



REFACTORIZACIÓN

Hecho por: Irene Verdeja, Miguel Fernández y Víctor García

Refactorización A - Extraer método

Consiste en fragmentar el código y convertir en métodos los diferentes fragmentos.

```
1 import java.util.Scanner;
2
3 public class RefactorizarA {
4
5     public static void main(String[] args) {
6         Scanner sc=new Scanner(System.in);
7         int numFrases=0;
8         System.out.print("Escriba el numero de frases que vas a introducir: ");
9         numFrases=sc.nextInt();sc.nextLine();
10        String[] frases=new String[numFrases];
11        for(int i=0;i<frases.length;i++) {
12            System.out.print("Escriba la frase numero "+i+": ");
13            frases[i]=sc.nextLine();
14        }
15        for(String cadena:frases) {
16            System.out.println(cadena);
17        }
18        sc.close();
19    }
20}
21 }
```

Código sin refactorizar

```
import java.util.Scanner;
public class RefactorizadoA {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String[] frases=new String[numeroFrases(sc)];
        addFrases(sc,frases);
        imprimirFrases(frases);
        sc.close();
    }

    private static void imprimirFrases(String[] frases) {
        for(String cadena:frases) {
            System.out.println(cadena);
        }
    }

    private static void addFrases(Scanner sc,String[] frases) {
        for(int i=0;i<frases.length;i++) {
            System.out.print("Escriba la frase numero "+i+": ");
            frases[i]=sc.nextLine();
        }
    }

    private static int numeroFrases(Scanner sc) {
        System.out.print("Escriba el numero de frases que vas a introducir: ");
        return Integer.parseInt(sc.nextLine());
    }
}
```

Código refactorizado

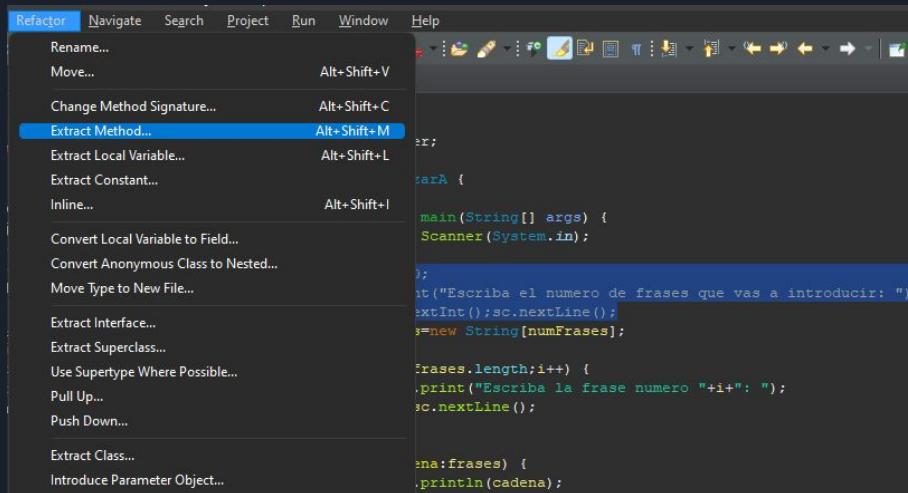
Pasos:

1. Fragmenta el código

```
1 package ED;
2
3 import java.util.Scanner;
4
5 public class RefactorizarA {
6
7     public static void main(String[] args) {
8         Scanner sc=new Scanner(System.in);
9
10        int numFrases=0;
11        System.out.print("Escriba el numero de frases que vas a introducir: ");
12        numFrases=sc.nextInt();sc.nextLine();
13        String[] frases=new String[numFrases];
14
15        for(int i=0;i<frases.length;i++) {
16            System.out.print("Escriba la frase numero "+i+": ");
17            frases[i]=sc.nextLine();
18        }
19
20        for(String cadena:frases) {
21            System.out.println(cadena);
22        }
23
24        sc.close();
25    }
26
27 }
```

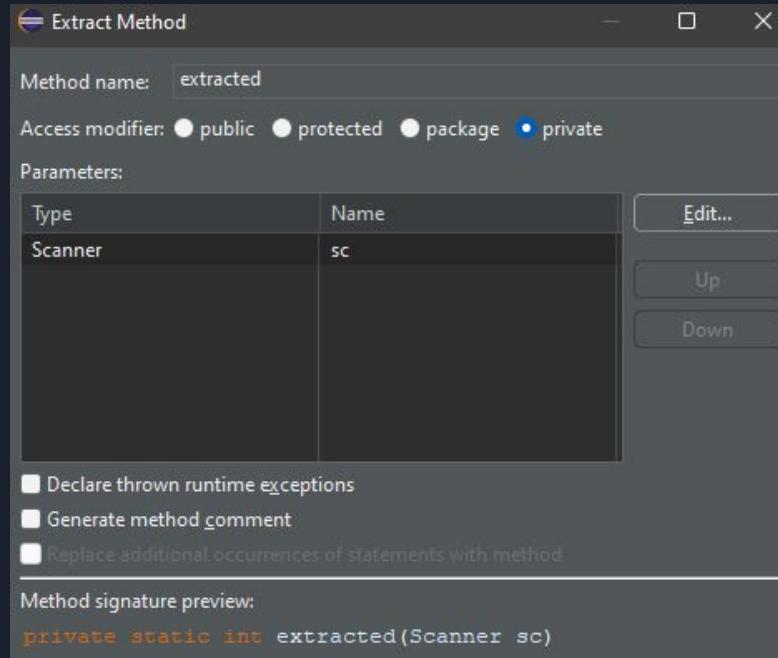
Pasos:

2. Selecciona el fragmento del código a extraer y en el apartado de Refactor, selecciona Extract Method para extraerlo en un método.



