

Nombre: Jose Juan Laynez Zapeta

Carnet: 202308221

Examen Parcial No. 3. Segunda Serie

Identificación de errores a primera vista:

```
public class ErrorfulCode {  
    public static void main(String[] args) {  
        // Compile-time error: Missing semicolon  
        int number = 10;  
        // Runtime error: Division by zero  
        int result = number / 0;  
        // Compile-time error: Undefined variable  
        String message = greeting;  
        // Runtime error: ArrayIndexOutOfBoundsException  
        int[] numbers = {1, 2, 3};  
        System.out.println(numbers[3]);  
        // Compile-time error: Method not found  
        myMethod();  
        // Compile-time error: Incorrect type for variable assignment  
        String value = 123;  
        // Logical error: Incorrect condition for loop  
        for (int i = 0; i <= 5; i++) {  
            if (i == 5) {  
                System.out.println("Reached five!");  
            }  
        }  
        // Compile-time error: Missing return type  
        public static void myMethod(String param) {  
            System.out.println("This is a method.");  
        }  
    }  
}
```

Al ver el código, se pudo ver algunos errores:

- ➔ En la línea 5, le falta el punto y coma. Se soluciona agregándolo al final.
- ➔ En la línea 8, se divide entre cero "number/0". Se puede solucionar cambiando el cero por cualquier numero.
- ➔ En la línea 11, no existe la variable "greeting". Se soluciona creando la variable "greeting" o directamente agregando el valor.
- ➔ En la línea 14 y 15, se crea una variable de longitud 3, sin embargo, al ser java un lenguaje no inclusivo, así que dará un error de tipo "Out of bounds" . Se soluciona cambiando el 3 por un 2, si se desea acceder al último elemento.
- ➔ En la línea 18, myMethod() no podrá ser ejecutado por que se pide un parámetro. Se puede ya sea quitar el parámetro, por que no se usa, o agregarlo a la hora de emplearlo.

- ➔ En la línea 21, se crea un String, pero se ingresa un integer, ya que el 123 ingresado no lleva comillas. Se puede solucionar agregando comillas alrededor de 123, o convirtiendo la variable en integer.
- ➔ El ciclo for estaba bien.

En total, se encontraron 7 errores.

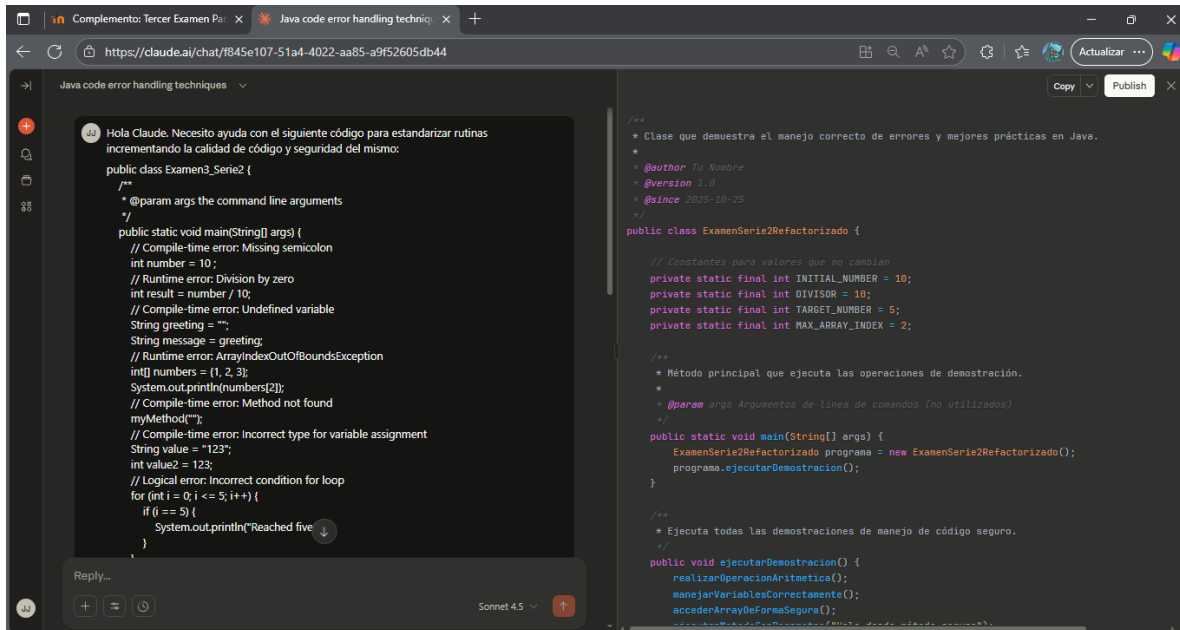
Código arreglado:

```
public static void main(String[] args) {  
    // Compile-time error: Missing semicolon  
    int number = 10 ;  
    // Runtime error: Division by zero  
    int result = number / 10;  
    // Compile-time error: Undefined variable  
    String greeting = "";  
    String message = greeting;  
    // Runtime error: ArrayIndexOutOfBoundsException  
    int[] numbers = {1, 2, 3};  
    System.out.println(numbers[2]);  
    // Compile-time error: Method not found  
    myMethod("");  
    // Compile-time error: Incorrect type for variable assignment  
    String value = "123";  
    int value2 = 123;  
    // Logical error: Incorrect condition for loop  
    for (int i = 0; i <= 5; i++) {  
        if (i == 5) {  
            System.out.println("Reached five!");  
        }  
    }  
}  
  
// Compile-time error: Missing return type  
public static void myMethod(String param) {  
    System.out.println("This is a method.");  
}
```

