Excepciones de **Java**

ArithmeticException: Se lanza cuando ocurre un error aritmético, como una división por cero.

Ejemplo

public class ArithmeticExample {

public static int divide(int dividend, int divisor) {

try {

// Aquí es donde podría ocurrir un ArithmeticException

return dividend / divisor;

} catch (ArithmeticException e) {

System.err.println("Error: No se puede dividir por cero.");

return 0;

}

}

public static void main(String[] args) {

System.out.println("10 / 2 = " + divide(10, 2));

System.out.println("10 / 0 = " + divide(10, 0));

}

}

NullPointerException: Ocurre cuando intentas acceder a un objeto que es nulo.

Ejemplo

public class NullPointerExample {

public static void printName(String name) {

try {

System.out.println("La longitud del nombre es: " + name.length());

} catch (NullPointerException e) {

System.err.println("Error: El nombre es null, no se puede calcular la longitud.");

}

}

public static void main(String[] args) {

System.out.println ("Juan");

System.out.println (null);

}

}

ArrayIndexOutOfBoundsException: Se lanza cuando intentas acceder a un índice fuera del rango de un arreglo.

Ejemplo

public class ArrayIndexExample {

public static void accessElement(int[] array, int index) {

try {

System.out.println("Elemento en el índice " + index + ": " + array[index]);

} catch (ArrayIndexOutOfBoundsException e) {

System.err.println("Error: Índice " + index + " fuera del rango. El tamaño del arreglo es " + array.length + ".");

}

}

public static void main(String[] args) {

int[] numeros = {1, 2, 3, 4, 5}; // Un arreglo de 5 elementos

accessElement(numeros, 2);

accessElement(numeros, 10);

}

}

ClassCastException: Ocurre cuando intentas convertir un objeto a un tipo incompatible.

Ejemplo

public class Animal {

public void sound() {

System.out.println("Animal sound");

}

}

public class Dog extends Animal {

@Override

public void sound() {

System.out.println("Bark");

}

}

public class ClassCastExample {

public static void makeAnimalSound(Animal animal) {

try {

Dog dog = (Dog) animal; // Intentar hacer un cast

dog.sound();

} catch (ClassCastException e) {

System.out.println("Error: No se puede convertir a Dog.");

}

}

public static void main(String[] args) {

Animal genericAnimal = new Animal();

Animal dogAnimal = new Dog();

makeAnimalSound(genericAnimal);

makeAnimalSound(dogAnimal);

}

}

IllegalArgumentException: Se lanza cuando se pasa un argumento inválido a un método.

Ejemplo

public class AgeValidator {

public static void validateAge(int age) {

if (age < 0) {

throw new IllegalArgumentException("La edad no puede ser negativa");

}

System.out.println("Edad válida: " + age);

}

public static void main(String[] args) {

try {

validateAge(25); // Edad válida

validateAge(-5); // Lanza IllegalArgumentException

} catch (IllegalArgumentException e) {

System.err.println("Error: " + e.getMessage());

}

}

}

StringIndexOutOfBoundsException: Ocurre cuando se intenta acceder a un índice fuera del rango de una cadena.

Ejemplo

public class StringIndexExample {

public static void printCharAt(String text, int index) {

try {

char character = text.charAt(index);

System.out.println("Carácter en el índice " + index + ": " + character);

} catch (StringIndexOutOfBoundsException e) {

System.err.println("Error: Índice " + index + " está fuera del rango para el texto.");

}

}

public static void main(String[] args) {

String exampleText = "Java Programming";

printCharAt(exampleText, 5);

printCharAt(exampleText, 20);

}

}

NumberFormatException: Se lanza cuando se intenta convertir un texto a un número y el formato es incorrecto.

Ejemplo

public class NumberFormatExample {

public static void convertToInt(String str) {

try {

int number = Integer.parseInt(str);

System.out.println("Número convertido: " + number);

} catch (NumberFormatException e) {

System.err.println("Error: No se puede convertir '" + str + "' a un número.");

}

}

public static void main(String[] args) {

convertToInt("123");

convertToInt("abc");

}

}

ClassNotFoundException: Ocurre cuando se intenta acceder a una clase que no se puede encontrar.

