

▲ ▼	HP 205A... 61,90 €	HP 62 Ori... 38,99 €	HP 301XL... 62,99 €	HP 301XL... 62,99 €	HP 338 78,99 €	HP 17A O... 77,90 €	HP 126A... 64,90 €	Cartucho... 104,89 €
--	-------------------------------------	---------------------------------------	--------------------------------------	--------------------------------------	---------------------------------	--------------------------------------	-------------------------------------	---------------------------------------


[ANDROIDE](#) [JAVA CENTRAL](#) [JAVA DE ESCRITORIO](#) [JAVA EMPRESARIAL](#) [FUNDAMENTOS DE JAVA](#) [IDIOMAS JVM](#) [DESARROLLO DE SOFTWARE](#) [DI](#)
[Inicio](#) » [Java central](#) » Ejemplo de lista de excepciones de Java

SOBRE ABHINAV NATH GUPTA



Abhinav posee una Maestría en Ciencias de la Computación e Ingeniería del Instituto Nacional de Tecnología de Karnataka. Ha terminado su graduación del Departamento de Tecnología de la Información en el Anand Engineering College, Agra. Durante sus estudios ha estado involucrado en una gran cantidad de proyectos que van desde Redes y Criptografía. Trabaja como ingeniero de desarrollo de software en una empresa de desarrollo de software en Bangalore, donde se dedica principalmente a proyectos basados en Nodejs. Está interesado en la criptografía, la seguridad de los datos, la criptomonedas y la computación en la nube, y ha publicado artículos sobre estos temas. Se le puede contactar en abhi.aec89@gmail.com.

[Twitter](#) [Facebook](#) [LinkedIn](#)

Ejemplo de lista de excepciones de Java

Publicado por: Abhinav Nath Gupta | en Core Java | 11 de octubre de 2019 | 0 comentarios | 172 reproducciones

En este artículo discutiremos la lista de excepciones de Java. Discutiremos cuáles son las excepciones, cuándo ocurren y sus tipos.

1. ¿Qué es una excepción en Java?

La excepción es un mecanismo que Java utiliza para manejar cualquier caso / escenario de uso imprevisto. Básicamente, se produce una excepción cuando ocurre algo inesperado durante la ejecución del código que no está cubierto en ningún bloque de código.

Tabla de contenido

1. ¿Qué es una excepción en Java?
2. Mecanismos de manejo de errores de Java
 - 2.1. Clase arrojable
 - 2.2. Clase de error
 - 2.3. Clase de excepción
3. Lista de excepciones de Java
 - 3.1 CloneNotSupportedException
 - 3.2 Excepción interrumpida
 - 3.3 ReflectiveOperationException
 - 3.3.1 ClassNotFoundException
 - 3.3.2. IllegalAccessException
 - 3.3.3 InstantiationException
 - 3.3.4. NoSuchElementException
 - 3.3.5 NoSuchMethodException
 - 3.4 RuntimeException
 - 3.4.1. ArithmeticException
 - 3.4.2. ArrayStoreException
 - 3.4.3. ClassCastException
 - 3.4.4 EnumConstantNotPresentException
 - 3.4.5. Argumento de excepción ilegal
 - 3.4.5.1. IllegalThreadStateException
 - 3.4.5.2. NumberFormatException

BOLETIN INFORMATIVO

i**163,726** personas con **información** privilegiada

de actualizaciones semanales

blancos gratuitos!

Únase a ellos ahor

[acceso exclusivo](#) a las

en el mundo de Java, así como sobre Android, Scala, Groovy y tecnologías relacionadas.

Dirección de correo electrónico:

 Your email address

Reciba alertas de trabajo desarrollador en su área

[Regístrate](#)

ÚNETE A NOSOTROS



4. Resumen
5. Descargar el código fuente



2. Mecanismos de manejo de errores de Java

En esta sección cubriremos los mecanismos de manejo de errores de Java.

