TRABAJO 2, EXCEPCIONES EN JAVA

UNIVERSIDAD DE SANTANDER

XAVI ALESSANDRO SANCHEZ VILLAMIZAR
02230131019

JONATHAN ROLANDO REY CASTILLO
Programacion 2 – CUC13271B

CUCUTA

NORTE DE SANTANDER

2024

Excepciones (Exceptions)

Una excepción es un evento, que ocurre durante la ejecución de un programa, que interrumpe el flujo normal de las instrucciones del programa.

Cuando se produce un error dentro de un método, éste crea un objeto y lo entrega al sistema en tiempo de ejecución. El objeto, llamado objeto de excepción, contiene información sobre el error, incluyendo su tipo y el estado del programa cuando se produjo el error. Crear un objeto de excepción y entregárselo al sistema en tiempo de ejecución se denomina lanzar una excepción.

Después de que un método lanza una excepción, el sistema en tiempo de ejecución intenta encontrar algo para manejarla. El conjunto de posibles "algo" para manejar la excepción es la lista ordenada de métodos que han sido llamados para llegar al método donde se produjo el error. La lista de métodos se conoce como pila de llamadas (véase la figura siguiente).

Los Tres Tipos de Excepciones

- ***Checked Exception:*** El primer tipo de excepcion son las excepciones comprobadas (o tambien checked exception) son errores que pueden ser anticipados por un programa bien escrito. Suelen estar provocados por factores externos, como la entrada de datos por parte del usuario o la interacción con sistemas externos. Por ejemplo, intentar abrir un archivo inexistente con **java.io.FileReader** lanzará un **java.io.FileNotFoundException**. Un buen programa debería capturar esta excepción e informar al usuario del error, permitiéndole proporcionar un nombre de archivo válido.

- ***Error:*** El segundo tipo de excepcion es ‘Error’ Error son problemas inesperados que surgen de factores externos fuera del control de la aplicación. A diferencia de las excepciones comprobadas, suelen ser irrecuperables. Por ejemplo, incluso si un archivo se abre con éxito, un mal funcionamiento del hardware podría causar un **java.io.IOError** durante la lectura. El programa puede decidir informar al usuario o simplemente registrar el error y salir.

- ***Runtime Exception:*** El tercer tipo de excepcion son las excepciones de tiempo de ejecución (o runtime exception) son condiciones internas imprevistas que la aplicación generalmente no puede manejar, como errores de lógica o uso incorrecto de una API. Por ejemplo, si se pasa un valor null al constructor de FileReader, este lanzará NullPointerException. Aunque se pueden capturar, es preferible corregir el error. Estas excepciones no están sujetas al Requisito de Captura o Especificación y se conocen como excepciones no verificadas, junto con los errores.

Throwable

La clase Throwable es la superclase de todos los errores y excepciones del lenguaje Java. Sólo los objetos que son instancias de esta clase (o una de sus subclases) son lanzados por la máquina virtual de Java o pueden ser lanzados por la sentencia throw de Java.

Excepcion

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

- **AbsentInformationException:** Una excepción que indica la ausencia de información necesaria en un contexto específico, como la falta de datos en un objeto o recurso.
- **AclNotFoundException:** Esta excepción se produce cuando se intenta acceder a un Control de Acceso de Lista (ACL) que no se puede encontrar, generalmente debido a que el recurso protegido por la ACL no existe o no se puede acceder.
- **ActivationException:** Una excepción que se lanza cuando ocurre un error durante el proceso de activación de un objeto o recurso, como la inicialización incorrecta de una instancia.
- **AgentInitializationException:** Esta excepción se produce cuando hay un problema durante la inicialización de un agente, generalmente asociado con la configuración incorrecta o la incapacidad de cargar recursos necesarios para el agente.
- **AgentLoadException:** Una excepción que indica un problema al cargar un agente, típicamente debido a la imposibilidad de encontrar o cargar recursos necesarios para el agente.
- **AlreadyBoundException:** Esta excepción se lanza cuando se intenta enlazar un nombre a un objeto en un registro (como en el RMI) que ya está enlazado a otro objeto.
- **AttachNotSupportedException:** Una excepción que se produce cuando un intento de adjuntar un agente a una máquina virtual Java no es compatible, lo que puede suceder en entornos restringidos o con configuraciones específicas de seguridad.
- **AWTException**: Esta excepción se produce cuando se encuentra un problema específico en el Toolkit Abstracto de Windows (AWT), como la incapacidad de acceder a los recursos gráficos subyacentes o al sistema de ventanas
- **BackingStoreException**: Una excepción que indica un problema al acceder o manipular el almacenamiento de respaldo, como en la persistencia de preferencias en Java.
- **BadAttributeValueExpException**: Esta excepción se lanza cuando una expresión de valor de atributo (Attribute Value Expression) no es válida o no puede ser evaluada correctamente.
- **BadBinaryOpValueExpException**: Una excepción que se produce cuando una expresión de valor de operación binaria no es válida o no puede ser evaluada correctamente.

