

## Unidad 1

# ***Introducción al acceso a datos y manejo de ficheros***

## Tarea AD01

### **Manejando información almacenada en ficheros**

1.	Enunciado de la tarea.....	2
2.	Crear una clase Usuario.....	4
3.	Implementación del menú .....	7
4.	Agregar un usuario a la lista de usuarios.....	8
5.	Borrar un usuario de la lista de usuarios.....	11
6.	Guardar la lista de usuarios en un archivo (serialización).....	13
7.	Cargar la lista de usuarios desde el archivo (deserialización) .....	15
8.	Mostrar los usuarios en consola .....	19
9.	Exporta a fichero .txt la lista de usuarios.....	21
10.	Salir de la aplicación .....	23
11.	¿Existe el archivo ‘user.dat’? .....	24
12.	Advertencia sobre datos no guardados antes de salir .....	25
13.	¿Recuperar datos de disco? .....	26
14.	Resultados del programa en consola .....	27
14.1.	Inicio del programa.....	27
14.2.	Agregar un usuario .....	29
14.3.	Mostrar los usuarios en consola.....	30
14.4.	Borrar usuario .....	31
14.5.	Guardar lista (Serialización) .....	33
14.6.	Cargar lista (deserialización) .....	34
14.7.	Exportación archivo TXT .....	35
14.8.	¿Cargar datos guardados? .....	36
14.9.	Salida del programa .....	38

## 1. Enunciado de la tarea

En esta tarea debes realizar una aplicación en Java que se utilizará para gestionar los usuarios de un comercio online. El proyecto Java creado deberá llamarse **Nombre\_Apellido1\_AD1\_E1** y deberá tener en cuenta los siguientes requisitos:

- Debes crear una clase **Usuario** que contenga los siguientes atributos:

- identificador (de tipo String)
- contraseña (de tipo String)
- dirección (de tipo String)
- año de nacimiento (de tipo int)

- El objetivo principal es serializar una lista de objetos de la clase **Usuario**, de modo que se pueda guardar en un archivo, y que de igual modo se pueda deserializar esa lista desde el archivo.

- La aplicación debe permitir lo siguiente (implementando un menú para ello):

1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario de la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.
0. Salir de la aplicación

Consideraciones:

- No olvides usar la interfaz **Serializable**.
- Asegúrate de manejar las excepciones correctamente.
- El archivo debe ser un archivo binario (**user.dat**), salvo el **user.txt** al que se exporte con la opción al efecto.
- Para borrar un usuario se deberá facilitar su identificador.
- Al ejecutarse la aplicación, se comprobará si existe el fichero **user.dat**. Si no existe, se avisará al usuario de que la aplicación no tiene datos grabados. Si existe, se cargarán los datos guardados de modo que los usuarios que hubiera en el fichero se cargarán en una lista.
- En caso de que, al ejecutar la aplicación, se modifique algún dato, y el usuario intente salir de la aplicación sin haber guardado los datos, se le avisará indicando que, si se sale sin guardar los datos, perderá los cambios realizados. Podría ser con un mensaje como: **"Ha habido cambios en el programa que todavía no se han guardado. Si desea guardarlos ejecute la opción correspondiente del menú. Si sale ahora no se guardarán. ¿Está seguro de que desea salir sin guardar?"**.

- Si se ejecuta la opción de recuperar datos, habiendo realizado cambios en el programa, se advertirá al usuario que los cambios se perderán, puesto que se cargarán los datos del fichero en la lista, y se le pedirá confirmación antes de continuar: "*Ha realizado cambios que no ha guardado en disco. Si continúa la carga del archivo se restaurarán los datos de disco y se perderán los cambios no guardados. ¿Desea continuar con la carga y restaurar los datos del archivo? (S/N).*"
- Se deben mostrar todos los mensajes informativos que se consideren oportunos.

## 2. Crear una clase Usuario

The screenshot shows an IDE interface with the following details:

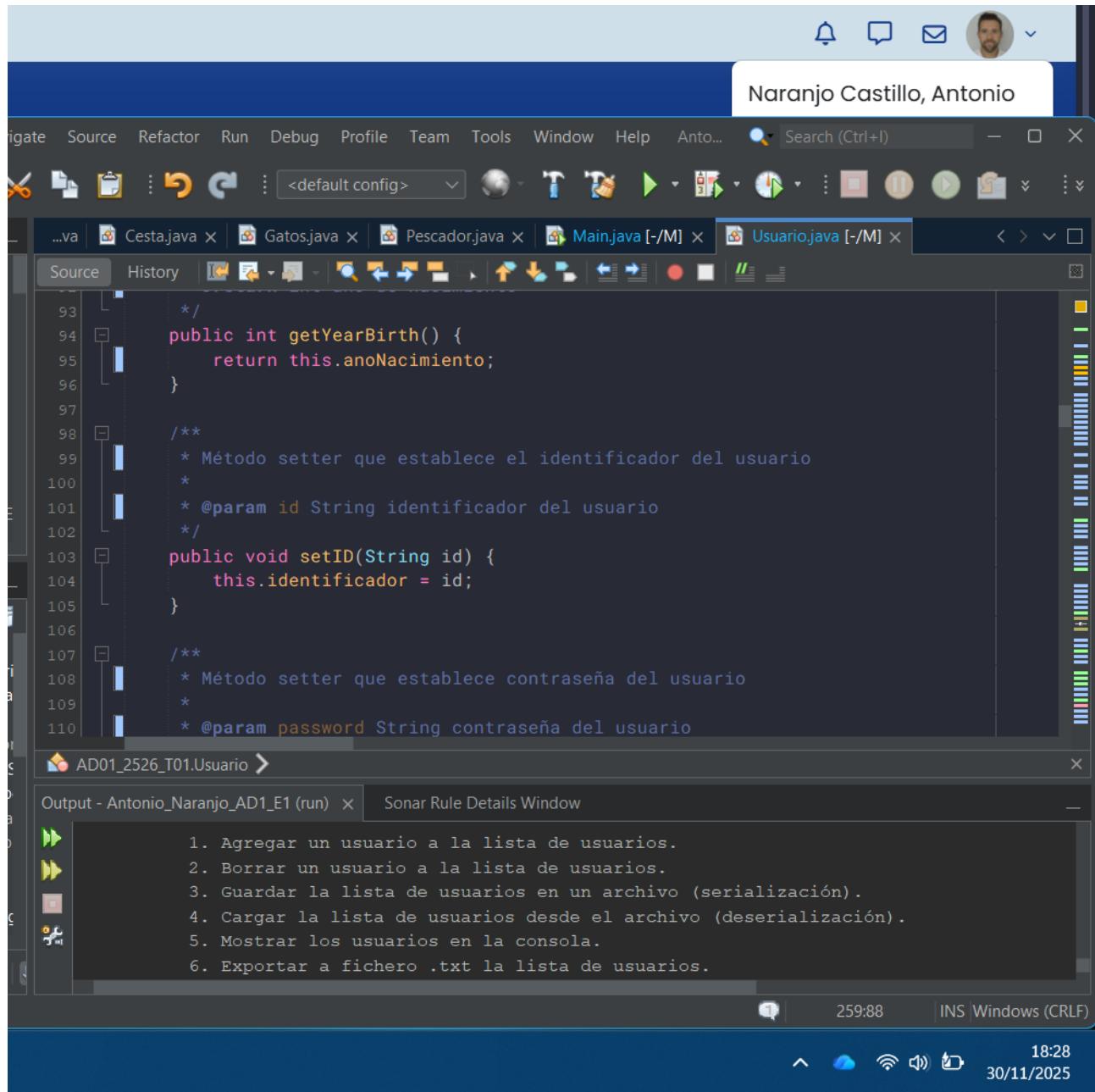
- Title Bar:** Shows the user's name "Naranjo Castillo, Antonio".
- Toolbar:** Includes standard icons for file operations like剪切 (Cut), 复制 (Copy), and 粘贴 (Paste).
- MenuBar:** Contains options like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... A search bar is also present.
- Project Explorer:** Shows files like Cesta.java, Gatos.java, Pescador.java, Main.java, and Usuario.java.
- Code Editor:** Displays the code for the Usuario.java class. The constructor is defined as follows:

```
46
47     /**
48      * Método constructor de la clase Usuario
49      *
50      * @param id String que representa la identificación de los usuarios
51      * @param password String que representa la contraseña de los usuarios
52      * @param address String que almacena la dirección de los usuarios
53      * @param yearBirth Dato numérico de tipo entero (int) que almacena el año
54      */
55     public Usuario(String id, String password, String address, int yearBirth)
56         this.identificador = id;
57         this.contrasena = password;
58         this.direccion = address;
59         this.anoNacimiento = yearBirth;
60     }
61
62     /**

```

- Output Window:** Shows a list of tasks:
1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.
- System Tray:** Shows icons for battery, signal, and date/time (30/11/2025, 18:24).

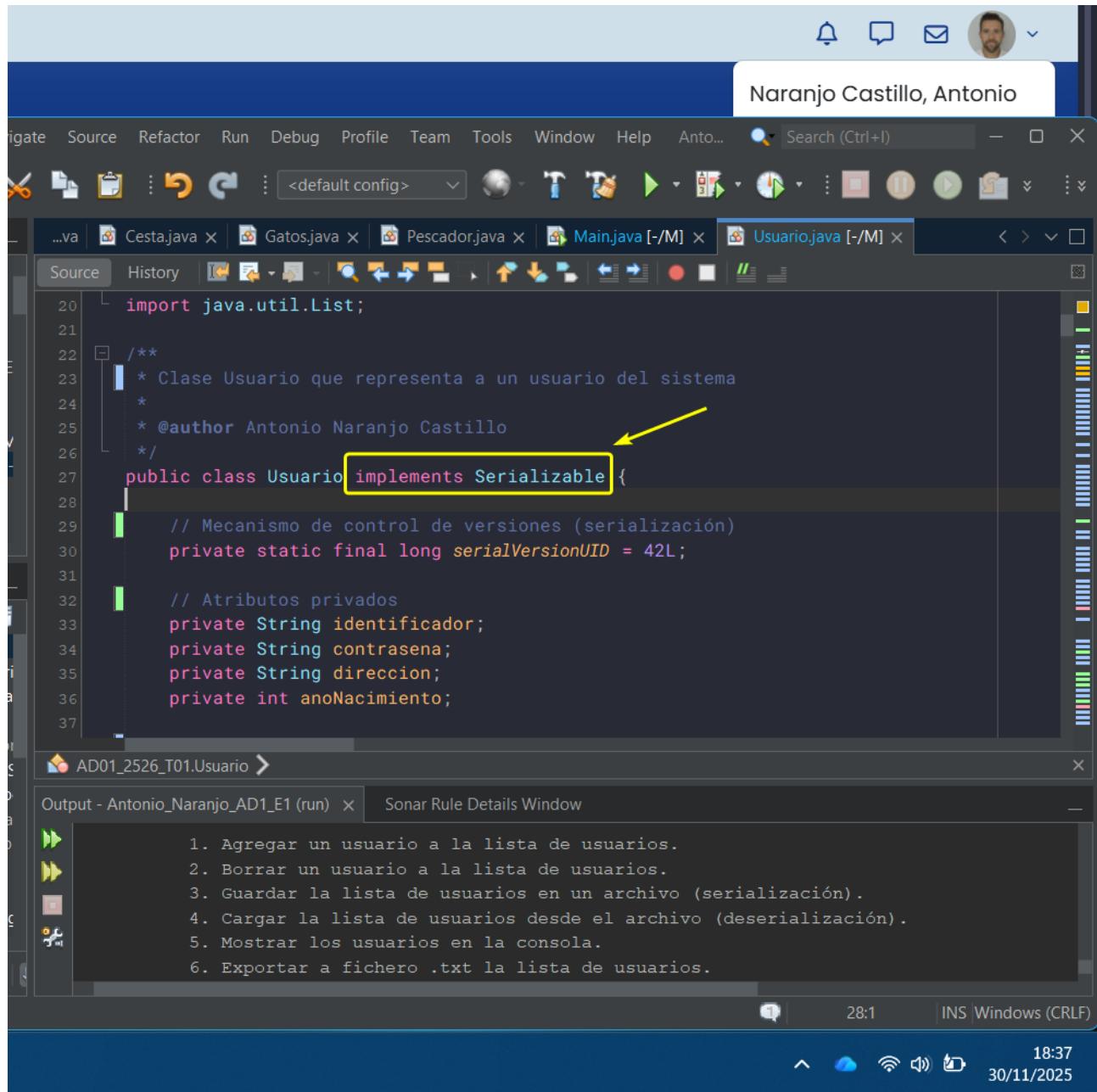
Se crea una clase Usuario implementando el método constructor que define los cuatro atributos solicitados en la tarea, identificador (de tipo String), contraseña (de tipo String), dirección (de tipo String) y año de nacimiento (de tipo int).



The screenshot shows an IDE interface with the following details:

- Toolbar:** Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, Auto... (with a search bar), and various icons for file operations.
- Project Bar:** Shows files like Cestajava.java, Gatos.java, Pescador.java, Main.java, and Usuario.java.
- Code Editor:** Displays the source code for the Usuario.java class. The code includes methods for getting and setting the birth year, ID, and password, along with Javadoc comments.
- Output Window:** Shows a list of tasks or steps:
  1. Agregar un usuario a la lista de usuarios.
  2. Borrar un usuario a la lista de usuarios.
  3. Guardar la lista de usuarios en un archivo (serialización).
  4. Cargar la lista de usuarios desde el archivo (deserialización).
  5. Mostrar los usuarios en la consola.
  6. Exportar a fichero .txt la lista de usuarios.
- System Tray:** Shows icons for battery, signal, and date/time (30/11/2025, 18:28).

