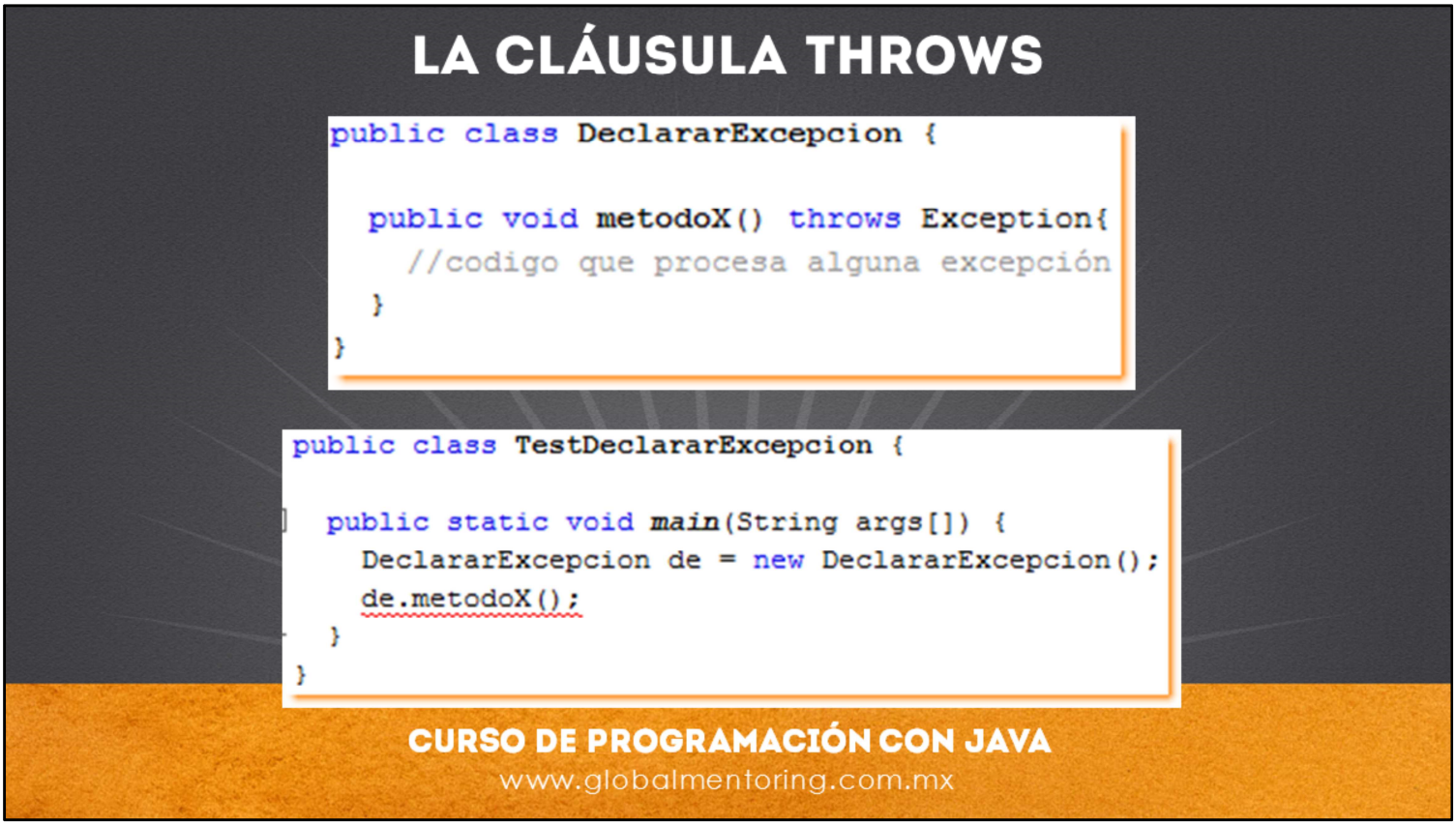
Hola, te saluda nuevamente Ubaldo Acosta. Espero que estés listo para comenzar con esta lección..

Vamos a estudiar el tema de excepciones en Java.

¿Estás listo? ¡Vamos!

