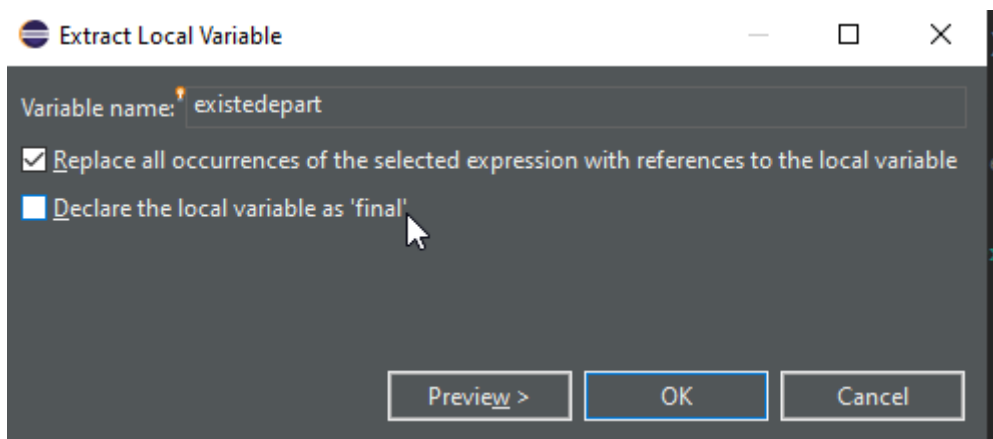
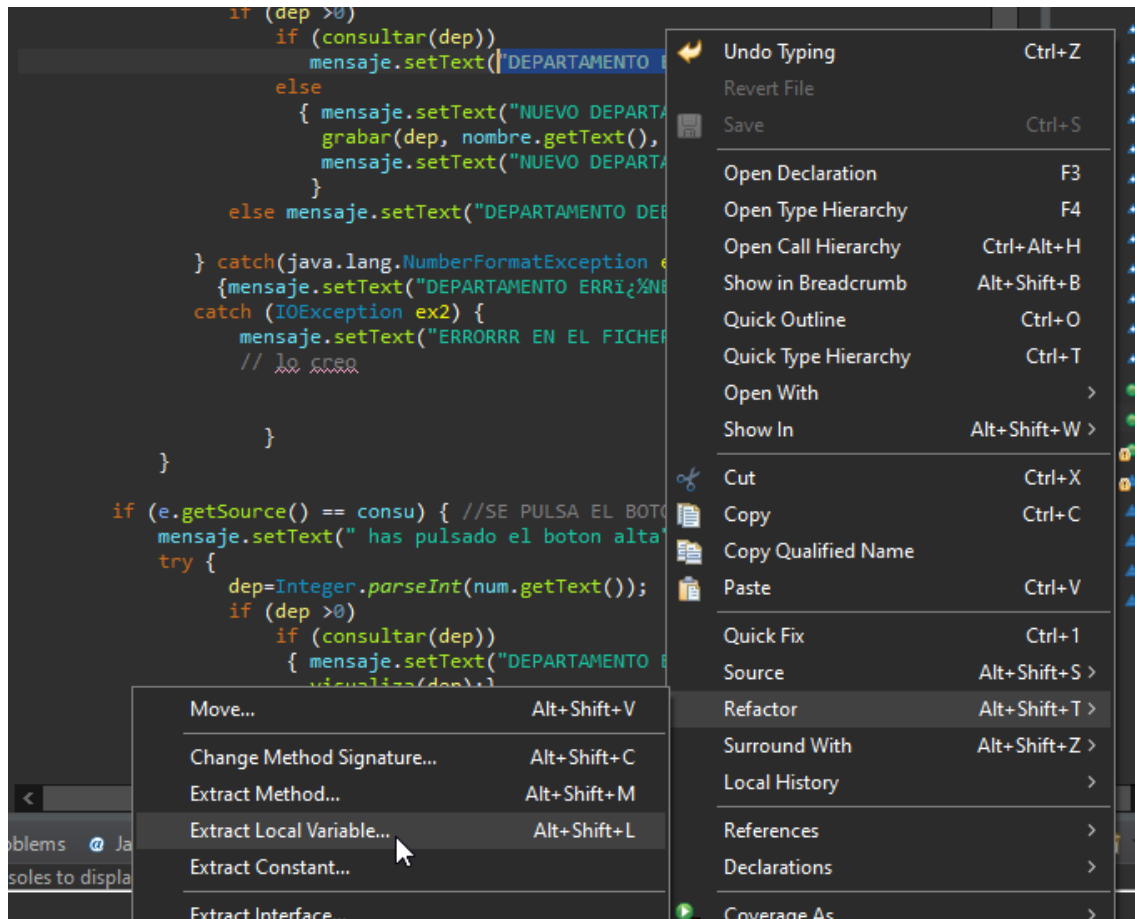


