

Java version 11

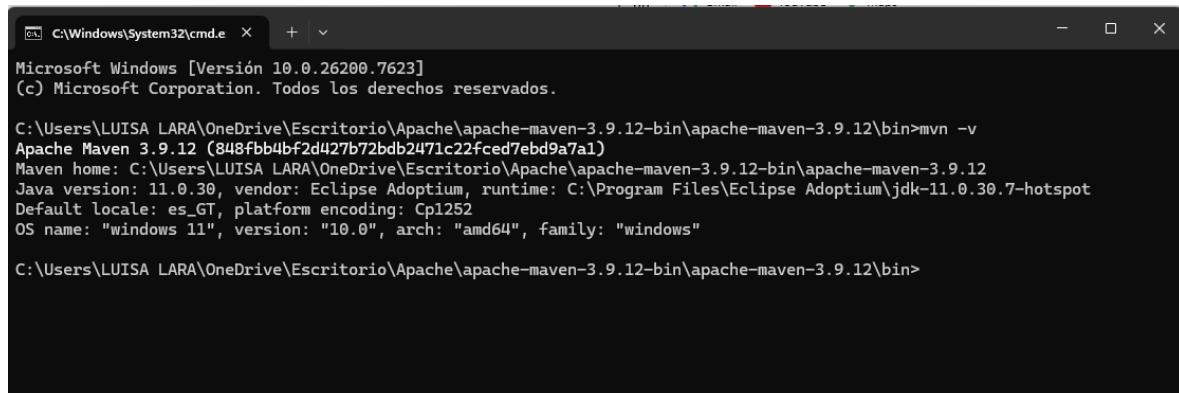


```
Símbolo del sistema + ×
Microsoft Windows [Versión 10.0.26200.7623]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\LUISA LARA>java -version
openjdk version "11.0.30" 2026-01-20
OpenJDK Runtime Environment Temurin-11.0.30+7 (build 11.0.30+7)
OpenJDK 64-Bit Server VM Temurin-11.0.30+7 (build 11.0.30+7, mixed mode)

C:\Users\LUISA LARA>
```

Apache Maven 3.9.12



```
C:\Windows\System32\cmd.e + ×
Microsoft Windows [Versión 10.0.26200.7623]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\LUISA LARA\OneDrive\Escritorio\Apache\apache-maven-3.9.12-bin\apache-maven-3.9.12\bin>mvn -v
Apache Maven 3.9.12 (848fbb4bf2d427b72bdb2471c22fcfd7ebd9a7a1)
Maven home: C:\Users\LUISA LARA\OneDrive\Escritorio\Apache\apache-maven-3.9.12-bin\apache-maven-3.9.12
Java version: 11.0.30, vendor: Eclipse Adoptium, runtime: C:\Program Files\Eclipse Adoptium\jdk-11.0.30.7-hotspot
Default locale: es_GT, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

C:\Users\LUISA LARA\OneDrive\Escritorio\Apache\apache-maven-3.9.12-bin\apache-maven-3.9.12\bin>
```

EJECUCION DE LOS PROGRAMAS EN CONSOLA

The screenshot shows the Eclipse IDE interface. The left side features the Package Explorer with a project named 'App'. The central area displays the code for 'App.java':

```
1 package com.tarea_uno.App;
2
3 /**
4  * Hello world!
5  */
6
7 public class App
8 {
9     public static void main( String[] args )
10    {
11        System.out.println( "Hola mundo desde java!" );
12    }
13 }
```

The right side shows the Outline view with the main method highlighted. Below the editor, the Console tab is active, showing the output of the program:

```
Hola mundo desde java!
```

