11: Q71 – Q77 – Scala type parametrization & variance interview Q&As

Posted on September 24, 2016 by Arulkumaran Kumaraswamipillai

Q71. What is type parameterization?

A71. **Type parameterization** allows you to write **generic** classes. For example, a List[A] is a generic type and take a type parameter of "A", which can be a String, Integer, Double, Animal, or any other user defined class types.

Q72. Why is the following example a naive approach to get type safety (E.g. An Lion can be only added to a Lion cage)?

```
1
2  object Circus extends App {
```

```
3    val lionCage = new LionCage(new Lion())// can
4    val elephantCage = new ElephantCage(new Elepha
5
6    println(elephantCage.hasAnimal(new Elephant())
7    println(elephantCage.hasAnimal(new Lion())) //
8  }
9
10 //animals
11 abstract class Animal { def name: String }
12 class Elephant extends Animal { def name = "Elep
13 class Lion extends Animal { def name = "Lion" }
14
15 //cages
16 abstract class Cage {
17   def hasAnimal(anAnimal: Animal): Boolean
18 }
19
20 class ElephantCage(thisAnimal: Elephant) extends
21   def hasAnimal(anAnimal: Animal): Boolean = thi
22 }
23
24 class LionCage(val thisAnimal: Lion) extends Cag
25   def hasAnimal(anAnimal: Animal): Boolean = thi
26 }
27
```

A72. In the example shown above, the definition of the two classes ElephantCage and LionCage give additional type safety as you can only pass “Elephant” and “Lion” respectively. In other words you can’t cage an “Elephant” into a “LionCage” and a “Lion” into an “ElephantCage”.

As you start to have more animals, the explosion of class hierarchy to cater for each type of “Animal” becomes hard to maintain. In order to avoid this issue, Scala allow parametrized (aka Generic) classes. The type parameters must be declared in the definition of a class, and must be bound to a real type when instantiating that class. For example, define it as “Cage[A <: Animal]”, which means any sub types of “Animal” in the class definition, and bound at runtime with “new Cage[Elephant](new Elephant())” as shown below. Just a single “Cage” class.

```
1
2  object Circus extends App {
3    val lionCage = new Cage[Lion](new Lion)// can
4    val elephantCage = new Cage[Elephant](new Elep
5
6    println(elephantCage.hasAnimal(new Elephant())
7    println(elephantCage.hasAnimal(new Lion())) //
8  }
9
10 //animals
```

12: Q78 – Q80 Rec
Spring, Hibernate, & N
Testing & Profiling/Sa
Other Interview Q&A f
Free Java Interviev

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 In
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)

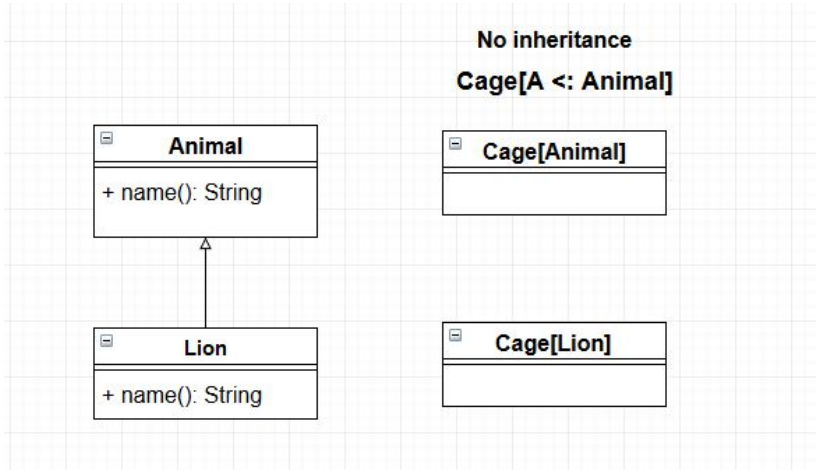
```
11 abstract class Animal { def name: String }
12 class Elephant extends Animal { def name = "Elep
13 class Lion extends Animal { def name = "Lion" }
14
15 //Cage with type parametrization
16 class Cage[A <: Animal](thisAnimal: A) {
17   def hasAnimal(anAnimal: Animal): Boolean = thi
18 }
19
```

Q73. What is type variance? What are the different types of variances>

A73. **Variance** defines **inheritance** relationships of parameterized types. For example, if “Elephant” is a subclass (aka subtype) of an “Animal”, is “List[Elephant]” subclass (aka subtype) of List[Animal]? It depends on how you define the variance. There are 3 types of variances **1)** Invariant **2)** Covariant **3)** Contravariant.

Invariant

When there is no variance annotation as in “Cage[+A]” (i.e. + means Covariant) or “Cage[-A]” (i.e. – means Contravariant), it is known as invariant, which means “Cage[Lion]” and “Cage[Animal]” don’t inherit from each other. This is the assumption the Scala compiler makes when there is no variance annotations like “+” or “-“. When it is invariant, you can’t assign a an object of type Cage[Lion] to Cage[Animal] as shown below.



Scala Invariant

Spring & HIbernate Tu
Tools Tutorials (19)
Other Tutorials (45)

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

```
1
2 // Illegal: Cage[Lion] is not a subtype of Cage[A
3 var lionCage: Cage[Animal] = new Cage[Lion](new L
4
```

The above code “Cage[A <: Animal]….” is invariant.

Q74. What change will you have to make so that Cage[Lion] inherits from Cage[Animal]?
Q74. Make it covariant by adding the “+” annotation as in “Cage[+A <: Animal]….”

