Tipo de Excepciones

Throwable

Es la superclase de todos los errores y excepciones en el lenguaje Java. Sólo los objetos que son instancias de esta clase son lanzados por la máquina virtual o pueden ser lanzados por la declaración de lanzamiento. Del mismo modo, sólo esta clase o una de sus subclases puede ser el tipo de argumento en una cláusula catch. Para fines de verificación de excepciones en tiempo de compilación, Throwable y cualquier subclase de Throwable que no sea también una subclase de RuntimeException o Error se consideran excepciones verificadas.

se utilizan dos subclases, Error y Exception, para indicar que han ocurrido situaciones excepcionales. Normalmente, estas instancias se crean recientemente en el contexto de la situación excepcional para incluir información relevante (como datos de seguimiento de la pila).

Un Throwable contiene una instantánea de la pila de ejecución de su hilo en el momento de su creación. También puede contener una cadena de mensaje que brinda más información sobre el error.

Exception

La clase Exception y sus subclases son una forma de Throwable que indica condiciones que una aplicación razonable podría querer detectar.

La clase Exception y cualquier subclase que no sea también subclase de RuntimeException son excepciones marcadas. Las excepciones marcadas deben declararse en la cláusula throws de un método o constructor si pueden ser lanzadas por la ejecución del método o constructor y propagarse fuera del límite del método o constructor.

1. **AbsentInformationException:** Se lanza cuando se solicita información que no está disponible, como en la depuración de código cuando no hay detalles sobre variables locales.
2. **AclNotFoundException:** Indica que no se encontró un control de acceso (ACL) específico.
3. **ActivationException:** Se produce cuando hay un problema durante la activación o desactivación de un objeto remoto.
4. **AgentInitializationException:** Lanzada cuando hay un problema durante la inicialización de un agente Java.
5. **AgentLoadException:** Indica que un agente de Java no pudo cargarse correctamente.
6. **AlreadyBoundException:** Se lanza cuando se intenta volver a vincular un objeto en un registro de nombres y ya está vinculado.
7. **AttachNotSupportedException:** Se produce cuando la operación de adjuntar no es compatible con la máquina virtual Java (JVM).
8. **AWTException:** Representa una excepción relacionada con la Abstract Window Toolkit (AWT) de Java.
9. **BackingStoreException:** Lanzada cuando hay un problema en el almacenamiento de respaldo, utilizado por las preferencias del sistema Java.
10. **BadAttributeValueExpException:** Se lanza cuando hay un problema con el valor de un atributo utilizado en una expresión de evaluación.