- **BadLocationException**: Se lanza cuando se intenta acceder a una ubicación inválida o fuera de rango en un contexto específico, como en la manipulación de texto en Swing.
- **BadStringOperationException**: Una excepción que indica un problema con una operación de cadena de caracteres que no se puede realizar correctamente o es inválida.
- **BrokenBarrierException**: Esta excepción se lanza cuando uno de los hilos que espera en una barrera cíclica es interrumpido antes de que la barrera se complete.
- **CardException**: Se lanza cuando ocurre un error con una tarjeta inteligente o un lector de tarjetas inteligentes, como la comunicación fallida o el estado no válido de la tarjeta.
- **CertificateException**: Una excepción que se produce cuando ocurre un error relacionado con certificados digitales, como la invalidación de un certificado o problemas de autenticación.
- **ClassNotLoadedException**: Esta excepción se lanza cuando una clase necesaria no puede ser cargada en tiempo de ejecución, generalmente debido a problemas de configuración o de clase faltante en el classpath.
- **CloneNotSupportedException**: Se lanza cuando se intenta clonar un objeto que no implementa la interfaz Cloneable, indicando que la clonación no es posible.
- **DataFormatException**: Una excepción que indica que los datos recibidos no tienen el formato esperado o no pueden ser procesados correctamente.
- **DatatypeConfigurationException**: Esta excepción se produce cuando hay un problema con la configuración del tipo de datos, como errores en el formato de fecha o problemas con los esquemas XML.
- **DestroyFailedException**: Se lanza cuando un intento de destruir un recurso o un objeto falla, como en la administración de recursos en Java.
- **ExecutionControl**.**ExecutionControlException**: Una excepción general que indica un problema con el control de la ejecución, como restricciones de seguridad o errores en la configuración del entorno de ejecución.
- **ExecutionException**: Se lanza cuando una excepción es lanzada durante la ejecución de una tarea en un hilo, normalmente encapsulada dentro de otra excepción, como en el manejo de excepciones en hilos.
- **ExpandVetoException**: Una excepción que se produce cuando se veta la expansión de un nodo en un árbol o estructura de datos, usualmente en contextos donde se necesita confirmación antes de expandir un elemento.
- **FontFormatException**: Se lanza cuando hay un problema con el formato de una fuente, como errores en el archivo de fuente o incompatibilidad con el sistema operativo.
- **GeneralSecurityException**: Una excepción base para problemas de seguridad generalizados en Java, como errores en la autenticación, cifrado, o manejo de claves.
- **GSSException**: Se lanza cuando ocurre un error en el Servicio de Seguridad Genérico (GSS), utilizado para la seguridad en la comunicación de red, como problemas de autenticación o negociación de tokens.

