Java SE 8 Interview Questions and Answers (Part-1)

APRIL 2, 2018 BY RAMBABU POSA — 27 COMMENTS

In this post, we are going to discuss some important Java SE 8 Interview Questions with Answers. I will write one more post to discuss remaining Java SE 8 Interview Questions.

Java 8 Interview Questions

1. Why do we need change to Java again?
2. Java SE 8 New Features?
3. Advantages of Java SE 8 New Features?
4. What is Lambda Expression?
5. What are the three parts of a Lambda Expression? What is the type of Lambda Expression?
6. What is a Functional Interface? What is SAM Interface?
7. Is is possible to define our own Functional Interface? What is @FunctionalInterface? What are the rules to define a Functional Interface?
8. Is @FunctionalInterface annotation mandatory to define a Functional Interface? What is the use of @FunctionalInterface annotation? Why do we need Functional Interfaces in Java?

9. When do we go for Java 8 Stream API? Why do we need to use Java 8 Stream API in our projects?
10. Explain Differences between Collection API and Stream API?
11. What is Spliterator in Java SE 8?Differences between Iterator and Spliterator in Java SE 8?
12. What is Optional in Java 8? What is the use of Optional?Advantages of Java 8 Optional?
13. What is Type Inference? Is Type Inference available in older versions like Java 7 and Before 7 or it is available only in Java SE 8?

Java 8 Interview Questions and Answers

In this section, we will pickup each question from previous section and answer it with in-detailed description. If you need any more information and examples, please go through previous Java SE 8 posts available in JournalDEV.

Why do we need change to Java again?

Oracle Corporation has introduced a lot of new concepts in Java SE 8 to introduce the following benefits:

- **To Utilize Current Multi-Core CPUs Efficiently**
  Recently, we can observe drastic changes in Hardware. Now-a-days, all systems are using Multi-Core CPUs (2,4,8,16-Core etc.) to deploy and run their Applications. We need new Programming Constructs in Java to utilize these Multi-Core Processors efficiently to develop Highly Concurrently and Highly Scalable applications.

- **To Utilize FP Features**
  Oracle Corporation has introduced a lot of FP(Functional Programming) concepts as part of Java SE 8 to utilize the advantages of FP.

Java SE 8 New Features?

- Lambda Expressions
- Functional Interfaces
- Stream API
- Date and Time API
- Interface Default Methods and Static Methods
- Spliterator
- Method and Constructor References
- Collections API Enhancements
- Concurrency Utils Enhancements
- Fork/Join Framework Enhancements
- Internal Iteration
- Parallel Array and Parallel Collection Operations
- Optional
- Type Annotations and Repeatable Annotations
- Method Parameter Reflection
- Base64 Encoding and Decoding

- IO and NIO2 Enhancements
- Nashorn JavaScript Engine
- javac Enhancements
- JVM Changes
- Java 8 Compact Profiles: compact1,compact2,compact3
- JDBC 4.2
- JAXP 1.6
- Java DB 10.10
- Networking
- Security Changes

Advantages of Java SE 8 New Features?

We can get the following benefits from Java SE 8 New Features:

- More Concise and Readable code
- More Reusable code
- More Testable and Maintainable Code
- Highly Concurrent and Highly Scalable Code
- Write Parallel Code
- Write Database Like Operations
- Better Performance Applications
- More Productive code

What is Lambda Expression?

Lambda Expression is an anonymous function which accepts a set of input parameters and returns results.

Lambda Expression is a block of code without any name, with or without parameters and with or without results. This block of code is executed on demand.

What are the three parts of a Lambda Expression? What is the type of Lambda Expression?

A Lambda Expression contains 3 parts:

- Parameter List
  A Lambda Expression can contain zero or one or more parameters. It is optional.

- Lambda Arrow Operator
  “->” is known as Lambda Arrow operator. It separates parameters list and body.

- Lambda Expression Body

The type of "Journal Dev" is java.lang.String. The type of "true" is Boolean. In the same way, what is the type of a Lambda Expression?
The Type of a Lambda Expression is a Functional Interface.

Example:- What is the type of the following Lambda Expression?

```
() -> System.out.println("Hello World");
```

This Lambda Expression does not have parameters and does return any results. So it's type is "java.lang.Runnable" Functional Interface.

What is a Functional Interface? What is SAM Interface?

A Functional Interface is an interface, which contains one and only one abstract method. Functional Interface is also know as SAM Interface because it contains only one abstract method.