Ejemplo

public class ClassLoaderExample {

public static void loadClass(String className) {

try {

Class<?> clazz = Class.forName(className); // Intentar cargar una clase

System.out.println("Clase cargada: " + clazz.getName());

} catch (ClassNotFoundException e) {

System.err.println("Error: Clase '" + className + "' no encontrada.");

}

}

public static void main(String[] args) {

loadClass("java.util.ArrayList");

loadClass("java.util.NonExistentClass");

}

}

NegativeArraySizeException: Se lanza cuando intentas crear un arreglo con tamaño negativo.

Ejemplo

public class NegativeArrayExample {

public static void createArray(int size) {

try {

int[] array = new int[size]; // Intentar crear un arreglo con tamaño negativo

System.out.println("Arreglo creado con tamaño: " + array.length);

} catch (NegativeArraySizeException e) {

System.err.println("Error: No se puede crear un arreglo con tamaño negativo.");

}

}

public static void main(String[] args) {

createArray(5);

createArray(-10);

}

}

NoClassDefFoundError: Ocurre cuando una clase no puede ser encontrada en tiempo de ejecución.

Ejemplo

public class NoClassDefFoundErrorExample {

public static void triggerError() {

throw new NoClassDefFoundError("Simulando NoClassDefFoundError");

}

public static void main(String[] args) {

try {

triggerError(); // Lanza NoClassDefFoundError

} catch (NoClassDefFoundError e) {

System.err.println("Error: " + e.getMessage());

}

}

}

NoSuchMethodException: Ocurre cuando intentas acceder a un método que no existe.

Ejemplo

import java.lang.reflect.Method;

public class Main {

public static void main(String[] args) {

try {

Class<?> clazz = String.class;

Method method = clazz.getMethod("nonExistentMethod");

} catch (NoSuchMethodException e) {

System.out.println("Method not found: " + e.getMessage());

}

}

}

NoSuchFieldException: Se lanza cuando intentas acceder a un campo que no existe.

Ejemplo

import java.lang.reflect.Field;

public class Main {

public static void main(String[] args) {

try {

Class<?> clazz = String.class;

Field field = clazz.getField("nonExistentField");

} catch (NoSuchFieldException e) {

System.out.println("Field not found: " + e.getMessage());

}

}

}

IllegalStateException: Se lanza cuando el estado de un objeto no permite la operación solicitada.

Ejemplo

import java.util.Stack;

public class Main {

public static void main(String[] args) {

Stack<Integer> stack = new Stack<>();

try {

stack.pop();

} catch (IllegalStateException e) {

System.out.println("Illegal state: " + e.getMessage());

}

}

}

IllegalMonitorStateException: Ocurre cuando se intenta operar con un monitor en un estado ilegal.

Ejemplo

public class Main {

public static void main(String[] args) {

Object lock = new Object();

try {

lock.notify(); // IllegalMonitorStateException

} catch (IllegalMonitorStateException e) {

System.out.println("Illegal monitor state: " + e.getMessage());

}

}

}

InstantiationException: Se lanza cuando intentas instanciar una clase abstracta o interfaz.

Ejemplo

abstract class AbstractClass {

public AbstractClass() {

}

}

public class Main {

public static void main(String[] args) {

try {

AbstractClass instance = AbstractClass.class.newInstance();

} catch (InstantiationException e) {

System.out.println("Instantiation error: " + e.getMessage());

} catch (IllegalAccessException e) {

System.out.println("Illegal access: " + e.getMessage());

}

}

}

ArrayStoreException: Ocurre cuando intentas almacenar un tipo incorrecto en un arreglo.

Ejemplo

public class Main {

public static void main(String[] args) {

try {

Object[] array = new String[2];

array[0] = 10; // Causes ArrayStoreException

} catch (ArrayStoreException e) {

System.out.println("Array store error: " + e.getMessage());

}

}

}

CloneNotSupportedException: Se lanza cuando intentas clonar un objeto que no implementa la interfaz Cloneable.

Ejemplo

class MyClass implements Cloneable {

@Override

protected Object clone() throws CloneNotSupportedException {

throw new CloneNotSupportedException("Clone not supported for this object.");

}

}

public class Main {

public static void main(String[] args) {

MyClass myObject = new MyClass();

try {

MyClass clonedObject = (MyClass) myObject.clone();

} catch (CloneNotSupportedException e) {

System.out.println("Clone not supported: " + e.getMessage());

}

}

}

NoSuchElementException: Se lanza cuando intentas obtener un elemento de una colección que está vacía o no tiene más elementos.