# Refactorización

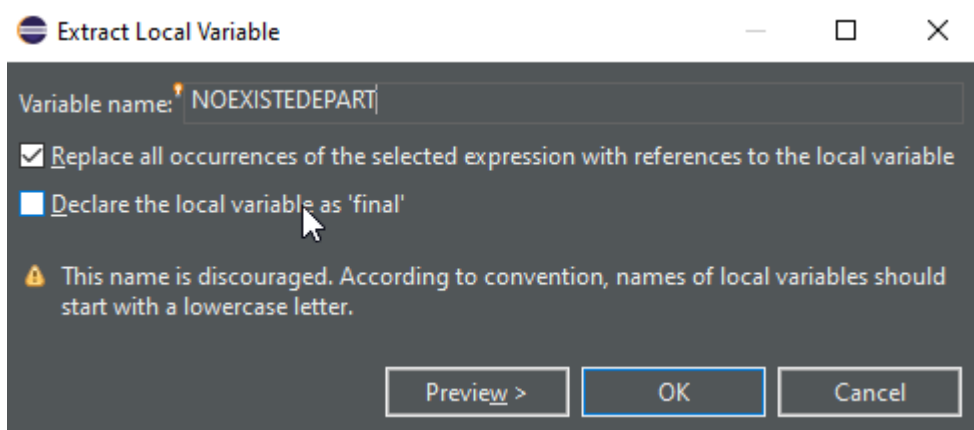
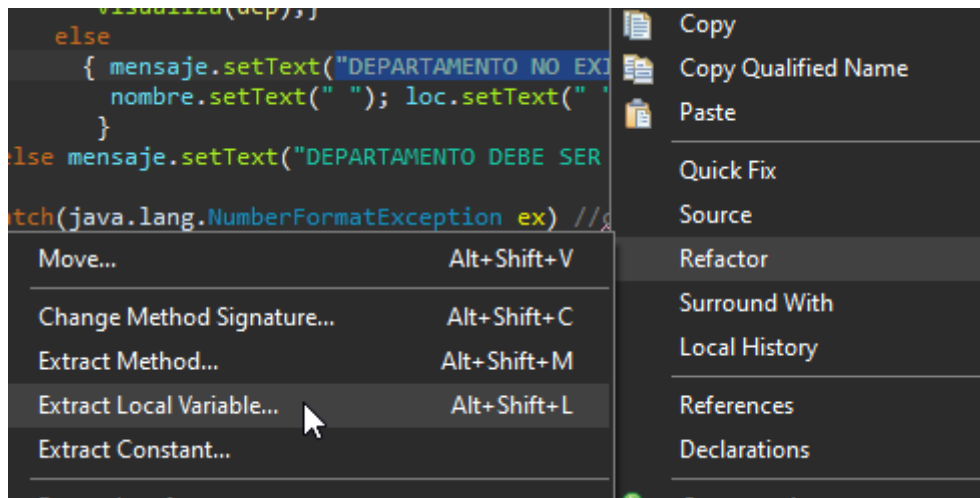
Ejercicio 2:

Extraemos una variable local de la cadena "DEPARTAMENTO EXISTE".



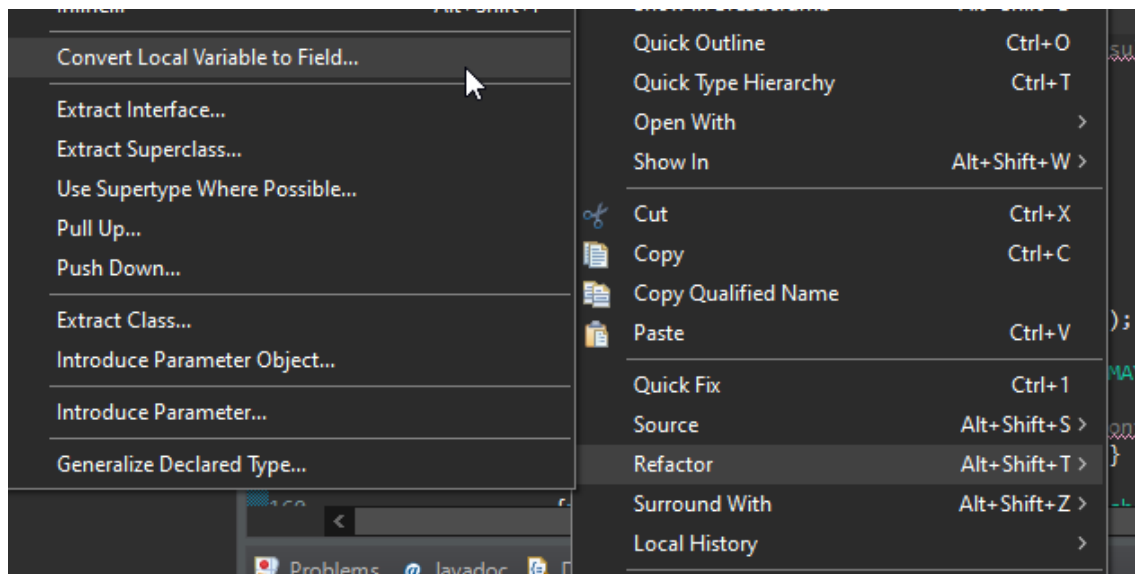
```
String existedepart = "DEPARTAMENTO EXISTE.";
if (e.getSource() == balta) { //SE PULSA EL BOTON alta
    mensaje.setText(" has pulsado el boton alta");
```

Extraemos una variable local de la cadena "DEPARTAMENTO NO EXISTE".



```
String NOEXISTEDEPART = "DEPARTAMENTO NO EXISTE.";  
if (e.getSource() == consu) { //SE PULSA EL BOTON consultar  
  mensaje.setText(" has pulsado el boton alta");  
  try {  
    dep=Integer.parseInt(num.getText());  
    if (dep > 0)  
      if (consultar(dep))  
      { mensaje.setText(existedepart);  
        visualiza(dep);}  
    else  
    { mensaje.setText(NOEXISTEDEPART);  
      nombre.setText(" "); loc.setText(" ");  
    }  
  }  
}
```

La convertimos a un atributo de la clase.