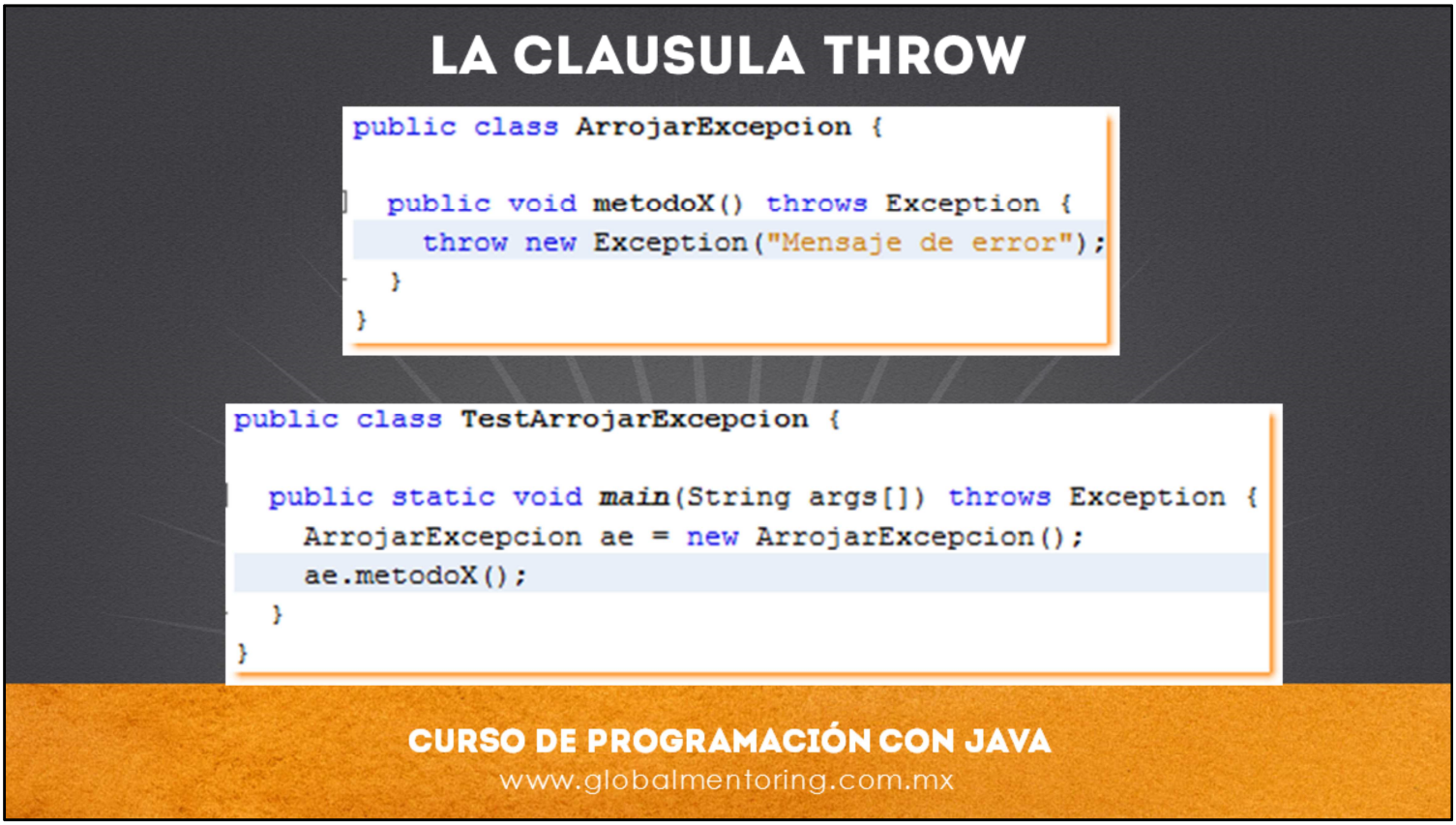


En esta lección vamos a continuar revisando más temas acerca de las excepciones en Java. Vamos a hablar ahora de la clausula throws, la cual nos permite especificar el tipo de excepción que arroja cierto método.

Esto se puede deber a que el método no atrapó la excepción y por lo tanto se va a propagar al método que lo mando llamar.

Una excepción que extiende de Exception SÍ estamos obligados a declararla en la firma del método, ya que este tipo de excepciones, al igual que cuando comentábamos que se deben procesar por un bloque try/catch, si no son procesadas por este bloque, entonces estamos obligados a declararlas dentro del método donde se hace uso del método que arroja la excepción.

