Las excepciones se agrupan en varias categorías según su tipo y causa:

### 1. Excepciones de Tipo Checked

Estas excepciones son verificadas en tiempo de compilación, lo que significa que el programador debe manejarlas con try-catch o lanzarlas con throws. Ejemplos comunes incluyen:

- **IOException**: Se lanza cuando ocurre un error de entrada/salida, como problemas al leer o escribir en archivos.
- **SQLException**: Ocurre cuando hay un error de acceso a una base de datos, como problemas de conexión o consultas inválidas.
- **ClassNotFoundException**: Se lanza cuando el programa intenta cargar una clase que no se encuentra en el classpath.

Estas excepciones son forzadas a ser manejadas para evitar que el programa falle inesperadamente.

### 2. Excepciones de Tipo Unchecked

También conocidas como excepciones de ejecución, estas no se verifican en tiempo de compilación y, por lo tanto, no es obligatorio manejarlas. Son comunes en errores lógicos o de programación, y se derivan de RuntimeException. Ejemplos:

- **NullPointerException**: Se lanza cuando el programa intenta acceder a un objeto que no ha sido inicializado.
- **IndexOutOfBoundsException**: Ocurre cuando se intenta acceder a un índice fuera del rango en listas o arreglos.
- **ArithmeticException**: Se lanza cuando ocurre un error en una operación matemática, como la división entre cero.
- **IllegalArgumentException**: Se lanza cuando un método recibe un argumento inapropiado.

Al ser errores de lógica de programación, estas excepciones se pueden prevenir con una validación adecuada de datos.

### 3. Errores (Errors)

Los errores son condiciones graves que generalmente están fuera del control del programa y no se espera que se manejen con try-catch. Algunos ejemplos son:

- **StackOverflowError**: Ocurre cuando la pila se llena, comúnmente debido a una recursión sin fin.

- **OutOfMemoryError**: Se lanza cuando la máquina virtual de Java se queda sin memoria disponible.
- **VirtualMachineError**: Representa errores en la propia máquina virtual de Java, como problemas graves de memoria o interrupciones en el procesamiento.

Estos errores son raros, pero cuando ocurren, el programa normalmente termina, y no es habitual tratar de manejarlos.

**4. Excepciones Personalizadas**

Los programadores pueden definir sus propias excepciones para manejar situaciones específicas en sus aplicaciones. Estas excepciones extienden de la clase Exception o RuntimeException y permiten describir errores de una manera más detallada y específica para la lógica de la aplicación.

**Breve resumen**

- **Checked Exceptions**: Verificadas en compilación; deben manejarse o declararse (IOException, SQLException).
- **Unchecked Exceptions**: No verificadas en compilación; pueden evitarse con validaciones (NullPointerException, ArithmeticException).
- **Errores (Errors)**: Problemas graves del sistema difíciles de manejar (OutOfMemoryError).
- **Excepciones Personalizadas**: Definidas por el programador para manejar errores específicos de la aplicación.