

# **Refactor 2**

## **Ejercicio 21**

```
1 package Ejercicios;
2
3 public class Ejercicio21 {
4     public static void main(String[] args) {
5         try {
6             Ejercicio21 ejercicio21 = new Ejercicio21(); // Crear una instancia de Ejercicio21
7             ejercicio21.titulo(); // Imprimir el banner
8             ejercicio21.codigo(); // Ejecutar el método de prueba para comparar resultados
9         } catch (Exception e) {
10             printError();
11         }
12     }
13
14     // Métodos de funcionalidad principal
15     void codigo() {
16         Pedido pedido = new Pedido();
17         titulo1();
18         System.out.println("Comparación de resultados:");
19         System.out.println("");
20         NoRef();
21         System.out.println(precio(pedido));
22         System.out.println();
23         Ref();
24         System.out.println(refactorPrecio(pedido));
25         System.out.println();
26     }
27
28     // Métodos de impresión de títulos y mensajes
29     private void titulo() {
30         System.out.println();
31         System.out.println("Patrón de refactorización usual 21:\n\nUso de refactorización para mejorar la claridad y mantenimiento del código");
32         System.out.println("RESOLUCIÓN DEL EJ21");
33         System.out.println();
34     }
35
36     private void titulo1() {
37         System.out.println(" " + "-----");
38         System.out.println(" " + " Prueba de métodos refactorizados y no refactorizados: ");
39         System.out.println(" " + "-----");
40     }
41
42     static void printError() {
43         System.out.println();
44         System.out.println(" " + "-----");
45         System.out.println(" " + " Salida forzada ");
46         System.out.println(" " + "-----");
47         System.out.println();
48     }
49
50     void NoRef() {
51         System.out.println();
52         System.out.println(" " + "-----");
53         System.out.println(" " + " Método no refactorizado ");
54         System.out.println(" " + "-----");
55         System.out.println();
56     }
57
58     void Ref() {
59         System.out.println();
60         System.out.println(" " + "-----");
61         System.out.println(" " + " Método refactorizado ");
62         System.out.println(" " + "-----");
63         System.out.println();
64     }
65
66     // Método no refactorizado
67     boolean precio(Pedido pedido) {
68         double precioBase = pedido.precioBase();
69         return (precioBase > 100);
70     }
71
72     // Método refactorizado
73     boolean refactorPrecio(Pedido pedido) {
74         return (pedido.precioBase() > 100);
75     }
76
77     // Método de prueba
78     void test(Pedido pedido) {
79         System.out.println("Comparación de resultados: ");
80         System.out.println();
81         NoRef();
82         System.out.println(precio(pedido));
83         Ref();
84         System.out.println(refactorPrecio(pedido));
85     }
86 }
```

```
87     // Definición de la clase Pedido
88     class Pedido {
89         private int precioBase = 101;
90
91         int precioBase() {
92             return precioBase;
93         }
94     }
95 }
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main
● > c:; cd 'c:\Users\zhiya\Desktop\IMF\Primer Año\David\Entornos de Desarrollo (90 horas)\'Entornos de Desarrollo'; & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio21'
```

Patrón de refactorización usual 21:

Uso de refactorización para mejorar la claridad y mantenimiento del código.

#### RESOLUCIÓN DEL EJ21

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Comparación de resultados:

-----  
Método no refactorizado  
-----

true

-----  
Método refactorizado  
-----

true

## **Ejercicio 22**

```

1 package Ejercicios;
2 public class Ejercicio22 {
3     // Variables de instancia para cantidad y precio
4     private double cantidad;
5     private double precioUnitario;
6
7     public static void main(String[] args) {
8         try {
9             Ejercicio22 ejercicio22 = new Ejercicio22(); // Crear una instancia de Ejercicio22
10            ejercicio22.titulo(); // Imprimir el banner
11            ejercicio22.codigo(); // Ejecutar el método de prueba para comparar resultados
12        } catch (Exception e) {
13            printError();
14        }
15    }
16
17    // Métodos de funcionalidad principal
18    void codigo() {
19        cantidad = 150;
20        precioUnitario = 10;
21
22        titulo1();
23        System.out.println("Comparación de resultados:");
24        System.out.println();
25        NoRef();
26        double resultadoAntes = antesRefactorizar(cantidad, precioUnitario);
27        System.out.println("Resultado sin refactorizar: " + resultadoAntes);
28        System.out.println();
29        Ref();
30        double resultadoDespues = despuesRefactorizar();
31        System.out.println("Resultado refactorizado: " + resultadoDespues);
32    }
33
34    // Métodos de impresión de títulos y mensajes
35    private void titulo() {
36        System.out.println();
37        System.out.println("Patrón de refactorización usual 22:\n\nSeparación de lógica de cálculo:\nRefactoriza métodos para mejorar su claridad y mantenimiento");
38        System.out.println("RESOLUCIÓN DEL EJ22");
39        System.out.println();
40    }
41
42    private void titulo1() {
43        System.out.println("      " + "-----");
44        System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
45        System.out.println("      " + "-----");
46    }
47
48    static void printError() {
49        System.out.println();
50        System.out.println("      " + "-----");
51        System.out.println("      " + " Salida forzada   ");
52        System.out.println("      " + "-----");
53        System.out.println();
54    }
55
56    void NoRef() {
57        System.out.println();
58        System.out.println("      " + "-----");
59        System.out.println("      " + " Método no refactorizado  ");
60        System.out.println("      " + "-----");
61        System.out.println();
62    }
63
64    void Ref() {
65        System.out.println();
66        System.out.println("      " + "-----");
67        System.out.println("      " + " Método refactorizado  ");
68        System.out.println("      " + "-----");
69        System.out.println();
70    }
71
72    // Método antesRefactorizar que calcula el precio final de la cantidad y el precio
73    public double antesRefactorizar(double cantidad, double precioUnitario) {
74        // Calcular el precio base
75        double preciobase = cantidad * precioUnitario;
76        // Aplicar el descuento basado en el precio base
77        if (preciobase > 1000) {
78            // Descuento del 10% si el precio base es mayor a 1000
79            return preciobase * 0.90;
80        } else {
81            // Descuento del 5% si el precio base es menor o igual a 1000
82            return preciobase * 0.95;
83        }
84    }
85
86    // Método despuesRefactorizar que también calcula el precio final pero usando el método precioBase()
87    public double despuesRefactorizar() {
88        // Aplicar el descuento basado en el precio base calculado por precioBase()
89        if (precioBase() > 1000) {
90            // Descuento del 10% si el precio base es mayor a 1000
91            return precioBase() * 0.90;
92        } else {
93            // Descuento del 5% si el precio base es menor o igual a 1000
94            return precioBase() * 0.95;
95        }
96    }
97
98    // Método precioBase que calcula el precio base multiplicando la cantidad por el precio unitario

