

## ***Unidad 1***

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

#### **Tarea AD01**

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

1.	Crear una clase Usuario.....	2
2.	Implementación del menú .....	5
3.	Agregar un usuario a la lista de usuarios.....	6
4.	Borrar un usuario de la lista de usuarios.....	9
5.	Guardar la lista de usuarios en un archivo (serialización) .....	11
6.	Cargar la lista de usuarios desde el archivo (deserialización) .....	13
7.	Mostrar los usuarios en consola .....	17
8.	Exporta a fichero .txt la lista de usuarios.....	19
9.	Salir de la aplicación .....	21
10.	¿Existe el archivo ‘user.dat’? .....	22
11.	Advertencia sobre datos no guardados antes de salir .....	23
12.	¿Recuperar datos de disco? .....	24
13.	Resultados del programa en consola .....	25
13.1.	Inicio del programa.....	25
13.2.	Agregar un usuario .....	27
13.3.	Mostrar los usuarios en consola.....	28
13.4.	Borrar usuario .....	29
13.5.	Guardar lista (Serialización) .....	31
13.6.	Cargar lista (deserialización) .....	32
13.7.	Exportación archivo TXT .....	33
13.8.	¿Cargar datos guardados? .....	34
13.9.	Salida del programa .....	36

## 1. Crear una clase Usuario

The screenshot shows an IDE interface with the following details:

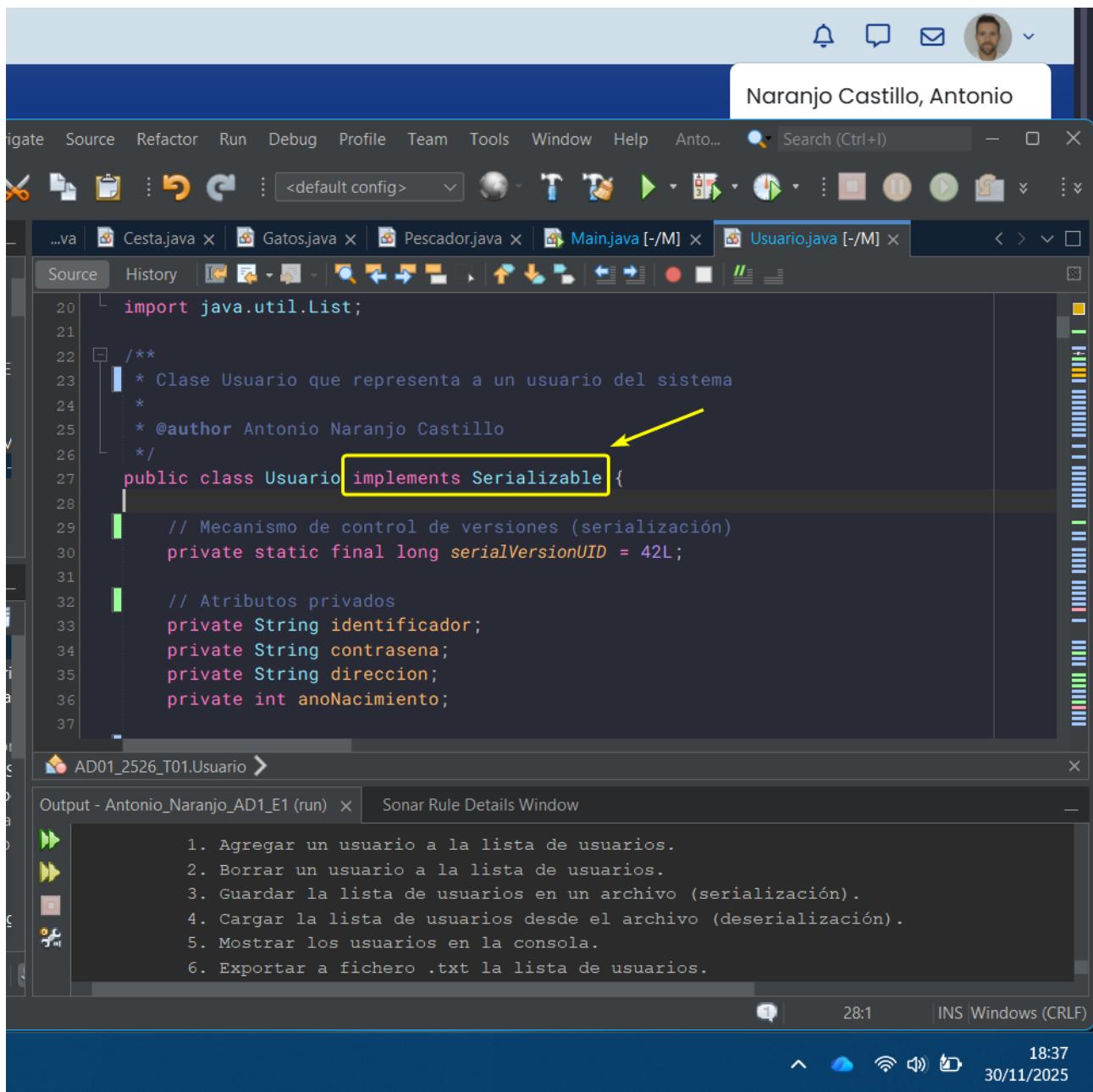
- Title Bar:** Shows the user's name "Naranjo Castillo, Antonio".
- Toolbar:** Includes standard icons for file operations like剪切 (Cut), 复制 (Copy), and 粘贴 (Paste).
- MenuBar:** Contains options like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... A search bar is also present.
- Project Explorer:** Shows files like Cestajava.java, Gatos.java, Pescador.java, Main.java, and Usuario.java.
- Code Editor:** Displays the code for the Usuario.java class. The constructor is shown with Javadoc comments and parameter annotations.
- 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.
- Status Bar:** Shows the time as 259:88 and the date as 30/11/2025.

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:

- Top Bar:** Shows navigation options like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... along with a search bar labeled "Search (Ctrl+I)".
- User Profile:** Displays the name "Naranjo Castillo, Antonio" and a small profile picture.
- Toolbar:** Includes icons for file operations (New, Open, Save, Print), code navigation (Find, Replace, Go To), and other development tools.
- Project Explorer:** Shows files like ...va, Cestajava.x, Gatos.java, Pescador.java, Main.java, and Usuario.java.
- Code Editor:** Displays the Java code for the `Usuario.java` class. The code includes methods for getting and setting the birth year, ID, and password, each annotated with Javadoc comments.
- Output Window:** Titled "AD01\_2526\_T01.Usuario > Output - Antonio\_Naranjo\_AD1\_E1 (run) x", it lists the following 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 system status, with the date and time displayed as 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.



```
20 import java.util.List;
21
22 /**
23  * Clase Usuario que representa a un usuario del sistema
24  *
25  * @author Antonio Naranjo Castillo
26 */
27 public class Usuario implements Serializable {implements Serializable
28
29     // Mecanismo de control de versiones (serialización)
30     private static final long serialVersionUID = 42L;
31
32     // Atributos privados
33     private String identificador;
34     private String contrasena;
35     private String direccion;
36     private int anoNacimiento;
37 }
```

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

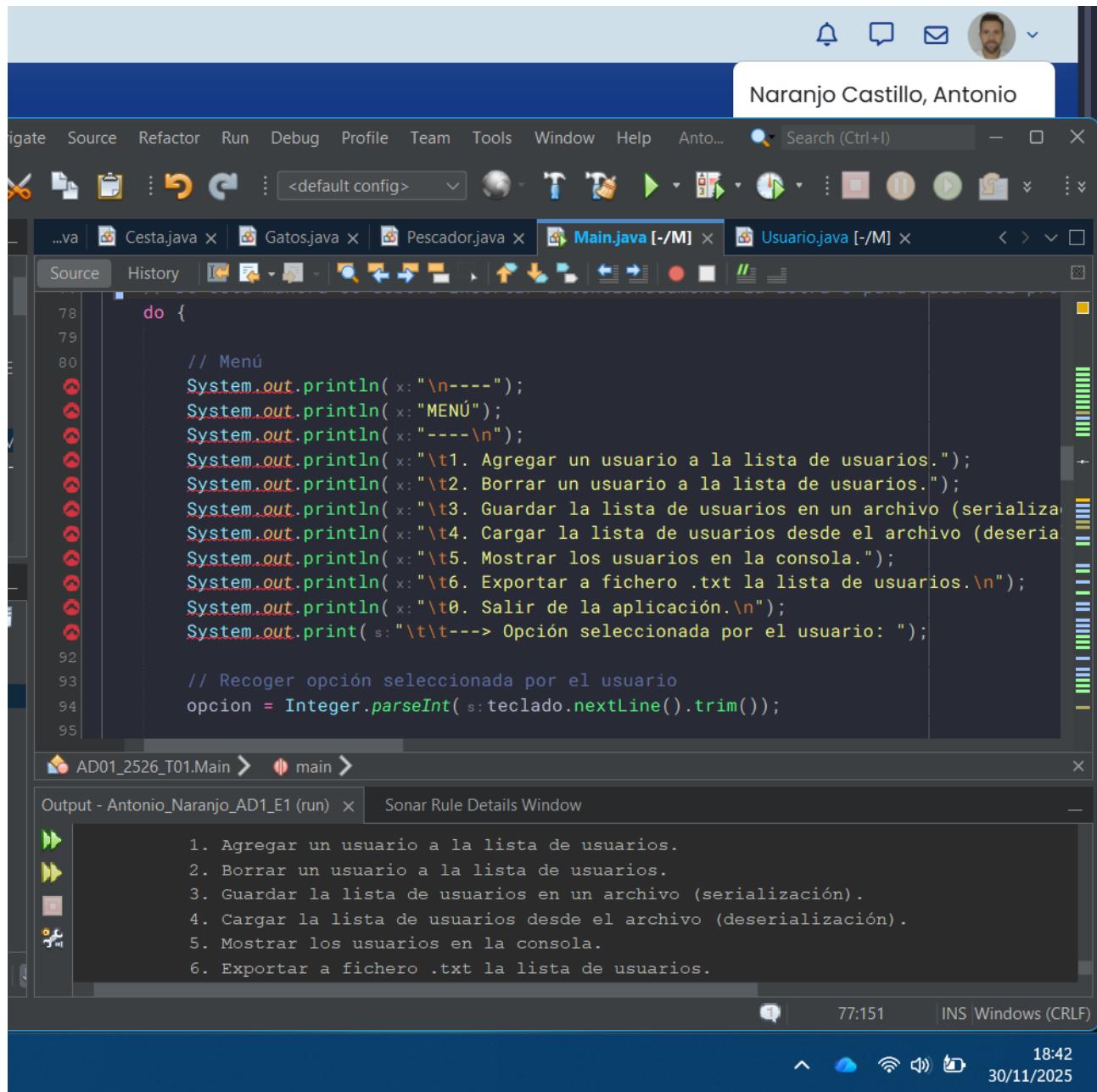
1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.

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.

## 2. Implementación 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 file operations like Open, Save, and Print.
- MenuBar:** Lists "Navigate", "Source", "Refactor", "Run", "Debug", "Profile", "Team", "Tools", "Window", "Help", and "Anto...".
- Search Bar:** A search field with placeholder text "<default config>" and a search icon.
- Project Explorer:** Shows files: "...va", "Cestajava.java", "Gatos.java", "Pescador.java", "Main.java [-/M]", and "Usuario.java [-/M]".
- Code Editor:** Displays the code for "Main.java" which implements a menu loop using System.out.println statements to print the menu options and System.in.read to get user input.
- Output Window:** Titled "Output - Antonio\_Naranjo\_AD1\_E1 (run)", it displays the menu options numbered 1 to 6.
- System Tray:** Shows the date and time as "30/11/2025 18:42".

```

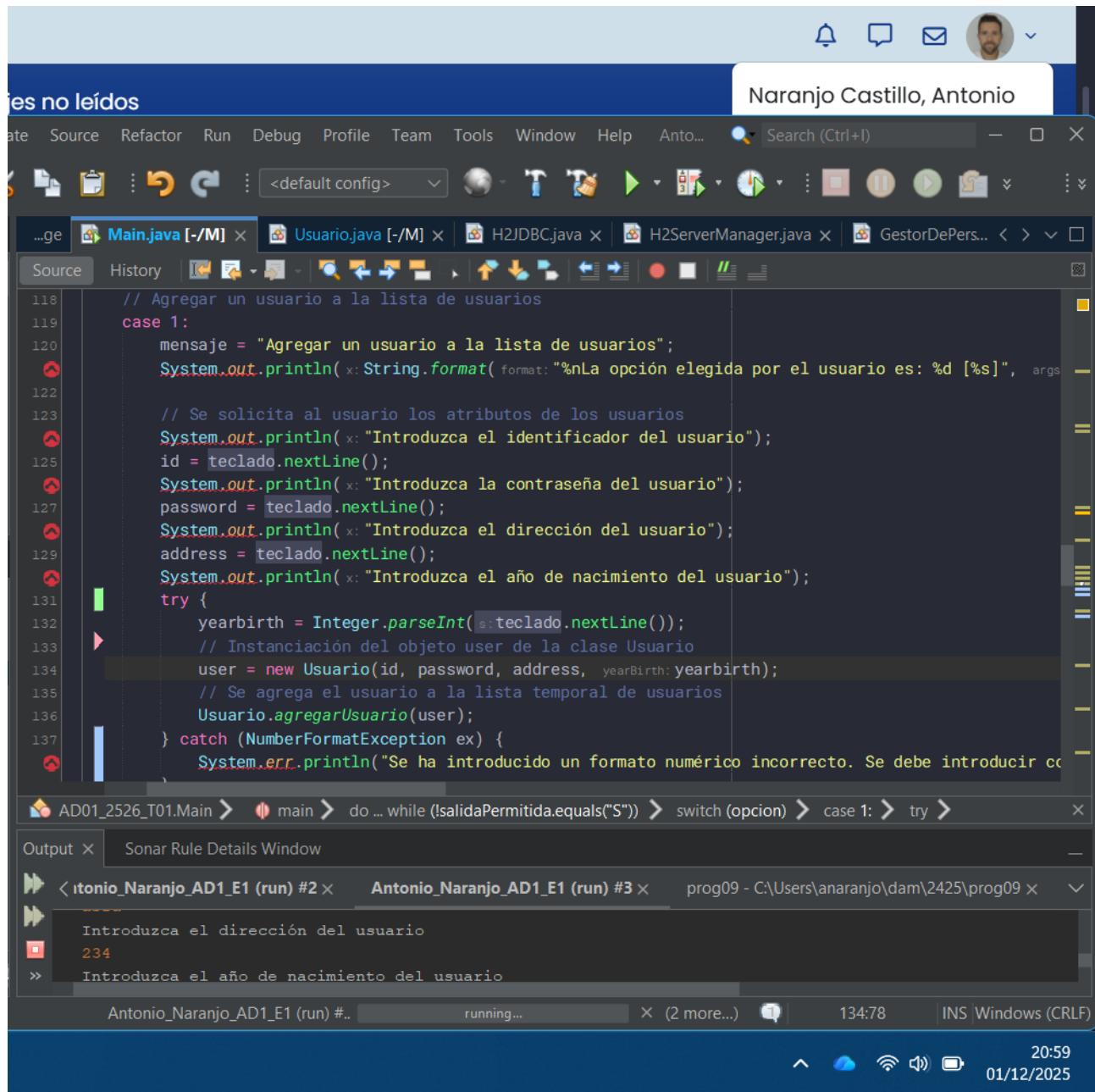
do {
    // Menú
    System.out.println( x: "\n----");
    System.out.println( x: "MENÚ");
    System.out.println( x: "----\n");
    System.out.println( x: "\t1. Agregar un usuario a la lista de usuarios.");
    System.out.println( x: "\t2. Borrar un usuario a la lista de usuarios.");
    System.out.println( x: "\t3. Guardar la lista de usuarios en un archivo (serialización).");
    System.out.println( x: "\t4. Cargar la lista de usuarios desde el archivo (deserialización).");
    System.out.println( x: "\t5. Mostrar los usuarios en la consola.");
    System.out.println( x: "\t6. Exportar a fichero .txt la lista de usuarios.\n");
    System.out.println( x: "\t0. Salir de la aplicación.\n");
    System.out.print( s: "\t\t--> Opción seleccionada por el usuario: ");

    // Recoger opción seleccionada por el usuario
    opcion = Integer.parseInt( s: teclado.nextLine().trim());
}