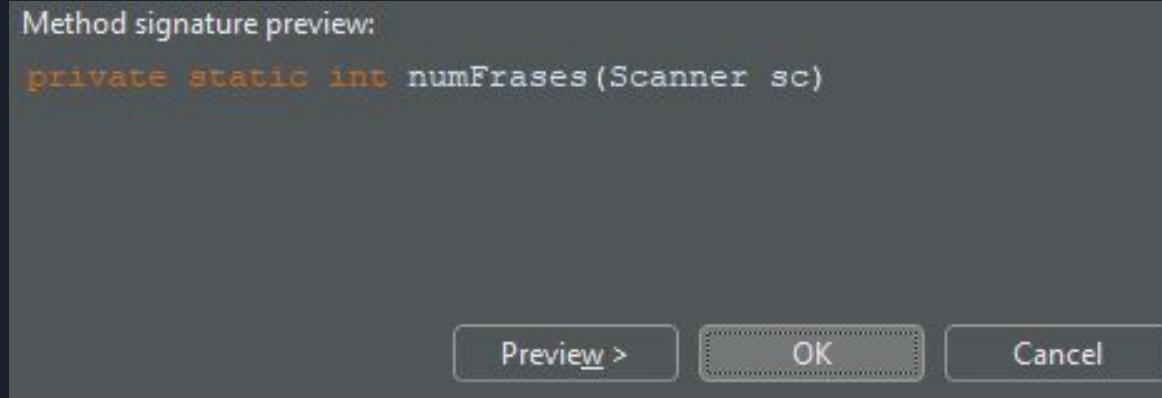
Pasos:

3. Desde el panel de extracción podemos configurar el método que se va crear



Pasos:

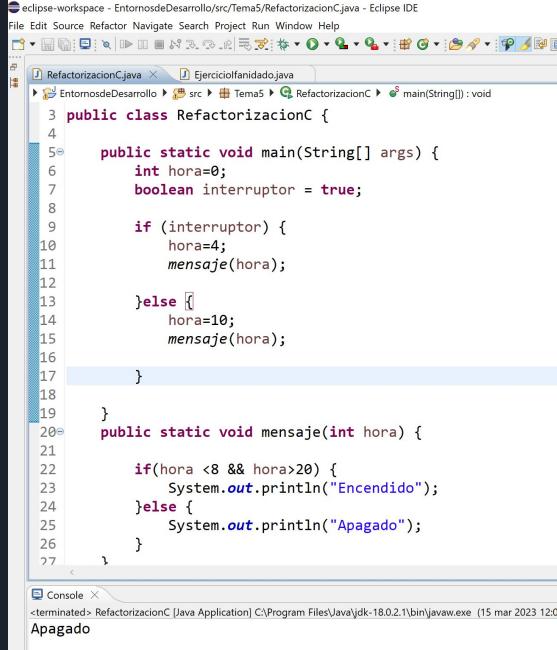
4. En la parte de abajo podremos ver una vista previa del método y finalizar la configuración clicando en Ok



5. Finalmente, realizaremos los pasos anteriores a cada uno de los fragmentos

Refactorización C - Consolidar fragmentos duplicados en condicionales

Consiste en unir fragmentos de código que se repiten en las diferentes partes de un condicional y llevarlos fuera del mismo. Se muestra en el código de la siguiente imagen sin refactorizar.

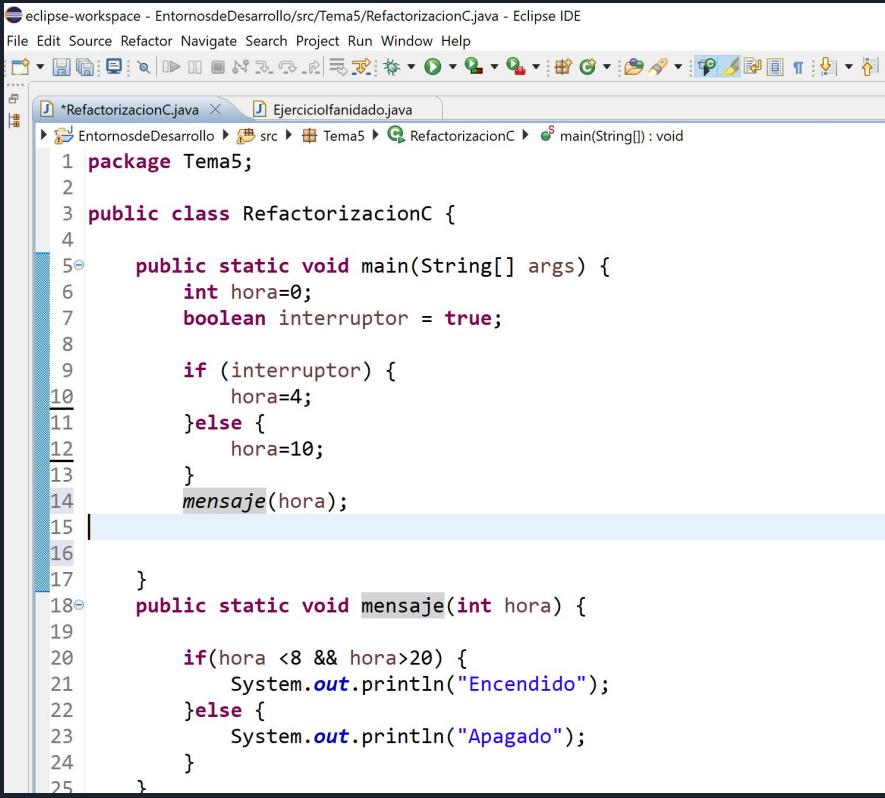


```
eclipse-workspace - EntornoDesarrollo/src/Tema5/RefactorizacionC.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
EjercicioFanidado.java
EntornoDesarrollo src Tema5 RefactorizacionC main(String[])
3 public class RefactorizacionC {
4
5     public static void main(String[] args) {
6         int hora=0;
7         boolean interruptor = true;
8
9         if (interruptor) {
10             hora=4;
11             mensaje(hora);
12
13         }else {
14             hora=10;
15             mensaje(hora);
16         }
17     }
18
19     public static void mensaje(int hora) {
20
21         if(hora <8 && hora>20) {
22             System.out.println("Encendido");
23         }else {
24             System.out.println("Apagado");
25         }
26     }
27 }
```

Console <terminated> RefactorizacionC [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (15 mar 2023 12:03)

Apagado

Por lo que procedemos a eliminar las líneas de código dentro de la condicional if y sacamos el mensaje fuera de dicha condicional, con lo que nos ahorraremos la repetición de esa parte.