```

```
99     public double precioBase() {  
100         return cantidad * precioUnitario;  
101     }  
102 }  
103
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos  
de Desarrollo (90 horas) > Entornos de Desarrollo > main  
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\  
java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\User  
s\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb1936667  
9bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e  
33\bin' 'Ejercicios.Ejercicio22'
```

Patrón de refactorización usual 22:

Separación de lógica de cálculo:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ22

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Comparación de resultados:

-----  
Método no refactorizado  
-----

Resultado sin refactorizar: 1350.0

-----  
Método refactorizado  
-----

Resultado refactorizado: 1350.0

## **Ejercicio 23**

```
1 package Ejercicios;
2 public class Ejercicio23 {
3     int cantidad = 10;
4     int precio = 500;
5
6     public static void main(String[] args) {
7         try {
8             Ejercicio23 ejercicio23 = new Ejercicio23(); // Crear una instancia de Ejercicio23
9             ejercicio23.titulo(); // Imprimir el banner
10            ejercicio23.codigo(); // Ejecutar el método de prueba para comparar resultados
11        } catch (Exception e) {
12            printError();
13        }
14    }
15
16    // Métodos de funcionalidad principal
17    void codigo() {
18        titulo();
19        System.out.println("Comparación de resultados:");
20        System.out.println();
21        NoRef();
22        System.out.println("Resultado sin refactorizar: " + noRefactor());
23        System.out.println();
24        Ref();
25        System.out.println("Resultado refactorizado: " + refactor());
26        System.out.println();
27        test();
28    }
29
30    // Métodos de impresión de títulos y mensajes
31    private void titulo() {
32        System.out.println();
33        System.out.println("Patrón de refactorización usual 23:\n\nSeparación de lógica de cálculo:\n Refactoriza métodos para mejorar su claridad y mantenimiento.\n");
34        System.out.println("RESOLUCIÓN DEL EJ23");
35        System.out.println();
36    }
37
38    private void titulo() {
39        System.out.println(" " + "-----");
40        System.out.println(" " + " Prueba de métodos refactorizados y no refactorizados: ");
41        System.out.println(" " + "-----");
42    }
43
44    static void printError() {
45        System.out.println();
46        System.out.println(" " + "-----");
47        System.out.println(" " + " Salida forzada ");
48        System.out.println(" " + "-----");
49        System.out.println();
50    }
51
52    void NoRef() {
53        System.out.println();
54        System.out.println(" " + "-----");
55        System.out.println(" " + " Método no refactorizado ");
56        System.out.println(" " + "-----");
57        System.out.println();
58    }
59
60    void Ref() {
61        System.out.println();
62        System.out.println(" " + "-----");
63        System.out.println(" " + " Método refactorizado ");
64        System.out.println(" " + "-----");
65        System.out.println();
66    }
67
68    double noRefactor() {
69        final double precioBase;
70        precioBase = cantidad * precio;
71        return precioBase;
72    }
73
74    double refactor() {
75        final int precioBase = cantidad * precio;
76        final double descuento;
77        if (precioBase > 1000) {
78            descuento = 0.9;
79        } else {
80            descuento = 0.95;
81        }
82        return precioBase * descuento;
83    }
84
85    void test() {
86        System.out.println("Comparación de resultados:");
87        System.out.println();
88        System.out.println("Resultado sin refactorizar: " + noRefactor());
89        System.out.println("Resultado refactorizado: " + refactor());
90    }
91}
92
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main >
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bf a14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\b in' 'Ejercicios.Ejercicio23'
```

Patrón de refactorización usual 23:

Separación de lógica de cálculo:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ23

---

Prueba de métodos refactorizados y no refactorizados:

---

Comparación de resultados:

---

Método no refactorizado

---

Resultado sin refactorizar: 5000.0

---

Método refactorizado

---

Resultado refactorizado: 4500.0

Comparación de resultados:

Resultado sin refactorizar: 5000.0

Resultado refactorizado: 4500.0

# Ejercicio 24

```
1 package Ejercicios;
2 public class Ejercicio24 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio24 ejercicio24 = new Ejercicio24(); // Crear una instancia de Ejercicio24
6             ejercicio24.titulo(); // Imprimir el banner
7             ejercicio24.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13 // Métodos de funcionalidad principal
14 void codigo() {
15     String idioma1 = "RUS";
16     String idioma2 = "ALE";
17     int nivelingles = 1;
18
19     titulo1();
20     System.out.println("Comparación de resultados:");
21     System.out.println();
22     NoRef();
23     antesRefactorizar(idioma1, idioma2, nivelingles);
24     System.out.println();
25     Ref();
26     despuesRefactorizar(idioma1, idioma2, nivelingles);
27 }
28
29 // Métodos de impresión de títulos y mensajes
30 private void titulo() {
31     System.out.println();
32     System.out.println("Patrón de refactorización usual 24:\n\nUso de variables autoexplicativas:\n Refactoriza métodos para mejorar su claridad y mantenimiento");
33     System.out.println("RESOLUCIÓN DEL EJ24");
34     System.out.println();
35 }
36
37 private void titulo1() {
38     System.out.println("      " + "-----");
39     System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
40     System.out.println("      " + "-----");
41 }
42
43 static void printError() {
44     System.out.println();
45     System.out.println("      " + "-----");
46     System.out.println("      " + " Salida forzada      ");
47     System.out.println("      " + "-----");
48     System.out.println();
49 }
50
51 void NoRef() {
52     System.out.println();
53     System.out.println("      " + "-----");
54     System.out.println("      " + " Método no refactorizado  ");
55     System.out.println("      " + "-----");
56     System.out.println();
57 }
58
59 void Ref() {
60     System.out.println();
61     System.out.println("      " + "-----");
62     System.out.println("      " + " Método refactorizado  ");
63     System.out.println("      " + "-----");
64     System.out.println();
65 }
66
67 public static void antesRefactorizar(String idioma1, String idioma2, int nivelingles) {
68     if ((idioma1.toUpperCase().indexOf("RUS") > -1) &&
69         (idioma2.toUpperCase().indexOf("ALE") > -1) &&
70         (nivelingles > 0)) {
71         System.out.println("MENSAJES EN INGLES");
72     } else {
73         System.out.println("MENSAJES EN OTRO IDIOMA");
74     }
75 }
76
77 public static void despuesRefactorizar(String idioma1, String idioma2, int nivelingles) {
78     final boolean esRuso = idioma1.toUpperCase().indexOf("RUS") > -1;
79     final boolean esAleman = idioma2.toUpperCase().indexOf("ALE") > -1;
80     final boolean ingles = nivelingles > 0;
81
82     if (esRuso && esAleman && ingles) {
83         System.out.println("MENSAJES EN INGLES");
84     } else {
85         System.out.println("MENSAJES EN OTRO IDIOMA");
86     }
87 }
88 }
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main >
● > & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bf a14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio24'
```