```

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.

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



The screenshot shows the Eclipse IDE interface with the following details:

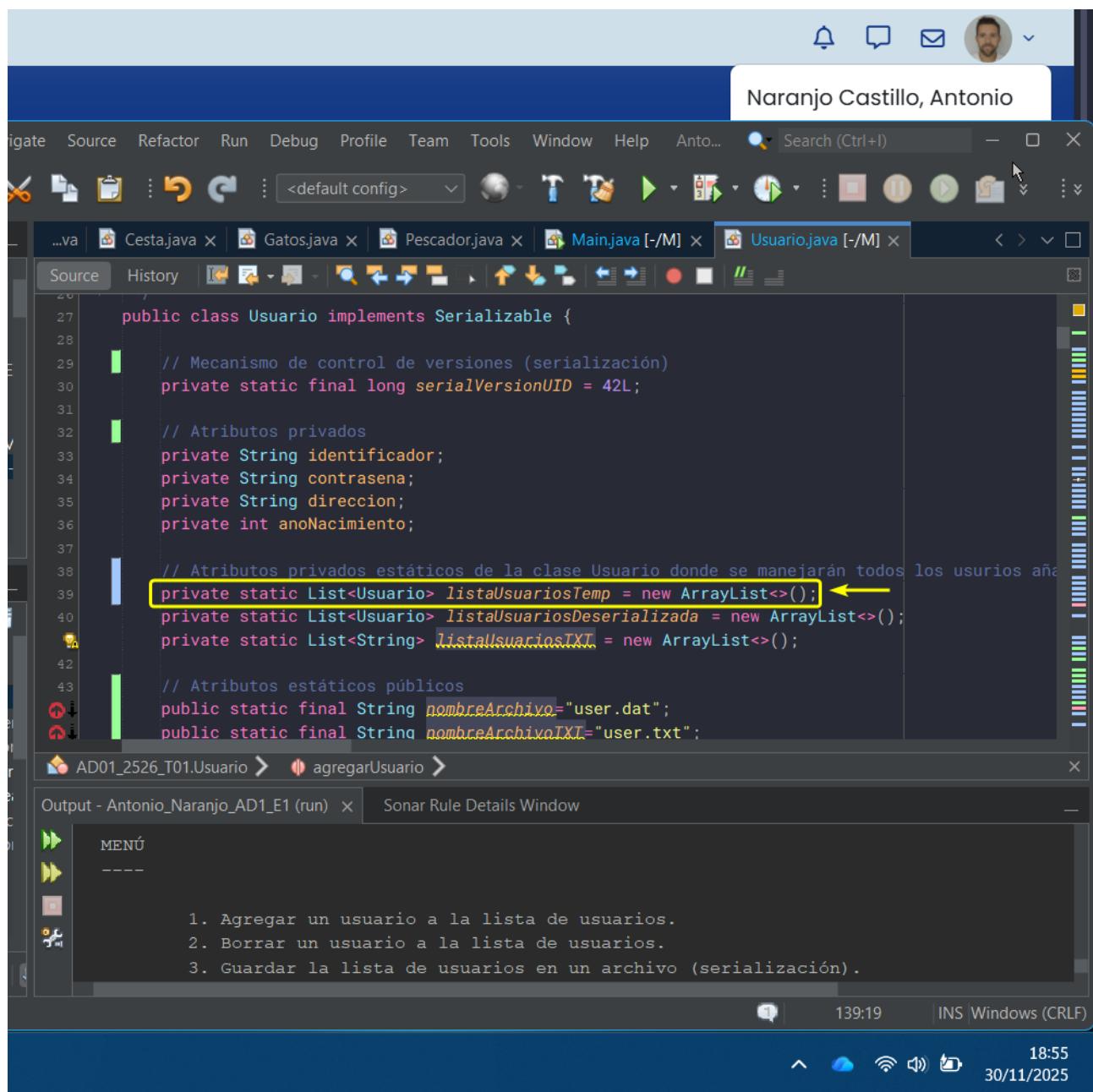
- Title Bar:** Shows "Naranjo Castillo, Antonio" as the current user.
- Toolbar:** Includes standard Eclipse icons for file operations, search, and help.
- ActionBar:** Displays tabs for "Main.java [-/M]", "Usuario.java [-/M]", "H2JDBC.java", "H2ServerManager.java", and "GestorDePers...".
- Source Editor:** The main window displays Java code for adding a user. The code includes a switch statement for option 1, which prints a message, reads user input for ID, password, address, and year of birth, creates a new User object, and adds it to a list. It also handles a NumberFormatException if the year of birth is not a valid integer.
- Output View:** Shows the execution path: AD01\_2526\_T01.Main > main > do ... while (IsalidaPermitida.equals("S")) > switch (opcion) > case 1: > try >. It also displays the console output:

```

Introduzca el identificador del usuario
234
Introduzca la contraseña del usuario
Introduzca el dirección del usuario
Introduzca el año de nacimiento del usuario
  
```

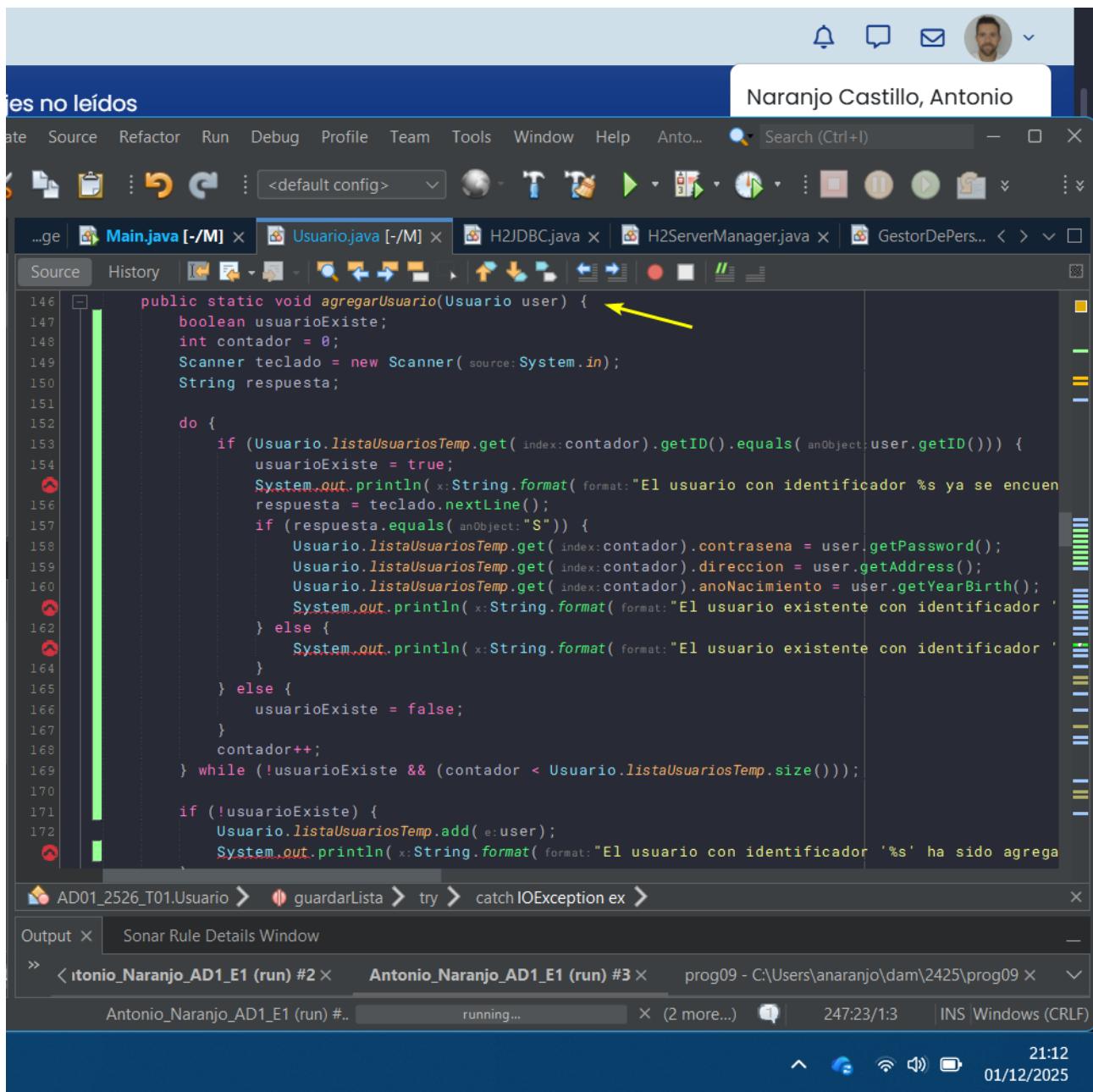
- Bottom Status Bar:** Shows the run configuration "Antonio\_Naranjo\_AD1\_E1 (run) #2", the status "running...", and the date/time "134:78 20:59 01/12/2025".

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 contrasena;  
    private String direccion;  
    private int anoNacimiento;  
  
    // 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";  
}
```

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 open 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 ).direccion = 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     }

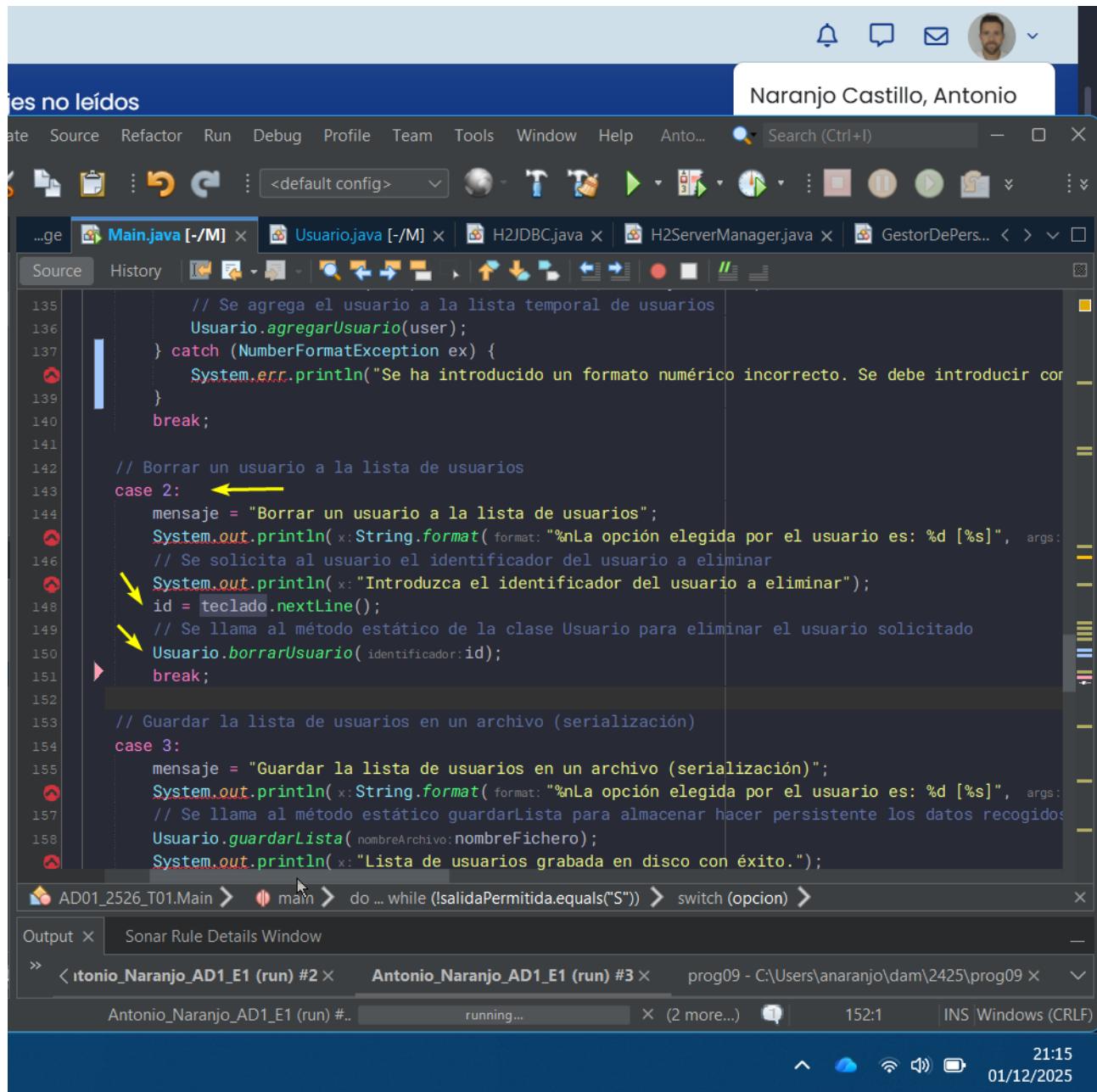
```

A yellow arrow points to the line `if ( Usuario.listaUsuariosTemp.get( index:contador ).getID().equals( anObject: user.getID() ) ) {`. The status bar at the bottom shows the path: AD01\_2526\_T01.Usuario > guardarLista > try > catch IOException ex >. The output tab shows the message "running...".

