

Industrial strength Java/JEE Career Companion to open more doors

[Home](#)
[Java FAQs](#)
[600+ Java Q&As](#)
[Career](#)
[Tutorials](#)
[Member](#)
[Why?](#)
[Can u Debug?](#)
[Java 8 ready?](#)
[Top X](#)
[Productivity Tools](#)
[Judging Experience?](#)

[Home](#) > [member-paid](#) > 03: Q13 – Q18 Scala FP combinators interview questions & answers

03: Q13 – Q18 Scala FP combinators interview questions & answers

Posted on [July 28, 2016](#) by [Arulkumaran Kumaraswamipillai](#)

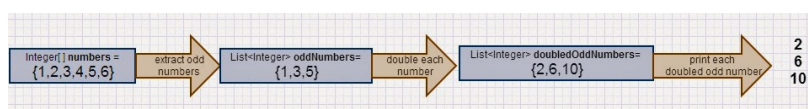


Tweet



Q13. What are the different approaches in Scala to solve the following task?

Given numbers 1 to 6, and you need to extract out the odd numbers and double them, and finally print the result.



Java functional programming focusing on transforming data

A13. There are 3 ways to solve the problem.

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

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

[Ice Breaker Interview](#)

[Core Java Interview C](#)

[JEE Interview Q&A \(3](#)

[Pressed for time? Jav](#)

[SQL, XML, UML, JSC](#)

[Hadoop & BigData Int](#)

[Java Architecture Inte](#)

[Scala Interview Q&As](#)

[Scala way of coding](#)

[01: Coding Scala](#)

[01: ♥ Q1 – Q6 Scala](#)

[02: Q6 – Q12 Scala](#)

[03: Q13 – Q18 Scala](#)

[04: Q19 – Q26 Scala](#)

[05: Q27 – Q32 Scala](#)

[06: Q33 – Q40 Scala](#)

[07: Q41 – Q48 Scala](#)

[08: Q49 – Q58 Scala](#)

[09: Q59 – Q65 High](#)

[10: Q66 – Q70 Pat](#)

[11: Q71 – Q77 – S](#)

- 1) Loops (e.g. for, while, do-while)
- 2) Recursion
- 3) Combinators (e.g. map, foreach, filter, flatMap, foldLeft, etc)

Loop

```

1
2 package com.mytutorial.problem1
3
4 object LoopInScala extends App {
5
6     val myList = List(1,2,3,4,5,6)
7
8     for(x <- myList) {
9         if(x % 2 != 0) {
10             println(x * 2)
11         }
12     }
13 }
14

```

Recursion

`::` means “Common method or object”. “`::`” in this expression is the method of the class List.

tail method returns a list consisting of all elements except the first.

x represents the “head”, which is the first element.

Nil represents an empty list.

x :: tail means x is the head (i.e. first element of the list), and tail is “all elements except the first”.

```

1
2 package com.mytutorial.problem1
3
4 object RecursionInScala extends App {
5
6     val myList = List(1, 2, 3, 4, 5, 6)
7
8     def printDoubledOddNums(listOfNums: List[Int])
9         case Nil =>
10         case x :: tail => { if (x % 2 != 0) println(
11     }

```

12: Q78 – Q80 Rec
[Spring, Hibernate, & I](#)
[Testing & Profiling/Sa](#)
[Other Interview Q&A 1](#)
[Free Java Interview](#)

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 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 \(1\)](#)

```

12
13     printDoubledOddNums(myList);
14 }
15

```

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

Functional Combinators

Some of the functional combinators in Scala are like “**map**”, which applies the supplied function to each element and return the result to a new list. The “**foreach**” functional combinator does the same things the “map”, but does not return a value. **filter** removes any element that returns false to the supplied predicate (e.g. $x \% 2 \neq 0$). There are many other functional combinators like flatten, foldLeft, foldRight, zip, find, partition, dropWhile, etc.

```

1
2     package com.mytutorial.problem1
3
4     object FunctionalCombinatorInScala extends App {
5
6         val myList = List(1, 2, 3, 4, 5, 6)
7
8         myList.filter { x => x % 2 != 0 } //returns a
9             .foreach { x => println( x * 2) } //pr
10
11     }
12

```

Q14. Which approach among loops, iteration, and higher-order combinators will you favor in which scenario?

