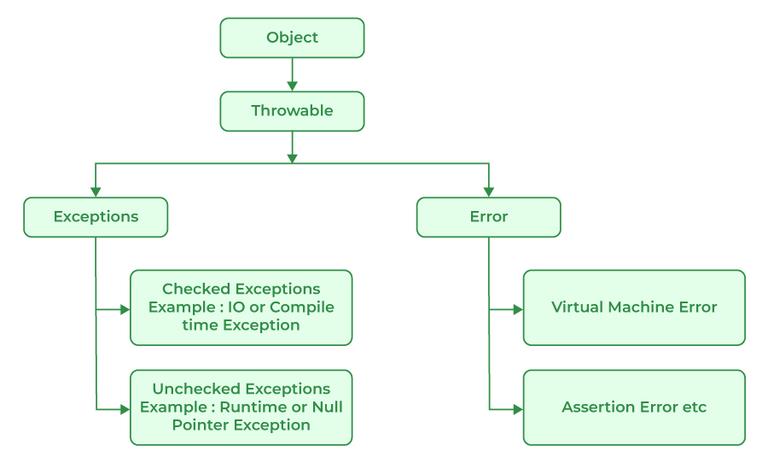
**There are generally two types of exceptions in Java:**

* **Built-in Exceptions:** Built-in exceptions in Java are provided by the Java Libraries. These exceptions can be further divided into two subcategories i.e., checked and unchecked Exceptions. Below are some of the built-in exceptions in Java:
  + ArrayIndexOutOfBoundsExceptions
  + ClassNotFoundException
  + FileNotFoundException
  + IOException
  + NullPointerException
  + ArithmeticException
  + InterruptedException
  + RuntimeException
* **User-Defined Exceptions:** User-defined exceptions are defined by the programmers themselves to handle some specific situations or errors which are not covered by built-in exceptions. To define user-defined exceptions a new class that extends the appropriate exception class must be defined. User-defined Exceptions in Java are used when the built-in exceptions are in Java.

**154. Difference between an Error and an Exception.**

|  |  |
| --- | --- |
| **Errors** | **Exceptions** |
| Recovering from Errors is not possible. | Recover from exceptions by either using a try-catch block or throwing exceptions back to the caller. |
| Errors are all unchecked types in Java. | It includes both checked as well as unchecked types that occur. |
| Errors are mostly caused by the environment in which the program is running. | The program is mostly responsible for causing exceptions. |
| Errors can occur at compile time as well as run time. Compile Time: Syntax Error, Run Time: Logical Error. | All exceptions occur at runtime but checked exceptions are known to the compiler while unchecked are not. |
| They are defined in java.lang.Error package. | They are defined in java.lang.Exception package |
| **Examples**: java.lang.StackOverflowError, java.lang.OutOfMemoryError | **Examples**: Checked Exceptions: SQLException, IOException Unchecked Exceptions: ArrayIndexOutOfBoundException, NullPointerException, ArithmeticException. |

**155. Explain the hierarchy of Java Exception classes.**



All exception and error types in Java are subclasses of the class throwable, which is the base class of the hierarchy. This class is then used for exceptional conditions that user programs should catch. NullPointerException is an example of such an exception. Another branch, error is used by the Java run-time system to indicate errors having to do with the JRE. StackOverflowError is an example of one of such error.

**156. Explain Runtime Exceptions.**

Runtime Exceptions are exceptions that occur during the execution of a code, as opposed to compile-time exceptions that occur during compilation. Runtime exceptions are unchecked exceptions, as they aren’t accounted for by the JVM.

**Examples of runtime exceptions in Java include:**

* NullPointerException: This occurs when an application attempts to use a null object reference.
* ArrayIndexOutOfBoundsException: This occurs when an application attempts to access an array index that is out of bounds.
* ArithmeticException: This occurs when an application attempts to divide by zero.
* IllegalArgumentException: This occurs when a method is passed on an illegal or inappropriate argument.

Unlike checked exceptions, runtime exceptions do not require a declaration in the throws clause or capture in a try-catch block. However, handling runtime exceptions is advisable in order to provide meaningful error messages and prevent a system crash. Because runtime exceptions provide more specific information about the problem than checked exceptions, they enable developers to detect and correct programming errors more easily and quickly.

**157. What is NullPointerException?**

It is a type of run-time exception that is thrown when the program attempts to use an object reference that has a null value. The main use of NullPointerException is to indicate that no value is assigned to a reference variable, also it is used for implementing data structures like linked lists and trees.

[2:32 PM] thilagai, Deva (Cognizant)

**What is the difference between Checked Exception and Unchecked Exception?**

**Checked Exception:**

Checked Exceptions are the exceptions that are checked during compile time of a program. In a program, if some code within a method throws a checked exception, then the method must either handle the exception or must specify the exception using the throws keyword.