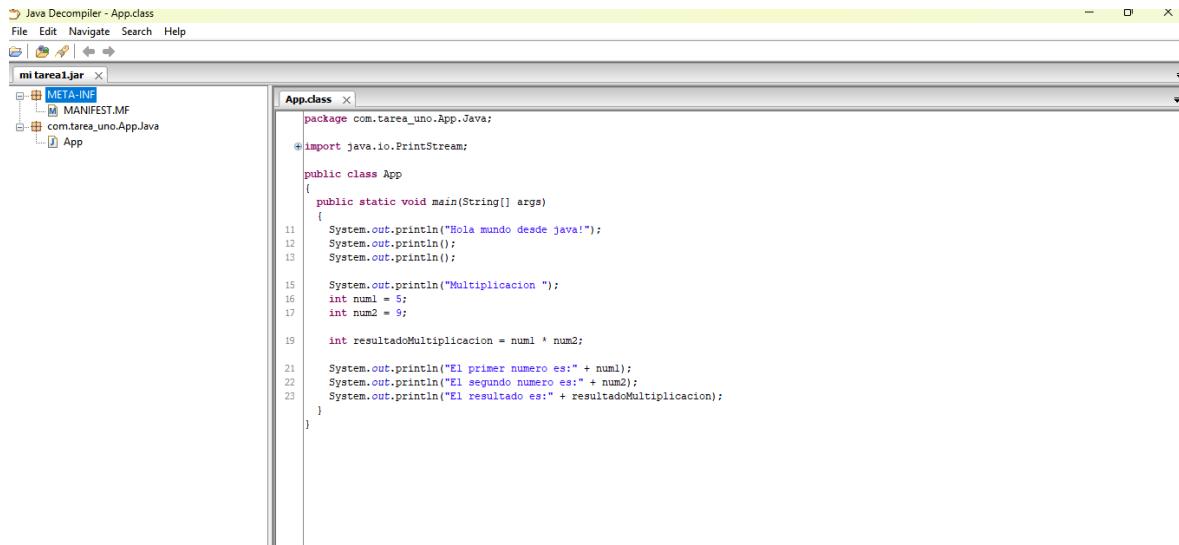
The screenshot shows the Eclipse IDE interface. The left side features the Package Explorer with a project named 'App'. The central area displays the code for 'App.java':

```
1 package com.tarea_uno.App;
2
3 /**
4  * Hello world!
5  */
6
7 public class App
8 {
9     public static void main( String[] args )
10    {
11        System.out.println( "Hola mundo desde java!" );
12        System.out.println();
13        System.out.println();
14
15        System.out.println( "Multiplicacion" );
16        int num1 = 5;
17        int num2 = 9;
18
19        int resultadoMultiplicacion = num1 * num2;
20
21        System.out.println("El primer numero es:" + num1);
22        System.out.println("El segundo numero es:" + num2);
23        System.out.println("El resultado es:" + resultadoMultiplicacion);
24
25
26
27    }
28 }
```

The right side shows the Outline view with the main method highlighted. Below the editor, the Console tab is active, showing the output of the program:

```
Multiplicacion
El primer numero es:5
El segundo numero es:9
El resultado es:45
```

INGENIERIA INVERSA (ANTES DE OFUSCAR)



The screenshot shows the Java Decomplier interface with the file "mi tarea1.jar" open. The left pane displays the file structure: META-INF/MANIFEST.MF and com.tarea_1.App.java. The right pane shows the decompiled code for App.java:

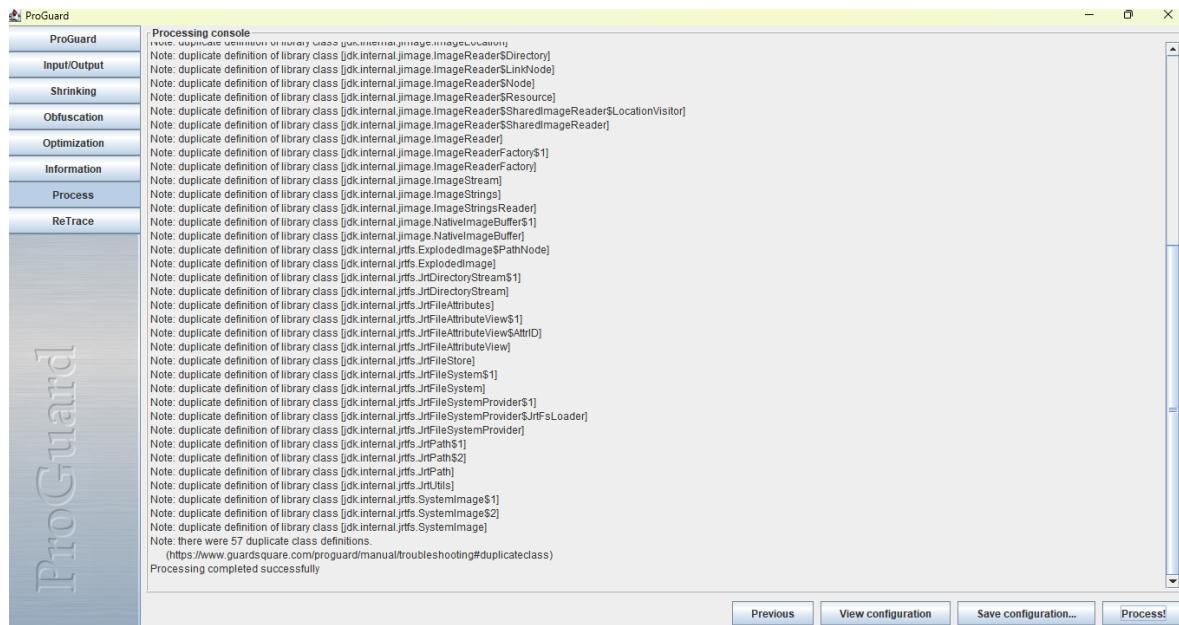
```
package com.tarea_1.App;
import java.io.PrintStream;
public class App
{
    public static void main(String[] args)
    {
        System.out.println("Hola mundo desde java!");
        System.out.println();
        System.out.println();

        System.out.println("Multiplicacion ");
        int num1 = 5;
        int num2 = 9;

        int resultadoMultiplicacion = num1 * num2;
        System.out.println("El primer numero es:" + num1);
        System.out.println("El segundo numero es:" + num2);
        System.out.println("El resultado es:" + resultadoMultiplicacion);
    }
}
```