Se implementan los métodos getters y setters para obtener o establecer los atributos anteriores para cada objeto usuario creado mediante la clase Usuario.



```
import java.util.List;

/**
 * Clase Usuario que representa a un usuario del sistema
 *
 * @author Antonio Naranjo Castillo
 */
public class Usuario implements Serializable {  
    // Mecanismo de control de versiones (serialización)  
    private static final long serialVersionUID = 42L;  
  
    // Atributos privados  
    private String identificador;  
    private String contrasena;  
    private String direccion;  
    private int anoNacimiento;
```

Output - Antonio\_Naranjo\_AD1\_E1 (run) x Sonar Rule Details Window

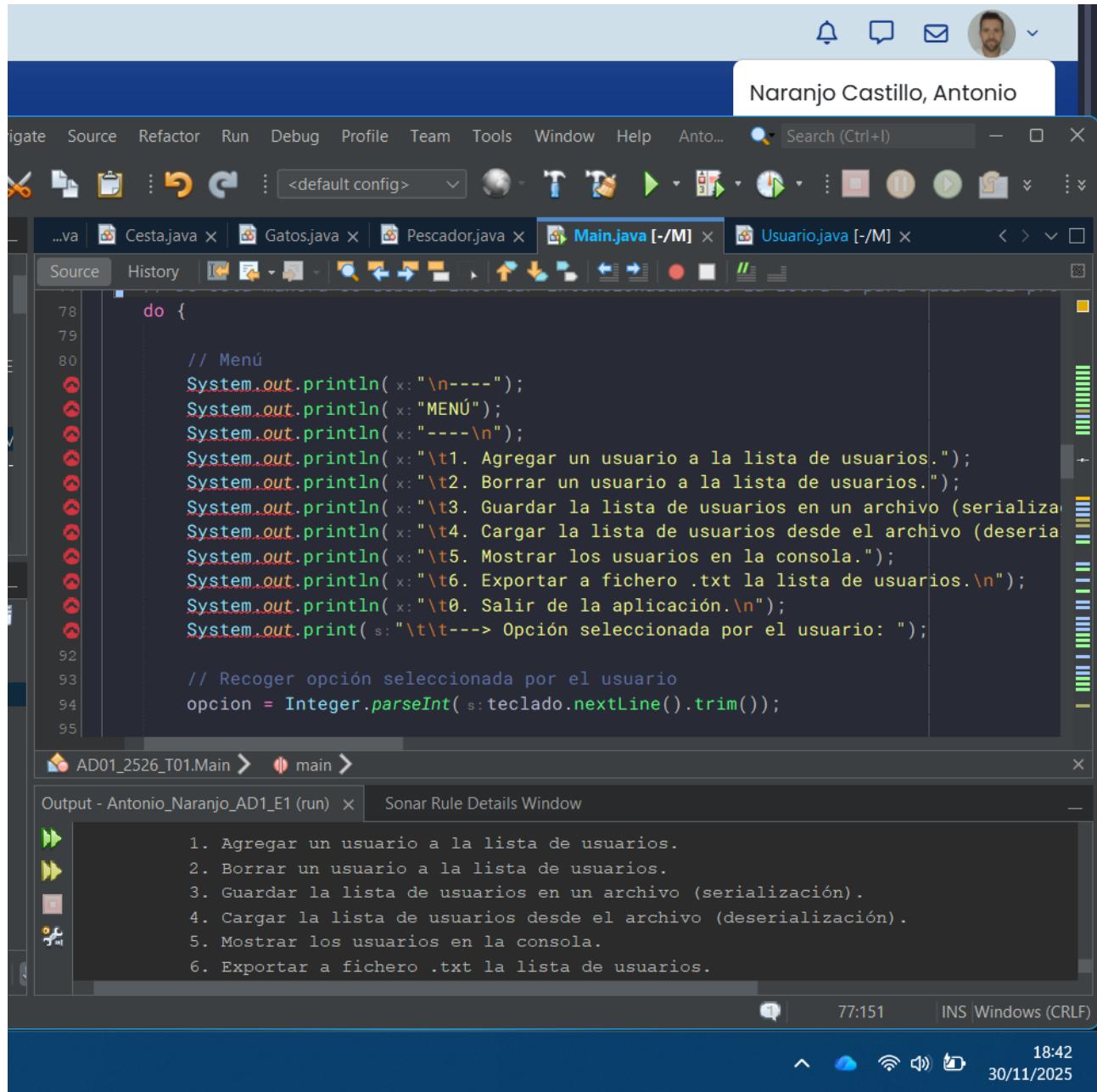
1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.

28:1 INS Windows (CRLF)

18:37 30/11/2025

Importante, para poder serializar una lista de objetos de la clase Usuario, de modo que se pueda guardar en un archivo, y que de igual modo se pueda deserializar esa lista desde el archivo, la clase Usuario debe implementarse desde la interfaz Serializable.

### 3. Implementación del menú



The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name "Naranjo Castillo, Antonio".
- Menu Bar:** Includes options like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... along with a search bar.
- Toolbar:** Contains various icons for file operations, navigation, and tools.
- Project Explorer:** Shows files like Cestajava.java, Gatos.java, Pescador.java, Main.java [-M], and Usuario.java [-M].
- Code Editor:** Displays Java code for a menu system. The code includes a do-while loop that prints a menu and handles user input. It lists six options: 1. Agregar un usuario a la lista de usuarios., 2. Borrar un usuario a la lista de usuarios., 3. Guardar la lista de usuarios en un archivo (serialización), 4. Cargar la lista de usuarios desde el archivo (deserialización), 5. Mostrar los usuarios en la consola., and 6. Exportar a fichero .txt la lista de usuarios.\n.
- Output Window:** Shows the menu options numbered 1 through 6.
- System Tray:** Shows the date and time (30/11/2025, 18:42) and some system icons.

Se implementa un bucle do-while para reproducir el menú tantas veces como el String salidaPermitida distinta de "S" (Salida permitida = SI) se presenten por parte del usuario. Se deberá insertar intencionadamente la letra S para salir del programa, evitando errores de salida del programa al pulsar accidentalmente cualquier otra tecla del teclado.

## 4. Agregar un usuario a la lista de usuarios

The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name "Naranjo Castillo, Antonio".
- Toolbar:** Includes icons for file operations, search, and other common functions.
- Code Editor:** Displays Java code in the Main.java file. The code handles user input for a new user and creates a new Usuario object. A try-catch block is used to handle NumberFormatException if a non-numeric birth year is entered.

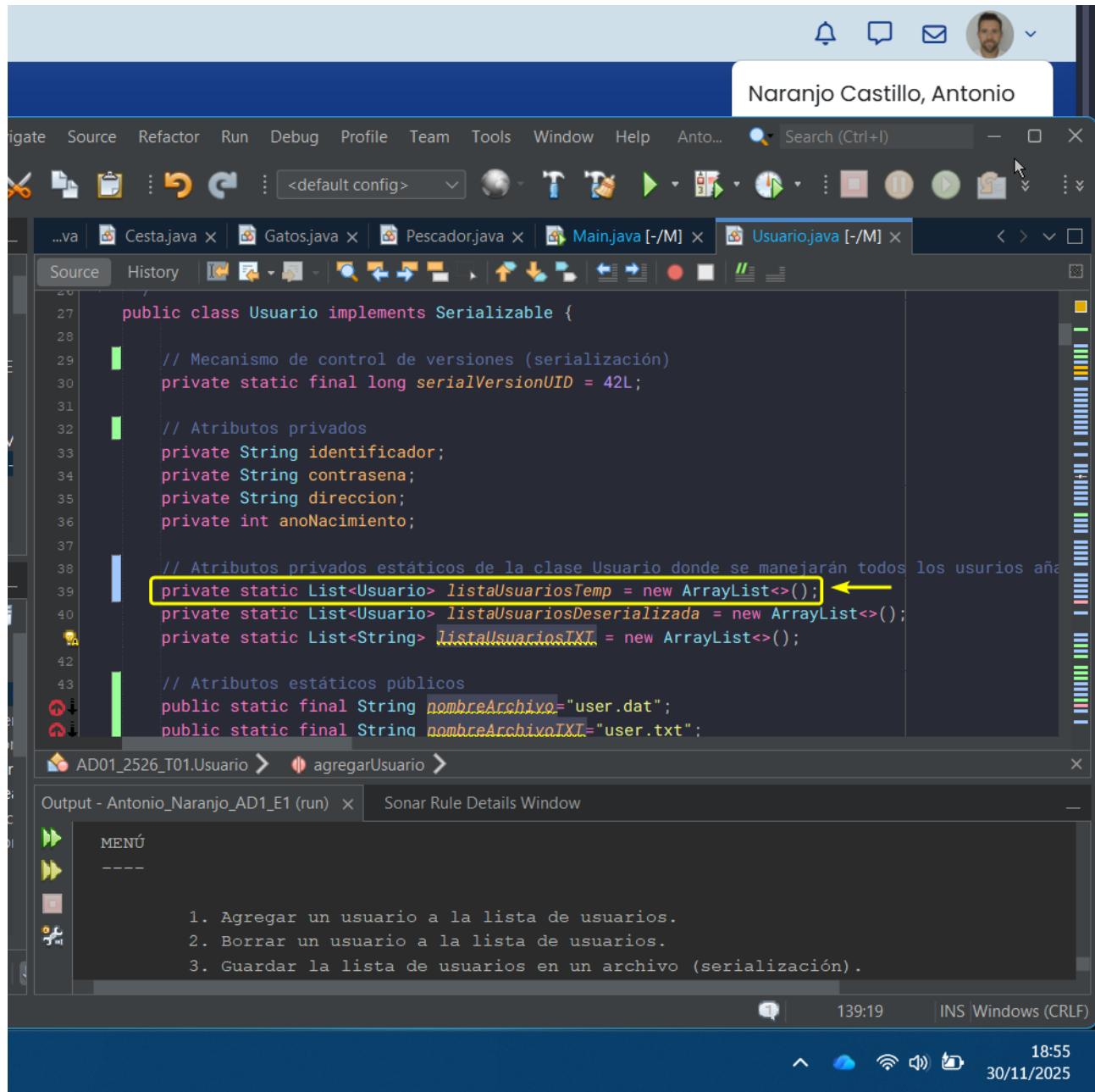
```
118 // Agregar un usuario a la lista de usuarios
119 case 1:
120     mensaje = "Agregar un usuario a la lista de usuarios";
121     System.out.println( String.format("%nLa opción elegida por el usuario es: %d [%s]", args[0], opcion));
122
123     // Se solicita al usuario los atributos de los usuarios
124     System.out.println("Introduzca el identificador del usuario");
125     id = teclado.nextLine();
126     System.out.println("Introduzca la contraseña del usuario");
127     password = teclado.nextLine();
128     System.out.println("Introduzca el dirección del usuario");
129     address = teclado.nextLine();
130     System.out.println("Introduzca el año de nacimiento del usuario");
131     try {
132         yearbirth = Integer.parseInt(teclado.nextLine());
133         // Instanciación del objeto user de la clase Usuario
134         user = new Usuario(id, password, address, yearBirth:yearbirth);
135         // Se agrega el usuario a la lista temporal de usuarios
136         Usuario.agregarUsuario(user);
137     } catch (NumberFormatException ex) {
138         System.err.println("Se ha introducido un formato numérico incorrecto. Se debe introducir como año de nacimiento");
139     }
140 }
```

- Output Window:** Shows the terminal output of the application execution. It prompts for user information and prints the birth year "234".

```
Introduzca el dirección del usuario
234
Introduzca el año de nacimiento del usuario
```

- System Tray:** Shows the date and time as "01/12/2025 20:59".

Una vez seleccionada la opción 1 por el usuario, se procede a solicitar los cuatro atributos que definen el objeto usuario de la clase Usuario, para terminar con la instanciación de dicho objeto usuario, así como la aplicación del método estático agregarUsuario() de la clase Usuario aportando como argumento el objeto usuario recientemente instanciado. Se captura la posible excepción en la cual el usuario del aplicativo pudiera introducir un dato no numérico para la definición del atributo ‘año de nacimiento’.



```
public class Usuario implements Serializable {  
    // Mecanismo de control de versiones (serialización)  
    private static final long serialVersionUID = 42L;  
  
    // Atributos privados  
    private String identificador;  
    private String contraseña;  
    private String dirección;  
    private int añoNacimiento;  
  
    // Atributos privados estáticos de la clase Usuario donde se manejarán todos los usuarios añadidos  
    private static List<Usuario> listaUsuariosTemp = new ArrayList<>(); ←  
    private static List<Usuario> listaUsuariosDeserializada = new ArrayList<>();  
    private static List<String> listaUsuariosTXT = new ArrayList<>();  
  
    // Atributos estáticos públicos  
    public static final String nombreArchivo="user.dat";  
    public static final String nombreArchivoTXT="user.txt";  
}
```

Output - Antonio\_Naranjo\_AD1\_E1 (run) x Sonar Rule Details Window

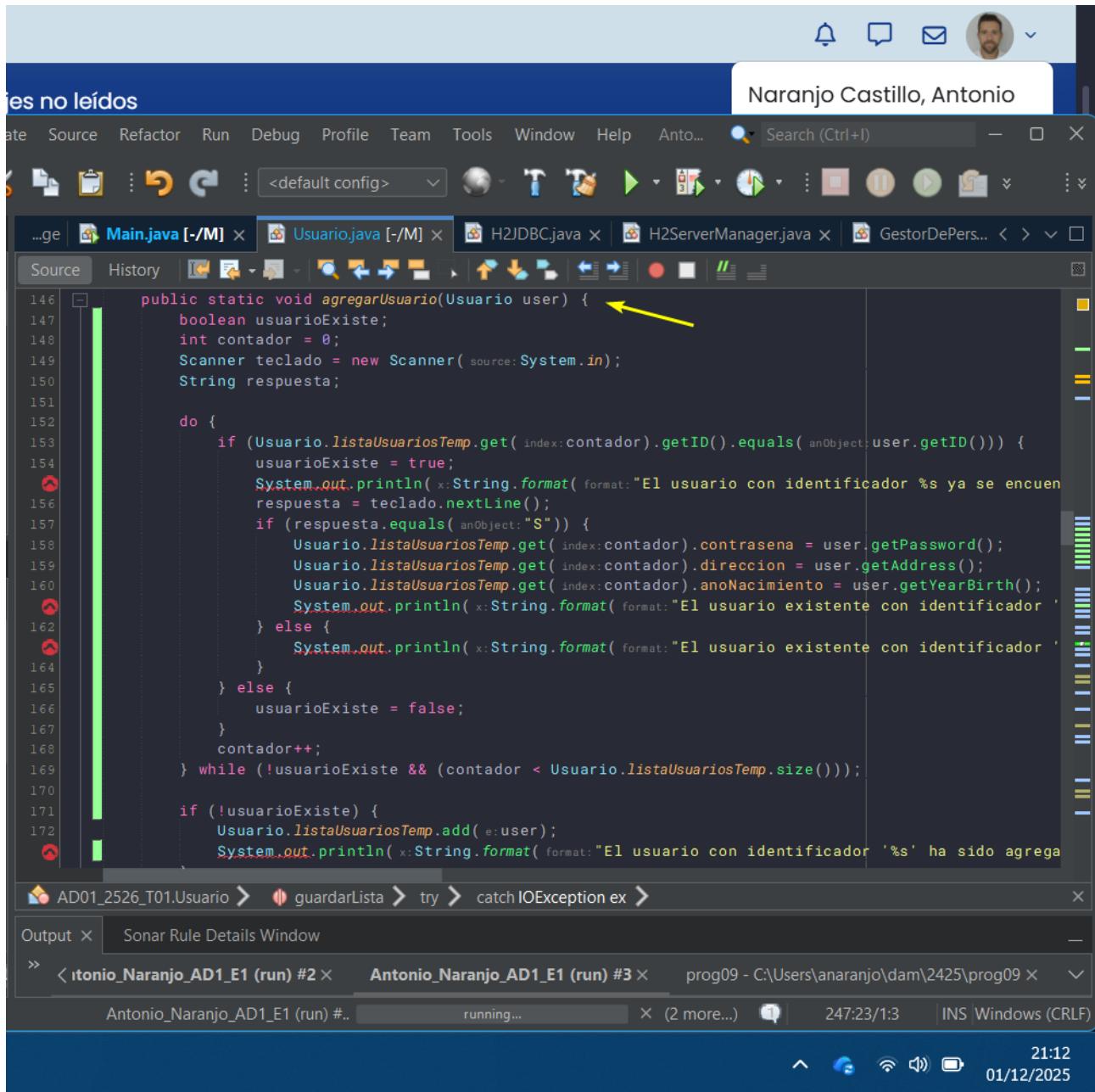
MENÚ

1. Agregar un usuario a la lista de usuarios.  
2. Borrar un usuario a la lista de usuarios.  
3. Guardar la lista de usuarios en un archivo (serialización).

139:19 INS Windows (CRLF)

18:55 30/11/2025

En la clase Usuario se crea una lista temporal donde se almacenarán los distintos objetos usuarios que se vayan instanciando, se trata de un atributo estático privado de la clase Usuario.