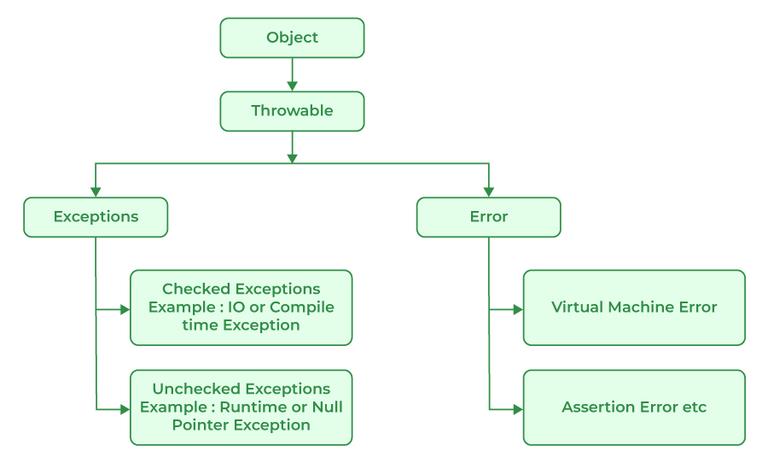
Checked exceptions are of two types:

* Fully checked exceptions: all its child classes are also checked, like IOException, and InterruptedException.
* Partially checked exceptions: some of its child classes are unchecked, like an Exception.

**Unchecked Exception:**

Unchecked are the exceptions that are not checked at compile time of a program. Exceptions under Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked.

**160. What is the base class for Error and Exception?**



Error is an illegal operation performed by the user which causes abnormality in the program. Exceptions are the unexpected events or conditions that comes while running the program, exception disrupts the normal flow of the program’s instructions.

Errors and Exceptions both have a common parent class which is java.lang.Throwable class.

[2:32 PM] thilagai, Deva (Cognizant)

**What will happen if you put System.exit(0) on the try or catch block? Will finally block execute?**

System.exit(int) has the capability to throw SecurityException. So, if in case of security, the exception is thrown then finally block will be executed otherwise JVM will be closed while calling System. exit(0) because of which finally block will not be executed.

[2:33 PM] thilagai, Deva (Cognizant)

**How do exceptions affect the program if it doesn’t handle them?**

Exceptions are responsible for abruptly terminating the running of the program while executing and the code written after the exception occurs is not executed.

**166. What is the use of the final keyword?**

The final keyword is used to make functions non-virtual. By default, all the functions are virtual so to make it non-virtual we use the final keyword.

**167. What purpose do the keywords final, finally, and finalize fulfill?**

**i). final:**

final is a keyword is used with the variable, method, or class so that they can’t be overridden.

**Example:**

Java

// Java Program to use final// keywordimport java.io.\*;// Driver Classclass GFG {    // Main function    public static void main(String[] args)    {        final int x = 100;        x = 50;    }}

**Output:**

./GFG.java:6: error: cannot assign a value to final variable x  
          x=50;  
          ^  
1 error

**ii). finally**

finally is a block of code used with “try-catch” in exception handling. Code written in finally block runs despite the fact exception is thrown or not.

**Example:**

Java

// Java Program to implement finallyimport java.io.\*;// Driver classclass GFG {    // Main function    public static void main(String[] args)    {        int x = 10;        // try block        try {            System.out.println("Try block");        }        // finally block        finally {            System.out.println(                "Always runs even without exceptions");        }    }}

**Output**

Try block  
Always runs even without exceptions

**iii). finalize**

It is a method that is called just before deleting/destructing the objects which are eligible for Garbage collection to perform clean-up activity.

Example:

Java

/\*package whatever // do not write package name here \*/import java.io.\*;class GFG {    public static void main(String[] args)    {        System.out.println("Main function running");        System.gc();    }    // Here overriding finalize method    public void finalize()    {        System.out.println("finalize method overridden");    }}

**Output**

Main function running

**168. What is the difference between this() and super() in Java?**

|  |  |
| --- | --- |
| **this( )** | **super( )** |
| It represents the current instance of the class. | It represents the current instance of the parent class. |
| Calls the default constructor of the same class. | Calls the default constructor of the base class. |
| Access the methods of the same class. | Access the methods of the parent class. |
| Points current class instance. | Points the superclass instance. |

**What is a thread?**

Threads in Java are subprocess with lightweight with the smallest unit of processes and also has separate paths of execution. These threads use shared memory but they act independently hence if there is an exception in threads that do not affect the working of other threads despite them sharing the same memory. A thread has its own program counter, execution stack, and local variables, but it shares the same memory space with other threads in the same process. Java provides built-in support for multithreading through the **Runnable interface** and the **Thread class**.

**174. Differentiate between process and thread?**

A process and a thread are both units of execution in a computer system, but they are different in several ways:

|  |  |
| --- | --- |
| **Process** | **Thread** |
| A process is a program in execution. | A thread is a single sequence of instructions within a process. |
| The process takes more time to terminate. | The thread takes less time to terminate. |
| The process takes more time for context switching. | The thread takes less time for context switching. |
| The process is less efficient in terms of communication. | Thread is more efficient in terms of communication. |
| The process is isolated. | Threads share memory. |
| The process has its own Process Control Block, Stack, and Address Space. | Thread has Parents’ PCB, its own Thread Control Block, and Stack and common Address space. |
| The process does not share data with each other. | Threads share data with each other. |

[2:45 PM] thilagai, Deva (Cognizant)

**What is the difference between the ‘throw’ and ‘throws’ keyword in java?**

* The ‘**throw**’ keyword is used to manually throw the exception to the calling method.
* And the ‘**throws**’ keyword is used in the function definition to inform the calling method that this method throws the exception. So if you are calling, then you have to handle the exception.

