Universidad Continental

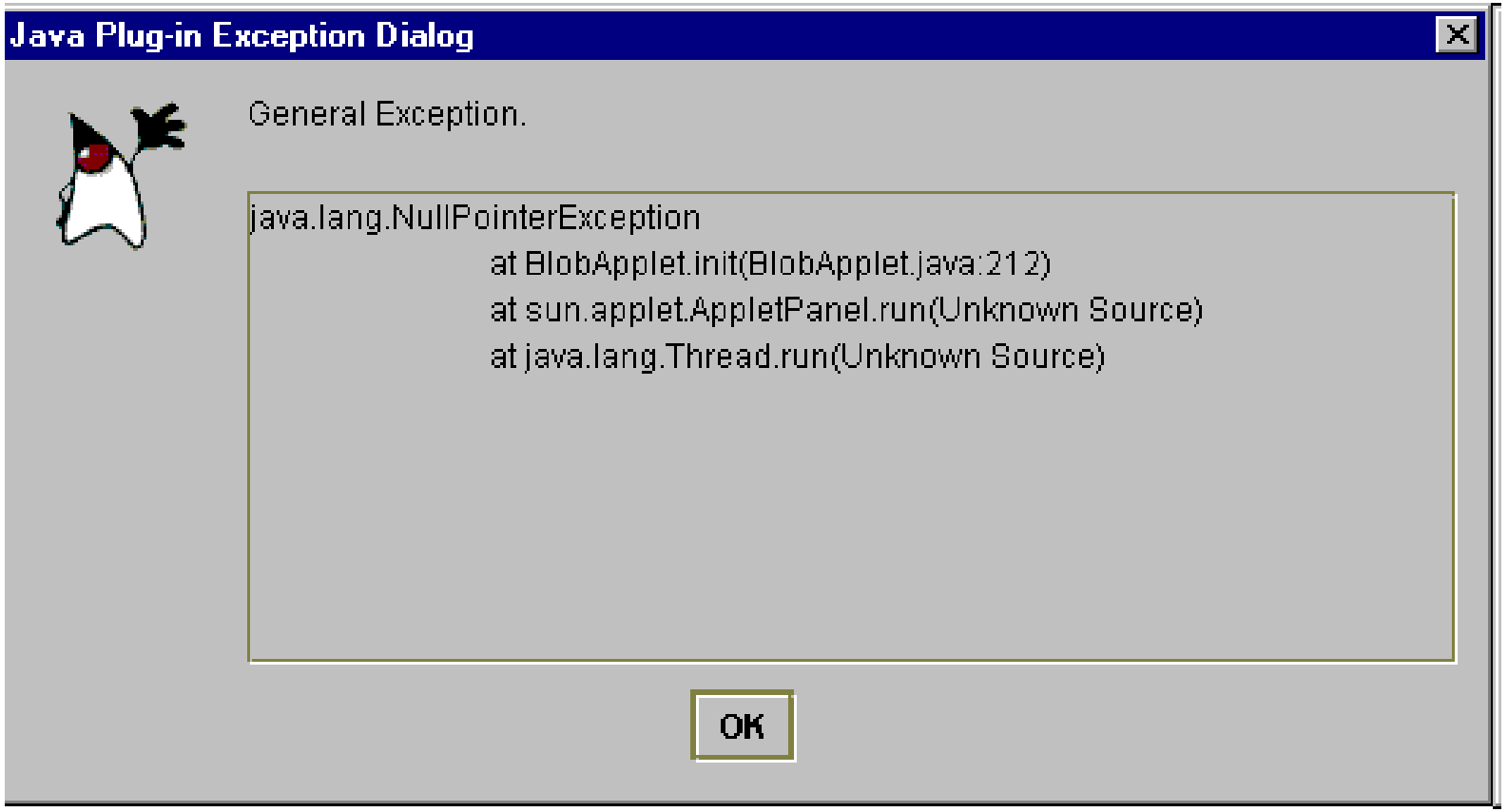
UNIDAD 02
EXCEPCIONES

PROGRAMACIÓN ORIENTADA A OBJETOS

Eric Gustavo Coronel Castillo
ecoronel@continental.edu.pe

OBJETIVO

Control de errores en tiempo de ejecución.



INTRODUCCIÓN

- Uno de los mayores problemas en la programación es el tratamiento de errores, que pueden ser generados por:
  - Fallas o limitaciones del hardware (por ejemplo errores de lectura de archivos)
  - Fallas en el software (casos en los cuales no se ha considerado cierta casuística en el desarrollo del software).
- Para facilitar el tratamiento de errores en Java se ha creado el concepto de Excepción, el cual se refiere a una situación de error en la ejecución de un programa, cada vez que ocurre una excepción (un error) el programa debe tratarla, normalmente mostrando un mensaje de error y ejecutando alguna rutina de tratamiento de errores.