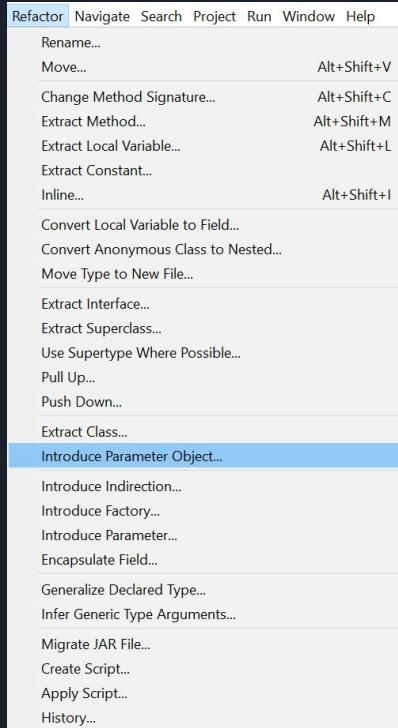
The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - EntornosdeDesarrollo/src/Tema5/RefactorizacionC.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left margin shows line numbers from 1 to 25. The code editor displays the following Java code:

```
1 package Tema5;
2
3 public class RefactorizacionC {
4
5     public static void main(String[] args) {
6         int hora=0;
7         boolean interruptor = true;
8
9         if (interruptor) {
10             hora=4;
11         }else {
12             hora=10;
13         }
14         mensaje(hora);
15     }
16
17 }
18 public static void mensaje(int hora) {
19
20     if(hora <8 && hora>20) {
21         System.out.println("Encendido");
22     }else {
23         System.out.println("Apagado");
24     }
25 }
```

Refactorización O - Reemplazar array con objeto

Paso 1:

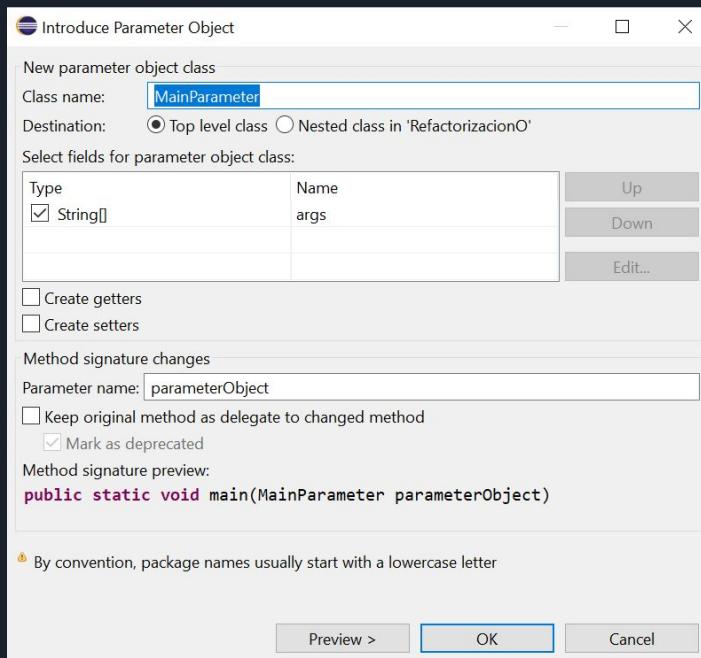
Se empieza a refactorizar, y para ello vamos a subrayar public static void main, y luego le damos a Introduce Parameter Object... :



Refactorización O - Reemplazar array con objeto

Paso 2:

Tras darle a Introduce Parameter Object..., aparece una pequeña ventana en la que se va a crear una nueva clase llamada MainParameter.



The Java code editor displays the generated code for the `MainParameter` class. The code is as follows:

```
1 package EntornosDelDesarrollo;
2
3 public class MainParameter {
4     public String[] args;
5
6     public MainParameter(String[] args) {
7         this.args = args;
8     }
9 }
```

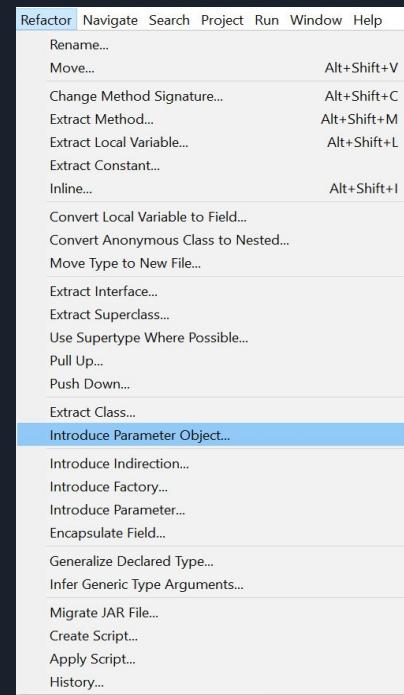
The code editor also shows the project structure in the 'Project Explorer' on the left, including packages like `bin`, `bin.Recupacion`, `bin.RepasoParaLosExamenes`, etc., and files like `RefactorizacionO.java`, `RefactorizarO.java`, and `MainParameter.java`.

Refactorización O - Reemplazar array con objeto

Paso 3:

Después, hacemos los mismos pasos con la clase RefactorizarO. Subrayamos el public RefactorizarO de la clase, y le damos a Refactor y a Introduce Parameter Object....