Una excepción que extiende de RuntimeException NO estamos obligados a declararla en la firma del método al usar un método que arroje este tipo de excepciones.



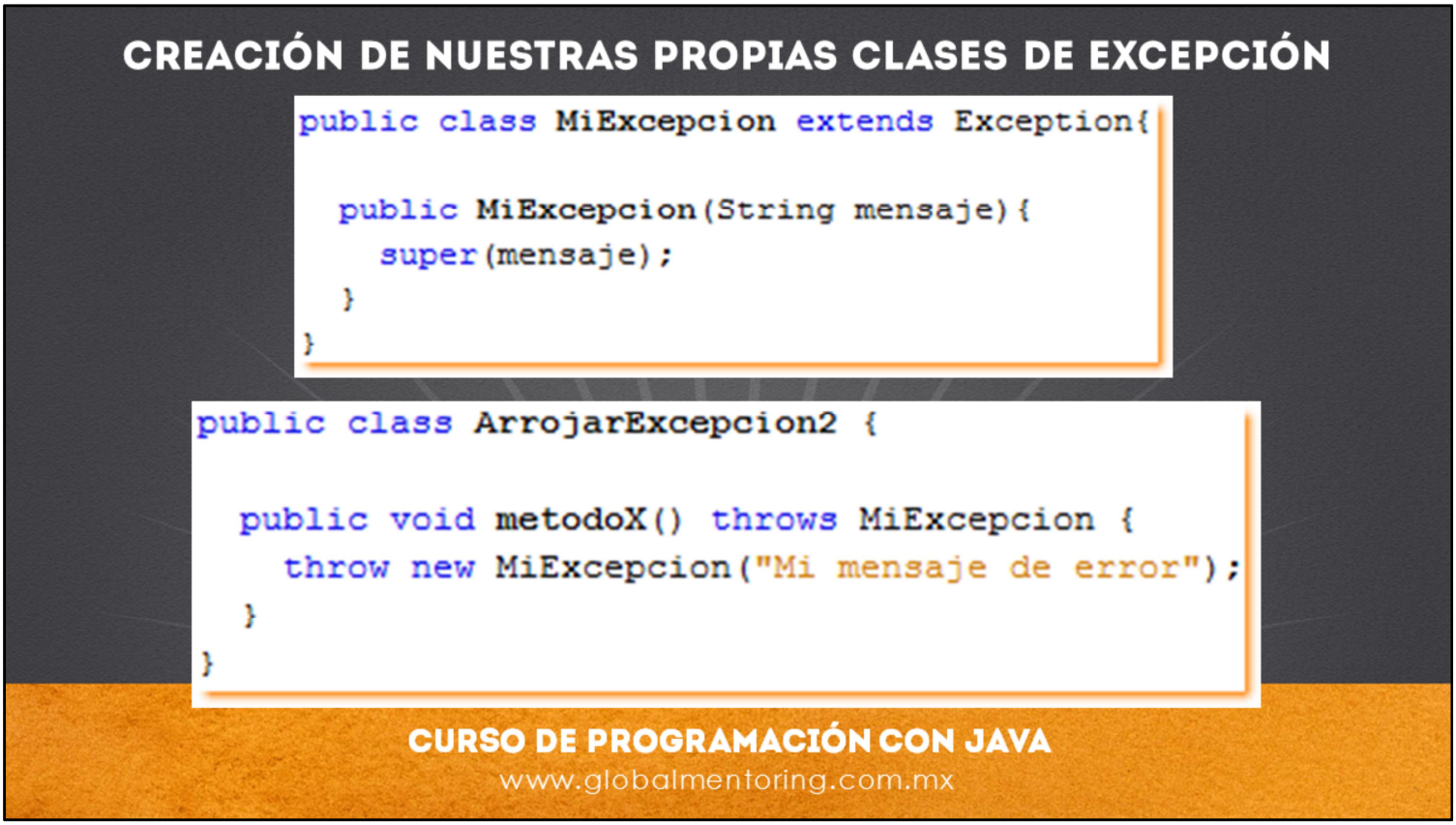
Vamos a revisar ahora la cláusula throw. Esta palabra reservada nos permite LANZAR explícitamente una excepción.

Por ejemplo, podemos utilizar una clase de excepción que extiende de la clase Exception o cualquier otra clase de tipo exception y en conjunto con el operador new, crear un nuevo objeto de esta clase y así arrojar una excepción de manera explícita.