2.1 Clase arrojable

La

`Throwable`

clase es la superclase de todos los errores y excepciones en el lenguaje Java. Solo los objetos que son instancias de esta clase (o una de sus subclases) son arrojados por la máquina virtual Java o pueden ser arrojados por la

`throw`

instrucción Java . Del mismo modo, solo esta clase o una de sus subclases puede ser el tipo de argumento en una

`catch`

cláusula.

Las instancias de dos subclases,

`Error`

y

`Exception`

, se usan convencionalmente para indicar que se han producido situaciones excepcionales. Por lo general, estas instancias se crean recientemente en el contexto de una situación excepcional para incluir información relevante (como los datos de seguimiento de la pila).

2.2 Clase de error

Una

`Error`

es una subclase de

`Throwable`

que indica problemas serios que una aplicación razonable no debería intentar detectar. La mayoría de estos errores son condiciones anormales.

No se requiere que un método declare en su

`throws`

cláusula ninguna subclase

`Error`

que pueda arrojarse durante la ejecución del método pero que no se detecte, ya que estos errores son condiciones anormales que nunca deberían ocurrir.

Por lo tanto,

`Error`

y sus subclases se consideran excepciones no verificadas a los efectos de la verificación de excepciones en tiempo de compilación.

2.3 Clase de excepción

La clase

`Exception`

y las subclases que no descienden

`RuntimeException`

se llaman *excepciones marcadas*.



jerarquía.

3. Lista de excepciones de Java

En esta sección cubriremos todas las clases de excepción definidas en Java.

3.1 CloneNotSupportedException

Lanzado para indicar que el

`clone`

método en cualquier clase ha sido llamado para clonar un objeto, pero que la clase no implementa la

`Cloneable`

interfaz.

Se puede usar para manejar excepciones en torno al

`clone`

método mientras se anula para implementar la clonación personalizada. Más detalles se pueden encontrar aquí .

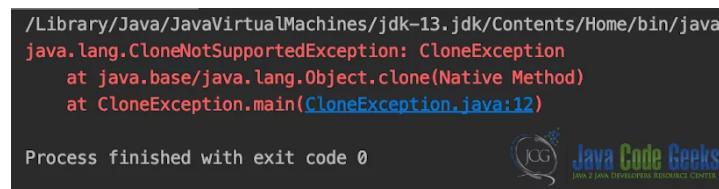
El ejemplo se muestra en el fragmento de código a continuación.

CloneException.java

```

01  public class CloneException {
02      String name;
03
04      CloneException(String name) {
05          this.name = name;
06      }
07
08      public static void main(String[] args) {
09          try {
10              CloneException expOne = new CloneException("CloneException");
11
12              CloneException expTwo = (CloneException) expOne.clone();
13
14              System.out.println(expTwo.name);
15          } catch (CloneNotSupportedException c) {
16              c.printStackTrace();
17          }
18      años
19  }
```

La salida de `CloneException.java` se muestra en la Fig. 1 a continuación.



```
/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java
java.lang.CloneNotSupportedException: CloneException
    at java.base/java.lang.Object.clone(Native Method)
    at CloneException.main(CloneException.java:12)

Process finished with exit code 0
```

Fig. 1. Salida de la clase `CloneException`

3.2 Excepción interrumpida

Lanzado cuando un hilo está esperando, durmiendo u ocupado, y el hilo se interrumpe, ya sea antes o durante la actividad. Más detalles se pueden encontrar aquí .

El ejemplo de esta excepción se muestra en el fragmento de código a continuación.