**Example -**

class Main {  
   public static int testExceptionDivide(int a, int b) throws ArithmeticException{  
       if(a == 0 || b == 0)  
           throw new ArithmeticException();  
       return a/b;  
   }   public static void main(String args[]) {  
       try{  
           testExceptionDivide(10, 0);  
 }  
 catch(ArithmeticException e){  
 //Handle the exception  
 }  
 }  
}

Here in the above snippet, the method testExceptionDivide throws an exception. So if the main method is calling it then it must have handled the exception. Otherwise, the main method can also throw the exception to JVM.

And the method testExceptionDivide 'throws’ the exception based on the condition.

**19. What are the differences between constructor and method of a class in Java?**

|  |  |
| --- | --- |
| **Constructor** | **Method** |
| Constructor is used for initializing the object state. | Method is used for exposing the object's behavior. |
| Constructor has no return type. | Method should have a return type. Even if it does not return anything, return type is void. |
| Constructor gets invoked implicitly. | Method has to be invoked on the object explicitly. |
| If the constructor is not defined, then a default constructor is provided by the java compiler. | If a method is not defined, then the compiler does not provide it. |
| The constructor name should be equal to the class name. | The name of the method can have any name or have a class name too. |
| A constructor cannot be marked as final because whenever a class is inherited, the constructors are not inherited. Hence, marking it final doesn't make sense. Java throws compilation error saying - modifier final not allowed here | A method can be defined as final but it cannot be overridden in its subclasses. |
| Final variable instantiations are possible inside a constructor and the scope of this applies to the whole class and its objects. | A final variable if initialised inside a method ensures that the variable cant be changed only within the scope of that method. |

[2:47 PM] thilagai, Deva (Cognizant)

**Java works as “pass by value” or “pass by reference” phenomenon?**

Java always works as a “pass by value”. There is nothing called a “pass by reference” in Java. However, when the object is passed in any method, the address of the value is passed due to the nature of object handling in Java. When an object is passed, a copy of the reference is created by Java and that is passed to the method. The objects point to the same memory location. 2 cases might happen inside the method:

* **Case 1:** When the object is pointed to another location: In this case, the changes made to that object do not get reflected the original object before it was passed to the method as the reference points to another location.

For example:

class InterviewBitTest{  
   int num;  
   InterviewBitTest(int x){   
       num = x;    }   InterviewBitTest(){        num = 0;   
   }  
}class Driver {  
   public static void main(String[] args){  
       //create a reference       InterviewBitTest ibTestObj = new InterviewBitTest(20);  
 //Pass the reference to updateObject Method  
 updateObject(ibTestObj);  
 //After the updateObject is executed, check for the value of num in the object.  
 System.out.println(ibTestObj.num);  
 }  
 public static void updateObject(InterviewBitTest ibObj){  
 // Point the object to new reference  
 ibObj = new InterviewBitTest();  
 // Update the value   
 ibObj.num = 50;  
 }  
}  
Output:  
20

* **Case 2:** When object references are not modified: In this case, since we have the copy of reference the main object pointing to the same memory location, any changes in the content of the object get reflected in the original object.

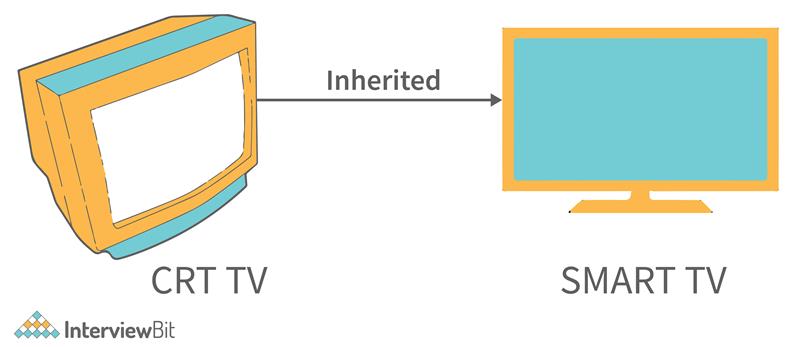
For example:

class InterviewBitTest{  
   int num;  
   InterviewBitTest(int x){   
       num = x;    }   InterviewBitTest(){        num = 0;   
   }  
}class Driver{  
   public static void main(String[] args){  
       //create a reference       InterviewBitTest ibTestObj = new InterviewBitTest(20);  
 //Pass the reference to updateObject Method  
 updateObject(ibTestObj);  
 //After the updateObject is executed, check for the value of num in the object.  
 System.out.println(ibTestObj.num);  
 }  
 public static void updateObject(InterviewBitTest ibObj){  
 // no changes are made to point the ibObj to new location// Update the value of num  
 ibObj.num = 50;  
 }  
}  
Output:  
50

**What is the ‘IS-A ‘ relationship in OOPs java?**

‘IS-A’ relationship is another name for inheritance. When we inherit the base class from the derived class, then it forms a relationship between the classes. So that relationship is termed an ‘IS-A’ Relationship.

