

Programming Mitra

(<https://programmingmitra.blogspot.in/>)

A Friend To Java Programming Language And Its Related Frameworks

[HOME \(HTTPS://PROGRAMMINGMITRA.BLOGSPOT.IN/\)](https://programmingmitra.blogspot.in/)

[JAVA \(/SEARCH/LABEL/JAVA\)](/search/label/java)

[JAVA 8 \(/SEARCH/LABEL/JAVA%208\)](/search/label/java%208)

[INTERVIEW QUE ANS \(/SEARCH/LABEL/INTERVIEW%200%26A\)](/search/label/interview%200%26a)

[JPA \(/SEARCH/LABEL/JPA\)](/search/label/jpa)

[SPRING \(/SEARCH/LABEL/SPRING\)](/search/label/spring)

[SOLR \(/SEARCH/LABEL/SOLR\)](/search/label/solr)

[ABOUT \(/P/ABOUT.HTML\)](/p/about.html)

Creating objects through Reflection in Java with Example

posted by Naresh Joshi (<https://plus.google.com/108302142446482002391>) | on May 15, 2016 | 7 comments
(<https://programmingmitra.blogspot.in/2016/05/creating-objects-through-reflection-in-java-with-example.html#comment-form>)

In Java, we generally create objects using the `new` keyword or we use some DI framework e.g. Spring to create an object which internally use Java Reflection API to do so. In this Article, we are going to study the reflective ways to create objects.

There are two methods present in Reflection API which we can use to create objects

1. `Class.newInstance()`
(<https://docs.oracle.com/javase/8/docs/api/java/lang/Class.html#newInstance-->) → Inside `java.lang` package
2. `Constructor.newInstance()`
(<https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Constructor.html#newInstance-java.lang.Object...->) → Inside `java.lang.reflect` package

CONNECT WITH US

 FACEBOOK

([HTTP://WWW.FACEBOOK.COM/PROGRAMMIN
GMITRA](http://www.facebook.com/programmingmitra))

 GOOGLE

([HTTP://PLUS.GOOGLE.COM/B/1035343738405
51473813/COMMUNITIES/10684574262004084
0023](http://plus.google.com/b/103534373840551473813/communities/106845742620040840023))

 LINKEDIN

([HTTPS://WWW.LINKEDIN.COM/IN/NARESH-
JOSHI-89493255?
TRK=NAV_RESPONSIVE_TAB_PROFILE_PIC](https://www.linkedin.com/in/naresh-joshi-89493255?trk=nav_responsive_tab_profile_pic))

POPULAR POSTS

However there are total 5 ways create objects in Java, if you are not aware of them please go through this article 5 Different ways to create objects in Java with Example (<https://programmingmitra.blogspot.in/2016/05/different-ways-to-create-objects-in-java-with-example.html>).

Both `Class.newInstance()` and `java.lang.reflect.Constructor.newInstance()` are known as reflective methods because these two uses reflection API to create the object. Both are not static and we can call earlier one on a class level object while latter one needs constructor level object which we can get by using the class level object.

Class.newInstance()

The `Class` class is the most popular class in Java after the `Object` class. However, this class lies in the `java.lang` package but plays a major role in Reflection API (`java.lang.reflect.*` package).

In order to use `Class.newInstance()` we first need to get the class level instance of that class for which we want to create objects. We can do this by two ways one is writing complete name of the class and appending `.class` to it and another is using `Class.forName()` method, So in below code

`Employee.class` is similar to `(Employee) Class.forName("org.programming.mitra.exercises.Employee")`

Below code demonstrates how we can create objects using `Class.newInstance()`

```
Employee emp = Employee.class.newInstance();
```

Or

```
Employee emp = (Employee) Class.forName("org.programming.mitra.exercises.Employee").newInstance();
```



(<https://programmingmitra.blogspot.in/2017/02/automatic-spring-data-jpa-auditing-saving-CreatedBy-createddate-lastmodifiedby-lastmodifieddate-automatically.html>)

Spring Data JPA Auditing: Saving CreatedBy, CreatedDate, LastModifiedBy, LastModifiedDate automatically (<https://programmingmitra.blogspot.in/2017/02/automatic-spring-data-jpa-auditing-saving-CreatedBy-createddate-lastmodifiedby-lastmodifieddate-automatically.html>)



