**2016 Starting Out with Java Tony Gladis**

**Handling Exceptions**

* Una excepción es un objeto que es generado en memoria como el resultado de un error o un evento inesperado.
* Cuando una excepción es generada, se dice que ha sido “lanzada/arrojada”
* Si la excepción no se detecta y manipula por la aplicación, causará que la aplicación se detenga.
* Para detectar una excepción que ha sido lanzada y prevenir que detenga nuestra aplicación, Java nos permite crear manejadores de excepciones.
* Un manejador de excepciones es una sección de código que responde, atiende a las excepciones cuando son lanzadas.
* El proceso de interceptar y responder a las excepciones se llama **manejo de excepciones**.
* Si nuestro código no maneja la excepción cuando es lanzada, el manejador de excepciones que trae java por defecto trata con dicha excepción.
* El manejador de eventos que trae Java por defecto imprime un mensaje de error y termina el programa.

**Exception Classes**

* Una excepción es un objeto. Los objetos de tipo Excepción son credos de las clases en la API de Java.
* La API tiene una extensa jerarquía de clases de excepciones.
* Todas las clases en la jerarquía heredan de la clase **Throwable.**
* Debajo de la clase Throwable están las clases **Error** y **Excepction**
* Las Clases que heredan de **Error** son para excepciones que son lanzadas cuando un error crítico ocurre, tal como un error interno de la máquina virtual de java. Nuestras aplicaciones no deberían de manejar estos errores porque son el resultado de algo serio.
* Todas las excepciones que manejaremos son instancias de clases que heredan de **Exception**.
* Dos ejemplos son que heredan de Excepcion son IOException y RuntimeException. Estas clases también sirven como superclases.
* **IOException** sirve como superclase para exceptiones que están relacionadas a operaciones de entrada y salida.
* **RuntimeException** sirve como superclase para excepciones que resultan de errores de programación, como el acceso a un elemento que no existe en un array.
* Dos clases que heredan de la clase IOException son: **EOFException** y **FileNotFoundException**.
* **EOFException** es lanzado cuando una aplicación intenta leer más allá del final de un archivo.
* **FileNotFoundException** es lanzado cuando una aplicación intenta abrir un archivo que no existe.
* Las clases de las excepciones están en paquetes en la API de Java. Por ejemplo, FileNotFoundException están en el paquete java.io
* Cuando manejamos una excepción que no están en el paquete java.lang necesitamos importar las sentencias apropiadas.

**Handling an Exception**

* Para manejar una excepción, usamos una sentencia **try**.

|  |
| --- |
|  |

* El bloque try (***try block statements***) es una o más sentencias que se ejecutan y que pueden lanzar una excepción.
* La aplicación no se detendrá si el bloque try lanza una excepción.
* El bloque catch esta seguido por el código (*Exception parameterName*). Aquí, *ExceptionType* es el nombre de la clase excepción y *parameterName* es el nombre de la variable/instancia.
* En *catch block statemens*  se colocan los mensajes indicando el error ocurrido.
* El código en el bloque **try** es ejecutado. Si este código lanza una excepción, La Máquina Virtual de Java busca una cláusula **catch** que puede tratar con esa excepción.
* Para que la cláusula catch pueda manejar la excepción, su parámetro debe ser del mismo tipo que la excepción generada.
* Cuando manejamos excepciones, podemos utilizar referencia polimórfica en el parámetro de la cláusula catch.
* Si una clausula catch usa un parámetro de tipo Exception, entonces es capaz de capturar cualquier excepción que herede de la clase Exception.
* Por ejemplo, si utilizamos la clase NumberFormatException para capturar una excepción, podemos cambiarla por la clase Exception y el código aun funcionara por que la clase NumberformatException hereda de la clase Exception.
* Se pueden poner múltiples catch después del try, las excepciones deben ser de diferentes tipos. No se debe repetir el tipo de excepción.

**FINALLY**

* La sentencia try puede tener la cláusula **finally** (opcional).
* Debe aparecer después de la última cláusula catch (después de todas las cláusulas catch).
* Dentro de las llaves del block finally se colocan una o más líneas de código que siempre se ejecutan después de que el bloc try se halla ejecutado y después de que todos los bloques catch hayan sido ejecutados ya sea si ocurre una excepción o no.