The screenshot shows an IDE interface with several tabs at the top: Main.java, Usuario.java, H2JDBC.java, H2ServerManager.java, and GestorDePers... The Main.java tab is active, displaying the following Java code:

```

146     public static void agregarUsuario(Usuario user) {
147         boolean usuarioExiste;
148         int contador = 0;
149         Scanner teclado = new Scanner( source: System.in );
150         String respuesta;
151
152         do {
153             if ( Usuario.listaUsuariosTemp.get( index:contador ).getID().equals( anObject: user.getID() ) ) {
154                 usuarioExiste = true;
155                 System.out.println( x: String.format( format: "El usuario con identificador %s ya se encuentra almacenado en la lista temporal" );
156                 respuesta = teclado.nextLine();
157                 if ( respuesta.equals( anObject: "S" ) ) {
158                     Usuario.listaUsuariosTemp.get( index:contador ).contraseña = user.getPassword();
159                     Usuario.listaUsuariosTemp.get( index:contador ).dirección = user.getAddress();
160                     Usuario.listaUsuariosTemp.get( index:contador ).anoNacimiento = user.getYearBirth();
161                     System.out.println( x: String.format( format: "El usuario existente con identificador %s ha sido actualizado" );
162                 } else {
163                     System.out.println( x: String.format( format: "El usuario existente con identificador %s no ha sido actualizado" );
164                 }
165             } else {
166                 usuarioExiste = false;
167             }
168             contador++;
169         } while ( !usuarioExiste && ( contador < Usuario.listaUsuariosTemp.size() ) );
170
171         if ( !usuarioExiste ) {
172             Usuario.listaUsuariosTemp.add( e:user );
173             System.out.println( x: String.format( format: "El usuario con identificador '%s' ha sido agregado a la lista temporal" );
174         }
175     }

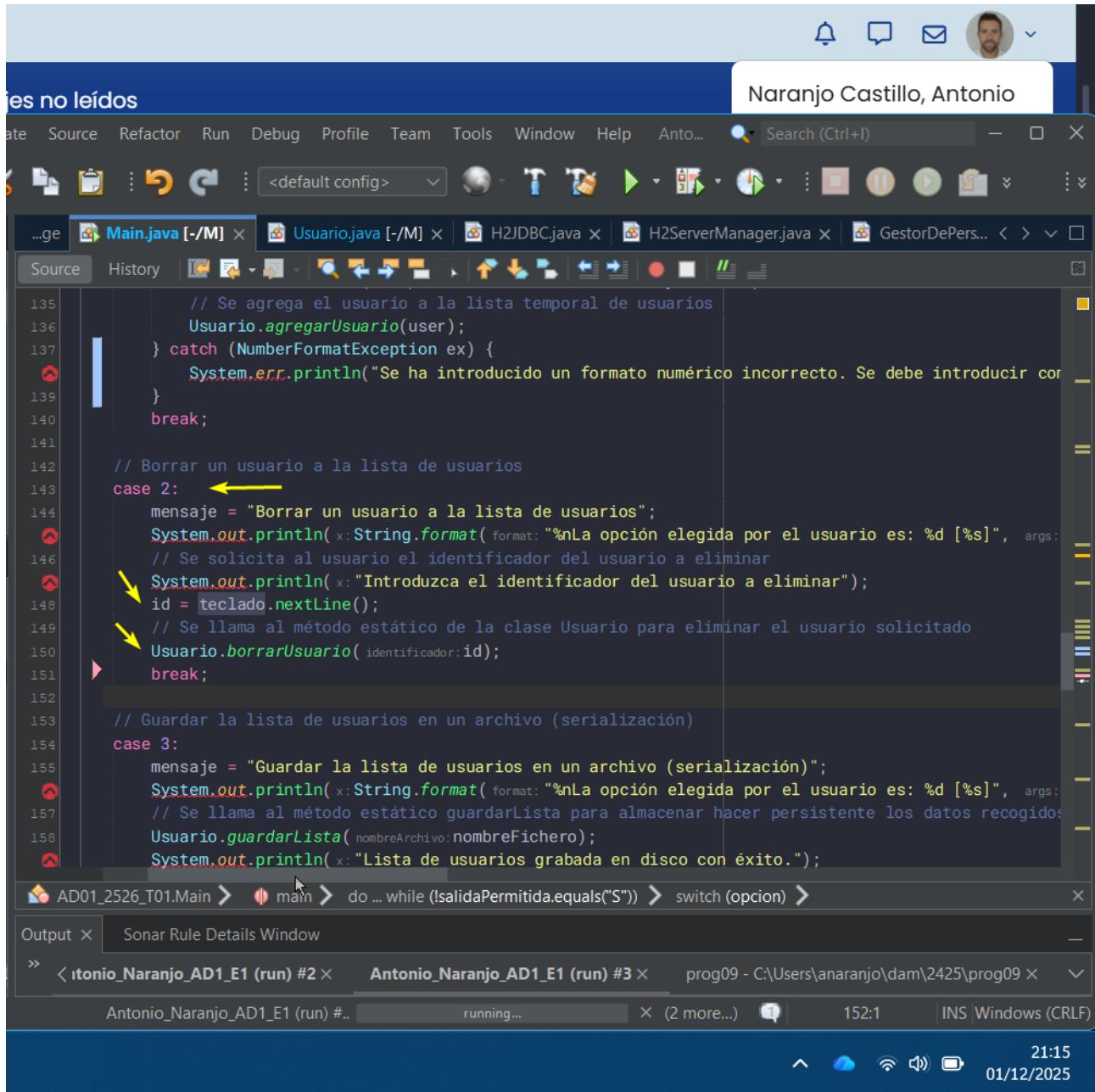
```

The code implements a do-while loop to check if a user already exists in the temporary list. If it does, it asks the user if they want to update the existing user's information. If the user enters 'S', it updates the user's password and address. If the user enters anything else, it does not update the user. If the user does not exist in the list, it adds the new user to the list.

En esta imagen se presenta el método estático `agregarUsuario()` para incorporar a la lista temporal indicada con anterioridad cada uno de los usuarios que se vayan dando de alta. Se tiene la precaución de solicitar al usuario de la aplicación que, en caso de que el usuario a introducir ya se encuentre almacenado en la lista temporal, para ello se comprueba si el atributo identificación ya existe entre los usuarios almacenados en la lista temporal, en tal caso, el usuario deberá introducir el carácter “S” si desea actualizar el resto de los atributos. En caso de que no exista el objeto usuario en la lista temporal, se procede a añadirlo.

Se emplea un bucle do-while tal que mientras el usuario no exista vaya comprobando en cada elemento de la lista temporal el atributo identificación, y con la ayuda de un contador.

## 5. Borrar un usuario de la lista de usuarios



The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name "Naranjo Castillo, Antonio".
- Toolbar:** Includes icons for file operations, search, and other common functions.
- Code Editor:** Displays the `Main.java` file with the following code (lines 135 to 563):
 

```

135     // Se agrega el usuario a la lista temporal de usuarios
136     Usuario.agregarUsuario(user);
137 } catch (NumberFormatException ex) {
138     System.err.println("Se ha introducido un formato numérico incorrecto. Se debe introducir cor-
139 }
140 break;

// Borrar un usuario a la lista de usuarios
142 case 2: ←
143     mensaje = "Borrar un usuario a la lista de usuarios";
144     System.out.println(x: String.format( format: "%nLa opción elegida por el usuario es: %d [%s]", args:
145     // Se solicita al usuario el identificador del usuario a eliminar
146     System.out.println(x: "Introduzca el identificador del usuario a eliminar");
147     id = teclado.nextLine();
148     // Se llama al método estático de la clase Usuario para eliminar el usuario solicitado
149     Usuario.borrarUsuario( identificador:id);
150     break;

// Guardar la lista de usuarios en un archivo (serialización)
153 case 3:
154     mensaje = "Guardar la lista de usuarios en un archivo (serialización)";
155     System.out.println(x: String.format( format: "%nLa opción elegida por el usuario es: %d [%s]", args:
156     // Se llama al método estático guardarLista para almacenar hacer persistente los datos recogidos
157     Usuario.guardarLista( nombreArchivo:nombreFichero);
158     System.out.println(x: "Lista de usuarios grabada en disco con éxito.");
      
```
- Toolbars:** Includes Source, History, and various navigation and search tools.
- Status Bar:** Shows the current run configuration ("AD01\_2526\_T01.Main"), the status ("running..."), and the date/time ("01/12/2025 21:15").

Para borrar un usuario de la lista temporal de usuarios, el usuario del programa deberá seleccionar la opción 2, acto seguido, el programa solicitará la identificación del usuario a eliminar para ejecutar el método estático de la clase Usuario `borrarUsuario()` pasando como argumento el atributo identificador del usuario a borrar.

The screenshot shows an IDE interface with the following details:

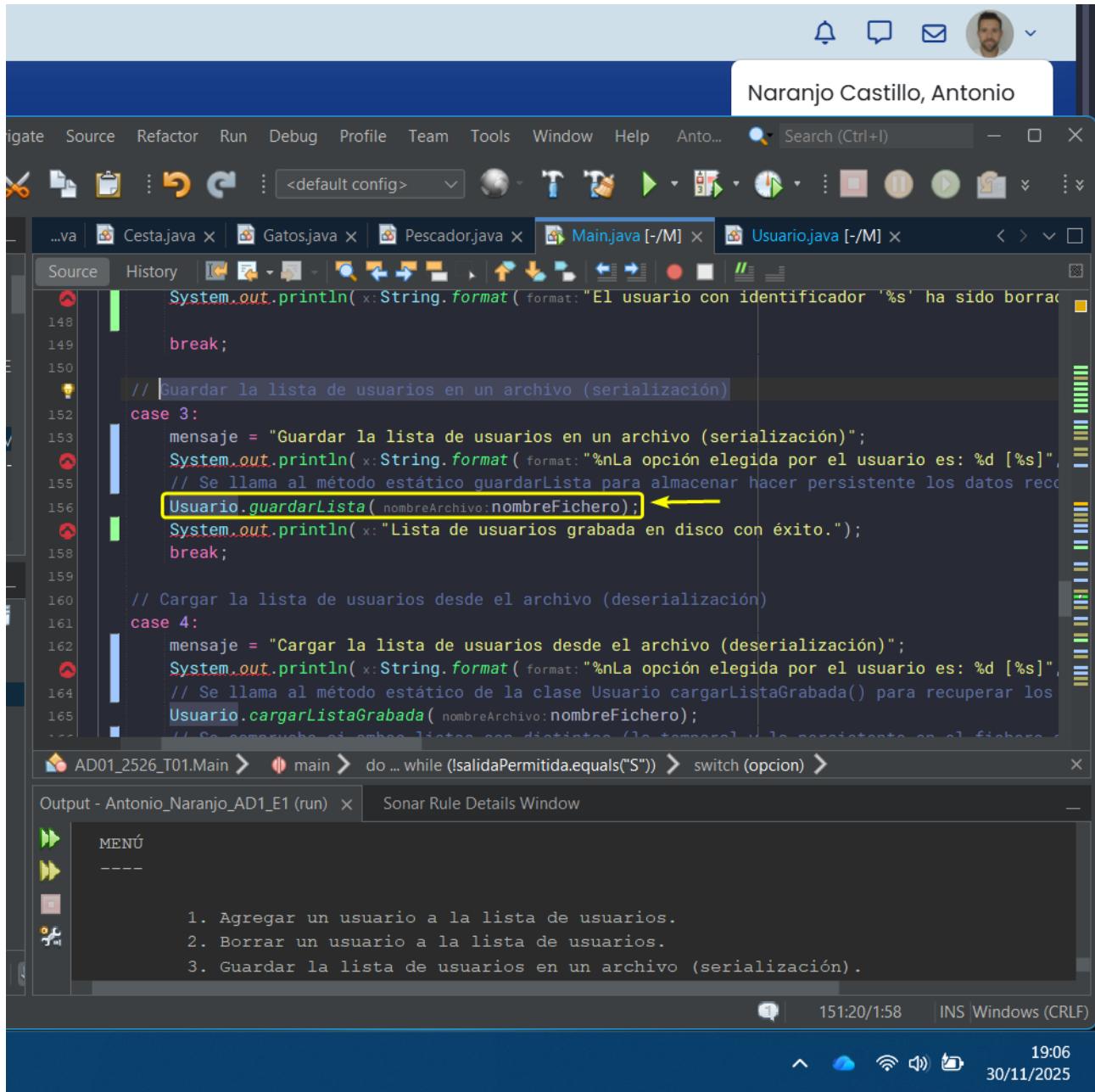
- Title Bar:** Shows "Naranjo Castillo, Antonio" in the top right corner.
- Toolbar:** Includes icons for file operations (New, Open, Save), search, and various development tools.
- Project Explorer:** Shows files like Main.java, Usuario.java, H2JDBC.java, H2ServerManager.java, and GestorDePers... listed.
- Code Editor:** Displays the `borrarUsuario` method from the `Usuario.java` class. The code uses an iterator to safely remove users from a temporary list.

```
199
200     /**
201      * Método para borrar un usuario a la lista de usuarios temporal aportando
202      * el atributo identificador del objeto usuario
203      *
204      * @param identificador String cadena de caracteres identificador del
205      * usuario
206      */
207     public static void borrarUsuario(String identificador) {
208         // Booleano que determina si el usuario existe
209         boolean usuarioExiste = false;
210         // Se instancia un iterador de la lista de usuarios para eliminar el usuario de manera segura
211         Iterator<Usuario> iterador = Usuario.listaUsuariosTemp.iterator();
212         while (iterador.hasNext()) {
213             Usuario user = iterador.next();
214             if (user.getID().equals(anObject: identificador)) {
215                 iterador.remove();
216                 usuarioExiste = true;
217             }
218         }
219         if (usuarioExiste) {
220             System.out.println(x:String.format(format:"El usuario con identificador '%s' ha sido borrado"))
221         } else {
222             System.out.println(x:String.format(format:"El usuario con identificador '%s' no existe en la lista"))
223         }
224     }
225 }
```

- Toolbars:** Includes Source, History, and various navigation and selection tools.
- Status Bar:** Shows the current run configuration, output status (running...), and system information (247:23:1:3, 21:17, 01/12/2025).

Se presenta el método `borrarUsuario` estático de la clase `Usuario`, haciendo uso de un iterador para garantizar el borrado del objeto usuario contenido en la lista temporal de manera segura. Se hace uso de una variable tipo booleano para manejar el mensaje de salida por consola de la existencia del objeto usuario a eliminar en la lista de usuarios temporal.

## 6. Guardar la lista de usuarios en un archivo (serialización)



The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name "Naranjo Castillo, Antonio".
- Toolbar:** Standard IDE tools like file operations, search, and help.
- Project Explorer:** Lists projects: ...va, Cestajava, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M].
- Code Editor:** Displays Java code. A specific line of code is highlighted: `Usuario.guardarLista(nombreArchivo:nombreFichero);`. An arrow points from the text above to this line.
- Output Window:** Shows the command: `AD01_2526_T01.Main > main > do ... while (!salidaPermitida.equals("S")) > switch (opcion) >`.
- Output Sub-Window:** Titled "Output - Antonio\_Naranjo\_AD1\_E1 (run)", it displays a menu with options:
  - MENÚ
  - 
  - 1. Agregar un usuario a la lista de usuarios.
  - 2. Borrar un usuario a la lista de usuarios.
  - 3. Guardar la lista de usuarios en un archivo (serialización).
- System Tray:** Shows the date (30/11/2025), time (19:06), and system status icons.

Para guardar la lista temporal de usuarios en un fichero ubicado en el disco duro del PC, el usuario deberá establecer la opción 3, de esta manera se conseguirá la persistencia de los datos quedando grabados en un archivo del disco duro.