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.

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



```

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;

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

153 // Guardar la lista de usuarios en un archivo (serialización)
154 case 3:
155     mensaje = "Guardar la lista de usuarios en un archivo (serialización)";
156     System.out.println(x: String.format( format: "%nLa opción elegida por el usuario es: %d [%s]", args:
157     // Se llama al método estático guardarLista para almacenar hacer persistente los datos recogidos
158     Usuario.guardarLista( nombreArchivo:nombreFichero);
159     System.out.println(x: "Lista de usuarios grabada en disco con éxito.");

```

The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name: Naranjo Castillo, Antonio.
- Toolbar:** Standard Java development toolbar with icons for file operations, search, and navigation.
- Code Editor:** Displays Java code in Main.java. The code implements a switch statement with three cases: 1, 2, and 3. Case 2 is highlighted with a pink arrow pointing to its label. Inside case 2, there is a print statement asking for user input, another print statement prompting for the user ID, and a call to the static method Usuario.borrarUsuario() passing the ID as an argument. Two yellow arrows point to the print statements and the method call respectively.
- Output Window:** Shows the execution path: AD01\_2526\_T01.Main > main > do ... while (salidaPermitida.equals("S")) > switch (opcion) >. It also lists running tasks and the current run configuration: Antonio\_Naranjo\_AD1\_E1 (run) #2.
- System Tray:** Shows the date (01/12/2025), time (21:15), and battery status.

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:

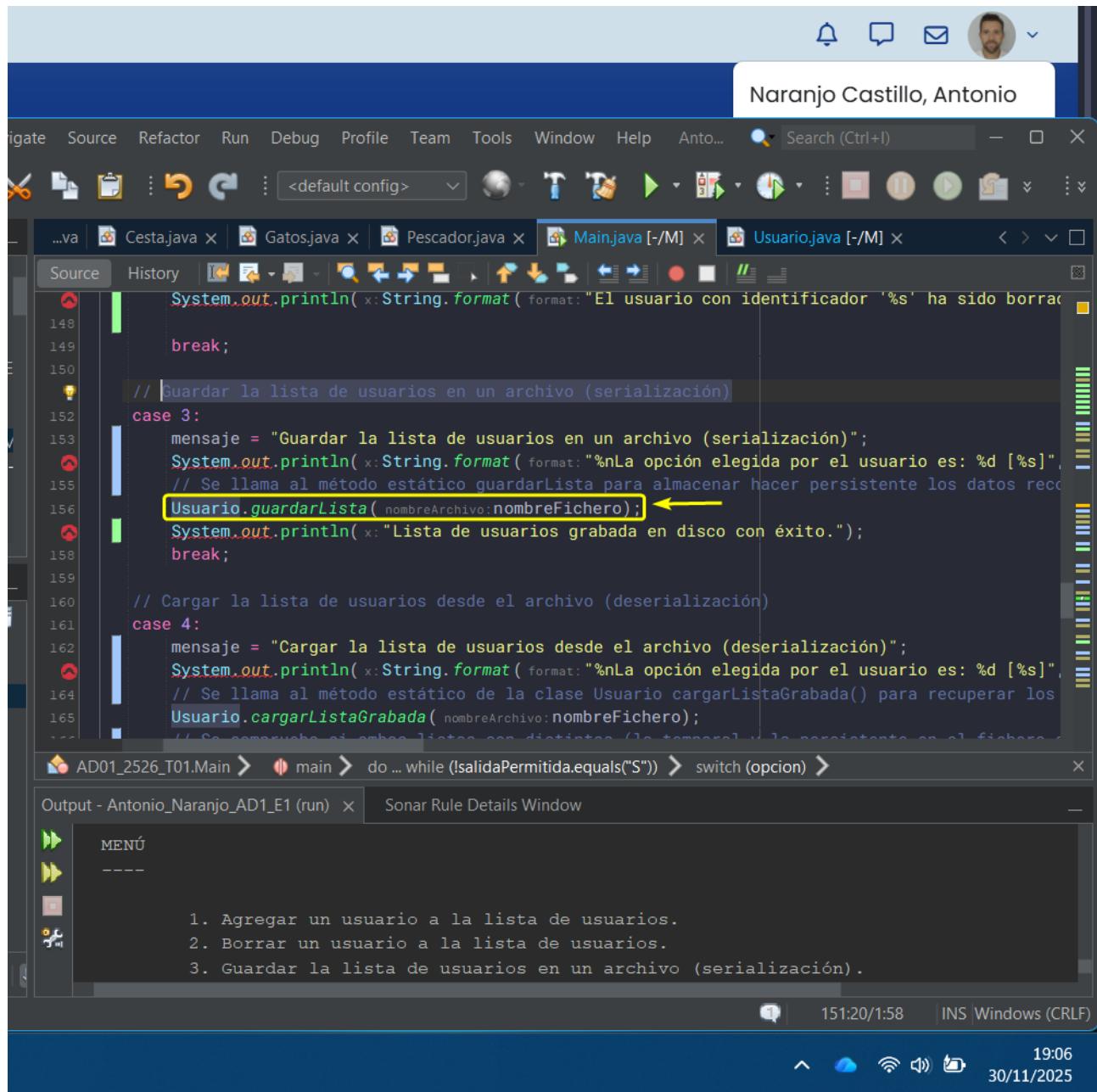
- Title Bar:** Shows "Naranjo Castillo, Antonio" in the top right corner.
- Toolbar:** Includes icons for file operations (New, Open, Save), search, and various project management tools.
- Project Explorer:** Shows files like Main.java, Usuario.java, H2JDBC.java, H2ServerManager.java, and GestorDePers...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

```
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 }
```

- Output Tab:** Shows the current run configuration: `Antonio_Naranjo_AD1_E1 (run) #2`.
- Bottom Status Bar:** Displays the time as 21:17 and the date as 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.

## 5. 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 files: ...va, Cestajava.x, Gatos.java.x, Pescador.java.x, Main.java [-/M] x, and Usuario.java [-/M] x.
- Code Editor:** Displays Java code. A specific line of code is highlighted: `Usuario.guardarLista(nombreArchivo:nombreFichero);`. An arrow points from the text 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 and time: 15/11/2025, 19:06.

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.

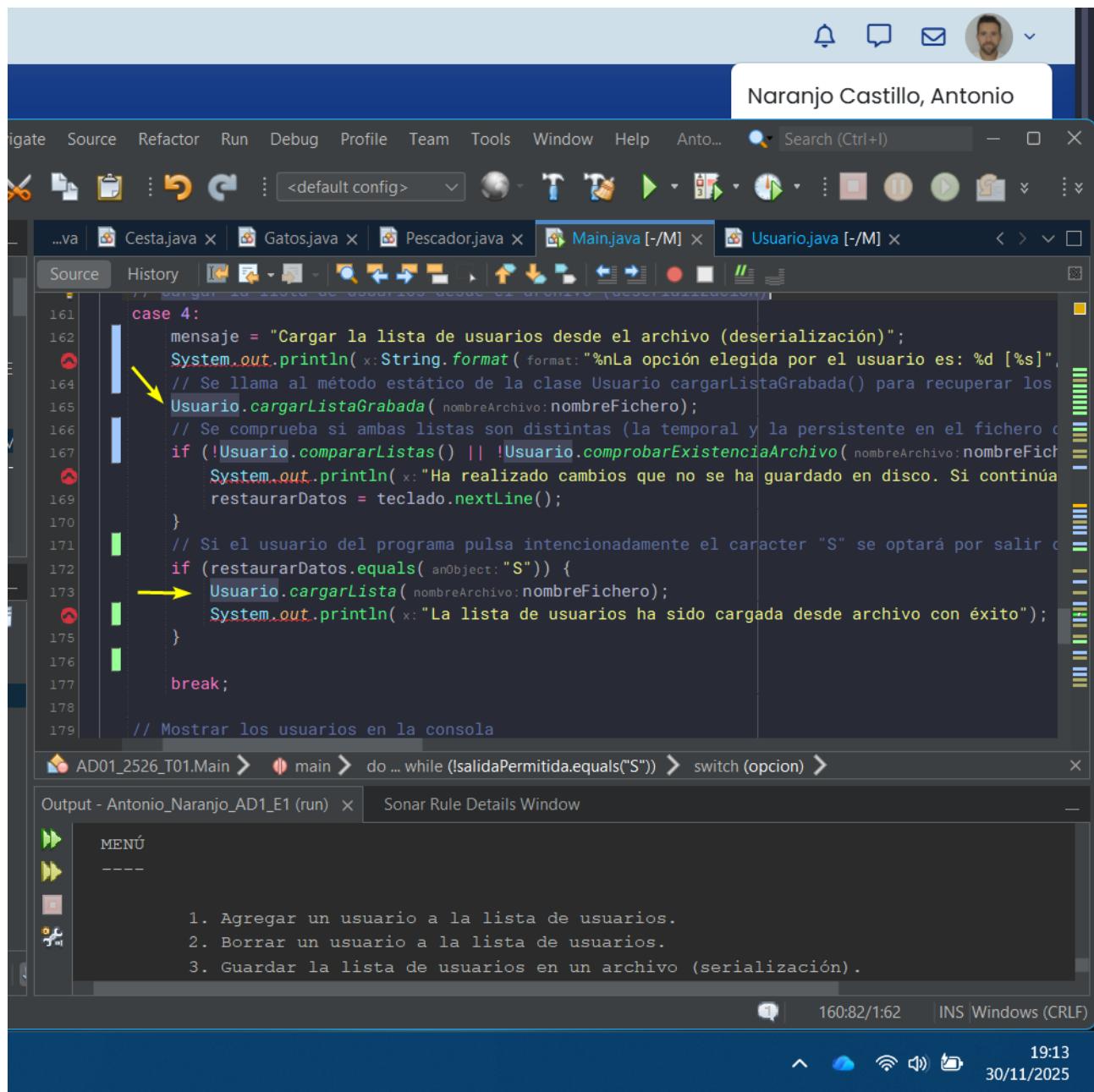
Se ejecuta el método estático de la clase Usuario `guardarLista()` aportando como argumento el nombre del archivo en cuestión.

The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the 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 ...va, Cestajava.x, Gatos.java, Pescador.java, Main.java, and Usuario.java.
- Code Editor:** Displays Java code for the `guardarLista` method in the `Usuario.java` class. The code uses `FileOutputStream` and  `ObjectOutputStream` to serialize a list of users.
- Output Window:** Shows a terminal-like window with the following menu and list:
  - 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, 19:08).

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).

