Register | Login | Logout | Contact Us

Java-Success.com

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

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 › constructors-methods-initializers › Java initializers, constructors, regular methods and static factory methods – when to use them with examples.

Java initializers, constructors, regular methods and static factory methods – when to use them with examples.

Posted on November 22, 2014 by Arulkumaran Kumaraswamipillai — 2 Comments ↓

0 Like Share Tweet 0 G+1 Share

Q1. What are “**static initializers**” or “**static blocks with no function names**” in Java?
A1. When a class is loaded, all blocks that are declared static and don’t have function name (i.e. static initializers) are

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

open all | close all
- Ice Breaker Interview
  - 01: ♦ 15 Ice breake
  - 02: ♥♦ 8 real life ex
  - 03: ♦10+ Know you
  - 04: Can you think c
  - 05: ♥ What job inte
  - 06: ► Tell me abou
  - 07: ♥ 20+ Pre inter
- Core Java Interview C
  - Java Overview (4)
    - 01: ♦ ♥ 17 Java c

executed even before the constructors are executed. As the name suggests they are typically used to initialize static fields.

```
public class StaticInitilaizer {
   public static final int A = 5;
   public static final int B;
   //note that since final above line cannot do

   //Static initializer block, which is executed

   static {
      if(A == 5)
         B = 10;
      else
        B = 5;
   }

   public StaticInitilaizer(){}  //constructor i
}

```

The following code gives an Output of A=5, B=10.

```
public class Test {
    System.out.println("A =" + StaticInitilaizer.
}

```

Q2. How will you initialize an instance variable say *dueDate* to first day of next month?
A2. Like static initializers, you can use an initializer block for instance variables. Initializer blocks for instance variables look just like static initializer blocks, but without the 'static' keyword.

```
public class Initilization {

    private Date dueDate;

    //initializer block
    {
        Calendar cal = GregorianCalendar.getInst
        cal.add(Calendar.MONTH, 1);
        cal.set(Calendar.DAY_OF_MONTH, 1);
        dueDate = cal.getTime( );          //dueD
    }

    //...

    public static void main(String[ ] args) {
        Initilization init = new Initilization(
```

02: ♥♦ Java Com
03: ♦ 9 Core Jav
04: ♦ Top 10 mos
Data types (6)
01: Java data typ
02: ♥♦ 10 Java S
03: ♦ ♥ Java auto
04: Understandin
05: Java primitiv
Working with Da
constructors-metho
Java initializers,
Reserved Key Wor
♥♦ 6 Java Modifi
Java identifiers
Classes (3)
♦ Java abstract c
♦ Java class loa
♦ Java classes a
Objects (8)
▶ Beginner Java
♥♦ HashMap & H
♦ 5 Java Object
♦ Java enum inte
♦ Java immutabl
♦♥ Object equals
Java serializatio
Mocks, stubs, do
OOP (10)
♥ Design princip
♦ 30+ FAQ Java
♦ Why favor com
08: ♦ Write code
Explain abstracti
How to create a
Top 5 OOPs tips
Top 6 tips to go a
Understanding C
What are good r
GC (2)
♦ Java Garbage

```
        System.out.println("dueDate = " + init.d
    }
}

```

Constructors Vs Regular Methods

Q3. What is the difference between constructors and other regular methods? What happens if you do not provide a constructor? Can you call one constructor from another? How do you call the superclass' constructor?
A3.

Constructors	**Regular meth**

Constructors must have the same name as the class name and cannot return a value. The constructors are called only once per creation of an object while regular methods can be called many times. E.g. for a ***Pet.class***
`public Pet() {} // constructor`
	Regular method
	number of times
	`// regular`
	`public voic`
	Note: method na
	differentiate a cc
	convention is to
	lowercase like:
	`// regular`
	`public Pet`

Q4. What happens if you do not provide a constructor?
A4. Java does not actually require an explicit constructor in the class description. If you do not include a constructor, the Java compiler will create a default constructor in the byte code with an empty argument. This default constructor is equivalent to the explicit "***Pet( ){ }***". If a class includes one or

03: Java GC tun
Generics (5)
♥ Java Generics
♥ Overloaded m
♦ 12 Java Gener
♦ 7 rules to reme
3 scenarios to ge
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
IO (7)
♥ 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)
01: ♥♦ 15 Beginr
02: ♥♦ 10+ Java
03: ♦ More Java
04: ♦ 6 popular J
05: ♦ How a thre
06: ♦ 10+ Atomic
07: 5 Basic multi
08: ♦ ThreadLoc
09: Java FutureT
10: ♦ ExecutorSe
Java ExecutorSe
Producer and Cc
Algorithms (5)
♦ Splitting input t
♦ Tree traversal
♥ ♦ Java coding

more explicit constructors like “*public Pet(int id)*” or “***Pet(){ }***” etc, the java compiler does not create the default constructor “***Pet( ){ }***”.

Q5. Can you call one constructor from another?
A5. Yes, by using ***this( )*** syntax. E.g.

```
public Pet(int id) {
    this.id = id;                     // “this”
}
public Pet (int id, String type) {
    this(id);                         // calls c
    this.type = type;                // ”this” m
}

```

Q6. How to call the superclass constructor?
A6. If a class called “*SpecialPet*” extends your “*Pet*” class then you can use the keyword “*super()*” to invoke the superclass’s constructor. E.g.

```
public SpecialPet(int id) {
     super(id);                      //must be t
}

```

To call a regular method in the super class use: “***super****.myMethod();*”. This can be called at any line. Some frameworks based on JUnit add their own initialization code, and not only do they need to remember to invoke their parent’s *setUp* method, you, as a user, need to remember to invoke them after you wrote your initialization code:

```
public class DBUnitTestCase extends TestCase {
  public void setUp() {
    super.setUp();
    // do my own initialization
  }
}

public void cleanUp() throws Throwable
{
     try {
       … // Do stuff here to clean up your objec
     }
     catch (Throwable t) {}
     finally{
         super.cleanUp(); //clean up your parent
```

Searching algori
Swapping, partiti
Annotations (2)
8 Java Annotatic
More Java anno
Collection and Data
♦ Find the first no
♦ Java Collectior
♥ Java Iterable \
♥♦ HashMap & H
♦ Sorting objects
02: ♦ Java 8 Stre
04: Understandir
4 Java Collectior
If Java did not ha
Java 8: Different
Part-3: Java Tree
Sorting a Map by
When to use whi
Differences Betwee
♥ Java Iterable \
♦ Multithreading
♦ Why do Proxy,
Core Java Modif
Differences betw
Java Collection i
Event Driven Progr
Event Driven Pro
Event Driven Pro
Exceptions (2)
♦ Java exceptior
Top 5 Core Java
Java 7 (2)
Java 7 fork and j
Java 7: Top 8 ne
Java 8 (24)
01: ♦ 19 Java 8 I
02: ♦ Java 8 Stre
03: ♦ Functional
04: ♥♦ Top 6 tips
04: Convert Lists

```
                                 // super.regularMethod
        }
}

```

Q7. Why do super(..) and this(..) calls need to be in the first line of a constructor?
A7. The parent class' constructor needs to be called before the subclass' constructor. This will ensure that if you call any methods on the parent class in your constructor, the parent class has already been set up correctly.

In cases where a parent class has a default constructor the call to super is inserted for you automatically by the compiler. Enforcing super to appear first, enforces that constructor bodies are executed in the correct order. *Object –> Pet –> SuperPet*

The compiler also forces you to declare ***this(..)*** as the first statement within a constructor, otherwise, you will get compile-time error.

Q8. Can constructors have private access modifiers? If yes, can you give an example?
A8. Yes**. Singleton** (i.e design pattern) classes use **private constructors** as shown below**.**

```
public final class MySingletonFactory {

    private static final MySingletonFactory inst

    private MySingletonFactory( ){}

    public static MySingletonFactory getInstance
        return instance;
    }
}

```

Use cases for **private constructors**:

- The classes with a private constructor cannot be extended from outside even if not declared as final.

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)
♦ Java Garbage
01: jvisualvm to
02: jvisualvm to
05: Java primitiv
06: ♦ 10+ Atomic
5 JMX and MBea
Reactive Programn
07: Reactive Pro
10: ♦ ExecutorSe
3. Multi-Threadir
Swing & AWT (2)
5 Swing & AWT i
Q6 – Q11 Swing
JEE Interview Q&A (3
JEE Overview (2)
♦ 8 Java EE (aka
Java EE intervie
Web basics (8)
01: ♦ 12 Web ba
02: HTTP basics
03: Servlet interv

- The classes with a private method cannot be invoked from outside. Only the factory methods within that class like *getInstance( )*, *deepCopy(…),* etc can access a private constructor.

Q9. What are **static factory methods** in Java?
A9. The **factory method pattern** is a way to encapsulate object creation. Without a factory method, you would simply call the class' constructor directly:

```
Pet p = new Pet( );

```

With this pattern, you would instead call the factory method:

```
Pet p = Pet.getInstance();


```

The constructors are marked private, so they cannot be called except from inside the class, and the factory method is marked as static so that it can be called without first having an object.

Java API have many factory methods like *Calendar.getInstance( )*, *Integer.valueOf(5)*, *DriverManager.getConnection( )*, *Class.forName( )*, etc.

Q10. What are the benefits of static factory methods over using constructors directly?
A10.

- Factory can choose from many subclasses (or implementations of an interface) to return. This allows the caller to specify the behavior desired via parameters, without having to know or understand a potentially complex class hierarchy.
- The factory can apply the fly weight design pattern to cache objects and return cached objects instead of

04: JSP overviev
05: Web patterns
06: ♦ MVC0, MV
07: When to use
08: Web.xml inte
WebService (11)
01: ♥♦ 40+ Java
02: ♦ 6 Java RES
03: ♥ JAX-RS hc
04: 5 JAXB inter
05: RESTFul We
06: RESTful Wel
07: HATEOAS R
08: REST constr
09: 11 SOAP We
10: SOAP Web S
11: ♥ JAX-WS hc
JPA (2)
10: Spring, Java
8 JPA interview c
JTA (1)
JTA interview Q&
JDBC (4)
♦ 12 FAQ JDBC
JDBC Overview
NamedParamete
Spring, JavaCon
JMS (5)
♦ 16 FAQ JMS ir
Configuring JMS
JMS versus AMC
Spring JMS with
Spring JMS with
JMX (3)
5 JMX and MBe
Event Driven Prc
Yammer metrics
JNDI and LDAP (1)
JNDI and LDAP
Pressed for time? Jav
Job Interview Ice B

creating a new object every time. In other words, objects can be pooled and reused. This is the reason why you should favor using *Integer.**valueOf**(6)* as opposed to **new** *Integer*(6).

- The factory methods have more meaningful names than the constructors. For example, *getInstance( )*, *valueOf( )*, *getConnection( ), deepCopy( )*, etc.

Here is a factory method example to deeply clone a list of objects.

```
public static List<Car> deepCopy(List<Car> listC
     List<Car> copiedList = new ArrayList<Car>(1
     for (Car car : listCars) {
         Car carCopied = new Car();
         carCopied.setColor((car.getColor()));
         copiedList.add(carCopied);
    }
    return copiedList;
}

```

Popular Posts

♦ 11 Spring boot interview questions & answers

856 views

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

825 views

18 Java scenarios based interview Questions and Answers

447 views

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

400 views

♦ 7 Java debugging interview questions & answers

311 views

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

301 views

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

292 views

01: ♦ 15 Ice brea
02: ♥♦ 8 real life
03: ♦10+ Know y
⊟ FAQ Core Java Jol
♥♦ Q1-Q10: Top
♦ Q11-Q23: Top
♦ Q24-Q36: Top
♦ Q37-Q42: Top
♦ Q43-Q54: Top
01: ♥♦ 15 Beginr
02: ♥♦ 10+ Java
⊟ FAQ JEE Job Inter
♦ 12 FAQ JDBC
♦ 16 FAQ JMS ir
♦ 8 Java EE (aka
♦ Q01-Q28: Top
♦ Q29-Q53: Top
01: ♦ 12 Web ba
06: ♦ MVC0, MV
JavaScript mista
JavaScript Vs Ja
JNDI and LDAP
JSF interview Q&
JSON interview
⊟ FAQ Java Web Ser
01: ♥♦ 40+ Java
02: ♦ 6 Java RES
05: RESTFul We
06: RESTful Wel
09: 11 SOAP We
⊟ Java Application Ar
001A: ♦ 7+ Java
001B: ♦ Java arc
04: ♦ How to go
⊟ Hibernate Job Inter
01: ♥♦ 15+ Hiber
01b: ♦ 15+ Hiber
06: Hibernate Fir
8 JPA interview
⊟ Spring Job Intervie
♦ 11 Spring boot

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

286 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 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.

‹ Working with Date and Time in Java

07: 5 Basic multi-threading interview questions & answers ›

Posted in constructors-methods-initializers, member-paid

01: ♥♦ 13 Spring
01b: ♦ 13 Spring
04 ♦ 17 Spring b
05: ♦ 9 Spring B
⊟ Java Key Area Ess
♦ Design pattern
♥ Top 10 causes
♥♦ 01: 30+ Writir
♦ 12 Java desigr
♦ 18 Agile Devel
♦ 5 Ways to debu
♦ 9 Java Transac
♦ Monitoring/Pro
02: ♥♦ 13 Tips tc
15 Security key a
4 FAQ Performa
4 JEE Design Pa
5 Java Concurre
6 Scaling your Ja
8 Java memory i
⊟ OOP & FP Essenti
♦ 30+ FAQ Java
01: ♦ 19 Java 8 I
04: ♥♦ Top 6 tips
⊟ Code Quality Job In
♦ Ensuring code
♦ 5 Java unit tes
⊟ SQL, XML, UML, JSC
⊟ ERD (1)
♦ 10 ERD (Entity
⊟ NoSQL (2)
♦ 9 Java Transac
3. Understanding
⊟ Regex (2)
♥♦ Regular Expr
Regular Express
⊟ SQL (7)
♦ 15 Database d
♦ 14+ SQL interv
♦ 9 SQL scenario
Auditing databas

2 comments on “Java initializers, constructors, regular methods and static factory methods – when to use them with examples.”

Gvvenkat_nj says:

July 11, 2016 at 2:03 am

FYI – somehow an ad has made it to this page. true distraction.. Just FYI

Reply

Arulkumaran Kumaraswamipillai says:

July 11, 2016 at 10:17 am

Removed, and thanks for letting me know.

Reply

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

Deleting records
SQL Subquery ir
Transaction mar
UML (1)
♦ 12 UML intervi
JSON (2)
JSON interview
JSON, Jackson,
XML (2)
XML basics inter
XML Processing
XSD (2)
11 FAQ XSD inte
XSD reuse interv
YAML (2)
YAML with Java
YAML with Sprin
Hadoop & BigData Int
♥ 01: Q1 – Q6 Had
02: Q7 – Q15 Hadc
03: Q16 – Q25 Hac
04: Q27 – Q36 Apa
05: Q37 – Q50 Apa
05: Q37-Q41 – Dat
06: Q51 – Q61 HBa
07: Q62 – Q70 HDI
Java Architecture Inte
♥♦ 01: 30+ Writing
001A: ♦ 7+ Java int
001B: ♦ Java archit
01: ♥♦ 40+ Java W
02: ♥♦ 13 Tips to w
03: ♦ What should I
04: ♦ How to go ab
05: ETL architectur
1. Asynchronous pi
2. Asynchronous pi
Scala Interview Q&As
01: ♥ Q1 – Q6 Sca
02: Q6 – Q12 Scala
03: Q13 – Q18 Sca

- 04: Q19 – Q26 Sca
- 05: Q27 – Q32 Sca
- 06: Q33 – Q40 Sca
- 07: Q41 – Q48 Sca
- 08: Q49 – Q58 Sca
- 09: Q59 – Q65 Hig
- 10: Q66 – Q70 Pat
- 11: Q71 – Q77 – S
- 12: Q78 – Q80 Rec

⊟ Spring, Hibernate, & I
- ⊟ Spring (18)
  - ⊟ Spring boot (4)
    - ♦ 11 Spring bo
    - 01: Simple Sp
    - 02: Simple Sp
    - 03: Spring boo
  - ⊟ Spring IO (1)
    - Spring IO tuto
  - ⊟ Spring JavaConf
    - 10: Spring, Ja
    - Spring, JavaC
    - Spring, JavaC
    - Spring, JavaC
  - 01: ♥♦ 13 Spring
  - 01b: ♦ 13 Spring
  - 02: ► Spring DII
  - 03: ♥♦ Spring DI
  - 04 ♦ 17 Spring b
  - 05: ♦ 9 Spring B
  - 06: ♥ Debugging
  - 07: Debugging S
  - Spring loading p
- ⊟ Hibernate (13)
  - 01: ♥♦ 15+ Hiber
  - 01b: ♦ 15+ Hiber
  - 02: Understandir
  - 03: Identifying ar
  - 04: Identifying ar
  - 05: Debugging H
  - 06: Hibernate Fi
  - 07: Hibernate mi

08: Hibernate au
09: Hibernate en
10: Spring, Java
11: Hibernate de
12: Hibernate cu
AngularJS (2)
♥ 8 AngularJS in
More Angular JS
Git & SVN (6)
♥ Git & Maven fc
♥ Merging Vs rel
♥ Understanding
6 more Git interv
8 Git Source cor
Setting up Cygw
JMeter (2)
♥ JMeter for test
♦ JMeter perforn
JSF (2)
JSF interview Q&
More JSF intervi
Maven (3)
♥ Git & Maven fc
12 Maven intervi
7 More Maven ir
Testing & Profiling/Sa
Automation Testing
♥ Selenium and
Code Coverage (2)
Jacoco for unit te
Maven and Cobe
Code Quality (2)
♥ 30+ Java Code
♦ Ensuring code
jvisualvm profiling (
01: jvisualvm to
02: jvisualvm to
03: jvisualvm to
Performance Testir
♥ JMeter for test
♦ JMeter perforn

- Unit Testing Q&A (2
  - BDD Testing (4)
    - Java BDD (Be
    - jBehave and E
    - jBehave and j
    - jBehave with t
  - Data Access Uni
    - ♥ Unit Testing
    - Part #3: JPA H
    - Unit Test Hibe
    - Unit Test Hibe
  - JUnit Mockito Sp
    - JUnit Mockito
      - Spring Con
      - Unit Testing
    - Part 1: Unit te
    - Part 2: Mockit
    - Part 3: Mockit
    - Part 4: Mockit
    - Part 5: Mockit
  - Testing Spring T
    - Integration Un
    - Unit testing Sp
  - ♦ 5 Java unit tes
  - JUnit with Hamc
  - Spring Boot in ur
- Other Interview Q&A f
  - Finance Domain In
    - 12+ FX or Forex
    - 15 Banking & fin
  - FIX Interview Q&A
    - 20+ FIX basics ir
    - Finding your way
  - Groovy Interview Q
    - Groovy Coding (
      - Cash balance
      - Sum grades G
    - ♥ Q1 – Q5 Groov
    - ♦ 20 Groovy clos
    - ♦ 9 Groovy meta
    - Groovy method (

- Q6 – Q10 Groov
- JavaScript Intervie
- JavaScript Top I
- ♥ Q1 – Q10 J
- ♦ Q11 – Q20 .
- ♦ Q21 – Q30 .
- ♦ Q31 – Q37 .
- JavaScript mi
- JavaScript Vs J
- JavaScript Vs
- Unix Interview Q&
- ♥ 14 Unix intervi
- ♥ Hidden Unix, (
- sed and awk to v
- Shell script inter
- Unix history com
- Unix remoting in
- Unix Sed comm
- ▶ Free Java Intervie
- ▶ Java Integration
- ▶ Java Beginner I
- 02: ▶ Spring DIP, I
- 06: ▶ Tell me abou

As a Java Architect

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

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)

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

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.

↑

Responsive Theme powered by WordPress