En el código podemos observar cómo dentro del métodoX se lanza una nueva excepción de tipo Exception, utilizando la palabra reservada throw seguida de new y la clase de tipo Exception que queremos instanciar. Normalmente las clases de excepción en su constructor tiene una cadena que será el mensaje que se mandará al stacktrace al momento en que ocurra el problema o se lance la excepción.

Posteriormente lo que observamos es el que método main al hacer uso de metodoX obliga a procesar la excepción por medio de un bloque try/catch, o a declarar en la firma del método que es metodoX puede llegar a lanzar la excepción de tipo Exception, y lo vemos en la firma del método por medio de la palabra throws Exception.

De hecho, con un proceso similar es como nuestra aplicación pueden convertir excepciones de tipo Exception (checked exceptions) a tipo RuntimeException (uncheked exceptions), creando nuestras propias clases de tipo runtime exception y al momento de envolver una excepción de tipo Exception con un bloque try/catch, se puede relanzar una excepción de tipo runtime exception, de esta manera no obligamos a los métodos que utilicen nuestros métodos a que declaren el tipo de excepción que se puede arrojar, ya que se han convertido a tipo runtime exception. Más adelante vamos a ver cómo realizar esta conversión.

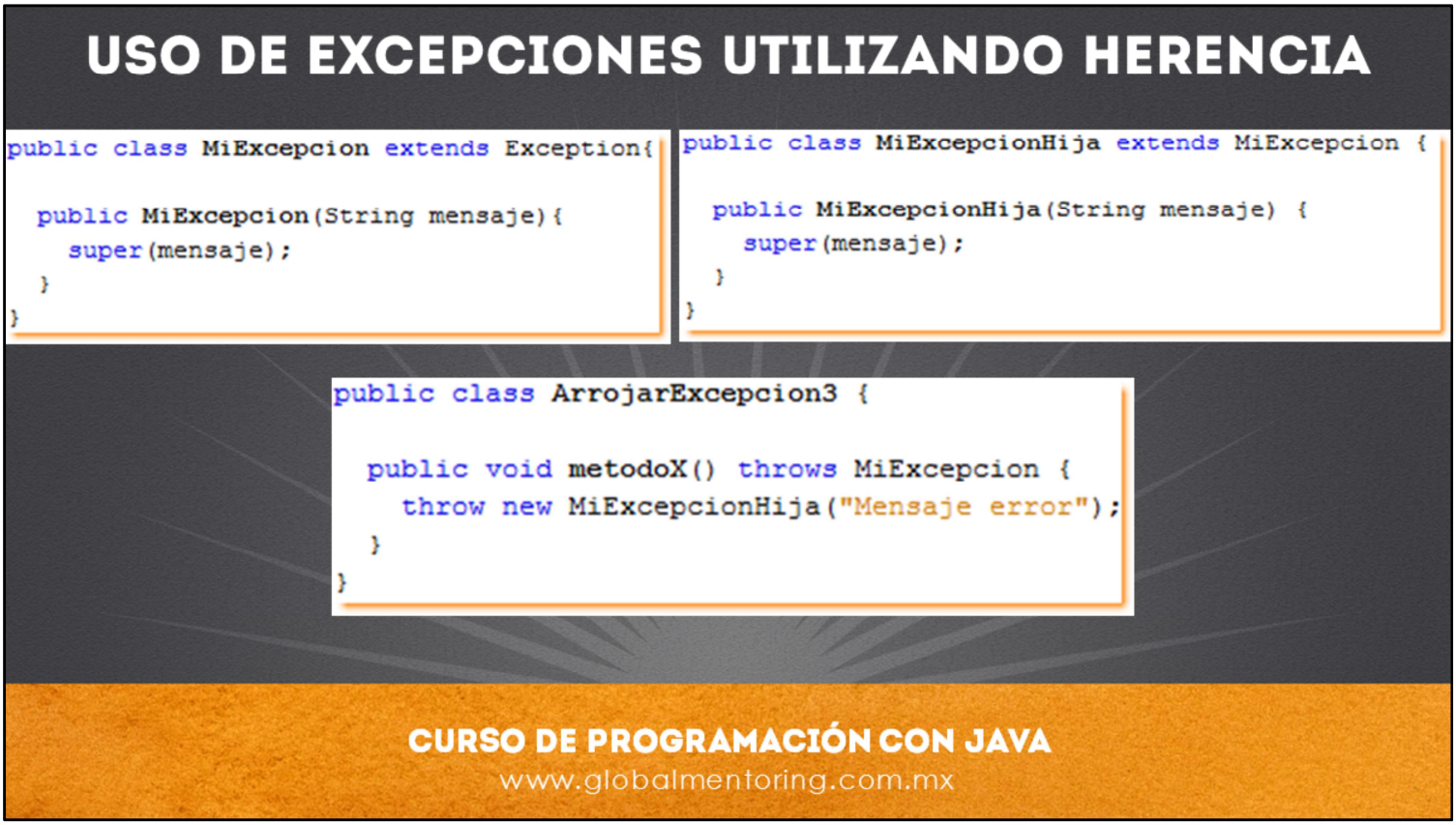


Vamos a ver a continuación cómo podemos crear nuestras propias clases de excepción, las cuales pueden extender de la clase Exception, o RuntimeException.