Output al compilarlo:

```
Output - examen3_Serie2 (run)  
run:  
3  
This is a method.  
Reached five!  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Para la siguiente sección de la actividad hice uso de Claude.ai. Usé esta herramienta ya que la he usado anteriormente para resolución de dudas en cuanto a mis proyectos anteriores y para solicitud de revisión de código y me ha resultado muy efectiva.



```
Hola Claude. Necesito ayuda con el siguiente código para estandarizar rutinas incrementando la calidad de código y seguridad del mismo:
```

```
public class Examen3_Serie2 {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // Compile-time error: Missing semicolon  
        int number = 10;  
        // Runtime error: Division by zero  
        int result = number / 10;  
        // Compile-time error: Undefined variable  
        String greeting = "";  
        String message = greeting;  
        // Runtime error: ArrayIndexOutOfBoundsException  
        int[] numbers = {1, 2, 3};  
        System.out.println(numbers[2]);  
        // Compile-time error: Method not found  
        myMethod("");  
        // Compile-time error: Incorrect type for variable assignment  
        String value = "123";  
        int value2 = 123;  
        // Logical error: Incorrect condition for loop  
        for (int i = 0; i <= 5; i++) {  
            if (i == 5) {  
                System.out.println("Reached five");  
            }  
        }  
    }  
}
```

```
/**  
 * Clase que demuestra el manejo correcto de errores y mejores prácticas en Java.  
 *  
 * @author Tu Nombre  
 * @version 1.0  
 * @since 2025-10-25  
 */  
public class ExamenSerie2Refactorizado {  
    /**  
     * Constantes para valores que no cambian  
     */  
    private static final int INITIAL_NUMBER = 10;  
    private static final int DIVISOR = 10;  
    private static final int TARGET_NUMBER = 5;  
    private static final int MAX_ARRAY_INDEX = 2;  
    /**  
     * Método principal que ejecuta las operaciones de demostración.  
     */  
    /**  
     * @param args Argumentos de línea de comandos (no utilizados)  
     */  
    public static void main(String[] args) {  
        ExamenSerie2Refactorizado programa = new ExamenSerie2Refactorizado();  
        programa.ejecutarDemostracion();  
    }  
    /**  
     * Ejecuta todas las demostraciones de manejo de código seguro.  
     */  
    public void ejecutarDemostracion() {  
        realizarOperacionArismetica();  
        manejarVariablesCorrectamente();  
        accederArrayDeFormaSegura();  
    }  
}
```

Entre las cosas más destacables de la solución proporcionada por Claude.ai, se puede ver que se modularizó el proyecto en gran manera, separando todo dentro de funciones diferentes. Además, se puede destacar el uso de “try-catch” para asegurar que no se lance el error directamente, en caso de haber uno.

De igual forma, se pudo ver como se dejó documentación y comentarios acerca del código creado, gracias al cual, es mas fácil de comprender el funcionamiento del código y su interpretación.

A continuación, y para finalizar, se dejará el código propuesto por esta IA.

