Register | Login | Logout | Contact Us

# Java-Success.com

Industrial strength Java/JEE Career Companion to open more doors

search here … Go

Home | Java FAQs | 600+ Java Q&As | Career | Tutorials | Member | Why?

Can u Debug? | Java 8 ready? | Top X | Productivity Tools | Judging Experience?

Home › Interview › Core Java Interview Q&A › IO › 06: ♥ Java 8 way of File reading and functionally processing the data

# 06: ♥ Java 8 way of File reading and functionally processing the data

Posted on July 7, 2015 by Arulkumaran Kumaraswamipillai

5 Like Share Tweet 1 G+1 4 Share

```
package com.read.file;

import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.stream.Stream;

public class MyFileReader {

    public static void main(String[] args) {
        Path file = Paths.get("C:\\Users\\akumar

        try
        {
            //Java 8: Stream class
            Stream<String> lines = Files.lines(

            for( String line : (Iterable<String>
```

9 tips to earn more | What can u do to go places? | **945+** paid members. LinkedIn Group. **Reviews**

## 600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

open all | close all

- ⊞ Ice Breaker Interview
- ⊟ Core Java Interview (
  - ⊞ Java Overview (4)
  - ⊞ Data types (6)
  - ⊞ constructors-methc
  - ⊞ Reserved Key Wor
  - ⊞ Classes (3)
  - ⊞ Objects (8)
  - ⊞ OOP (10)
  - ⊞ GC (2)
  - ⊞ Generics (5)
  - ⊞ FP (8)
  - ⊟ IO (7)

```
            {
                System.out.println("read=" + lin
            }

        } catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

**Output:**

```
read=A big brown fox
read=jumped over the fence
```

## #1 double colon notation ::

The new double colon (::) operator that Java 8 has to convert a normal method into lambda expression. So,

**Instead of:**

```
list.forEach(n -> System.out.println(n));
```

**You can do:**

```
list.forEach(System.out::println);
```

## #2 Why is **stream::iterator** used?

"**lines::iterator**" where **iterator()** is an instance method on "BaseStream<T,Stream<T>>" from which **java.util.Stream<T>** extends. The "iterator()" returns an "Iterator<T>". The for each loop works on Iterable<T>.

```
for (element : iterable);
```

So, given a Stream s, the following results in an Iterable:

♥ Reading a text
♦ 15 Java old I/C
06: ♥ Java 8 way
Processing large
Processing large
Read a text file f
Reloading config
⊞ Multithreading (12)
⊞ Algorithms (5)
⊞ Annotations (2)
⊞ Collection and Data
⊞ Differences Betwee
⊞ Event Driven Progr
⊞ Exceptions (2)
⊞ Java 7 (2)
⊞ Java 8 (24)
⊞ JVM (6)
⊞ Reactive Programn
⊞ Swing & AWT (2)
⊞ JEE Interview Q&A (3
⊞ Pressed for time? Jav
⊞ SQL, XML, UML, JSC
⊞ Hadoop & BigData Int
⊞ Java Architecture Inte
⊞ Scala Interview Q&As
⊞ Spring, Hibernate, & I
⊞ Testing & Profiling/Sa
⊞ Other Interview Q&A f
⊞ ▶ Free Java Interview

**As a Java Architect**

Java architecture & design concepts interview Q&As with diagrams | What should be a typical Java EE architecture?

```
s::iterator
```

If you want to use this directly in an enhanced-for loop, you have to apply a cast in order to establish a target type for the method reference.

```
for( String line : (Iterable<String>) lines::iter
```

### **#3** Iterator Vs Iterable difference?

An **Iterable<T>** is a simple representation of a series of elements that can be iterated over, and it does not have any iteration state such as a "current element". Instead, it has a "**iterator()**" method that produces an Iterator. Implementing this interface allows an object to be the target of the "**for-each loop**" statement.

An **Iterator<T>** is the object with iteration state to let you check if it has more elements using hasNext() and move to the next() element.

## Read from the classpath

```
package com.read.file;

import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.stream.Stream;

public class MyFileReader {

    public static void main(String[] args) {
        Path file = null;
        try
        {
            //read from the classpath
            URL url = MyFileReader.class.getReso
            file = Paths.get(url.toURI());