Ejemplo

import java.util.Iterator;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

ArrayList<Integer> list = new ArrayList<>();

Iterator<Integer> iterator = list.iterator();

try {

Integer value = iterator.next();

} catch (java.util.NoSuchElementException e) {

System.out.println("No element found: " + e.getMessage());

}

}

}

ConcurrentModificationException: Ocurre cuando intentas modificar una colección mientras estás iterando sobre ella.

Ejemplo

import java.util.ArrayList;

import java.util.ConcurrentModificationException;

import java.util.Iterator;

public class Main {

public static void main(String[] args) {

ArrayList<String> list = new ArrayList<>();

list.add("Item1");

list.add("Item2");

list.add("Item3");

try {

for (String item : list) {

list.remove(item);

}

} catch (ConcurrentModificationException e) {

System.out.println("Concurrent modification error: " + e.getMessage());

}

}

}

IllegalAccessException: Se lanza cuando intentas acceder a un campo o método sin permisos adecuados.

Ejemplo

import java.lang.reflect.Field;

class PrivateClass {

private String secret = "Top Secret";

}

public class Main {

public static void main(String[] args) {

PrivateClass obj = new PrivateClass();

try {

Field field = PrivateClass.class.getDeclaredField("secret");

field.setAccessible(true);

System.out.println(field.get(obj));

} catch (IllegalAccessException e) {

System.out.println("Illegal access: " + e.getMessage());

} catch (NoSuchFieldException e) {

System.out.println("Field not found: " + e.getMessage());

}

}

}

EnumConstantNotPresentException: Ocurre cuando intentas usar una constante de enumeración que no existe.

Ejemplo

public class Main {

public enum Colors {

RED, BLUE, GREEN

}

public static void main(String[] args) {

try {

Colors color = Colors.valueOf("YELLOW");

} catch (EnumConstantNotPresentException e) {

System.out.println("Enum constant not present: " + e.getMessage());

}

}

}

IllegalFormatConversionException: Se lanza cuando una conversión de formato es ilegal o incorrecta.

Ejemplo

public class Main {

public static void main(String[] args) {

try {

System.out.printf("This is an integer: %d", "NotAnInteger"); // Causes IllegalFormatConversionException

} catch (IllegalFormatConversionException e) {

System.out.println("Illegal format conversion: " + e.getMessage());

}

}

}

OutOfMemoryError: Ocurre cuando el sistema se queda sin memoria.

Ejemplo

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

try {

ArrayList<int[]> list = new ArrayList<>();

while (true) {

list.add(new int[10000000]); // Causes OutOfMemoryError

}

} catch (OutOfMemoryError e) {

System.out.println("Out of memory: " + e.getMessage());

}

}

}

ClassCastException: Ocurre cuando se intenta hacer un cast entre clases incompatibles.

Ejemplo

public class Main {

public static void main(String[] args) {

try {

Object obj = "I am a string";

Integer num = (Integer) obj; // Causes ClassCastException

} catch (ClassCastException e) {

System.out.println("Class cast error: " + e.getMessage());

}

}

}

AbstractMethodError: Se lanza cuando un método abstracto no tiene implementación.

Ejemplo

abstract class AbstractClass {

abstract void abstractMethod();

}

class ConcreteClass extends AbstractClass {

@Override

void abstractMethod() {

System.out.println("Abstract method implemented in ConcreteClass");

} }

public class Main {

public static void main(String[] args) {

try {

AbstractClass instance = new ConcreteClass();

instance.abstractMethod(); // Causes AbstractMethodError at runtime

} catch (AbstractMethodError e) {

System.out.println("Abstract method error: " + e.getMessage());

}

}

}

FileNotFoundException: Esta excepción se lanza cuando se intenta acceder a un archivo que no existe. Se suele usar al trabajar con archivos en operaciones como lectura o escritura, y el archivo especificado no puede ser encontrado.

Ejemplo

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

public class Main {

public static void main(String[] args) {

try {

File file = new File("non\_existent\_file.txt");

FileInputStream fileInputStream = new FileInputStream(file);

} catch (FileNotFoundException e) {

System.out.println("File not found: " + e.getMessage());

}

}

}

IOException: Se trata de una excepción general que indica un problema con operaciones de entrada/salida (E/S), como errores de lectura o escritura en archivos, interrupciones de comunicación o problemas con la red.

