# Java SE 8 Interview Questions and Answers (Part-2)

APRIL 2, 2018 BY RAMBABU POSA — 12 COMMENTS

In my previous post, I have discussed some important Java SE 8 Interview Questions and Answers. In this post, we are going to discuss some more Java SE 8 Interview Questions and Answers.

Before reading this post, please go through my previous post at: "Java SE 8 Interview Questions (Part 1)".

**Table of Contents** [hide]

## Java SE 8 Interview Questions

1. What is Internal Iteration in Java SE 8?
2. Differences between External Iteration and Internal Iteration?
3. What are the major drawbacks of External Iteration?
4. What are the major advantages of Internal Iteration over External Iteration?
5. What is the major drawback of Internal Iteration over External Iteration?
6. What is the major advantage of External Iteration over Internal Iteration?
7. When do we need to use Internal Iteration? When do we need to use External Iteration?
8. Differences between Intermediate Operations and Terminal Operations of Java 8's Stream API?
9. Is it possible to provide method implementations in Java Interfaces? If possible, how do we provide them?
10. What is a Default Method? Why do we need Default methods in Java 8 Interfaces?
11. What is a Static Method? Why do we need Static methods in Java 8 Interfaces?
12. Differences between Functional Programming and Object-Oriented Programming?
13. Explain issues of Old Java Date API? What are the advantages of Java 8's Date and Time API over Old Date API and Joda Time API?
14. Why do we need new Date and Time API in Java SE 8?Explain how Java SE 8 Data and Time API solves issues of Old Java Date API?
15. What are the Differences between Java's OLD Java Date API and Java 8's Date and Time API?
16. What is Multiple Inheritance? How Java 8 supports Multiple Inheritance?
17. What is Diamond Inheritance Problem? How Java 8 Solves this problem?

## Java SE 8 Interview Questions and Answers

## What is Internal Iteration in Java SE 8?

Before Java 8, We don't Internal Iteration concept. Java 8 has introduced a new feature known as "Internal Iteration". Before Java 8, Java Language has only External Iteration to iterate elements of an Aggregated Object like Collections, Arrays etc.

Internal Iteration means "Iterating an Aggregated Object elements one by one internally by Java API". Instead of Java Application do iteration externally, We ask Java API to do this job internally.

## Differences between External Iteration and Internal Iteration?

| S.NO. | EXTERNAL ITERATION | INTERNAL ITERATION |
|---|---|---|
| 1. | Available before Java 8 too. | It is introduced in Java SE 8 |
| 2. | Iterating an Aggregated Object elements externally. | Iterating an Aggregated Object elements internally (background). |
| 3. | Iterate elements by using for-each loop and Iterators like Enumeration, Iterator, ListIterator. | Iterate elements by using Java API like “forEach” method. |
| 4. | Iterating elements in Sequential and In-Order only. | Not required to iterate elements in Sequential order. |
| 5. | It follows OOP approach that is Imperative Style. | It follows Functional Programming approach that is Declarative Style. |
| 6. | It does NOT separate responsibilities properly that is, it defines both “What is to be done” and “How it is to be done”. | It defines only “What is to be done”. No need to worry about “How it is to be done”. Java API takes care about “How to do”. |
| 7. | Less Readable Code. | More Readable code. |

## What are the major drawbacks of External Iteration?

External Iteration has the following drawbacks:

- We need to write code in Imperative Style.
- There is no clear separation of Responsibilities. Tightly-Coupling between “What is to be done” and “How it is to be done” code.
- Less Readable Code.
- More Verbose and Boilerplate code.
- We have to iterate elements in Sequential order only.
- It does not support Concurrency and Parallelism properly.

## What are the major advantages of Internal Iteration over External Iteration?

Compare to External Iteration, Internal Iteration has the following advantages:

- As it follows Functional Programming style, we can write Declarative Code.
- More Readable and concise code.
- Avoids writing Verbose and Boilerplate code
- No need to iterate elements in Sequential order.
- It supports Concurrency and Parallelism properly.
- We can write Parallel code to improve application performance.

- Clear separation of Responsibilities. Loosely-Coupling between "What is to be done" and "How it is to be done" code.
- We need to write code only about "What is to be done" and Java API takes care about "How it is to be done" code.

## What is the major drawback of Internal Iteration over External Iteration?

Compare to External Iteration, Internal Iteration has one major drawback:

- In Internal Iteration, as Java API takes care about Iterating elements internally, we do NOT have control over Iteration.