Field name:

Access modifier

☒ public ☐ protected ☐ package ☐ private

Initialize in

☒ Field declaration  
☐ Current method  
☐ Class constructors

☐ Declare field as 'static'

☐ Declare field as 'final'

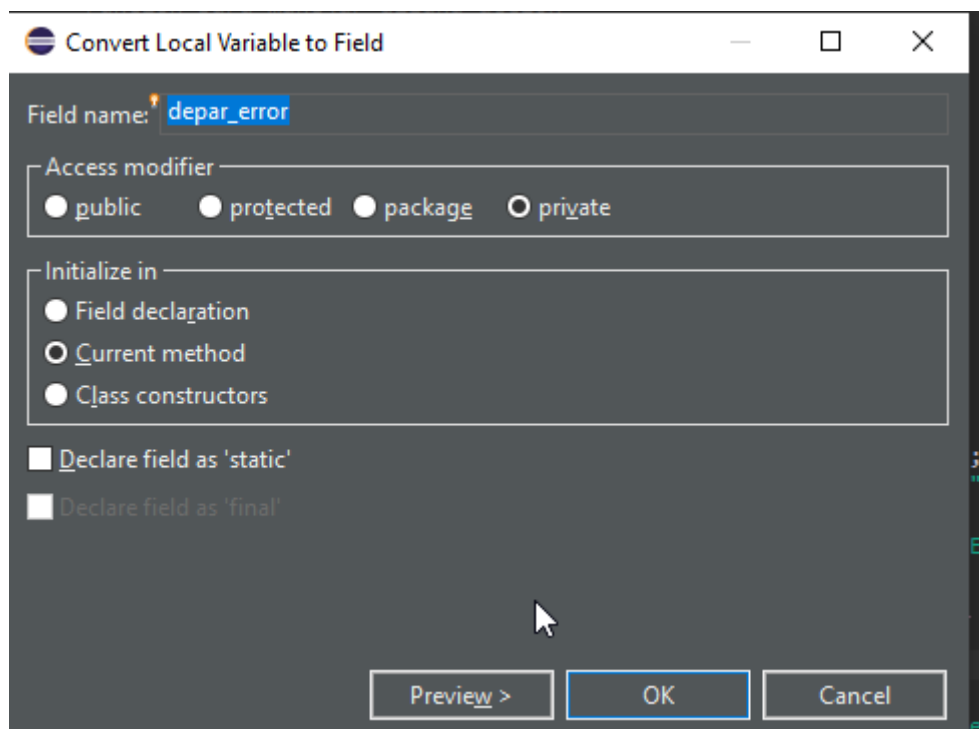
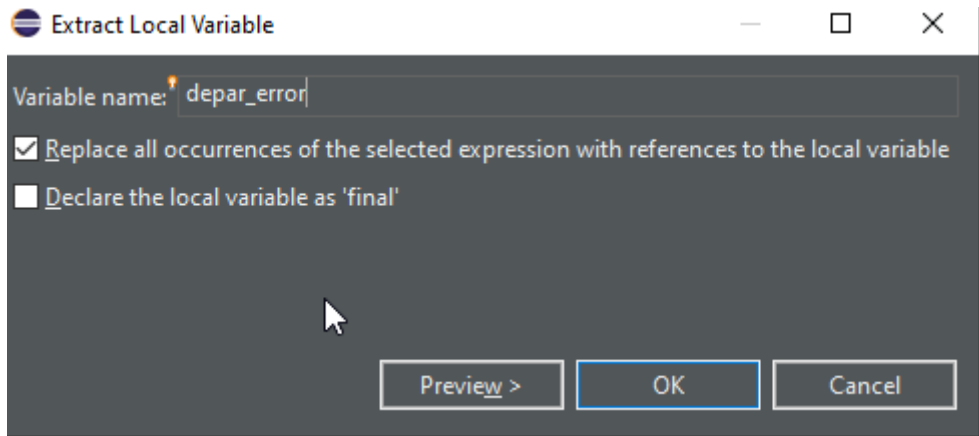
```
JLabel mensaje=new JLabel("-----")
JLabel titulo=new JLabel("GESTIÓ")

JLabel lnum = new JLabel("NUMERO D")
JLabel lnom = new JLabel("NOMBRE:")
JLabel lloc = new JLabel("LOCALIDA")

JButton balta= new JButton("Inserta")
JButton consu= new JButton("Consult")
JButton borra= new JButton("Borrar")
JButton breset=new JButton("Limpiar")
JButton modif=new JButton("Modifica")
JButton ver=new JButton("Ver por co")
JButton fin=new JButton("CERRAR");
Color c; //para poner colores
// WHITE, LIGHTGRAY, GRAY, DARKGRAY, B
private String nOEXISTEDEPART;

/**
```