- **IllegalClassFormatException**: Una excepción que se produce cuando el formato de una clase no es válido o no cumple con las especificaciones de la JVM (Java Virtual Machine), típicamente encontrada durante la carga dinámica de clases.
- **IllegalConnectorArgumentsException**: Se lanza cuando se proporcionan argumentos ilegales o inválidos al conectar con un depurador remoto en Java.
- **IncompatibleThreadStateException**: Una excepción que indica un estado incompatible o incoherente de un hilo en un contexto específico, como durante la depuración de aplicaciones multihilo.
- **InterruptedException**: Se lanza cuando un hilo en ejecución es interrumpido por otro hilo, generalmente utilizado para la gestión de la concurrencia y la sincronización de hilos.
- **IntrospectionException**: Esta excepción se produce cuando hay un problema durante el proceso de introspección de un objeto, generalmente asociado con la reflexión en Java.
- **InvalidApplicationException**: Se lanza cuando se intenta ejecutar una aplicación que es inválida o no cumple con los requisitos necesarios para su funcionamiento.
- **InvalidMidiDataException**: Una excepción que indica un problema con los datos MIDI, como errores de formato o datos no válidos para la reproducción de música MIDI.
- **InvalidPreferencesFormatException**: Se lanza cuando hay un problema con el formato de los datos de preferencias en Java, generalmente relacionado con la serialización o deserialización de preferencias de usuario.
- **InvalidTargetObjectTypeException**: Esta excepción se produce cuando se intenta asignar un objeto a un tipo de destino que no es compatible, como en la reflexión o la conversión de tipos en Java.
- **InvalidTypeException**: Se lanza cuando se encuentra un tipo de datos no válido o incompatible en un contexto específico, como en la manipulación de tipos de datos en tiempo de ejecución.
- **InvocationException**: Una excepción que indica un error durante la invocación de un método, típicamente asociada con la reflexión o la invocación dinámica de métodos en Java.
- **IOException**: Se lanza cuando ocurre un error de entrada o salida durante la operación con archivos, streams u otros recursos de entrada/salida.
- **JMException**: Una excepción base para problemas relacionados con la gestión de Java Management Extensions (JMX), utilizada para la administración remota y el monitoreo de aplicaciones Java.
- **JShellException**: Se lanza cuando ocurre un error durante la ejecución o manipulación del entorno de JShell, la herramienta de línea de comandos de Java.
- **KeySelectorException**: Esta excepción se produce cuando hay un problema con la selección de una clave en el contexto de criptografía, típicamente utilizado en Java Cryptography Architecture (JCA).

- **LambdaConversionException**: Se lanza cuando hay un problema durante la conversión o adaptación de una expresión lambda a un tipo de interfaz funcional en Java.
- **LastOwnerException**: Una excepción que se produce cuando se intenta eliminar el último propietario de un recurso o entidad, como en la gestión de bloqueos o recursos compartidos.
- **LineUnavailableException**: Se lanza cuando no se puede acceder a una línea de entrada o salida de audio, típicamente asociada con la reproducción o grabación de audio en Java.
- **MarshalException**: Esta excepción se produce cuando ocurre un error durante la serialización o deserialización de objetos, especialmente en el contexto de Java Architecture for XML Binding (JAXB).
- **MidiUnavailableException**: Se lanza cuando no se puede acceder a un dispositivo MIDI o no está disponible para su uso en la reproducción o manipulación de música MIDI en Java.
- **MimeTypeParseException**: Una excepción que indica un error durante el análisis de un tipo MIME, utilizado en la manipulación de contenido multimedia y la comunicación web.
- **NamingException**: Una excepción base para problemas relacionados con el servicio de nombres en Java, como en la gestión de directorios LDAP o registros de objetos en RMI.
- **NoninvertibleTransformException**: Se lanza cuando se intenta invertir una transformación que no es invertible, como en la manipulación de matrices de transformación en Java2D.
- **NotBoundException**: Esta excepción se produce cuando se intenta acceder a un objeto en un registro (como en el RMI) que no está vinculado actualmente.
- **NotOwnerException**: Se lanza cuando se realiza una operación que requiere ser el propietario de un recurso, pero el usuario actual no es el propietario.
- **ParseException**: Una excepción que se produce cuando ocurre un error durante el análisis de datos o texto, como en la conversión de cadenas de texto a otros tipos de datos.
- **ParserConfigurationException**: Se lanza cuando hay un problema con la configuración del analizador XML, como errores en el análisis o validación de documentos XML.
- **PrinterException**: Una excepción que indica un problema durante la impresión de un documento o archivo, como errores de comunicación con la impresora o problemas de configuración.
- **PrintException**: Se lanza cuando ocurre un error durante la impresión de un documento, similar a PrinterException pero más generalizada.
- **PrivilegedActionException**: Una excepción que encapsula otras excepciones lanzadas durante la ejecución de una acción privilegiada dentro de un contexto de seguridad.