11. **BadBinaryOpValueExpException:** Se produce cuando hay un problema con un operador binario en una expresión de evaluación.
12. **BadLocationException:** Se utiliza en el paquete javax.swing.text para indicar un problema con una ubicación de texto, como una posición de cursor no válida.
13. **BadStringOperationException:** Lanzada cuando hay un problema con una operación de cadena, como la concatenación, que no se puede realizar correctamente.
14. **BrokenBarrierException:** Se lanza cuando una barrera de concurrencia está rota debido a una excepción lanzada por una de las tareas que intenta esperar en la barrera.
15. **CardException:** Representa un problema con una tarjeta inteligente, como un error durante la comunicación con la tarjeta.
16. **CertificateException:** Indica un problema con un certificado, como un error de formato o una firma no válida.
17. **ClassNotLoadedException:** Se produce cuando una clase no se puede cargar en la JVM debido a un problema en el proceso de carga de clases.
18. **CloneNotSupportedException:** Lanzada cuando un objeto no es clonable, es decir, no implementa la interfaz Cloneable y se llama al método clone().
19. **DataFormatException:** Se produce cuando hay un problema con el formato de los datos, como en la compresión o descompresión de datos.
20. **DatatypeConfigurationException:** Indica un problema con la configuración de un tipo de datos, como un error al crear un objeto XMLGregorianCalendar.
21. **FontFormatException:** Indica un problema con el formato de una fuente, como un archivo de fuente que no cumple con las especificaciones.
22. **DestroyFailedException:** Lanzada cuando no se puede destruir un objeto correctamente, como un objeto bloqueado.
23. **ExecutionControl.ExecutionControlException:** Representa una excepción en el control de ejecución de Java, por ejemplo, en la gestión de políticas de seguridad.
24. **ExecutionException:** Lanzada cuando una tarea ejecutada por un Executor encuentra una excepción durante su ejecución.
25. **ExpandVetoException:** Se lanza cuando se veta una operación de expansión, como en un componente gráfico.
26. **GeneralSecurityException:** Clase base para todas las excepciones de seguridad en Java.
27. **GSSException:** Se lanza cuando ocurre un error en el mecanismo de seguridad GSS-API (Generic Security Services Application Program Interface).
28. **IllegalClassFormatException:** Lanzada cuando hay un problema con el formato de una clase, como durante la instrumentación de código.
29. **IllegalConnectorArgumentsException:** Se produce cuando hay argumentos ilegales proporcionados al conector en JMX (Java Management Extensions).
30. **IncompatibleThreadStateException:** Indica un problema con el estado de un hilo que es incompatible con la operación que se está intentando realizar.
31. **InterruptedException:** Lanzada cuando un hilo se encuentra en estado de espera (o duerme) y otro hilo lo interrumpe.
32. **IntrospectionException:** Se produce cuando hay un problema durante la introspección de una clase, como al buscar propiedades o métodos.
33. **InvalidApplicationException:** Indica un problema con una aplicación, como cuando una aplicación no es válida o no puede ejecutarse.
34. **InvalidMidiDataException:** Lanzada cuando hay datos MIDI no válidos, como al intentar cargar un archivo MIDI dañado.

35. **InvalidPreferencesFormatException:** Se lanza cuando hay un problema con el formato de las preferencias del sistema Java.
36. **InvalidTargetObjectTypeException:** Indica que el tipo de objeto de destino es inválido en el contexto de un método específico.
37. **InvalidTypeException:** Lanzada cuando se proporciona un tipo de objeto no válido, como en operaciones de reflexión.
38. **InvocationException:** Se produce cuando hay un problema durante la invocación de un método, como en la reflexión o la invocación remota.
39. **IOException:** Clase base para todas las excepciones relacionadas con operaciones de entrada/salida (IO).
40. **JMException:** Clase base para todas las excepciones relacionadas con la administración y monitorización de Java (JMX).
41. **JShellException:** Indica un problema relacionado con JShell, la herramienta de línea de comandos de Java.
42. **KeySelectorException:** Se lanza cuando hay un problema con la selección de claves en la API de seguridad de Java.
43. **LambdaConversionException:** Lanzada cuando hay un problema con la conversión de una expresión lambda, como un error de tipo.
44. **LastOwnerException:** Indica que no se puede eliminar el último propietario de un bloqueo en un objeto.
45. **LineUnavailableException:** Lanzada cuando una línea de audio no está disponible, como al intentar abrir una línea de grabación o reproducción.
46. **MarshalException:** Se produce cuando hay un problema con la serialización o deserialización de objetos, como en la comunicación remota.
47. **MidiUnavailableException:** Indica que no se pudo acceder a los dispositivos MIDI, como sintetizadores o secuenciadores.
48. **MimeTypeParseException:** Lanzada cuando hay un problema con el formato de un tipo MIME, como al analizar una cadena MIME inválida.
49. **NamingException:** Clase base para todas las excepciones relacionadas con el servicio de nombres JNDI (Java Naming and Directory Interface).
50. **NoninvertibleTransformException:** Se produce cuando una transformación no es invertible, como al intentar invertir una matriz singular.
51. **NotBoundException:** Lanzada cuando se intenta acceder a un objeto en un registro de nombres que no está vinculado.
52. **NotOwnerException:** Indica que el solicitante no es el propietario de un bloqueo en un objeto.
53. **ParseException:** Lanzada cuando hay un problema durante el análisis de una cadena, como en el análisis de fechas o números.
54. **ParserConfigurationException:** Se produce cuando hay un problema con la configuración del analizador XML, como un error en las opciones de configuración.
55. **PrinterException:** Indica un problema con la impresión, como cuando no se puede imprimir un documento.
56. **PrintException:** Se produce cuando hay un error al imprimir un documento.
57. **PrivilegedActionException:** Lanzada cuando una acción privilegiada, definida por la interfaz PrivilegedAction, lanza una excepción.
58. **PropertyVetoException:** Indica un veto sobre una propiedad, como cuando se intenta cambiar una propiedad que está sujeta a veto.
59. **ReflectiveOperationException:** Clase base para todas las excepciones relacionadas con operaciones de reflexión, como NoSuchMethodException o IllegalAccessException.

