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 › Java 8 › Non-trival Java 8 tutorial to appreciate the power of functional programming (FP)

Non-trival Java 8 tutorial to appreciate the power of functional programming (FP)

Posted on February 26, 2015 by Arulkumaran Kumaraswamipillai

0 Like Share 0 G+1 Share

Prerequisite: Understanding of Java 8 streams.

Q. You have a list of key value pairs ("student name", "subject") as shown below.

Input

```
("Sam Peter", "English"),
("Sam Peter", "Science"),
("John English", "Science"),
("Neil Graham", "Maths"),
("Neil Graham", "Arts"),
```

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)
⊞ Multithreading (12)
⊞ Algorithms (5)
⊞ Annotations (2)
⊞ Collection and Data
⊞ Differences Betwee
⊞ Event Driven Progr
⊞ Exceptions (2)
⊞ Java 7 (2)

```
("Neil Graham", "Music")

```

The required task is to list all the subjects for each student as shown below.

Result

```
"Samp Pter"    --> [English, Science]
"John English" --> [Science]
"Neil Graham"  --> [Maths, Arts, Music]

```

A. Here is the Java 8 example with lambda expressions, and then let’s discuss this in detail.

```
package com.java8.examples;

import java.util.Arrays;
import java.util.List;
import java.util.Map;
import static java.util.stream.Collectors.*;

public class KeyValuePair<K,V> {

    private final K key;
    private final V value;

    public K getKey() {
        return key;
    }

    public V getValue() {
        return value;
    }

    public KeyValuePair(K key, V value) {
        this.key = key;
        this.value = value;
    }

    public static void main(String[] args) {
         List<KeyValuePair<String,String>> stude
                  new KeyValuePair<String, Strin
                  new  KeyValuePair<String, Stri
                  new  KeyValuePair<String, Stri
                  new  KeyValuePair<String, Stri
                  new  KeyValuePair<String, Stri
                  new  KeyValuePair<String, Stri

                );

         Map<String, List<String>> result =
                 studentSubjects.stream()
                 .collect(groupingBy(KeyValuePai

```

Java 8 (24)
01: ♦ 19 Java 8 I
02: ♦ Java 8 Stre
03: ♦ Functional
04: ♥♦ Top 6 tips
04: Convert Lists
04: Understandir
05: ♥ 7 Java FP
05: ♦ Finding the
06: ♥ Java 8 way
07: ♦ Java 8 API
08: ♦ Write code
10: ♦ ExecutorSe
Fibonacci numbe
Java 8 String str
Java 8 using the
Java 8: 7 useful
Java 8: Different
Java 8: Does “O
Java 8: What is
Learning to write
Non-trival Java 8
Top 6 Java 8 fea
Top 8 Java 8 fea
Understanding J
JVM (6)
Reactive Programr
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 1
▶ Free Java Interview

```
        System.out.println(result);
    }
}


```

Key Points

1) The key and value objects are marked as final. One of the tenets of FP is immutability.
2) The “KeyValuePair” class is written with generics to take any data type. In this example K is of type String, and V is of type String. In fact the “main” method should be moved to a separate class of its own.
3) Note the static import of “Collectors”, which performs reduction operations, such as accumulating elements into collections, summarizing elements according to various criteria, etc.

```
import static java.util.stream.Collectors.*;
```

Which is used for methods “groupingBy”, “mapping”, and “toList” methods. This can be rewritten with “Collectors” as shown below

```
Map<String, List<String>> result =
                 studentSubjects.stream()
                 .collect(Collectors.groupingBy(K
                               Collectors.mapping
                               Collectors.toList(


```

Q. Why the forEach, collect, mapping, etc methods not added to **java.util.List** nor the **java.util.Collection** interface ?
A. stream( ) is a default method added to the Collection interface in Java 8. The stream( ) returns a java.util.Stream interface with multiple abstract methods like filter, map, sorted, collect, etc.

If these were added to the Collection interface and once published, it is impossible to add new methods to an interface without breaking the existing implementation. Hence, “**default**” methods were introduced in Java 8. From Java 8

16 Technical Key Areas

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)

100+ Java pre-interview

onwards, **implementation methods** can now be added to interfaces providing a default implementation of the declared behavior.

Q. Why these methods were added to Stream and Collector instead of adding as a default method to the Collection class?
A. Due to **backward compatibility**. Many existing projects extend pre Java 8 implementation of “Collection” and should be refactored.

Popular Posts

- ♦ 11 Spring boot interview questions & answers
  825 views
- ♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers
  766 views
- 18 Java scenarios based interview Questions and Answers
  400 views
- 001A: ♦ 7+ Java integration styles & patterns interview questions & answers
  388 views
- 01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers
  295 views
- ♦ 7 Java debugging interview questions & answers
  293 views
- 01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints
  285 views
- ♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers
  279 views
- ♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers
  239 views
- 001B: ♦ Java architecture & design concepts interview questions & answers
  201 views

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 .....?

open all | close all

- Career Making Know-
- Job Hunting & Resum

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.**945+** paid members. join my LinkedIn Group. **Reviews**

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.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ 02: ♦ Java 8 Streams, lambdas, intermediate vs terminal ops, and lazy loading with simple examples

Understanding atomicity with a Java example ›

Posted in Java 8

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.