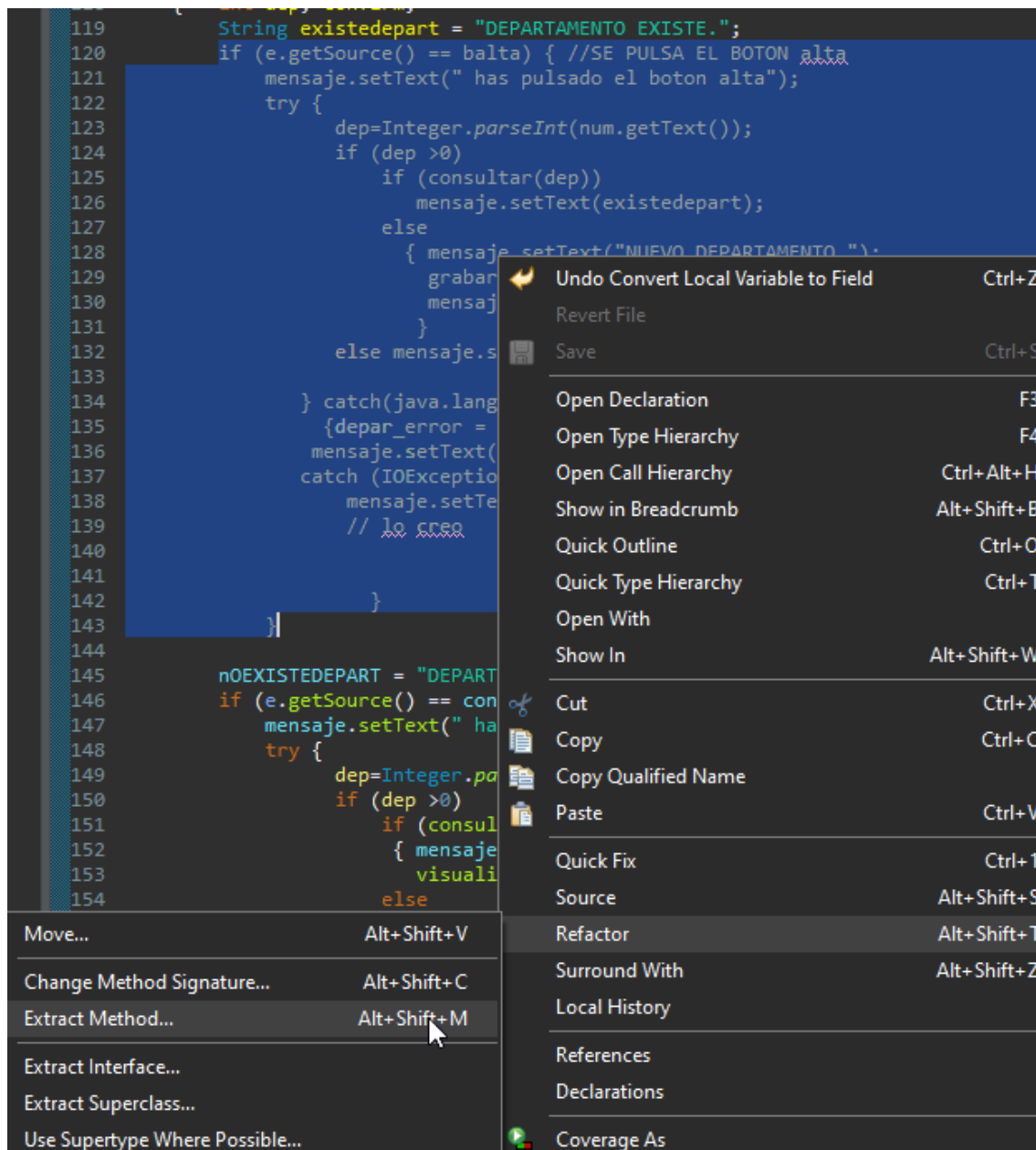
Creamos una variable local de la cadena "DEPARTAMENTO ERRÓNEO" y la convertimos en un atributo de clase.



```
Color c; //para poner colores
// WHITE, LIGHTGRAY, GRAY, DARKGRAY
private String noEXISTEDEPART;
private String depar_error;
```

### Ejercicio 3:

Seleccionamos cada if que haga referencia a cada una de las funciones que buscamos, le damos clic derecho, refactorizar, y extract method.



```

* */
public void actionPerformed(ActionEvent e)
{
    int dep, confirm;
    String existedepart = "DEPARTAMENTO EXISTE.";

    insertar_departamento(e, existedepart);

    consultar(e, existedepart);

    borrar(e, existedepart);

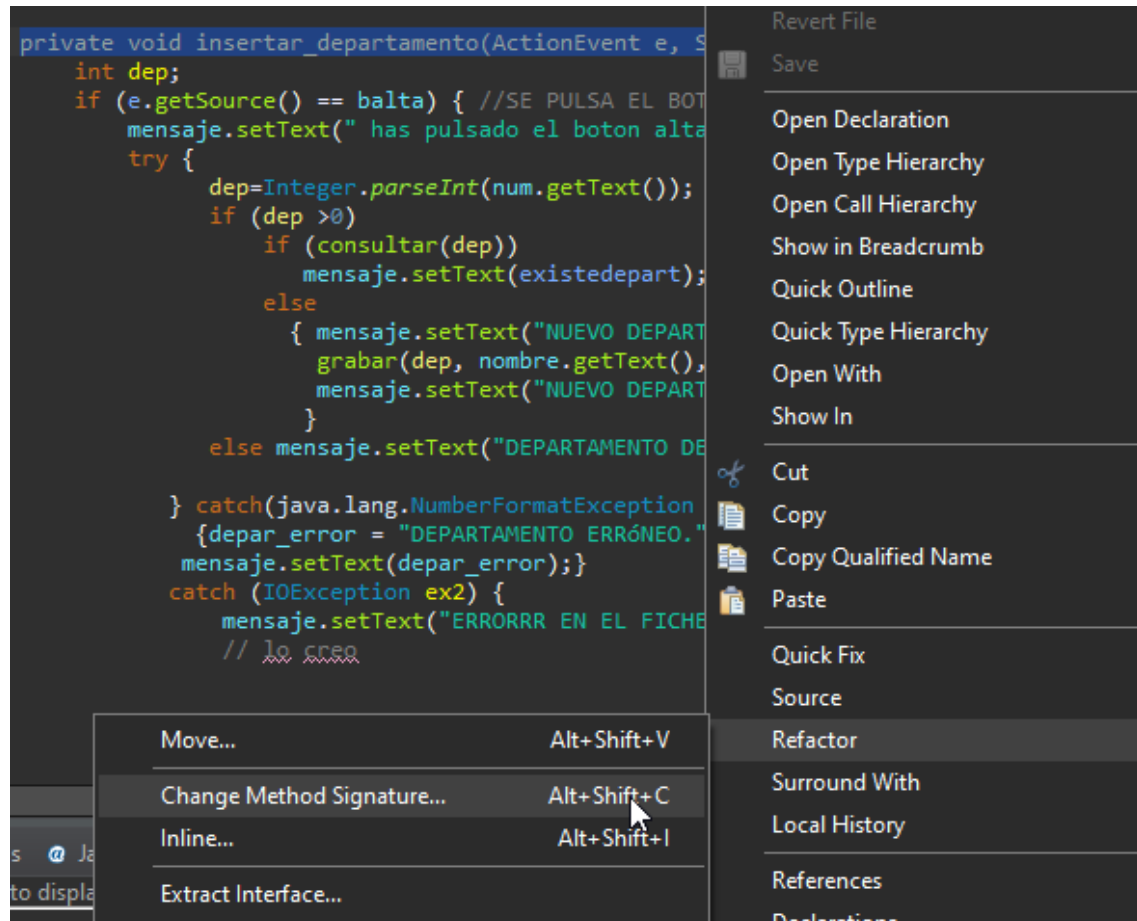
    modificar(e, existedepart);

    if (e.getSource() == fin) { //SE PULSA EL BOTON salir
        System.exit(0);
        //dispose();
    }
    if (e.getSource() == ver) { //SE PULSA EL BOTON ver de