60. **RefreshFailedException:** Se produce cuando falla la actualización de un objeto, como un objeto de caché.
61. **RuntimeException:** Clase base para todas las excepciones no comprobadas en Java. Las excepciones de tiempo de ejecución no están obligadas a ser manejadas o declaradas.
62. **SAXException:** Clase base para todas las excepciones relacionadas con el procesamiento de XML mediante SAX (Simple API for XML).
63. **ScriptException:** Lanzada cuando hay un problema durante la ejecución de un script, como en JavaScript o Groovy.
64. **ServerNotActiveException**: Indica que el servidor en una conexión de activación no está activo.
65. **SQLException:** Clase base para todas las excepciones relacionadas con operaciones de bases de datos JDBC (Java Database Connectivity).
66. **StringConcatException:** Lanzada cuando hay un problema con la concatenación de cadenas, como un error de tipo.
67. **TimeoutException:** Se produce cuando expira un tiempo de espera, como al esperar una respuesta de un servidor.
68. **TooManyListenersException:** Indica que se agregó un número excesivo de escuchadores a un componente gráfico en AWT (Abstract Window Toolkit).
69. **TransformerException:** Clase base para todas las excepciones relacionadas con transformaciones XML, como XSLT (Extensible Stylesheet Language Transformations).
70. **TransformException:** Lanzada cuando hay un problema durante una transformación, como en la transformación de un objeto.
71. **UnmodifiableClassException:** Indica que una clase no se puede modificar, como en la instrumentación de código.
72. **UnsupportedAudioFileException:** Se lanza cuando se intenta leer un archivo de audio no compatible, como un archivo con un formato no admitido.
73. **UnsupportedCallbackException:** Indica que un mecanismo de seguridad no puede manejar un tipo específico de callback.
74. **UnsupportedFlavorException:** Se produce cuando se solicita un tipo de datos no compatible en el portapapeles de AWT (Abstract Window Toolkit).
75. **UnsupportedLookAndFeelException:** Indica que el aspecto y la sensación (look and feel) solicitados no son compatibles con el sistema.
76. **URIReferenceException:** Lanzada cuando hay un problema con una referencia de URI, como un URI mal formado.
77. **URISyntaxException:** Indica que hay un error en la sintaxis de un URI.
78. **VMStartException:** Se produce cuando no se puede iniciar una máquina virtual Java (JVM).
79. **XAException:** Clase base para todas las excepciones relacionadas con transacciones distribuidas XA.
80. **XMLParseException:** Se produce cuando hay un error durante el análisis de un documento XML, como un documento mal formado.
81. **XMLSignatureException:** Lanzada cuando hay un problema con una firma XML, como una firma no válida.
82. **XMLStreamException:** Indica un problema durante la lectura o escritura de un flujo XML, como un error de formato.
83. **XPathException:** Se produce cuando hay un problema con una expresión XPath, como una expresión no válida.

Error

Un Error es una subclase de Throwable que indica problemas graves que una aplicación razonable no debería intentar detectar. La mayoría de estos errores son condiciones anormales.

No es necesario que un método declare en su cláusula throws ninguna subclase de error que pueda generarse durante la ejecución del método, pero no detectarse, ya que estos errores son condiciones anormales y que nunca deberían ocurrir. Es decir, Error y sus subclases se consideran excepciones no comprobadas a los efectos de la verificación de excepciones en tiempo de compilación.

1.AnnotationFormatError: Esta excepción se lanza cuando se encuentra un formato incorrecto en una anotación.