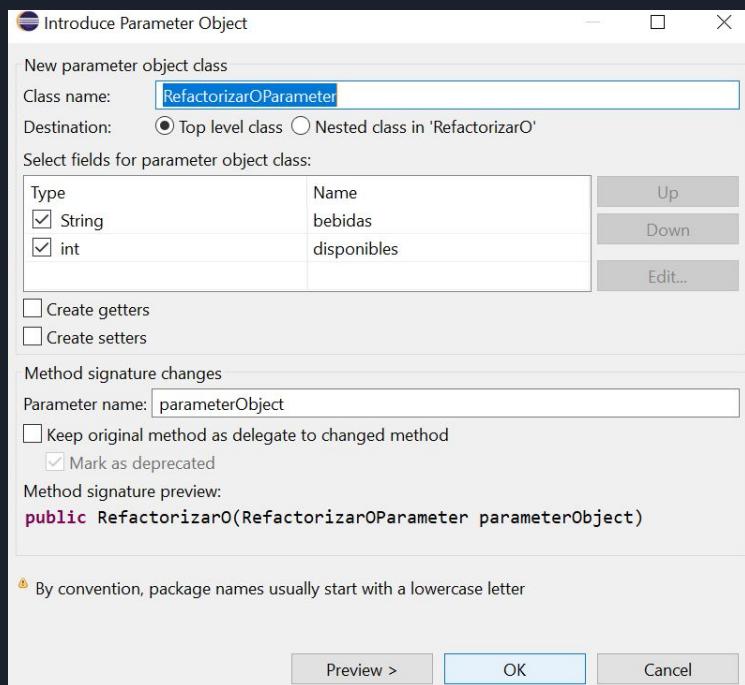
```
8  public RefactorizarO(String bebidas, int disponibles) {  
9      this.bebidas = bebidas;  
10     this.disponibles = disponibles;  
11 }
```



Refactorización O - Reemplazar array con objeto

Paso 4:

Aparece una pequeña ventana en la que se va a crear una nueva clase llamada RefactorizarOParparameter.

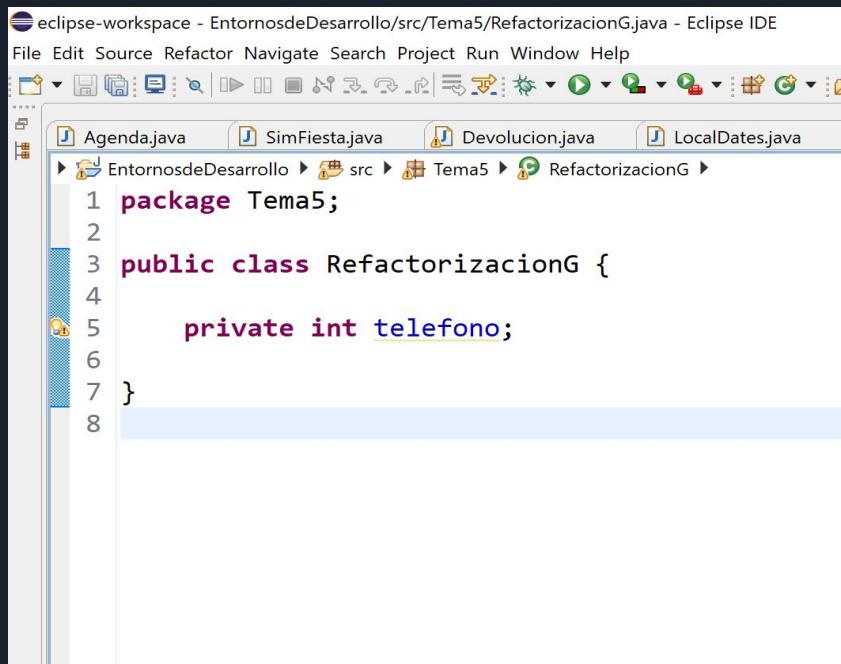


```
8°   public RefactorizarO(RefactorizarOParparameter parameterObject) {  
9     this.bebidas = parameterObject.bebidas;  
10    this.disponibles = parameterObject.disponibles;  
11 }
```

```
RefactorizacionO.java  RefactorizarO.java  RefactorizarOParparameter.java ×  
1 package EntornosDelDesarrollo;  
2  
3 public class RefactorizarOParparameter {  
4   public String bebidas;  
5   public int disponibles;  
6  
7°   public RefactorizarOParparameter(String bebidas, int disponibles) {  
8     this.bebidas = bebidas;  
9     this.disponibles = disponibles;  
10 }  
11 }
```

Refactorización G - Encapsular un atributo

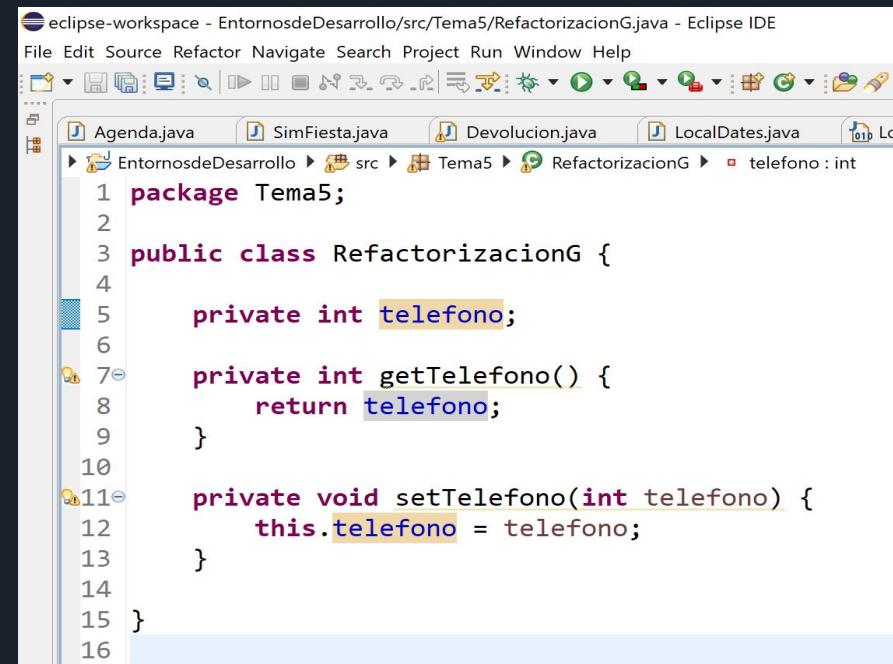
A partir de un atributo público se convierte a privado y le creamos métodos de acceso.