Ejemplo

public class Main {

public static void main(String[] args) {

try {

System.setSecurityManager(new SecurityManager());

System.exit(0); // Causes SecurityException

} catch (SecurityException e) {

System.out.println("Security exception: " + e.getMessage());

}

}

}

SecurityException: Esta excepción se lanza cuando se viola una política de seguridad, como cuando se intenta realizar una operación prohibida por el SecurityManager. Por ejemplo, intentar salir del programa cuando un SecurityManager no lo permite.

Ejemplo

public class Main {

public static void main(String[] args) {

try {

System.setSecurityManager(new SecurityManager());

System.exit(0); // Causes SecurityException

} catch (SecurityException e) {

System.out.println("Security exception: " + e.getMessage());

}

}

}

UnsupportedOperationException: Esta excepción se lanza cuando se intenta realizar una operación que no está soportada por la implementación actual. Un ejemplo común es cuando se intenta modificar una lista creada con Arrays.asList, que es una lista inmutable.

Ejemplo

import java.util.Arrays;

import java.util.List;

public class Main {

public static void main(String[] args) {

try {

List<String> list = Arrays.asList("A", "B", "C");

list.add("D"); // Causes UnsupportedOperationException

} catch (UnsupportedOperationException e) {

System.out.println("Unsupported operation: " + e.getMessage());

}

}

}

StackOverflowError: Este error se lanza cuando la memoria del stack (pila) se desborda debido a demasiada recursión o demasiadas llamadas a métodos. Puede ser causado por llamadas recursivas sin una condición de parada clara.

Ejemplo

public class Main {

public static void main(String[] args) {

try {

recursiveMethod(); // Causes StackOverflowError

} catch (StackOverflowError e) {

System.out.println("Stack overflow: " + e.getMessage());

}

}

static void recursiveMethod() {

recursiveMethod(); // Infinite recursion

}

}

TimeoutException: Se lanza cuando una operación excede el tiempo límite establecido. Es común en escenarios donde se esperan resultados en un período determinado, como al usar Future.get() con un tiempo límite.

Ejemplo

import java.util.concurrent.\*;

public class Main {

public static void main(String[] args) {

ExecutorService executor = Executors.newSingleThreadExecutor();

Future<String> future = executor.submit(() -> {

Thread.sleep(5000);

return "Task completed

}

}

IllegalThreadStateException: Se lanza cuando se intenta realizar una operación no apropiada en un hilo debido a su estado actual, como iniciar un hilo que ya ha sido iniciado.

Ejemplo

public class Main {

public static void main(String[] args) {

Thread thread = new Thread(() -> System.out.println("Thread running"));

thread.start();

try {

thread.start(); // Causa IllegalThreadStateException

} catch (IllegalThreadStateException e) {

System.out.println("Illegal thread state: " + e.getMessage());

}

}

}

IllegalAccessException: Se lanza cuando se intenta acceder a un campo o método de una clase y no se tiene el nivel de acceso adecuado. Por ejemplo, intentar acceder a un campo privado sin autorización.

Ejemplo

import java.lang.reflect.Field;

public class Main {

public static void main(String[] args) {

try {

Class<?> clazz = Class.forName("java.lang.String");

Field field = clazz.getDeclaredField("value");

field.get(null); // Causa IllegalAccessException

} catch (IllegalAccessException | NoSuchFieldException | ClassNotFoundException e) {

System.out.println("Error: " + e.getMessage());

}

}

}

InvocationTargetException: Se lanza cuando un método llamado mediante reflexión (por ejemplo, Method.invoke()) lanza una excepción. El InvocationTargetException encapsula la excepción real lanzada por el método invocado.

Ejemplo

import java.lang.reflect.Method;

import java.lang.reflect.InvocationTargetException;

public class Main {

public static void main(String[] args) {

try {

Class<?> clazz = Class.forName("java.lang.String");

Method method = clazz.getMethod("charAt", int.class);

method.invoke("Hello", 10); // Causa StringIndexOutOfBoundsException dentro de InvocationTargetException

} catch (InvocationTargetException | NoSuchMethodException | IllegalAccessException | ClassNotFoundException e) {

System.out.println("Error: " + e.getMessage());

}

}

}

InterruptedException: Se lanza cuando un hilo es interrumpido mientras está esperando, durmiendo o bloqueado. La interrupción se puede usar para solicitar a un hilo que termine su trabajo.