(<https://programmingmitra.blogspot.in/2016/05/different-ways-to-create-objects-in-java-with-example.html>)

5 Different ways to create objects in Java with Example

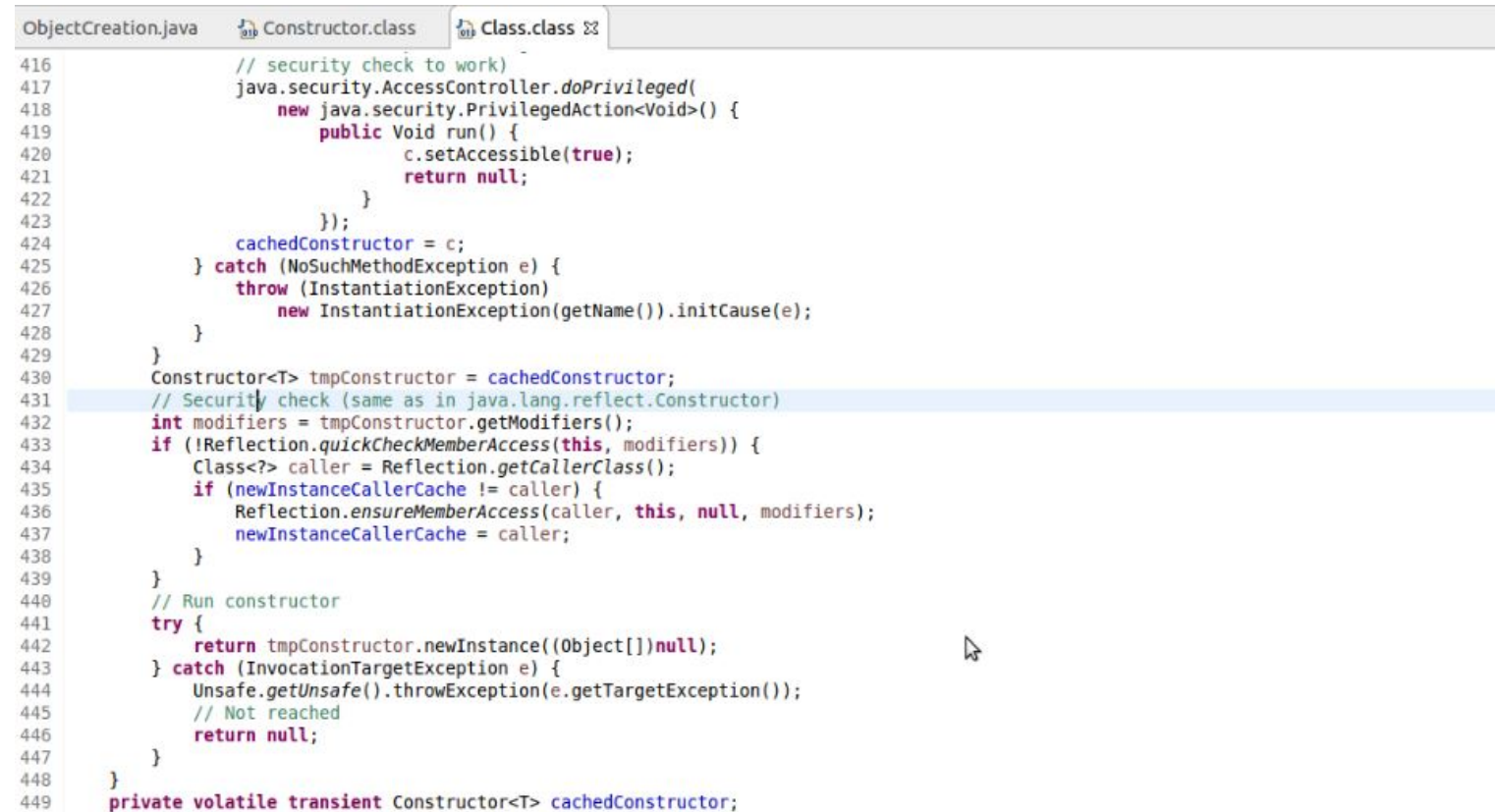
(<https://programmingmitra.blogspot.in/2016/05/different-ways-to-create-objects-in-java-with-example.html>)