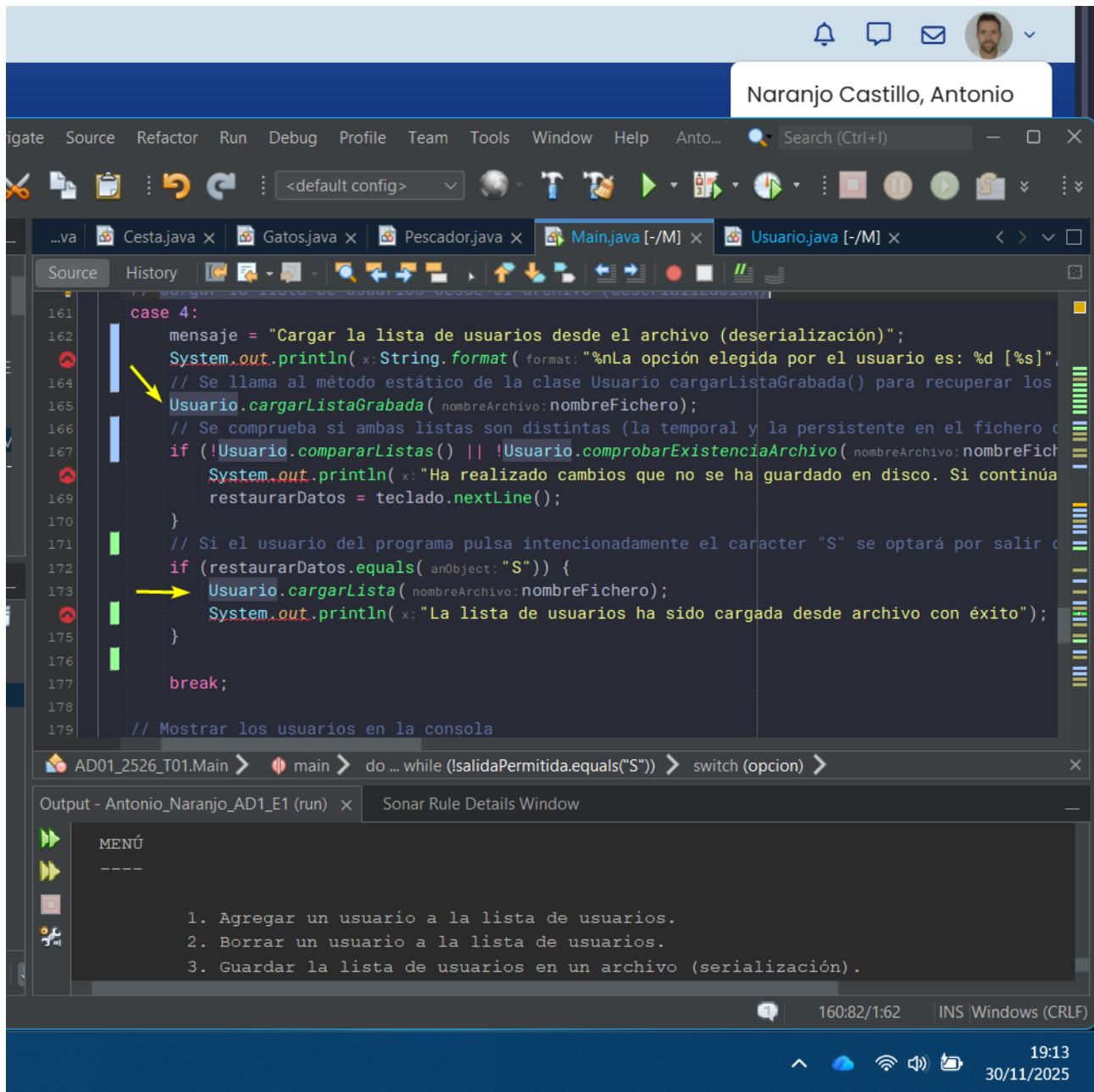
Se ejecuta el método estático de la clase Usuario guardarLista() aportando como argumento el nombre del archivo en cuestión.

The screenshot shows an IDE interface with the following details:

- Top Bar:** Shows navigation tabs like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto...; a search bar labeled "Search (Ctrl+I)"; and user profile icons.
- Title Bar:** Displays the current project name: "Naranjo Castillo, Antonio".
- Toolbar:** Includes standard icons for file operations (New, Open, Save, Print), code navigation, and build/run.
- Code Editor:** Shows the source code for the `Usuario.java` class, specifically the `guardarLista` method. The code uses `FileOutputStream` and  `ObjectOutputStream` to serialize a list of users to a file.
- Output Window:** Shows the command-line output of the application, which includes a menu with options 1, 2, and 3, and a message indicating the serialization process.
- System Tray:** Shows the date and time (30/11/2025, 19:08), battery level, signal strength, and other system status indicators.

Se muestra el método estático de la clase Usuario `guardarLista`, para el cual, se emplean objetos de las clases `FileOutPutStream` y `ObjectOutPutStream`, para transformar la lista de objetos usuarios en bytes (out), y luego, estos bytes guardarlos en el archivo en cuestión (fileOut).

## 7. Cargar la lista de usuarios desde el archivo (deserialización)



```

161
162     case 4:
163         mensaje = "Cargar la lista de usuarios desde el archivo (deserialización)";
164         System.out.println(x.String.format(format:"%nLa opción elegida por el usuario es: %d [%s]");
165         // Se llama al método estático de la clase Usuario cargarListaGrabada() para recuperar los
166         // Usuario.cargarListaGrabada( nombreArchivo:nombreFichero);
167         if (!Usuario.compararListas() || !Usuario.comprobarExistenciaArchivo( nombreArchivo:nombreFich
168             System.out.println(x:"Ha realizado cambios que no se ha guardado en disco. Si continúa
169             restaurarDatos = teclado.nextLine();
170         }
171         // Si el usuario del programa pulsa intencionadamente el carácter "S" se optará por salir d
172         if (restaurarDatos.equals(anObject:"S")) {
173             Usuario.cargarLista( nombreArchivo:nombreFichero);
174             System.out.println(x:"La lista de usuarios ha sido cargada desde archivo con éxito");
175         }
176
177         break;
178
179     // Mostrar los usuarios en la consola

```

AD01\_2526\_T01.Main > main > do ... while (!salidaPermitida.equals("S")) > switch (opcion) >

Output - Antonio\_Naranjo\_AD1\_E1 (run) > Sonar Rule Details Window

MENÚ

1. Agregar un usuario a la lista de usuarios.  
2. Borrar un usuario a la lista de usuarios.  
3. Guardar la lista de usuarios en un archivo (serialización).

160:82/1:62 | INS | Windows (CRLF)  
19:13  
30/11/2025

El usuario del programa seleccionará la opción 4 para cargar los datos persistentes del fichero ubicado en el disco duro del PC. Posteriormente, se ejecutará el método estático de la clase Usuario cargarListaGrabada aportando el nombre del fichero a implementar. La lista de usuarios almacenada en tal fichero se guardará en la lista de usuarios deserializada, y posteriormente, se comparará con la lista temporal por medio del método estático cargarLista() para determinar si ambas listas son iguales, además de, comprobar la existencia del fichero en cuestión, para que en caso de que no lo sean o no exista el fichero, el usuario del programa podrá determinar si desea actualizar los datos o realizar cualquier otra opción del menú.

The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name "Naranjo Castillo, Antonio".
- Toolbar:** Includes standard icons for Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... along with a search bar.
- Project Bar:** Displays multiple open files: ...va, Cestajava.java, Gatos.java, Pescador.java, Main.java, and Usuario.java.
- Code Editor:** The main window displays Java code for reading users from a file. The code uses `FileInputStream` to read bytes and `ObjectInputStream` to deserialize them into a list of `Usuario` objects.
- Status Bar:** Shows the current time as 13:19 and the date as 30/11/2025.

```
278     * almacenarán los datos de los usuarios.
279     */
280     public static void cargarListaGrabada(String nombreArchivo) {
281
282         try {
283             // Obtener los bytes almacenados en el archivo
284             FileInputStream fileIn = new FileInputStream( name:nombreArchivo);
285             // Convertir los bytes en objetos
286             ObjectInputStream in = new ObjectInputStream( in:fileIn) {
287
288                 // Se almacenan los objetos usuarios deserializados en una lista
289                 Usuario.listadoUsuariosDeserializada = (List<Usuario>) in.readObject();
290
291             // Manejo de excepciones
292             } catch (FileNotFoundException ex) {
293                 System.err.println("Error: El archivo " + nombreArchivo + " no existe.");
294             } catch (IOException ex) {
295                 System.err.println( x:"Error de Entrada/Salida: Falló la lectura del fichero.");
296             } catch (ClassNotFoundException ex) {
297                 System.out.println("Error: La clase " + ex.getMessage() + " no se encontró.");
298             }
299         }
300     }
```

Se presenta el método `cargarListaGrabada` empleándose objetos `FileInputStream` y `ObjectInputStream`, el primero para obtener los bytes del archivo guardado en el disco, y el segundo para transformar los bytes en una lista de objetos usuarios que posteriormente se almacenarán en la lista de usuarios deserializada.

The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the name "Naranjo Castillo, Antonio".
- Toolbar:** Includes standard icons for file operations, search, and navigation.
- Project Explorer:** Lists files like Cestajava.java, Gatos.java, Pescador.java, Main.java, and Usuario.java.
- Code Editor:** Displays Java code for comparing two lists of users. The code uses nested loops and the equals() method to check if two lists are equal.
- Output Window:** Shows a menu with options: 1. Agregar un usuario a la lista de usuarios., 2. Borrar un usuario a la lista de usuarios., 3. Guardar la lista de usuarios en un archivo (serialización.).
- System Tray:** Shows the date and time (30/11/2025, 19:29).

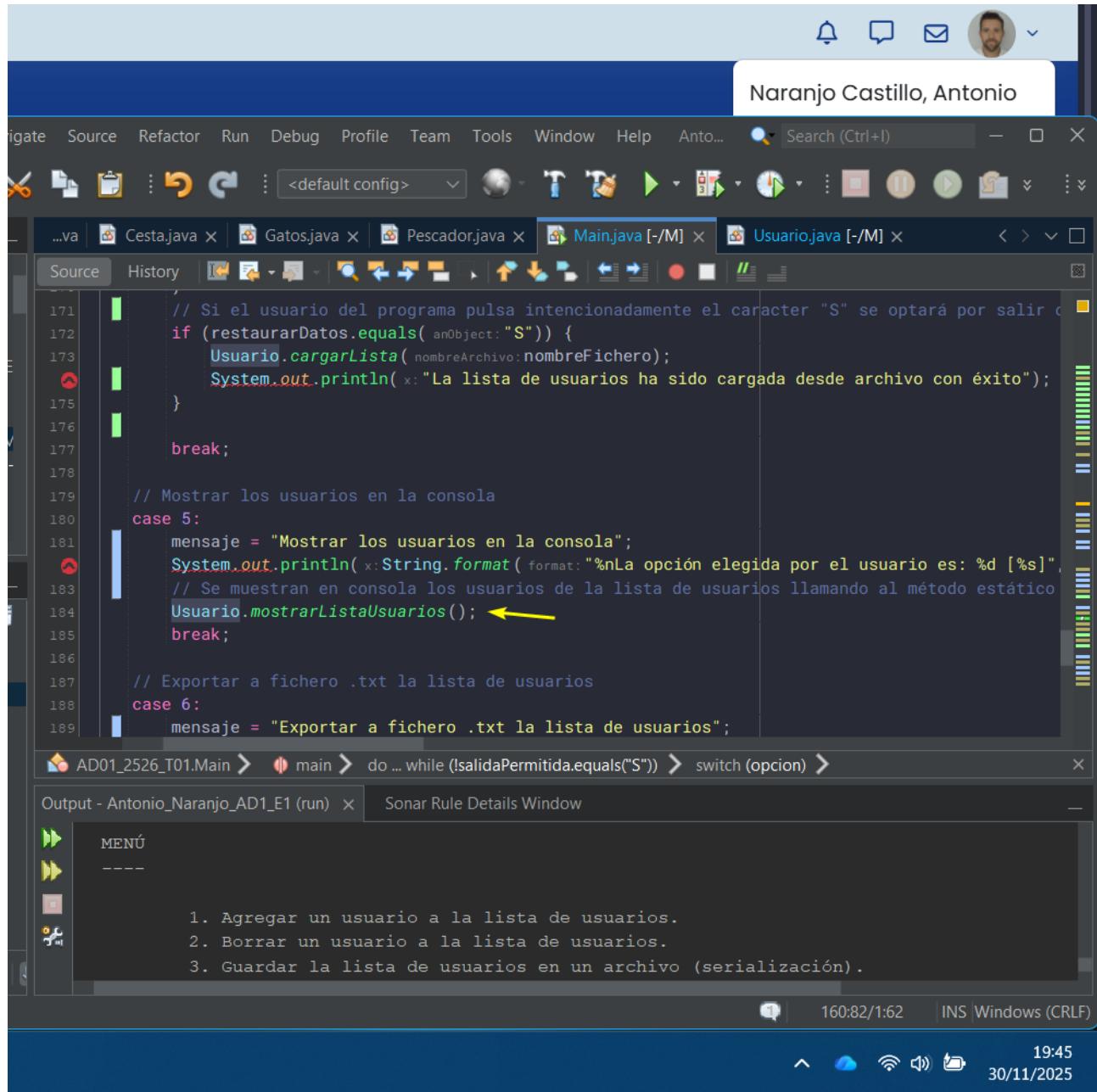
Posteriormente, con el método igualmente estático compararListas(), se compararán las listas deserializada y temporal para comprobar si son iguales, para ello, primero se comprueba si ambas listas tienen el mismo número de elementos, en segundo lugar, para cada elemento se comprueba que sus atributos sean iguales, a la vez que se comprueban que los usuarios siguen el mismo orden en ambas listas.

The screenshot shows an IDE interface with the following details:

- Top Bar:** Shows navigation links like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto...; a search bar labeled "Search (Ctrl+I)"; and user profile icons.
- Title Bar:** Displays the current project name: "Naranjo Castillo, Antonio".
- Toolbar:** Includes standard icons for file operations (New, Open, Save, Print), navigation (Back, Forward, Home), and other development tools.
- Code Editor:** Shows the Java code for the `cargarLista` method in the `Usuario.java` file. The code reads a file named `nombreArchivo` using `ObjectInputStream` to deserialize a list of `Usuario` objects.
- Output Window:** Shows the command line output for the application, which includes a menu with three options:
  - 1. Agregar un usuario a la lista de usuarios.
  - 2. Borrar un usuario a la lista de usuarios.
  - 3. Guardar la lista de usuarios en un archivo (serialización).
- System Tray:** Shows the date and time (30/11/2025, 19:35), battery level, signal strength, and other system status indicators.

Se muestra el método `cargarLista()`, mediante el cual se actualizan los objetos usuarios de la lista temporal a partir de los datos obtenidos del fichero grabado en el disco duro del PC. De manera similar al método `cargarListaGrabada`, se empleará un objeto `FileInputStream` envuelto en un objeto `ObjectInputStream` al cual se le pasa como argumento para obtener los bytes del archivo del disco duro y transformarlos en una lista de objetos usuarios.

## 8. Mostrar los usuarios en consola



The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name "Naranjo Castillo, Antonio".
- Toolbar:** Standard IDE tools like file operations, search, and help.
- Project Explorer:** Lists files: ...va, Cestajava.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M].
- Code Editor:** Displays Java code. A yellow arrow points to the line `Usuario.mostrarListaUsuarios();` in the `case 5` block.
- Output Window:** Shows the menu options:
  - MENÚ
  - 
  - 1. Agregar un usuario a la lista de usuarios.
  - 2. Borrar un usuario a la lista de usuarios.
  - 3. Guardar la lista de usuarios en un archivo (serialización).
- System Tray:** Shows the date and time (30/11/2025, 19:45), battery level, signal strength, and other system icons.

Cuando el usuario del programa seleccione la opción 5 se mostrarán los resultados en la consola, mostrando todos los usuarios almacenados en la lista temporal de usuarios.