Ejemplo

public class Main {

public static void main(String[] args) {

Thread thread = new Thread(() -> {

try {

Thread.sleep(10000); // Causa InterruptedException si el hilo es interrumpido

} catch (InterruptedException e) {

System.out.println("Thread was interrupted.");

}

});

thread.start();

thread.interrupt(); // Interrumpe el hilo, causando InterruptedException

}

}

AssertionError: Se lanza cuando una aserción (comúnmente usada para pruebas y validaciones) falla. Esto puede ocurrir si se utiliza la instrucción assert y la condición evaluada es falsa.

Ejemplo

public class Main {

public static void main(String[] args) {

try {

assert false : "Assertion failed"; // Causa AssertionError

} catch (AssertionError e) {

System.out.println("Assertion error: " + e.getMessage());

}

}

}

EOFException: Se lanza cuando se alcanza inesperadamente el final de un flujo de datos (como un archivo o una entrada de red). Suele ocurrir cuando se espera más datos de un flujo pero este termina antes de tiempo.

Ejemplo

import java.io.\*;

public class Main {

public static void main(String[] args) {

try {

DataInputStream dis = new DataInputStream(new ByteArrayInputStream(new byte[0]));

dis.readInt(); // Causa EOFException porque no hay suficientes datos

} catch (EOFException e) {

System.out.println("End of file exception: " + e.getMessage());

} catch (IOException e) {

System.out.println("IO Exception: " + e.getMessage());

}

}

}

UnsupportedEncodingException: Se lanza cuando se intenta utilizar un nombre de codificación no válido al crear un String a partir de un array de bytes.

Ejemplo

import java.io.UnsupportedEncodingException;

public class Main {

public static void main(String[] args) {

try {

byte[] byteArray = { 65, 66, 67, 68 };

String str = new String(byteArray, "invalid\_encoding"); // Puede lanzar UnsupportedEncodingException

} catch (UnsupportedEncodingException e) {

System.out.println("Codificación no soportada: " + e.getMessage());

}

}

}

InterruptedException: Se lanza cuando un hilo está esperando, durmiendo o bloqueado, y otro hilo lo interrumpe.

Ejemplo

public class Main {

public static void main(String[] args) {

Thread t = new Thread(() -> {

try {

Thread.sleep(1000); // Puede lanzar InterruptedException

} catch (InterruptedException e) {

System.out.println("Hilo interrumpido.");

}

});

t.start();

t.interrupt(); // Interrumpe el hilo t

}

}

VerifyError: Este error se lanza cuando la JVM verifica un archivo de clase y encuentra un problema con su estructura o consistencia. Puede ocurrir debido a problemas de compilación o alteraciones en tiempo de ejecución.

Ejemplo

public class Main {

public static void main(String[] args) {

try {

// Simulando un error de verificación

throw new VerifyError("Simulated VerifyError");

} catch (VerifyError e) {

System.out.println("Verification error: " + e.getMessage());

}

}

}

NoSuchFieldError: Este error se lanza cuando se intenta acceder a un campo de clase que no existe. Puede ocurrir si el código intenta usar una clase con cambios recientes o si no se recompila adecuadamente.

Ejemplo

class ClassWithField {

public int existingField = 10;

}

public class Main {

public static void main(String[] args) {

try {

ClassWithField obj = new ClassWithField();

int nonExistentField = obj.nonExistentField; // Causa NoSuchFieldError

} catch (NoSuchFieldError e) {

System.out.println("No such field error: " + e.getMessage());

}

}

}

IllegalAccessError: Este error se lanza cuando un programa intenta acceder a un campo o método que no tiene permiso para usar. A menudo ocurre debido a cambios de acceso en el tiempo de ejecución o errores de compilación.

Ejemplo

class SuperClass {

private void privateMethod() {

System.out.println("Private method in SuperClass");

}

}

class SubClass extends SuperClass {

void accessPrivateMethod() {

try {

this.privateMethod(); // Causa IllegalAccessError

} catch (Error e) {

System.out.println("Illegal access error: " + e.getMessage());

}

}

}

public class Main {

public static void main(String[] args) {

SubClass subClass = new SubClass();

subClass.accessPrivateMethod();

}

}

IllegalCharsetNameException: Se lanza cuando se intenta crear un nombre de conjunto de caracteres que no es válido según la especificación de conjuntos de caracteres.