InterruptedException.java

```

01  public class InterruptedException extends Thread {
02      public void run() {
03          for (int i = 0; i < 10; i++) {
04              try {
05                  Thread.sleep(1000);
06                  if (i == 7) {
07                      throw new InterruptedException(); // to simulate the interrupt exception
08                  }
09              } catch (InterruptedException e) {
10                  System.err.println("Sleep is disturbed. " + e);
11                  e.printStackTrace();
12              }
13          }
14      }
}
```



28 }

La salida de InterruptException.java se muestra en la Fig.2 a continuación.

```
/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java
Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 5
Iteration: 6
Sleep is disturbed. java.lang.InterruptedException
java.lang.InterruptedException
    at InterruptException.run(InterruptException.java:7)
Iteration: 7
Iteration: 8
Iteration: 9
Process finished with exit code 0
```



Fig. 2. Salida de la clase InterruptException

3.3 ReflectiveOperationException

Superclase común de excepciones lanzadas por operaciones reflexivas en la reflexión central. Esta clase de excepción no se usa directamente, se usa a través de sus subclases. Más detalles se pueden encontrar aquí .

3.3.1 ClassNotFoundException

Lanzado cuando una aplicación intenta cargar en una clase a través de su nombre de cadena usando:

- El

método en clase

- El método

o

en clase

pero no se pudo encontrar una definición para la clase con el nombre especificado. Más detalles se pueden encontrar aquí .

Ejemplo de esto se muestra en el fragmento a continuación.

ClassNotFoundException.java

```
1  public class ClassNotFoundException {
2      public static void main(String[] args) {
3          try {
4              Class.forName("example.javacodegeeks.MyInvisibleClass");
5          } catch (java.lang.ClassNotFoundException e) {
6              e.printStackTrace();
7          }
8      }
9 }
```

La salida de ClassNotFoundException.java se muestra en la Fig. 3 a continuación.

```
/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar"
java.lang.ClassNotFoundException: example.javacodegeeks.MyInvisibleClass
at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:694)
at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:178)
at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:521)
at java.base/java.lang.Class.forName0Native Method
at java.base/java.lang.Class.forName(Class.java:332)
at ClassNotFoundException.main(ClassNotFoundException.java:5)
```



IllegalAccessException.java

```

01  public class IllegalAccessException {
02      public static void main(String[] args) throws InstantiationException, java.lang.IllegalAccessException {
03          Class<?> classVar = ClassWithPrivateConstructor.class;
04          ClassWithPrivateConstructor t = (ClassWithPrivateConstructor) classVar.newInstance();
05          t.sampleMethod();
06      }
07  }
08
09  class ClassWithPrivateConstructor {
10      private ClassWithPrivateConstructor() {}
11
12      public void sampleMethod() {
13          System.out.println("Method 'sampleMethod' Called");
14      }
15 }

```

La salida de `IllegalAccessException.java` se muestra en la Fig. 4 a continuación.

```

/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=39048:/Applications/IntelliJ IDEA.app/Contents/lib"
Exception in thread "main" java.lang.IllegalAccessException: class IllegalAccessException cannot access a member of class ClassWithPrivateConstructor with modifiers "private"
        at java.base/java.lang.reflect.AccessibleObject.checkAccess(AccessibleObject.java:113)
        at java.base/java.lang.reflect.Constructor.newInstance(Constructor.java:498)
        at java.base/java.lang.Class.newInstance(Class.java:593)
        at IllegalAccessException.main(IllegalAccessException.java:12)
Process finished with exit code 1

```

Fig 4. Salida de `IllegalAccessException.java`

3.3.3 Excepción de instanciación

Se lanza cuando una aplicación intenta crear una instancia de una clase utilizando el

`newInstance`

método en clase

`class`

, pero el objeto de clase especificado no se puede instanciar. La creación de instancias puede fallar por una variedad de razones que incluyen pero no se limitan a:

- el objeto de clase representa una clase abstracta, una interfaz, una clase de matriz, un tipo primitivo o
- `void`
- la clase no tiene constructor nular

Más detalles se pueden encontrar aquí .

3.3.4 NoSuchFieldException

Señala que la clase no tiene un campo con un nombre especificado.

Los detalles se pueden encontrar aquí .

El ejemplo se muestra en el fragmento de código a continuación.

