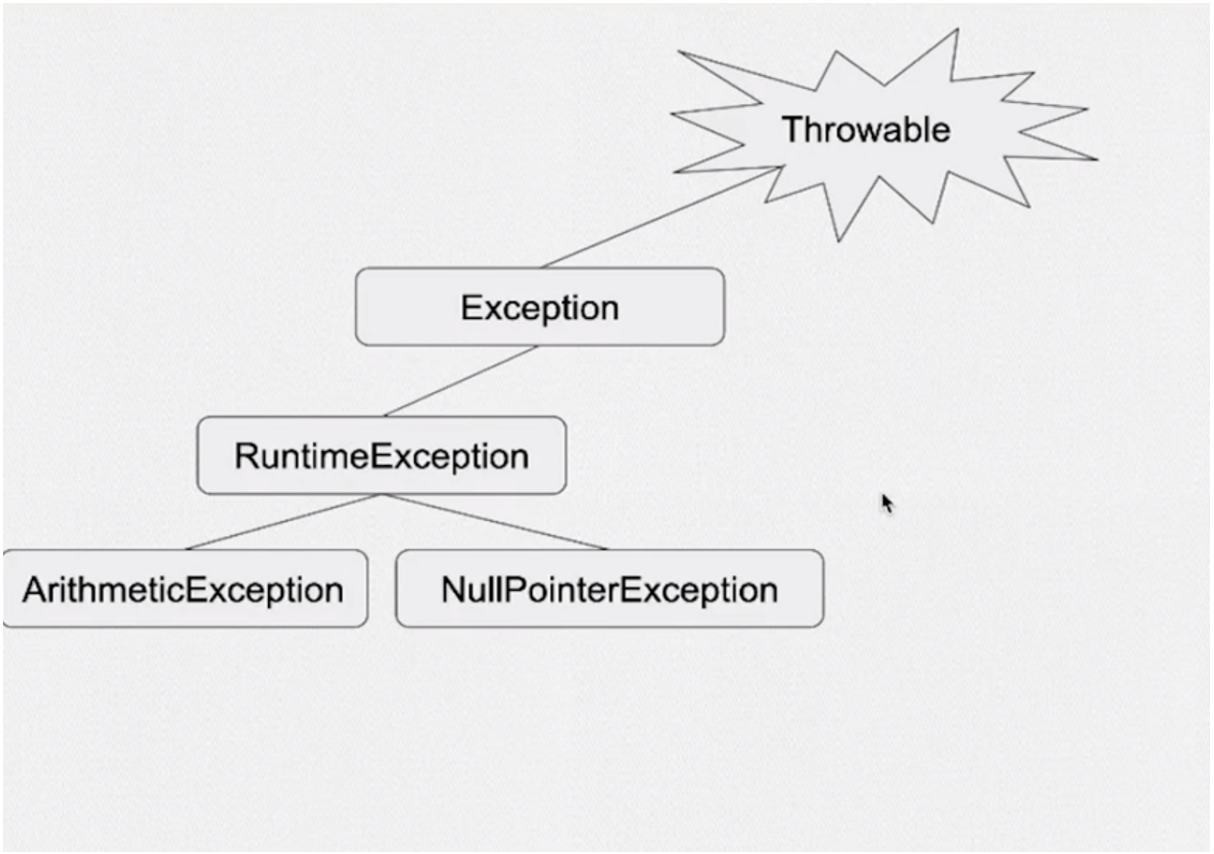

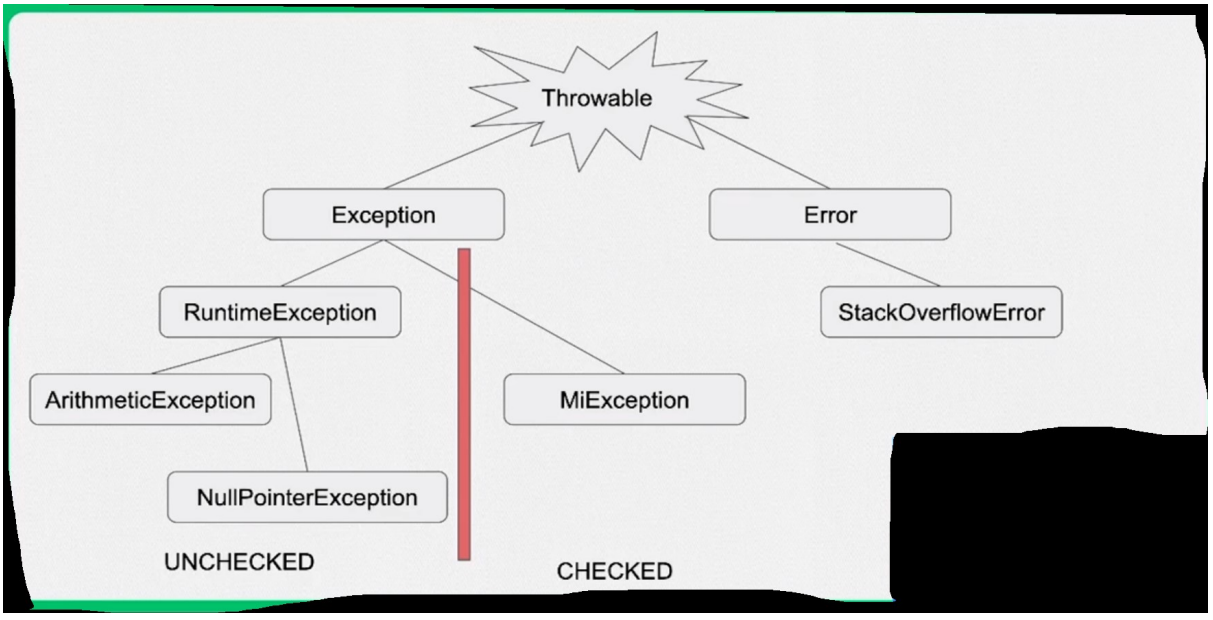

Errores = son lanzados por programadores que mantiene la jvm.

Exeption = Son lanzados por los desarrolladores que programan sobre la jvm .

Unchecked ----> Runtime→El compilador continúa su proceso, por ser una exeption no verificada

Checked--------> ejp Exception → el compilador obliga a tratar la exeption o si no, no compilara.

- Existe una gran jerarquía de clases que representan excepciones. Por ejemplo, *ArithmeticException* es hija de *RuntimeException*, que hereda de *Exception*, que a su vez es hija de la clase de excepciones más ancestral, *Throwable*. Conocer bien esta jerarquía significa saber cómo usar las excepciones en su aplicación.
- *Throwable* es la clase que debe extenderse para poder lanzar un objeto en la pila (usando la palabra reservada *throw*)
- Es en la clase *Throwable* donde tenemos casi todo el código relacionado con las excepciones, incluyendo *getMessage()* e *printStackTrace ()*. El resto de la jerarquía tiene solo algunas sobrecargas de constructores para comunicar mensajes específicos.
- La jerarquía que comenzó con la clase *Throwable* se divide en excepciones y errores. Las excepciones se utilizan en los códigos de aplicación. Los errores son utilizados exclusivamente por la máquina virtual.
- Las clases que heredan de *Error* se utilizan para informar errores en la máquina virtual. Los desarrolladores de aplicaciones no deben crear errores que hereden de *Error*.
- *StackOverflowError* es un error de máquina virtual para informar que la pila de ejecución no tiene más memoria.
- Las excepciones se dividen en dos categorías amplias: las que el compilador comprueba obligatoriamente y las que no.
- Los primeros se denominan *checked* y se crean por pertenecer a una jerarquía que no pasa *RuntimeException*.
- Los segundos están *unchecked* y se crean como descendientes de *RuntimeException*.