La intención de crear nuestras propias clase de Excepción es personalizar los mensajes de error que se envían al usuario, por ejemplo, un error de Base de Datos, se puede atrapar y convertir en un mensaje de error legible al usuario que usa nuestra aplicación, o como hemos comentado anteriormente, podemos crear clase de excepción que desciendan de la clase runtime exception para no obligar a los usuarios a usar bloques try/catch o declarar en las firmas de los métodos el tipo de excepción que arrojará en caso de que suceda alguna excepción del tipo que hemos declarado.

Como observamos en el código, estamos creando una clase de tipo MiExcepcion la cual hereda de la clase Exception, sin embargo puede extender de cualquier otra clase de excepción que nos interese. Posteriormente esta clase contiene un constructor el cual recibe el mensaje que nos servirá para inicializar el atributo mensaje de la clase padre, mandando a llamar el constructor padre por medio de la palabra super. Y con esto queda inicializada nuestra excepción.

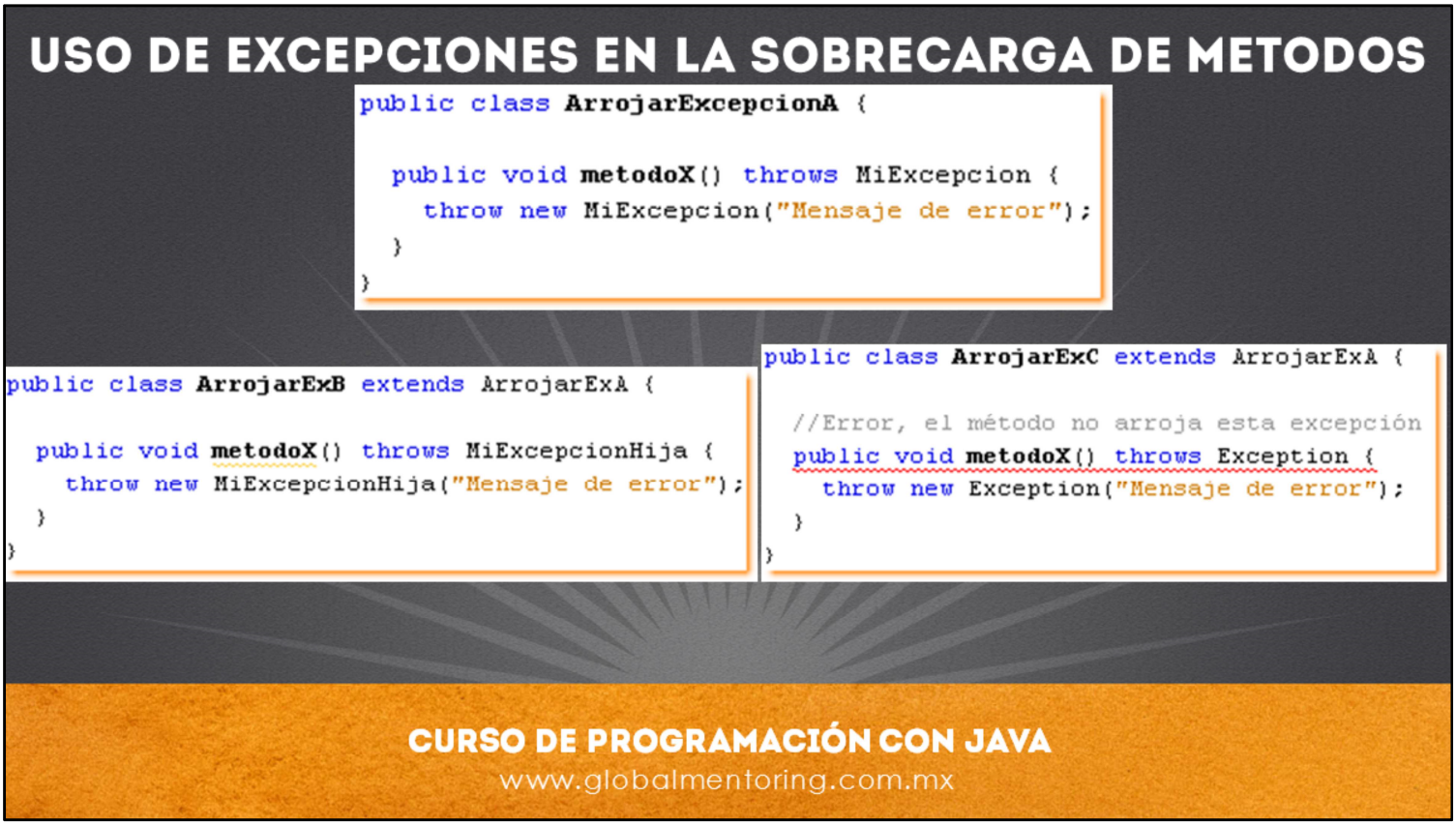
Porteriormente en la clase ArrojarExcepcion2 podemos observar que se lanzamos una excepción de tipo MiExcepcion, y como es de tipo Excepcion, se obliga al metodoX a declarar en su firma el tipo de excepción que se lanzará en caso de error. Así es como crearemos nuestras propias clase de excepción y así también es como las usaremos dentro de nuestro código Java.