NoSuchFieldException.java

```

01  class SampleClass {
02      int age;
03
04      SampleClass(int age) {
05          age = age;
06      }
07  }
08
09
10  public class NoSuchFieldException {
11      public static void main(String[] args) {
12          try {
13              String propertyName = "name";
14              SampleClass.class.getDeclaredField(propertyName);
15          } catch (java.lang.NoSuchFieldException e) {
16              e.printStackTrace();
17          }
18      años
19  }

```

La salida se muestra en el código Fig. 5 a continuación.

```

/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java "
java.lang.NoSuchFieldException: name
        at java.base/java.lang.Class.getDeclaredField(Class.java:2412)
        at NoSuchFieldException.main(NoSuchFieldException.java:14)

```

Más detalles se pueden encontrar aquí .

El ejemplo se muestra en el fragmento de código a continuación.

NoSuchMethodFoundException.java

```

01 public class NoSuchMethodFoundException {
02     public static void main(String[] args) {
03         try {
04             String propertyName = "getAge";
05             SampleClass.class.getDeclaredMethod(propertyName);
06         } catch (java.lang.NoSuchMethodException e) {
07             e.printStackTrace();
08         }
09     }
10 }
11 class SampleMethodClass {
12     int age;
13
14     SampleMethodClass(int age) {
15         age = age;
16     }
17 }
18 años

```

La salida se muestra en la Fig. 6 a continuación.

```
/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ
java.lang.NoSuchMethodException: SampleClass.getAge()
at java.base/java.lang.Class.getDeclaredMethod(class.java:2476)
at NoSuchMethodFoundException.main(NoSuchMethodFoundException.java:5)

Process finished with exit code 0
```

Fig. 6. Salida de NoSuchMethodFoundException.java

3.4 RuntimeException

RuntimeException

es la superclase de esas excepciones que se pueden generar durante el funcionamiento normal de la máquina virtual Java.

RuntimeException

y sus subclases son *excepciones no verificadas*. Las excepciones no verificadas *no* necesitan ser declaradas en un método o throws

cláusula de constructor .

Más detalles se pueden encontrar aquí .

3.4.1 Excepción aritmética

Lanzado cuando ha ocurrido una condición aritmética excepcional. Por ejemplo, un número entero "dividir por cero" arroja una instancia de esta clase.

Más detalles se pueden encontrar aquí .

El ejemplo se muestra en el fragmento de código a continuación.

ArithemeticExceptionExample.java

```

1 public class ArithemeticExceptionExample {
2     public static void main(String[] args) {
3         try {
4             int a = 12 / 0;
5         } catch (ArithemeticException e) {
6             e.printStackTrace();
7         }
8     }
9

```

La salida se muestra en la Fig. 7. a continuación.

```
/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ
java.lang.ArithemeticException: / by zero
at ArithemeticExceptionExample.main(ArithemeticExceptionExample.java:4)

Process finished with exit code 0
```

Fig. 7. Salida de ArithemeticExceptionExample.java

3.4.2 ArrayStoreException

Lanzado para indicar que se ha intentado almacenar el tipo de objeto incorrecto en una matriz de objetos.



```

07     e.printStackTrace();
08 }
09 }
10 }
11 }

```

La salida de `ArrayStoreException.java` se muestra en la Fig.8 a continuación.

```

/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java
java.lang.ArrayStoreException: java.lang.String
at ArrayStoreException.main(ArrayStoreException.java:5)

Process finished with exit code 0
|  

 Java Code Geeks
JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

```

Fig. 8. Salida de `ArrayStoreException.java`

3.4.3 ClassCastException

Lanzado para indicar que el código ha intentado lanzar un objeto a una subclase de la que no es una instancia.

Más detalles se pueden encontrar aquí .

El ejemplo se muestra en el fragmento de código a continuación.

