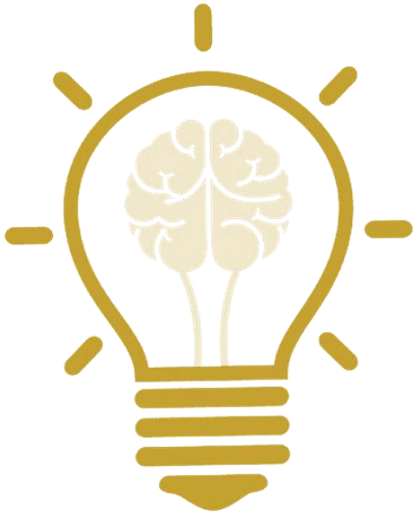
PROGRAMACIÓN EN JAVA

Materia PJA-001 material creado por el Ing. Jim Requena

Programación Orientada a Objetos en JAVA (Manejo de Excepciones en Java)



Materia PJA-001 material creado por el Ing. Jim Requena

Objetivos de la sesión

- Comprender qué son las excepciones y por qué son importantes
- Aprender a usar bloques try-catch para manejar excepciones
- Entender la jerarquía de excepciones en Java
- Practicar la creación y lanzamiento de excepciones personalizadas

¿Qué son las excepciones?

- Eventos que ocurren durante la ejecución del programa que interrumpen el flujo normal de instrucciones
- Representan condiciones de error o situaciones inesperadas
- Java utiliza objetos para representar excepciones

Tipos de excepciones

- Checked Exceptions
  - Deben ser declaradas en la firma del método o manejadas con try-catch
    Ejemplo: IOException, SQLException

- Unchecked Exceptions (Runtime Exceptions)
  - No es obligatorio declararlas o manejarlas
    Ejemplo: NullPointerException, ArrayIndexOutOfBoundsException

- Errores
  - Problemas graves, generalmente irrecuperables
    Ejemplo: OutOfMemoryError, StackOverflowError

Bloque try-catch

Sintaxis básica:

```
try {
        // Código que puede lanzar una excepción
}  catch (TipoDeExcepcion e) {
     // Código para manejar la excepción
}
```

***Ejemplo**:*

```
try {
        int resultado = 10 / 0;
}  catch (ArithmeticException e) {
       System.out.println("Error: División por cero");
}
```

Bloque try-catch-finally

- *El bloque finally se ejecuta siempre, haya o no excepción*
- *Útil para cerrar recursos o realizar limpieza*

```
try {
        // Código que puede lanzar una excepción
} catch (Exception e) {
// Manejo de la excepción
} finally {
// Código que se ejecuta siempre
}
```

Múltiples bloques catch

- ***Se pueden manejar diferentes tipos de excepciones***
- ***El orden es importante: de más específico a más general***

```
try {
        // Código que puede lanzar excepciones
} catch (ArithmeticException e) {
        System.out.println("Error aritmético: " + e.getMessage());
} catch (NullPointerException  e) {
        System.out.println("Error de referencia nula: " + e.getMessage());
} catch (Exception  e) {
        System.out.println("Error general: " + e.getMessage());

}
```

Lanzar excepciones

- ***Se usa la palabra clave throw***
- ***Se puede lanzar cualquier objeto que herede de Throwable***

```
public void verificarEdad(int edad) throws IllegalArgumentException {
        if (edad < 0) {
                throw new IllegalArgumentException("La edad no puede ser negativa");
        }
}
```

Lanzar excepciones

- ***Se usa la palabra clave throw***
  Es la acción de producir o "lanzar" el error cuando ocurre.

