How Java Internally works?

Compilation and Interpretation. ( .Java get the class file and .class get the result). Compilation checks the Syntax, Rules then Translate the Class file. If the any error found that called compile time error.

Interpretation have JDK, Javac, JRE, JVM(load the file), Jit and Execute, Read, and translate after that we get result. If the any error found that called run time error.

How HashMap Internally Works?

21. What is Data Type ?

Data type specifies the size and type of values that can be stored in an identifier (variable).

It’s also called the space in the memory.

There are two type:-

1. Primitive data type.

2. Non-Primitive data type/Reference data type.

(a)Primitive datatype: Primitive data is a set of data having predefined characterize. Java developers build eight different data types that mostly we used. Byte,Short,Int,Long=(Integeer).Fload,Double=(Ferection).Boolean,Char=(True or False).

**b. Referenced Data:** Any built-in class or interface that system created or we created to declare a variable other than primitive eight types of data, that are considered as a referenced data type like String, Integer is a built-in class in java. (Starts with a capital letter). Example:

54. **What are wrapper classes?**  
Java provides specialized classes corresponding to each of the primitive data types. These are called wrapper classes.  
Examples: Integer, Character, Double, Boolean, etc.

\*\*\* What is Auto boxing and unboxing ?

Converting a primitive data type like int, float, etc. to it's corresponding wrapper class object(Integer,Float etc.) is called boxing or autoboxing. Example of boxing,

int i = 0;  
Integer O = new Integer(i);

Unboxing is the exact opposite of boxing where wrapper class object is converted into primitive data type. Example of unboxing,

i = O.intValue();

-- Differentiate between the constructors and methods in Java?

Methods Constructors

1. Used to represent the behavior of an object 1. Used to initialize the state of an object

2. Must have a return type 2. Do not have any return type

3. Needs to be invoked explicitly 3. Is invoked implicitly

4. No default method is provided by the compiler 4. A default constructor is provided by the compiler if the class has none

5. Method name may or may not be same as class name 5. Constructor name must always be the same as the class name

33. What is String?

String is a non- primitive data type. String is a build in class of java that can be hold a sequence of character.

Strings: The String data type is used to store a sequence of character (text). String value must be surrounded by double quotes. (By W3 School)

34. What is constructor?

A constructor is a special type of method that is used to initialize the object. Constructor is a block of code which execute at the time of object creation and it construct an object. To construct an object means let’s say you want to change a value of a reference then you can use constructor.

There are two type of constructor: 1. Default constructor 2. Parameterized constructor.

Rules for constructor are:

[

Constructor name should be same as class name.

A constructor must have no return type not even void.

It is called automatically at the time of object creation.

Default constructor (no parameter), Parameterized constructor.

It can be overloaded, but cannot be overridden

35. Why we use the constructor in java?

we use the constructor to **initialize the instance variable of the class.**

**Make a variable for useable.**

**19. What is Inheritance in Java?**  
         **Answer:** Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system). Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

         Simply we can say, when we get all the variables and methods from parents class to child class by creating a parent-child relationship, then it is called inheritance.

**Example:**

**20. What is Encapsulation?**

**Answer:**Encapsulation is a process of wrapping the variables and methods together into a single unit. It is also hiding the data in order to make it safe from any modification. That hidden data can be restricted to the members of that class.  
Levels are Public, Protected, Private, Internal, and Protected Internal.  
In short,  
Encapsulation is basically denying the access to the internal implementation or knowledge about internals to the external world, while abstraction is giving a generalized view of any implementation that helps the external world to interact with it.

 **21. What is Polymorphism?**

**Answer:**We can simply saypoly means many and morph means forms, together polymorphism means many forms. **When we create object of a child class by using the reference of parent class, then we can called Polymorphism.**

        There are two kinds of Polymorphism.  
              i. Compile-time polymorphism. (Method overloading is compile-time polymorphism)  
             ii. Runtime polymorphism.  (Method overriding is run time polymorphism)

Example:

**24. What is the interface?**

**Interface is special type of method**

**Answer:**An interface is a collection of abstract methods (only abstract/unimplemented methods). Before using any abstract method, it need to implements in a class, and only then the method can be used.

It has static constants and abstract methods.

Since Java 8, we can have **default and static methods** in an interface.

Since Java 9, we c an have **private methods** in an interface.

* It is used to achieve abstraction.
* By interface, we can support the functionality of multiple inheritance.
* It can be used to achieve loose coupling.

Some roles that need to keep in mind:   
class  can extends another class  
interface can extends another interface  
class can implements  one or more interface at the same time.

class >> extends >> class

interface >> extends >> interface

interface >> implements >> class

## \*\* What is Garbage Collection in Java?

it is a way to destroy the unused objects.