The screenshot shows an IDE interface with the following details:

- Top Bar:** Shows navigation tabs like Navigate, Source, Refactor, Run, etc., and a search bar labeled "Search (Ctrl+I)".
- Title Bar:** Displays the current project name "Naranjo Castillo, Antonio".
- Toolbars:** Includes standard Java development tools like file operations, build, and run.
- Code Editor:** Shows the `Usuario.java` file content. The code includes a static method `mostrarListaUsuarios()` and an overridden `toString()` method. The `toString()` method uses `String.format` to print user attributes: ID, Password, Address, and Year of Birth.
- Terminal Window:** A separate window titled "AD01\_2526\_T01.Usuario" displays a menu:

```
MENÚ
-----
1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
```
- System Tray:** Shows icons for battery, signal, and date/time (30/11/2025, 19:48).

Se presentan los métodos `mostrarListaUsuarios()`, estático, y `toString()`, para imprimir a modo de cadena de texto todos y cada uno de los objetos usuarios contenidos en la lista temporal de usuarios.

El método `toString()` devuelve un String cadena de caracteres mostrando todos los atributos que definen a cada usuario, según su método constructor presentado al inicio de este documento.

## 9. Exporta a fichero .txt la lista de usuarios

The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name "Naranjo Castillo, Antonio".
- Toolbar:** Standard IDE tools like剪切 (Cut), 复制 (Copy), 粘贴 (Paste), etc.
- MenuBar:** Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, Auto... (Auto-completion).
- Search Bar:** Search (Ctrl+I).
- Toolbars:** Source, History, and several others related to code navigation and search.
- Code Editor:** Displays Java code in the `Usuario.java` file. The code handles option 6 for exporting users to a file. A yellow arrow points to the line `Usuario.escribirTXT(nombreArchivo:nombreFicheroTXT);`.
- Output Window:** Shows the menu options: 1. Agregar un usuario a la lista de usuarios., 2. Borrar un usuario a la lista de usuarios., 3. Guardar la lista de usuarios en un archivo (serialización.).
- System Tray:** Shows the date and time (30/11/2025, 19:52), battery level, signal strength, and other system icons.

Se selecciona la opción 6 para escribir en un archivo de texto cada uno de los objetos almacenados en la lista temporal. Para ello se emplea el método estático `escribirTXT()`.

Se define como atributo estático público de la clase `Usuario` el nombre del archivo “`user.txt`” de esta manera al pulsar ENTER por defecto se usará tal archivo (de la misma manera se procede con el archivo `user.dat` al iniciar el programa).

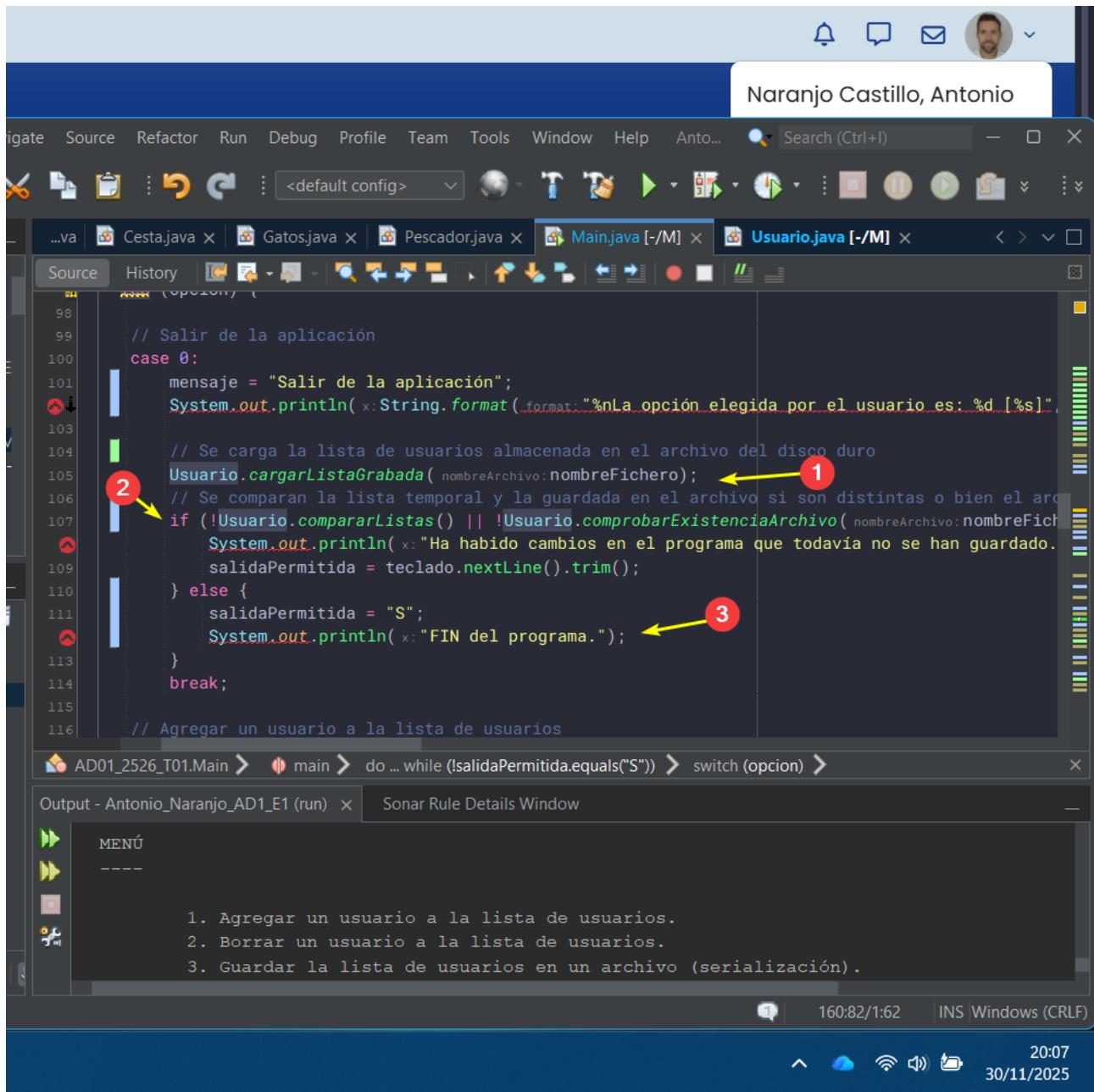
The screenshot shows an IDE interface with the following details:

- Top Bar:** Shows navigation links like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto...; a search bar labeled "Search (Ctrl+I)"; and user profile icons.
- Toolbar:** Includes standard icons for file operations (New, Open, Save, Print), code navigation (Find, Replace, Go To), and build/run (Run, Stop, Build).
- Project Explorer:** Shows projects like ...va, Cestajava, Gatos.java, Pescador.java, Main.java, and Usuario.java.
- Code Editor:** Displays the `Usuario.java` file with the following code:public static void escribirTXT(String nombreArchivo) {  
 // Se limpia la lista para reutilizarla si es el caso  
 Usuario.listaUsuariosTXT.clear();  
 // Se alimenta la lista de usuarios empleando el método toString() para cada objeto Usuarios  
 for (Usuario user : Usuario.listaUsuariosTemp) {  
 listaUsuariosTXT.add(user.toString());  
 }  
  
 try {  
 // Se convierte los caracteres en bytes  
 FileWriter fw = new FileWriter(fileName: nombreArchivo, append: false);  
 // Se establece un flujo de escritura  
 BufferedWriter bw = new BufferedWriter(out: fw)) {  
  
 // Por cada cadena de caracteres almacenada en la lista listaUsuariosTXT se escribe en el archivo.  
 // Se introducir un salto de carro después de cada String str.  
 for (String str : Usuario.listaUsuariosTXT) {  
 bw.write(str);  
 bw.newLine();  
 }  
 // Manejo de excepciones de Entrada/Salida  
 } catch (IOException ex) {  
 }  
}
- Output Window:** Shows the output for the task: "AD01\_2526\_T01.Usuario".
- Sonar Rule Details Window:** Shows a menu with options: 1. Agregar un usuario a la lista de usuarios., 2. Borrar un usuario a la lista de usuarios., 3. Guardar la lista de usuarios en un archivo (serialización.).
- System Tray:** Shows icons for battery, signal, and date/time (30/11/2025, 19:57).

El método `escribirTXT` estático, de la clase `Usuario`, emplea dos objetos para transformar los caracteres en bytes, un objeto `BufferedWriter` que envuelve a un `FileWriter`, el primero capta un flujo de caracteres y el segundo los transforma en bytes para finalmente hacerlos persistente en el fichero \*.txt en cuestión.

Previamente, se define una lista de cadenas de caracteres `listaUsuariosTXT`, limpia inicialmente, Strings obtenidos de aplicar el método `toString()` sobre cada objeto usuario almacenada en la lista temporal. Se emplea un bucle `for-each` para ello, y posteriormente se vuelve a utilizar para escribir cada String en el archivo .txt sumándole un salto de carro y de línea mediante el método `newLine()`.

## 10. Salir de la aplicación



```

98
99     // Salir de la aplicación
100    case 0:
101        mensaje = "Salir de la aplicación";
102        System.out.println( x:String.format( "%nLa opción elegida por el usuario es: %d [%s]" ,
103                                     opcion, mensaje));
104
105        // Se carga la lista de usuarios almacenada en el archivo del disco duro
106        Usuario.cargarListaGrabada( nombreArchivo:nombreFichero); 1
107
108        // Se comparan la lista temporal y la guardada en el archivo si son distintas o bien el archivo no existe
109        if (!Usuario.compararListas() || !Usuario.comprobarExistenciaArchivo( nombreArchivo:nombreFichero))
110            System.out.println( x:"Ha habido cambios en el programa que todavía no se han guardado.");
111            salidaPermitida = teclado.nextLine().trim();
112        } else {
113            salidaPermitida = "S";
114            System.out.println( x:"FIN del programa."); 3
115        }
116        break;
117
118    // Agregar un usuario a la lista de usuarios

```

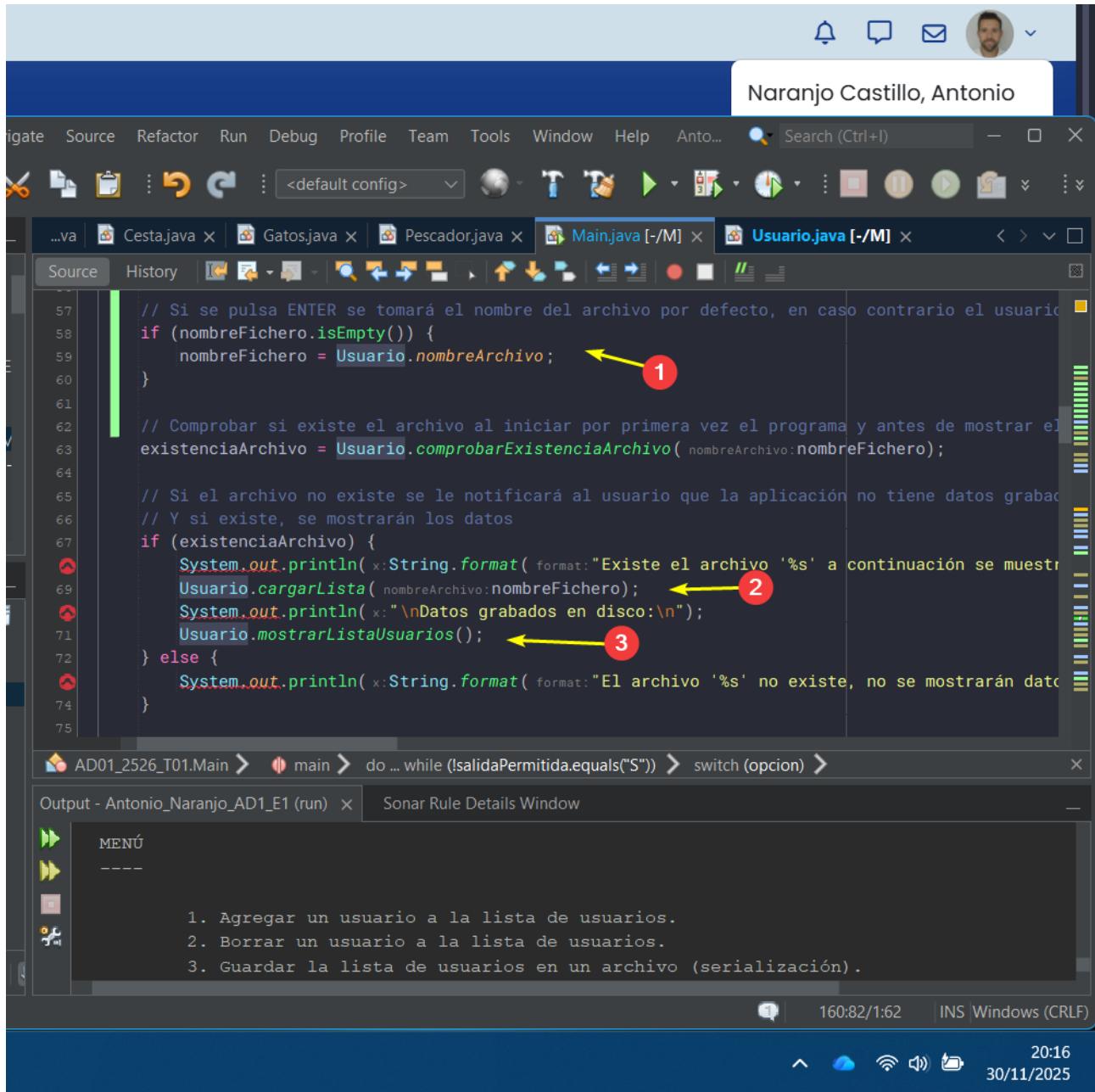
The screenshot shows the Java code for exiting the application. The code handles option 0 by printing a message, calling the `cargarListaGrabada()` method (marked with a red circle 1), comparing the temporary and stored lists (marked with a red circle 2), and finally printing a "FIN del programa." message if no changes were made (marked with a red circle 3).

The IDE interface includes tabs for Main.java and Usuario.java, and a bottom status bar showing the date and time.

Se selecciona la opción 0 para salir del programa, pero antes, se llamará al método `cargarListaGrabada()` pasando como argumento el nombre del fichero en disco, para posteriormente comprobar si la lista obtenida deserializada coincide con la lista temporal y también comprobar si ese archivo existe en el disco duro, ofreciendo la oportunidad de volver atrás, no salir del programa y poder llevar a cabo otra opción ofrecida por el menú del aplicativo.

Finalmente, si el archivo existe y ambas listas de objetos usuarios, tanto la temporal como la deserializada, son iguales, el programa sí finalizará sin preguntar al usuario dado que los cambios están guardados en el fichero del disco duro.