2. AssertionError: Se genera cuando una afirmación (o aserción) en el código Java falla. Las afirmaciones se utilizan típicamente para realizar verificaciones de seguridad o estado en el código.

3. AWTError: Esta excepción indica un error en la Abstract Window Toolkit (AWT), que es una biblioteca gráfica en Java.

4. CoderMalfunctionError: Se lanza cuando se encuentra un error en el funcionamiento interno del codificador.

5. FactoryConfigurationError: Indica que ha ocurrido un error en la configuración de una fábrica, como por ejemplo la configuración de una fábrica de objetos o de proveedores de servicios.

6. IOError: Esta excepción se utiliza para representar errores de entrada/salida (IO) que no se pueden recuperar o solucionar de manera programática.

7. LinkageError: Se lanza cuando se produce un error en la vinculación o conexión entre componentes de un programa Java, como por ejemplo en la carga de clases.

BootstrapMethodError: Esta excepción indica un error en el proceso de inicialización del método de arranque (bootstrap method) para una clase invocada mediante la invocación dinámica (dynamic invocation) de Java.

ClassCircularityError: Se lanza cuando se detecta una referencia circular entre clases durante la inicialización de una clase.

ClassFormatError: Indica un error en el formato binario de una clase Java, lo que puede ocurrir cuando el archivo .class no cumple con las especificaciones definidas en la JVM.

ExceptionInInitializerError: Se lanza cuando se produce una excepción durante la inicialización estática de una clase. Esto puede ocurrir si un bloque estático de la clase o la inicialización estática de un campo lanza una excepción.

IncompatibleClassChangeError: Esta excepción se produce cuando el cambio en la estructura de una clase es incompatible con la estructura esperada por otra clase que la utiliza.

NoClassDefFoundError: Indica que la JVM no puede encontrar la definición de una clase durante la carga de clases en tiempo de ejecución, a pesar de que la clase estaba disponible durante la compilación.

UnsatisfiedLinkError: Se produce cuando la JVM no puede encontrar una biblioteca nativa requerida por una aplicación Java. Esto puede ocurrir si la biblioteca nativa no está en la ruta de acceso especificada o si hay problemas de compatibilidad entre la biblioteca y la plataforma.

VerifyError: Esta excepción se lanza cuando la JVM detecta un problema durante la verificación de la estructura de una clase, lo que puede ocurrir si la clase viola las reglas de verificación definidas por la JVM.

8.SchemaFactoryConfigurationError: Similar a FactoryConfigurationError, pero específico para errores en la configuración de una fábrica de esquemas XML.

9. ServiceConfigurationError: Indica un error en la configuración de un servicio.

10.ThreadDeath: Esta excepción se lanza cuando un hilo de Java se detiene de forma repentina y no se puede continuar.

11.TransformerFactoryConfigurationError: Similar a FactoryConfigurationError, pero específico para errores en la configuración de una fábrica de transformadores XML.

12.VirtualMachineError: Esta es la clase base para las excepciones relacionadas con errores internos en la máquina virtual Java (JVM).

InternalError: Esta excepción indica un error interno en la JVM que generalmente se origina en el propio sistema Java. No se espera que las aplicaciones manejen este tipo de errores, ya que indican problemas graves dentro del entorno de ejecución de Java.

OutOfMemoryError: Se lanza cuando la JVM no puede asignar más memoria para la creación de nuevos objetos porque se ha alcanzado el límite máximo de memoria disponible. Esto puede ocurrir cuando una aplicación intenta crear demasiados objetos grandes o cuando hay una fuga de memoria.

StackOverflowError: Esta excepción se produce cuando la pila de llamadas de un programa Java se desborda, generalmente debido a una recursión infinita o a una profundidad excesiva de llamadas a métodos.

UnknownError: Esta excepción se utiliza para representar errores internos no identificados o no esperados en la JVM. Similar a InternalError, indica un problema grave que normalmente requiere intervención por parte del desarrollador o del administrador del sistema.