**Example** - Consider a Television (Typical CRT TV). Now another Smart TV  that is inherited from television class. So we can say that the Smart iv is also a TV. Because CRT TV things can also be done in the Smart TV.



So here ‘IS-A’ Relationship formed. **[ SmartTV ‘IS-A’ TV ]**.

### ****Which among String or String Buffer should be preferred when there are lot of updates required to be done in the data?****

StringBuffer is mutable and dynamic in nature whereas String is immutable. Every updation / modification of String creates a new String thereby overloading the string pool with unnecessary objects. Hence, in the cases of a lot of updates, it is always preferred to use StringBuffer as it will reduce the overhead of the creation of multiple String objects in the string pool.

[2:50 PM] thilagai, Deva (Cognizant)

### ****Will the finally block get executed when the return statement is written at the end of try block and catch block as shown below?****

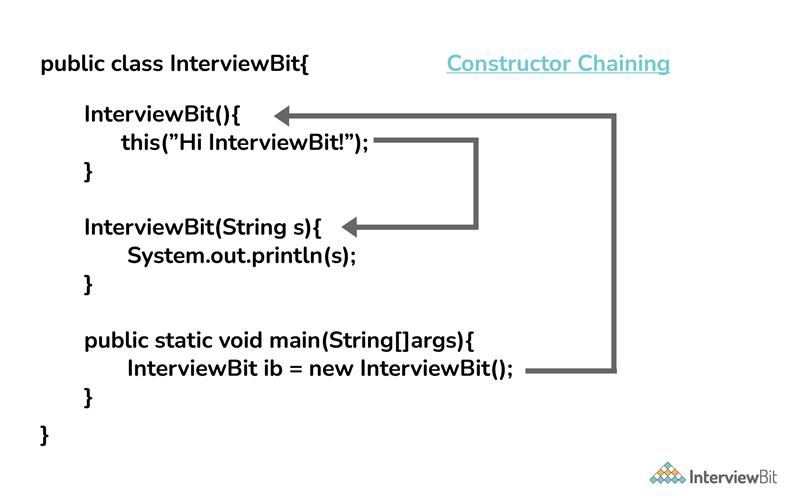
public int someMethod(int i){  
   try{  
       //some statementreturn 1;  
   }catch(Exception e){  
       //some statementreturn 999;  
 }finally{  
 //finally block statements  
 }  
}

finally block will be executed irrespective of the exception or not. The only case where finally block is not executed is when it encounters ‘System.exit()’ method anywhere in try/catch block.

[2:51 PM] thilagai, Deva (Cognizant)

### ****Can you call a constructor of a class inside the another constructor?****

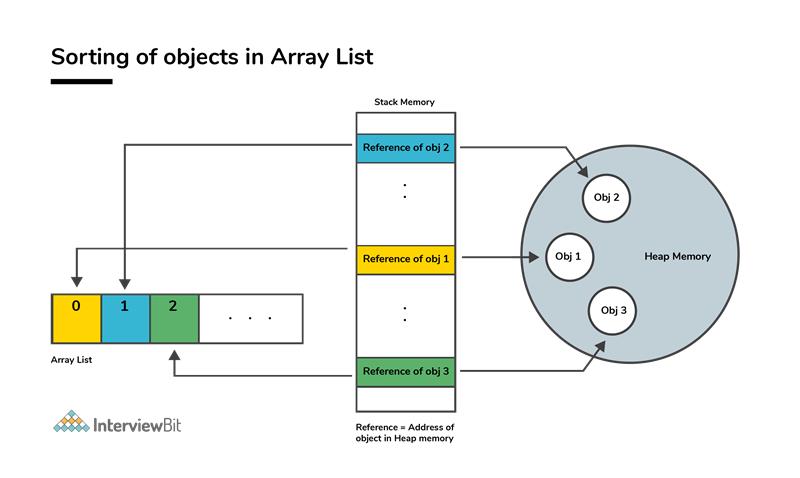
Yes, the concept can be termed as constructor chaining and can be achieved using this().



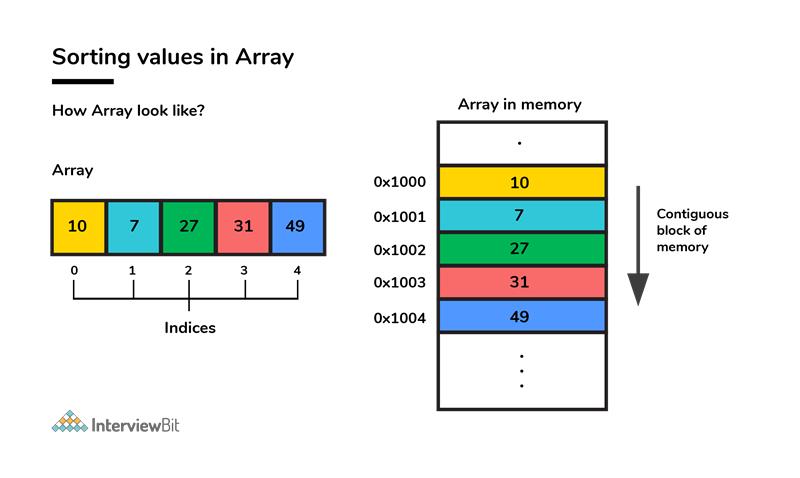
[2:52 PM] thilagai, Deva (Cognizant)

### ****Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.****

In the case of ArrayList, data storing in the form of primitive data types (like int, float, etc.) is not possible. The data members/objects present in the ArrayList have references to the objects which are located at various sites in the memory. Thus, storing of actual objects or non-primitive data types (like Integer, Double, etc.) takes place in various memory locations.