Ejemplo

import java.nio.charset.Charset;

import java.nio.charset.IllegalCharsetNameException;

public class Main {

public static void main(String[] args) {

try {

Charset charset = Charset.forName("invalid\_charset\_name"); // Causa IllegalCharsetNameException

} catch (IllegalCharsetNameException e) {

System.out.println("Illegal charset name: " + e.getMessage());

}

}

}

MalformedURLException: Se lanza cuando se crea un URL con un formato incorrecto o inválido. Puede ser causado por una sintaxis incorrecta o componentes faltantes.

Ejemplo

import java.net.MalformedURLException;

import java.net.URL;

public class Main {

public static void main(String[] args) {

try {

URL url = new URL("invalid\_url"); // Causa MalformedURLException

} catch (MalformedURLException e) {

System.out.println("Malformed URL: " + e.getMessage());

}

}

}

BrokenBarrierException: Se lanza cuando un hilo espera en una barrera (como CyclicBarrier) y la barrera se rompe debido a que uno de los hilos falla o se interrumpe.

Ejemplo

import java.util.concurrent.BrokenBarrierException;

import java.util.concurrent.CyclicBarrier;

public class Main {

public static void main(String[] args) {

CyclicBarrier barrier = new CyclicBarrier(2);

Thread t1 = new Thread(() -> {

try {

barrier.await(); // Esperando en la barrera

} catch (InterruptedException | BrokenBarrierException e) {

System.out.println("Barrier broken: " + e.getMessage());

}

});

t1.start();

t1.interrupt(); // Interrumpe el hilo, lo que rompe la barrera

}

}

BufferUnderflowException: Se lanza cuando se intenta leer datos de un buffer que no tiene suficientes datos para satisfacer la solicitud. A menudo ocurre al trabajar con buffers de java.nio.

Ejemplo

import java.nio.BufferUnderflowException;

import java.nio.ByteBuffer;

public class Main {

public static void main(String[] args) {

ByteBuffer buffer = ByteBuffer.allocate(4);

buffer.put((byte) 1);

try {

buffer.getInt(); // Causa BufferUnderflowException porque el buffer no tiene suficientes datos

} catch (BufferUnderflowException e) {

System.out.println("Buffer underflow: " + e.getMessage());

}

}

}

IllegalCharsetNameException: Se lanza cuando se intenta crear un nombre de conjunto de caracteres que no es válido según la especificación de conjuntos de caracteres.

Ejemplo

import java.nio.charset.Charset;

import java.nio.charset.IllegalCharsetNameException;

public class Main {

public static void main(String[] args) {

try {

Charset charset = Charset.forName("invalid\_charset\_name"); // Causa IllegalCharsetNameException

} catch (IllegalCharsetNameException e) {

System.out.println("Illegal charset name: " + e.getMessage());

}

}

}

MalformedURLException: Se lanza cuando se crea un URL con un formato incorrecto o inválido. Puede ser causado por una sintaxis incorrecta o componentes faltantes.

Ejemplo

import java.net.MalformedURLException;

import java.net.URL;

public class Main {

public static void main(String[] args) {

try {

URL url = new URL("invalid\_url"); // Causa MalformedURLException

} catch (MalformedURLException e) {

System.out.println("Malformed URL: " + e.getMessage());

}

}

}

BrokenBarrierException: Se lanza cuando un hilo espera en una barrera (como CyclicBarrier) y la barrera se rompe debido a que uno de los hilos falla o se interrumpe.

Ejemplo

import java.util.concurrent.BrokenBarrierException;

import java.util.concurrent.CyclicBarrier;

public class Main {

public static void main(String[] args) {

CyclicBarrier barrier = new CyclicBarrier(2);

Thread t1 = new Thread(() -> {

try {

barrier.await(); // Esperando en la barrera

} catch (InterruptedException | BrokenBarrierException e) {

System.out.println("Barrier broken: " + e.getMessage());

}

});

t1.start();

t1.interrupt(); // Interrumpe el hilo, lo que rompe la barrera

}

}

CharacterCodingException: Se lanza cuando hay un error en la conversión de caracteres, como al intentar codificar o decodificar datos con un conjunto de caracteres incorrecto o incompatible.