```

#### Ejercicio 4:

Para cambiar la firma de un método, seleccionamos su declaración, hacemos clic derecho, refactorizar, y change method signature. Ahí añadiremos la cadena con valor por defecto "PRUEBA" y haremos que devuelvan un entero.



# Change Method Signature

Access modifier: private Return type: void Method name: insertarDepartamento

Parameters Exceptions

Type	Name	Default value
ActionEvent	e	-
String	existedepart	-
String	prueba	"PRUEBA"

☐ Keep original method as delegate to changed method

☒ Mark as deprecated

Method signature preview:

```
private void insertarDepartamento(ActionEvent e, String
existedepart, String prueba)
```

Access modifier: private Return type: int Method name: insertarDepartamento

### Ejercicio 5:

Añadimos la clase anidada indicada y el código a la función indicada.

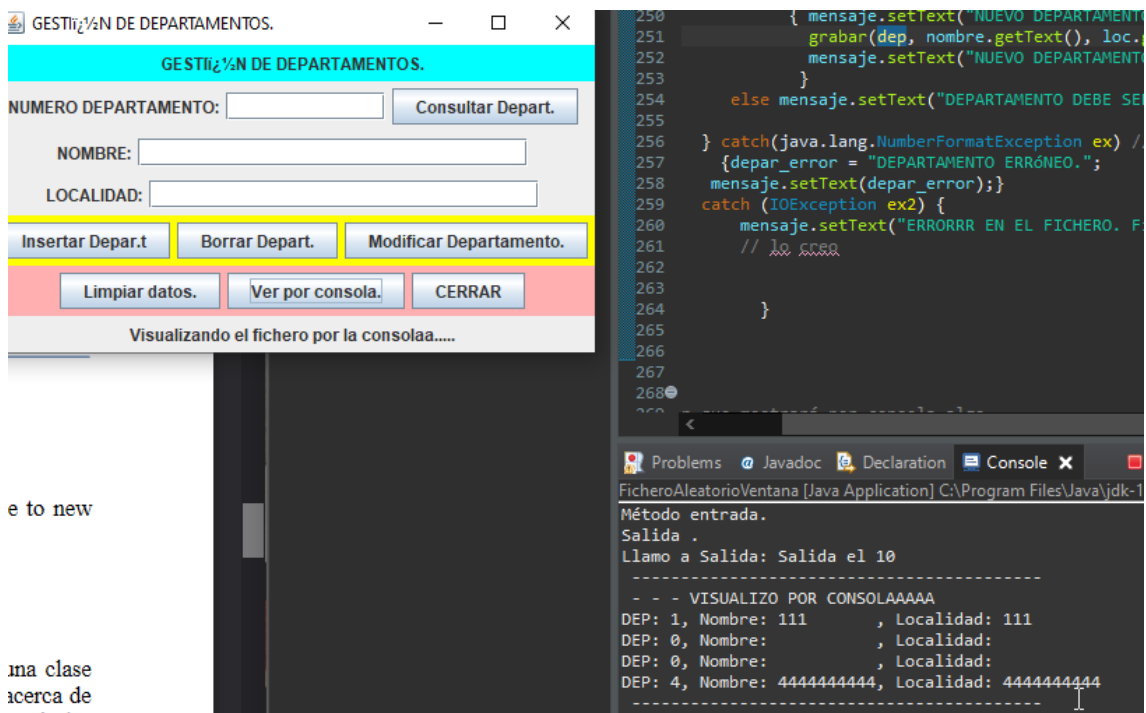
```
FicheroAleatorioVentana.java  VentanaDepart.java X
435      * Para grabar algo
436      * @param dep
437      * @param nom
438      * @param loc
439      * */
440  void grabar(int dep, String nom, String loc)
441  {
442      long pos; StringBuffer buffer = null;
443      File fichero = new File("AleatorioDep.dat");
444      try {
445          RandomAccessFile file = new RandomAccessFile(fichero,
446              // Calculo del reg a leer
447              pos=44 * (dep-1);
448              //if (file.length()==0) return false; // si está% ya