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - EntornosdeDesarrollo/src/Tema5/RefactorizacionG.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left sidebar shows the project structure: EntornosdeDesarrollo > src > Tema5 > RefactorizacionG. The code editor displays the following Java code:

```
1 package Tema5;
2
3 public class RefactorizacionG {
4
5     private int telefono;
6
7 }
8
```

Código sin refactorizar



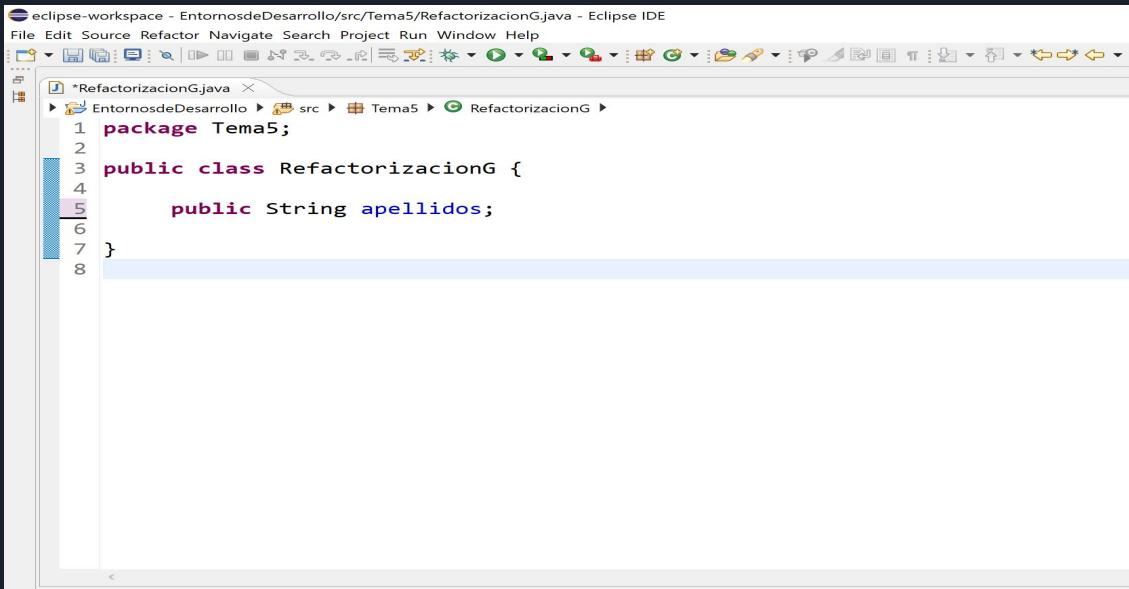
The screenshot shows the Eclipse IDE interface with the same title bar and menu. The code editor now contains the refactored Java code:

```
1 package Tema5;
2
3 public class RefactorizacionG {
4
5     private int telefono;
6
7     private int getTelefono() {
8         return telefono;
9     }
10
11     private void setTelefono(int telefono) {
12         this.telefono = telefono;
13     }
14
15 }
16
```

Código refactorizado

Pasos

1. El patrón de encapsular atributo tiene que ver con refactorizar un atributo público a partir de un código simple.



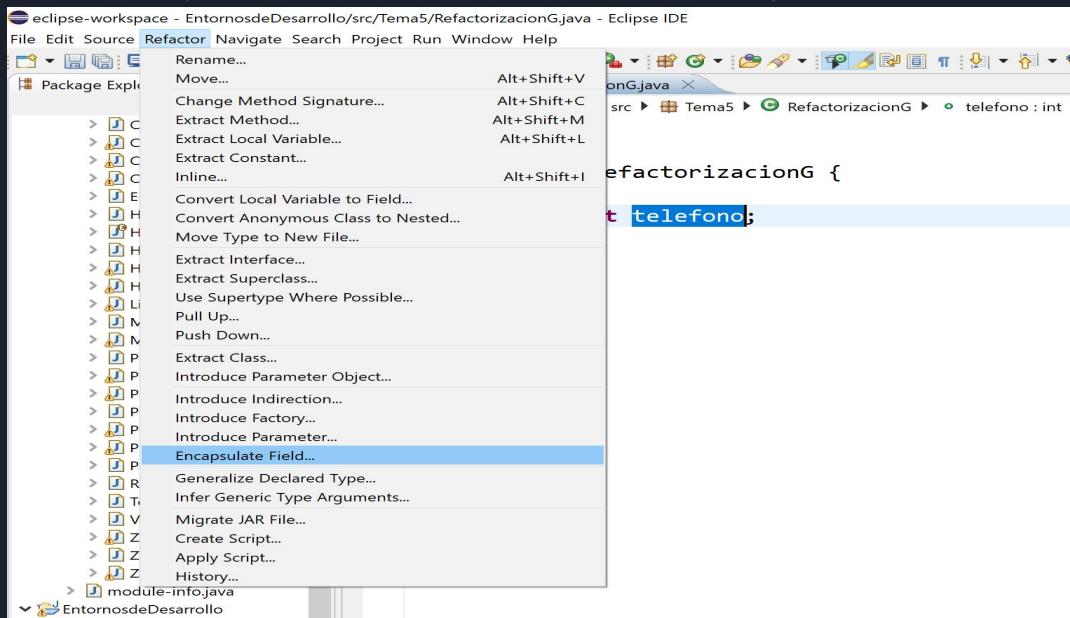
The screenshot shows the Eclipse IDE interface with a Java code editor open. The title bar reads "eclipse-workspace - EntornosdeDesarrollo/src/Tema5/RefactorizacionG.java - Eclipse IDE". The code editor displays the following Java code:

```
1 package Tema5;
2
3 public class RefactorizacionG {
4
5     public String apellidos;
6
7 }
8
```

The line "public String apellidos;" is highlighted with a blue selection bar. The Eclipse toolbar and menu bar are visible at the top of the window.