Patrón de refactorización usual 24:

Uso de variables autoexplicativas:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ24

---

Prueba de métodos refactorizados y no refactorizados:

---

Comparación de resultados:

---

Método no refactorizado

---

MENSAJES EN INGLES

---

Método refactorizado

---

MENSAJES EN INGLES

# Ejercicio 25

```
1 package Ejercicios;
2 public class Ejercicio25 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio25 ejercicio25 = new Ejercicio25(); // Crear una instancia de Ejercicio25
6             ejercicio25.titulo(); // Imprimir el banner
7             ejercicio25.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13 // Métodos de funcionalidad principal
14 void codigo() {
15     titulo();
16     System.out.println("Comparación de resultados:");
17     System.out.println();
18     NoRef();
19     noRefactor();
20     System.out.println();
21     Ref();
22     refactor();
23     System.out.println();
24     test();
25 }
26
27 // Métodos de impresión de títulos y mensajes
28 private void titulo() {
29     System.out.println();
30     System.out.println("Patrón de refactorización usual 25:\n\nUso de constantes y variables autoexplicativas:\n Refactoriza métodos para mejorar su claridad y mantenimiento");
31     System.out.println("RESOLUCIÓN DEL EJ25");
32     System.out.println();
33 }
34
35 private void titulo() {
36     System.out.println("      " + "-----");
37     System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
38     System.out.println("      " + "-----");
39 }
40
41 static void printError() {
42     System.out.println();
43     System.out.println("      " + "-----");
44     System.out.println("      " + " Salida forzada      ");
45     System.out.println("      " + "-----");
46     System.out.println();
47 }
48
49 void NoRef() {
50     System.out.println();
51     System.out.println("      " + "-----");
52     System.out.println("      " + " Método no refactorizado  ");
53     System.out.println("      " + "-----");
54     System.out.println();
55 }
56
57 void Ref() {
58     System.out.println();
59     System.out.println("      " + "-----");
60     System.out.println("      " + " Método refactorizado  ");
61     System.out.println("      " + "-----");
62     System.out.println();
63 }
64
65 void noRefactor() {
66     int radio = 5;
67     double tmp = Math.PI * radio * radio;
68     System.out.println(tmp);
69     tmp = 2 * Math.PI * radio;
70     System.out.println(tmp);
71 }
72
73 void refactor() {
74     int radio = 5;
75     final double area = Math.PI * radio * radio;
76     System.out.println(area);
77     final double perimetro = 2 * Math.PI * radio;
78     System.out.println(perimetro);
79 }
80
81 void test() {
82     System.out.println("Comparación de resultados:");
83     System.out.println();
84     noRefactor();
85     refactor();
86 }
87 }
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main
● > & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bf a14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\b in' 'Ejercicios.Ejercicio25'
```

Patrón de refactorización usual 25:

Uso de constantes y variables autoexplicativas:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ25

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Comparación de resultados:

-----  
Método no refactorizado  
-----

78.53981633974483  
31.41592653589793

-----  
Método refactorizado  
-----

78.53981633974483  
31.41592653589793

Comparación de resultados:

78.53981633974483  
31.41592653589793  
78.53981633974483

31.41592653589793

## **Ejercicio 26**

```
1 package Ejercicios;
2 public class Ejercicio26 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio26 ejercicio26 = new Ejercicio26(); // Crear una instancia de Ejercicio26
6             ejercicio26.titulo(); // Imprimir el banner
7             ejercicio26.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13    // M\'etodos de funcionalidad principal
14    void codigo() {
15        int horas = 40;
16        int horasExtra = 10;
17        int salarioBase = 20;
18
19        titulo1();
20        System.out.println("Comparaci\'on de resultados:");
21        System.out.println();
22        NoRef();
23        int salarioAntes = antesRefactorizar(horas, horasExtra, salarioBase);
24        System.out.println("Salario antes de refactorizar: " + salarioAntes);
25        System.out.println();
26        Ref();
27        int salarioDespues = despuesRefactorizar(horas, horasExtra, salarioBase);
28        System.out.println("Salario despu\'s de refactorizar: " + salarioDespues);
29        System.out.println();
30        test(horas, horasExtra, salarioBase);
31    }
32
33    // M\'etodos de impresi\'on de t\'itulos y mensajes
34    private void titulo() {
35        System.out.println();
36        System.out.println("Patr\'on de refactorizaci\'on usual 26:\n\nUso correcto de par\'ametros:\n Refactoriza m\'etodos para mejorar su claridad y mantenimiento.\n");
37        System.out.println("RESOLuci\'on DEL EJ26");
38        System.out.println();
39    }
40
41    private void titulo1() {
42        System.out.println("-----");
43        System.out.println(" Prueba de m\'etodos refactorizados y no refactorizados ");
44        System.out.println("-----");
45    }
46
47    static void printError() {
48        System.out.println();
49        System.out.println("-----");
50        System.out.println("----- Salida forzada -----");
51        System.out.println("-----");
52        System.out.println();
53    }
54
55    void NoRef() {
56        System.out.println();
57        System.out.println("-----");
58        System.out.println("----- M\'etodo no refactorizado -----");
59        System.out.println("-----");
60        System.out.println();
61    }
62
63    void Ref() {
64        System.out.println();
65        System.out.println("-----");
66        System.out.println("----- M\'etodo refactorizado -----");
67        System.out.println("-----");
68        System.out.println();
69    }
70
71    public int antesRefactorizar(int horas, int horasExtra, int salarioBase) {
72        // Mal uso de par\'ametros: 'horas'
73        horas = horas + (int) (horasExtra * 1.5);
74        return horas * salarioBase;
75    }
76
77    public int despuesRefactorizar(int horas, int horasExtra, int salarioBase) {
78        // Uso correcto de par\'ametros: variable final
79        final int horasTrabajadas = horas + (int) (horasExtra * 1.5);
80        return horasTrabajadas * salarioBase;
81    }
82
83    void test(int horas, int horasExtra, int salarioBase) {
84        System.out.println("Comparaci\'on de resultados:");
85        System.out.println();
86        int salarioAntes = antesRefactorizar(horas, horasExtra, salarioBase);
87        System.out.println("Salario antes de refactorizar: " + salarioAntes);
88        int salarioDespues = despuesRefactorizar(horas, horasExtra, salarioBase);
89        System.out.println("Salario despu\'s de refactorizar: " + salarioDespues);
90    }
91 }
92 }
```

```
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio26'
```

Patrón de refactorización usual 26:

Uso correcto de parámetros:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ26

---

---

Prueba de métodos refactorizados y no refactorizados:

---

Comparación de resultados:

---

Método no refactorizado

---

Salario antes de refactorizar: 1100

---

Método refactorizado

---

Salario después de refactorizar: 1100

Comparación de resultados:

Salario antes de refactorizar: 1100

Salario después de refactorizar: 1100

## **Ejercicio 27**

```
● ● ●
1 package Ejercicios;
2 public class Ejercicio27 {
3     private String[] animales = { "Gato", "Perro", "Loro", "Cerdo", "Tortuga" };
4     private String[] animales2 = { "Gato", "Cerda", "Loro", "Vaca", "Tortuga" };
5
6     public static void main(String[] args) {
7         try {
8             Ejercicio27 ejercicio27 = new Ejercicio27(); // Crear una instancia de Ejercicio27
9             ejercicio27.titulo(); // Imprimir el banner
10            ejercicio27.codigo(); // Ejecutar el método de prueba para comparar resultados
11        } catch (Exception e) {
12            printError();
13        }
14    }
15
16    // Métodos de funcionalidad principal
17    void codigo() {
18        titulo1();
19        System.out.println("Comparación de resultados:");
20        System.out.println();
21        NoRef();
22        System.out.println("Resultado sin refactorizar: " + buscarAnimal(animales));
23        System.out.println();
24        Ref();
25        System.out.println("Resultado refactorizado: " + refactorBuscarAnimal2(animales2));
26        System.out.println();
27        test();
28    }
29
30    // Métodos de impresión de títulos y mensajes
31    private void titulo() {
32        System.out.println();
33        System.out.println("Patrón de refactorización usual 27:\n\nUso de bucles mejorados:\n Refactoriza métodos para mejorar su claridad y mantenimiento.\n");
34        System.out.println("RESOLUCIÓN DEL EJ27");
35        System.out.println();
36    }
37
38    private void titulo1() {
39        System.out.println("      " + "-----");
40        System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
41        System.out.println("      " + "-----");
42    }
43
44    static void printError() {
45        System.out.println();
46        System.out.println("      " + "-----");
47        System.out.println("      " + " Salida forzada   ");
48        System.out.println("      " + "-----");
49        System.out.println();
50    }
51
52    void NoRef() {
53        System.out.println();
54        System.out.println("      " + "-----");
55        System.out.println("      " + " Método no refactorizado  ");
56        System.out.println("      " + "-----");
57        System.out.println();
58    }
59
60    void Ref() {
61        System.out.println();
62        System.out.println("      " + "-----");
63        System.out.println("      " + " Método refactorizado  ");
64        System.out.println("      " + "-----");
65        System.out.println();
66    }
67
68    String buscarAnimal(String[] animales) {
69        for (int indice = 0; indice < animales.length; indice++) {
70            if (animales[indice].equals("Perro")) {
71                return "Perro";
72            }
73            if (animales[indice].equals("Tortuga")) {
74                return "Tortuga";
75            }
76            if (animales[indice].equals("Loro")) {
77                return "Loro";
78            }
79        }
80        return "No encontrado";
81    }
82
83    String refactorBuscarAnimal2(String[] animales) {
84        for (String animal : animales) {
85            if (animal.equals("Perro")) {
86                return "Perro";
87            }
88            if (animal.equals("Tortuga")) {
89                return "Tortuga";
90            }
91            if (animal.equals("Loro")) {
92                return "Loro";
93            }
94        }
95    }
96}
```

```
94     }
95     return "No encontrado";
96 }
97
98 void test() {
99     System.out.println("Comparación de resultados:");
100    System.out.println();
101    System.out.println("Resultado sin refactorizar: " + buscarAnimal(animales));
102    System.out.println("Resultado refactorizado: " + refactorBuscarAnimal2(animales2));
103 }
104 }
105
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main
● > & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhuya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio27'
```

Patrón de refactorización usual 27:

Uso de bucles mejorados:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ27

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Comparación de resultados:

-----  
Método no refactorizado  
-----

Resultado sin refactorizar: Perro

-----  
Método refactorizado  
-----

Resultado refactorizado: Loro

Comparación de resultados:

Resultado sin refactorizar: Perro

Resultado refactorizado: Loro

## **Ejercicio 28**

```
1 package Ejercicios;
2 public class Ejercicio28 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio28 ejercicio28 = new Ejercicio28(); // Crear una instancia de Ejercicio28
6             ejercicio28.titulo(); // Imprimir el banner
7             ejercicio28.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13    // Métodos de funcionalidad principal
14    void codigo() {
15        titulo1();
16        System.out.println("Comparación de resultados:");
17        System.out.println();
18        NoRef();
19        antesRefactorizar();
20        System.out.println();
21        Ref();
22        despuesRefactorizar();
23        System.out.println();
24    }
25
26    // Métodos de impresión de títulos y mensajes
27    private void titulo() {
28        System.out.println();
29        System.out.println("Patrón de refactorización usual 28:\n\nSeparación de responsabilidades:\nRefactoriza métodos para mejorar su claridad y mantenimiento.\n");
30        System.out.println("RESOLUCIÓN DEL EJ28");
31        System.out.println();
32    }
33
34    private void titulo1() {
35        System.out.println("      " + "-----");
36        System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
37        System.out.println("      " + "-----");
38    }
39
40    static void printError() {
41        System.out.println();
42        System.out.println("      " + "-----");
43        System.out.println("      " + " Salida forzada      ");
44        System.out.println("      " + "-----");
45        System.out.println();
46    }
47
48    void NoRef() {
49        System.out.println();
50        System.out.println("      " + "-----");
51        System.out.println("      " + " Método no refactorizado  ");
52        System.out.println("      " + "-----");
53        System.out.println();
54    }
55
56    void Ref() {
57        System.out.println();
58        System.out.println("      " + "-----");
59        System.out.println("      " + " Método refactorizado  ");
60        System.out.println("      " + "-----");
61        System.out.println();
62    }
63
64    public void antesRefactorizar() {
65        // Antes de la refactorización
66        System.out.println("Estado del código antes de la refactorización:");
67
68        // Clase tipoEmpleado antes de la refactorización
69        class tipoEmpleado {
70            private String tipo;
71            private double horabase;
72
73            public tipoEmpleado(String t, double h) {
74                this.tipo = t;
75                this.horabase = h;
76            }
77
78            public String getTipo() {
79                return tipo;
80            }
81
82            public double getHorabase() {
83                return horabase;
84            }
85        }
86
87        // Clase Empleado antes de la refactorización
88        class Empleado {
89            private int horas;
90            private int horasextra;
91            private tipoEmpleado tipo;
92
93            public double calculoHoras() {
94                if (tipo.getTipo().equals("Supervisor")) {
95                    return horas + horasextra * 1.40;
96                }
97                if (tipo.getTipo().equals("Dependiente")) {
98                    return horas + horasextra * 1.75;
99                }
100            }
101        }
102    }
103}
```

```

99         }
100        return horas + horasextra * 1.5;
101    }
102
103    public double getSueldo() {
104        return tipo.getHorabase() * calculoHoras();
105    }
106}
107
108 // Creación de objetos y Llamadas a métodos antes de la refactorización
109 tipoEmpleado tipo = new tipoEmpleado("Supervisor", 10.0);
110 Empleado empleado = new Empleado();
111 empleado.tipo = tipo;
112 empleado.horas = 8;
113 empleado.horasextra = 2;
114 double sueldo = empleado.getSueldo();
115 System.out.println("Sueldo antes de refactorizar: " + sueldo);
116}
117
118 public void despuesRefactorizar() {
119     System.out.println("Estado del código después de la refactorización:");
120
121     // Clase tipoEmpleado después de la refactorización
122     class tipoEmpleado {
123         private String tipo;
124         private double horabase;
125
126         public tipoEmpleado(String t, double h) {
127             this.tipo = t;
128             this.horabase = h;
129         }
130
131         public double getHorabase() {
132             return horabase;
133         }
134
135         public double calculoHoras(int horas, int horasextra) {
136             if (tipo.equals("Supervisor")) {
137                 return horas + horasextra * 1.40;
138             }
139             if (tipo.equals("Dependiente")) {
140                 return horas + horasextra * 1.75;
141             }
142             return horas + horasextra * 1.5;
143         }
144     }
145
146     // Clase Empleado después de la refactorización
147     class Empleado {
148         private int horas;
149         private int horasextra;
150         private tipoEmpleado tipo;
151
152         public double getSueldo() {
153             return tipo.getHorabase() * tipo.calculoHoras(horas, horasextra);
154         }
155     }
156
157     // Creación de objetos y Llamadas a métodos después de la refactorización
158     tipoEmpleado tipo = new tipoEmpleado("Supervisor", 10.0);
159     Empleado empleado = new Empleado();
160     empleado.tipo = tipo;
161     empleado.horas = 8;
162     empleado.horasextra = 2;
163     double sueldo = empleado.getSueldo();
164     System.out.println("Sueldo después de refactorizar: " + sueldo);
165 }
166}
167