Ejemplo

import java.nio.charset.CharacterCodingException;

import java.nio.charset.Charset;

import java.nio.charset.CharsetEncoder;

import java.nio.ByteBuffer;

public class Main {

public static void main(String[] args) {

try {

CharsetEncoder encoder = Charset.forName("US-ASCII").newEncoder();

encoder.encode(ByteBuffer.wrap(new byte[]{(byte) 0xC3, (byte) 0x28})); // Puede lanzar CharacterCodingException

} catch (CharacterCodingException e) {

System.out.println("Character coding error: " + e.getMessage());

}

}

}

EOFException: Se lanza cuando se alcanza inesperadamente el final de un flujo de entrada antes de completar una operación esperada. Suele ocurrir al leer datos de archivos o flujos.

Ejemplo

import java.io.ByteArrayInputStream;

import java.io.EOFException;

import java.io.ObjectInputStream;

public class Main {

public static void main(String[] args) {

try {

ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(new byte[]{});

ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);

objectInputStream.readObject(); // Puede lanzar EOFException

} catch (EOFException e) {

System.out.println("End of file error: " + e.getMessage());

} catch (ClassNotFoundException | IOException e) {

e.printStackTrace();

}

}

}

ZipException: Se lanza cuando se encuentra un error con archivos ZIP o cuando se trata de operar con datos comprimidos de forma incorrecta.  
Ejemplo

import java.util.zip.ZipException;

import java.util.zip.ZipFile;

import java.io.File;

import java.io.IOException;

public class Main {

public static void main(String[] args) {

try {

ZipFile zipFile = new ZipFile(new File("archivo\_invalido.zip")); // Puede lanzar ZipException

} catch (ZipException e) {

System.out.println("Zip error: " + e.getMessage());

} catch (IOException e) {

e.printStackTrace();

}

}

}

ClassFormatError: Se lanza cuando se encuentra un problema con el formato de un archivo de clase. Suele ocurrir cuando la JVM encuentra un archivo de clase corrupto o mal formado.

Ejemplo

public class Main {

public static void main(String[] args) {

try {

// Simulando un error de formato de clase

throw new ClassFormatError("Simulated class format error");

} catch (ClassFormatError e) {

System.out.println("Class format error: " + e.getMessage());

}

}

}

DataFormatException: Se lanza cuando el formato de datos es incorrecto o inaceptable, generalmente en el contexto de operaciones de compresión o descompresión, como en archivos ZIP o compresión ZLIB.

Ejemplo

import java.util.zip.DataFormatException;

import java.util.zip.Inflater;

public class Main {

public static void main(String[] args) {

byte[] invalidData = {1, 2, 3, 4, 5};

Inflater inflater = new Inflater();

inflater.setInput(invalidData);

try {

byte[] output = new byte[10];

inflater.inflate(output); // Puede lanzar DataFormatException

} catch (DataFormatException e) {

System.out.println("Data format error: " + e.getMessage());

}

}

}

PatternSyntaxException: Se lanza cuando la sintaxis de una expresión regular es inválida. Esto puede ocurrir cuando el patrón contiene errores como caracteres no permitidos, paréntesis desbalanceados, o secuencias de escape incorrectas.

Ejemplo

import java.util.regex.Pattern;

import java.util.regex.PatternSyntaxException;

public class Main {

public static void main(String[] args) {

try {

Pattern pattern = Pattern.compile("(invalid(regex)"); // Causa PatternSyntaxException

} catch (PatternSyntaxException e) {

System.out.println("Pattern syntax error: " + e.getDescription());

}

}

}

IllegalSelectorException: Se lanza cuando se usa un selector (como en java.nio.channels.Selector) de manera incorrecta, como intentar registrar un canal en más de un selector o usar un selector cerrado.

Ejemplo

import java.nio.channels.Selector;

import java.nio.channels.SocketChannel;

import java.nio.channels.IllegalSelectorException;

public class Main {

public static void main(String[] args) {

try {

Selector selector = Selector.open();

SocketChannel channel = SocketChannel.open();

channel.register(selector, -1); // Causa IllegalSelectorException

} catch (IllegalSelectorException | IOException e) {

System.out.println("Illegal selector error: " + e.getMessage());

}

}

}

UnresolvedAddressException: Se lanza cuando se intenta conectar a una dirección de red no resuelta, como un nombre de host incorrecto o una dirección IP inválida.