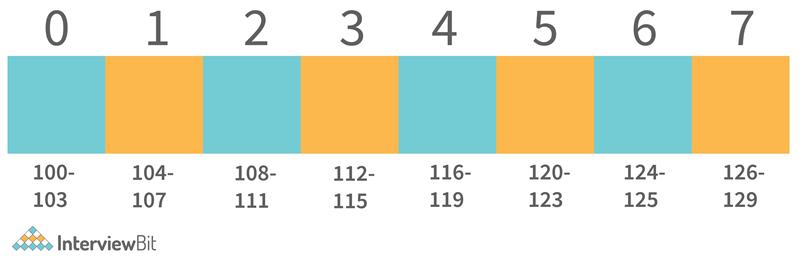
However, the same does not apply to the arrays. Object or primitive type values can be stored in arrays in contiguous memory locations, hence every element does not require any reference to the next element.



### ****Why does the java array index start with 0?****

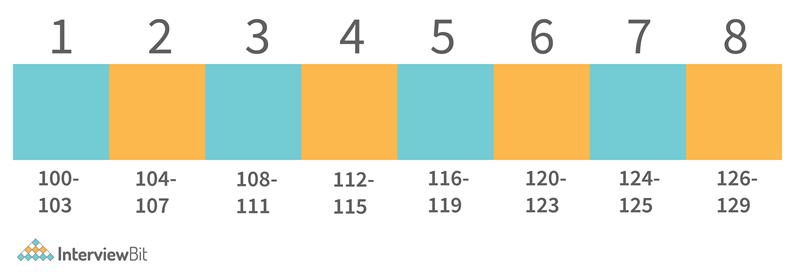
It is because the 0 index array avoids the extra arithmetic operation to calculate the memory address.

Example - Consider the array and assume each element takes 4-byte memory space. Then the address will be like this -



Now if we want to access index 4. Then internally java calculates the address using the formula-

**[Base Address + (index \* no\_of\_bytes)]**. So according to this. The starting address of the index 4 will be - **[100 + (4\*4)] = 116**. And exactly that's what the address is calculated.   
Now consider the same with 1 index Array -



Now if we apply the same formula here. Then we get - **116** as the starting address of the 4th index. Which is wrong. Then we need to apply formula - [**Base Address + ((index-1) \* no\_of\_bytes)]**.

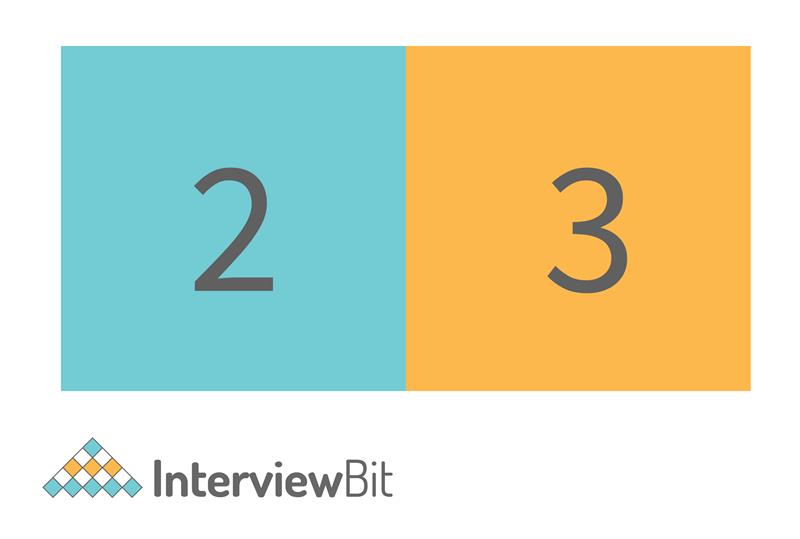
And for calculating this, an extra arithmetic operation has to be performed. And consider the case where millions of addresses need to be calculated, this causes complexity. So to avoid this, ) the index array is supported by java.