```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 hora  
s) > Entornos de Desarrollo > main >  
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCo  
deDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceS  
torage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e  
• 33\bin' 'Ejercicios.Ejercicio28'
```

Patrón de refactorización usual 28:

Separación de responsabilidades:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ28

---

Prueba de métodos refactorizados y no refactorizados:

---

Comparación de resultados:

---

Método no refactorizado

---

Estado del código antes de la refactorización:

Sueldo antes de refactorizar: 108.0

---

Método refactorizado

---

Estado del código después de la refactorización:

Sueldo después de refactorizar: 108.0

## **Ejercicio 29**

```
● ○ ●
1 package Ejercicios;
2
3 public class Ejercicio29 {
4     private int horas;
5     private int horasExtras;
6     private double horaBase;
7     private TipoEmpleado tipo;
8
9     class TipoEmpleado {
10         public String toString() {
11             return "supervisor";
12         }
13     }
14
15     public static void main(String[] args) {
16         try {
17             Ejercicio29 ejercicio29 = new Ejercicio29(); // Crear una instancia de Ejercicio29
18             ejercicio29.titulo(); // Imprimir el banner
19             ejercicio29.codigo(); // Ejecutar el método de prueba para comparar resultados
20         } catch (Exception e) {
21             printError();
22         }
23     }
24
25
26
27     // Métodos de funcionalidad principal
28     void codigo() {
29         horas = 7;
30         horasExtras = 2;
31         horaBase = 35;
32         tipo = new TipoEmpleado();
33         System.out.println(tipo.toString());
34
35         titulo1();
36         System.out.println("Comparación de resultados:");
37         System.out.println();
38         NoRef();
39         System.out.println(toString());
40         System.out.println();
41         Ref();
42         System.out.println(toString());
43         System.out.println();
44     }
45
46     // Métodos de impresión de títulos y mensajes
47     private void titulo() {
48         System.out.println();
49         System.out.println(
50             "Patrón de refactorización usual 29:\n\nVisibilidad de atributos:\n Refactoriza métodos para mejorar su claridad y mantenimiento.\n");
51         System.out.println("RESOLUCIÓN DEL EJ29");
52         System.out.println();
53     }
54
55     private void titulo1() {
56         System.out.println("      " + "-----");
57         System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
58         System.out.println("      " + "-----");
59     }
60
61     static void printError() {
62         System.out.println();
63         System.out.println("      " + "-----");
64         System.out.println("      " + " Salida forzada      ");
65         System.out.println("      " + "-----");
66         System.out.println();
67     }
68
69     void NoRef() {
70         System.out.println();
71         System.out.println("      " + "-----");
72         System.out.println("      " + " Método no refactorizado  ");
73         System.out.println("      " + "-----");
74         System.out.println();
75     }
76
77     void Ref() {
78         System.out.println();
79         System.out.println("      " + "-----");
80         System.out.println("      " + " Método refactorizado   ");
81         System.out.println("      " + "-----");
82         System.out.println();
83     }
84
85     public String toString() {
86         return "Ejercicio29" + "\n" +
87             "horas: " + horas + "\n" +
88             "horasExtras: " + horasExtras + "\n" +
```

```
89     "horaBase: " + horaBase + "\n" +
90     "tipo: " + tipo + "\n";
91 }
92 }
93
94
95
96
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main
```

```
● > c:; cd 'c:\Users\zhiya\Desktop\IMF\Primer Año\David\Entornos de Desarrollo (90 horas)\Entornos de Desarrollo'; & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio29'
```

Patrón de refactorización usual 29:

Visibilidad de atributos:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ29

supervisor

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Comparación de resultados:

-----  
Método no refactorizado  
-----

Ejercicio29

horas: 7

horasExtras: 2

horaBase: 35.0

tipo: supervisor

-----  
Método refactorizado  
-----

Ejercicio29

horas: 7