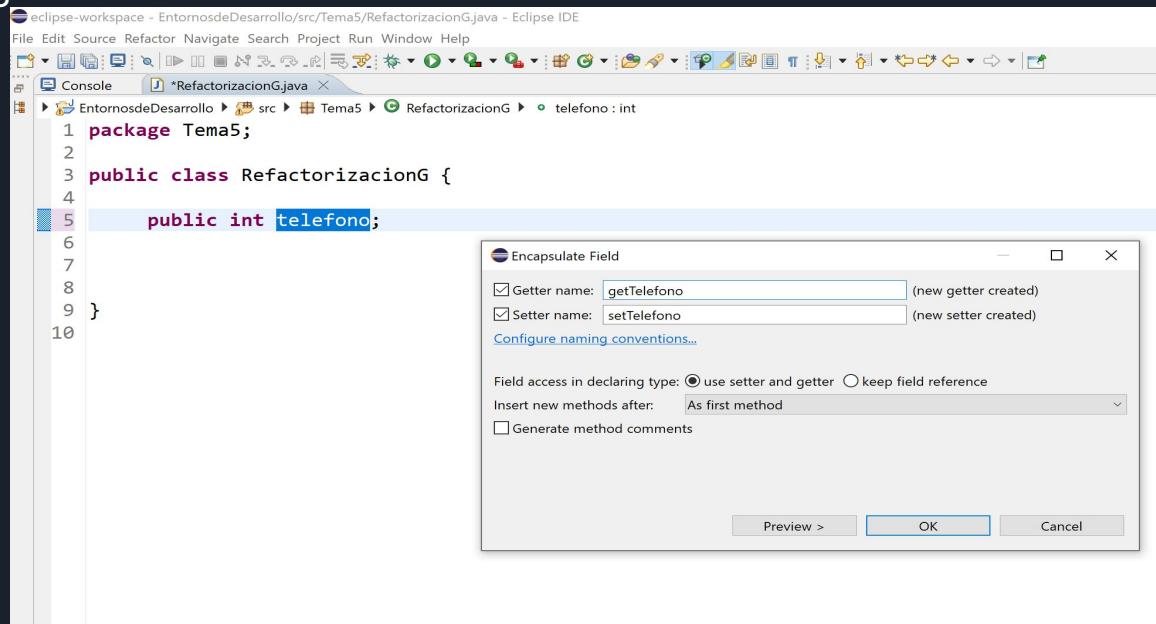
Pasos

2. Seleccionando el nombre del atributo le damos al botón de Refactor en el entorno de desarrollo Eclipse y nos aparecen unas opciones. Elegimos la de Encapsulated Field.