[2:54 PM] thilagai, Deva (Cognizant)

**How does the size of ArrayList grow dynamically? And also state how it is implemented internally.**

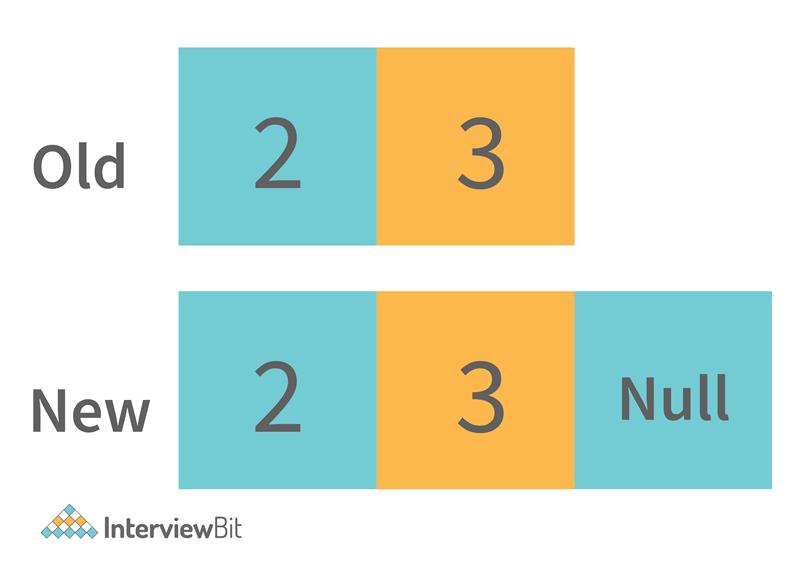
ArrayList is implemented in such a way that it can grow dynamically. We don't need to specify the size of ArrayList. For adding the values in it, the methodology it uses is -

1. Consider initially that there are 2 elements in the ArrayList. **[2, 3]**.

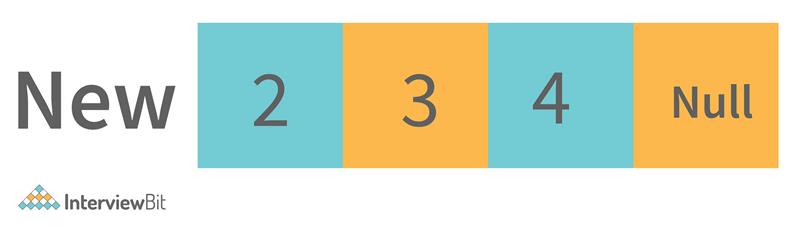


2. If we need to add the element into this. Then internally what will happen is-

* ArrayList will allocate the new ArrayList of Size (current size + half of the current size). And add the old elements into the new. Old - [2, 3],    New - [2, 3, null].



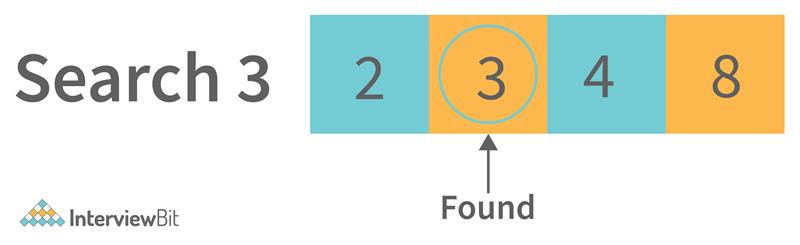
* Then the new value will be inserted into it. [2, 3, 4, null]. And for the next time, the extra space will be available for the value to be inserted.



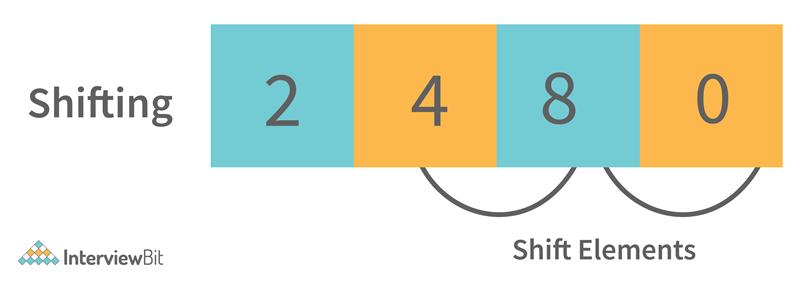
3. This process continues and the time taken to perform all of these is considered as the amortized constant time.

This is how the ArrayList grows dynamically. And when we delete any entry from the ArrayList then the following steps are performed -

1. It searches for the element index in the array. Searching takes some time. Typically it’s O(n) because it needs to search for the element in the entire array.



2. After searching the element, it needs to shift the element from the right side to fill the index.



So this is how the elements are deleted from the ArrayList internally. Similarly, the search operations are also implemented internally as defined in removing elements from the list (searching for elements to delete).

[2:55 PM] thilagai, Deva (Cognizant)

**Define System.out.println().**

**System.out.println()** is used to print the message on the console. **System** - It is a class present in **java.lang package**. Out is the static variable of type PrintStream class present in the **System class. println()** is the method present in the PrintStream class.

So if we justify the statement, then we can say that if we want to print anything on the console then we need to call the **println()** method that was present in PrintStream class. And we can call this using the output object that is present in the System class.