- **PropertyVetoException**: Se lanza cuando se produce un veto a un cambio de propiedad, típicamente utilizado en la manipulación de eventos de cambio de propiedades en JavaBeans.
- **ReflectiveOperationException**: Una excepción base para problemas relacionados con la reflexión en Java, como errores de acceso, invocación o manipulación de elementos en tiempo de ejecución.
- **RefreshFailedException**: Una excepción que indica un fallo al actualizar un recurso o una entidad, como en el contexto de la actualización de datos en tiempo real.
- **RuntimeException**: Una excepción que indica un error durante la ejecución de una aplicación, típicamente asociada con problemas imprevistos o errores en tiempo de ejecución que no fueron manejados adecuadamente.
- **SAXException**: Se lanza cuando ocurre un error durante el procesamiento de eventos SAX (Simple API for XML), como problemas de lectura o manipulación de documentos XML.
- **ScriptException**: Una excepción que se produce cuando hay un error durante la ejecución de un script, como en el contexto de Java Scripting API (javax.script).
- **ServerNotActiveException**: Esta excepción se produce cuando se intenta realizar una operación en un objeto de servidor que no está activo o no está disponible.
- **SQLException**: Se lanza cuando ocurre un error relacionado con una base de datos o una operación SQL, como problemas de conexión, consultas inválidas o errores de sintaxis.
- **StringConcatException**: Una excepción que se produce cuando hay un error durante la concatenación de cadenas de texto, típicamente asociado con la manipulación de cadenas en Java.
- **TimeoutException**: Se lanza cuando expira un tiempo de espera durante la espera de un evento, una operación o una respuesta, como en el contexto de operaciones de red o comunicación.
- **TooManyListenersException**: Esta excepción se produce cuando se intenta agregar más de un listener a un componente que solo admite un único listener, como en el contexto de los eventos en la interfaz de usuario.
- **TransformerException**: Una excepción que indica un error durante la transformación de datos, típicamente asociada con la manipulación de documentos XML mediante XSLT (eXtensible Stylesheet Language Transformations).
- **TransformException**: Se lanza cuando ocurre un error durante una transformación de datos en general, sin especificar el tipo específico de transformación.
- **UnmodifiableClassException**: Esta excepción se produce cuando se intenta modificar una clase que no puede ser modificada, generalmente debido a restricciones de seguridad o inmutabilidad.
- **UnsupportedAudioFileException**: Se lanza cuando se intenta leer un archivo de audio en un formato no compatible o no soportado por la plataforma.
- UnsupportedCallbackException: Una excepción que indica que se ha solicitado un callback no compatible o no soportado por la aplicación o la plataforma.

- **UnsupportedFlavorException**: Esta excepción se produce cuando se intenta transferir datos en un formato no compatible o no soportado por el portapapeles o el sistema de transferencia de datos.
- **UnsupportedLookAndFeelException**: Se lanza cuando se intenta utilizar un aspecto visual (look and feel) que no es compatible o no está disponible en la plataforma.
- **URIReferenceException**: Una excepción que se produce cuando hay un problema con una referencia URI, como errores en la construcción o resolución de URIs.
- **URISyntaxException**: Se lanza cuando se encuentra una URI que no cumple con la sintaxis especificada por RFC 3986.
- **VMStartException**: Esta excepción se produce cuando hay un problema durante el inicio o la inicialización de una máquina virtual Java (JVM).
- **XAException**: Una excepción que indica un error durante una operación de transacción distribuida, típicamente asociada con el uso de la API X/Open XA.
- **XMLParseException**: Se lanza cuando ocurre un error durante el análisis de un documento XML, indicando problemas de sintaxis, estructura o formato incorrecto.
- **XMLSignatureException**: Una excepción que se produce cuando hay un error durante la generación o validación de una firma XML, típicamente asociada con la seguridad y la integridad de los documentos XML.
- **XMLStreamException**: Se lanza cuando ocurre un error durante el procesamiento de eventos de streaming XML, como problemas de lectura, escritura o manipulación de datos XML.
- **XPathException**: Una excepción que se produce cuando hay un error durante la evaluación de una expresión XPath, utilizada para navegar y seleccionar nodos en documentos XML.

