Excepciones

Contenido

Introducción

Las excepciones son los eventos que se producen cuando hay un error en la ejecución del programa. Para controlarlas bien, es necesario agrupar los errores y tratarlos con códigos alternativos, además de propagar los errores y separar el código de tratamiento de errores del resto.

Tipos de excepción:

- **Verificada (checked):** Son excepciones que el compilador comprueba si están capturadas o manejadas, son errores recuperables. (El propio entorno te avisa que su captura es obligatoria. Un ejemplo es el **throws IOException** que se ve en JavaFX
- **No verificada (unchecked):** Son los errores de programación. Es posible controlarla o capturarla, pero no es obligatorio, aunque puede dar la ocasión de que finalice forzosamente el programa. Los comunes son **NullPointerException** (elemento nulo) o **InputMismatchException** (Por ejemplo, se mete un carácter no numérico en un int)

Métodos de Excepción

- **String toString():** Devuelve un string con el nombre de la excepción y el mensaje de esta.
- **Class<?> getClass():** Devuelve la clase de la excepción
- **String getMessage():** Devuelve el mensaje
- **PrintStackTrace:** Imprime el objeto desde el que se invoca junto a las llamadas a los métodos donde se ha producido.

Manejos de Excepciones

Captura de Excepciones

Es el llamado bloque “try-catch”, utilizado para guardar dentro del try el código que pueda hacer que salte la excepción, y dentro del catch el código de gestión.

```
public static void main(String[] args) {
    try{
        int num = new Scanner(System.in).nextInt();
    }catch (Exception e){
        System.out.println("Eso no es un número");
        System.out.println(e.toString());
    }
}
```

Se pueden concatenar capturas, es decir:

```
public static void main(String[] args) {
    try{
       [...]
    }catch (SQLException e){
        System.out.println("Error de base de datos");
        System.out.println(e.toString());
    } catch (IOException e){
        System.out.println(e.toString());
    }
}
```

Además, se puede declarar un bloque “finally”, que se ejecutará después del bloque try-catch, ya sea si no se produce una excepción o, en su defecto, se produce, sin importar que esté capturada o no en un catch.

```
public static void main(String[] args) {
    try{
       [...]
    }catch (SQLException e){
        System.out.println("Error de base de datos");
        System.out.println(e.toString());
    } catch (IOException e){
        System.out.println(e.toString());
    } finally {
        System.out.println("Capturado");
    }
}
```

Captura de excepciones con recursos

Es un try catch donde el comando a fallar se declara dentro de la declaración del try, útil para bases de datos o entrada de ficheros:

```
public static void main(String[] args) {
    try(Connection db = DriverManager.getConnection("x")){
       [...]
    }catch (SQLException e){
        System.out.println("Error de base de datos");
        System.out.println(e.toString());
    } catch (IOException e){
        System.out.println(e.toString());
    } finally {
        System.out.println("Capturado");
    }
}
```

Propagación de Excepciones

En vez de tener que capturar todas las excepciones, se pueden propagar hacia el método para que se capture allí. Se hace en la declaración del método, con **throws:**

```
public static void main(String[] args) throws IOException,
SQLException, NullPointerException {
     [...]
}
```

En este caso no se imprimiría nada, pero si hiciéramos lo siguiente, sí que saldría información:

```
public static void main(String[] args){
     try{
     }catch (SQLException e){
         System.out.println(e.toString());
     } catch (IOException e){
         System.out.println(e.toString());
     }
}
public void metodo1() throws IOException{
}
public void metodo3() throws SQLException{
}
```

Lanzamiento de Excepciones

Además de capturarlas, se pueden lanzar de forma intencionada otras excepciones cuando al usuario le interesa. Se hace con **trow (TipoException)(mensaje)**

```
public static void main(String[] args){
     try{
         metodo1(0);
     }catch (Exception e){
         System.out.println(e.toString());
     }
}
public static void metodo1(int numero){
    if (numero==0){
        throw new RuntimeException("Mal");
    }
}
```

Se pueden crear también en un try-catch, a gusto de consumidor.

Creación de Excepciones

Se puedn declarar excepciones para dar cobertura a todos los errores que se quieran gestionar. Las excepciones deben ser declaradas como subclases de Exception (checked) o RuntimeException (unchecked)

```
class ExcepcionNumeros extends RuntimeException{
    public ExcepcionNumeros(String mensaje){
        super(mensaje);
    }
}

public class Main {
    public static void main(String[] args){
         try{
             metodo1(0);
         }catch (ExcepcionNumeros e){
             System.out.println(e.toString());
         }
    }
    public static void metodo1(int numero){
        if (numero==0){
            throw new ExcepcionNumeros("El numero no puede ser 0");
        }
    }
}
```

Recomendaciones:

- Evitar catch vacíos (usando throws por ejemplo)
- Liberar recursos con finally o try con recursos
- Evitar lanzar excepciones genéricas (no sirve el Exception e)
- Procurar utilizar excepciones existentes (Reciclar)
- Cualquier excepción que deje la aplicación en un estado irrecuperable será unuchecked
- No abusar de las excepciones y evitar su uso en códigos bien diseñados