## 11. ¿Existe el archivo ‘user.dat’?

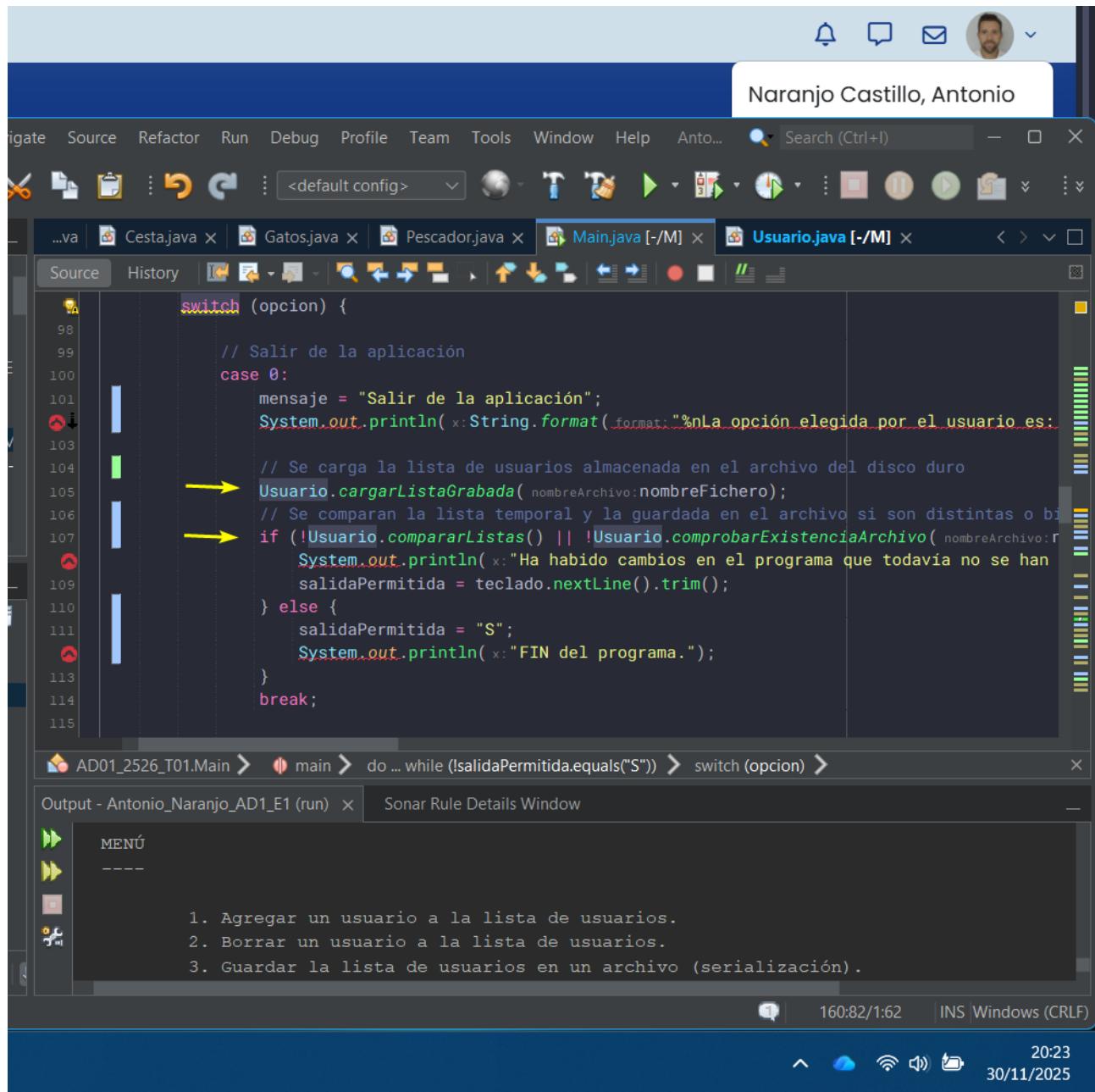


The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name: "Naranjo Castillo, Antonio".
- Toolbar:** Standard IDE tools like file operations, search, and help.
- Project Explorer:** Lists files: "...va", Cestajava.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M].
- Code Editor:** Displays Java code for Main.java. The code checks if a file named 'user.dat' exists. If it does, it reads the data and prints it to the console. If it doesn't, it informs the user. Three specific lines of code are highlighted with red circles and numbered 1, 2, and 3:
  - Line 59: `nombreFichero = Usuario.nombreArchivo;` (highlighted with circle 1)
  - Line 69: `Usuario.cargarLista(nombreArchivo:nombreFichero);` (highlighted with circle 2)
  - Line 70: `Usuario.mostrarListaUsuarios();` (highlighted with circle 3)
- Output Window:** Shows the menu options:
  - MENÚ
  - 
  - 1. Agregar un usuario a la lista de usuarios.
  - 2. Borrar un usuario a la lista de usuarios.
  - 3. Guardar la lista de usuarios en un archivo (serialización).
- System Tray:** Shows the date and time: 30/11/2025, 20:16.

Al ejecutarse la aplicación, se comprueba si existe el fichero user.dat. Para el caso, se ha tenido en cuenta que si el usuario presiona ENTER se usará el nombre user.dat por defecto. Para ello, se emplea el método estático comprobarExistenciaArchivo(). Si no existe, se avisará al usuario de que la aplicación no tiene datos grabados. Si existe, se cargarán los datos guardados de modo que los usuarios que hubiera en el fichero se cargarán en una lista empleando para ello el método estático cargarLista(), y luego, con el método mostrarListaUsuarios() mostrando en consola la lista de los objetos usuarios toString() almacenados en el fichero.

## 12. Advertencia sobre datos no guardados antes de salir



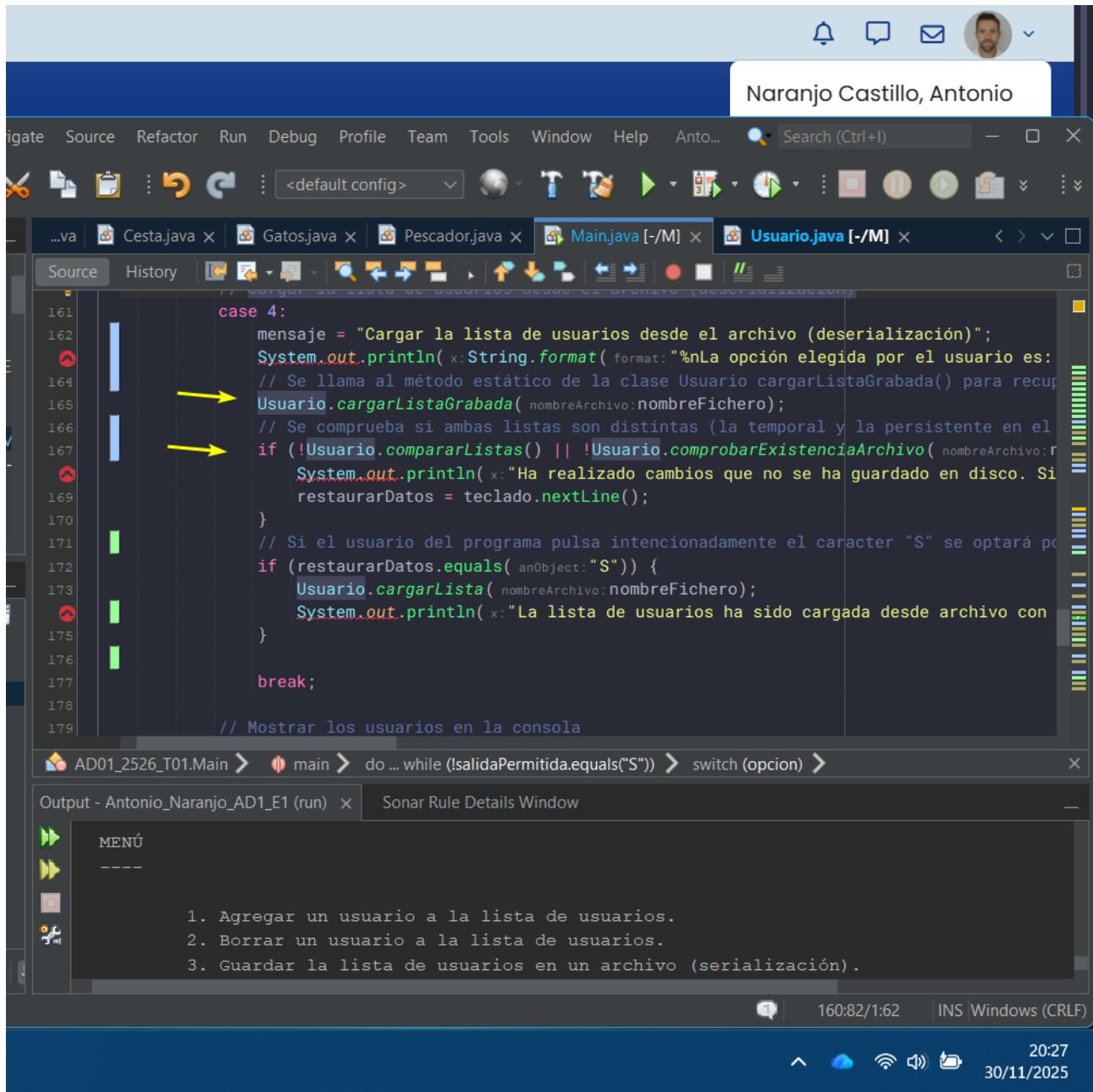
```
switch (opcion) {
    case 0:
        mensaje = "Salir de la aplicación";
        System.out.println(String.format("%nLa opción elegida por el usuario es:"));
        // Se carga la lista de usuarios almacenada en el archivo del disco duro
        Usuario.cargarListaGrabada(nombreArchivo:nombreFichero);
        // Se comparan la lista temporal y la guardada en el archivo si son distintas o bien
        if (!Usuario.compararListas() || !Usuario.comprobarExistenciaArchivo(nombreArchivo:nombreFichero))
            System.out.println("Ha habido cambios en el programa que todavía no se han guardado");
            salidaPermitida = teclado.nextLine().trim();
        } else {
            salidaPermitida = "S";
            System.out.println("FIN del programa.");
        }
    break;
}
```

The screenshot shows an IDE interface with the following details:

- Toolbar:** Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, Auto... (with a search bar labeled "Search (Ctrl+I)"), and various icons for file operations.
- Project Bar:** Shows files like Cesta.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M].
- Code Editor:** Displays the Java code above, with annotations and arrows highlighting specific parts of the code related to reading from a file and comparing lists.
- Output Bar:** Shows the execution path: AD01\_2526\_T01.Main > main > do ... while (!salidaPermitida.equals("S")) > switch (opcion).
- Output Window:** Shows the menu options:
  - MENÚ
  - 
  - 1. Agregar un usuario a la lista de usuarios.
  - 2. Borrar un usuario a la lista de usuarios.
  - 3. Guardar la lista de usuarios en un archivo (serialización).
- System Tray:** Shows the date (30/11/2025), time (20:23), and system status indicators.

En caso de ejecutar la aplicación y modificar algún dato, si el usuario del programa intenta salir de la aplicación sin haber guardado los cambios, se le avisa indicando que los cambios no han sido guardados. Tal y como se explicó en el punto anterior sobre la selección de la opción 0, se comprueba si la lista temporal es igual a la lista deserializada, y si no es así o el archivo no existe, se le avisará al usuario del programa antes de salir.

### 13. ¿Recuperar datos de disco?



```

161
162     mensaje = "Cargar la lista de usuarios desde el archivo (deserialización)";
163     System.out.println(x:String.format(format:"%nLa opción elegida por el usuario es:");
164     // Se llama al método estático de la clase Usuario cargarListaGrabada() para recuperar los datos del archivo
165     Usuario.cargarListaGrabada(nombreArchivo:nombreFichero);
166     // Se comprueba si ambas listas son distintas (la temporal y la persistente en el fichero)
167     if (!Usuario.compararListas() || !Usuario.comprobarExistenciaArchivo(nombreArchivo:nombreFichero))
168         System.out.println(x:"Ha realizado cambios que no se ha guardado en disco. Si lo desea, pulse 'S' para restaurarlos");
169     restaurarDatos = teclado.nextLine();
170 }
171 // Si el usuario del programa pulsa intencionadamente el carácter "S" se optará por restaurar los cambios
172 if (restaurarDatos.equals(anObject:"S")) {
173     Usuario.cargarLista(nombreArchivo:nombreFichero);
174     System.out.println(x:"La lista de usuarios ha sido cargada desde archivo con éxito");
175 }
176
177 break;
178
179 // Mostrar los usuarios en la consola

```

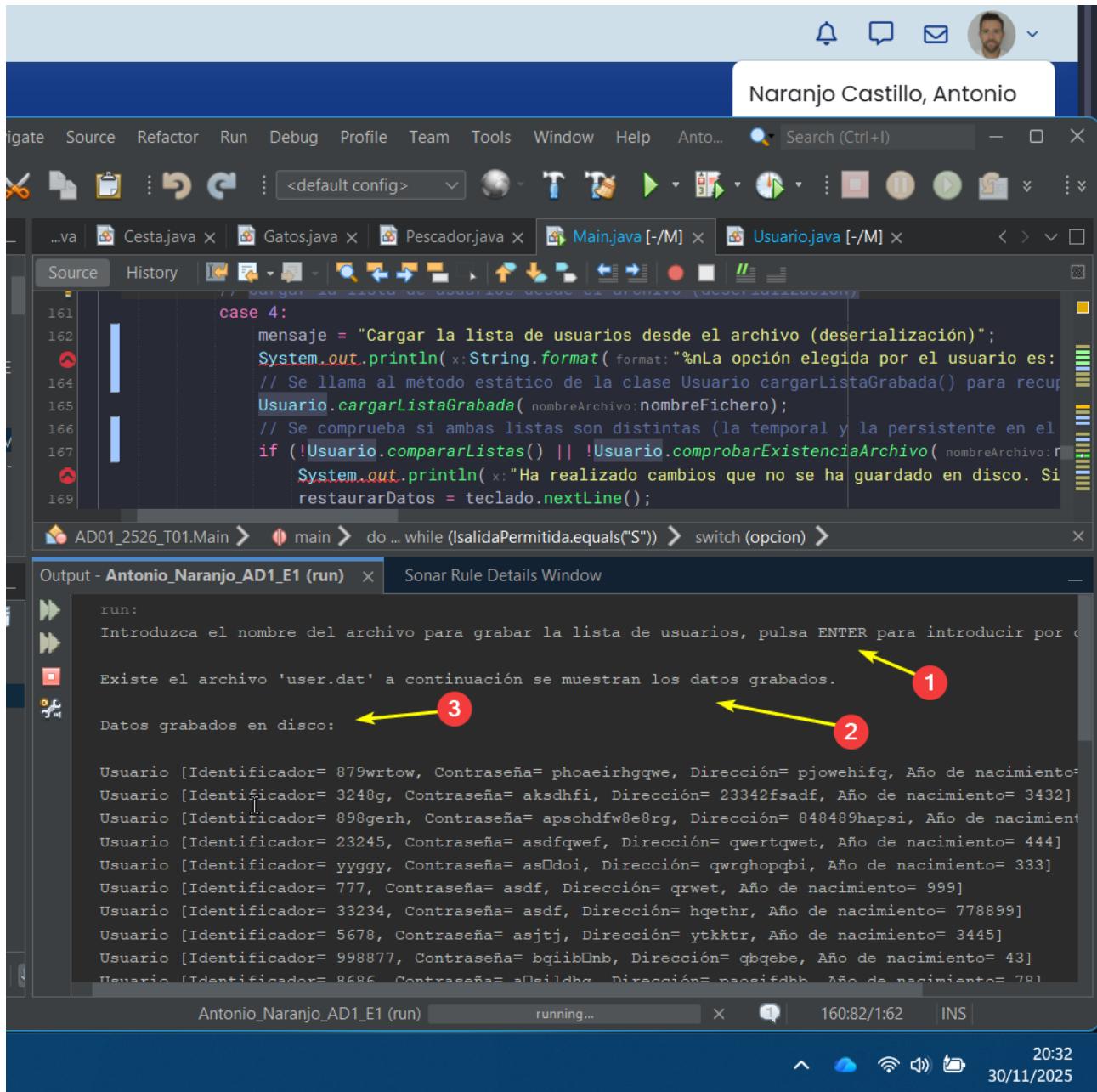
The screenshot shows a Java IDE interface with several tabs open at the top: ...va, Cestajava.x, Gatos.java, Pescador.java, Main.java [-/M] (active), and Usuario.java [-/M]. The code editor displays a switch statement with case 4 handling user data persistence via serialization and deserialization. Two yellow arrows highlight the line `Usuario.cargarListaGrabada(nombreArchivo:nombreFichero);`. The output window below shows a menu with options 1, 2, and 3 related to user management.