RuntimeException

RuntimeException es la superclase de aquellas excepciones que se pueden generar durante el funcionamiento normal de la Máquina Virtual Java.

- **AnnotationTypeMismatchException**: Una excepción que se lanza cuando ocurre un tipo de anotación no compatible o incorrecto, como en la reflexión de anotaciones en Java.
- **ArithmeticException**: Se lanza cuando ocurre un error aritmético durante la ejecución de una operación matemática, como la división por cero o el desbordamiento de un número.
- **ArrayStoreException**: Esta excepción se produce cuando se intenta almacenar un elemento de tipo incorrecto en un array, como en la manipulación de arrays en Java.
- **BufferOverflowException**: Se lanza cuando se intenta escribir más datos en un buffer de lo que puede contener, indicando una operación de escritura que excede la capacidad del buffer.
- **BufferUnderflowException**: Una excepción que se produce cuando se intenta leer más datos de los disponibles en un buffer, típicamente asociada con operaciones de lectura que exceden la cantidad de datos presentes en el buffer.
- **CannotRedoException**: Se lanza cuando no se puede realizar una operación de rehacer (redo) en un contexto específico, como en el manejo de ediciones y deshacer/rehacer en interfaces de usuario.
- **CannotUndoException**: Esta excepción se produce cuando no se puede realizar una operación de deshacer (undo) en un contexto específico, como en el manejo de ediciones y deshacer/rehacer en interfaces de usuario.
- **CatalogException**: Una excepción que indica un problema con el catálogo de recursos o información, como errores en la gestión de metadatos o recursos de la aplicación.
- **ClassCastException**: Se lanza cuando se produce una conversión de tipos inválida o incompatible, como intentar convertir un objeto a un tipo que no es compatible con su clase real.
- **ClassNotPreparedException**: Esta excepción se produce cuando una clase no está preparada o inicializada correctamente para su uso, típicamente asociada con la carga y preparación de clases en tiempo de ejecución.
- **CMMException**: Se lanza cuando ocurre un error relacionado con el perfil de gestión de color (Color Management Module) en Java, como problemas de configuración o acceso.
- **CompletionException**: Una excepción que se utiliza para indicar errores durante la ejecución de una operación asíncrona, especialmente en el contexto de CompletableFuture en Java.
- **ConcurrentModificationException**: Se lanza cuando se detecta una modificación concurrente no permitida en una estructura de datos que no es segura para la concurrencia, como en colecciones no sincronizadas.