`ClassCastException.java`

```

01 public class ClassCastException {
02     public static void main(String[] args) {
03         try{
04             Object newObject = new Integer(0);
05             System.out.println((String)newObject);
06         }catch (java.lang.ClassCastException e){
07             e.printStackTrace();
08         }
09     }
10 }

```

La salida de `ClassCastException` se muestra en la Fig.9 a continuación.

```

/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java -jar /Applications/IntelliJ IDEA.app/Contents/Lib/Ideas.jar!/application/lib/JAVA.app/Contents/lib/
java.lang.ClassCastException: class java.lang.Integer cannot be cast to class java.lang.String (java.lang.Integer and java.lang.String are in module java.base of loader 'bootstrap')
at ClassCastException.main(ClassCastException.java:5)
Process finished with exit code 0
|  

 Java Code Geeks
JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

```

Fig. 9 Salida de `ClassCastException.java`

3.4.4 EnumConstantNotPresentException

Se lanza cuando una aplicación intenta acceder a una constante enum por nombre y el tipo enum no contiene ninguna constante con el nombre especificado.

Más detalles se pueden encontrar aquí .

3.4.5 IllegalArgumentException

Lanzado para indicar que un método ha pasado un argumento ilegal o inapropiado.

Más detalles se pueden encontrar aquí .

3.4.5.1 IllegalThreadStateException

Lanzado para indicar que un subproceso no está en un estado apropiado para la operación solicitada. Ver, por ejemplo, los métodos `suspend`

y
resume

en clase
Thread

```

03     try {
04         IllegalThreadStateException d1 = new IllegalThreadStateException();
05         d1.start();
06         d1.start();
07     } catch (java.lang.IllegalThreadStateException e) {
08         e.printStackTrace();
09     }
10 }
11 }
```

La salida del fragmento de código anterior se muestra en la Fig. 10 a continuación.

```
/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ
java.lang.IllegalThreadStateException
at java.base/java.lang.Thread.start(Thread.java:790)
at IllegalThreadStateException.main(IllegalThreadStateException.java:6)

Process finished with exit code 0
```

Fig. 10 Salida de IllegalThreadStateException.java

3.4.5.2 NumberFormatException

Se arroja para indicar que la aplicación ha intentado convertir una cadena a uno de los tipos numéricos, pero que la cadena no tiene el formato apropiado.

Más detalles se pueden encontrar aquí .

El ejemplo se muestra en el fragmento de código a continuación.

NumberFormatException.java

```

01 public class NumberFormatException {
02     public static void main(String args[]) {
03         try {
04             int num = Integer.parseInt("XYZ");
05             System.out.println(num);
06         } catch (java.lang.NumberFormatException e) {
07             e.printStackTrace();
08         }
09     }
10 }
```

La salida de NumberFormatException.java se muestra en la Fig.11 a continuación.

```
/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar
java.lang.NumberFormatException: For input string: "XYZ"
at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:68)
at java.base/java.lang.Integer.parseInt(Integer.java:658)
at java.base/java.lang.Integer.parseInt(Integer.java:778)
at NumberFormatException.main(NumberFormatException.java:4)

Process finished with exit code 0
```

Fig. 11 Salida de NumberFormatException.java

3.4.6 IllegalMonitorStateException

Lanzado para indicar que un subprocesso ha intentado esperar en el monitor de un objeto o para notificar a otros subprocessos que esperan en el monitor de un objeto sin poseer el monitor especificado.

Más detalles se pueden encontrar aquí .

El código de ejemplo se muestra en el fragmento a continuación.

IllegalMonitorStateException.java

```

01 import java.util.concurrent.TimeUnit;
02
03 public class IllegalMonitorStateException {
04     public static void main(String[] args) {
05         try {
06             Utility.syncObject.wait();
07         } catch (InterruptedException ex) {
08             System.err.println("An InterruptedException was caught: " + ex.getMessage());
09             ex.printStackTrace();
10         }
11     }
12 }
13
14 class Utility {
15
16
17     public final static Object syncObject = new Object();
18
19     public static class HaltThread extends Thread {
20
21         @Override
22         public void run() {
23             synchronized (syncObject) {
24                 try {
25                     System.out.println("[HaltThread]: Waiting for another thread "
26                         + "to notify me...");
27                     syncObject.wait();
28                 }
29             }
30         }
31     }
32 }
```

```

41     synchronized (syncObject) {
42         try {
43             System.out.println("[StartingThread]: Sleeping for some time...");
44             TimeUnit.SECONDS.sleep(5);
45             System.out.println("[StartingThread]: Woke up!");
46
47             System.out.println("[StartingThread]: About to notify another thread...");
48             syncObject.notify();
49             System.out.println("[StartingThread]: Successfully notified some other thread!");
50         } catch (InterruptedException ex) {
51             System.err.println("[HaltThread]: An InterruptedException was caught: "
52                             + ex.getMessage());
53             ex.printStackTrace();
54         }
55     }
56 }
57 }
58 }
```