Si se ejecuta la opción de recuperar datos, habiendo realizado cambios en el programa, se advertirá al usuario que los cambios se perderán, puesto que se cargarán los datos del fichero en la lista, y se le pedirá confirmación antes de continuar. Este punto se tiene en cuenta al aplicar la opción 4 del menú, y en cierto modo ya se ha detallado cómo se procesa este aspecto, básicamente, se compara la lista deserializada del archivo con la lista temporal de tal manera que si son diferentes se le da la opción al usuario de retroceder antes de sustituir el contenido de la lista temporal.

El usuario deberá seleccionar el carácter “S” intencionadamente para que se proceda con la carga de la lista deserializada, evitando en cierto modo, que accidentalmente el usuario pudiera pulsar otra tecla del teclado.

## 14. Resultados del programa en consola

### 14.1. Inicio del programa



The screenshot shows an IDE interface with several tabs open at the top: ...va, Cestajava x, Gatos.java x, Pescador.java x, Main.java [-/M] x, and Usuario.java [-/M] x. The Main.java tab is active. Below the tabs, there's a toolbar with various icons. The main editor area contains Java code. The code includes a switch statement with a case 4 block that handles file I/O for reading user data from a file named 'user.dat'. The output window below shows the program's interaction with the user:

```

run:
Introduzca el nombre del archivo para grabar la lista de usuarios, pulsa ENTER para introducir por defecto 'user.dat'

Existe el archivo 'user.dat' a continuación se muestran los datos grabados.

Datos grabados en disco: 1
Usuario [Identificador= 879wrtow, Contraseña= phoaerhggwe, Dirección= pjowehifq, Año de nacimiento= 1999]
Usuario [Identificador= 3248g, Contraseña= aksdhfi, Dirección= 23342fsadf, Año de nacimiento= 3432]
Usuario [Identificador= 898gerh, Contraseña= apsohdfw8e8rg, Dirección= 848489hapsi, Año de nacimiento= 1999]
Usuario [Identificador= 23245, Contraseña= asdfqwef, Dirección= qwertqwet, Año de nacimiento= 444]
Usuario [Identificador= yyggy, Contraseña= asldoi, Dirección= qrghopqbi, Año de nacimiento= 333]
Usuario [Identificador= 777, Contraseña= asdf, Dirección= qrwet, Año de nacimiento= 999]
Usuario [Identificador= 33234, Contraseña= asdf, Dirección= hqethr, Año de nacimiento= 778899]
Usuario [Identificador= 5678, Contraseña= asjtz, Dirección= ytkktr, Año de nacimiento= 3445]
Usuario [Identificador= 998877, Contraseña= bqiibOnb, Dirección= qbgebe, Año de nacimiento= 43]
Usuario [Identificador= 8686, Contraseña= sPnildha, Dirección= nnsifdhh, Año de nacimiento= 781]

```

The output window has three numbered callouts pointing to specific parts of the text:

- Callout 1 points to the line "Existe el archivo 'user.dat' a continuación se muestran los datos grabados."
- Callout 2 points to the line "Datos grabados en disco:"
- Callout 3 points to the first line of the user data list.

The status bar at the bottom shows "Antonio\_Naranjo\_AD1\_E1 (run)" and "running...". The system tray at the bottom right shows the date and time as "20:32 30/11/2025".

Al iniciar el programa se pregunta al usuario del programa el nombre del archivo binario. Si pulsa ENTER se asignará el nombre por defecto ‘user.dat’.

Luego, si el archivo existe se imprime en consola y acto seguido se presenta la lista de usuarios guardados en el fichero.

The screenshot shows an IDE interface with several tabs at the top: Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... A search bar is also present. Below the tabs, there are toolbars for file operations like Open, Save, and Print, and for navigation like Find, Replace, and Go To.

The main workspace displays Java code in a source editor:

```
161     mensaje = "Cargar la lista de usuarios desde el archivo (deserialización)";
162     System.out.println(x:String.format( format:"%nLa opción elegida por el usuario es:");
163     // Se llama al método estático de la clase Usuario cargarListaGrabada() para recuperar
164     // los datos de la lista persistente en el disco duro
165     Usuario.cargarListaGrabada( nombreArchivo:nombreFichero);
166     // Se comprueba si ambas listas son distintas (la temporal y la persistente en el disco)
167     if (!Usuario.compararListas() || !Usuario.comprobarExistenciaArchivo( nombreArchivo:nombreFichero));
168         System.out.println( x:"Ha realizado cambios que no se ha guardado en disco. Si desea
169         restaurarDatos = teclado.nextLine();
```

The code editor has several annotations: a red circle with '1' points to the line 'user2.dat'; another red circle with '2' points to the message 'El archivo 'user2.dat' no existe, no se mostrarán datos grabados.'; and a red circle with '3' points to the word 'MENÚ' in the menu options.

The bottom pane shows a terminal window titled 'Output - Antonio\_Naranjo\_AD1\_E1 (run)'. It displays the following text:

```
run:
Introduzca el nombre del archivo para grabar la lista de usuarios, pulsa ENTER para introducir por defecto:
user2.dat 1
El archivo 'user2.dat' no existe, no se mostrarán datos grabados. 2
--- 3
MENÚ
---
1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.

0. Salir de la aplicación
```

The terminal window has a status bar at the bottom showing 'Antonio\_Naranjo\_AD1\_E1 (run)', 'running...', '160:82/1:62', 'INS', '20:37', and '30/11/2025'.

En caso de no existir se indica que no existe, para ello, se establece un nombre no existente en el directorio raíz del proyecto. Luego, se muestra el menú del programa.

## 14.2. Agregar un usuario

The screenshot shows an IDE interface with several tabs at the top: ...va, Cesta.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M]. The Main.java tab is active, displaying Java code. The code includes a switch statement where option 1 leads to printing a success message and then prompting for user information (ID, password, address, birth year) which is then added to a list. The output window shows the program's execution, starting with a welcome message, then selecting option 1, followed by the user input for ID, password, address, and birth year, and finally confirming the user was added successfully. The status bar at the bottom indicates the application is running.

```
System.out.println("Lista de usuarios grabada en disco con éxito.");
break;
// Cargar la lista de usuarios desde el archivo (deserialización)
case 4:
mensaje = "Cargar la lista de usuarios desde el archivo (deserialización)";
System.out.println(String.format("%nLa opción elegida por el usuario es: %s", mensaje));
// Se llama al método estático de la clase Usuario cargarListaGrabada() para recuperar la lista de usuarios
}

AD01_2526_T01.Main > main > do ... while (!salidaPermitida.equals("S")) > switch (opcion) >
```

Output - Antonio\_Naranjo\_AD1\_E1 (run) > Sonar Rule Details Window

```
0. Salir de la aplicación.

----> Opción seleccionada por el usuario: 1

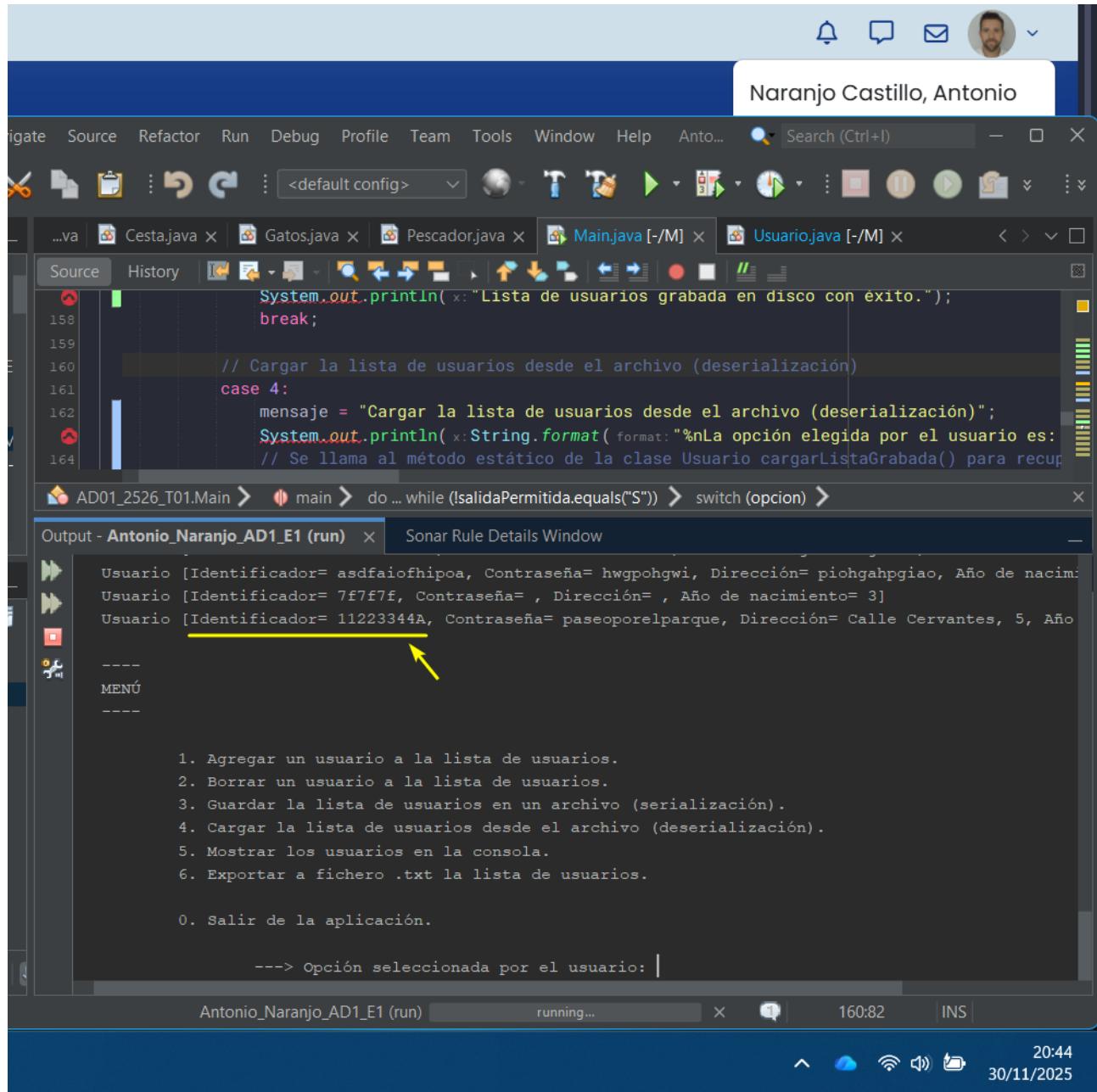
La opción elegida por el usuario es: 1 [Agregar un usuario a la lista de usuarios]
Introduzca el identificador del usuario
11223344A
Introduzca la contraseña del usuario
paseoporelparkue
Introduzca el dirección del usuario
Calle Cervantes, 5
Introduzca el año de nacimiento del usuario
1972
El usuario con identificador '11223344A' ha sido agregado con éxito

----> MENÚ
---->
```

Antonio\_Naranjo\_AD1\_E1 (run) running... 160:82 INS 20:40 30/11/2025

La opción 1 agrega un usuario previa solicitud de los atributos del objeto usuario. Acto seguido vuelve a mostrarse el menú.

### 14.3. Mostrar los usuarios en consola



The screenshot shows an IDE interface with the following details:

- Toolbar:** Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, Auto... (with a search bar), and various icons for file operations.
- Project Explorer:** Shows files like Cestajava.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M].
- Code Editor:** Displays Java code for Main.java. Line 164 contains the line: `System.out.println(x: "Lista de usuarios grabada en disco con éxito.");`
- Output Window:** Titled "Output - Antonio\_Naranjo\_AD1\_E1 (run)". It shows the output of the application:

```
Usuario [Identificador= asdfaiofhipoa, Contraseña= hwgpohgwi, Dirección= piohgahpgiao, Año de nacimiento= 2000]
Usuario [Identificador= 7f7f7f, Contraseña= , Dirección= , Año de nacimiento= 3]
Usuario [Identificador= 11223344A, Contraseña= paseoporelparque, Dirección= Calle Cervantes, 5, Año de nacimiento= 1990]
-----
MENÚ
-----
1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.

0. Salir de la aplicación.

---> Opción seleccionada por el usuario: |
```

A yellow arrow points to the last user entry in the list.
- System Tray:** Shows icons for battery, signal, and date/time (30/11/2025, 20:44).

Se selecciona la opción 5 para mostrar todos los usuarios observándose en último lugar el usuario recientemente añadido (a la lista temporal).

#### 14.4. Borrar usuario

The screenshot shows an IDE interface with several tabs open at the top: ...va, Cesta.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M]. The Main.java tab is active, displaying code related to user management. The code includes logic for saving users to disk and loading them from a file. A yellow arrow points to the identifier '11223344A' entered into the terminal window below.

Output - Antonio\_Naranjo\_AD1\_E1 (run) x Sonar Rule Details Window

```
Introduzca el identificador del usuario a eliminar
11223344A
El usuario con identificador '11223344A' ha sido borrado con éxito
```

----  
MENÚ  
----

1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.
  
0. Salir de la aplicación.

---> Opción seleccionada por el usuario: |

Antonio\_Naranjo\_AD1\_E1 (run) running... x 160:82 INS 20:46 30/11/2025

Se selecciona la opción 2 y se pasa como argumento la identificación del usuario, se aporta la identificación del usuario recientemente aportado.

The screenshot shows an IDE interface with several tabs at the top: Source, History, and a search bar. Below the tabs, there are icons for file operations like剪切 (Cut), 复制 (Copy), and 粘贴 (Paste). The main workspace displays Java code in a source editor. The code includes methods for printing user lists and handling user input. The output window below shows the execution of the application, displaying a list of users and a menu with numbered options. A yellow arrow points from the word 'Usuario' in the output to the corresponding line in the code.

```
System.out.println("Lista de usuarios grabada en disco con éxito.");
break;

// Cargar la lista de usuarios desde el archivo (deserialización)
case 4:
    mensaje = "Cargar la lista de usuarios desde el archivo (deserialización)";
    System.out.println(String.format("%nLa opción elegida por el usuario es:"));
    // Se llama al método estático de la clase Usuario cargarListaGrabada() para recuperar la lista de usuarios
}

AD01_2526_T01.Main > main > do ... while (!salidaPermitida.equals("S")) > switch (opcion) >
```

Output - Antonio\_Naranjo\_AD1\_E1 (run) > Sonar Rule Details Window

```
Usuario [Identificador= 22334wwtrw, Contraseña= wtwert2345, Dirección= gt243t3gfrwe, Año de nacimiento= 2000]
Usuario [Identificador= asdfaiofhipoa, Contraseña= hwgpohgw, Dirección= pioghahpgiao, Año de nacimiento= 2001]
Usuario [Identificador= 7f7f7f, Contraseña= , Dirección= , Año de nacimiento= 3]
```

MENÚ

1. Agregar un usuario a la lista de usuarios.  
2. Borrar un usuario a la lista de usuarios.  
3. Guardar la lista de usuarios en un archivo (serialización).  
4. Cargar la lista de usuarios desde el archivo (deserialización).  
5. Mostrar los usuarios en la consola.  
6. Exportar a fichero .txt la lista de usuarios.  
0. Salir de la aplicación.

---> Opción seleccionada por el usuario: |

Se comprueba que el usuario ha sido eliminado con éxito volviendo a mostrar la lista de usuarios y observando que en último lugar ya no aparece el usuario anterior.