- **DateTimeException**: Una excepción que indica un problema relacionado con la manipulación de fechas y horas, como errores en la creación o manipulación de objetos LocalDate, LocalTime o LocalDateTime en Java.
- **DOMException**: Se lanza cuando ocurre un error durante la manipulación de documentos XML utilizando el Modelo de Objeto de Documento (DOM), como problemas de acceso o manipulación de nodos.
- **DuplicateRequestException**: Esta excepción se produce cuando se intenta realizar una solicitud duplicada que no es válida o no está permitida, como en el contexto de servicios web o solicitudes de red.
- **EmptyStackException**: Se lanza cuando se intenta acceder a un elemento de una pila vacía, indicando que no hay elementos disponibles para ser extraídos.
- **EnumConstantNotPresentException**: Una excepción que se produce cuando se intenta acceder a una constante de enumeración que no está presente, generalmente debido a cambios en la definición de la enumeración.
- **EventException**: Se lanza cuando ocurre un error durante la manipulación de eventos en el modelo de eventos de DOM, como problemas de registro o ejecución de manejadores de eventos.
- **FileSystemAlreadyExistsException**: Una excepción que se produce cuando se intenta crear un sistema de archivos que ya existe, como en la creación de directorios o archivos en Java.
- **FileSystemNotFoundException**: Se lanza cuando no se puede encontrar un sistema de archivos específico, indicando que el sistema de archivos requerido no está disponible o no se puede acceder.
- **FindException**: Esta excepción se produce cuando ocurre un error durante una operación de búsqueda o búsqueda de recursos, como en la búsqueda de archivos o directorios en el sistema de archivos.
- **IllegalArgumentException**: Se lanza cuando se pasa un argumento ilegal o no válido a un método, como en la validación de parámetros de entrada en Java.
- **IllegalCallerException**: Una excepción que se produce cuando un método es llamado ilegalmente por un invocador no autorizado o en un contexto incorrecto.
- **IllegalMonitorStateException**: Se lanza cuando ocurre un error relacionado con el uso incorrecto de monitores o bloqueos en hilos, como intentar liberar un bloqueo que no se posee.
- **IllegalPathStateException**: Una excepción que indica un error durante la manipulación de rutas (paths), como en la construcción o manipulación de rutas en Java2D.
- **IllegalStateException**: Se lanza cuando se produce un estado ilegal o incoherente en un contexto específico, indicando que una operación no puede continuar debido a un estado incorrecto.
- **IllformedLocaleException**: Una excepción que se produce cuando se encuentra un formato de localización (locale) no válido o incorrecto, como en la configuración de locales en Java.

- **ImagingOpException**: Se lanza cuando ocurre un error durante una operación de procesamiento de imágenes, como problemas de conversión, escala o manipulación de imágenes en Java.
- **InaccessibleObjectException**: Una excepción que se lanza cuando se intenta acceder a un objeto inaccesible, como en la reflexión o el acceso a campos o métodos privados.
- **IncompleteAnnotationException**: Se lanza cuando una anotación no está completamente definida o falta información requerida, como en la reflexión de anotaciones en Java.
- **InconsistentDebugInfoException**: Una excepción que se produce cuando se encuentra información de depuración inconsistente o incorrecta, típicamente asociada con el depurador de Java.
- **IndexOutOfBoundsException**: Se lanza cuando se accede a un índice fuera del rango válido en una estructura de datos, como en la manipulación de arrays o listas.
- **InternalException**: Una excepción que indica un error interno no especificado o inesperado en un sistema o aplicación, como en el manejo de excepciones genéricas.
- **InvalidCodeIndexException**: Se lanza cuando se encuentra un índice de código (code index) no válido o incorrecto, como en el contexto de la depuración o el análisis de bytecode en Java.
- **InvalidLineNumberException**: Una excepción que se produce cuando se encuentra un número de línea no válido o incorrecto, típicamente asociado con la depuración de código en Java.
- **InvalidModuleDescriptorException**: Se lanza cuando se encuentra un descriptor de módulo (module descriptor) no válido o incorrecto, como en el contexto de la modularidad en Java.
- **InvalidModuleException**: Una excepción que se produce cuando se encuentra un módulo no válido o incorrecto, como en el contexto de la modularidad en Java.
- **InvalidRequestStateException**: Una excepción que se lanza cuando se encuentra un estado no válido o incorrecto en una solicitud o proceso.
- **InvalidStackFrameException**: Se lanza cuando se encuentra un marco de pila (stack frame) no válido o incorrecto, típicamente asociado con problemas en la ejecución o gestión de la pila de llamadas en un programa.
- **JarSignerException**: Una excepción que indica un problema durante el proceso de firma de un archivo JAR, como errores en la generación o verificación de firmas digitales.
- **JMRuntimeException**: Se lanza cuando ocurre un error general relacionado con la gestión de Java Management (JMX), como problemas en la administración remota o el monitoreo de aplicaciones Java.
- **JSException**: Esta excepción se produce cuando ocurre un error durante la ejecución de scripts JavaScript, especialmente en el contexto de integración de JavaScript en aplicaciones Java.