A14. As a general rule of thumb, favor “higher-order combinators” over the other two. Looping should only be considered in performance critical sections, and recursion should be considered in some edge cases where combinators cannot be used.

Combinators are not only much easier to read, but also promotes shorter code and fewer mistakes. Combinators allow you to combine small functions into big functions.

Q15. What will be the output of the following code snippet?











```

1
2     package com.mytutorial.problem1
3

```

100+ Java pre-interview 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](#)

```
4 object MapVsFlatMap extends App {  
5  
6   val myList = List(List(1, 2), List(4, 5))  
7  
8   println(myList.map(x => x))  
9  
10  println(myList.flatMap(x => x))  
11 }  
12
```

A15. The “flatMap” flattens the list of lists to a single list.

```
1  
2 List(List(1, 2), List(4, 5))  
3 List(1, 2, 4, 5)  
4
```

Q16. Can the following code snippet be re written with the “foldLeft” combinator?

```
1  
2 package com.mytutorial.problem1  
3  
4 object DoubleAndSumNumbers extends App {  
5  
6   val myList = List(3, 4, 5);  
7  
8   val result = myList.map { x => x * 2 }  
9                       .reduce { (x, y) => x + y }  
10  
11  println("result = " + result)  
12 }  
13
```

A16. Yes. The above code doubles each number in the “map” function and then “reduce” all the numbers i.e. (6, 8, 10) by adding each number to the next number:

Pass 1: 14, 10

Pass 2: 24

Using “foldLeft”

```
1  
2 package com.mytutorial.problem1  
3  
4 object DoubleAndSumNumbers extends App {  
5  
6   val myList = List(3, 4, 5);  
7
```

```

8   val result = myList.map(x => x * 2 ).foldLeft(
9   println("result = " + result)
10  }
11

```

It starts with 0 as the initial result and then sums the result with the next doubled number.

Pass 1: 0 + 6

Pass 2: 6 + 8

Pass 3: 14 + 10

Q17. How will you go about combining two functions?

A17. You can use the functional combinators like “**combine**” and “**andThen**”. Here is an example that is used with a “Future” object, which is used for asynchronous (i.e. non blocking) processing.

```

1
2   package com.mytutorial.problem1
3
4   import scala.concurrent.Future
5   import scala.util.Random
6   import scala.concurrent.ExecutionContext.Implicits
7   import scala.actors.threadpool.TimeUnit
8   import scala.util.{Failure, Success}
9
10  object FutureCombinator extends App {
11
12    println("Future is an object holding a value")
13
14    val f = Future {
15      TimeUnit.SECONDS.sleep(5)
16      10
17    }
18
19    f.map { x => println(x) }.andThen {
20      case Success(x) => println("Successfully pri
21      case Failure(x) => x.printStackTrace();
22    }. andThen {
23      case Success(x) => println("Wow I am next")
24      case Failure(x) => x.printStackTrace();
25    }
26
27    println("Future become available asynchronousl
28    TimeUnit.SECONDS.sleep(10) //keep the JVM aliv
29  }
30

```

Output

```

1

```

```
2 Future is an object holding a value
3 Future become available asynchronously
4 10
5 Successfully printed the number
6 Wow I am next
7
```

Q18. How will you split the following list into odd and even numbers>

```
1
2 val numbers = List(1, 2, 3, 4, 5, 6)
3
```

A18. You can use the “partition” function.

```
1
2 package com.mytutorial.problem1
3
4 object PartitionScala extends App {
5
6     val numbers = List(1, 2, 3, 4, 5, 6)
7
8     val result = numbers.partition { x => x % 2 !=
9
10    println(result) // (List(1, 3, 5),List(2, 4, 6
11 }
12
```

Popular Member Posts

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

850 views

♦ [Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers](#)

769 views

001A: ♦ [7+ Java integration styles & patterns interview questions & answers](#)

399 views

[18 Java scenarios based interview Questions and Answers](#)

387 views

♦ [7 Java debugging interview questions & answers](#)

308 views

01b: ♦ [13 Spring basics Q8 – Q13 interview questions & answers](#)

305 views

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

297 views

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

293 views

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

246 views

001B: ♦ Java architecture & design concepts
interview questions & answers

204 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. **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](https://www.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](#)

◀ 01b: ♦ 15+ Hibernate basics Q8 – Q15 interview questions & answers

YAML with Java using the SnakeYaml library tutorial ▶

Posted in member-paid, Scala Interview Q&As

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.