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



```

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: %d [%s]");
164     // Se llama al método estático de la clase Usuario cargarListaGrabada() para recuperar los
165     // datos persistentes almacenados en el fichero
166     Usuario.cargarListaGrabada(nombreArchivo:nombreFichero);
167     // Se comprueba si ambas listas son distintas (la temporal y la persistente en el fichero)
168     if (!Usuario.compararListas() || !Usuario.comprobarExistenciaArchivo(nombreArchivo:nombreFichero))
169         System.out.println(x:"Ha realizado cambios que no se ha guardado en disco. Si continúa
170         restaurarDatos = teclado.nextLine();
171     }
172     // Si el usuario del programa pulsa intencionadamente el carácter "S" se optará por salir del
173     if (restaurarDatos.equals(anObject:"S"))
174         Usuario.cargarLista(nombreArchivo:nombreFichero);
175         System.out.println(x:"La lista de usuarios ha sido cargada desde archivo con éxito");
176     }
177
178     break;
179
180     // 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.

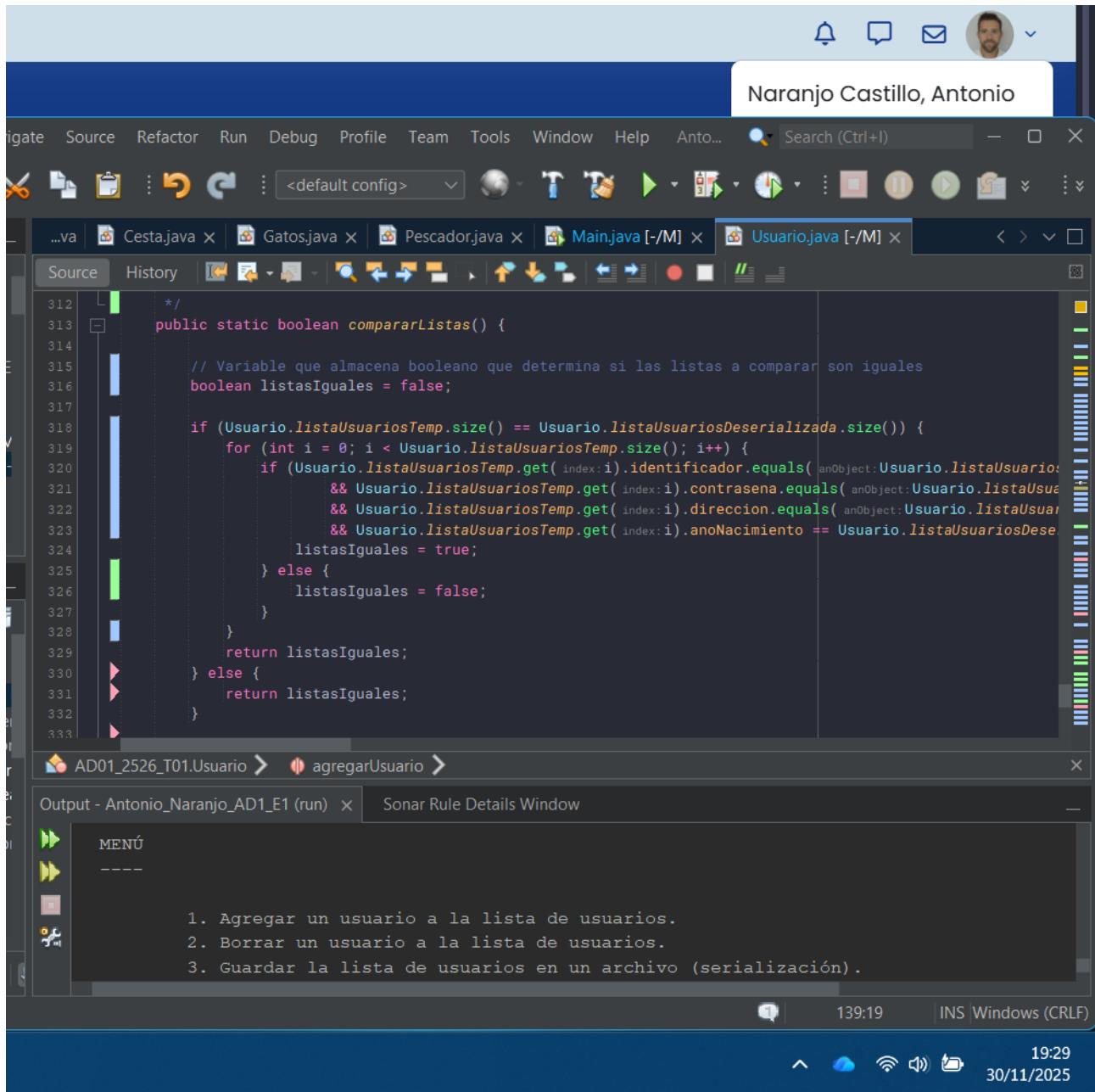
The screenshot shows an IDE interface with the following details:

- Top Bar:** Shows navigation tabs like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... along with a search bar labeled "Search (Ctrl+I)".
- Title Bar:** Displays the name "Naranjo Castillo, Antonio".
- Toolbar:** Includes icons for file operations like Open, Save, Cut, Copy, Paste, and various toolbars.
- Project Explorer:** Shows multiple Java files: ...va, Cestajava.x, Gatos.java, Pescador.java, Main.java, and Usuario.java.
- Code Editor:** Displays the Java code for the `cargarListaGrabada` method. The code uses `FileInputStream` to read bytes from a file and `ObjectInputStream` to deserialize them into a list of `Usuario` objects. It handles exceptions for file not found, I/O errors, and class not found.
- Output Window:** Shows the command line output for running the application, including 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).
```

The system tray at the bottom right shows the date (30/11/2025), time (19:24), battery level, signal strength, and network status.

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 a static method `compararListas()`. The code compares two lists of users based on their attributes: identificador, contraseña, dirección, and añoNacimiento.
- 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., and 3. Guardar la lista de usuarios en un archivo (serialización).
- System Tray:** Shows the date (30/11/2025), time (19:29), and battery status.

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 several tabs at the top: ...va, Cestajava x, Gatos.java x, Pescador.java x, Main.java [-/M] x, and Usuario.java [-/M] x. The main window displays Java code for the `cargarLista` method:

```
202     public static void cargarLista(String nombreArchivo) {  
203         try {  
204             // Obtener los bytes almacenados en el archivo  
205             FileInputStream fileIn = new FileInputStream( name:nombreArchivo);  
206             // Convertir los bytes en objetos  
207             ObjectInputStream in = new ObjectInputStream( in:fileIn)) {  
208                 // Se actualiza la lista temporal según datos almacenados en el fichero del disco duro  
209                 Usuario.listaUsuariosTemp = (List<Usuario>) in.readObject();  
210             }  
211             // Manejo de excepciones  
212             } catch (FileNotFoundException ex) {  
213                 System.err.println("Error: El archivo '" + nombreArchivo + "' no existe.");  
214             } catch (IOException ex) {  
215                 System.err.println( x: "Error de Entrada/Salida: Falló la lectura del fichero.");  
216             } catch (ClassNotFoundException ex) {  
217                 System.err.println( x: "Error de clase: No se pudo encontrar la clase Usuario.");  
218             }  
219         }  
220     }  
221 }  
222 }
```

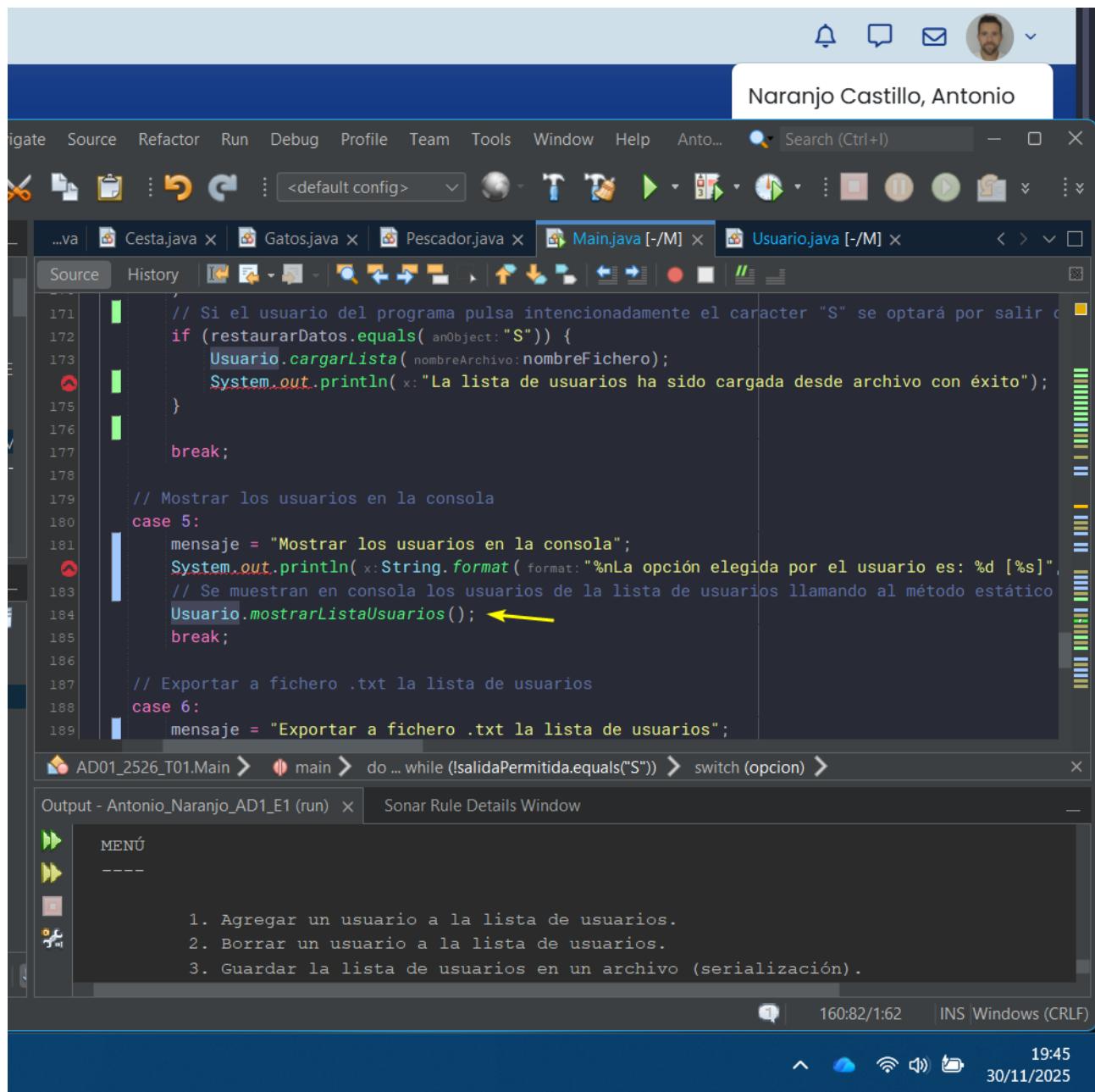
Below the code editor, there is a terminal window titled "AD01\_2526\_T01.Usuario > agregarUsuario >". It contains the following text:

```
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).
```

The status bar at the bottom right shows the time as 19:35 and the date as 30/11/2025.

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.

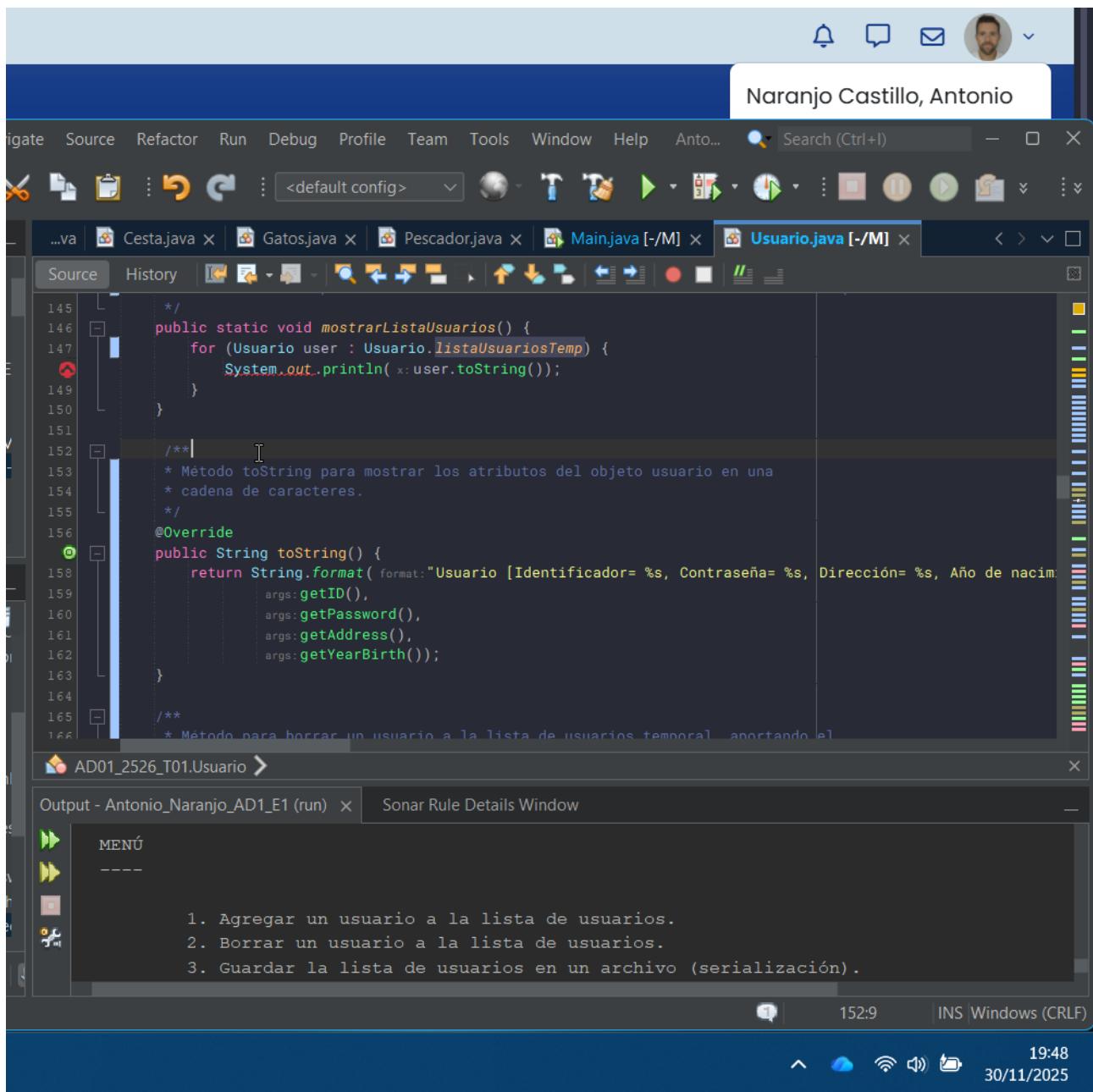
## 7. Mostrar los usuarios en consola



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 navigation.
- 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();`.
- 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.

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.