Ejemplo

import java.net.InetSocketAddress;

import java.nio.channels.SocketChannel;

import java.nio.channels.UnresolvedAddressException;

public class Main {

public static void main(String[] args) {

try {

SocketChannel socketChannel = SocketChannel.open();

socketChannel.connect(new InetSocketAddress("invalid.address", 8080)); // Causa UnresolvedAddressException

} catch (UnresolvedAddressException | IOException e) {

System.out.println("Unresolved address error: " + e.getMessage());

}

}

}

DuplicateFormatFlagsException: Se lanza cuando un formato tiene banderas duplicadas. Es común al usar métodos de formato de cadenas, como String.format, y se especifican banderas inconsistentes.

Ejemplo

import java.util.DuplicateFormatFlagsException;

public class Main {

public static void main(String[] args) {

try {

String.format("%--10d", 42); // Puede lanzar DuplicateFormatFlagsException

} catch (DuplicateFormatFlagsException e) {

System.out.println("Duplicate format flags error: " + e.getMessage());

}

}

}

EmptyStackException: Se lanza cuando se intenta realizar una operación en una pila vacía. Esto suele ocurrir al usar estructuras de datos como java.util.Stack y se intenta hacer un pop() o peek() cuando la pila está vacía.

Ejemplo

import java.util.EmptyStackException;

import java.util.Stack;

public class Main {

public static void main(String[] args) {

Stack<Integer> stack = new Stack<>();

try {

stack.pop(); // Puede lanzar EmptyStackException

} catch (EmptyStackException e) {

System.out.println("Stack is empty: " + e.getMessage());

}

}

}

UndeclaredThrowableException: Se lanza cuando un método proxy lanza una excepción que no está declarada en su interfaz. Generalmente ocurre en el contexto de la reflexión y las proxies dinámicas.

Ejemplo

import java.lang.reflect.InvocationHandler;

import java.lang.reflect.Method;

import java.lang.reflect.Proxy;

import java.lang.reflect.UndeclaredThrowableException;

public class Main {

interface MyInterface {

void myMethod();

}

static class MyInvocationHandler implements InvocationHandler {

public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {

throw new RuntimeException("Simulated exception"); // Causa UndeclaredThrowableException

}

}

public static void main(String[] args) {

MyInterface proxy = (MyInterface) Proxy.newProxyInstance(

MyInterface.class.getClassLoader(),

new Class[]{MyInterface.class},

new MyInvocationHandler()

);

try {

proxy.myMethod();

} catch (UndeclaredThrowableException e) {

System.out.println("Undeclared throwable error: " + e.getMessage());

}

}

}

WriteAbortedException: Se lanza cuando se intenta escribir un objeto serializable y la serialización falla debido a un error en el proceso de escritura. Normalmente, esto ocurre cuando hay un error durante la serialización de un objeto dentro del flujo de datos.

Ejemplo

import java.io.\*;

public class Main {

public static void main(String[] args) {

try {

// Simula un objeto que no puede ser serializado

NotSerializable notSerializable = new NotSerializable();

// Intentar escribir un objeto no serializable

FileOutputStream fos = new FileOutputStream("file.ser");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(notSerializable); // Puede lanzar WriteAbortedException

} catch (WriteAbortedException e) {

System.out.println("Write aborted: " + e.getMessage());

} catch (IOException e) {

e.printStackTrace();

}

}

}

CharacterCodingException: Se lanza cuando hay un error al codificar o decodificar caracteres, como al intentar usar un conjunto de caracteres no válido.

Ejemplo

import java.nio.charset.Charset;

import java.nio.charset.CharsetEncoder;

import java.nio.charset.CharacterCodingException;

import java.nio.ByteBuffer;

public class Main {

public static void main(String[] args) {

try {

CharsetEncoder encoder = Charset.forName("US-ASCII").newEncoder();

encoder.encode(ByteBuffer.wrap(new byte[] { (byte) 0xC3, (byte) 0x28 })); // Puede lanzar CharacterCodingException

} catch (CharacterCodingException e) {

System.out.println("Character coding error: " + e.getMessage());

}

}

}

AclNotFoundException: Se lanza cuando no se puede encontrar una Lista de Control de Acceso (ACL) especificada, generalmente en el contexto de seguridad o permisos de acceso.

Ejemplo

import java.security.acl.Acl;

import java.security.acl.AclNotFoundException;

public class Main {

public static void main(String[] args) {

try {

// Simulación de un error donde no se encuentra una ACL

Acl acl = null; // No se encuentra la ACL

if (acl == null) {

throw new AclNotFoundException("ACL no encontrada");

}

} catch (AclNotFoundException e) {

System.out.println("ACL not found: " + e.getMessage());

}

}

}