```
horasExtras: 2  
horaBase: 35.0  
tipo: supervisor
```

## **Ejercicio 30**

```
● ● ●
1 package Ejercicios;
2 public class Ejercicio30 {
3     private String tipo; // Variable de instancia tipo String
4
5     public static void main(String[] args) {
6         try {
7             Ejercicio30 ejercicio30 = new Ejercicio30(); // Crear una instancia de Ejercicio30
8             ejercicio30.titulo(); // Imprimir el banner
9             ejercicio30.codigo(); // Ejecutar el método de prueba para comparar resultados
10        } catch (Exception e) {
11            printError();
12        }
13    }
14
15    // Métodos de funcionalidad principal
16    void codigo() {
17        setTipo("Ejemplo");
18
19        titulo1();
20        System.out.println("Comparación de resultados:");
21        System.out.println();
22        NoRef();
23        antesRefactorizar();
24        System.out.println();
25        Ref();
26        despuésRefactorizar();
27        System.out.println("");
28    }
29
30    // Métodos de impresión de títulos y mensajes
31    private void titulo() {
32        System.out.println();
33        System.out.println("Patrón de refactorización usual 30:\n\nEncapsulamiento de campos:\n Refactoriza métodos para mejorar su claridad y mantenimiento.\n"); System.out.println("RESOLUCIÓN DEL EJ30");
34        System.out.println();
35    }
36
37    private void titulo1() {
38        System.out.println(" " + "-----");
39        System.out.println(" " + " Prueba de métodos refactorizados y no refactorizados: ");
40        System.out.println(" " + "-----");
41    }
42
43
44    static void printError() {
45        System.out.println();
46        System.out.println(" " + "-----");
47        System.out.println(" " + " Salida forzada ");
48        System.out.println(" " + "-----");
49        System.out.println();
50    }
51
52    void NoRef() {
53        System.out.println();
54        System.out.println(" " + "-----");
55        System.out.println(" " + " Método no refactorizado ");
56        System.out.println(" " + "-----");
57        System.out.println();
58    }
59
60    void Ref() {
61        System.out.println();
62        System.out.println(" " + "-----");
63        System.out.println(" " + " Método refactorizado ");
64        System.out.println(" " + "-----");
65        System.out.println();
66    }
67
68    void antesRefactorizar() {
69        // Antes de refactorizar
70        System.out.println("Antes de refactorizar:");
71        System.out.println("Tipo: " + tipo); // Acceso directo al campo tipo
72    }
73
74    void despuésRefactorizar() {
75        // Después de refactorizar
76        System.out.println("Después de refactorizar:");
77        System.out.println("Tipo: " + getTipo()); // Con getter accedemos al campo tipo
78    }
79
80    // Getter para tipo
81    public String getTipo() {
82        return this.tipo;
83    }
84
85    // Setter para tipo
86    public void setTipo(String tipo) {
87        this.tipo = tipo;
88    }
89
90    void test() {
91        System.out.println("Comparación de resultados:");
92        System.out.println();
93        antesRefactorizar();
94        despuésRefactorizar();
95    }

```

```
95 }  
96 }  
97 }
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos  
de Desarrollo (90 horas) > Entornos de Desarrollo > main  
● > c:; cd 'c:\Users\zhiya\Desktop\IMF\Primer Año\David\Entornos d  
e Desarrollo (90 horas)\Entornos de Desarrollo'; & 'C:\Program Fi  
les\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+Sh  
owCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\R  
oaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052  
c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejerci  
cios.Ejercicio30'
```

Patrón de refactorización usual 30:

Encapsulamiento de campos:

Refactoriza métodos para mejorar su claridad y mantenimiento.

RESOLUCIÓN DEL EJ30

---

Prueba de métodos refactorizados y no refactorizados:

---

Comparación de resultados:

---

Método no refactorizado

---

Antes de refactorizar:

Tipo: Ejemplo

---

Método refactorizado

---

Después de refactorizar:

Tipo: Ejemplo

## **Ejercicio 31**

```

1 package Ejercicios;
2 public class Ejercicio31 {
3     private int horas;
4     private int horasExtra;
5     private final double horaBase;
6     private String tipo;
7
8     public static void main(String[] args) {
9         try {
10             Ejercicio31 ejercicio31 = new Ejercicio31("Supervisor", 20.0); // Crear una instancia de Ejercicio31
11             ejercicio31.titulo(); // Imprimir el banner
12             ejercicio31.codigo(); // Ejecutar el método de prueba para comparar resultados
13         } catch (Exception e) {
14             printError();
15         }
16     }
17
18     // Métodos de funcionalidad principal
19     void codigo() {
20         horas = 7;
21         horasExtra = 2;
22         tipo = "Supervisor";
23
24         titulo();
25         System.out.println("Comparación de resultados:");
26         System.out.println();
27         NoRef();
28         System.out.println("Sueldo antes de refactorizar: " + getSueldoNoRefactor());
29         System.out.println();
30         Ref();
31         System.out.println("Sueldo después de refactorizar: " + getSueldo());
32         System.out.println("");
33     }
34
35     // Métodos de impresión de títulos y mensajes
36     private void titulo() {
37         System.out.println();
38         System.out.println("Patrón de refactorización usual 31:\n\nEncapsulamiento y separación de responsabilidades:\n Refactoriza métodos para mejorar su claridad y mantenimiento");
39         System.out.println("RESOLUCIÓN DEL EJ31");
40         System.out.println();
41     }
42
43     private void titulo() {
44         System.out.println("-----");
45         System.out.println("----- Prueba de métodos refactorizados y no refactorizados: -----");
46         System.out.println("-----");
47     }
48
49     static void printError() {
50         System.out.println();
51         System.out.println("-----");
52         System.out.println("----- Salida forzada -----");
53         System.out.println("-----");
54         System.out.println();
55     }
56
57     void NoRef() {
58         System.out.println();
59         System.out.println("-----");
60         System.out.println("----- Método no refactorizado -----");
61         System.out.println("-----");
62         System.out.println();
63     }
64
65     void Ref() {
66         System.out.println();
67         System.out.println("-----");
68         System.out.println("----- Método refactorizado -----");
69         System.out.println("-----");
70         System.out.println();
71     }
72
73     // Métodos originales sin refactorizar
74     public double getSueldoNoRefactor() {
75         if (tipo.equals("Supervisor")) {
76             return horaBase + (horas + horasExtra * 1.40);
77         } else if (tipo.equals("Dependiente")) {
78             return horaBase + (horas + horasExtra * 1.75);
79         }
80         return horaBase + (horas + horasExtra * 1.5);
81     }
82
83     // Métodos refactorizados
84     public String getTipo() {
85         return this.tipo;
86     }
87
88     public void setTipo(String t) {
89         this.tipo = t;
90     }
91
92     Ejercicio31(String t, double h) {
93         this.tipo = t;
94         this.horaBase = h;
95     }
96
97     public double getHoraBase() {
98         return horaBase;
99     }
100
101    public double calculoHoras(int horas, int horasExtra) {
102        if (tipo.equals("Supervisor")) {
103            return horas + horasExtra * 1.40;
104        } else if (tipo.equals("Dependiente")) {
105            return horas + horasExtra * 1.75;
106        }
107        return horas + horasExtra * 1.5;
108    }

```

```
188     }
189
190     public double getSueldo() {
191         return getHoraBase() + calculoHoras(horas, horasExtra);
192     }
193
194     void test() {
195         System.out.println("Comparación de resultados:");
196         System.out.println();
197         System.out.println("Sueldo antes de refactorizar: " + getSueldoNoRefactor());
198         System.out.println("Sueldo después de refactorizar: " + getSueldo());
199         System.out.println("");
200     }
201 }
202
203 }
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo main
● > c:; cd 'c:\Users\zhiya\Desktop\IMF\Primer Año\David\Entornos de Desarrollo (90 horas)\Entornos de Desarrollo'; & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio31'
```

Patrón de refactorización usual 31:

Encapsulamiento y separación de responsabilidades:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ31

---

Prueba de métodos refactorizados y no refactorizados:

---

Comparación de resultados:

---

Método no refactorizado

---

Sueldo antes de refactorizar: 29.8

---

Método refactorizado

---

Sueldo después de refactorizar: 29.8

## **Ejercicio 32**

```

1 package Ejercicios;
2 public class Ejercicio32 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio32 ejercicio32 = new Ejercicio32(); // Crear una instancia de Ejercicio32
6             ejercicio32.titulo(); // Imprimir el banner
7             ejercicio32.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13    // Métodos de funcionalidad principal
14    void codigo() {
15        Equipo equipo = new Equipo("Los Guerreros", "Carlos Perez");
16        Jugador jugador = new Jugador(equipo);
17
18        titulo1();
19        System.out.println("Demostración de resultados:");
20        System.out.println();
21        NoRef();
22        System.out.println("El equipo del jugador es: " + jugador.getEquipo().getNombreEquipo());
23        System.out.println("El capitán del equipo es: " + jugador.getCapitanDeMiEquipo());
24        System.out.println();
25        Ref();
26        System.out.println("El equipo del jugador es: " + jugador.getEquipo().getNombreEquipo());
27        System.out.println("El capitán del equipo es: " + jugador.getCapitanDeMiEquipo());
28        System.out.println("");
29    }
30
31    // Métodos de impresión de títulos y mensajes
32    private void titulo() {
33        System.out.println();
34        System.out.println("Patrón de refactorización usual 32:\n\nEncapsulamiento de la relación Equipo-Jugador:\n Refactoriza métodos para mejorar su claridad y mantenimiento");
35        System.out.println("RESOLUCIÓN DEL EJ32");
36        System.out.println();
37    }
38
39    private void titulo1() {
40        System.out.println("      " + "-----");
41        System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
42        System.out.println("      " + "-----");
43    }
44
45    static void printError() {
46        System.out.println();
47        System.out.println("      " + "-----");
48        System.out.println("      " + " Salida forzada      ");
49        System.out.println("      " + "-----");
50        System.out.println();
51    }
52
53    void NoRef() {
54        System.out.println();
55        System.out.println("      " + "-----");
56        System.out.println("      " + " Método no refactorizado ");
57        System.out.println("      " + "-----");
58        System.out.println();
59    }
60
61    void Ref() {
62        System.out.println();
63        System.out.println("      " + "-----");
64        System.out.println("      " + " Método refactorizado ");
65        System.out.println("      " + "-----");
66        System.out.println();
67    }
68
69    // Define the Equipo (Team) class
70    static class Equipo {
71        private final String nombreEquipo;
72        private final String capitan;
73
74        public Equipo(String nombreEquipo, String capitan) {
75            this.nombreEquipo = nombreEquipo;
76            this.capitan = capitan;
77        }
78
79        public String getCapitan() {
80            return capitan;
81        }
82
83        public String getNombreEquipo() {
84            return nombreEquipo;
85        }
86    }
87
88    // Define the Jugador (Player) class
89    static class Jugador {
90        private Equipo miEquipo;
91
92        public Jugador(Equipo equipo) {
93            this.miEquipo = equipo;
94        }
95
96        public Equipo getEquipo() {
97            return miEquipo;
98        }
99
100       public void setEquipo(Equipo equipo) {
101           this.miEquipo = equipo;
102       }
103
104       public String getCapitanDeMiEquipo() {
105           return miEquipo.getCapitan();
106       }

```

```
107 }  
108 }  
109
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos  
de Desarrollo (90 horas) > Entornos de Desarrollo > main  
● > c:; cd 'c:\Users\zhiya\Desktop\IMF\Primer Año\David\Entornos d  
e Desarrollo (90 horas)\Entornos de Desarrollo'; & 'C:\Program Fi  
les\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+Sh  
owCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\R  
oaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052  
c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejerci  
cios.Ejercicio32'
```

Patrón de refactorización usual 32:

Encapsulamiento de la relación Equipo-Jugador:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ32

---

Prueba de métodos refactorizados y no refactorizados:

---

Demostración de resultados:

---

Método no refactorizado

---

El equipo del jugador es: Los Guerreros

El capitán del equipo es: Carlos Perez

---

Método refactorizado

---

El equipo del jugador es: Los Guerreros

El capitán del equipo es: Carlos Perez

## **Ejercicio 33**

```

1 package Ejercicios;
2 public class Ejercicio33 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio33 ejercicio33 = new Ejercicio33(); // Crear una instancia de Ejercicio33
6             ejercicio33.titulo(); // Imprimir el banner
7             ejercicio33.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13 // Métodos de funcionalidad principal
14 void codigo() {
15     Date fechafin = new Date();
16     Date diasiguiente = new Date(fechafin.getAnio(), fechafin.getMes(), fechafin.getDia() + 1);
17
18     titulo1();
19     System.out.println("Comparación de resultados:");
20     System.out.println();
21     NoRef();
22     System.out.println(diasiguiente.toString());
23     System.out.println();
24     Ref();
25     Date diaSiguiente2 = Date.siguienteDia(fechafin);
26     System.out.println(diaSiguiente2.toString());
27     System.out.println();
28     test(diasiguiente, diaSiguiente2);
29     System.out.println("");
30 }
31
32 // Métodos de impresión de títulos y mensajes
33 private void titulo() {
34     System.out.println();
35     System.out.println("Patrón de refactorización usual 33:\n\nEncapsulamiento de lógica de negocio en métodos estáticos:\n Refactoriza métodos para mejorar su claridad y mantenimiento");
36     System.out.println("RESOLUCIÓN DEL EJ33");
37     System.out.println();
38 }
39
40 private void titulo1() {
41     System.out.println("      " + "-----");
42     System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
43     System.out.println("      " + "-----");
44 }
45
46 static void printError() {
47     System.out.println();
48     System.out.println("      " + "-----");
49     System.out.println("      " + " Salida forzada ");
50     System.out.println("      " + "-----");
51     System.out.println();
52 }
53
54 void NoRef() {
55     System.out.println();
56     System.out.println("      " + "-----");
57     System.out.println("      " + " Método no refactorizado ");
58     System.out.println("      " + "-----");
59     System.out.println();
60 }
61
62 void Ref() {
63     System.out.println();
64     System.out.println("      " + "-----");
65     System.out.println("      " + " Método refactorizado ");
66     System.out.println("      " + "-----");
67     System.out.println();
68 }
69
70 private void test(Date diasiguiente, Date diaSiguiente2) {
71     System.out.println("Comparación de resultados:");
72     System.out.println();
73     System.out.println(diasiguiente.toString());
74     System.out.println(diaSiguiente2.toString());
75 }
76 }
77
78 // Clase Date
79 class Date {
80     private int año = 2024;
81     private int mes = 4;
82     private int dia = 24;
83
84     public Date() {}
85
86     public Date(int año, int mes, int dia) {
87         this.año = año;
88         this.mes = mes;
89         this.dia = dia;
90     }
91
92     public int getAño() {
93         return this.año;
94     }
95
96     public int getMes() {
97         return this.mes;
98     }
99
100    public int getDia() {
101        return this.dia;
102    }
103
104    public String toString() {
105        StringBuffer cadena = new StringBuffer();
106        cadena.append(getDia() + "/");
107        cadena.append(getMes() + "/");
108        cadena.append(getAño());
109        return cadena.toString();
110    }
111
112    static Date siguienteDia(Date uno) {

```