RuntimeExceptions

1. AnnotationTypeMismatchException: Se lanza cuando una anotación espera un tipo de elemento pero recibe un tipo diferente.
2. ArithmeticException: Se produce cuando se intenta realizar una operación aritmética inválida, como la división por cero.
3. ArrayStoreException: Se lanza cuando se intenta almacenar un objeto de tipo incorrecto en un arreglo.
4. BufferOverflowException: Indica que se ha excedido la capacidad de un búfer al intentar escribir más datos de los que puede contener.
5. BufferUnderflowException: Se produce cuando se intenta leer más datos de los que están disponibles en un búfer.
6. CannotRedoException: Indica que no se puede realizar la operación de rehacer en un historial de acciones.
7. CannotUndoException: Se lanza cuando no se puede realizar la operación de deshacer en un historial de acciones.
8. CatalogException: Representa un problema con un catálogo XML, como un error en la estructura o contenido.
9. ClassCastException: Se produce cuando se intenta realizar una conversión de tipo entre clases que no son compatibles.
10. ClassNotPreparedException: Indica que una clase no está preparada para su uso, como en la instrumentación de código.
11. CMMException: Lanzada cuando hay un problema con el módulo de administración de colores (CMM) en Java.
12. CompletionException: Se produce cuando una tarea futura (future) completa con una excepción.
13. ConcurrentModificationException: Lanzada cuando se detecta una modificación concurrente no permitida en una estructura de datos que no es segura para la concurrencia.
14. DateTimeException: Indica un problema con una operación relacionada con la fecha y la hora, como una fecha inválida.
15. DOMException: Se produce cuando hay un problema con el Document Object Model (DOM) en XML, como un error de sintaxis.
16. DuplicateRequestException: Indica que se ha realizado una solicitud duplicada que no se puede procesar.
17. EmptyStackException: Se lanza cuando se intenta realizar una operación en una pila vacía.
18. EnumConstantNotPresentException: Indica que falta una constante de enumeración que se esperaba.
19. EventException: Lanzada cuando hay un problema con un evento, como un error durante el manejo de eventos.
20. FileSystemAlreadyExistsException: Indica que ya existe un sistema de archivos con el mismo nombre.
21. FileSystemNotFoundException: Se lanza cuando no se encuentra un sistema de archivos.
22. FindException: Indica un problema durante una operación de búsqueda, como al buscar un archivo o recurso.

23. IllegalArgumentException: Se produce cuando se pasa un argumento no válido a un método.
24. IllegalCallerException: Indica que el llamador de un método no es válido o no tiene permiso para realizar la llamada.
25. IllegalMonitorStateException: Se lanza cuando se realiza una operación de sincronización en un objeto sin haber adquirido previamente su bloqueo.
26. IllegalPathStateException: Indica que se ha producido un error en el estado de una ruta de acceso.
27. IllegalStateException: Lanzada cuando el estado de un objeto no es válido para realizar la operación solicitada.
28. IllformedLocaleException: Se produce cuando se encuentra un formato de configuración de localización no válido.
29. ImagingOpException: Indica un problema durante una operación de procesamiento de imágenes.
30. InaccessibleObjectException: Lanzada cuando se intenta acceder a un objeto que no es accesible debido a restricciones de seguridad.
31. IncompleteAnnotationException: Indica que falta un elemento requerido en una anotación.
32. InconsistentDebugInfoException: Se lanza cuando la información de depuración de un archivo de clase es inconsistente.
33. IndexOutOfBoundsException: Lanzada cuando se intenta acceder a un índice fuera del rango válido en una estructura de datos, como un arreglo o una lista.
34. InternalException: Indica un problema interno en la aplicación.
35. InvalidCodeIndexException: Se produce cuando se encuentra un índice de código no válido.
36. InvalidLineNumberException: Lanzada cuando se intenta acceder a un número de línea no válido en un archivo de código fuente.
37. InvalidModuleDescriptorException: Indica un problema con el descriptor de un módulo Java.
38. InvalidModuleException: Lanzada cuando se intenta cargar un módulo Java no válido.
39. InvalidRequestStateException: Se produce cuando una solicitud no está en un estado válido para ser procesada.
40. InvalidStackFrameException: Indica un problema con un marco de pila no válido, como al intentar recuperar información de depuración.
41. JarSignerException: Lanzada cuando hay un problema durante la firma de un archivo JAR.
42. JMRuntimeException: Se lanza cuando ocurre un error en la administración y monitorización de Java (JMX).
43. JSException: Indica un problema con la ejecución de JavaScript desde Java.
44. LayerInstantiationException: Lanzada cuando no se puede instanciar una capa en un servicio de red.
45. LSException: Se produce cuando hay un problema con el Lenguaje de Esquema (LS) en XML.
46. MalformedParameterizedTypeException: Indica un problema con un tipo de parámetro mal formado.