- ***Se puede lanzar cualquier objeto que herede de Throwable***
  Es la clase raíz para todos los objetos que pueden ser "lanzados" (thrown) en Java

```
public void verificarEdad(int edad) throws IllegalArgumentException {
        if (edad < 0) {
                throw new IllegalArgumentException("La edad no puede ser negativa");
        }
}
```

- ***En Java, "Throwable" representa todo lo que puede ser "lanzado" como parte del mecanismo de manejo de excepciones.***

Ejercicio Práctico

Crear un sistema simple de gestión de inventario:

1. Clase Producto con atributos nombre, precio y cantidad
2. Método para vender productos que lance una excepción si no hay suficiente stock
3. Excepción personalizada StockInsuficienteException
4. Programa principal que maneje las excepciones al vender productos

iTDS | Profesionales en Educación

Solución al ejercicio Práctico 1/3
Clase StockInsuficienteException

StockInsuficienteException.java

```java
public class StockInsuficienteException extends Exception {
    public StockInsuficienteException(String mensaje) {
            super(mensaje);
    }
}
```

Esta es una excepción personalizada que extiende de la clase Exception. Se utiliza para indicar que no hay suficiente stock de un producto. El constructor toma un mensaje que se pasa al constructor de la clase padre (Exception)

Solución al ejercicio Práctico Clase Producto 2/3

Producto.java

```
public class Producto {
      private String nombre;
      private double precio;
      private int cantidad;

public Producto(String nombre, double precio, int cantidad) {
     this.nombre = nombre;
     this.precio = precio;
     this.cantidad = cantidad;
     }

     public void vender(int cantidadVendida) throws StockInsuficienteException {
               if (cantidadVendida > cantidad) {
                   throw new StockInsuficienteException("Stock insuficiente para " + nombre);
           }
               cantidad -= cantidadVendida;
               System.out.println("Venta realizada: " + cantidadVendida + " " + nombre);
               }
     public String getNombre() {
               return nombre;
     }
}
```

Esta clase representa un producto con nombre, precio y cantidad en stock.

El método vender() es clave:
Verifica si hay suficiente stock para la venta.
Si no hay suficiente, lanza la excepción StockInsuficienteException.

Si hay suficiente, reduce la cantidad en stock e imprime un mensaje de venta exitosa.

Solución al ejercicio Práctico Clase Producto 3/3

MainProducto.java

```
public class MainProducto {
	public static void main(String[] args) {
		Producto p1 = new Producto("Laptop", 1000, 5);
		Producto p2 = new Producto("Teléfono", 500, 10);

		try {
			p1.vender(3);
			p2.vender(12);  // Esto lanzará una excepción
		} catch (StockInsuficienteException e) {
			System.out.println("Error: " + e.getMessage());
		}

		try { p1.vender(3); // Esto lanzará una excepción
		} catch (StockInsuficienteException e) {
			System.out.println("Error: " + e.getMessage());
		}
	}
}
```

Esta es la clase principal que demuestra el uso de las otras clases:

- Crea dos productos: una laptop (5 en stock) y un teléfono (10 en stock).
- Intenta vender 3 laptops (exitoso) y luego 12 teléfonos (lanzará una excepción).
- Luego intenta vender 3 laptops más (lanzará una excepción porque solo quedan 2).

El uso de bloques try-catch permite manejar las excepciones de manera controlada, mostrando mensajes de error apropiados cuando no hay suficiente stock.

Tarea

1. Extiende el sistema de inventario para incluir una clase Inventario que gestione múltiples productos

2. Implementa un método en Inventario para realizar una venta que pueda manejar múltiples productos

3. Crea una nueva excepción ProductoNoEncontradoException para cuando se intente vender un producto que no existe en el inventario

4. Utiliza un bloque try-with-resources para manejar la escritura de un registro de ventas en un archivo

5. Implementa un mecanismo de logging para registrar todas las excepciones que ocurran

iTDS | Profesionales en Educación

Recursos adicionales

- Documentación de Java sobre Excepciones: docs.oracle.com/javase/tutorial/essential/exceptions/
- Tutorial sobre Manejo de Excepciones en Java: www.javatpoint.com/exception-handling-in-java

Gracias!