## What is the major advantage of External Iteration over Internal Iteration?

Compare to Internal Iteration, External Iteration has one major advantage:

- In External Iteration, as Java API does NOT take care about Iterating elements, we have much control over Iteration.

## When do we need to use Internal Iteration? When do we need to use External Iteration?

We need to understand the situations to use either Internal Iteration or External Iteration.

- When we need more control over Iteration, we can use External Iteration.
- When we do NOT need more control over Iteration, we can use Internal Iteration.
- When we need to develop Highly Concurrency and Parallel applications and we , we should use Internal Iteration.

## Differences between Intermediate Operations and Terminal Operations of Java 8's Stream API?

| S.NO. | STREAM INTERMEDIATE OPERATIONS | STREAM TERMINAL OPERATIONS |
|---|---|---|
| 1. | Stream Intermediate operations are not evaluated until we chain it with Stream Terminal Operation. | Stream Terminal Operations are evaluated on it's own. No need other operations help. |
| 2. | The output of Intermediate Operations is another Stream. | The output of Intermediate Operations is Not a Stream. Something else other than a Stream. |
| 3. | Intermediate Operations are evaluated Lazily. | Terminal Operations are evaluated Eagerly. |
| 4. | We can chain any number of Stream Intermediate | We can NOT chain Stream Terminal Operations. |

| | | |
|---|---|---|
| | Operations. | |
| 5. | We can use any number of Stream Intermediate Operations per Statement. | We can use only one Stream Terminal Operation per Statement. |

## Is it possible to provide method implementations in Java Interfaces? If possible, how do we provide them?

In Java 7 or earlier, It is not possible to provide method implementations in Interfaces. Java 8 on-wards, it is possible.

In Java SE 8, We can provide method implementations in Interfaces by using the following two new concepts:

- Default Methods
- Static Methods

## What is a Default Method? Why do we need Default methods in Java 8 Interfaces?

A Default Method is a method which is implemented in an interface with "default" keyword. It's new featured introduced in Java SE 8.

**We need Default Methods because of the following reasons:**

- It allow us to provide method's implementation in Interfaces.
- To add new Functionality to Interface without breaking the Classes which implement that Interface.
- To provide elegant Backwards Compatibility Feature.
- To ease of extend the existing Functionality.
- To ease of Maintain the existing Functionality.

## What is a Static Method? Why do we need Static methods in Java 8 Interfaces?

A Static Method is an Utility method or Helper method, which is associated to a class (or interface). It is not associated to any object.

**We need Static Methods because of the following reasons:**

- We can keep Helper or Utility methods specific to an interface in the same interface rather than in a separate Utility class.
- We do not need separate Utility Classes like Collections, Arrays etc to keep Utility methods.
- Clear separation of Responsibilities. That is we do not need one Utility class to keep all Utility methods of Collection API like Collections etc.
- Easy to extend the API.
- Easy to Maintain the API.

## Differences between Functional Programming and Object-Oriented Programming?

| FUNCTIONAL PROGRAMMING | OOP |
|---|---|
| Does not exist State | Exists State |
| Uses Immutable data | Uses Mutable data |
| It follows Declarative Programming Model | It follows Imperative Programming Model |
| Stateless Programming Model | Stateful Programming Model |
| Main Fcous on: “What you are doing” | Main focus on “How you are doing” |
| Good for Parallel (Concurrency) Programming | Poor for Parallel (Concurrency) Programming |
| Good for BigData processing and analysis | NOT Good for BigData processing and analysis |
| Supports pure Encapsulation | It breaks Encapsulation concept |
| Functions with No-Side Effects | Methods with Side Effects |
| Functions are first-class citizens | Objects are first-class citizens |
| Primary Manipulation Unit is “Function” | Primary Manipulation Unit is Objects(Instances of Classes) |
| Flow Controls: Function calls, Function Calls with Recursion | Flow Controls: Loops, Conditional Statements |
| It uses “Recursion” concept to iterate Collection Data. | It uses “Loop” concept to iterate Collection Data. For example:-For-each loop in Java |
| Order of execution is less importance. | Order of execution is must and very important. |
| Supports both “Abstraction over Data” and | Supports only “Abstraction over Data”. |

| | |
|---|---|
| "Abstraction over Behavior". | |
| We use FP when we have few Things with more operations. | We use OOP when we have few Operations with more Things. For example: Things are classes and Operations are Methods in Java. |

NOTE:- For more information about FP, IP and OOP comparisons, Please go through my previous post at: "Compare FP, OOP(IP)"

## Explain issues of Old Java Date API? What are the advantages of Java 8's Date and Time API over Old Date API and Joda Time API?

