EJERCICIOS DE APRENDIZAJE

En este módulo vamos a empezar a manejar los errores y las excepciones de nuestro código para poder seguir trabajando sin que el código se detenga.

VIDEOS: Te sugerimos ver los videos relacionados con este tema, antes de empezar los ejercicios, los podrás encontrar en tu aula virtual o en nuestro canal de YouTube.

1. Inicializar un objeto de la clase Persona ejercicio 3 de la guía 8 Servicios, a null y tratar de invocar el método esMayorDeEdad() a través de ese objeto. Luego, englobe el código con una cláusula try-catch para probar la nueva excepción que debe ser controlada.

2. Definir una Clase que contenga algún tipo de dato de tipo array y agregue el código para generar y capturar una excepción ArrayIndexOutOfBoundsException (índice de arreglo fuera de rango).

3. Defina una clase llamada DivisionNumero. En el método main utilice un Scanner para leer dos números en forma de cadena. A continuación, utilice el método parseInt() de la clase Integer, para convertir las cadenas al tipo int y guardarlas en dos variables de tipo int. Por ultimo realizar una división con los dos numeros y mostrar el resultado.

4. Todas estas operaciones puede tirar excepciones a manejar, el ingreso por teclado puede causar una excepción de tipo InputMismatchException, el método Integer.parseInt() si la cadena no puede convertirse a entero, arroja una NumberFormatException y además, al dividir un número por cero surge una ArithmeticException. Manipule todas las posibles excepciones utilizando bloques try/catch para las distintas excepciones

5. Escribir un programa en Java que juegue con el usuario a adivinar un número. La computadora debe generar un número aleatorio entre 1 y 500, y el usuario tiene que intentar adivinarlo. Para ello, cada vez que el usuario introduce un valor, la computadora debe decirle al usuario si el número que tiene que adivinar es mayor o menor que el que ha introducido el usuario. Cuando consiga adivinarlo, debe indicárselo e imprimir en pantalla el número de veces que el usuario ha intentado adivinar el número. Si el usuario introduce algo que no es un número, se debe controlar esa excepción e indicarlo por pantalla. En este último caso también se debe contar el carácter fallido como un intento.

6. Dado el método metodoA de la clase A, indique:

   a) ¿Qué sentencias y en qué orden se ejecutan si se produce la excepción MioException?

   b) ¿Qué sentencias y en qué orden se ejecutan si no se produce la excepción MioException?

```
class A {
public void metodoA() {
   sentencia_a1
   sentencia_a2
try {
   sentencia_a3
   sentencia_a4
} catch (MioException e){
   sentencia_a6
   }
   sentencia_a5
   }
}
```

7. Dado el método metodoB de la clase B, indique:

a) ¿Qué sentencias y en qué orden se ejecutan si se produce la excepción MioException?

b) ¿Qué sentencias y en qué orden se ejecutan si no se produce la excepción MioException?

```
class B {
public void metodoB() {
   sentencia_b1
try {
   sentencia_b2
} catch (MioException e){
   sentencia_b3
}
finally
   sentencia_b4
  }
}
```

8. Indique que se mostrará por pantalla cuando se ejecute cada una de estas clases:

```
class Uno{
   private static int metodo() {
         int valor=0;
         try {
               valor = valor+1;
               valor = valor + Integer.parseInt (”42”);
               valor = valor +1;
               System.out.println(”Valor final del try:” + valor) ;
         } catch (NumberFormatException e) {
               Valor = valor + Integer.parseInt(”42”);
               System.out.println(“Valor final del catch:” + valor);
         } finally {
               valor = valor + 1;
               System.out.println(”Valor final del finally: ” + valor) ;
         }
         valor = valor +1;
         System.out.println(”Valor antes del return: ” + valor) ;
         return valor;
   }
```

```
    public static void main (String[] args) {
        try {
            System.out.println (metodo()) ;
        }catch(Exception e) {
            System.err.println(”Excepcion en metodo() ”) ;
            e.printStackTrace();
        }
    }
}

class Dos{
    private static int metodo() {
        int valor=0;
        try{
            valor = valor + 1;
            valor = valor + Integer.parseInt (”W”);
            valor = valor + 1;
            System.out.println(”Valor final del try: ” + valor) ;
        } catch ( NumberFormatException e ) {
            valor = valor + Integer.parseInt (”42”);
            System.out.println(”Valor final del catch: ” + valor) ;
        } finally {
            valor = valor + 1;
            System.out.println(”Valor final del finally: ” + valor) ;
        }
        valor = valor + 1;
        System.out.println(”Valor antes del return: ” + valor) ;
        return valor;
    }

    public static void main (String[] args) {
        try{
            System.out.println ( metodo ( ) ) ;
        } catch(Exception e) {
            System.err.println ( ” Excepcion en metodo ( ) ” ) ;
            e.printStackTrace();
        }
    }
}
```

```
class Tres{
	private static int metodo( ) {
		int valor=0;
		try{
			valor = valor + 1;
			valor = valor + Integer.parseInt (”W”);
			valor = valor + 1;
			System.out.println(”Valor final del try: ” + valor);
		} catch(NumberFormatException e) {
			valor = valor + Integer.parseInt (”W”);
			System.out.println(”Valor final del catch: ” + valor);
		} finally{
			valor = valor + 1;
			System.out.println(”Valor final del finally:” + valor);
		}
		valor = valor + 1;
		System.out.println(”Valor antes del return: ” + valor) ;
		return valor;
		}

	public static void main (String[] args) {
		try{
			System.out.println( metodo ( ) ) ;
		} catch(Exception e) {
			System.err.println(”Excepcion en metodo ( ) ” ) ;
			e.printStackTrace();
		}
	}
}
```

9. Dado el método metodoC de la clase C, indique:

a) ¿Qué sentencias y en qué orden se ejecutan si se produce la excepción MioException?

b) ¿Qué sentencias y en qué orden se ejecutan si no se produce la excepción MioException?

c) ¿Qué sentencias y en qué orden se ejecutan si se produce la excepción TuException?

```
class C {
   void metodoC() throws TuException{
		sentencia_c1
		try {
			sentencia_c2
			sentencia_c3
		} catch (MioException e){
```

```
                sentencia_c4
            } catch (TuException e){
                sentencia_c5
                throw (e)
            }
            finally
                sentencia_c6
    }
}
```

IMPORTANTE: A partir de la próxima guía se debe aplicar en todos los ejercicios el manejo de excepciones cada vez que sea necesario controlar una posible excepción.