Vamos a revisar ahora el tema de uso de excepciones utilizando herencia. En el código mostrado vemos dos clases, una llamada MiExcepcion la cual extiende de la clase Exception. Posteriormente declaramos una clase llamada MiExcepcionHija la cual es una subclase de MiExcepcion, por lo tanto tenemos una jerarquía de clase de excepción que hemos creado.

Por otro lado, en la clase ArrojarExcepcion3 se arroja una subclase llamada MiExcepcionHija dentro del metodoX, pero si observamos a firma del método, el método declara que se arroja una excepción que es una superclase llamada MiExcepcion.

La intención de hacer esto, es que el método internamente puede arrojar excepciones que son subclases, pero la clase que use a este método solo procesa la excepción más genérica, y con ello podemos tener un rango más amplio para el manejo de excepciones, utilizando precisamente el concepto de herencia pero ahora dentro de la definición de clases de tipo Exception, y al igual que cuando trabajamos con polimorfismo, mucho del código en cualquier lenguaje de programación y también en Java, es crear métodos lo más genéricos posibles que nos ayuden a englobar más posibles excepciones en menos líneas de código, que al final nos ahorrará costos de mantenimiento de nuestras aplicaciones, al tener código lo más genérico y compacto posible.



Vamos a ver a continuación el uso de excepciones al momento de sobrecargar nuestros métodos en Java. Algunas de las reglas que debemos respetar son:

Un método que sobreescribe a un método padre que declara una excepción SÍ PUEDE:

- No arrojar ninguna excepción, es decir, omitir la excepción en su firma.
- Arrojar una o más excepciones arrojadas por el método padre.
- Arrojar una o más excepciones que son subclases de la excepción arrojada.