SAM Interface stands for Single Abstract Method Interface. Java SE 8 API has defined many Functional Interfaces.

Is is possible to define our own Functional Interface? What is @FunctionalInterface? What are the rules to define a Functional Interface?

Yes, it is possible to define our own Functional Interfaces. We use Java SE 8's @FunctionalInterface annotation to mark an interface as Functional Interface.

We need to follow these rules to define a Functional Interface:

- Define an interface with one and only one abstract method.
- We cannot define more than one abstract method.
- Use @FunctionalInterface annotation in interface definition.
- We can define any number of other methods like Default methods, Static methods.
- If we override java.lang.Object class's method as an abstract method, which does not count as an abstract method.

Is @FunctionalInterface annotation mandatory to define a Functional Interface? What is the use of @FunctionalInterface annotation? Why do we need Functional Interfaces in Java?

It is not mandatory to define a Functional Interface with @FunctionalInterface annotation. If we don't want, We can omit this annotation. However, if we use it in Functional Interface definition, Java Compiler forces to use one and only one abstract method inside that interface.

Why do we need Functional Interfaces? The type of a Java SE 8's Lambda Expression is a Functional Interface. Whereever we use Lambda Expressions that means we are using Functional Interfaces.

When do we go for Java 8 Stream API? Why do we need to use Java 8 Stream API in our projects?

When our Java project wants to perform the following operations, it's better to use Java 8 Stream API to get lot of benefits:

- When we want perform Database like Operations. For instance, we want perform groupby operation, orderby operation etc.
- When want to Perform operations Lazily.
- When we want to write Functional Style programming.
- When we want to perform Parallel Operations.
- When want to use Internal Iteration
- When we want to perform Pipelining operations.
- When we want to achieve better performance.

Explain Differences between Collection API and Stream API?

S.NO.	COLLECTION API	STREAM API
1.	It's available since Java 1.2	It is introduced in Java SE8
2.	It is used to store Data(A set of Objects).	It is used to compute data(Computation on a set of Objects).
3.	We can use both Spliterator and Iterator to iterate elements. We can use forEach to performs an action for each element of this stream.	We can't use Spliterator or Iterator to iterate elements.
4.	It is used to store limited number of Elements.	It is used to store either Limited or Infinite Number of Elements.
5.	Typically, it uses External Iteration concept to iterate Elements such as Iterator.	Stream API uses External Iteration to iterate Elements, using forEach methods.
6.	Collection Object is constructed Eagerly.	Stream Object is constructed Lazily.
7.	We add elements to Collection object only after it is computed completely.	We can add elements to Stream Object without any prior computation. That means Stream objects are computed on-demand.
8.	We can iterate and consume elements from a Collection	We can iterate and consume elements from a Stream

	Object at any number of times.	Object only once.

What is Spliterator in Java SE 8?Differences between Iterator and Spliterator in Java SE 8?

Spliterator stands for Splitable Iterator. It is newly introduced by Oracle Corporation as part Java SE 8.
Like Iterator and ListIterator, It is also one of the Iterator interface.

S.NO.	SPLITERATOR	ITERATOR
1.	It is introduced in Java SE 8.	It is available since Java 1.2.
2.	Splitable Iterator	Non-Splitable Iterator
3.	It is used in Stream API.	It is used for Collection API.
4.	It uses Internal Iteration concept to iterate Streams.	It uses External Iteration concept to iterate Collections.
5.	We can use Spliterator to iterate Streams in Parallel and Sequential order.	We can use Iterator to iterate Collections only in Sequential order.
6.	We can get Spliterator by calling spliterator() method on Stream Object.	We can get Iterator by calling iterator() method on Collection Object.
7.	Important Method: tryAdvance()	Important Methods: next(), hasNext()

What is Optional in Java 8? What is the use of Optional?Advantages of Java 8 Optional?

Optional:
Optional is a final Class introduced as part of Java SE 8. It is defined in java.util package.

It is used to represent optional values that is either exist or not exist. It can contain either one value or zero value. If it contains a value, we can get it. Otherwise, we get nothing.

It is a bounded collection that is it contains at most one element only. It is an alternative to "null" value.

Main Advantage of Optional is:

- It is used to avoid null checks.
- It is used to avoid "NullPointerException".

What is Type Inference? Is Type Inference available in older versions like Java 7 and Before 7 or it is available only in Java SE 8?

Type Inference means determining the Type by compiler at compile-time.