La salida de IllegalMonitorStateException.java se muestra en la Fig. 12 a continuación.

```

/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java "-javaagent
Exception in thread "main" java.lang.IllegalMonitorStateException
at java.base/java.lang.Object.wait(Native Method)
at java.base/java.lang.Object.wait(Object.java:326)
at IllegalMonitorStateException.main(IllegalMonitorStateException.java:6)

Process finished with exit code 1
```



Fig. 12 Salida de IllegalMonitorStateException.java

3.4.7 IllegalStateException

Señala que se ha invocado un método en un momento ilegal o inapropiado. En otras palabras, el entorno Java o la aplicación Java no está en un estado apropiado para la operación solicitada.

Más detalles se pueden encontrar aquí .

El ejemplo se muestra en el fragmento de código a continuación.

IllegalStateException.java

```

01 import java.util.Iterator;
02 import java.util.Vector;
03
04 public class IllegalStateException {
05     public static void main(String[] args) {
06         Vector<Integer> intVect = new Vector<Integer>(3);
07         intVect.add(1);
08         intVect.add(2);
09         intVect.add(3);
10         Iterator vectIter = intVect.iterator();
11         while (vectIter.hasNext()) {
12             Object obj1 = vectIter.next();
13             vectIter.remove();
14             vectIter.remove();
15         }
16 diecisésis
17     }
18 años }
```

El

next()

método de Iterator coloca el **cursor** en el elemento a devolver. Si

remove()

se llama al método, se elimina el elemento donde se posiciona el cursor. Si se llama al método remove () sin llamar al método next (), qué elemento debe eliminar la JVM porque el cursor no apuntará a ningún elemento. En este punto, llamar

remove()

es una **operación ilegal** .

La salida se muestra en la Fig.13 a continuación.

```
Process finished with exit code 1
```



Fig.13 Salida de IllegalStateException.java

3.4.8 IndexOutOfBoundsException

Se arroja para indicar que un índice de algún tipo (como una matriz, una cadena o un vector) está fuera de rango.

Más detalles se pueden encontrar aquí .

3.4.8.1 ArrayIndexOutOfBoundsException

Lanzado para indicar que se ha accedido a una matriz con un índice ilegal. El índice es negativo o mayor o igual que el tamaño de la matriz.

para más detalles ver aquí .

El ejemplo se muestra en el fragmento de código a continuación.

ArrayIndexOutOfBoundsException.java

```
01  public class ArrayIndexOutOfBoundsException {
02      public static void main(String[] args) {
03          int[] arr = new int[3];
04          try{
05              arr[10] = 12;
06          }catch (ArrayIndexOutOfBoundsException e){
07              e.printStackTrace();
08          }
09      }
10 }
```

La salida se muestra en la Fig. 14 a continuación.

```
/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ
java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 3
at ArrayIndexOutOfBoundsException.main(ArrayIndexOutOfBoundsException.java:5)

Process finished with exit code 0
```



Fig.14 Salida de ArrayIndexOutOfBoundsException.java

3.4.8.2 StringIndexOutOfBoundsException

Lanzado por

String

métodos para indicar que un índice es negativo o mayor que el tamaño de la cadena. Para algunos métodos, como el método charAt, esta excepción también se produce cuando el índice es igual al tamaño de la cadena.

Para más detalles ver aquí .

El ejemplo se muestra en el fragmento de código a continuación.