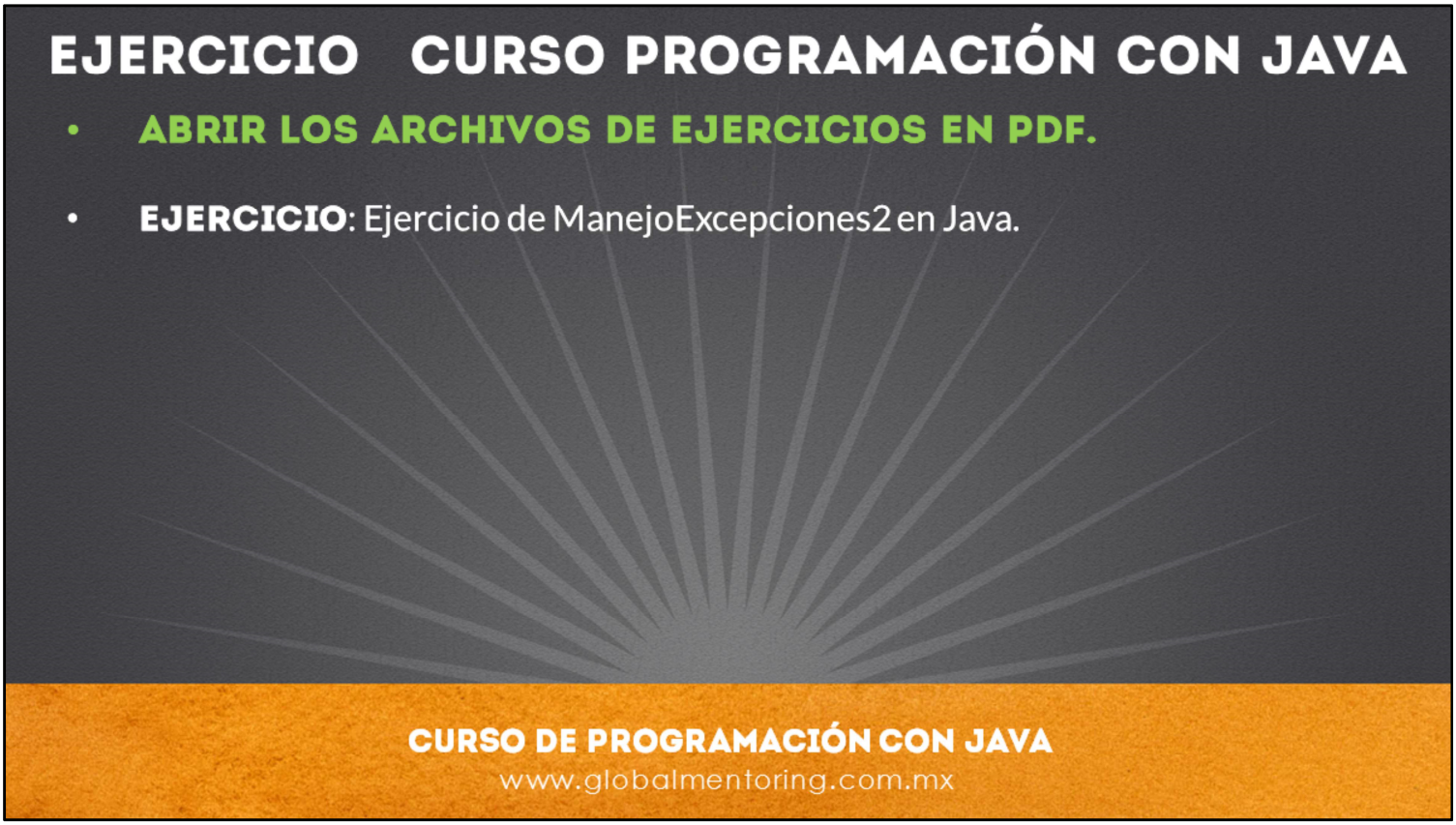
Un método que sobreescribe a un método padre que declara una excepción NO PUEDE:

- Arrojar excepciones adicionales no arrojadas en el método padre.
- Arrojar superclases de las excepciones arrojadas por el método padre.

En el código mostrado observamos una clase padre (ArrojarExcepcionA) y dos clases hijas (ArrojarExB y ArrojarExC), las cuales sobreescriben el metodoX definido en el padre. Lo que estamos aclarando en este código las excepciones que podemos utilizar en la firma de los métodos sobreescritos por las clases hijas.

Por un lado observamos que la clase ArrojarExB en la firma del metodoX en lugar de utilizar la misma excepción que en la firma del método de la clase padre (MiExcepcion), se utiliza una subclase (MiExcepcionHija), de esta manera es claro que no necesariamente debemos utilizar la misma clase de excepción en la firma del método, sino que puede ser alguna clase dentro de la jerarquía de clases de excepción que estamos utilizando, siendo siempre subclases, y no superclases.

Por otro lado, observamos en la clase ArrojarExC que se utiliza precisamente una super clase (Exception) de la clase de excepción utilizada en el padre (MiExcepcion), y por lo tanto esto no es válido y existe un problema de compilación en nuestro código. Más adelante veremos un ejemplo de este tipo de manejo de excepciones en la firma de los métodos que se sobreescriben.



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