```
* Clase que demuestra el manejo correcto de errores y mejores prácticas en Java.
*
* @author Tu Nombre
* @version 1.0
* @since 2025-10-25
*/
public class ExamenSerie2Refactorizado {

    // Constantes para valores que no cambian
    private static final int INITIAL_NUMBER = 10;
    private static final int DIVISOR = 10;
    private static final int TARGET_NUMBER = 5;
    private static final int MAX_ARRAY_INDEX = 2;

    /**
     * Método principal que ejecuta las operaciones de demostración.
     *
     * @param args Argumentos de línea de comandos (no utilizados)
     */
    public static void main(String[] args) {
        ExamenSerie2Refactorizado programa = new ExamenSerie2Refactorizado();
        programa.ejecutarDemostracion();
    }

    /**
     * Ejecuta todas las demostraciones de manejo de código seguro.
     */
    public void ejecutarDemostracion() {
        realizarOperacionAritmetica();
        manejarVariablesCorrectamente();
        accederArrayDeFormaSegura();
    }
}
```

```
} else {
    System.out.println("Mensaje: valor por defecto");
}
}

/**
 * Accede a un array de forma segura validando índices.
 */
private void accederArrayDeFormaSegura() {
    int[] numbers = {1, 2, 3};

    try {
        // Validación del índice antes de acceder
        if (MAX_ARRAY_INDEX >= 0 && MAX_ARRAY_INDEX < numbers.length) {
            System.out.println("Elemento en índice " + MAX_ARRAY_INDEX + ": " + numbers[MAX_ARRAY_INDEX]);
        } else {
            System.err.println("Índice fuera de rango");
        }
    } catch (ArrayIndexOutOfBoundsException e) {
        System.err.println("Error al acceder al array: " + e.getMessage());
    }
}

/**
 * Ejecuta un método con parámetro validado.
 *
 * @param param Parámetro de entrada que no debe ser nulo
 */
private void ejecutarMetodoConParametro(String param) {
    if (param == null) {
        System.err.println("El parámetro no puede ser nulo");
        return;
    }
}
```

```
ejecutarMetodoConParametro("Hola desde método seguro");
manejarConversionDeTipos();
ejecutarBucleConLogicaCorrecta();
}

/**
 * Realiza una operación aritmética con validación del divisor.
 */
private void realizarOperacionArithmetica() {
    int number = INITIAL_NUMBER;

    try {
        if (DIVISOR == 0) {
            throw new ArithmeticException("No se puede dividir por cero");
        }
        int result = number / DIVISOR;
        System.out.println("Resultado de la división: " + result);
    } catch (ArithmeticException e) {
        System.err.println("Error aritmético: " + e.getMessage());
    }
}

/**
 * Demuestra el manejo correcto de variables y valores nulos.
 */
private void manejarVariablesCorrectamente() {
    String greeting = "Hola Mundo";

    // Validación de null antes de usar la variable
    if (greeting != null && !greeting.isEmpty()) {
        String message = greeting;
        System.out.println("Mensaje: " + message);
    } else {
}
```

```
return;
}
}

myMethod(param);
}

/**
 * Método auxiliar que imprime un mensaje.
 *
 * @param param Mensaje a imprimir
 */
public static void myMethod(String param) {
    if (param == null || param.isEmpty()) {
        System.out.println("Este es un método con parámetro vacío.");
    } else {
        System.out.println("Este es un método. Parámetro: " + param);
    }
}

/**
 * Demuestra la conversión segura de tipos.
 */
private void manejarConversionDeTipos() {
    String value = "123";

    try {
        int value2 = Integer.parseInt(value);
        System.out.println("Conversión exitosa: " + value2);

        // Si necesitamos usar ambos tipos
        String valueAsString = value;
        int valueAsInt = value2;
        System.out.println("String: " + valueAsString + ", Int: " + valueAsInt);
    } catch (NumberFormatException e) {
}
```

```

    try {
        int value2 = Integer.parseInt(value);
        System.out.println("Conversión exitosa: " + value2);

        // Si necesitamos usar ambos tipos
        String valueAsString = value;
        int valueAsInt = value2;
        System.out.println("String: " + valueAsString + ", Int: " + valueAsInt);
    } catch (NumberFormatException e) {
        System.err.println("Error al convertir string a entero: " + e.getMessage());
    }
}

/**
 * Ejecuta un bucle con lógica correcta y clara.
 */
private void ejecutarBucleConLogicaCorrecta() {
    // Bucle con condición clara y lógica correcta
    for (int i = 0; i <= TARGET_NUMBER; i++) {
        if (i == TARGET_NUMBER) {
            System.out.println("¡Alcancamos el número " + TARGET_NUMBER + "!");
        }
    }

    // Alternativa más clara si solo queremos ejecutar algo al final
    System.out.println("Bucle completado hasta: " + TARGET_NUMBER);
}
}

```