It is not new feature in Java SE 8. It is available in Java 7 and before Java 7 too.

Before Java 7:-
Let us explore Java arrays. Define a String of Array with values as shown below:

```
String str[] = { "Java 7", "Java 8", "Java 9" };
```

Here we have assigned some String values at right side, but not defined it's type. Java Compiler automatically infers it's type and creates a String of Array.

Java 7:-
Oracle Corporation has introduced "Diamond Operator" new feature in Java SE 7 to avoid unnecessary Type definition in Generics.

```
Map<String,List<Customer>> customerInfoByCity = new HashMap<>();
```

Here we have not defined Type information at right side, simply defined Java SE 7's Diamond Operator "".

Java SE 8:-
Oracle Corporation has enhanced this Type Inference concept a lot in Java SE 8. We use this concept to define Lambda Expressions, Functions, Method References etc.

```
ToIntBiFunction<Integer,Integer> add = (a,b) -> a + b;
```

Here Java Compiler observes the type definition available at left-side and determines the type of Lambda Expression parameters a and b as Integers.

That's it about Java 8 Interview Questions.

I have discussed some Java SE 8 Interview Questions in this post and 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 REPL – jshell

NEXT »
Java Access Modifiers

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

vineet says

MAY 29, 2018 AT 7:16 AM

can anybody explain why Integer add=(a,b)-> a+b; this will not complie?
ToIntBiFunction add = (a,b) -> a + b; this will compile successfully
Reply

Anup Singh says
JUNE 4, 2018 AT 11:03 AM
You can not do it because type of Lambda expression is not an Integer but a functional interface.. So the code should be Functional_Interface var = Lambda Expression; I hope it helps. Lambda expression can be passed as parameter only where a method expects a functional interface as input. I hope it help.
Reply

sandy says
MAY 5, 2018 AT 8:14 AM
3. It is used in Stream API.-> you mentioned this point in differances between iterator and spliteratior
We can't use Spliterator or Iterator to iterate elements.-> you mentioned this point in diff between collection api and stream api which one is correct are we using spliterator in Stream api?
Reply

Manish Kumar says
JANUARY 7, 2018 AT 1:25 AM
This is about Type of Lambda, I think Lambda does not have any type, until It get assigned to variable. As we can see
() -> System.out.println("Hello World"); can be assigned to any functional interface which have public abstract void xxx() method. It is not specific to "java.lang.Runnable".
Reply

Chandrakanth says
OCTOBER 3, 2017 AT 7:25 PM
Hi
Very good points to note on.
But , I think @FunctionlInterface is mandatory to define Functional Interface. The last point in the answer of this question states the definition of the same.
Thank You
Reply

Manish Kumar says
JANUARY 7, 2018 AT 2:15 AM
@FunctionlInterface enforces interface to have only one abstract method so it is mandatory for functional interface,
But the point is Lambda can be assigned to any interface which have single abstract method, So the lambda can be used with even those interfaces which is not marked as functional.
Reply

Gundamaiah says
SEPTEMBER 26, 2017 AT 11:43 PM
Nice Questions.
There is one typo in the answer while comparing Spliterator and Iterator. You mentioned Spilterator instead of Iterator in 5th point.
Reply

Pankaj says
SEPTEMBER 27, 2017 AT 1:15 AM
Thanks for noticing the typo error, I have corrected it.
Reply

Pramod says
AUGUST 24, 2017 AT 2:10 AM
small mistake
while mentioning differences between stream and collection . You have mentioned We can use both Spliterator and Iterator to iterate elements.. in both places. Please correct it.
Reply

Pankaj says
AUGUST 24, 2017 AT 3:04 AM
Thanks for pointing it out, i have fixed it.
Reply

Deb says
AUGUST 19, 2017 AT 8:28 AM
Good one
Reply

Rohi says
AUGUST 12, 2017 AT 9:56 AM
some mistake in content … for stream you have mentioned …It uses External Iteration to iterate Elements.
Reply

Tanaya says
JANUARY 30, 2018 AT 9:35 PM
I agree , stream uses internal iteration, not external
Reply

Pankaj says
JANUARY 31, 2018 AT 7:36 PM
Thanks friends for the comments, I rechecked it and corrected the error. Thanks Again.
Reply

Indiver says
JULY 15, 2017 AT 6:56 PM
Very helpful!!
Keep it up
Reply

Uma says
JUNE 28, 2017 AT 3:43 PM
Nice one Sir, Keep posting!
Reply

vishal says

MARCH 23, 2017 AT 8:13 AM

Nice information, In some mins get to know new concepts of java8 has introduced.