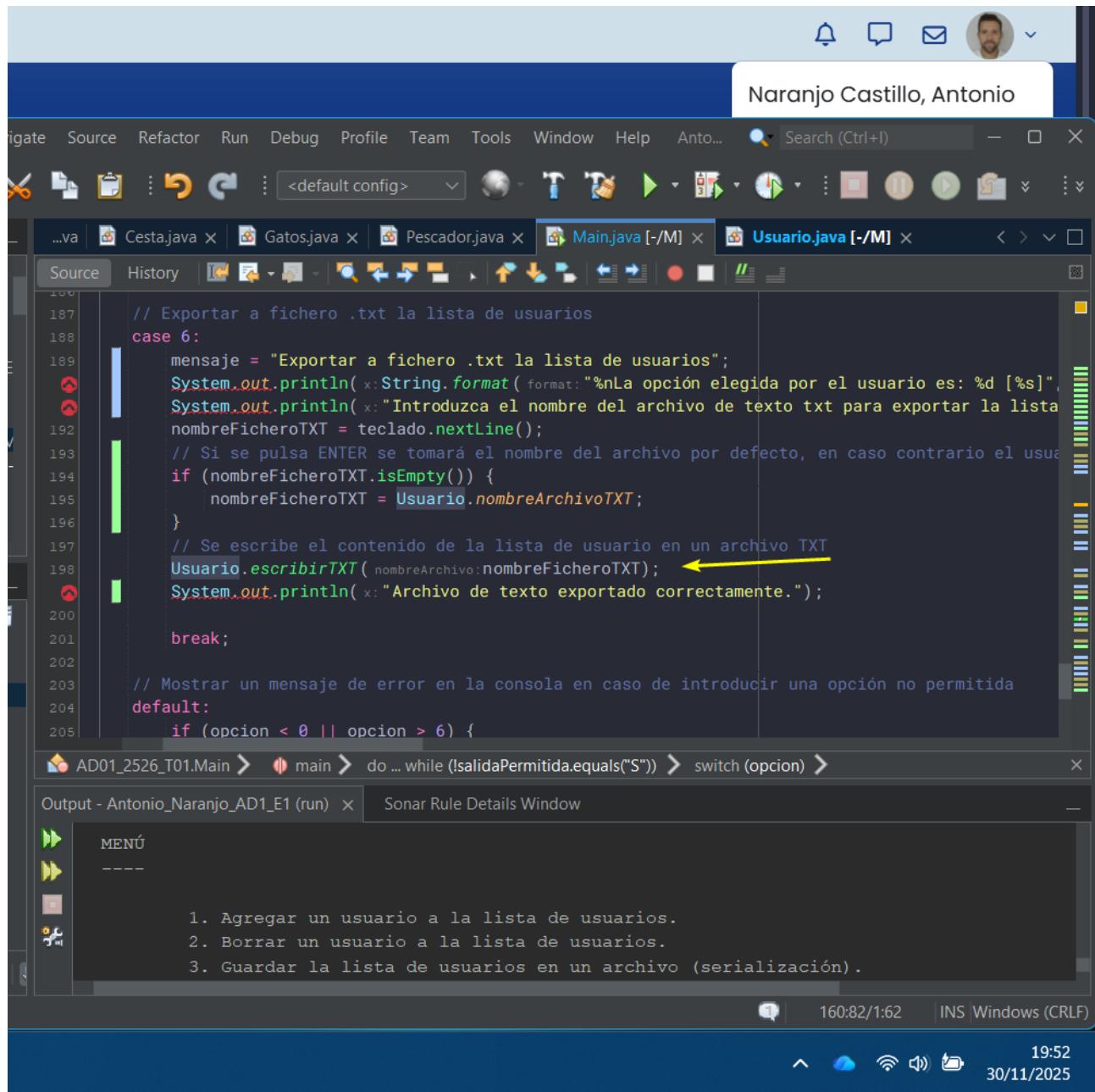
The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows "Naranjo Castillo, Antonio" in the top right corner.
- Toolbar:** Includes standard icons for file operations, navigation, and search.
- Project Bar:** Lists files: ...va, Cestajava.x, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M].
- Code Editor:** Displays the `Usuario.java` source code. The code includes methods `mostrarListaUsuarios()` and `toString()`. The `toString()` method uses `String.format` to return a string representing the user object's attributes.
- Output Window:** Shows a terminal window titled "AD01\_2526\_T01.Usuario" with the following 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 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.

## 8. 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 file operations, search, and navigation.
- Menu Bar:** Options like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... along with a search bar.
- Project Explorer:** Lists files: ...va, Cestajava.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M].
- Code Editor:** Displays Java code for the Usuario.java class. The code handles option 6, which exports the list of users to a file. A yellow arrow points to the line `Usuario.escribirTXT(nombreArchivo:nombreFicheroTXT);`.
- Output Window:** Shows the command line history and the output of the program. The output window displays a menu with options 1, 2, and 3, and the message "Archivo de texto exportado correctamente.".
- System Tray:** Shows the date and time (30/11/2025, 19:52), battery level, signal strength, and other system icons.

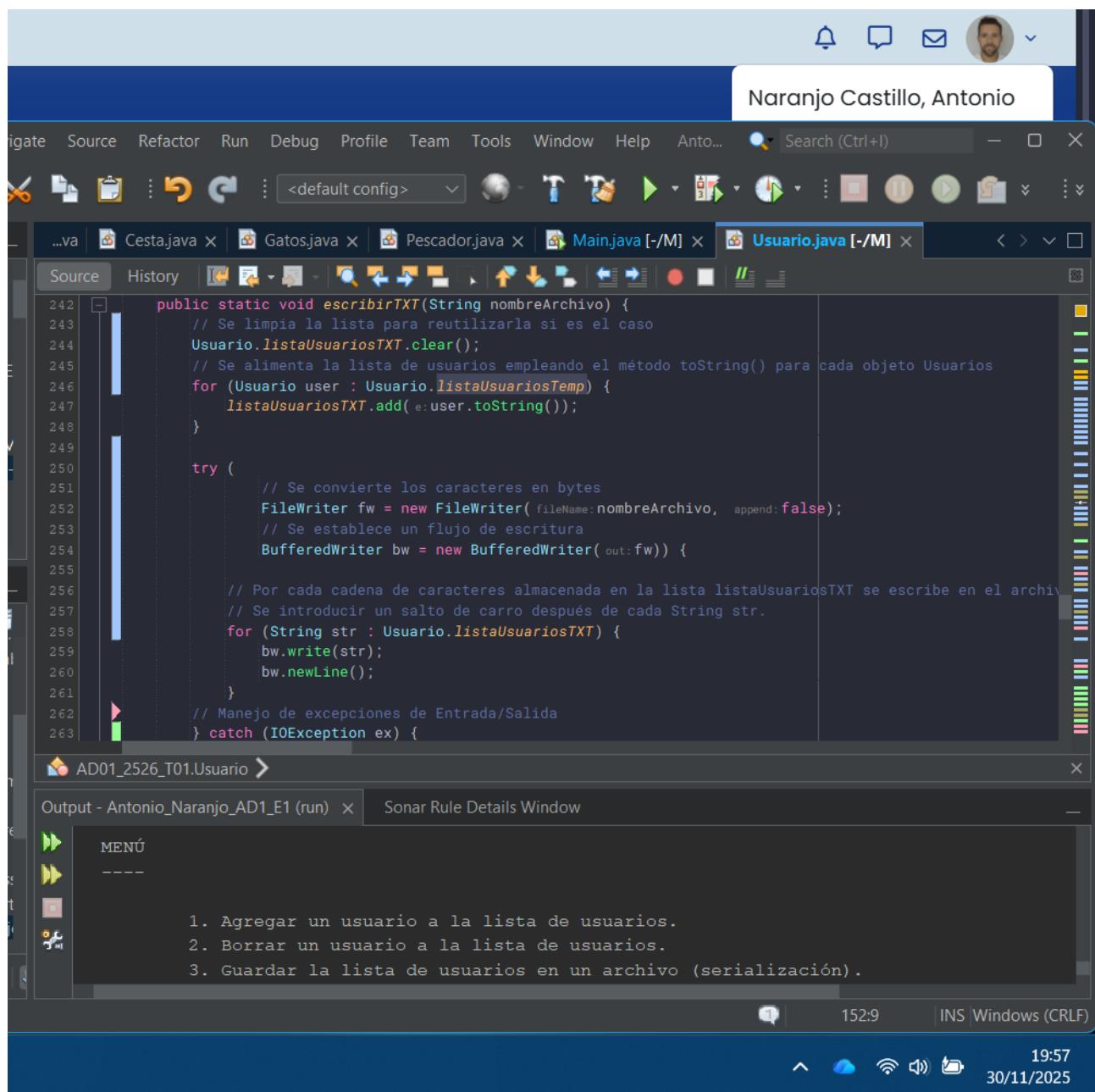
```

187 // Exportar a fichero .txt la lista de usuarios
188 case 6:
189     mensaje = "Exportar a fichero .txt la lista de usuarios";
190     System.out.println(x: String.format(format:"%nLa opción elegida por el usuario es: %d [%s]");
191     System.out.println(x:"Introduzca el nombre del archivo de texto txt para exportar la lista");
192     nombreFicheroTXT = teclado.nextLine();
193     // Si se pulsa ENTER se tomará el nombre del archivo por defecto, en caso contrario el usuario
194     if (nombreFicheroTXT.isEmpty()) {
195         nombreFicheroTXT = Usuario.nombreArchivoTXT;
196     }
197     // Se escribe el contenido de la lista de usuario en un archivo TXT
198     Usuario.escribirTXT(nombreArchivo:nombreFicheroTXT); ← Yellow arrow here
199     System.out.println(x:"Archivo de texto exportado correctamente.");
200
201     break;
202
203     // Mostrar un mensaje de error en la consola en caso de introducir una opción no permitida
204 default:
205     if (opcion < 0 || opcion > 6) {

```

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:

- 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 project files: ...va, Cestajava.x, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M].
- Code Editor:** The main window displays the `Usuario.java` source code. The code implements a static method `escribirTXT` that serializes a list of `Usuario` objects to a file. It uses `FileWriter` and `BufferedWriter` to handle the output stream, writing each object's string representation followed by a new line.
- Output Window:** Shows the output of the application named "AD01\_2526\_T01.Usuario". The menu "MENÚ" is open, displaying 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 as 30/11/2025 at 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 incialmente, 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 métod `newLine()`.

## 9. Salir de la aplicación

```

98
99     // Salir de la aplicación
100    case 0:
101        mensaje = "Salir de la aplicación";
102        System.out.println( 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( "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( "FIN del programa." ); 3
115        }
116        break;
117
118    // Agregar un usuario a la lista de usuarios

```

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

MENÚ

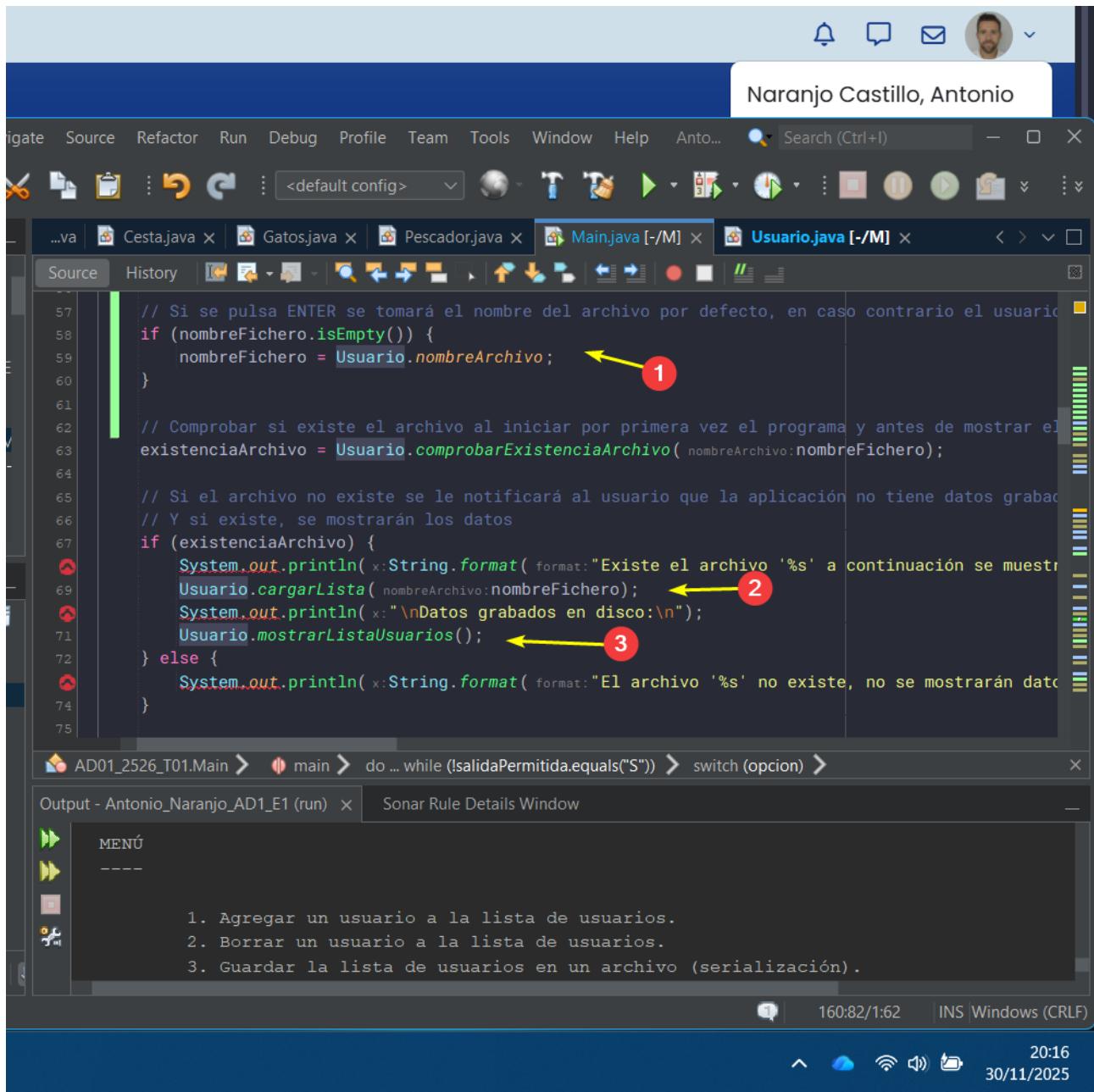
1. Agregar un usuario a la lista de usuarios.  
2. Borrar un usuario a la lista de usuarios.  
3. Guardar la lista de usuarios en un archivo (serialización).

160:82/1:62 INS Windows (CRLF) 20:07 30/11/2025

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.