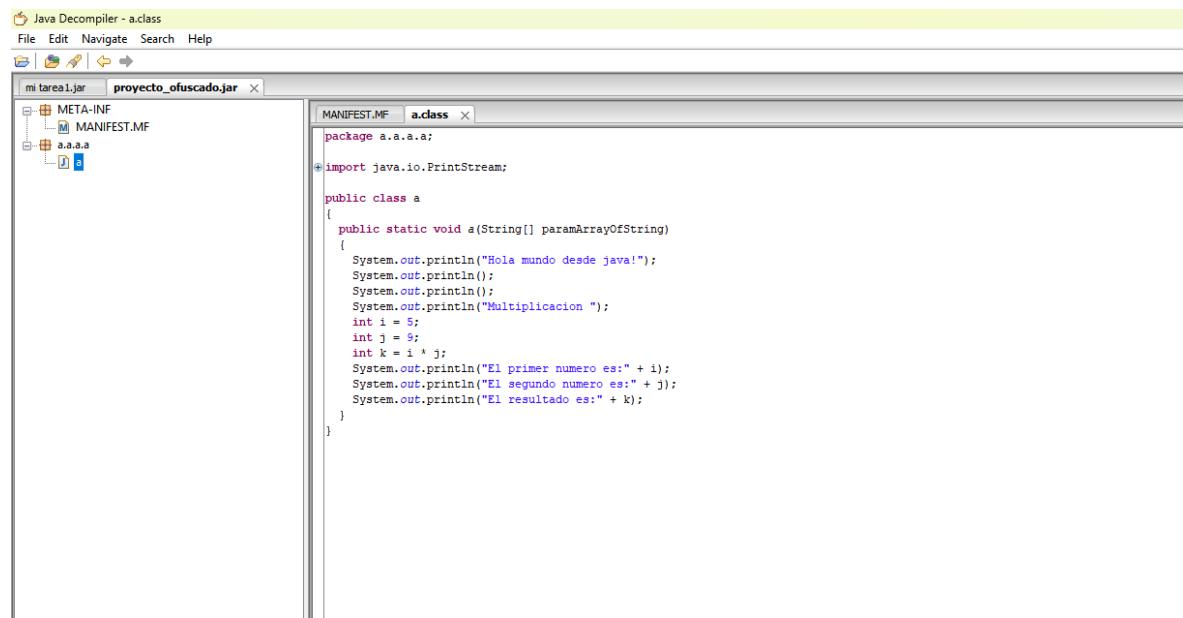
PROCESOS DE OFUSCACION

Se utilizó ProGuard



The screenshot shows the ProGuard Processing console window. The left sidebar lists various processing steps: ProGuard, Input/Output, Shrinking, Obfuscation, Optimization, Information, Process, and ReTrace. The main pane displays the processing log, which includes numerous "Note: duplicate definition of library class" messages for various internal Java classes, such as `[dk/internal/image/ImageReader$Directory]`, `[dk/internal/image/ImageReader$LinkNode]`, and `[dk/internal/image/ImageReader$Node]`. The log concludes with the message "Processing completed successfully".

INGENIERIA INVERSA (DESPUES DE OFUSCAR)



The screenshot shows the Java Decompile interface. The left pane displays the file structure of a JAR file, showing 'mi tarea1.jar' and 'proyecto_ofuscado.jar'. The right pane shows the decompiled code for a class named 'a'. The code prints "Hola mundo desde java!", initializes variables i=5, j=9, and k=i*j, and then prints the values of i, j, and k.

```
Java Decomplier - a.class
File Edit Navigate Search Help
mi tarea1.jar proyecto_ofuscado.jar
META-INF MANIFEST.MF a.class
MANIFEST.MF a.class
package a.a.a.a;
import java.io.PrintStream;
public class a
{
    public static void a(String[] paramArrayOfString)
    {
        System.out.println("Hola mundo desde java!");
        System.out.println();
        System.out.println();
        System.out.println("Multiplicacion ");
        int i = 5;
        int j = 9;
        int k = i * j;
        System.out.println("El primer numero es:" + i);
        System.out.println("El segundo numero es:" + j);
        System.out.println("El resultado es:" + k);
    }
}
```

CONCLUSION

Después de aplicar la ofuscación al descompilar el nuevo archivo jar se observa que los nombres de las clases y variables han cambiado, la clase ahora se llama “a” y adentro de la clase lo que anterior era num1 y num2, la multiplicación ahora se observa como “i” “j” “k”. Esto dificulta la comprensión del código y el nivel de protección de ingeniería inversa.