Reply

Ruwan says

MARCH 16, 2017 AT 4:52 AM

It's a great introduction. Thanks

Reply

Viswanathan says

JANUARY 30, 2017 AT 1:27 AM

Nice article to know Java 8 features

Reply

srao says

JANUARY 16, 2017 AT 10:42 PM

Very Good. I like the way you present the questions and Answers straight and clear,

Reply

symbion says

NOVEMBER 26, 2016 AT 11:42 PM

Hi there,

I like your site very much, lot to learn! Just one question:

Isn't Stream API the one uses Internal Iteration? In Part2, your first property of Internal Iteration is that it is intruduced in Java 8, so Collections are supposed to iterate Externally, right?

Thx

Reply

Rattan says

NOVEMBER 23, 2016 AT 10:27 AM

Nice article, much useful.

Just one thing I noticed though. Below would not compile :

Integer add = (a,b) -> a + b;

Because add must be of type functional interface.

Reply

Soumen says

MARCH 6, 2017 AT 7:29 PM

Very Nice Article.

As mentioned earlier

Integer add=(a,b)-> a+b;

will not compile.

Instead of Integer add, it should be ToIntBiFunction add as ToIntBiFunction functional interface takes two integer parameter.

ToIntBiFunction add = (a,b) -> a + b;

System.out.println(add.applyAsInt(new Integer(10),new Integer(10)));

Thanks!!!

Reply

Rambabu Posa says

MARCH 7, 2017 AT 1:42 PM

Thanks good catch. Updated

Reply

Rajib says

SEPTEMBER 29, 2016 AT 11:41 AM

Good !!!!!!!!!!!!!!!!

Reply

sushil says

JANUARY 19, 2016 AT 8:46 PM

Good One

Reply

Rajat says

SEPTEMBER 7, 2016 AT 3:55 AM

it's nice..thanks

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...

DOWNLOAD ANDROID APP

GET IT ON Google Play

CORE JAVA TUTORIAL

Java 10 Tutorials Java 9 Tutorials Java 8 Tutorials Java 7 Tutorials Core Java Basics OOPS Concepts Data Types and Operators String Manipulation Java Arrays Annotation and Enum Java Collections Java IO Operations Java Exception Handling MultiThreading and Concurrency Regular Expressions Advanced Java Concepts

IMPORTANT INTERVIEW QUESTIONS

Java Interview Questions

- Core Java Interview Questions
- String Interview Questions
- Multithreading Interview Questions
- Collections Interview Questions
- Exception Interview Questions
- Java Programming Interview Questions
- Java 8 Interview Questions Part 1
- Java 8 Interview Questions Part 2
- Servlet Interview Questions
- JSP Interview Questions
- Struts 2 Interview Questions
- JDBC Interview Questions
- Spring Interview Questions
- Hibernate Interview Questions
- JSF Interview Questions
- Web Services Interview Questions
- Scala Basic Interview Questions
- Scala Intermediate Interview Questions
- Scala Advanced Interview Questions
- Scala Interview Questions Summary
- Common Job Interview Questions

Miscellaneous

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

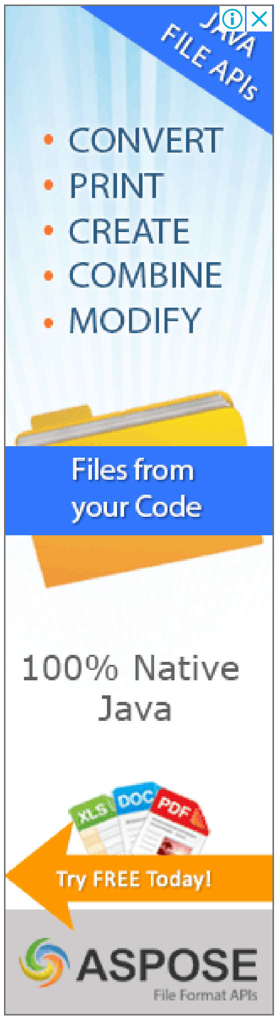
RECOMMENDED TUTORIALS

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


JAVA FILE APIs
• CONVERT
• PRINT
• CREATE
• COMBINE
• MODIFY
Files from your Code
100% Native Java
XLS
DOC
PDF
Try FREE Today!
ASPOSE
File Format APIs