**Garbage Collection in Java** is a process by which the programs perform memory management automatically. The Garbage Collector(GC) finds the unused objects and deletes them to reclaim the memory. In Java, dynamic memory allocation of objects is achieved using the new operator that uses some memory and the memory remains allocated until there are references for the use of the object.

#### **Why Java is not supporting multiple inheritance ?**

The reason behind this is to prevent ambiguity.

Consider a case where class B extends class A and Class C and both class A and C have the same method display().

Now java compiler cannot decide, which display method it should inherit. To prevent such situation, multiple inheritances is not allowed in java.

### Where to use wrapper class in java?

Java collection like list, Set, Map use always class(Double, Integer)

But can’t handle int or double.

\*\*\* Where do you use inheritance in your current project?

I create a base class for application login then I extends it for buy product class (Class cart), or Payment class.

That means I can get login common method from one class to another class.

\*\*\* Give me one interface example of what you use in your current project?

WebDriver dr=new ChromeDriver() ;

Here WevDriver is an interface, dr is an object of CromeDriver()

ChromeDriver is a class.

\*\*\* Where to use abstraction in your current project?

In my current project , I have a HR project or fainalcial account project inside it I have abstract class employee and have some abstract methods like getname(), getlocation(), getContactType().

Therefore, I extends it in new child class and use those methods for employee class.

That is the way I reuse my code and do my abstraction.

So, I can reuse my code easily and maintain abstraction .

\*\*\*Why does Java need an interface?

For 100% abstraction

For multiple inheritance . (Java8: we can use static and default methods and those have body)

\*\*\* Where to use Polymorphism in your current project?

Overloading:

1st used same method with no parameter

2nd used same method with one parameter

3rd used same method with two parameter

These way use same method name but add more parameter.

Overriding:

## **Difference Between Abstraction and Encapsulation**

|  |  |
| --- | --- |
| **Abstraction** | **Encapsulation** |
| Abstraction is a feature of OOPs that hides the **unnecessary** detail but shows the essential information. | Encapsulation is also a feature of OOPs. It hides the code and data into a **single** entity or unit so that the data can be protected from the outside world. |
| It solves an issue at the **design** level. | Encapsulation solves an issue at **implementation** level. |
| It focuses on the **external** lookout. | It focuses on **internal** working. |
| It can be implemented using **abstract classes** and **interfaces**. | It can be implemented by using the [**access modifiers**](https://www.javatpoint.com/access-modifiers) (private, public, protected). |
| It is the process of **gaining** information. | It is the process of **containing** the information. |
| In abstraction, we use **abstract classes** and **interfaces** to hide the code complexities. | We use the **getters** and **setters** methods to hide the data. |
| The objects are **encapsulated** that helps to perform abstraction. | The object need not to **abstract** that result in encapsulation. |

differences between abstract class and interface that are given below.

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1) Abstract class can **have abstract and non-abstract** methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| 2) Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| 3) Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| 4) Abstract class **can provide the implementation of interface**. | Interface **can't provide the implementation of abstract class**. |
| 5) The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| 6) An **abstract class** can extend another Java class and implement multiple Java interfaces. | An **interface** can extend another Java interface only. |
| 7) An **abstract class** can be extended using keyword "extends". | An **interface** can be implemented using keyword "implements". |
| 8) A Java **abstract class** can have class members like private, protected, etc. | Members of a Java interface are public by default. |
| 9)**Example:** public abstract class Shape{ public abstract void draw(); } | **Example:** public interface Drawable{ void draw(); } |

\*\*\* Between Errors vs Exceptions?

**Summary of Differences**

|  |  |
| --- | --- |
| Errors | Exceptions |
| Recovering from Error is not possible. | We can recover from exceptions by either using try-catch block or throwing exceptions back to caller. |
| All errors in java are unchecked type. | Exceptions include both checked as well as unchecked type. |
| Errors are mostly caused by the environment in which program is running. | Program itself is responsible for causing exceptions. |
| Errors occur at runtime and not known to the compiler. | All exceptions occurs at runtime but checked exceptions are known to 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. |

\*\*\*Different between Checked and unchecked exception ?

|  |  |
| --- | --- |
| Checked exceptions occur at compile time. | Unchecked exceptions occur at runtime. |
| The compiler checks a checked exception. | The compiler does not check these types of exceptions. |
| These types of exceptions can be handled at the time of compilation. | These types of exceptions cannot be a catch or handle at the time of compilation, because they get generated by the mistakes in the program. |
| They are the sub-class of the exception class. | They are runtime exceptions and hence are not a part of the Exception class. |
| Here, the JVM needs the exception to catch and handle. | Here, the JVM does not require the exception to catch and handle. |
| Examples of Checked exceptions:   * File Not Found Exception * No Such Field Exception * Interrupted Exception * No Such Method Exception * Class Not Found Exception | Examples of Unchecked Exceptions:   * No Such Element Exception * Undeclared Throwable Exception * Empty Stack Exception * Arithmetic Exception * Null Pointer Exception * Array Index Out of Bounds Exception * Security Exception |

How to handle data in your current project?