47. MalformedParametersException: Se lanza cuando se encuentra un problema con los parámetros de un método, como un nombre de parámetro no válido.
48. MirroredTypesException: Indica un problema con los tipos reflejados en la API de anotaciones.
49. MissingResourceException: Lanzada cuando no se puede encontrar un recurso, como un archivo de propiedades o una cadena localizada.
50. NashornException: Indica un problema con el motor JavaScript Nashorn.
51. NativeMethodException: Se produce cuando hay un error durante la invocación de un método nativo.
52. NegativeArraySizeException: Lanzada cuando se intenta crear un arreglo con un tamaño negativo.
53. NoSuchDynamicMethodException: Indica que no se pudo encontrar un método dinámico.
54. NoSuchElementException: Se lanza cuando no hay más elementos disponibles en una estructura de datos, como en un iterador.
55. NoSuchMechanismException: Indica que no se pudo encontrar un mecanismo, como en la configuración de seguridad.
56. NullPointerException: Se produce cuando se intenta acceder a un objeto que es null.
57. ObjectCollectedException: Lanzada cuando se intenta acceder a un objeto que ha sido recolectado por el recolector de basura.
58. ProfileDataException: Indica un problema con los datos de perfil en la JVM.
59. ProviderException: Lanzada cuando hay un problema con un proveedor de servicios.
60. ProviderNotFoundException: Se produce cuando no se encuentra un proveedor de servicios.
61. RangeException: Indica que se ha producido un problema con un rango, como al intentar acceder a un índice fuera del rango válido.
62. RasterFormatException: Lanzada cuando hay un problema con el formato de un raster de imagen.
63. RejectedExecutionException: Se lanza cuando una tarea es rechazada por un ejecutor, generalmente porque está en estado de apagado o sobrecargado.
64. ResolutionException: Indica un problema con la resolución de un recurso, como un archivo o una clase.
65. SecurityException: Lanzada cuando ocurre un problema relacionado con la seguridad, como una violación de política de seguridad.
66. SPIResolutionException: Se produce cuando hay un problema con la resolución de un proveedor de servicio en la infraestructura de proveedores de servicio.
67. TypeNotPresentException: Indica que un tipo de clase no está presente en tiempo de ejecución.
68. UncheckedIOException: Se lanza cuando se produce una excepción de E/S no comprobada, como IOException, en un contexto donde no se puede manejar correctamente.
69. UndeclaredThrowableException: Indica que se produjo una excepción no declarada en un método que no está definido para lanzar esa excepción.
70. UnknownEntityException: Lanzada cuando se encuentra una entidad desconocida, como en el análisis de XML.

71. UnknownTreeException: Indica que se encontró un árbol desconocido, como en el análisis de XML.
72. UnmodifiableModuleException: Se produce cuando se intenta modificar un módulo que no es modificable.
73. UnmodifiableSetException: Indica que un conjunto no se puede modificar, como en un conjunto devuelto por Collections.unmodifiableSet.
74. UnsupportedOperationException: Lanzada cuando se intenta realizar una operación que no está soportada, como en una colección de solo lectura.
75. VMDisconnectedException: Se produce cuando la conexión con la máquina virtual Java (JVM) se ha perdido.
76. VMMismatchException: Indica que hay una incompatibilidad entre la versión de la JVM y la versión de una clase.
77. VMOutOfMemoryException: Lanzada cuando la JVM se queda sin memoria.
78. WrongMethodTypeException: Se lanza cuando se invoca un método con un tipo de método incorrecto.
79. XPathException: Indica un problema con una expresión XPath, como una expresión no válida.