## 14.5. Guardar lista (Serialización)

The screenshot shows an IDE interface with several tabs at the top: Cesta.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M]. The Main.java tab is active, displaying the following code:

```
System.out.println("Lista de usuarios grabada en disco con éxito.");
break;
// Cargar la lista de usuarios desde el archivo (deserialización)
case 4:
    mensaje = "Cargar la lista de usuarios desde el archivo (deserialización)";
    System.out.println(String.format("%nLa opción elegida por el usuario es:"));
    // Se llama al método estático de la clase Usuario cargarListaGrabada() para recuperar la lista de usuarios.
```

The terminal output window shows the following text:

```
La opción elegida por el usuario es: 3 [Guardar la lista de usuarios en un archivo (serialización)]
Lista de usuarios grabada en disco con éxito. ← (Yellow arrow points here)

-----
MENÚ
-----

1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.

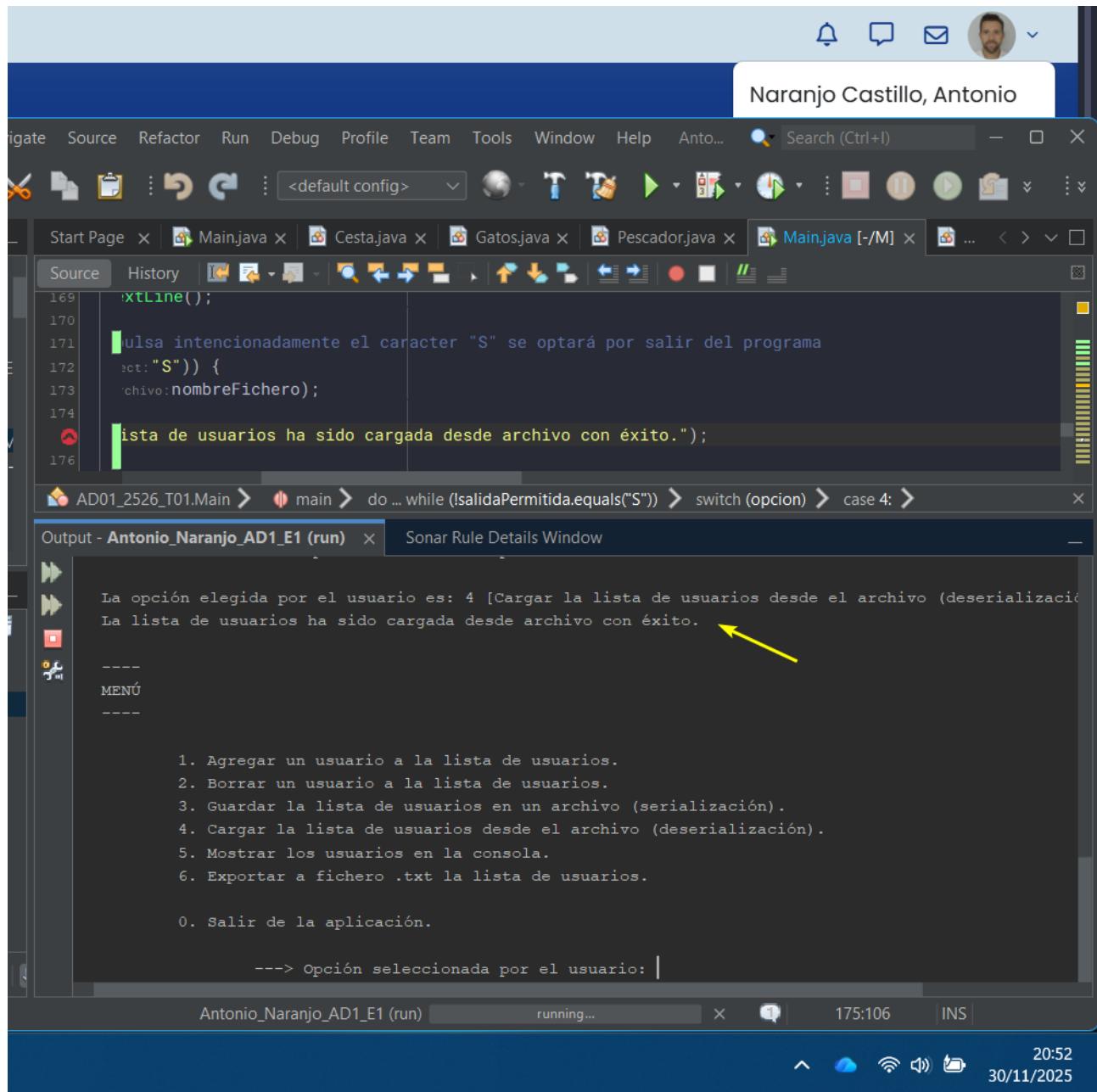
0. Salir de la aplicación.

---> Opción seleccionada por el usuario: |
```

The status bar at the bottom indicates the application is running, the time is 16:08, and the date is 30/11/2025.

Se selecciona la opción 3 del menú y se muestra un mensaje tras el guardado satisfactorio.

## 14.6. Cargar lista (deserialización)



The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name "Naranjo Castillo, Antonio".
- Toolbar:** Includes standard icons for Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... A search bar is also present.
- Project Explorer:** Lists files like Main.java, Cestajava, Gatosjava, Pescadorjava, and Main.java [-/M].
- Code Editor:** Displays Java code for handling user input and loading a user list from a file.
- Output Window:** Shows the application's console output:

```
La opción elegida por el usuario es: 4 [Cargar la lista de usuarios desde el archivo (deserialización)].  
La lista de usuarios ha sido cargada desde archivo con éxito.
```

A yellow arrow points from the text "La lista de usuarios ha sido cargada desde archivo con éxito." to the right.
- Bottom Status Bar:** Shows the run configuration "Antonio\_Naranjo\_AD1\_E1 (run)", status "running...", time "175:106", and date "30/11/2025".

Se selecciona la opción 4 del menú y se muestra un mensaje tras la carga satisfactoria.

## 14.7. Exportación archivo TXT

The screenshot shows an IDE interface with several tabs open, including 'Main.java' and 'Cestajava'. The code in Main.java includes logic for handling user input and exporting data to a file. The terminal window displays the execution of the program, showing the selection of option 6 (Exportar a fichero .txt la lista de usuarios) and the successful export of the user list. A yellow arrow points to the message 'Archivo de texto exportado correctamente.'

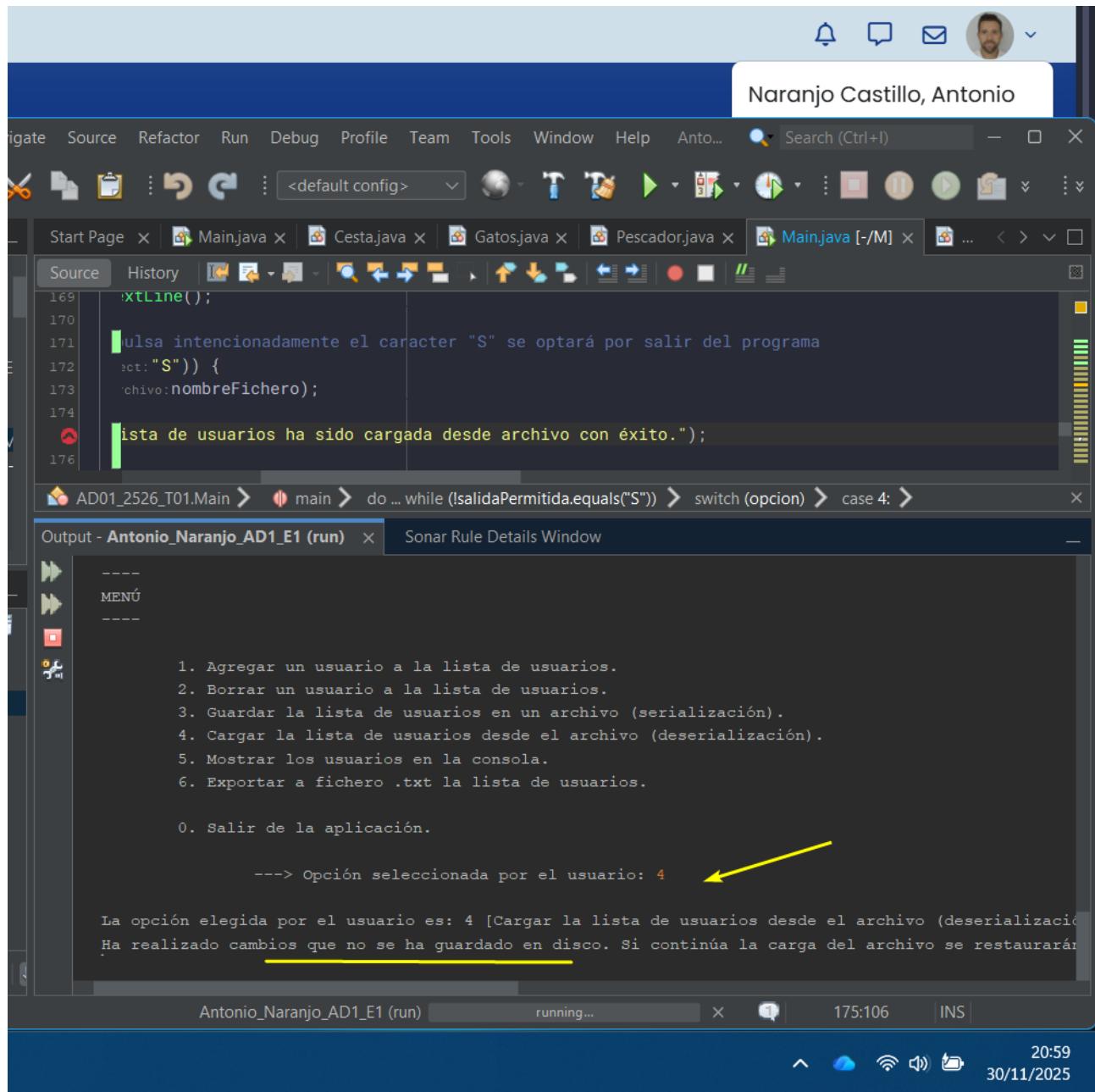
```
169     System.out.println();
170     System.out.println("Pulsa intencionadamente el caracter \"S\" se optará por salir del programa");
171     System.out.println("olect: " + s);
172     System.out.println("chivo: nombreFichero");
173
174     System.out.println("ista de usuarios ha sido cargada desde archivo con éxito.");
175
176 }
```

```
----> Opción seleccionada por el usuario: 6
La opción elegida por el usuario es: 6 [Exportar a fichero .txt la lista de usuarios]
Introduzca el nombre del archivo de texto txt para exportar la lista de usuarios, pulsa ENTER para :
Archivo de texto exportado correctamente. →
----<
MENÚ
----<

1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.
```

Se selecciona la opción 6 para proceder con la exportación del fichero de texto, se lanza un mensaje si se ha llevado a cabo satisfactoriamente y se vuelve a mostrar el menú.

## 14.8. ¿Cargar datos guardados?



The screenshot shows an IDE interface with the following details:

- Toolbar:** Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, Auto... (with a search bar labeled "Search (Ctrl+I)"), and various icons for file operations.
- Project Bar:** Shows multiple Java files: Main.java, Cestajava, Gatosjava, Pescadorjava, and another Main.java.
- Code Editor:** Displays Java code for a menu system. Line 175 contains a switch statement where if the input is "S", it exits the program. Line 176 handles the case where the user wants to load users from a file.
- Output Window:** Titled "Output - Antonio\_Naranjo\_AD1\_E1 (run)", it shows the application's menu options and the user's selection. The output text includes:
  - MENÚ
  - 1. Agregar un usuario a la lista de usuarios.
  - 2. Borrar un usuario a la lista de usuarios.
  - 3. Guardar la lista de usuarios en un archivo (serialización).
  - 4. Cargar la lista de usuarios desde el archivo (deserialización).
  - 5. Mostrar los usuarios en la consola.
  - 6. Exportar a fichero .txt la lista de usuarios.
  - 0. Salir de la aplicación.

----> Opción seleccionada por el usuario: 4

La opción elegida por el usuario es: 4 [Cargar la lista de usuarios desde el archivo (deserialización).  
Ha realizado cambios que no se ha guardado en disco. Si continúa la carga del archivo se restaurarán]
- Bottom Status:** Shows the run configuration ("Antonio\_Naranjo\_AD1\_E1 (run)"), status ("running..."), timestamp ("175:106"), and mode ("INS"). It also shows system icons for battery, signal, and time ("20:59 30/11/2025").

En caso de que el usuario pretenda cargar los datos guardados en el fichero binario, primeramente, el programa comparará la lista temporal de usuarios con la lista deserializada, de tal manera que si existen cambios de la opción de volver atrás y pueda guardar cambios o sustituya los valores si es finalmente la intención del usuario.

A modo de ejemplo se creó un usuario nuevo para alterar la lista temporal y así hacerla diferente a la deserializada.

The screenshot shows a Java application running in an IDE. The code in Main.java handles user input for exiting the program:

```
169    'xtLine();
170
171    pulsado = teclado.readLine();
172    if (pulsado.equalsIgnoreCase("S")) {
173        nombreFichero = "C:/Users/ASUS/Desktop/USUARIOS.txt";
174
175        listaDeUsuarios = cargarListaDeUsuarios(nombreFichero);
176
177        System.out.println("La lista de usuarios ha sido cargada desde archivo con éxito.");
178    }
179}
```

The terminal output shows the application's behavior:

```
Ha realizado cambios que no se ha guardado en disco. Si continúa la carga del archivo se restaurará:  
S  
La lista de usuarios ha sido cargada desde archivo con éxito.
```

Yellow arrows point to the character 'S' in the first line of the output and to the end of the second line.

The application menu includes:

- 1. Agregar un usuario a la lista de usuarios.
- 2. Borrar un usuario a la lista de usuarios.
- 3. Guardar la lista de usuarios en un archivo (serialización).
- 4. Cargar la lista de usuarios desde el archivo (deserialización).
- 5. Mostrar los usuarios en la consola.
- 6. Exportar a fichero .txt la lista de usuarios.
- 0. Salir de la aplicación.

The status bar at the bottom indicates the application is running and shows the time as 175:106.

Solo si el usuario pulsa “S” serán cuando se sustituyan los valores, de esta manera se elimina la posibilidad de pulsar cualquier otra letra accidentalmente.

## 14.9. Salida del programa

The screenshot shows an IDE interface with several tabs at the top: Start Page, Main.java, Cestajava, Gatosjava, Pescadorjava, and Main.java [-/M]. The Main.java tab is active, displaying Java code. The code includes a switch statement where option 0 leads to a println statement that outputs "La lista de usuarios ha sido cargada desde archivo con éxito.". Below the code editor is a terminal window titled "Output - Antonio\_Naranjo\_AD1\_E1 (run)". The terminal shows a menu of options from 1 to 6, followed by option 0 which exits the application. It then prompts for a selection and receives "0". A message indicates that option 0 was selected (Salir de la aplicación). It also warns about unsaved changes and asks if the user wants to save them. The user types "S" to save. The terminal concludes with "BUILD SUCCESSFUL (total time: 55 seconds)".

```
169     tos = teclado.nextLine();
170
171     // Si el usuario pulsa intencionadamente el carácter "S" se optará por salir del programa
172     if (tos.equals("S")) {
173         guardarLista(nombreArchivo);
174
175         println("La lista de usuarios ha sido cargada desde archivo con éxito.");
176     }
177 }
```

```
----  
1. Agregar un usuario a la lista de usuarios.  
2. Borrar un usuario a la lista de usuarios.  
3. Guardar la lista de usuarios en un archivo (serialización).  
4. Cargar la lista de usuarios desde el archivo (deserialización).  
5. Mostrar los usuarios en la consola.  
6. Exportar a fichero .txt la lista de usuarios.  
  
0. Salir de la aplicación.  
  
----> Opción seleccionada por el usuario: 0  
  
La opción elegida por el usuario es: 0 [Salir de la aplicación]  
Ha habido cambios en el programa que todavía no se han guardado. Si desea guardarlos ejecute la opción S  
BUILD SUCCESSFUL (total time: 55 seconds)
```

Al salir del programa, opción 0 del menú, si existen datos sin guardar se le mostrará al usuario un mensaje de salida advirtiendo de ello, solo si el usuario pulsa sobre el carácter “S” saldrá del programa.