Java's OLD Java Date API means Date API available before Java SE 8 that is Date, Calendar, SimpleDateFormat etc.

Java's Old Date API has the following Issues or Drawbacks compare to Java 8's Date and Time API and Joda Time API.

- Most of the API is deprecated.
- Less Readability.
- java.util.Date is Mutable and not Thread-Safe.
- java.text.SimpleDateFormat is not Thread-Safe.
- Less Performance.

Java SE 8's Date and Time API has the following Advantages compare to Java's OLD Date API.

- Very simple to use.
- Human Readable Syntax that is More Readability.
- All API is Thread-Safe.
- Better Performance.

## Why do we need new Date and Time API in Java SE 8?Explain how Java SE 8 Data and Time API solves issues of Old Java Date API?

We need Java 8's Date and Time API to develop Highly Performance, Thread-Safe and Highly Scalable Java Applications.

Java 8's Date and Time API solves all Java's Old Date API issues by following Immutability and Thread-Safety principles.

## What are the Differences between Java's OLD Java Date API and Java 8's Date and Time API?

**Differences between Java's OLD Java Date API and Java 8's Date and Time API:**

| S.NO. | JAVA'S OLD JAVA DATE API | JAVA 8'S DATE AND TIME API |
|---|---|---|
| 1. | Available before Java 8 too. | It is introduced in Java SE 8 |
| 2. | Not Thread Safe. | Thread Safe. |
| 3. | Mutable API. | Immutable API. |
| 4. | Less Performance. | Better Performance. |
| 5. | Less Readability. | More Readability. |
| 6. | It's not recommended to use as its deprecated. | It's always recommended to use. |
| 7. | Not Extendable. | Easy to Extend. |
| 8. | It defines months values from 0 to 11, that is January = 0. | It defines months values from 1 to 12, that is January = 1. |
| 9. | It's an old API. | It's a new API. |

## What is Multiple Inheritance? How Java 8 supports Multiple Inheritance?

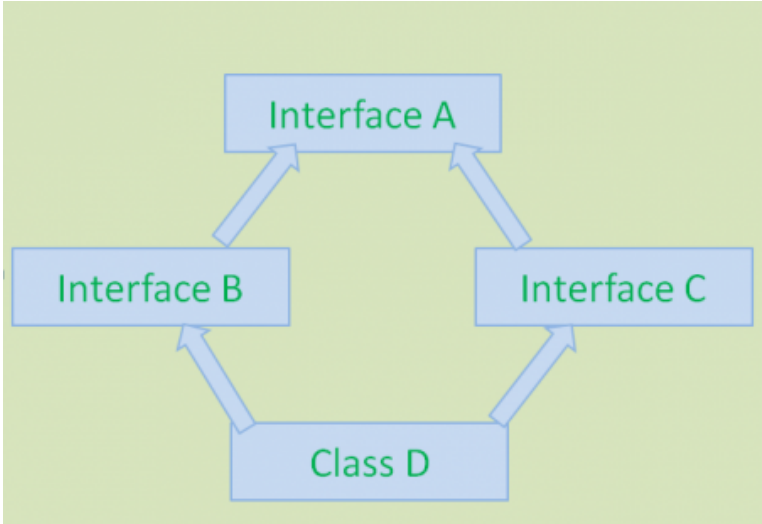
Multiple Inheritance means a class can inherit or extend characteristics and features from more than one parent class.

In Java 7 or Earlier, Multiple Inheritance is not possible because Java follows "A class should extend one and only one class or abstract class" Rule. However, it's possible to provide Multiple Implementation Inheritance using Interface because Java follows "A class can extend any number of Interfaces" Rule.

However, Java 8 supports "Implementing Methods in Interfaces" by introducing new features: Default methods in Interface. Because of this feature, Java 8 supports Multiple Inheritance with some limitations.

## What is Diamond Problem in Inheritance? How Java 8 Solves this problem?

A Diamond Problem is a Multiple Inheritance problem. In Java, It occurs when a Class extends more than one Interface which have same method implementation (Default method).