449
450          file.seek(pos);
451          file.writeInt(dep);
452          buffer = new StringBuffer( nom );
453          buffer.setLength(10);
454          file.writeChars(buffer.toString()); //insertar nombre
455          buffer = new StringBuffer( loc );
456          buffer.setLength(10);
457          file.writeChars(buffer.toString()); //insertar loc
458          file.close();
459          System.out.println(" GRABADOOO el "+dep);
460      } catch (IOException e1) {
461          System.out.println("ERROR AL grabarr AleatorioDep
462              e1.printStackTrace();
463      }
464  } // fin grabar
465
466  class claseAnidada{
467      void entrada(){
468          System.out.println("Método entrada. ");
469      }
470      String salida (int d) {
471          System.out.println("Salida . ");
472          return "Salida el " + d;
473      }
474  } // fin de clase anidada
475  } //fin clase
476
```

```
public void verporconsola() throws IOException {
    String nom="",loc=""; int dep=0; long pos;
    File fichero = new File("AleatorioDep.dat");
    RandomAccessFile file = new RandomAccessFile(fichero, "r");
    char cad[] = new char[10], aux;

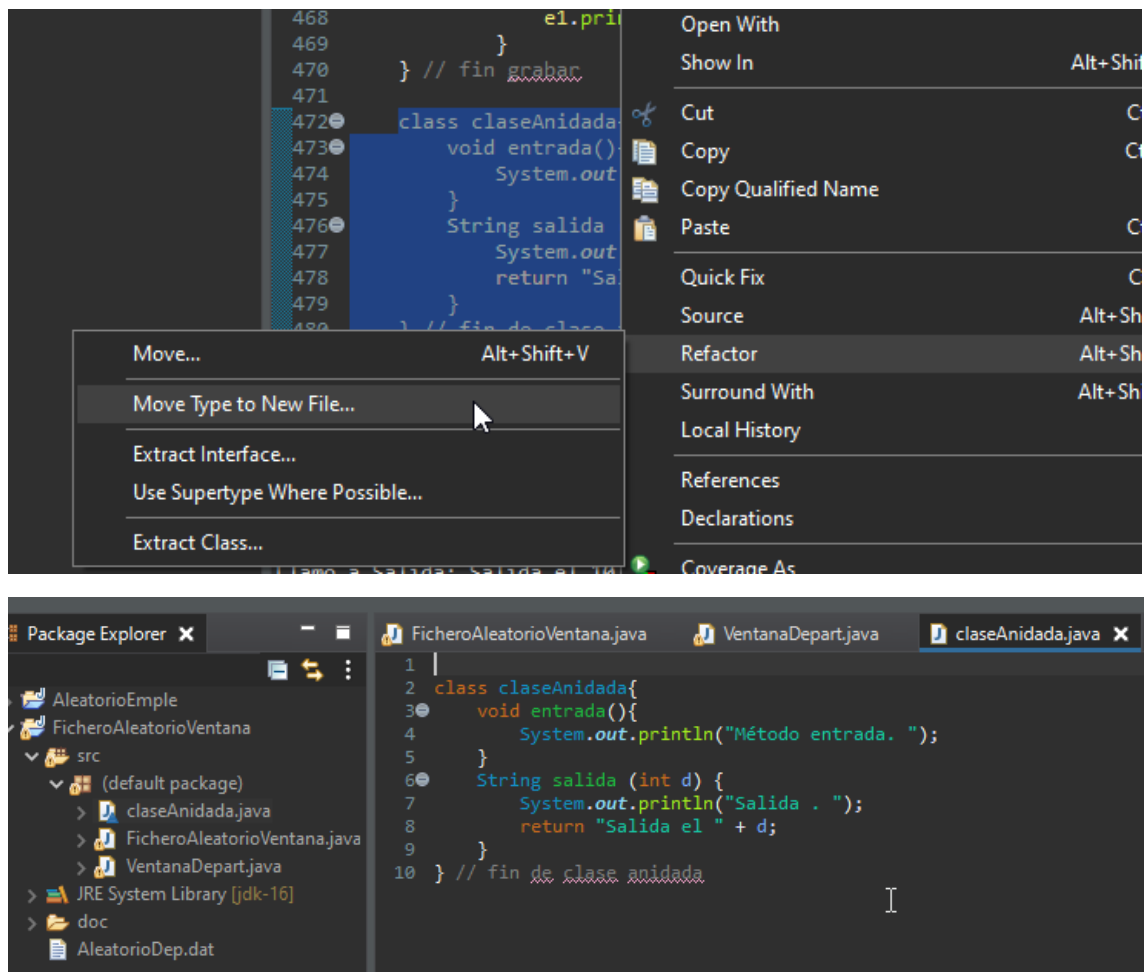
    FicheroAleatorioVentana fa=new FicheroAleatorioVentana();
    claseAnidada ej = new claseAnidada();
    ej.entrada();
    System.out.println("Llamo a Salida: "+ej.salida(10));
}
```



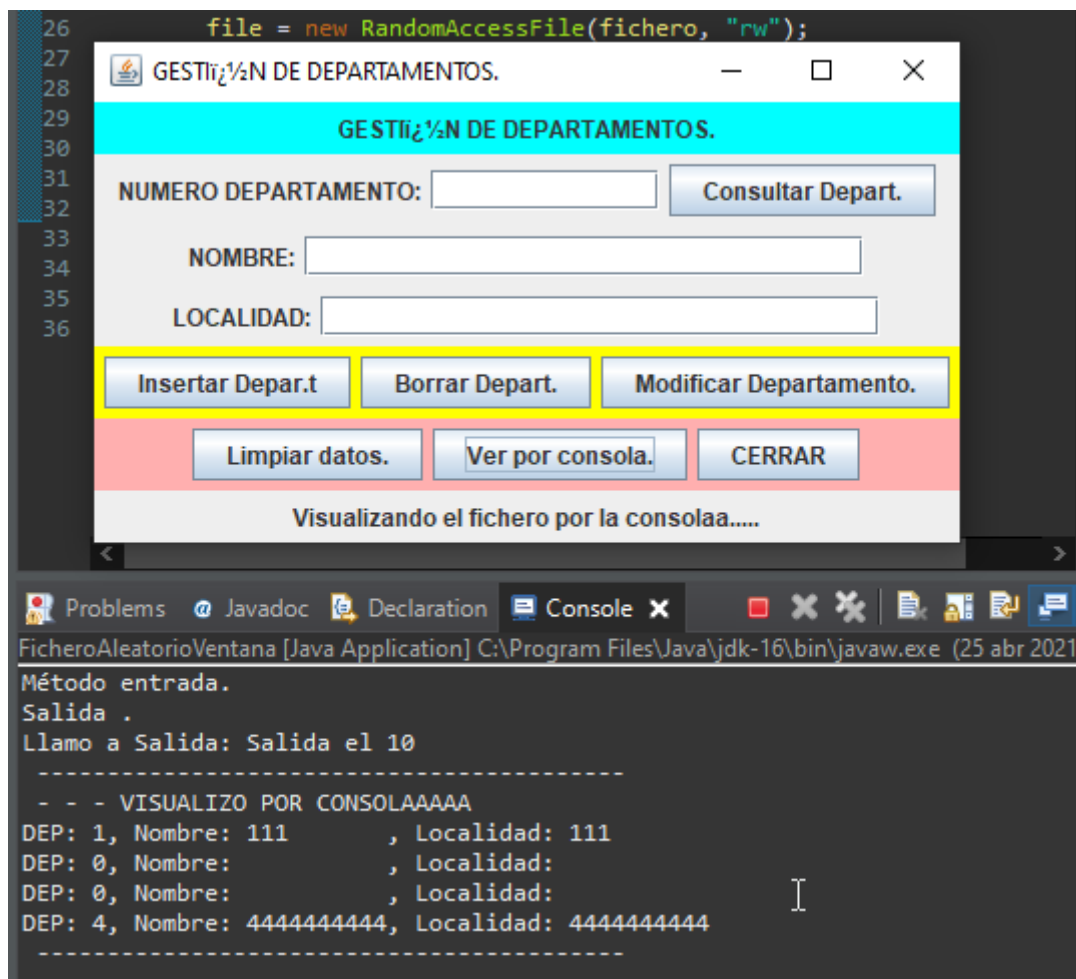
Probamos la ejecución.



Convertimos la clase anidada en una de nivel superior.

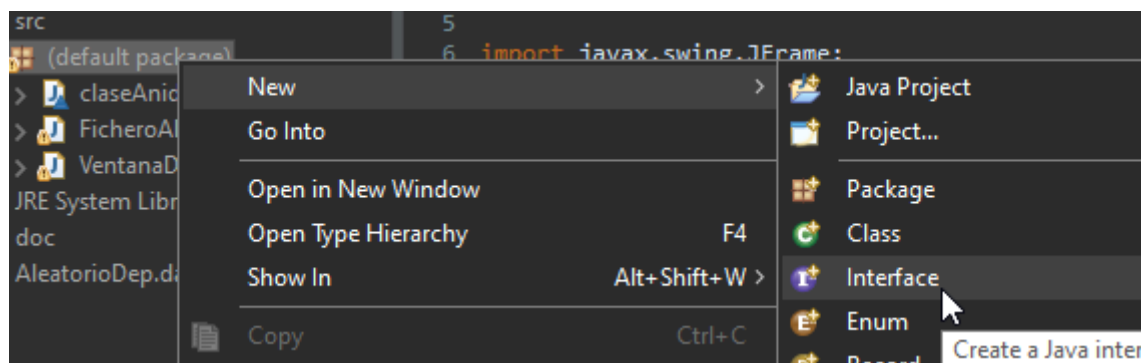


Volvemos a probar la ejecución.




Ejercicio 6:

Primero, creamos la interfaz y copiamos en ella el código del ejercicio 3.



### Java Interface

 The use of the default package is discouraged.

Source folder:

Package:

☐ Enclosing type:

---

Name:

Modifiers: ☐ public ☒ package ☐ private ☐ protected

Extended interfaces:

Do you want to add comments? (Configure templates and default value [here](#))