- **LayerInstantiationException**: Se lanza cuando no se puede instanciar una capa (layer) en un contexto específico, como en el manejo de capas en aplicaciones de gráficos o interfaz de usuario.
- **LSException**: Una excepción que se produce durante la manipulación de documentos XML utilizando el Lenguaje de Serialización (LS) de DOM, como problemas de escritura o lectura de documentos XML.
- **MalformedParameterizedTypeException**: Se lanza cuando se encuentra un tipo de parámetro (parameterized type) mal formado o incorrecto, típicamente asociado con la reflexión de tipos en Java.
- **MalformedParametersException**: Una excepción que se produce cuando se encuentra un parámetro mal formado o incorrecto en una firma de método, como en la validación de anotaciones de métodos en Java.
- **MirroredTypesException**: Se lanza cuando se intenta acceder a tipos reflejados (mirrored types) que no están disponibles o no son válidos en un contexto específico, como en la reflexión de tipos en Java.
- **MissingResourceException**: Una excepción que se lanza cuando no se puede encontrar un recurso específico, como un archivo de propiedades o una cadena de texto en un archivo de recursos.
- **NashornException**: Se lanza cuando ocurre un error durante la ejecución de scripts en Nashorn, el motor de JavaScript de la plataforma Java.
- **NativeMethodException**: Una excepción que indica un error durante la invocación de un método nativo, generalmente asociado con problemas en la interfaz entre código nativo y Java.
- **NegativeArraySizeException**: Se lanza cuando se intenta crear un array con un tamaño negativo, indicando un error en la creación o manipulación de arrays en Java.
- **NoSuchDynamicMethodException**: Esta excepción se produce cuando se intenta acceder a un método dinámico que no existe en un contexto específico, como en la invocación de métodos en lenguajes de programación dinámicos.
- **NoSuchElementException**: Se lanza cuando se intenta acceder a un elemento que no existe en una estructura de datos, como en la manipulación de colecciones o iteradores en Java.
- **NoSuchMechanismException**: Una excepción que se lanza cuando no se puede encontrar un mecanismo o algoritmo específico, como en la criptografía o la seguridad en Java.
- **NullPointerException**: Se lanza cuando se intenta acceder a un objeto que es null, indicando un error de referencia nula en Java.
- **ObjectCollectedException**: Una excepción que se lanza cuando se intenta acceder a un objeto que ha sido recolectado por el recolector de basura, indicando un error de referencia inválida.
- **ProfileDataException**: Se lanza cuando ocurre un error durante la manipulación de datos de perfil (profile data), como en la recolección o análisis de datos de rendimiento en Java.

- **ProviderException**: Una excepción que se lanza cuando ocurre un error relacionado con un proveedor de servicios, como en la carga o configuración de proveedores de seguridad en Java.
- **ProviderNotFoundException**: Se lanza cuando no se puede encontrar un proveedor de servicios específico, como en la carga de proveedores de seguridad o criptográficos en Java.
- **RangeException**: Una excepción que se produce cuando se encuentra un valor fuera de rango o no válido en un contexto específico, como en la manipulación de rangos numéricos o de fechas.
- **RasterFormatException**: Se lanza cuando ocurre un error durante la manipulación de formatos de ráster (raster), como en la conversión o manipulación de imágenes en Java.
- **RejectedExecutionException**: Una excepción que se lanza cuando se rechaza una tarea en un ejecutor de hilos, indicando que la ejecución de la tarea no es posible o no está permitida.
- **ResolutionException**: Se lanza cuando ocurre un error durante la resolución de un recurso o una dependencia, como en la resolución de nombres o referencias en Java.
- **SecurityException**: Una excepción que se lanza cuando ocurre un error de seguridad, como violaciones de políticas de seguridad o permisos insuficientes.
- **SPIResolutionException**: Se lanza cuando hay un problema durante la resolución de un proveedor de servicios (Service Provider Interface), como en la carga o configuración de proveedores de servicios en Java.
- **TypeNotPresentException**: Una excepción que se lanza cuando no se puede encontrar un tipo específico en un contexto de tiempo de ejecución, como en la reflexión o la carga dinámica de clases en Java.
- **UncheckedIOException**: Se lanza cuando ocurre un error durante la manipulación de operaciones de entrada/salida no verificadas, indicando problemas durante la lectura o escritura de datos.
- **UndeclaredThrowableException**: Una excepción que se lanza cuando se produce una excepción no declarada durante la invocación de un método reflectido, indicando errores durante la reflexión de métodos.
- **UnknownEntityException**: Se lanza cuando se encuentra una entidad desconocida o no válida en un contexto específico, como en la manipulación de XML o documentos HTML.
- **UnknownTreeException**: Una excepción que se produce cuando se encuentra un tipo de árbol (tree) desconocido o no válido, como en la manipulación de estructuras de datos en forma de árbol.
- **UnmodifiableModuleException**: Se lanza cuando se intenta modificar un módulo que no es modificable, típicamente asociado con operaciones de modificación en la modularidad de Java.
- **UnmodifiableSetException**: Una excepción que se produce cuando se intenta modificar un conjunto (set) que no es modificable, indicando que la operación de modificación no está permitida.