I can handle data with variable / object reference

By method parameter

By constructor parameter

Notepad data – with java properties class

Excel data – with Apache poi

Database data – with JDBC

TestNG - @parameter, @Dtaprovide, @Factory

Cucumber -by

Scenario(Data table)

Scenario outline (Example table)

* Can you tell me about your current framework?

My current framework is data driven framework with BDD approach / Hybrid framework

I built my framework from scratch

I used java programing language with selenium web Driver for automation

Maven – Build tool

TestNG – java unit testing framework

Cucumber – BDD automation approach

GitHub – version control

Jenkins -CI

My current framework has

Generic library (all common code with methods)

Page Factory (all Xpaths inside and its act like a repository)

Utility package (all utility methods inside like highlight, screenshot, config file)

DB utility package (database connection with other methods to get value from database)

Excel utility package (to handle excel with apache poi)

For Cucumber:

Feature file (all test steps)

Step def (all steps code)

Runner file( it will run the feature file with step def and also I can use different cucumber options like plugin, dry run, monochrome, strict and hooks )

I execute my code when application in ready and stable

I push my code into GitHub and connect with jenkins for CI

I checked my build from maven or jenkins

Also generate report by the help of testNG and shire reports with my team members.

\*\*\* Do you test full project automation in your current project? jowel

obviously, we don’t do automate our full application so have to find out scope first what part of the automate or testable.

### **Why use inheritance in java**

* For [Method Overriding](https://www.javatpoint.com/method-overriding-in-java) (so [runtime polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java) can be achieved).
* For Code Reusability.

### **Terms used in Inheritance**

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

### **The syntax of Java Inheritance**

1. **class** Subclass-name **extends** Superclass-name
2. {
3. //methods and fields
4. }

## **Types of inheritance in java**

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

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



**Note: Multiple inheritance is not supported in Java through class.**

![Text, letter

Description automatically generated]()

**25. What are runtime exceptions?**

**Answer:** Runtime(uncheck) exceptions are those exceptions that are not warned by the compiler but it is thrown at runtime.  
       Ex: StackOverflowException, MemoryoutException, ArithmaticException.

39. **What is a regular expression in java?**  
 A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. They can be used to search, edit,  or manipulate text and data.

**83. What are Serialization and Deserialization?**

**Answer:** Serialization is a process or mechanism of converting the state of an object into a byte stream.  
   Deserialization is the reverse process of serialization where the byte stream is used to recreate the actual Java object in memory.  
   To make a java object serializable we have to implement either the java.io.Serializable interface or its subinterface, java.io.Externalizable.

**47. What is the common usage of serialization?**

**Answer:**Whenever an object needs to be send over the network or stored in files, objects need to be serialized. Because our network infrastructure and hard disk are hardware components that understand only bites and bytes but not java objects.   
  
**48. What is the difference between final finally and finalize?**

***Answer: final****:* final is a reserved keyword in java. We can not use it as an identifier as it is reserved. We can use this keyword with variables, method, and also with classes. The final keyword in java has different meanings depending upon it is applied to variable, class, or method.

Not the change

**finally**: is a keyword that is used with try and catch block and guarantees that a section of code will be executed, even if an exception is thrown. The finally block will be executed after the try and catch blocks, but before control transfers back to its origin.

**Finally always run the blook of code**

**finalize method:**It is a method that the Java Garbage Collector always calls just before the deletion/ destroying the object which is eligible for garbage collection, so as to perform clean-up activity. Clean-up activity means closing the resources associated with that object like Database connection, network connection or we can say resources de-allocation. Remember it is not a reserved keyword. Once the finalize method completed immediately garbage collector destroys that object. finalize method is present in Object class and it syntax.

**57. What is a transient variable?**  
     **Answer:** Transient variable is a variable that may not be serialized.

90.**What is casting?**  
    **Answer:** Casting is the conversion of one type to another which means is taking an Object of one particular type and “turning it into” another Object type.  
91.**What is an immutable class?**  
   **Answer:** If the class is final. The string is an immutable class. To create a customize immutable class all the property from the final class has to be final.

**99. What is the difference between string and string-buffer?**  
       The main difference between String and StringBuffer is a string is immutable and StringBuffer is mutable

105.**What is the difference between StringBuilder and string-buffer?**  
   The main difference between StringBuilder and String Buffer is StringBuffer is Synchronized and StringBuilder is not synchronized. Also, StringBuilder is faster than         StringBuffer