

Java-Success.com

Industrial strength Java/JEE Career Companion for those who want to go places

[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](#) › [FP](#) › 05: ♥ 7 Java FP (lambda expressions) real life examples in wrangling normal & big data

05: ♥ 7 Java FP (lambda expressions) real life examples in wrangling normal & big data

Posted on [May 21, 2016](#) by [Arulkumaran Kumaraswamipillai](#)

This post extends [Transforming your thinking from OOP to FP](#). In Big-data, functional programming is prevalent when working with data sets. For example, writing a Spark job to work with RDDs (Resilient Distributed Data sets).

In **Imperative** (E.g. OOP, procedural programming, etc) programming you can say

```
1 x = x + 5
```

[9 tips to earn more](#) | [What can u do to go places?](#) | **945+** members. [LinkedIn Group](#). [Reviews](#)

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

[open all](#) | [close all](#)

[Ice Breaker Interview](#)

[Core Java Interview C](#)

[Java Overview \(4\)](#)

[Data types \(6\)](#)

[constructors-methc](#)

[Reserved Key Wor](#)

[Classes \(3\)](#)

[Objects \(8\)](#)

[OOP \(10\)](#)

[GC \(2\)](#)

[Generics \(5\)](#)

```
2 y = x + y
3
```

where you are assigning x to x + 5. if x were to be 2, then after assignment it becomes 7 (i.e. 2 + 5)

In **functional programming** (FP), you can't say "x = x + 5" why? if x were to be 2, "2 = 2 + 5" is **wrong**. FP does not have assignment statements. FP is all about computation as the evaluation of "**mathematical functions**" and avoids changing-states and mutability. In FP, you need to say

```
1 f(x) -> x + 5
2 f(x,y) -> x + y
3
```

where f(x) and f(x,y) are functions. Similarly, in the example below "(el1, el2) -> el1 + ',' + el2" is a **lambda expression** in FP. Where "el1" and "el2" are consecutive elements in a given list.

Example 1: Reducing a list of strings to CSV

stream -> reduce -> get

```
1 import java.util.Arrays;
2 import java.util.List;
3
4 public class Technology {
5
6     public static void main(String[] args) {
7         List<String> technologies = Arrays.asList(
8             // Java 8 FP
9             "Java", "JEE", "JDBC", "Spring", "Hibernate",
10            "SQL", "XML", "UML", "JSC", "Hadoop", "BigData",
11            "Scala", "Testing", "Profiling", "Sa",
12            "Other Interview Q&As", "Free Java Interview"
13        );
14    }
15 }
```

Output

```
1
2 Java,JEE,JDBC,Spring,Hibernate
3
```

FP (8)

01: ♦ 19 Java 8 I

02: ♦ Java 8 Stre

03: ♦ Functional

04: ♥♦ Top 6 tips

05: ♥ 7 Java FP

Fibonacci numbe

Java 8 String str

Java 8: What is c

IO (7)

Multithreading (12)

Algorithms (5)

Annotations (2)

Collection and Data

Differences Between

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 1

Free Java Interview

As a Java Architect

[Java architecture & design concepts](#)
[interview Q&As with diagrams](#) | [What should](#)

Example 2: Reducing a list of Integers to string (e.g. CSV)[be a typical Java EE architecture?](#)**stream -> map -> reduce -> get**

```

1 import java.util.Arrays;
2 import java.util.List;
3
4 public class Weights {
5
6     public static void main(String[] args) {
7         List<Integer> weights= Arrays.asList(25,
8         // Java 8 FP
9         String csvWeights = weights.stream()
10            .map(el1 -> el1.toString())
11            .reduce((el1, el2) -> el1 + ",",
12            .get());
13         System.out.println(csvWeights );
14     }
15 }
16

```

```

1
2 25,32,45,66,77
3

```

Example 3: Converting a List of unique objects to a map**Key=name -> value=Employee**

```

1 import java.util.Arrays;
2 import java.util.List;
3 import java.util.Map;
4 import java.util.stream.Collectors;
5
6 public class ListToMap {
7
8     public static void main(String[] args) {
9         List<Employee> employees = Arrays.asList
10            new Employee("Sam", 35, "English
11            new Employee("Alice", 42, "Scien
12         // Java 8 FP
13         //Assume that names are unique, and use
14         Map<String, Employee> mapEmployees = emp
15            .collect(C
16            .toMap(emp
17         System.out.println(mapEmployees);
18     }
19
20     //inner pojo class
21     static class Employee{
22         private String name;
23         private int age;
24         private String department;
25
26         public Employee(String name, int age, St

```

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 In
- ⊞ Low Latency (7)
- ⊞ Memory Management
- ⊞ 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)

```

27         super();
28         this.name = name;
29         this.age = age;
30         this.department = department;
31     }
32
33     //getters & setters
34     public String getName() {
35         return name;
36     }
37
38     public int getAge() {
39         return age;
40     }
41
42     public String getDepartment() {
43         return department;
44     }
45
46     //toString
47     @Override
48     public String toString() {
49         return "Employee [name=" + name + ",
50     }
51 }
52 }
53
54