Covariant

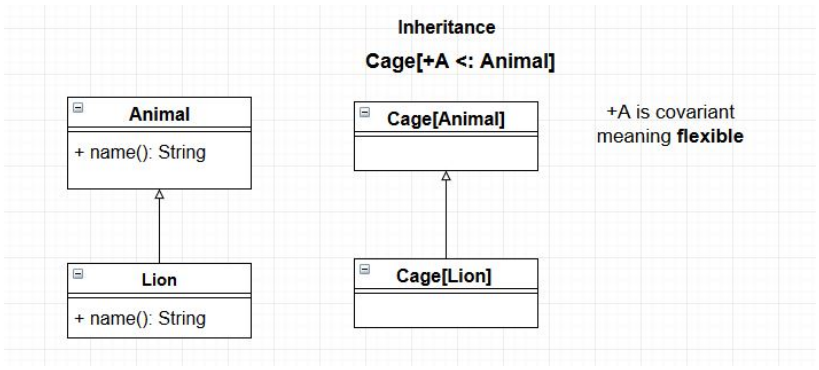
The assignment of a Cage[Lion] to a variable of type Cage[Animal] is now possible, since the covariance annotation “+A” will make “Cage[Lion]” a subclass of “Cage[Animal]”. Here is the code with “Cage[+A <: Animal]”

```
1
2  object Circus extends App {
3    val lionCage: Cage[Animal] = new Cage[Lion](ne
4    val elephantCage: Cage[Animal] = new Cage[Elep
5
6    println(elephantCage.hasAnimal(new Elephant())
7    println(elephantCage.hasAnimal(new Lion())) //
8  }
9
10 //animals
11 abstract class Animal { def name: String }
12 class Elephant extends Animal { def name = "Elep
13 class Lion extends Animal { def name = "Lion" }
14
15 //Cage with type parametrization
16 class Cage[+A <: Animal](thisAnimal: A) {
17   def hasAnimal(anAnimal: Animal): Boolean = thi
18 }
19
```

The “class Cage[+A <: Animal]” makes the following line possible.

```
1
2 var lionCage: Cage[Animal] = new Cage[Lion](new L
3
```

This means:

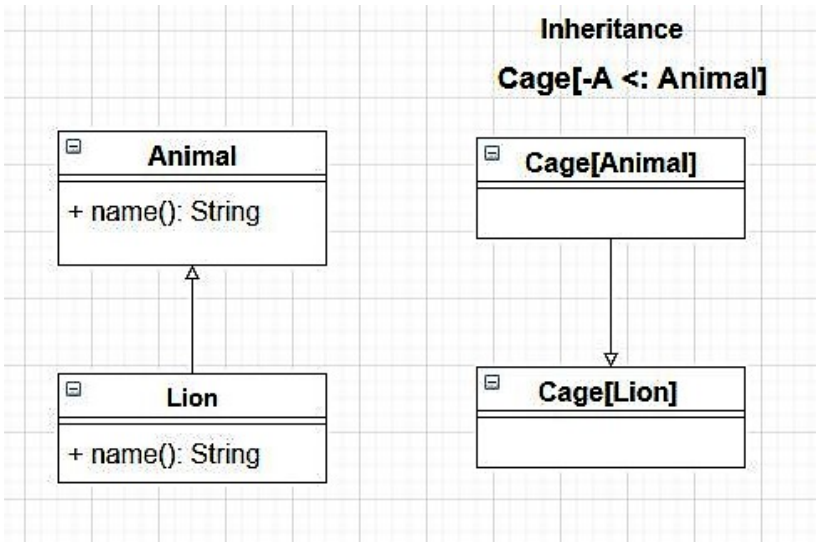


Scala covariant

Contravariant

The assignment of a Cage[Animal] to a variable of type Cage[Lion] is now possible, since the covariance annotation "-A" made "Cage[Animal]" a subclass of "Cage[Lion]".

```
1
2 var lionCage: Cage[Lion] = new Cage[Animal](new L
3
```



Scala Contravariant

```
1
2  object Circus extends App {
3    val lionCage: Cage[Lion] = new Cage[Animal](ne
4    val elephantCage: Cage[Lion] = new Cage[Animal
5
6    println(elephantCage.hasAnimal(new Elephant())
```

```
7    println(elephantCage.hasAnimal(new Lion())) //
8  }
9
10 //animals
11 abstract class Animal { def name: String }
12 class Elephant extends Animal { def name = "Elep
13 class Lion extends Animal { def name = "Lion" }
14
15 //Cage with type parametrization
16 class Cage[-A <: Animal](thisAnimal: A) {
17   def hasAnimal(anAnimal: Animal): Boolean = thi
18 }
19
```

Note:

```
1
2 val lionCage: Cage[Lion] = new Cage[Animal](new E
3
```

Q75. What is the purpose of covariant and contravariant types?
A75. In a nutshell, the covariant and contrvariant makes your code more **extendable**. The type parameters allow you to extend the generic classes by making it more flexible.

Q76. Does Scala have type erasure as in Java?
A76. Yes. Scala parametrized types loose the type information after compilation in the same way as Java does.

Q77. What are different type parameter modifiers in Scala?
A77.

Cage[T]	**T** is of any type	**Invariance**
Cage[T <: Animal]	**T** is sub type of an Animal	**upper bound**
Cage[T >: Animal]	**T** is super type of an Animal	**lower bound**
Cage[+T]	**Cage[Lion]** is a sub type of **Cage[Animal]**	**Covariance**

Cage[-T]	**Cage[Animal]** is a sub type of **Cage[Lion]**	**Contravariance**
Cage[T <% Mamal[T]]	There is an implicit conversion from T to Mamal[T]	**View bound**

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

294 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 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**

‹ 10: Q66 – Q70 Pattern matching in Scala Interview Q&As

12: Q78 – Q80 Recursion in Scala Q&As explained with diagrams ›

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

 ↑