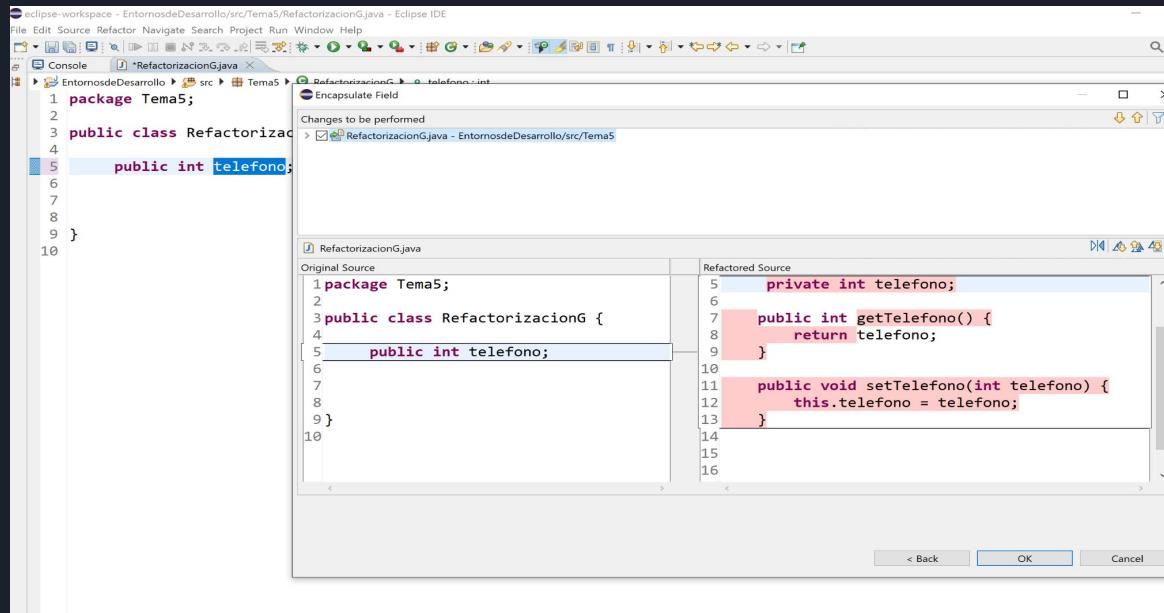
Pasos

3. Aquí aparece una ventana con más opciones que nos permitirán refactorizar el código.



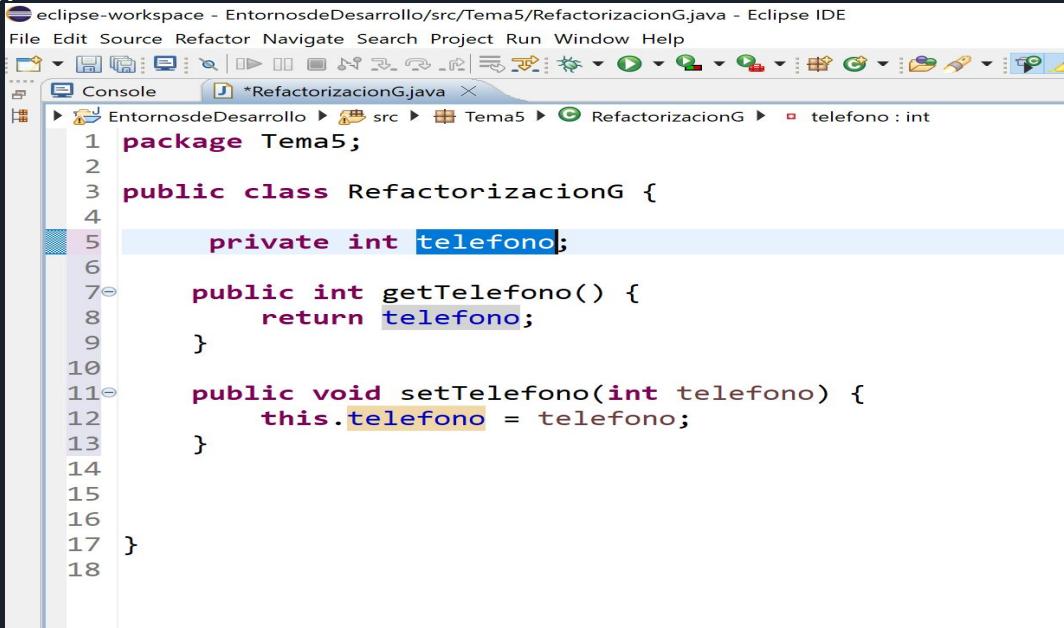
Pasos

4. Le damos al botón de Preview donde se puede apreciar el código antes y después de la refactorización.



Pasos

5. Le damos al botón de OK y se refactoriza el código quedando tal como en la Preview.



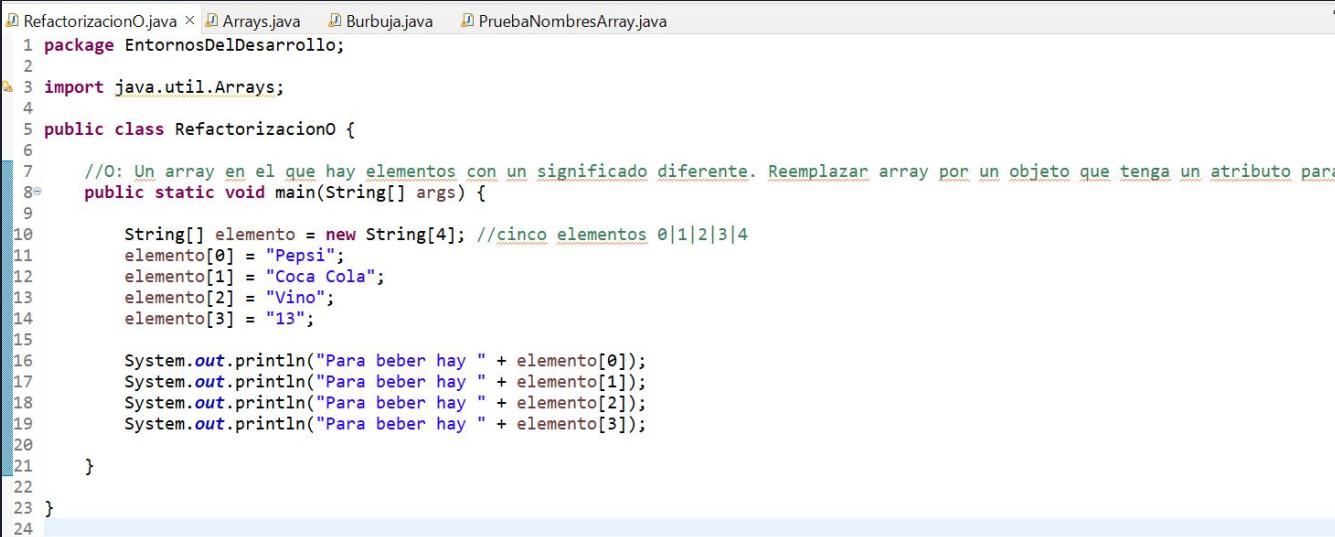
The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - EntornosdeDesarrollo/src/Tema5/RefactorizacionG.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, and Run. The left sidebar shows the project structure: EntornosdeDesarrollo > src > Tema5 > RefactorizacionG > telefono : int. The code editor displays the following Java code:

```
1 package Tema5;
2
3 public class RefactorizacionG {
4
5     private int telefono;
6
7     public int getTelefono() {
8         return telefono;
9     }
10
11    public void setTelefono(int telefono) {
12        this.telefono = telefono;
13    }
14
15
16
17 }
18
```

The variable "telefono" is highlighted in blue, indicating it has been refactored or selected. The code editor has a light gray background with white text and syntax highlighting.

Refactorización O - Reemplazar array con objeto

Consiste en coger un array cuyos elementos tienen un significado distinto, y reemplazarlos por objetos, y que cada objeto tenga un atributo para cada elemento.