TIPOS DE ERRORES

Errores de Sintaxis

- Los errores en la sintaxis son causados cuando el compilador de Java (javac) no puede reconocer una instrucción. Esto causa que el compilador devuelva un mensaje de error, usualmente con una línea de código de referencia.
- También se conoce a los errores de sintaxis como errores en tiempo de compilación.

TIPOS DE ERRORES

Errores de Lógica

- Los errores de lógica son conocidos como BUGS. Estos son los errores que nos tomarán un tiempo hasta encontrarlos. Si damos a elegir, cualquier programador elegiría los errores en tiempo de compilación.
- Para encontrar un error de lógica debemos identificar la clase donde podría estar el error, y si es posible el métodos o posibles métodos donde podría estar el error, luego de eso hacer una depuración (Debug) línea por línea (trace) en cada uno de los métodos.

TIPOS DE ERRORES

Errores de Ejecución

- Los errores de ejecución se producen cuando la aplicación esta en producción ó prueba (Testing), y aparecen por una situación anormal durante la ejecución de alguna instrucción.
- Por ejemplo:
  - Una división por cero.
  - No se tiene permiso de escritura sobre un archivo.
  - La base de datos no existe.
  - No se tiene permiso de acceso a una base de datos.

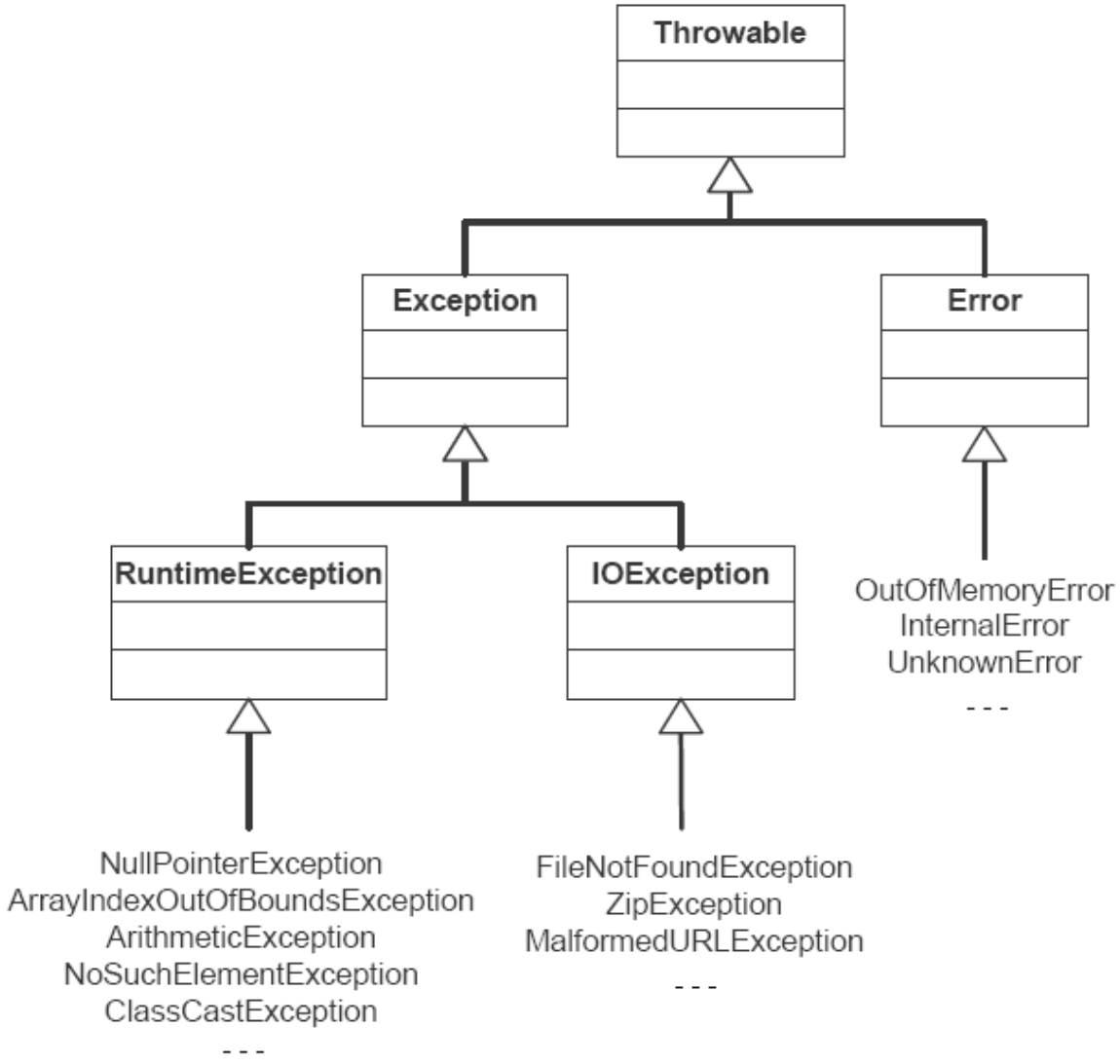
¿QUÉ ES UNA EXCEPCIÓN?