- **UnsupportedOperationException**: Se lanza cuando se intenta realizar una operación no soportada o no permitida en un contexto específico, indicando que la operación no es compatible con la implementación o el estado actual.
- **VMDisconnectedException**: Una excepción que se lanza cuando se pierde la conexión con una máquina virtual (VM) remota, indicando problemas de comunicación o conexión.
- **VMMismatchException**: Se lanza cuando hay un problema con la compatibilidad o la versión de la máquina virtual (VM), como en la interoperabilidad entre diferentes versiones de la VM.
- **VMOutOfMemoryException**: Una excepción que se produce cuando la máquina virtual (VM) se queda sin memoria, indicando que no hay suficiente memoria disponible para la ejecución de la aplicación.
- **WrongMethodTypeException**: Se lanza cuando se llama a un método con un tipo de método incorrecto o inapropiado, como en la invocación de métodos reflectidos en Java.
- **XPathException**: Una excepción que se lanza cuando hay un error durante la evaluación de una expresión XPath, utilizada para la navegación y selección de nodos en documentos XML.

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.

- **AnnotationFormatError**: Se lanza cuando ocurre un error relacionado con el formato de una anotación, típicamente durante la lectura o la escritura de metadatos de anotaciones en tiempo de ejecución.
- **AssertionError**: Una excepción que indica una falla en una aserción, es decir, una condición que se supone que siempre es verdadera en un punto determinado del código.
- **AWTError**: Se lanza cuando ocurre un error grave relacionado con la Abstract Window Toolkit (AWT), utilizado para construir interfaces gráficas de usuario en Java.
- **CoderMalfunctionError**: Una excepción que se lanza cuando hay un mal funcionamiento en un codificador, utilizado para convertir caracteres entre diferentes formatos de codificación.
- **FactoryConfigurationError**: Se lanza cuando hay un error en la configuración o inicialización de una fábrica (factory), como en la creación de instancias de clases mediante fábricas en Java.
- **IOError**: Una excepción que se utiliza para indicar un error relacionado con operaciones de entrada/salida (I/O) en Java, indicando problemas graves de lectura o escritura.
- **LinkageError**: Se lanza cuando ocurre un error durante la vinculación de clases o recursos en tiempo de ejecución, indicando problemas con la resolución de dependencias o enlaces dinámicos.
- **SchemaFactoryConfigurationError**: Una excepción que se lanza cuando hay un error en la configuración de una fábrica de esquemas (schema factory), utilizada para crear instancias de validadores de XML.
- **ServiceConfigurationError**: Se lanza cuando hay un error relacionado con la configuración o la carga de servicios en Java, como problemas con los archivos de configuración de servicios.
- **ThreadDeath**: Una excepción especial que se lanza para indicar que un hilo está siendo "matado" de manera explícita por otro hilo.
- **TransformerFactoryConfigurationError**: Se lanza cuando hay un error en la configuración o inicialización de una fábrica de transformadores (transformer factory), utilizada para crear instancias de transformadores de XML.
- **VirtualMachineError**: Se lanza cuando ocurre un error grave relacionado con la máquina virtual (VM), como problemas en la asignación de memoria o fallos internos de la VM.

Fuentes:

https://programming.guide/java/list-of-java-exceptions.html