## 10. ¿Existe el archivo ‘user.dat’?

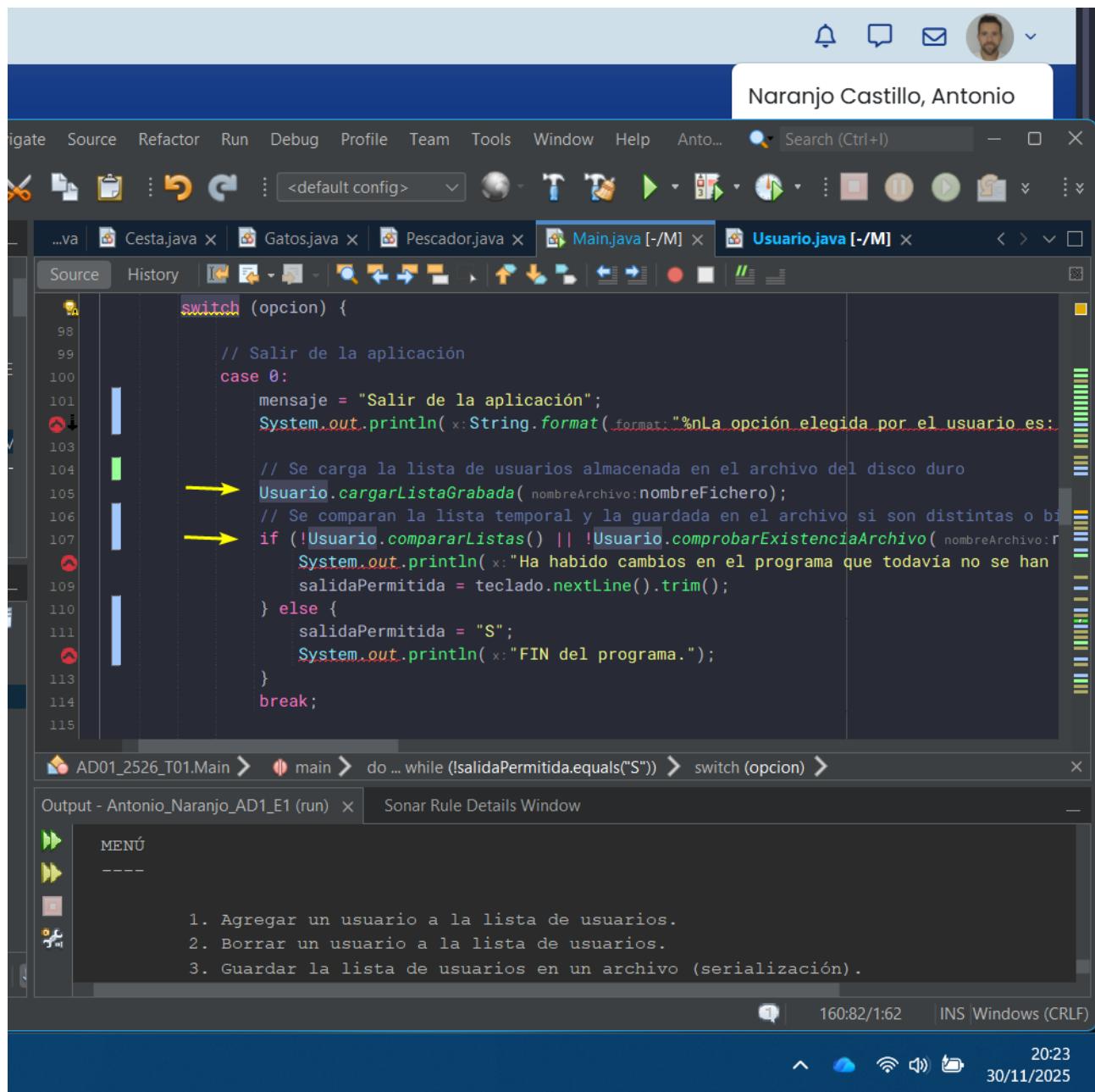


The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name: "Naranjo Castillo, Antonio".
- Toolbar:** Standard IDE tools like file operations, search, and navigation.
- Project Explorer:** Lists files: "...va", Cestajava.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M].
- Code Editor:** Displays Java code. Annotations with numbers 1, 2, and 3 point to specific lines:
  - Annotation 1 points to line 59: `nombreFichero = Usuario.nombreArchivo;`
  - Annotation 2 points to line 69: `Usuario.cargarLista(nombreArchivo:nombreFichero);`
  - Annotation 3 points to line 70: `Usuario.mostrarListaUsuarios();`
- Output Window:** Shows the execution path: AD01\_2526\_T01.Main > main > do ... while (!salidaPermitida.equals("S")) > switch (opcion) >. It also displays 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 Status:** Shows the system time as 160:82/1:62 and the date as 30/11/2025.

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.

## 11. 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: %s", mensaje));
        // 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;
}
```

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

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

MENÚ

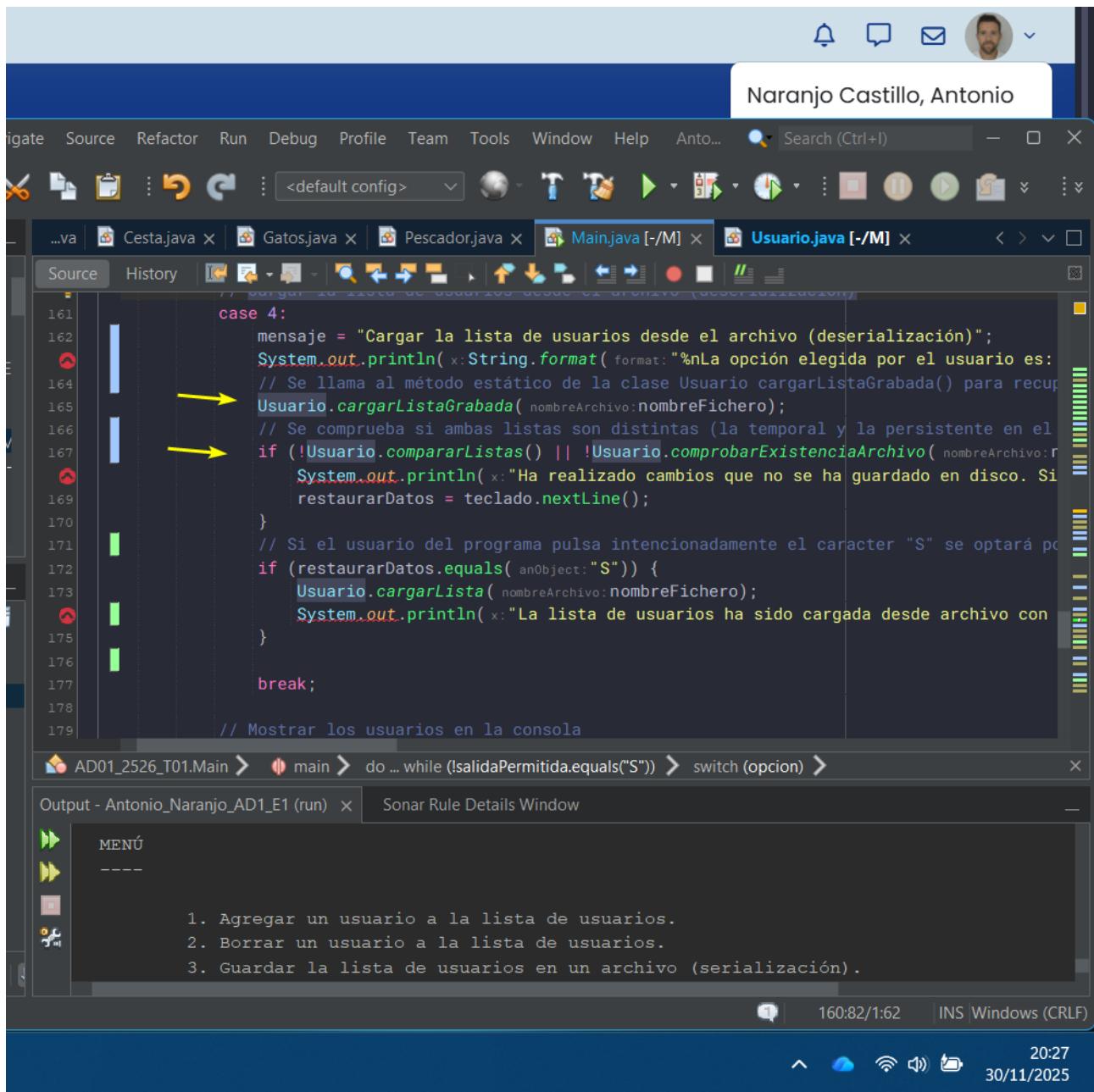
1. Agregar un usuario a la lista de usuarios.  
2. Borrar un usuario a la lista de usuarios.  
3. Guardar la lista de usuarios en un archivo (serialización).

160:82/1:62 | INS | Windows (CRLF)

20:23 | 30/11/2025

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.

## 12. ¿Recuperar datos de disco?



```

161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
    case 4:
        mensaje = "Cargar la lista de usuarios desde el archivo (deserialización)";
        System.out.println(x:String.format(format:"%nLa opción elegida por el usuario es:");
        // Se llama al método estático de la clase Usuario cargarListaGrabada() para recuperar los datos del archivo
        Usuario.cargarListaGrabada(nombreArchivo:nombreFichero);
        // Se comprueba si ambas listas son distintas (la temporal y la persistente en el fichero)
        if (!Usuario.compararListas() || !Usuario.comprobarExistenciaArchivo(nombreArchivo:nombreFichero))
            System.out.println(x:"Ha realizado cambios que no se ha guardado en disco. Si desea restaurarlos pulse 'S' para cargarlos");
            restaurarDatos = teclado.nextLine();
        }
        // Si el usuario del programa pulsa intencionadamente el carácter "S" se optará por cargar los datos
        if (restaurarDatos.equals(anObject:"S"))
            Usuario.cargarLista(nombreArchivo:nombreFichero);
            System.out.println(x:"La lista de usuarios ha sido cargada desde archivo con éxito");
        }
        break;
    }
    // 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)

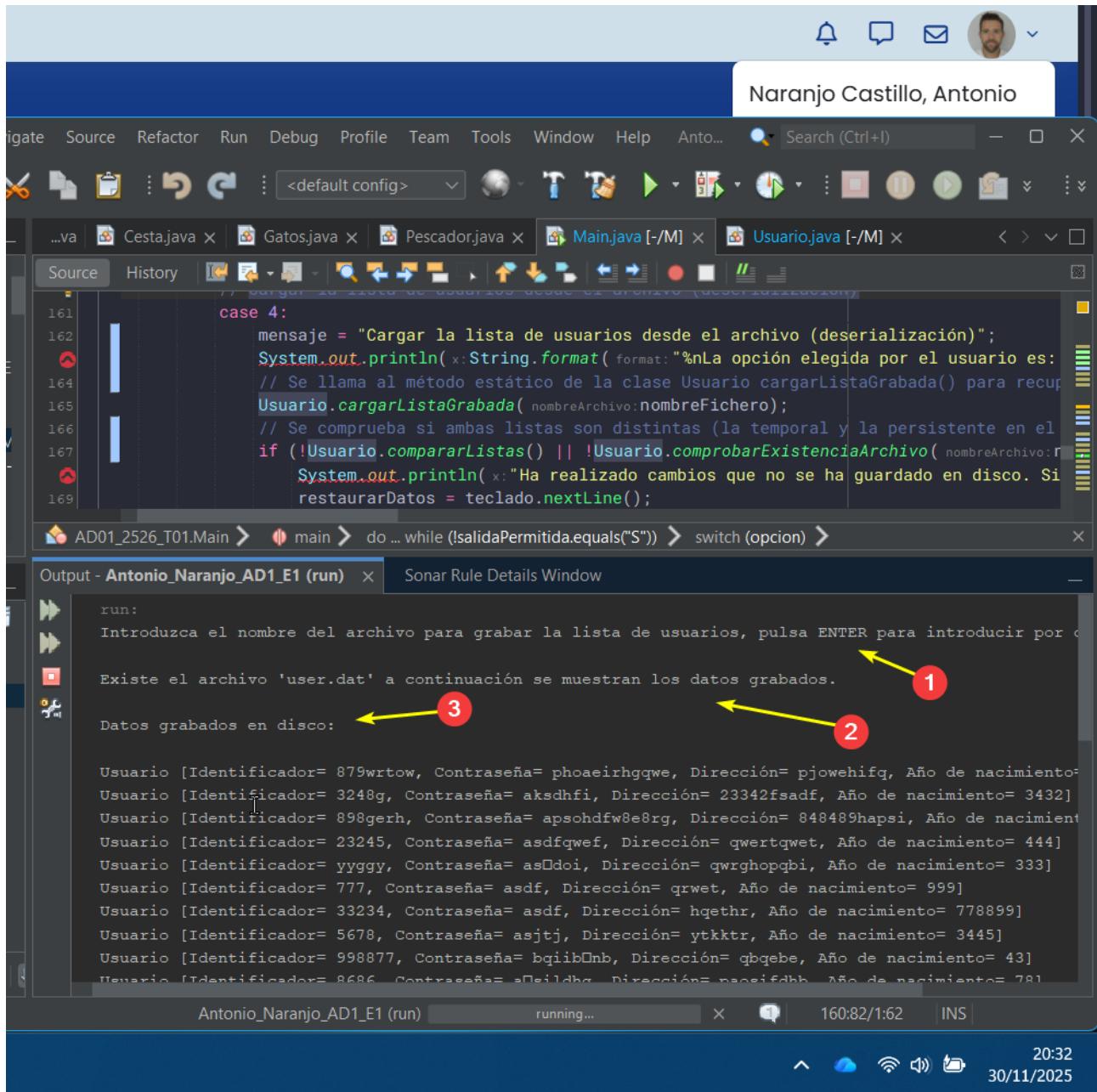
20:27 | 30/11/2025

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.

## 13. Resultados del programa en consola

### 13.1. Inicio del programa



The screenshot shows an IDE interface with the following details:

- Title Bar:** Naranjo Castillo, Antonio
- Menu Bar:** Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, Auto...
- Toolbar:** Includes icons for剪切 (Cut), 复制 (Copy), 粘贴 (Paste), Undo, Redo, 搜索 (Search), 打开 (Open), 保存 (Save), 编辑 (Edit), 运行 (Run), 调试 (Debug), etc.
- Project Explorer:** Shows files: Cestajava.x, Gatos.java.x, Pescador.java.x, Main.java [-/M] x, and Usuario.java [-/M] x.
- Code Editor:** Displays the code for Main.java, specifically the case 4 block. The code reads from a file named 'user.dat' and prints the contents to the console.
- Output Console:**
  - Shows the command run: followed by instructions to enter a file name.
  - Indicates that the file 'user.dat' exists and displays its contents.
  - Prints the list of users stored in the file.
- Bottom Status Bar:** Shows the session name (Antonio\_Naranjo\_AD1\_E1 (run)), status (running...), time (160:82/1:62), and mode (INS).
- System Tray:** Shows icons for battery, signal, and date/time (20:32, 30/11/2025).