[2:57 PM] thilagai, Deva (Cognizant)

### ****1. Check if a given string is palindrome using recursion.****

/\*  
\* Java program to check if a given inputted string is palindrome or not using recursion.  
\*/import java.util.\*;public class InterviewBit {   
   public static void main(String args[]) {   
       Scanner s = new Scanner(System.in);  
       String word = s.nextLine();       System.out.println("Is "+word+" palindrome? - "+isWordPalindrome(word));  
   }          public static boolean isWordPalindrome(String word){   
       String reverseWord = getReverseWord(word);        //if word equals its reverse, then it is a palindromeif(word.equals(reverseWord)){   
           return true;   
       }        return false;   
 }   
   
 public static String getReverseWord(String word){   
 if(word == null || word.isEmpty()){   
 return word;   
 }   
   
 return word.charAt(word.length()- 1) + getReverseWord(word.substring(0, word.length() - 1));   
 }   
}

### ****2. Write a Java Program to print Fibonacci Series using Recursion.****

class InterviewBit {  
    public static void printFibonacci(int val\_1, int val\_2, int num){  
        //Base Caseif(num == 0)  
            return;  
  
        //Printing the next Fibonacci number            System.out.print( val\_1 + val\_2 + " ");  
  
        //Recursively calling for printing Fibonacci for remaining length        printFibonacci(val\_2, val\_1+val\_2, --num);    }    public static void main(String args[]) {  
        System.out.println(" \*\*\* Fibonacci Series \*\*\* ");  
  
        //Printing the first two values  
 System.out.print("0 1 ");  
  
 //Calling Method to print the fibonacci for length 10  
 printFibonacci(0, 1, 10);  
 }  
}

In the above code, we are printing the base 2 Fibonacci values 0 and 1. And then based on the length of Fibonacci to be printed, we are using the helper function to print that.

### ****3. Write a Java program to check if the two strings are anagrams.****

The main idea is to validate the length of strings and then if found equal, convert the string to char array and then sort the arrays and check if both are equal.

import java.util.Arrays;import java.util.Scanner;public class InterviewBit {  
 public static void main(String[] args) {  
   Scanner s = new Scanner(System.in);  
   //Input from two strings   System.out.print("First String: ");  
   String string1 = s.nextLine();   System.out.print("Second String: ");  
   String string2 = s.nextLine();   // check for the lengthif(string1.length() == string2.length()) {  
     // convert strings to char arraychar[] characterArray1 = string1.toCharArray();  
 char[] characterArray2 = string2.toCharArray();  
 // sort the arrays  
 Arrays.sort(characterArray1);  
 Arrays.sort(characterArray2);  
 // check for equality, if found equal then anagram, else not an anagramboolean isAnagram = Arrays.equals(characterArray1, characterArray2);  
 System.out.println("Anagram: "+ isAnagram);  
 }  
}

### ****4. Write a Java Program to find the factorial of a given number.****

public class FindFactorial {  
   public static void main(String[] args) {  
       int num = 10;  
       long factorialResult = 1l;  
       for(int i = 1; i <= num; ++i)  
 {  
 factorialResult \*= i;  
 }  
 System.out.println("Factorial: "+factorialResult);  
 }  
}

### ****5. Given an array of non-duplicating numbers from 1 to n where one number is missing, write an efficient java program to find that missing number.****

Idea is to find the sum of n natural numbers using the formula and then finding the sum of numbers in the given array. Subtracting these two sums results in the number that is the actual missing number. This results in O(n) time complexity and O(1) space complexity.

public class IBMissingNumberProblem {  
  
   public static void main(String[] args) {  
  
       int[] array={4,3,8,7,5,2,6};  
       int missingNumber = findMissingNum(array);  
       System.out.println("Missing Number is "+ missingNumber);   
   }  
   public static int findMissingNum(int[] array) {  
       int n=array.length+1;  
       int sumOfFirstNNums=n\*(n+1)/2;  
       int actualSumOfArr=0;  
 for (int i = 0; i < array.length; i++) {  
 actualSumOfArr+=array[i];  
 }  
 return sumOfFirstNNums-actualSumOfArr;  
 }  
}

[2:57 PM] thilagai, Deva (Cognizant)

### ****Write a Java program to create and throw custom exceptions.****

class InterviewBit {  
    public static void main(String args[]) throws CustomException {  
  
        // Throwing the custom exception be passing the messagethrow new CustomException(" This is my custom Exception ");  
    }  
}//Creating Custom Exception Classclass CustomException extends Exception{  
    //Defining Constructor to throw exception messagepublic CustomException(String message){  
 super(message);  
 }  
}

We have created the exception class named with CustomException and called the base exception constructor with the error message that we want to print. And to avoid handling exceptions in the main method, we have used the throws keyword in the method declaration.

### ****8. Write a Java program to reverse a string.****

class InterviewBit{  
    public static void main(String[] args){  
        //Input String        String str = "Welcome to InterviewBit";  
                //Pointers.int i = 0, j = str.length()-1;  
                //Result character array to store the reversed string.char[] revString = new char[j+1];  
   
 //Looping and reversing the string.while(i < j){  
 revString[j] = str.charAt(i);  
 revString[i] = str.charAt(j);  
 i++;  
 j--;  
 }  
 //Printing the reversed String.  
 System.out.println("Reversed String = " + String.valueOf(revString));  
 }  
}

In the above code, we are storing the last character from the string to the first and the first value to the last in the output character array. And doing the same thing in the loop for the remaining 2nd to n-1 characters. This is how the string will be reversed.