StringIndexOutOfBoundsException.java

```
01  public class StringIndexOutOfBoundsException {
02      public static void main(String[] args) {
03          String sampleStr = "JavaCodeGeeks";
04          try{
05              System.out.println(sampleStr.charAt(100));
06          }catch (java.lang.StringIndexOutOfBoundsException e){
07              e.printStackTrace();
08          }
09      }
10 }
```

La salida se muestra en la Fig.15 a continuación.

```
/Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java "-javaagent:/Applications/IntelliJ
java.lang.StringIndexOutOfBoundsException: String index out of range: 100
at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:48)
at java.base/java.lang.String.charAt(String.java:709)
at StringIndexOutOfBoundsException.main(StringIndexOutOfBoundsException.java:5)

Process finished with exit code 0
```



Fig.15 Salida de StringIndexOutOfBoundsException.java

```

1  public class NegativeArraySizeException {
2      public static void main(String[] args) {
3          try{
4              int[] sampleArr = new int[-1];
5          }catch (java.lang.NegativeArraySizeException e){
6              e.printStackTrace();
7          }
8      }
9  }

```

La salida de NegativeArraySizeException.java se muestra en la Fig. 16 a continuación.

Fig 16. Salida de NegativeArraySizeException.java

3.4.10 NullPointerException

Se lanza cuando una aplicación intenta usar

null

en un caso donde se requiere un objeto. Éstos incluyen:

- Llamando al método de instancia de un

null

objeto.

- Acceder o modificar el campo de un

null

objeto.

- Tomando la longitud de

null

como si fuera una matriz.

- Acceder o modificar las ranuras de

null

como si fuera una matriz.

- Lanzando

null

como si fuera un

Throwable

valor.

Las aplicaciones deben lanzar instancias de esta clase para indicar otros usos ilegales del

null

objeto.

Para más detalles ver aquí .

El ejemplo se muestra en el fragmento de código a continuación.

NullPointerException.java

```

01  public class NullPointerException {
02      public static void main(String[] args) {
03          try{
04              String abc=null;
05              System.out.println(abc.toLowerCase());
06          }catch(java.lang.NullPointerException e){
07              e.printStackTrace();
08          }
09      }
10  }

```

La salida de NullPointerException.java se muestra en la Fig.17 a continuación.

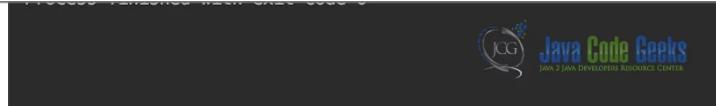


Fig 17. Salida de NullPointerException.java

3.4.11 SecurityException

El

```
SecurityException
```

indica que se ha producido una violación de seguridad y por lo tanto, la solicitud no puede ser ejecutada. Es arrojado por el administrador de seguridad para indicar una violación de seguridad.

Un ejemplo simple es usar un nombre de paquete que ya está definido en Java. Otro caso de uso sería que si JVM determina que el nombre del paquete **no es válido**.

Para más detalles ver aquí .

3.4.12 TypeNotPresentException

Se lanza cuando una aplicación intenta acceder a un tipo usando una cadena que representa el nombre del tipo, pero no se puede encontrar una definición para el tipo con el nombre especificado.

Esta excepción difiere de

```
ClassNotFoundException
```

que ClassNotFoundException es una excepción marcada, mientras que esta excepción no está marcada.

Tenga en cuenta que esta excepción se puede usar cuando se accede a variables de tipo indefinido, así como cuando se cargan tipos (por ejemplo, clases, interfaces o tipos de anotaciones).

Para más detalles visite aquí .

3.4.13 UnsupportedOperationException

Lanzado para indicar que la operación solicitada no es compatible.

Esta clase es miembro de Java Collections Framework.

Para más detalles ver aquí .

El ejemplo se muestra en el fragmento de código a continuación.