Annotations in the Output console:

- Annotation 1: Points to the message "Existe el archivo 'user.dat'".
- Annotation 2: Points to the message "Datos grabados en disco:".
- Annotation 3: Points to the list of User objects printed to the console.

```

case 4:
    mensaje = "Cargar la lista de usuarios desde el archivo (deserialización)";
    System.out.println( x:String.format( format: "%nLa opción elegida por el usuario es: " ) );
    // Se llama al método estático de la clase Usuario cargarListaGrabada() para recuperar la lista de usuarios
    Usuario.cargarListaGrabada( nombreArchivo:nombreFichero );
    // Se comprueba si ambas listas son distintas (la temporal y la persistente en el fichero)
    if ( !Usuario.compararListas() || !Usuario.comprobarExistenciaArchivo( nombreArchivo:nombreFichero ) )
        System.out.println( x:"Ha realizado cambios que no se ha guardado en disco. Si desea guardarlos pulse 'S' " );
    restaurarDatos = teclado.nextLine();

```

```

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:

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= asdfdoi, 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= qbqbe, Año de nacimiento= 43]
Usuario [Identificador= 8686, Contraseña= sPnildha, Dirección= nnsifdhh, Año de nacimiento= 781]

```

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 open at the top: ...va, Cestajava.x, Gatos.java, Pescador.java, Main.java [-/M] (active), and Usuario.java [-/M]. The main editor window displays Java code, specifically a switch statement handling option 4. The terminal window below shows the execution of the program. A user inputs 'user2.dat' (marked with a red circle 1) and receives a message stating it does not exist (marked with a red circle 2). After pressing enter (marked with a red circle 3), the menu is displayed.

```
case 4:
    mensaje = "Cargar la lista de usuarios desde el archivo (deserialización)";
    System.out.println(x:String.format(format:"%nLa opción elegida por el usuario es:");
    // Se llama al método estático de la clase Usuario cargarListaGrabada() para recuperar la lista de usuarios
    Usuario.cargarListaGrabada(nombreArchivo:nombreFichero);
    // Se comprueba si ambas listas son distintas (la temporal y la persistente en el fichero)
    if (!Usuario.compararListas() || !Usuario.comprobarExistenciaArchivo(nombreArchivo:nombreFichero))
        System.out.println(x:"Ha realizado cambios que no se ha guardado en disco. Si desea guardarlos pulse 'S' para salir");
    restaurarDatos = teclado.nextLine();
```

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

```
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

Antonio\_Naranjo\_AD1\_E1 (run) running... x 160:82/1:62 INS 20:37 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.

## 13.2. Agregar un usuario

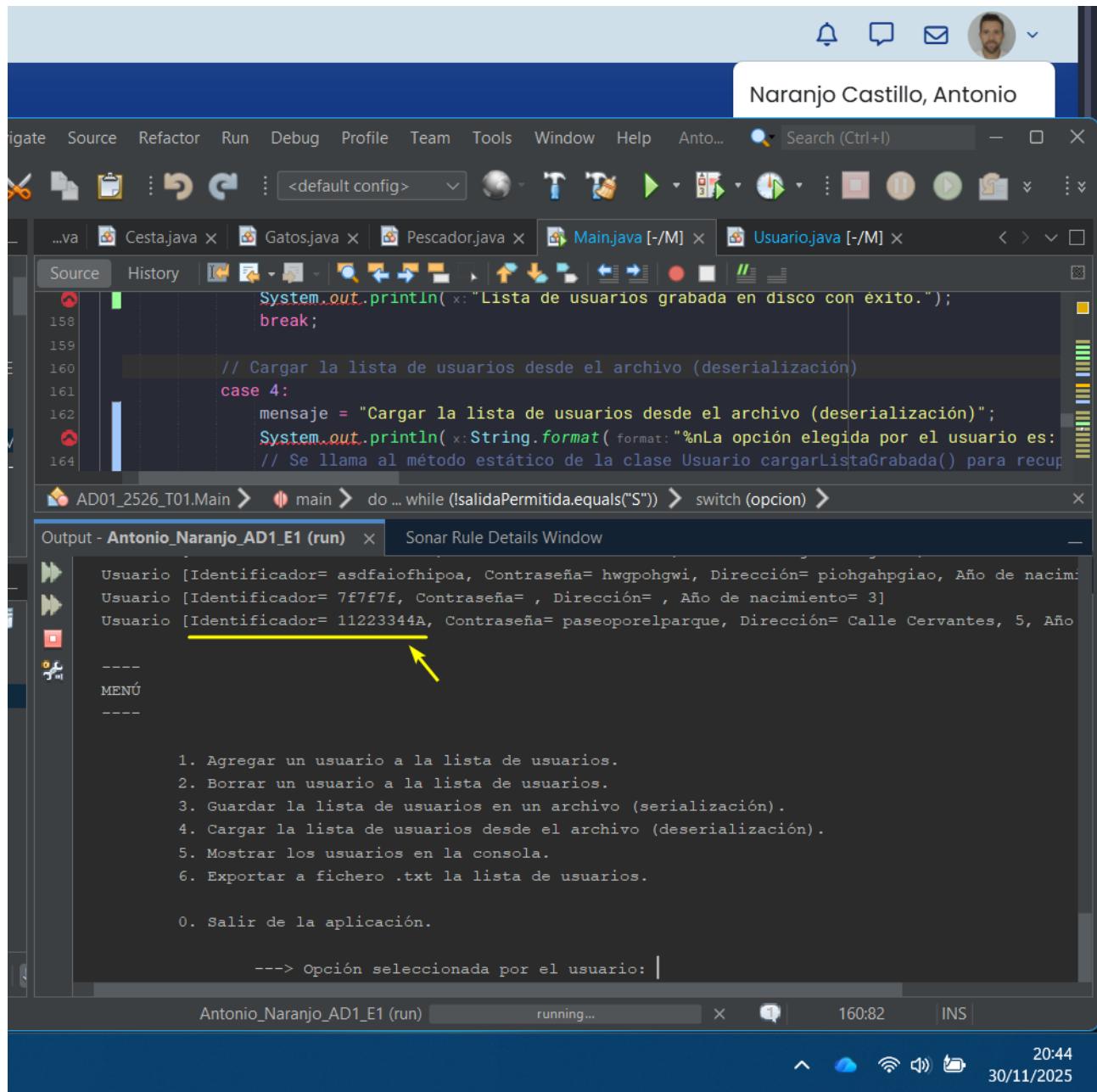
The screenshot shows an IDE interface with several tabs at the top: ...va, Cesta.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M]. The Main.java tab is active, displaying Java code. The code includes a switch statement where option 1 is selected, leading to the execution of `cargarListaGrabada()`. The output window shows the program's interaction with the user:

```
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  
----
```

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

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

### 13.3. Mostrar los usuarios en consola



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 Output window shows the execution 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]
```

A yellow arrow points to the last user entry in the list. Below the list is 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).
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 shows the application is running, the time is 16:08, and the date is 30/11/2025.

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

### 13.4. 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. Below the tabs is a toolbar with various icons. The main workspace shows a portion of the Main.java 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.
```

Below the code editor is a stack trace or call tree:

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

The Output window, titled "Output - Antonio\_Naranjo\_AD1\_E1 (run)", displays the following terminal session:

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

Two yellow arrows point to the user ID '11223344A' and the success message 'El usuario con identificador '11223344A' ha sido borrado con éxito'. The terminal also shows a menu and a list of 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).  
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 shows the process name "Antonio\_Naranjo\_AD1\_E1 (run)", status "running...", time "16:08:22", and date "30/11/2025". The system tray icons include signal strength, battery, and network.

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 Source (selected). The main area displays Java code for a `Main.java` file. The code includes logic for saving user lists to disk and loading them from files. The output window shows the application's behavior:

```
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
    // y se muestra en la consola.
```

Output - Antonio\_Naranjo\_AD1\_E1 (run) x 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  
----

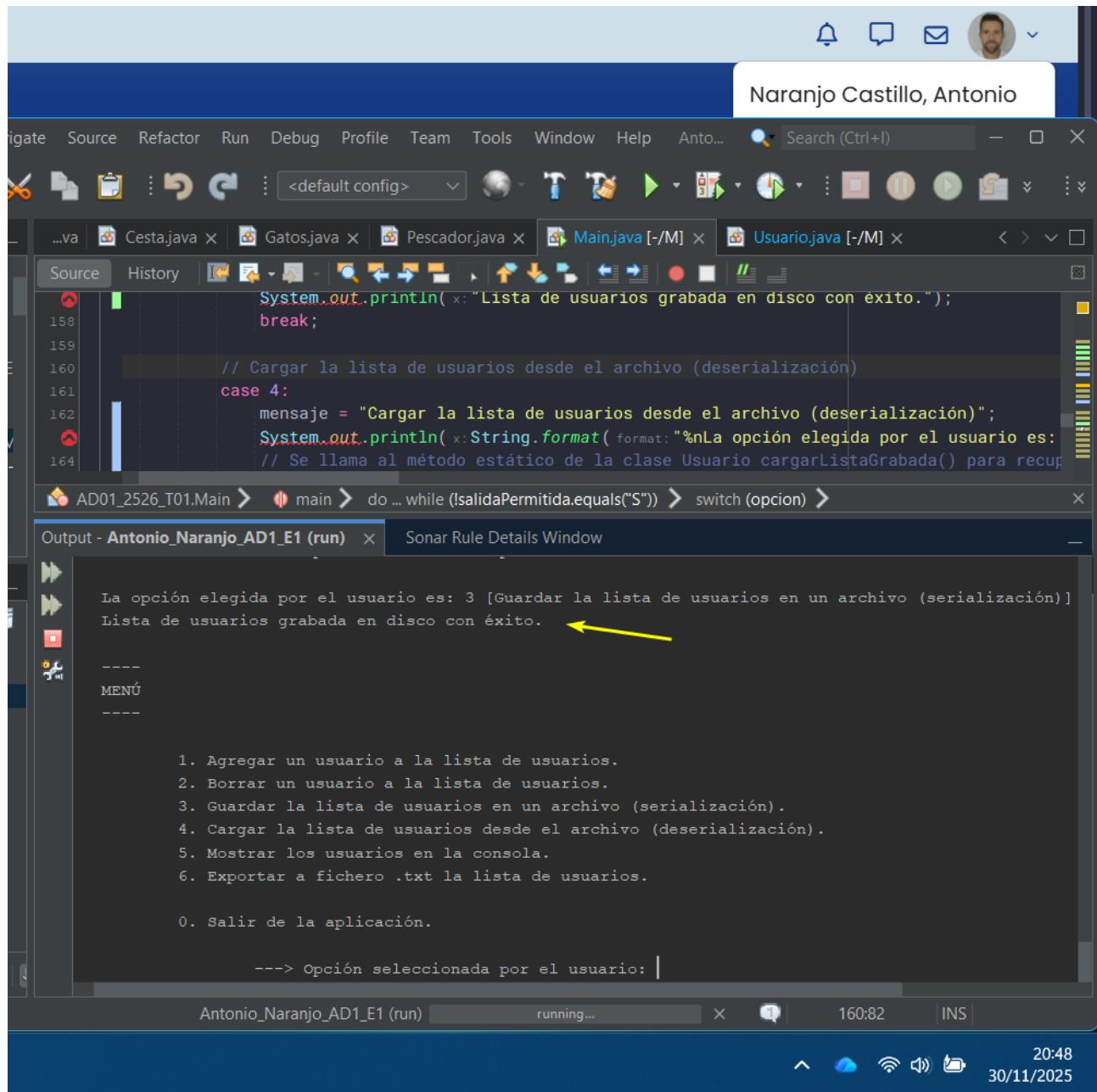
1. Agregar un usuario a la lista de usuarios.
2. Borrar un usuario a la lista de usuarios.
3. Guardar la lista de usuarios en un archivo (serialización).
4. Cargar la lista de usuarios desde el archivo (deserialización).
5. Mostrar los usuarios en la consola.
6. Exportar a fichero .txt la lista de usuarios.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

---> Opción seleccionada por el usuario: |

Antonio\_Naranjo\_AD1\_E1 (run) x 160:82 | INS | 20:47 30/11/2025

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.

### 13.5. Guardar lista (Serialización)



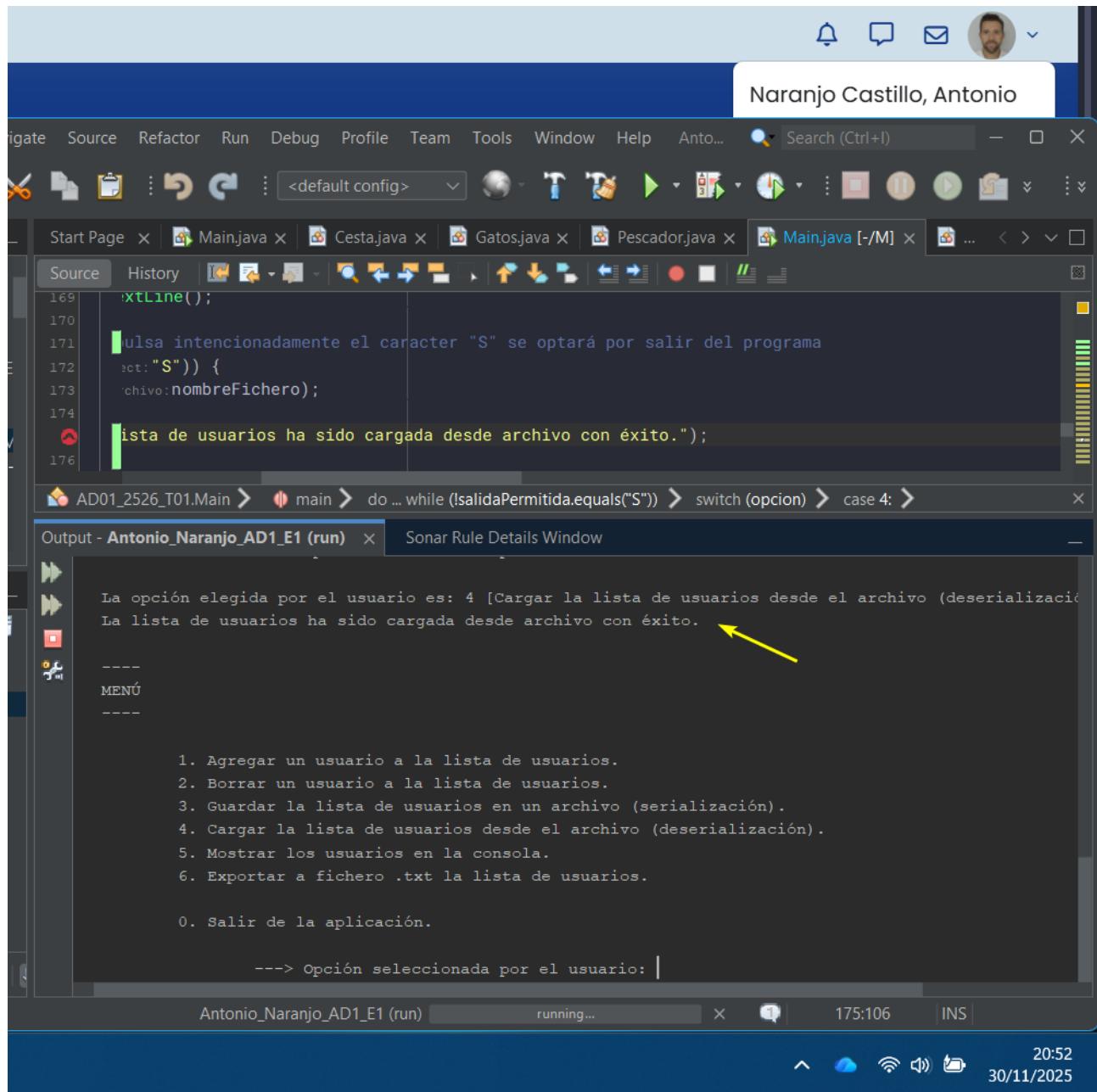
The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the user's name: "Naranjo Castillo, Antonio".
- Toolbar:** Includes standard icons for file operations like Open, Save, and Print.
- Menu Bar:** Contains options like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... A search bar is also present.
- Project Explorer:** Lists files: ...va, Cesta.java, Gatos.java, Pescador.java, Main.java [-/M], and Usuario.java [-/M].
- Code Editor:** Displays Java code. Lines 160-164 show:

```
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.
```
- Call Stack:** Shows the current call stack: AD01\_2526\_T01.Main > main > do ... while (!salidaPermitida.equals("S")) > switch (opcion) >.
- Output Window:** Titled "Output - Antonio\_Naranjo\_AD1\_E1 (run)". It displays the message "La opción elegida por el usuario es: 3 [Guardar la lista de usuarios en un archivo (serialización)]". Below it, a yellow arrow points to the message "Lista de usuarios grabada en disco con éxito.". The output window also lists menu options from 1 to 0.
- System Tray:** Shows the date and time: 30/11/2025, 20:48.

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

### 13.6. Cargar lista (deserialización)



The screenshot shows an IDE interface with the following details:

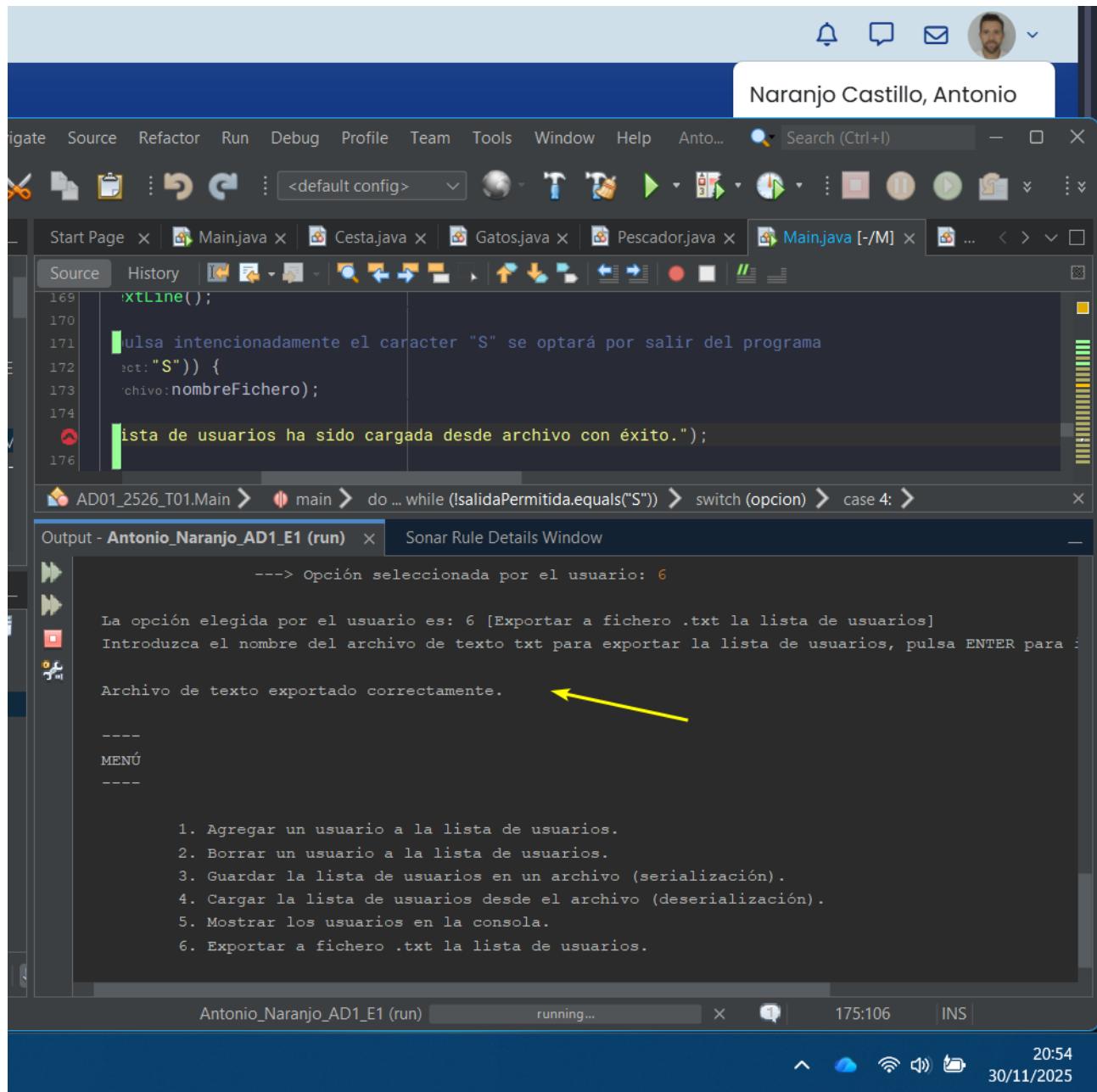
- Toolbar:** Includes icons for剪切 (Cut), 复制 (Copy), 粘贴 (Paste), etc.
- Menu Bar:** Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, Auto... (Auto-completion), Search (Ctrl+I).
- Project Bar:** Start Page, Main.java, Cestajava, Gatosjava, Pescadorjava, Main.java [-/M].
- Code Editor:** Displays Java code for reading a file named "fichero.nombreFichero".
- 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 to the last line of the output.
- Bottom Status Bar:** Antonio\_Naranjo\_AD1\_E1 (run), running..., 175:106, INS, 20:52, 30/11/2025.

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

### 13.7. Exportación archivo TXT



The screenshot shows an IDE interface with several tabs at the top: Start Page, Main.java, Cestajava, Gatosjava, Pescador.java, and Main.java [-/M]. The Main.java tab is active, displaying Java code. The code includes a method that handles the export of a user list to a .txt file if the user selects option 6.

```
169     System.out.println();
170     171     if (salidaPermitida.equals("S")) {
172         String nombreFichero;
173         nombreFichero = JOptionPane.showInputDialog("Introduzca el nombre del archivo txt para exportar la lista de usuarios, pulsa ENTER para cancelar");
174         if (nombreFichero != null) {
175             try {
176                 FileOutputStream fichero = new FileOutputStream(nombreFichero);
177                 ObjectOutputStream salida = new ObjectOutputStream(fichero);
178                 salida.writeObject(listaDeUsuarios);
179                 salida.close();
180                 System.out.println("La lista de usuarios ha sido cargada desde archivo con éxito.");
181             } catch (IOException e) {
182                 e.printStackTrace();
183             }
184         }
185     }
186 }
```

The output window shows the execution of the program. It prints the selected option (6), prompts for a file name, and then confirms that the file was exported correctly. A yellow arrow points to the confirmation message "Archivo de texto exportado correctamente.".

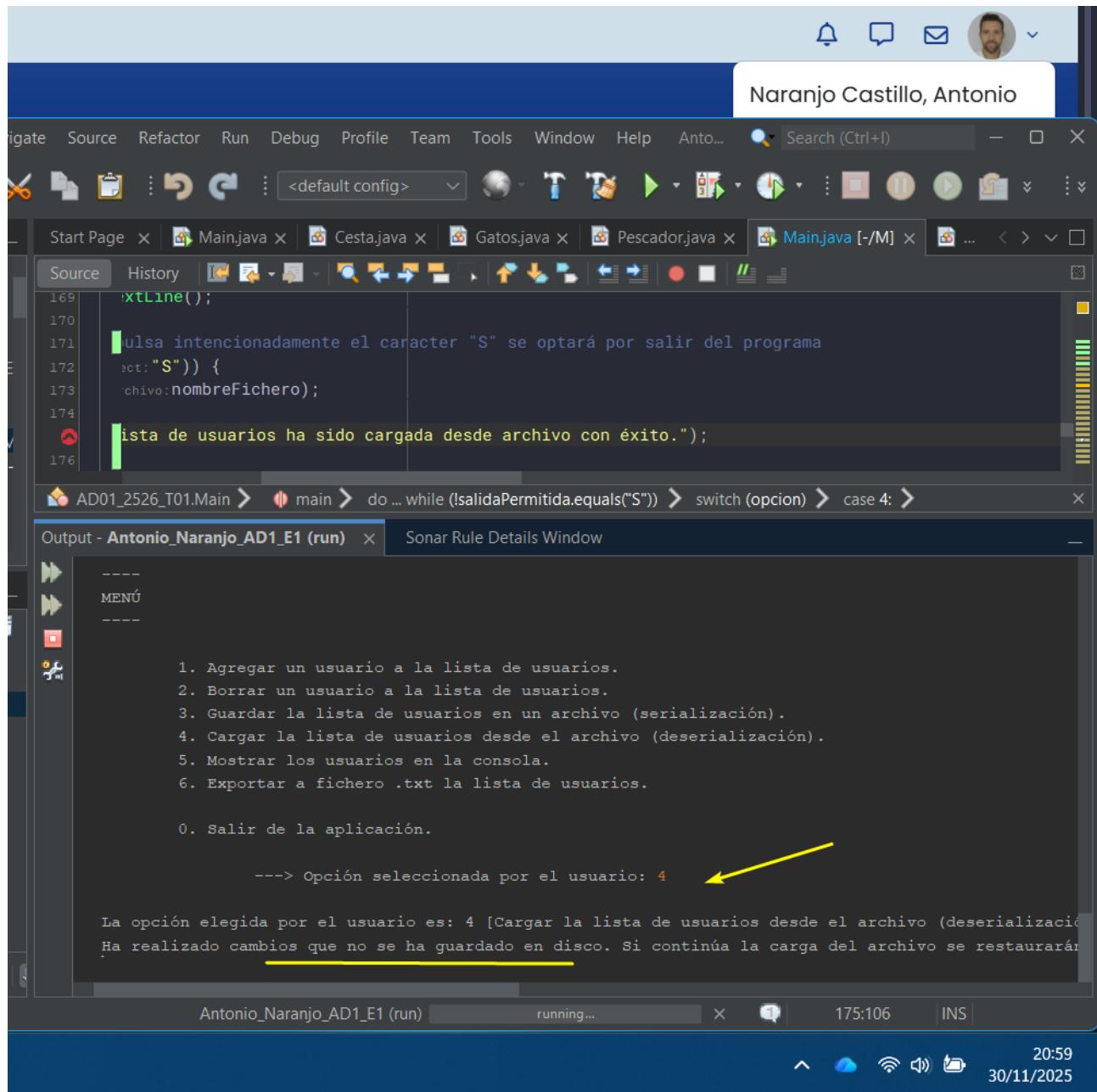
```
----> Opción seleccionada por el usuario: 6
La opción elegida por el usuario es: 6 [Exportar a fichero .txt la lista de usuarios]
Introduzca el nombre del archivo txt para exportar la lista de usuarios, pulsa ENTER para cancelar
Archivo de texto exportado correctamente. ←
----<
MENÚ
----<

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.
```

The status bar at the bottom shows the process is running, the time is 175:106, and the date is 30/11/2025.

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.

### 13.8. ¿Cargar datos guardados?



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 like Open, Save, and Cut.
- Menu Bar:** Lists options like Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Auto... A search bar is also present.
- Code Editor:** Displays Java code in Main.java. The code includes logic for handling character input and file serialization/deserialization.
- Output Window:** Shows the application's console output. It displays a menu of options (1-6) and then highlights the selection "4" with a yellow arrow. The output also includes a message about changes made to the file.
- Status Bar:** Shows the run configuration "Antonio\_Naranjo\_AD1\_E1 (run)", the status "running...", the time "175:106", and the date "30/11/2025".

En caso de que el usuario pretenda cargar los datos guardado 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 saving a file. If 'S' is pressed, it saves the file and exits. Otherwise, it exits without saving.

```
169     if (letra.equals("S")) {
170         // Guarda el archivo
171         Guardar();
172         System.out.println("La lista de usuarios ha sido cargada desde archivo con éxito.");
173     } else {
174         System.out.println("Has pulsado una letra incorrecta, se ha salido del programa.");
175     }
176 }
```

The terminal window shows the application's output:

```
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 highlight the 'S' key press and the success message in the terminal output.

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.

### 13.9. Salida del programa

The screenshot shows an IDE interface with several tabs at the top: Start Page, Main.java, Cestajava, Gatosjava, Pescador.java, Main.java [-/M], and another Main.java tab. The Main.java [-/M] 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 a selection of 0. It then asks if the user wants to save changes before exiting. The user types 'S' and the terminal responds with "BUILD SUCCESSFUL (total time: 55 seconds)". The status bar at the bottom right shows the date and time as 30/11/2025 and 21:05.

```
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.