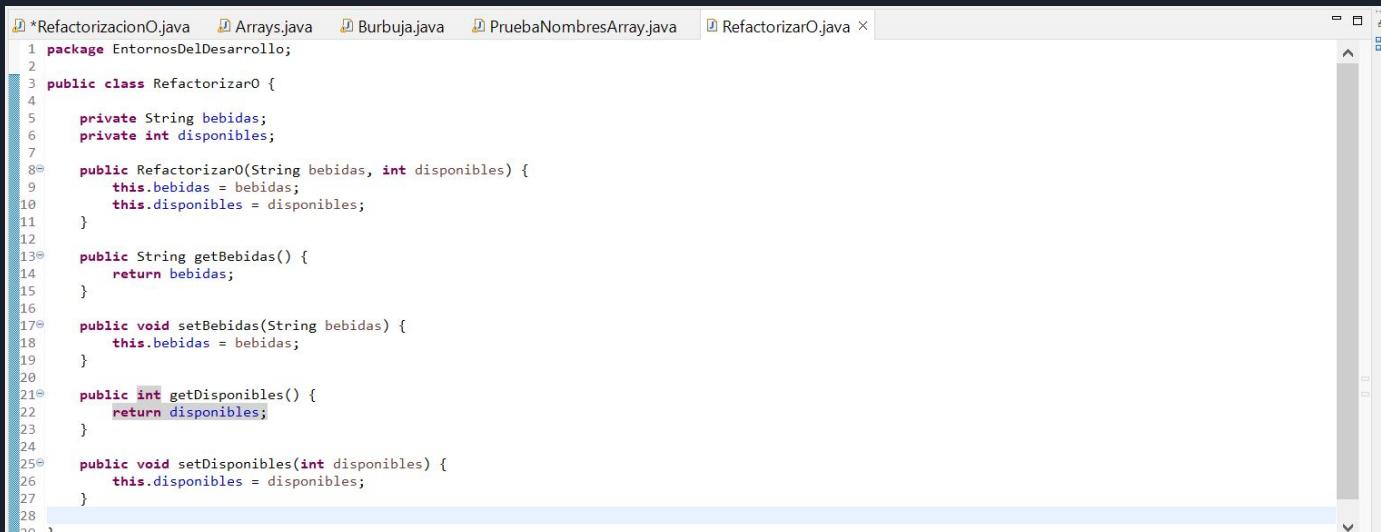


```
RefactorizacionO.java × Arrays.java Burbuja.java PruebaNombresArray.java
1 package EntornosDelDesarrollo;
2
3 import java.util.Arrays;
4
5 public class RefactorizacionO {
6
7     //O: Un array en el que hay elementos con un significado diferente. Reemplazar array por un objeto que tenga un atributo para
8     public static void main(String[] args) {
9
10        String[] elemento = new String[4]; //cinco elementos 0|1|2|3|4
11        elemento[0] = "Pepsi";
12        elemento[1] = "Coca Cola";
13        elemento[2] = "Vino";
14        elemento[3] = "13";
15
16        System.out.println("Para beber hay " + elemento[0]);
17        System.out.println("Para beber hay " + elemento[1]);
18        System.out.println("Para beber hay " + elemento[2]);
19        System.out.println("Para beber hay " + elemento[3]);
20
21    }
22
23 }
24
```

Refactorización O - Reemplazar array con objeto

Consiste en coger un array cuyos elementos tienen un significado distinto, y reemplazarlos por objetos, y que cada objeto tenga un atributo para cada elemento.

Código refactorizado:



```
*RefactorizacionO.java Arrays.java Burbuja.java PruebaNombresArray.java RefactorizarO.java
1 package EntornosDelDesarrollo;
2
3 public class RefactorizarO {
4
5     private String bebidas;
6     private int disponibles;
7
8     public RefactorizarO(String bebidas, int disponibles) {
9         this.bebidas = bebidas;
10        this.disponibles = disponibles;
11    }
12
13     public String getBebidas() {
14         return bebidas;
15     }
16
17     public void setBebidas(String bebidas) {
18         this.bebidas = bebidas;
19     }
20
21     public int getDisponibles() {
22         return disponibles;
23     }
24
25     public void setDisponibles(int disponibles) {
26         this.disponibles = disponibles;
27     }
28 }
```

JAVADOC

```
package pktSiete;

import java.time.LocalDate;
/**
 * Clase para definir una fiesta
 *
 * @author irene
 * @version 1.0
 * @since Marzo 2023
 */
public class SimFiesta {
    private String nombre;
    private String localidad;
    private LocalDate fInicio;
    private LocalDate fFin;

    /**
     * Constructor vacio
     */
    public SimFiesta() {
        this.nombre="";
        this.localidad="";
        this.fInicio=null;
        this.fFin=null;
    }

    /**
     * Constructor con parametros
     * @param nombre String. Nombre de la fiesta.
     * @param localidad String. Localidad donde se realiza la fiesta.
     * @param fInicio LocalDate. Fecha de inicio de la fiesta.
     * @param fFin LocalDate. Fecha de fin de la fiesta.
     */
    public SimFiesta(String nombre, String localidad, LocalDate fInicio, LocalDate fFin) {
        this.nombre=nombre;
        this.localidad=localidad;
        this.fInicio=fInicio;
        this.fFin=fFin;//chachi pirulis
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

MODULE PACKAGE CLASS USE TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

SEARCH: Search

Module Excepciones
Package pktSiete

Class SimFiesta

java.lang.Object²
 pktSiete.SimFiesta

```
public class SimFiesta
extends Object2

Clase para definir una fiesta
```

Since:
Marzo 2023

Version:
1.0

Author:
irene

Constructor Summary

Constructors

Constructor	Description
SimFiesta()	Constructor vacio
SimFiesta(String ² nombre, String ² localidad, LocalDate ² fInicio, LocalDate ² fFin)	Constructor con parametros

Method Summary

All Methods Instance Methods Concrete Methods

MODULE PACKAGE CLASS USE TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

SEARCH: Search

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
LocalDate ²	getffFin()	
LocalDate ²	getfInicio()	
String ²	getLocalidad()	
String ²	getNombre()	
void	setffFin(LocalDate ² fFin)	
void	setfInicio(LocalDate ² fInicio)	
void	setLocalidad(String ² localidad)	
void	setNombre(String ² nombre)	
String ²	toString()	

Methods inherited from class java.lang.Object²

equals², getClass², hashCode², notify², notifyAll², wait², wait², wait²

Constructor Details

SimFiesta

public SimFiesta()

Constructor vacío

Constructor Details**SimFiesta**

```
public SimFiesta()
```

Constructor vacio

SimFiesta

```
public SimFiesta(String✉ nombre,  
                 String✉ localidad,  
                 LocalDate✉ fInicio,  
                 LocalDate✉ fFin)
```

Constructor con parametros

Parameters:

nombre - String. Nombre de la fiesta.

localidad - String. Localidad donde se realiza la fiesta.

fInicio - LocalDate. Fecha de inicio de la fiesta.

fFin - LocalDate. Fecha de fin de la fiesta.

Method Details**getNombre**

```
public String✉ getNombre()
```

Method Details**getNombre**

```
public String2 getNombre()
```

setNombre

```
public void setNombre(String2 nombre)
```

getLocalidad

```
public String2 getLocalidad()
```

setLocalidad

```
public void setLocalidad(String2 localidad)
```

getfInicio

```
public LocalDate2 getfInicio()
```

setfInicio

```
public void setfInicio(LocalDate2 fInicio)
```

getfFin