(<https://programmingmitra.blogspot.in/2017/01/Java-cloning-copy-constructor-versus-Object-clone-or-cloning.html>)

Java Cloning - Copy Constructor versus Cloning (<https://programmingmitra.blogspot.in/2017/01/Java-cloning-copy-constructor-versus-Object-clone-or-cloning.html>)

`Class.newInstance()` internally itself use the `Constructor.newInstance()` to create the object as we can see in the source code of `Class` class, notice line no 430 and 442 in below image.



```
ObjectCreation.java  Constructor.class  Class.class
416 // security check to work)
417 java.security.AccessController.doPrivileged(
418     new java.security.PrivilegedAction<Void>() {
419         public Void run() {
420             c.setAccessible(true);
421             return null;
422         }
423     });
424     cachedConstructor = c;
425 } catch (NoSuchMethodException e) {
426     throw (InstantiationException)
427         new InstantiationException(getName()).initCause(e);
428 }
429 }
430 Constructor<T> tmpConstructor = cachedConstructor;
431 // Security check (same as in java.lang.reflect.Constructor)
432 int modifiers = tmpConstructor.getModifiers();
433 if (!Reflection.quickCheckMemberAccess(this, modifiers)) {
434     Class<?> caller = Reflection.getCallerClass();
435     if (newInstanceCallerCache != caller) {
436         Reflection.ensureMemberAccess(caller, this, null, modifiers);
437         newInstanceCallerCache = caller;
438     }
439 }
440 // Run constructor
441 try {
442     return tmpConstructor.newInstance((Object[])null);
443 } catch (InvocationTargetException e) {
444     Unsafe.getUnsafe().throwException(e.getTargetException());
445     // Not reached
446     return null;
447 }
448 }
449 private volatile transient Constructor<T> cachedConstructor;
```

(//3.bp.blogspot.com/-VHZe6CxlEzg/WjZ-
OVpVW7I/AAAAAAAAAMo8/HcXh19G87yMDj_rakEfxtMuK5jXdVjbEwCK4BGAYYCw/s1600/creati
ng-objects-using-reflection-in-java.JPG)

Constructor.newInstance()

In order to use `Constructor.newInstance()` method we first need to get constructor object for that class and then we can call `newInstance()` on it to create objects as shown below



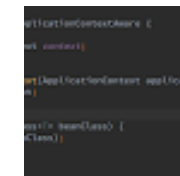
(<https://programmingmitra.blogspot.in/2017/02/automatic-jpa-auditing-persisting-audit-logs-automatically-using-entitylisteners.html>)

JPA Auditing: Persisting Audit Logs
Automatically using EntityListeners
(<https://programmingmitra.blogspot.in/2017/02/automatic-jpa-auditing-persisting-audit-logs-automatically-using-entitylisteners.html>)



(<https://programmingmitra.blogspot.in/2016/05/creating-objects-through-reflection-in-java-with-example.html>)

Creating objects through Reflection in Java with
Example
(<https://programmingmitra.blogspot.in/2016/05/creating-objects-through-reflection-in-java-with-example.html>)



(<https://programmingmitra.blogspot.in/2017/03/AutoWiring-Spring-Beans-Into-Classes-Not-Managed-By-Spring-Like-JPA-Entity-Listeners.html>)

AutoWiring Spring Beans Into Classes Not
Managed By Spring Like JPA Entity Listeners
(<https://programmingmitra.blogspot.in/2017/03/AutoWiring-Spring-Beans-Into-Classes-Not-Managed-By-Spring-Like-JPA-Entity-Listeners.html>)



(<https://programmingmitra.blogspot.in/2016/11/Java-Cloning-Types-of-Cloning-Shallow-Deep-in-Details-with-Example.html>)

```
Constructor<Employee> constructor = Employee.class.getConstructor();
Employee emp3 = constructor.newInstance();
```

It internally use `sun.reflect.ConstructorAccessor` class to get the object, which is Oracle's private API.

Difference between `Class.newInstance()` and `Constructor.newInstance()`

By name, both methods look same but there are differences between them which we are as following

1. **`Class.newInstance()`** can only invoke the no-arg constructor,
`Constructor.newInstance()` can invoke any constructor, regardless of the number of parameters.
2. **`Class.newInstance()`** requires that the constructor should be visible,
`Constructor.newInstance()` can also invoke private constructors under certain circumstances.
3. **`Class.newInstance()`** throws any exception (checked or unchecked) thrown by the constructor,
`Constructor.newInstance()` always wraps the thrown exception with an `InvocationTargetException`.

Due to above reasons `Constructor.newInstance()` is preferred over `Class.newInstance()`, that's why used by various frameworks and APIs like Spring, Guava, Zookeeper, Jackson, Servlet etc.

You can find complete code on this [Github Repository](#)

Java Cloning and Types of Cloning (Shallow and Deep) in Details with Example
(<https://programmingmitra.blogspot.in/2016/11/Java-Cloning-Types-of-Cloning-Shallow-Deep-in-Details-with-Example.html>)



(<https://programmingmitra.blogspot.in/2017/05/how-does-jvm-handle-method-overriding-internally.html>)

How Does JVM Handle Method Overloading and Overriding Internally
(<https://programmingmitra.blogspot.in/2017/05/how-does-jvm-handle-method-overriding-internally.html>)



(<https://programmingmitra.blogspot.in/2016/02/embedding-github-code-into-your-blogger.html>)

Embedding Github code into your Blogger blog
(<https://programmingmitra.blogspot.in/2016/02/embedding-github-code-into-your-blogger.html>)



(<https://programmingmitra.blogspot.in/2016/05/java-build-tools-comparisons-ant-vs.html>)

Java Build Tools Comparisons: Ant vs Maven vs Gradle
(<https://programmingmitra.blogspot.in/2016/05/java-build-tools-comparisons-ant-vs.html>)

CATEGORIES

(<https://github.com/njnareshjoshi/exercises/tree/master/src/org/programming/mitra/exercises>
) and please feel free to provide your valuable feedback.



Naresh Joshi

()

Ant
(<https://programmingmitra.blogspot.in/search/label/Ant>)

Blog Writing
(<https://programmingmitra.blogspot.in/search/label/Blog%20Writing>)

Build Tools
(<https://programmingmitra.blogspot.in/search/label/Build%20Tools>)

Gist
(<https://programmingmitra.blogspot.in/search/label/Gist>)

Github
(<https://programmingmitra.blogspot.in/search/label/Github>)

Gradle
(<https://programmingmitra.blogspot.in/search/label/Gradle>)

How in Java
(<https://programmingmitra.blogspot.in/search/label/How%20in%20java>)

Interview Q&A
(<https://programmingmitra.blogspot.in/search/label/Interview%20Q%26A>)

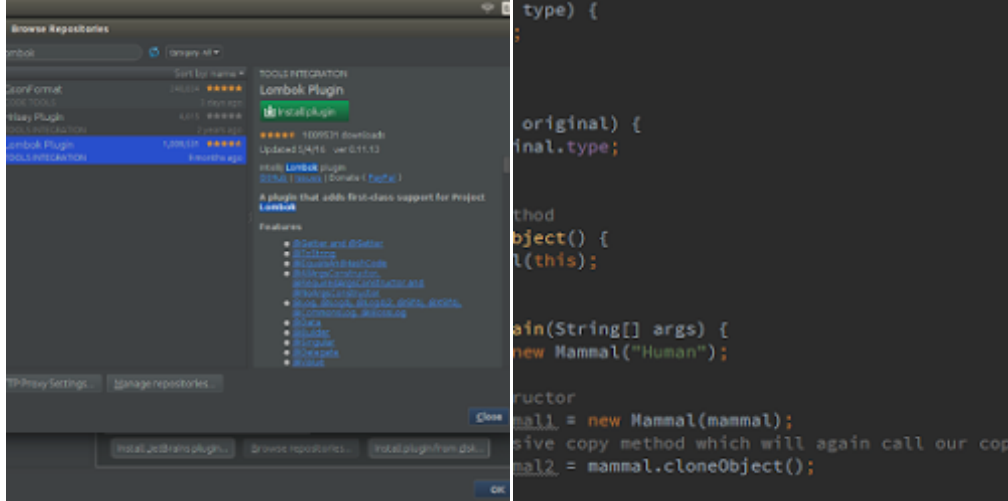
Java
(<https://programmingmitra.blogspot.in/search/label/Java>)

Java 8
(<https://programmingmitra.blogspot.in/search/label/Java%208>)

Java Cloning
(<https://programmingmitra.blogspot.in/search/label/Java%20Cloning>)

JPA
(<https://programmingmitra.blogspot.in/search/label/JPA>)

RELATED POSTS



(<https://programmingmitra.blogspot.com/2017/01/Project-Lombok-The-Boilerplate-Code-Extractor.html>)

Project Lombok : The Boilerplate Code Extractor

(<https://programmingmitra.blogspot.com/2017/01/Project-Lombok-The-Boilerplate-Code-Extractor.html>)

(<https://programmingmitra.blogspot.com/2017/01/java-cloning-why-copy-constructors-are-not-sufficient-or-good.html>)

Java Cloning - Even Copy Constructors Are Not Sufficient
(<https://programmingmitra.blogspot.com/2017/01/java-cloning-why-copy-constructors-are-not-sufficient-or-good.html>)

```

is Person implements Cloneable
private String name;
private int income;
private City city; // deep
private Country country; //

// No @Override, means we
public Person clone() throws
    Person clonedObj = (Pe
    clonedObj.city = this.
    return clonedObj;
}

```

(<https://programmingmitra.blogspot.com/2017/01/Java-cloning-copy-constructor-versus-Object-clone-or-cloning.html>)

Java Cloning – Copy Constructor versus Cloning

(<https://programmingmitra.blogspot.com/2017/01/Java-cloning-copy-constructor-versus-Object-clone-or-cloning.html>)

Lombok
(<https://programmingmitra.blogspot.in/search/label/Lombok>)

Maven
(<https://programmingmitra.blogspot.in/search/label/Maven>)

Reflection API
(<https://programmingmitra.blogspot.in/search/label/Reflection%20API>)

Solr
(<https://programmingmitra.blogspot.in/search/label/Solr>)

Spring
(<https://programmingmitra.blogspot.in/search/label/Spring>)

Spring Boot
(<https://programmingmitra.blogspot.in/search/label/Spring%20Boot>)

Spring Data
(<https://programmingmitra.blogspot.in/search/label/Spring%20Data>)

What in Java
(<https://programmingmitra.blogspot.in/search/label/What%20in%20java>)

Why in Java
(<https://programmingmitra.blogspot.in/search/label/Why%20in%20java>)

NEWSLETTER

Subscribe Our Newsletter

Enter your email address below to subscribe to our newsletter.

Email address

Newer Post

Older Post

SUBSCRIBE



(<https://programmingmitra.blogspot.in/2016/05/Why-Single-Java-Source-File-Can-Not-Have-More-Than-One-public-class.html>)

(<https://programmingmitra.blogspot.in/2016/05/different-ways-to-create-objects-in-java-with-example.html>)



BLOG ARCHIVE

- ▶ 2018 (<https://programmingmitra.blogspot.in/2018/>) (4)
- ▶ 2017 (<https://programmingmitra.blogspot.in/2017/>) (11)
- ▼ 2016 (<https://programmingmitra.blogspot.in/2016/>) (16)
 - ▶ November (<https://programmingmitra.blogspot.in/2016/11/>) (1)
 - ▶ October (<https://programmingmitra.blogspot.in/2016/10/>) (3)
 - ▶ June (<https://programmingmitra.blogspot.in/2016/06/>) (2)
 - ▼ May (<https://programmingmitra.blogspot.in/2016/05/>) (8)
 - JDK and JRE File Structure (<https://programmingmitra.blogspot.in/2016/05/jdk-and-jre-file-structure.html>)
 - Why Single Java Source File Can Not Have More Than...

7 Comments :



1.

YouLoseBellyFat
(<https://www.Blogger.Com/Profile/18175607118884749966>)

1

SEPTEMBER 22, 2016 7:12 PM ([HTTPS://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1474551772057#C1458377983273552424](https://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1474551772057#C1458377983273552424))
c++ codings (<http://cplusplus.happycodings.com/>) by examples

Reply



Naresh Joshi
(<https://www.Blogger.Com/Profile/01073925481525463593>)

1.a

OCTOBER 12, 2016 10:06 AM
([HTTPS://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1476246990665#C6983500766134359363](https://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1476246990665#C6983500766134359363))

Sorry Buddy but I do not good knowledge of C++

(<https://programmingmitra.blogspot.in/2016/05/Why-Single-Java-Source-File-Can-Not-Have-More-Than-One-public-class.html>)



Kiran Vegireddy
(<https://www.Blogger.Com/Profile/02450919583833301502>)

2

FEBRUARY 07, 2017 4:49 PM ([HTTPS://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1486466399024#C2367356646930687100](https://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1486466399024#C2367356646930687100))

excellent material

Creating objects through Reflection in Java with E...
(<https://programmingmitra.blogspot.in/2016/05/creating-objects-through-reflection-in-java-with-example.html>)

5 Different ways to create objects in Java with Ex...
(<https://programmingmitra.blogspot.in/2016/05/different-ways-to-create-objects-in-java-with-example.html>)

Reply



Naresh Joshi
(<https://www.Blogger.Com/Profile/01073925481525463593>)

2.a

FEBRUARY 07, 2017 7:42 PM
([HTTPS://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1486476726919#C469847736496520098](https://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1486476726919#C469847736496520098))

Thanks @Kiran

Plain Old Java Object (POJO) Explained
(<https://programmingmitra.blogspot.in/2016/05/plain-old-java-object-pojo-explained.html>)

Types of References in Java(Strong, Soft, Weak, Ph...
(<https://programmingmitra.blogspot.in/2016/05/types-of-references-in-javastrong-soft.html>)

Java Build Tools Comparisons: Ant vs Maven vs Grad...
(<https://programmingmitra.blogspot.in/2016/05/java-build-tools-comparisons-ant-vs.html>)

Introduction to Spring
(<https://programmingmitra.blogspot.in/2016/05/introduction-to-spring.html>)



HendraWD (<https://www.Blogger.Com/Profile/07341562601728808462>)

3

DECEMBER 22, 2017 7:37 PM ([HTTPS://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1513951669683#C4636677076030574410](https://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1513951669683#C4636677076030574410))

"Constructor.newInstance() can also invoke private constructors under certain circumstances" what kind of circumstances?

- February
(<https://programmingmitra.blogspot.in/2016/02/>) (1)
- January
(<https://programmingmitra.blogspot.in/2016/01/>)

Reply



1.

HendraWD

(<https://www.Blogger.Com/Profile/07341562601728808462>)

DECEMBER 22, 2017 7:52 PM

(<https://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?SHOWCOMMENT=1513952520505#C4374313643310469205>)

This code will do the trick, it will change the access modifier of the constructor to public

```
try {
```

```
    Constructor constructor = Singleton.class.getDeclaredConstructor();
    constructor.setAccessible(true);
    singleton = constructor.newInstance();
```

```
//      Constructor      constructor      =      (Constructor)
Class.forName("com.example.bloder.deck.SingletonActivity").getCon
structor();
// singleton = constructor.newInstance();
} catch (InstantiationException e) {
    e.printStackTrace();
} catch (IllegalAccessException e) {
    e.printStackTrace();
} catch (InvocationTargetException e) {
    e.printStackTrace();
} catch (NoSuchMethodException e) {
    e.printStackTrace();
}
```



2.

Naresh Joshi

(<https://www.Blogger.Com/Profile/01073925481525463593>)

3.a

16/01/)(1)

SUBMIT A QUERY OR SUGGESTION

Name

Email *

Message *

Send

3.b

DECEMBER 25, 2017 3:42 PM

(HTTPS://PROGRAMMINGMITRA.BLOGSPOT.COM/2016/05/CREATING-OBJECTS-THROUGH-REFLECTION-IN-JAVA-WITH-EXAMPLE.HTML?

SHOWCOMMENT=1514196731895#C1471811847461838035)

exactly..

Enter your comment...

 Comment as:

Sign out

Publish

Preview

☐ Notify me

(https://www.blogger.com/comment-iframe.g?

blogID=3832771837877502009&postID=1535359096555315567&blogspotRpcToken=1504689)