```

Output

```

1
2 {Alice=Employee [name=Alice, age=42, department=S
3

```

Example 4: Converting a List of non unique objects to a map

Key=name -> **value**=List<Employee>

```

1 import java.util.Arrays;
2 import java.util.List;
3 import java.util.Map;
4 import java.util.stream.Collectors;
5
6 public class ListToMap {
7
8     public static void main(String[] args) {
9         List<Employee> employees = Arrays.asList
10             new Employee("John", 35, "Englis
11             new Employee("Alice", 42, "Scien
12         // Java 8 FP
13         //Assume that names are NOT unique, and
14         //value will be a List of Employees
15         Map<String, List<Employee>> mapEmployees
16             .collect(C
17         System.out.println(mapEmployees);
18     }
19

```

- [Spring & Hibernate Ti](#)
- [Tools Tutorials \(19\)](#)
- [Other Tutorials \(45\)](#)

Preparing for Java written & coding tests

[open all](#) | [close all](#)

- [♦ Complete the given](#)
- [Can you write code? \(](#)
- [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](#)

```
20 //inner pojo class
21 static class Employee{
22     private String name;
23     private int age;
24     private String department;
25
26     public Employee(String name, int age, St
27         super();
28         this.name = name;
29         this.age = age;
30         this.department = department;
31     }
32
33     //getters & setters
34     public String getName() {
35         return name;
36     }
37
38     public int getAge() {
39         return age;
40     }
41
42     public String getDepartment() {
43         return department;
44     }
45
46     //toString
47     @Override
48     public String toString() {
49         return "Employee [name=" + name + ",
50     }
51 }
52 }
53 }
```

Output

```
1
2 {Alice=[Employee [name=Alice, age=42, department=
3 }
```

Example 5: Converting a Map keys to a List, sorted by values

entrySet -> stream -> sorted -> map -> collect

```
1 import java.util.Comparator;
2 import java.util.HashMap;
3 import java.util.List;
4 import java.util.Map;
5 import java.util.stream.Collectors;
6
7 public class MapToList {
8
9     public static void main(String[] args) {
10         Map<String, String> mapNameFaculty = new
11         mapNameFaculty.put("John", "Maths");
12         mapNameFaculty.put("Sam", "English");
13         mapNameFaculty.put("Alice", "Science");
```

```
14
15 //Convert to a List of names sorted by f
16 List<String> listOfNamesSortedByFaculty
17     .sorted(Comparator.comparing
18     .map(Map.Entry::getKey)
19     .collect(Collectors.toList)
20
21 System.out.println(listOfNamesSortedByFaculty)
22 }
23 }
24
```

Output

```
1
2 [Sam, John, Alice]
3
```

Now, to sort by “length” of the faculty name

```
1 import java.util.HashMap;
2 import java.util.List;
3 import java.util.Map;
4 import java.util.stream.Collectors;
5
6 public class MapToList {
7
8     public static void main(String[] args) {
9         Map<String, String> mapNameFaculty = new
10         mapNameFaculty.put("John", "Maths");
11         mapNameFaculty.put("Sam", "English");
12         mapNameFaculty.put("Alice", "Science");
13
14         //Convert to a List of names sorted by f
15         List<String> listOfNamesSortedByFaculty
16             .sorted((e1,e2) -> e1.get
17             .map(Map.Entry::getKey)
18             .collect(Collectors.toList)
19
20         System.out.println(listOfNamesSortedByFaculty)
21     }
22 }
23
```

Output

```
1
2 [John, Alice, Sam]
3
```

FP is more memory intensive than imperative programming because in FP data is not overwritten, but sequences of versions are created to represent the data modification.

Nowadays, both the memory & disk are cheap. FP gives the programmer a lot more control about wrangling the data. Very useful in big data for functions like map, flatMap, reduce, combine, sort, etc. The “map” applies a **given function** to every data record on different machines in a cluster. This can be run in **parallel**. The “reduce” combines the individual results on different machines “by “applying a given function” to every data to reach a final result.

Example 6: Sum the list of numbers across the Hadoop cluster with Apache Spark

```
1
2 SparkConf conf = new SparkConf().setAppName("Sequ
3 JavaSparkContext sc = new JavaSparkContext(conf);
4
5 List<Integer> data = Arrays.asList(1, 2, 3, 4, 5)
6 JavaRDD<Integer> distData = sc.parallelize(data);
7 distData.reduce((a, b) -> a + b)
8
```

Example 7: Counting the number of blank lines in a given text input with Apache Spark.

More detailed explanation at: [Apache Spark interview questions & answers](#)

```
1
2 JavaRDD<String> lines = sc.textFile("data.txt");
3 final Accumulator<Integer> blankLines = sc.accum
4 JavaPairRDD<String, Integer> counts = lines.flat
5 {
6     if ("".equals(line)) {
7         blankLines.add(1); // increment the
8     }
9     return Arrays.asList(line.split(" "));
10 }.mapToPair(word -> new Tuple2<String, Integer>
11     .reduceByKey((x, y) -> x + y);
12
13 System.out.println("Blank lines count: " + blank
14
```

Popular Posts

♦ [11 Spring boot interview questions & answers](#)

861 views

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

829 views

18 Java scenarios based interview Questions and Answers

448 views

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

407 views

♦ 7 Java debugging interview questions & answers

311 views

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

303 views

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

294 views

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

288 views

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

263 views

8 Git Source control system interview questions & answers

215 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](https://www.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.

◀ 04: Convert XML file To an Avro File with Apache Spark – writing & reading

05: Convert XML file To an Avro File with avro-maven-plugin & Apache Spark ▶

Posted in FP, 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.