            //Java 8: Stream class
            Stream<String> lines = Files.lines(

            for( String line : (Iterable<String>
            {
                System.out.println("read=" + lin
```

## Senior Java developers must have a good handle on

open all | close all

⊞ Best Practice (6)
⊞ Coding (26)
⊞ Concurrency (6)
⊞ Design Concepts (7)
⊞ Design Patterns (11)
⊞ Exception Handling (3
⊞ Java Debugging (21)
⊞ Judging Experience I
⊞ Low Latency (7)
⊞ Memory Managemen
⊞ Performance (13)
⊞ QoS (8)
⊞ Scalability (4)
⊞ SDLC (6)
⊞ Security (13)
⊞ Transaction Managen

## 80+ step by step Java Tutorials

open all | close all

⊞ Setting up Tutorial (6)
⊞ Tutorial - Diagnosis (2
⊞ Akka Tutorial (9)
⊞ Core Java Tutorials (2
⊞ Hadoop & Spark Tuto
⊞ JEE Tutorials (19)
⊞ Scala Tutorials (1)
⊞ Spring & HIbernate Tu
⊞ Tools Tutorials (19)
⊞ Other Tutorials (45)

```
            }

        } catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

**Output:**

```
read=A big brown fox
read=jumped over the fence
```

# Filter the line that has “fox”

```
package com.read.file;

import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Optional;
import java.util.stream.Stream;

public class MyFileReader {

    public static void main(String[] args) {
        Path file = null;

        try
        {
            //read from the classpath
            URL url = MyFileReader.class.getReso
            file = Paths.get(url.toURI());

            //Java 8: Stream class
            Stream<String> lines = Files.lines(

            //Java 8 Optional class with Lambda
            Optional<String> lineThatHasFox = li

            if(lineThatHasFox.isPresent()){
                System.out.println(lineThatHasFo
            }

        } catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

**Output:**

```
A big brown fox
```

## 100+ Preparing for pre-interview Java written home assignments & coding tests

open all | close all

- Can you write code? (
- ♦ Complete the given
- Converting from A to I
- Designing your classe
- Java Data Structures
- Passing the unit tests
- What is wrong with th
- Writing Code Home A
- Written Test Core Jav
- Written Test JEE (1)

## How good are your…to go places?

open all | close all

- Career Making Know-
- Job Hunting & Resum

**Note:** filter() is an **intermediate operation**, returning a Stream, and findFirst() is a **terminal operation**.

# Count lines

```
package com.read.file;

import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class MyFileReader {

    public static void main(String[] args) {
        Path file = null;

        try
        {
            //read from the classpath
            URL url = MyFileReader.class.getReso
            file = Paths.get(url.toURI());

            long count = Files.lines( file, Stan
            System.out.println("count lines=" +


        } catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

**Output:**

```
count lines=2
```

**Note:** count() is a **terminal operation** as it does not return a stream.

# Can you workout the output of the following code?

```
package com.read.file;

import static java.util.stream.Collectors.toList

import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
```

```java
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.Arrays;

public class MyFileReader {

    public static void main(String[] args) {
        Path file = null;

        try
        {
            //read from the classpath
            URL url = MyFileReader.class.getReso
            file = Paths.get(url.toURI());

           List<String> output = Files.lines( fi
                 .filter(s -> s.contains("fox"))
                 .map(line -> line.split("\\s+")
                 .flatMap(Arrays::stream)
                 .limit(2)
                 .collect(toList());

           System.out.println(output);


        } catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

**Output:**

An array of size two containing …..

```
[A, big]
```

# Popular Posts

♦ 11 Spring boot interview questions & answers

**892 views**

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

**851 views**

18 Java scenarios based interview Questions and Answers

**456 views**

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

**411 views**

♦ 7 Java debugging interview questions & answers

**315 views**

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

**313 views**

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

**307 views**

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

**289 views**

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

**267 views**

8 Git Source control system interview questions & answers

**216 views**

Bio | **Latest Posts**

## Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.

**About** Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold

35,000+ copies. Books are outdated and replaced with this subscription based site.

‹ ♥ Reading a text file in Java with the Scanner

♥ Java Iterable Vs Iterator differences and know how ›

**Posted in** IO, Java 8

**Tags:** Free Content

# Empowers you to open more doors, and fast-track

### Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

### Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

# Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

# © Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.

© 2016 Java-Success.com ↑ Responsive Theme powered by WordPress