[3:02 PM] thilagai, Deva (Cognizant)

### What is an array?

An array is a collection of elements of the same data type. It can be used to store data in a more organized and efficient way.

### 18. What are control structures in Java?

Control structures are statements that are used to control the flow of a program. They include if-else statements, switch statements, loops (for, while, do-while), and break and continue statements.

[3:02 PM] thilagai, Deva (Cognizant)

### What is a lambda expression in Java? Provide an example.

A lambda expression in Java is a function that can be created without belonging to any class. An example is (x, y) -> x + y, which takes two integer parameters and returns their sum.

### 32. What is the difference between a private and a protected method in Java?

A private method is only accessible within the same class, while a protected method is accessible within the same class and any subclass.

### What is the difference between a while loop and a do-while loop in Java?

A while loop executes the loop body if the condition is true, while a do-while loop executes the loop body at least once before checking the condition.

### Define Wrapper Classes in Java.

In Java, when you declare primitive datatypes, then Wrapper classes are responsible for converting them into objects(Reference types).

### Define package in Java.

The package is a collective bundle of classes and interfaces and the necessary libraries and JAR files. The use of packages helps in code reusability.

[3:06 PM] thilagai, Deva (Cognizant)

### Write a Java Program to print Fibonacci Series using Recursion.

class FibonacciExample2{

 static int n1=0,n2=1,n3=0;

 static void printFibonacci(int count){

    if(count>0){

         n3 = n1 + n2;

         n1 = n2;

         n2 = n3;

         System.out.print(" "+n3);

         printFibonacci(count-1);

     }

 }

 public static void main(String args[]){

  int count=10;

  System.out.print(n1+" "+n2);//printing 0 and 1

  printFibonacci(count-2);//n-2 because 2 numbers are already printed

 }

 }

### 66. Write a Java program to check if the two strings are anagrams.

import java.util.Arrays;

public class AnagramString {

    static void isAnagram(String str1, String str2) {

        String s1 = str1.replaceAll("\\s", "");

        String s2 = str2.replaceAll("\\s", "");

        boolean status = true;

        if (s1.length() != s2.length()) {

            status = false;

        } else {

            char[] ArrayS1 = s1.toLowerCase().toCharArray();

            char[] ArrayS2 = s2.toLowerCase().toCharArray();

            Arrays.sort(ArrayS1);

            Arrays.sort(ArrayS2);

            status = Arrays.equals(ArrayS1, ArrayS2);

        }

        if (status) {

            System.out.println(s1 + " and " + s2 + " are anagrams");

        } else {

            System.out.println(s1 + " and " + s2 + " are not anagrams");

        }

    }

    public static void main(String[] args) {

        isAnagram("Keep", "Peek");

        isAnagram("Mother In Law", "Hitler Woman");

    }

}

Output

Keep and Peek are anagrams

MotherInLaw and HitlerWoman are anagrams

### 67. Write a Java Program to find the factorial of a given number.

4! = 4\*3\*2\*1 = 24

5! = 5\*4\*3\*2\*1 = 120

### 68. Given an array of non-duplicating numbers from 1 to n where one number is missing, write an efficient java program to find that missing number.

Input: arr[] = {1, 2, 4, 6, 3, 7, 8}, N = 8

             Output: 5

            Explanation: The missing number between 1 to 8 is 5

### Write a Java program to create and throw custom exceptions.

// class that uses custom exception InvalidAgeException

public class TestCustomException1

{

    // method to check the age

    static void validate (int age) throws InvalidAgeException{

       if(age < 18){

        // throw an object of user defined exception

        throw new InvalidAgeException("age is not valid to vote");

    }

       else {

        System.out.println("welcome to vote");

        }

     }

    // main method

    public static void main(String args[])

    {

        try

        {

            // calling the method

            validate(13);

        }

        catch (InvalidAgeException ex)

        {

            System.out.println("Caught the exception");

            // printing the message from InvalidAgeException object

            System.out.println("Exception occured: " + ex);

        }

        System.out.println("rest of the code...");

    }

}

**Write a java program to check if any number given as input is the sum of 2 prime numbers.**

// C program to check if a prime number

// can be expressed as sum of

// two Prime Numbers

#include <stdio.h>

#include <math.h>

#include <stdbool.h>

// Function to check whether a number

// is prime or not

bool isPrime(int n)

{

            if (n <= 1)

                           return false;

            for (int i = 2; i <= sqrt(n); i++)

            {

                           if (n % i == 0)

                                          return false;

            }

            return true;

}

// Function to check if a prime number

// can be expressed as sum of

// two Prime Numbers

bool isPossible(int N)

{

            // if the number is prime,

            // and number-2 is also prime

            if (isPrime(N) && isPrime(N - 2))

                           return true;

            else

                           return false;

}

// Driver code

int main()

{

            int n = 13;

            if (isPossible(n))

                           printf("%s", "Yes");

            else

                           printf("%s", "No");

            return 0;

}

**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

[3:10 PM] thilagai, Deva (Cognizant)

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**.

**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**.

[3:10 PM] thilagai, Deva (Cognizant)

### 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.

[3:11 PM] thilagai, Deva (Cognizant)

### 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).