- Una excepción en Java es un objeto que describe una condición excepcional, es decir, un error que se ha dado en una parte del código.

- Cuando se origina un error se produce una condición de excepción, se crea un objeto que representa esa excepción y se lanza al método que ha causado el error. Este método puede elegir entre gestionar él mismo la excepción ó pasarla al método que lo ha invocado. De cualquiera de las dos formas, en un punto determinado se capturará la excepción y se procesará.

- Las excepciones pueden ser generadas por el intérprete de Java o de forma manual por el propio código. Normalmente, las excepciones generadas por Java están relacionadas con errores fundamentales que violan las reglas del lenguaje Java o las restricciones del entorno de ejecución Java. Las excepciones generadas de forma manual se utilizan generalmente para informar acerca de alguna condición de error personalizada, por ejemplo, un error en el proceso de la lógica del negocio.

TIPOS DE EXCEPCIONES



GESTIÓN DE EXCEPCIONES

Esquema General

```
try{
      // Bloque de instrucciones a controlar
} catch ( TipoDeExcepción1 var1 ) {
     // Gestión de excepción de tipo TipoDeExcepción1
} catch ( TipoDeExcepción2 var2 ) {
      // Gestión de excepción de tipo TipoDeExcepción2
}
. . .
finally {
      // Bloque de instrucciones que siempre se debe ejecutar
}
```

GESTIÓN DE EXCEPCIONES

Instrucción throw

Sirve para generar excepciones explícitamente.

```
try{


 if( n2 == 0 )  {
       throw new Exception("n2 debe ser mayor que 0");
 }


} catch ( Exception e ) {

 // Gestión de excepción

}
```

GESTIÓN DE EXCEPCIONES

Palabra clave: throws

- Si un método es capaz de generar una excepción que él mismo no puede gestionar, se debe especificar este comportamiento de manera que los métodos que llamen a ese primer método puedan protegerse contra esa excepción. Para ello se debe incluir la cláusula **throws** en la definición del método.
- La cláusula **throws** declara una lista de excepciones que el método puede lanzar. Esto es necesario para todas las excepciones, excepto las de tipo **Error** ó **RuntimeException**, o cualquiera de sus subclases. Todas las demás excepciones que un método puede lanzar se deben declarar en la cláusula **throws**, si no es así se produce un error de compilación.

```
tipo nombre_método ( lista_de_parametros ) throws lista_de_excepciones {

    // Implementación

}
```

EXCEPCIONES PERSONALIZADAS

- Aunque las excepciones que incorpora Java gestiona la mayoría de errores más comunes, es probable que se presenten situaciones en la que no encontremos una excepción apropiada para cierto tipo de errores muy específicos de un proceso en particular.
- Por ejemplo, supongamos que debemos controlar el rango de una nota, en este caso es de 0 a 20, no contamos con una excepción para este caso en particular.
- Para estos casos muy específicos, podemos crear nuestra propia excepción, para eso debemos crear una subclase de la clase Exception.

EXCEPCIONES PERSONALIZADAS

```
public class NotaFueraDeRango extends Exception {

    public NotaFueraDeRango () {
            super( "Error: Nota debe estar en el rango de [0,20]." );
    }

    public NotaFueraDeRango( String msg ) {
            super(msg);
    }

}
```

PROYECTO EJEMPLO

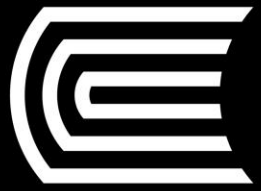
- La institución educativa **EduTec** necesita poner a disposición de sus estudiantes una aplicación que les permita calcular su promedio de una manera fácil y segura.
- Se le solicita a usted hacer el diseño y desarrollo de este requerimiento.
- Debe aplicar control de errores mediante excepciones para evitar que el programa aborte su ejecución frente a cualquier error que podría presentarse durante su ejecución.
- Se sabe que son 4 prácticas calificadas, de las cuales se elimina la de menor puntaje.
- También se tiene un examen parcial y un examen final.

PROYECTO EJEMPLO

- Para calcular el promedio final se debe aplicar la siguiente fórmula:

$$PF = \frac{PP + EP + 2 * EF}{4}$$

- Dónde:
  - PF : Promedio Final
  - PP : Promedio de Practicas
  - EP : Examen Parcial
  - EF : Examen Final
- Debe aplicar:
  - Excepciones personalizadas
  - Programación en capas para plantear la solución del problema.
  - Swing para crear las interfaces de usuario.



ucontinental.edu.pe