This above diagram shows Diamond Problem. To avoid this problem, Java 7 and Earlier versions does not support methods implementation in interface and also doesn't support Multiple Inheritance. Java 8 has introduced new feature: Default methods to support Multiple Inheritance with some limitations.

Sample Java SE 8 Code to show this Diamond Problem:

```
public interface A{
  default void display() { //code goes here }
}
public interface B extends A{ }
public interface C extends A{ }
public class D implements B,C{ }
```

In the above code snippet, class D gives compiltime errors because Java Compiler will get bit confusion about which display() has to provide in class D. Class D inherits display() method from both interfaces B and C. To solve this problem, Java SE 8 has given the following remedy:

```
public interface A{
  default void display() { //code goes here }
}
public interface B extends A{ }
public interface C extends A{ }
public class D implements B,C{
  void display() {
    B.super.display();
  }
}
```

This **B.super.display();** will solve this Diamond Problem.

That's it all about Java 8 Interview Questions. We will discuss some more Java SE 8 Interview Questions in my coming posts.

Please drop me a comment if you like my post or have any issues/suggestions.

« PREVIOUS
Java Access Modifiers

NEXT »
Scala Interview Questions

**About Rambabu Posa**
Rambabu Posa have 13 years of RICH experience as Sr Agile Lead Java/Scala/BigData/NoSQL Developer. Apart from Java, he is good at Spring4, Hibernate4, MEAN Stack, RESTful WebServices, NoSQL, BigData Hadoop Stack, Cloud, Scala, Groovy Grails, Play Framework, Lagom Framework and ConductR, TDD, BDD,Agile,DevOps and much more. His hobbies are Developing software, Learning new technologies, Love Walking, Reading Books, Watching TV and obviously sharing his knowledge through writing articles on JournalDev.

FILED UNDER: INTERVIEW QUESTIONS, JAVA

## Comments

**Preetesh Kumar says**
NOVEMBER 15, 2017 AT 3:01 AM
In the above (diamond problem) example, I have not got any compile-time error. when default method of "Interface A" is overridden in "interface B" as well as "interface C" then we are getting compile-time error as follows:
"Duplicate default methods named display with the parameters () and () are inherited from the types C and B"
Reply

**Srao says**
JANUARY 16, 2017 AT 11:55 PM
Good posts on Java SE 8 Interview questions and answers.
Gives an idea on what type of questions we can expect from Interviewers to prepare.
Great job.
Reply

**Ganesh says**
MARCH 28, 2017 AT 5:54 AM
You should said , java8 supports mupltiple inheritance with limitations, My questions what are that limitaions, how to work internally above code, why were you accessed display method with B class.
Reply

**Andrew Minchekov says**
DECEMBER 14, 2016 AT 4:44 AM
public interface A{
default void display() { //code goes here }
}
public interface B extends A{ }
public interface C extends A{ }
public class D implements B,C{ }
You will not have compile error with your code above.
You will only have compile error if classes B and C both have same display method signature.
Reply

**Rambabu Posa says**
DECEMBER 14, 2016 AT 9:04 AM

B & C are Interfaces. When we B extends A, then B should inherit that method too right?
Ram
Reply

**Muralikrishna says**
SEPTEMBER 3, 2016 AT 2:47 AM
Class able to create default methods ?
Please check and correct the code snippet.
Reply

**Palani says**
AUGUST 12, 2016 AT 11:01 PM
You dont have to have many duplicate questions. Otherwise, It is an informative blog.
Reply

**Sriphani says**
JULY 14, 2016 AT 6:27 PM
Interesting concepts and very understandable way of explanation.
Thanks.
Reply

**Anuj Gupta says**
FEBRUARY 25, 2016 AT 2:46 AM
Very Helpful Tutorial.. Thanks!!!
Reply

**Rakesh says**
JANUARY 30, 2016 AT 2:17 AM
Yours two Java 8 interview posts are too good. Waiting for more. Thanks l
Reply

**jahir says**
JANUARY 10, 2016 AT 11:31 PM
yes it is good tutorial.
Reply

**Suresh says**
JANUARY 6, 2016 AT 2:22 AM
Very good tutorial on Java 8 questions. Thank you so much author for your help.
Reply

## Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

- Scala Interview Questions Summary
- Common Job Interview Questions

## Miscellaneous

- Java ClassLoader
- String StringBuffer StringBuilder
- Java is Pass By Value
- Java Heap vs Stack Memory

RECOMMENDED TUTORIALS

## Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

## Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial