Hola, te saluda nuevamente Ubaldo Acosta. Espero que estés listo para comenzar con esta lección..

Vamos a estudiar el tema de excepciones en Java.

¿Estás listo? ¡Vamos!

En esta lección vamos a estudiar el tema de excepciones. Una excepción es una situación no esperada en la ejecución de un programa.

Un ejemplo de excepción es por ejemplo, si un argumento es válido o no, si una conversión de tipo de datos es incompatible, si existe una falla en la conexión a la base de datos, etc.

Existen muchas clases de Excepciones ya creadas en el API de Java para resolver varios de los problemas mencionados, pero si no existe la clase que se adecue a nuestras necesidades nosotros podemos crear nuestras propias clases de excepción, vamos a ver a continuación qué tipos de excepciones existen y cómo utilizar cada una de ellas.

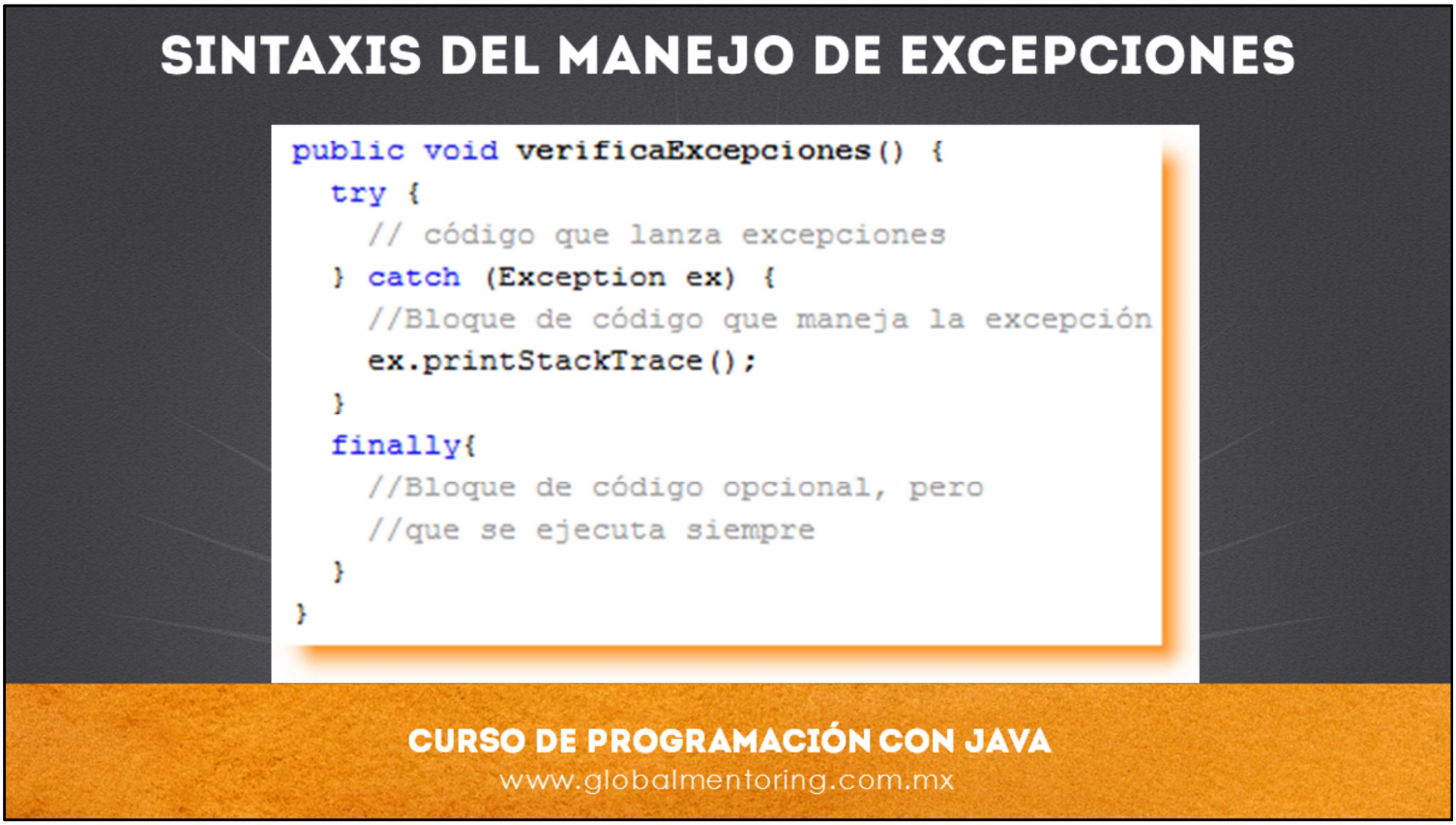
Vamos a estudiar los tipos de excepciones en el API de Java. Todas las clases en Java descienden de la clase Throwable y posteriormente existen una jerarquía de clases, sin embargo en general existen dos tipos de excepciones con las que vamos a trabajar, las que se conocen como checked exceptions y las uncheked exceptions. Veamos en qué consiste cada una:

A) Checked Exceptions o Excepciones que heredan de la clase Exception: Si nuestro programa arroja este tipo de excepción el compilador solicitará realizar alguna actividad con este tipo de excepción. Más adelante veremos como procesar una excepción de este tipo, por ejemplo procesarlas dentro de un bloque try/catch o arrojándolas en la declaración del método. Un ejemplo de este tipo de excepción de tipo checked exception es la clase SQLException, la cual desciende de la clase Exception y es arrojada cuando existen algún problema con temas relacionados con el uso y manejo de bases de datos.

B) Uncheked Exceptions o Excepciones que heredan de la clase RuntimeException: Este tipo de excepciones NO estamos obligados a procesarlas, por lo que es opcional el uso del bloque try/catch o en la declaración del método. Este tipo de excepciones también se les conoce como excepciones en tiempo de ejecución. Ejemplos de este tipo de excepciones son: NullPointerException, ArrayOutOfBoundsException, entre muchas otras.

Existen una polémica respecto a cual de los dos tipos de excepciones utilizar y la estrategia a seguir respecto al manejo de excepciones en nuestros programas. Sin embargo en los últimos años existe una tendencia respecto al uso de las excepciones de tipo uncheked exceptions, las cuales no obliga al programador a procesar este tipo de excepciones y por lo tanto es opcional agregar algún código de manejo de excepciones en nuestros métodos, obteniendo así un código más limpio y que permite escoger al equipo de programadores si procesan las excepciones agregando algún código para ello o simplemente dejan un código más limpio y se encargará de procesar la excepción quien finalmente esté en necesidad de realizar dicho procesamiento.

Ejemplos de equipos de trabajo que han optado por el uso de unchecked exceptions es el equipo de trabajo del framework de Spring, el cual optó no sólo por utilizar este tipo de excepciones, sino que al usar el framework de Spring muchas de las excepciones de tipo checked, las convirtió a tipo uncheked, por lo que es opcional procesar, en la mayoría de los casos, las excepciones que arroja el framework de Spring.



Vamos a revisar a continuación la sintaxis general para el manejo de excepciones, para ello utilizamos el bloque try/catch.

El bloque finally es opcional, pero si se pone, siempre se va a ejecutar, aunque no ocurra la excepción, por ello es que en ocasiones lo utilizaremos para asegurarnos que independientemente del problema que ocurra, se ejecute el código del bloque finally.

Podemos observar que el código que posiblemente arroja la excepción debe estar envuelto por el bloque try. Y si queremos procesar la excepción lo podremos hacer dentro del bloque catch. Este bloque es el que recibe el tipo de excepción que queremos procesar, pudiendo ser varios bloques catch y por cada bloque un tipo de excepción distinto. Cabe mencionar que el orden para procesar más de una excepción será procesar en primer lugar la excepción que sea la clase de menor jerarquía y finalmente la excepción de mayor jerarquía según la jerarquía de clases que describa a las excepciones que vamos a procesar.

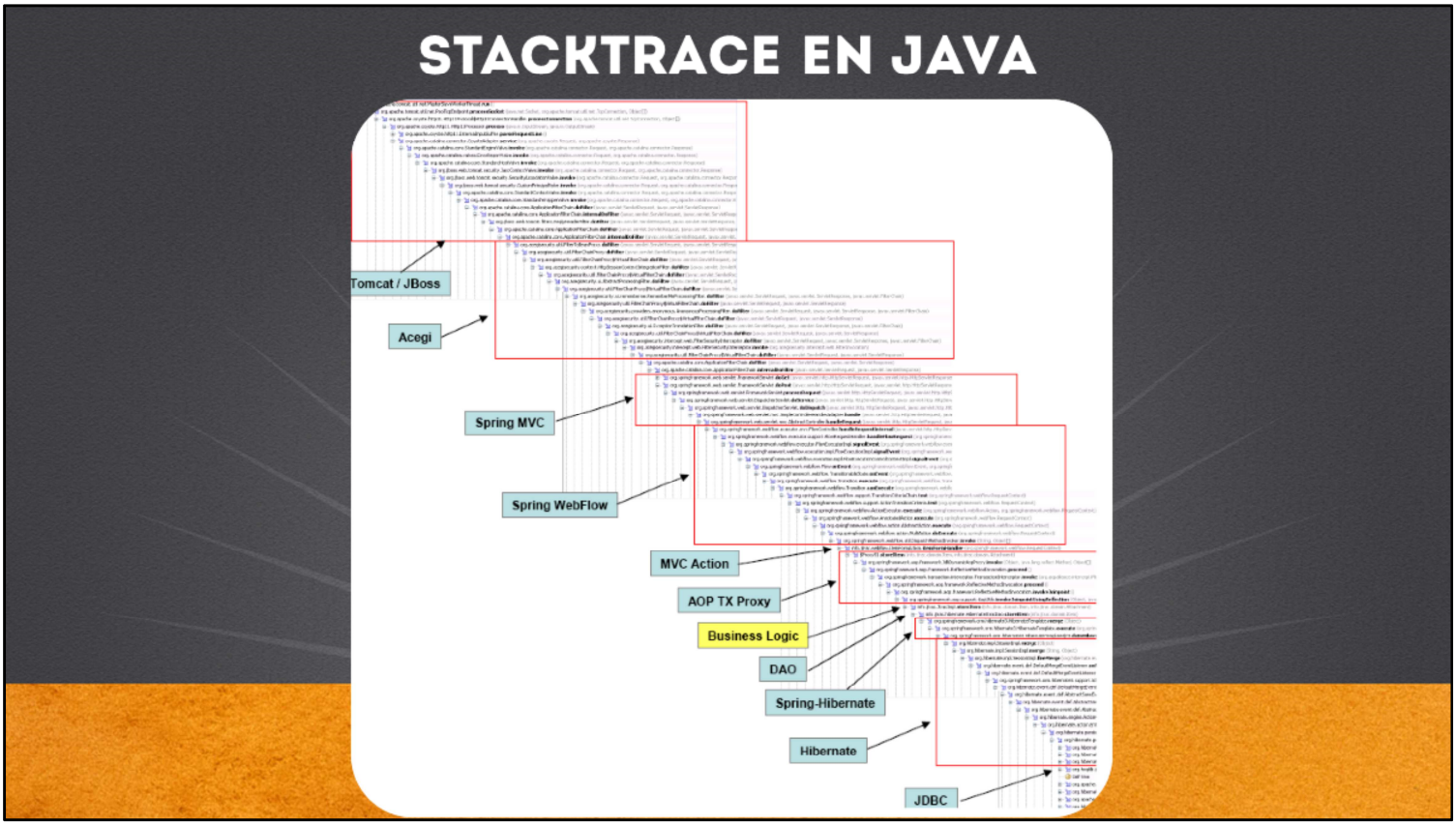
Un ejemplo de una excepción, es acceder a un elemento de un arreglo fuera de rango, esto nos arrojaría una excepción de tipo ArrayOutOfBoundException, por lo que por medio del bloque try/catch/finally es posible procesar este tipo de excepciones.

El bloque try/catch es opcional agregarlo para las excepciones de tipo unchecked exceptions, es decir las excepciones que descienden de la clase Runtime Exception. A lo largo de los ejercicios que vayamos creando iremos viendo varios ejemplos tanto de excepciones de tipo checked y unchecked exceptions, y así irnos familiarizando con cada una de este tipo de excepciones.

En la figura podemos observar algunas de las excepciones más comunes en Java, y podemos observar cuales son las clases padre de algunas de ellas. Podemos ver que de las excepciones más importantes a identificar son la clase Exception y la RuntimeException, ya que estos son los tipos de excepciones que hemos comentado anteriormente, la primera como las excepciones de tipo checked exception y la según las excepciones de tipo unchecked exceptions.

De hecho podemos observar que RuntimeException es una subclase de la clase Exception, sin embargo el compilador tratará las excepciones de manera distinta a todas las excepciones que descienda de la clase RuntimeException, de tal manera que debemos saber y tener claro esta división y cómo trabajar con cada uno de los tipos de excepción que hemos mencionado.

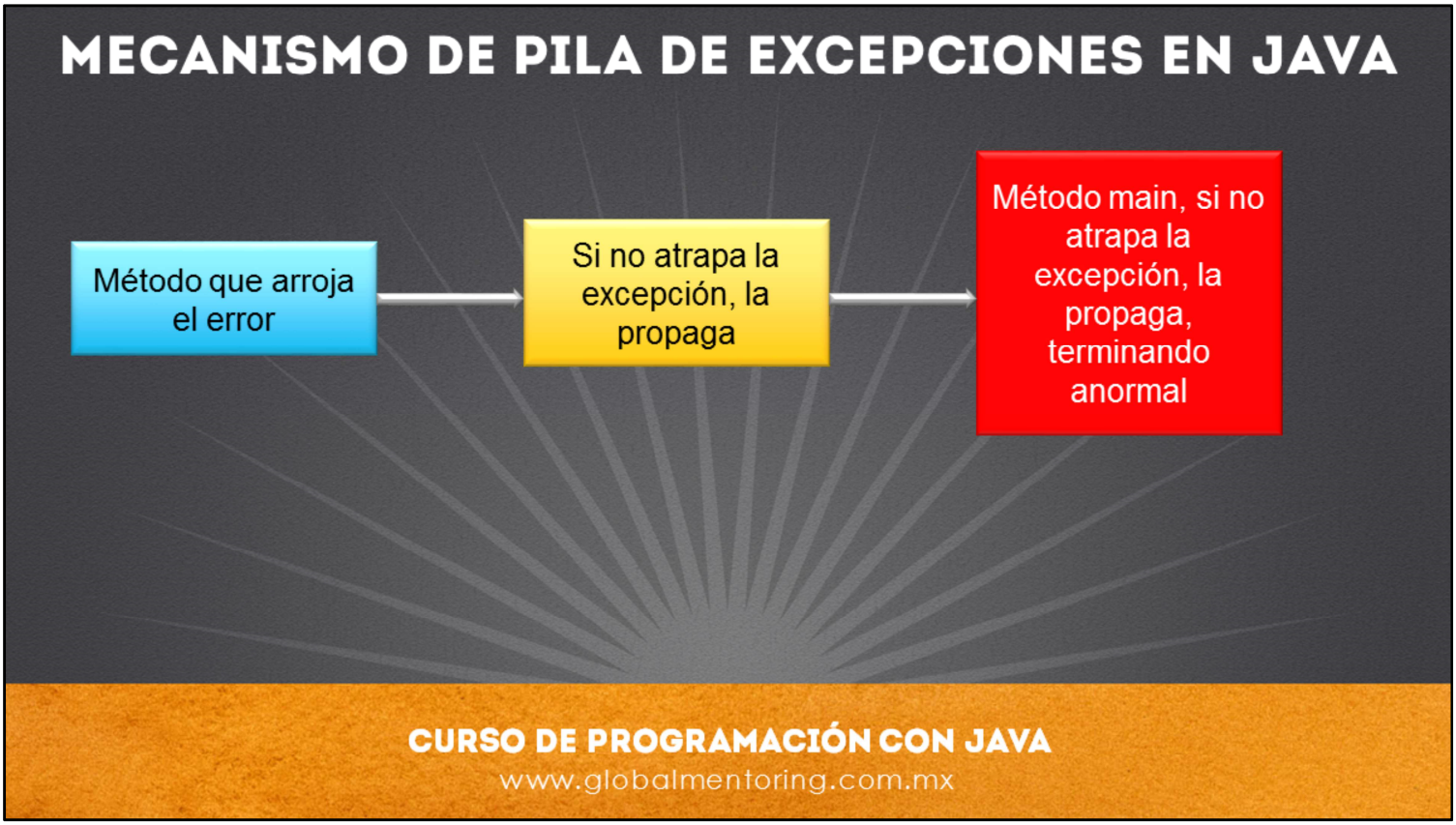
Vemos que la clase padre de todas las clases de excepción es la clase Throwable, y a partir de esta tendremos las excepciones más importantes. Las excepciones de tipo Error son las que normalmente son arrojadas por la máquina virtual de Java, por lo que normalmente serán errores de los que no nos podremos recuperar, en cambio las excepciones de tipo Exception o RuntimeException serán provocadas por nuestro código o el código de otros que estemos utilizando en nuestro programa.