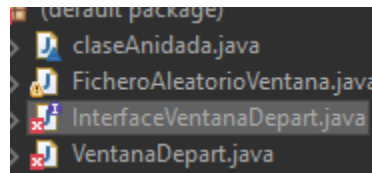
☐ Generate comments

```

1 import java.awt.event.ActionEvent;
2 import java.io.IOException;
3
4 import javax.swing.JOptionPane;
5
6 public interface InterfaceVentanaDepart {
7     public void modificar(ActionEvent e, String existe,
8         int dep;
9         int confirm;
10        if (e.getSource() == modif) { //SE PULSA EL
11            mensaje.setText(" has pulsado el boton M
12            try {
13                dep=Integer.parseInt(num.getText())
14                if (dep >0)
15                    if (consultar(dep))
16                        { mensaje.setText(existedeapar
17                          confirm=JOptionPane.showCor
18                            JOptionPane.OK_
19                          // si devuelve 0 es OK
20                          //mensaje.setText(" has p
21                          if (confirm==0)
22                          { modificar(dep);
23                            mensaje.setText(" REGIST
24                          }
25                      }
26                      else
27                          { mensaje.setText(noEXISTEDE

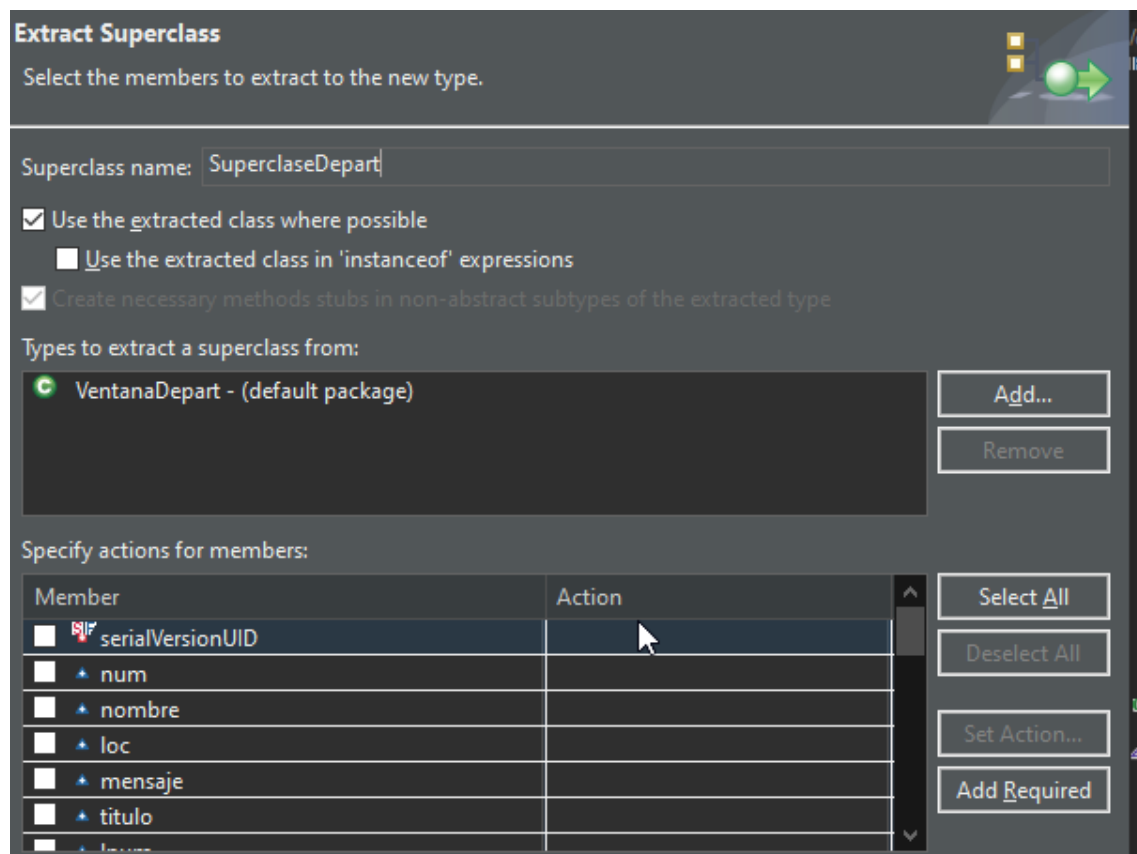
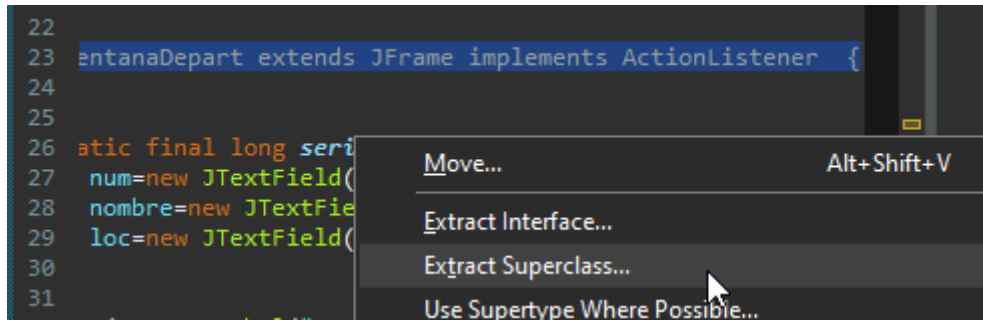
```

Así se ve el explorador de paquetes.



Ejercicio 7:

Empezamos extrayendo una superclase de la clase VentanaDepart.



```
import java.awt.GraphicsConfiguration;

public class SuperclaseDepart extends JFrame {

    public SuperclaseDepart() throws HeadlessException {
        super();
    }

    public SuperclaseDepart(GraphicsConfiguration gc) {
        super(gc);
    }

    public SuperclaseDepart(String title) throws HeadlessException {
        super(title);
    }

    public SuperclaseDepart(String title, GraphicsConfiguration gc) {
        super(title, gc);
    }

}
```

```
public class VentanaDepart extends SuperclaseDepart implements ActionListener
```

Podemos ver los constructores generados, y ahora la clase VentanaDepart extiende a la superclase.