[UnsupportedOperationException.java](#)

```
01 import java.util.Arrays;
02 import java.util.List;
03
04 public class UnsupportedOperationException {
05     public static void main(String[] args) {
06         String[] flowersAsArrays = { "Ageratum", "Allium", "Poppy", "Catmint" };
07         List<String> flowersAsArrayList = Arrays.asList(flowersAsArrays);
08         try{
09             flowersAsArrayList.add("Celosia");
10         } catch (java.lang.UnsupportedOperationException e){
11             e.printStackTrace();
12         }
13     }
14 }
```

La salida de UnsupportedOperationException.java se muestra en la Fig.18 a continuación.

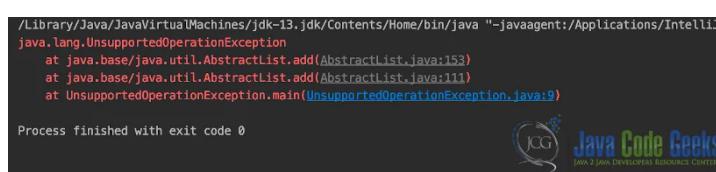


Fig 18. Salida de UnsupportedOperationException.java

4. Resumen

En resumen, hemos discutido todas las excepciones predefinidas en Java, con los ejemplos de código relevantes. Espero que esto te dé una idea sobre qué son las excepciones en Java y cómo usar las excepciones predefinidas en Java.



(No hay valoraciones aún) Comenzó el debate 172 reproducciones Tweet.

¿Quieres saber cómo desarrollar tus habilidades para convertirte en un Rockstar de Java?

Suscríbete a nuestro boletín para comenzar a rockear ahora
Para comenzar, le ofrecemos nuestros eBooks más vendidos **GRATIS!**

1. JPA Mini libro
2. Guía de resolución de problemas de JVM
3. Tutorial JUnit para pruebas unitarias
4. Tutorial de anotaciones de Java
5. Preguntas de la entrevista Java
6. Preguntas de la entrevista de primavera
7. Diseño de la interfaz de usuario de Android

y muchos más

Dirección de correo electrónico:

Your email address

Reciba alertas de trabajo de Java y desarrollador en su área

Regístrate



¿TE GUSTA ESTE ARTÍCULO? LEER MÁS DE JAVA CODE GEEKS



API Development Simplified

Int to char Java Example

Conviértete en Data Scientist **Java Cheat Sheet**

Ad Postman

javacodegeeks.com

Ad IE Business School

javacodegeeks.com



Cursos De Programación Web

Java Collections Tutorial

Java for Beginners, Java Programming Examples

Components involved in a robust Micro Service Arch

Ad Ubiquum Code Academy

javacodegeeks.com

javacodegeeks.com

javacodegeeks.com

Deja una respuesta

 Start the discussion...

 Suscribir ▾

**CURSOS**

- [Minibooks](#)
- [Noticias](#)
- [Recursos](#)
- [Tutoriales](#)

LA RED CODE GEEKS

- [Geeks de código .NET](#)
- [Java Code Geeks](#)
- [System Code Geeks](#)
- [Web Code Geeks](#)

Android Alert Dialog Example

- [Android OnClickListener Example](#)
- [How to convert Character to String and a String to Character Array in Java](#)
- [Java Inheritance example](#)
- [Java write to File Example](#)
- [java.io.FileNotFoundException – How to solve File Not Found Exception](#)
- [java.lang.ArrayIndexOutOfBoundsException – How to handle Array Index Out Of Bounds Exception](#)
- [java.lang.NoClassDefFoundError – How to solve No Class Def Found Error](#)
- [JSON Example With Jersey + Jackson](#)
- [Spring JdbcTemplate Example](#)

ultimate Java to Java developers resource center; targeted at the technical team lead (senior developer), project manager and junior dev JCGs serve the Java, SOA, Agile and Telecom communities with daily domain experts, articles, tutorials, reviews, announcements, code snippets and source projects.

DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples Java Code Geeks is not connected to Oracle Corporation and is not sponsored by Oracle Corporation.