El stacktrace (traza de la pila de errores) de una excepción es el conjunto de mensajes de error desde el origen del error hasta la última clase que recibe el error. De esta manera es precisamente como su nombre lo dice, un rastreo del error de la excepción y con ello podemos conocer más fácilmente el origen del error y por lo tanto cómo corregirlo.

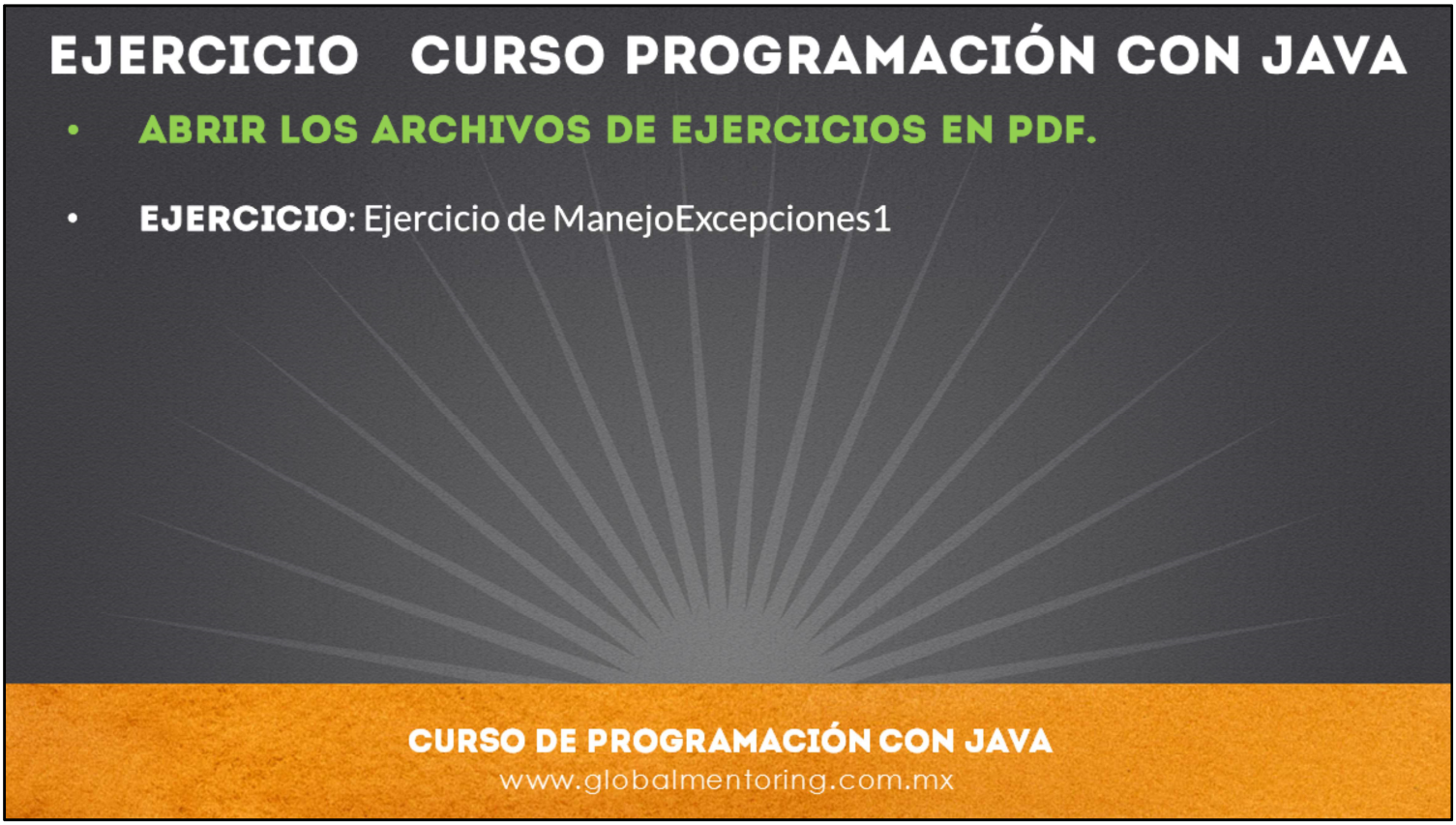
El log o bitácora de una aplicación normalmente incluye mensajes pero también el stack trace de los errores que van sucediendo en nuestra aplicación y por tanto los archivos de log, que son normalmente archivos de texto, serán muy importantes para detectar la raíz de los problemas que sucedan en nuestra aplicación.

Así que debemos estar familiarizados con estos términos ya que son de los que más utilizaremos cuando tengamos problemas en nuestras aplicaciones Java, y veremos cómo analizar los errores a los que nos enfrentemos y así empecemos a ganar experiencia para darles solución.



Podemos observar en la figura de una manera más gráfica el mecanismo en que funciona la pila de excepciones o stacktrace. Según hemos comentado, una pila de excepciones es aquella que va acumulando las excepciones desde el método que originó el problema, hasta el método main, o el último método que recibió el problema, si es que no se cacha antes dicha excepción.

Si una excepción no se atrapa con un bloque try/catch, se propaga la excepción al método que lo mando llamar, y así sucesivamente hasta que algún método lo atrapa, o sino el último método que lo recibe (por ejemplo el método main) arroja finalmente la excepción, terminando de manera anormal el programa.


EJERCICIO CURSO PROGRAMACIÓN CON JAVA
• ABRIR LOS ARCHIVOS DE EJERCICIOS EN PDF.
• EJERCICIO: Ejercicio de ManejoExcepciones1
CURSO DE PROGRAMACIÓN CON JAVA
www.globalmentoring.com.mx

En Global Mentoring promovemos la Pasión por la Tecnología Java. Te invitamos a visitar nuestro sitio Web donde encontrarás cursos Java Online desde Niveles Básicos, Intermedios y Avanzados, y así te conviertas en un experto programador Java.

Además agregamos nuevos cursos para que continúes con tu preparación como programador Java profesional. A continuación te presentamos nuestro listado de cursos:

- Programación con Java
- Fundamentos de Java
- Programación con Java
- Java con JDBC
- HTML, CSS y JavaScript
- Servlets y JSP's
- Struts Framework
- Hibernate Framework
- Spring Framework
- JavaServer Faces
- Java EE (EJB, JPA y Web Services)
- JBoss Administration
- Android con Java
- HTML5 y CSS3

Datos de Contacto:

Sitio Web: www.globalmentoring.com.mx
Email: informes@globalmentoring.com.mx