```
112     static Date siguienteDia(Date var) {
113         return new Date(var.getAnio(), var.getMes(), var.getDia() + 1);
114     }
115 }
116
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main >
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio33'
```

Patrón de refactorización usual 33:

Encapsulamiento de lógica de negocio en métodos estáticos:  
Refactoriza métodos para mejorar su claridad y mantenimiento.

#### RESOLUCIÓN DEL EJ33

-----  
Prueba de métodos refactorizados y no refactorizados:

Comparación de resultados:

-----  
Método no refactorizado  
-----

25/4/2024

-----  
Método refactorizado  
-----

25/4/2024

Comparación de resultados:

25/4/2024

25/4/2024

## **Ejercicio 34**

```

1 package Ejercicios;
2 public class Ejercicio34 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio34 ejercicio34 = new Ejercicio34(); // Crear una instancia de Ejercicio34
6             ejercicio34.titulo(); // Imprimir el banner
7             ejercicio34.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13    // Métodos de funcionalidad principal.
14    void codigo() {
15        Telefono telefono = new Telefono("+123456789");
16        Jugador jugador = new Jugador(telefono);
17
18        titulo();
19        System.out.println("Demostración de resultados:");
20        System.out.println();
21        NoRef();
22        System.out.println("El teléfono del jugador es: " + jugador.getTelefono().getTelefono());
23        System.out.println("El prefijo del teléfono es: " + jugador.getTelefono().getPrefijo());
24        System.out.println();
25        Ref();
26        System.out.println("El teléfono del jugador es: " + jugador.getTelefono().getTelefono());
27        System.out.println("El prefijo del teléfono es: " + jugador.getTelefono().getPrefijo());
28    }
29
30    // Métodos de impresión de títulos y mensajes
31    private void titulo() {
32        System.out.println();
33        System.out.println("Patrón de refactorización usual 34:\n\nEncapsulamiento y separación de responsabilidades:\n Refactoriza métodos para mejorar su claridad y mantenimiento.\n");
34        System.out.println("RESOLUCION DEL EJ34");
35        System.out.println();
36    }
37
38    private void titulo1() {
39        System.out.println("      " + "-----");
40        System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
41        System.out.println("      " + "-----");
42    }
43
44    static void printError() {
45        System.out.println();
46        System.out.println("      " + "-----");
47        System.out.println("      " + " Salida forzada   ");
48        System.out.println("      " + "-----");
49        System.out.println();
50    }
51
52    void NoRef() {
53        System.out.println();
54        System.out.println("      " + "-----");
55        System.out.println("      " + " Método no refactorizado ");
56        System.out.println("      " + "-----");
57        System.out.println();
58    }
59
60    void Ref() {
61        System.out.println();
62        System.out.println("      " + "-----");
63        System.out.println("      " + " Método refactorizado ");
64        System.out.println("      " + "-----");
65        System.out.println();
66    }
67
68    // Define the Telefono (Phone) class
69    static class Telefono {
70        private String telefono;
71
72        public Telefono(String telefono) {
73            this.telefono = telefono;
74        }
75
76        public String getTelefono() {
77            return telefono;
78        }
79
80        public void setTelefono(String telefono) {
81            this.telefono = telefono;
82        }
83
84        public String getPrefijo() {
85            return telefono.substring(0, 3); // Example: Assuming prefix is the first 3 characters
86        }
87    }
88
89    // Define the Jugador (Player) class
90    static class Jugador {
91        private Telefono telefono;
92
93        public Jugador(Telefono telefono) {
94            this.telefono = telefono;
95        }
96
97        public Telefono getTelefono() {
98            return telefono;
99        }
100
101       public void setTelefono(Telefono telefono) {
102           this.telefono = telefono;
103       }
104   }
105 }
106

```

```
> nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio34'
```

Patrón de refactorización usual 34:

Encapsulamiento y separación de responsabilidades:

Refactoriza métodos para mejorar su claridad y mantenimiento.

#### RESOLUCIÓN DEL EJ34

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Demostración de resultados:

-----  
Método no refactorizado  
-----

El teléfono del jugador es: +123456789

El prefijo del teléfono es: +12

-----  
Método refactorizado  
-----

El teléfono del jugador es: +123456789

El prefijo del teléfono es: +12

## Ejercicio 35

```

1 package Ejercicios;
2 public class Ejercicio35 {
3     private double unidades = 13;
4     private double precioUnitario = 100;
5
6     public static void main(String[] args) {
7         try {
8             Ejercicio35 ejercicio35 = new Ejercicio35(); // Crear una instancia de Ejercicio35
9             ejercicio35.titulo(); // Imprimir el banner
10            ejercicio35.codigo(); // Ejecutar el método de prueba para comparar resultados
11        } catch (Exception e) {
12            printError();
13        }
14    }
15
16    // Métodos de funcionalidad principal
17    void codigo() {
18        titulo1();
19        System.out.println("Comparación de resultados:");
20        System.out.println();
21        NoRef();
22        System.out.println(descuentoSource(unidades, precioUnitario));
23        System.out.println();
24        Ref();
25        System.out.println(descuentoRefactor(unidades, precioUnitario));
26        System.out.println();
27        test();
28    }
29
30    // Métodos de impresión de títulos y mensajes
31    private void titulo() {
32        System.out.println();
33        System.out.println("Patrón de refactorización usual 35:\n\nUso de constantes:\n Refactoriza métodos para mejorar su claridad y mantenimiento\n");
34        System.out.println("RESOLUCIÓN DEL EJ35");
35        System.out.println();
36    }
37
38    private void titulo1() {
39        System.out.println("      " + "-----");
40        System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
41        System.out.println("      " + "-----");
42    }
43
44    static void printError() {
45        System.out.println();
46        System.out.println("      " + "-----");
47        System.out.println("      " + " Salida forzada   ");
48        System.out.println("      " + "-----");
49        System.out.println();
50    }
51
52    void NoRef() {
53        System.out.println();
54        System.out.println("      " + "-----");
55        System.out.println("      " + " Método no refactorizado  ");
56        System.out.println("      " + "-----");
57        System.out.println();
58    }
59
60    void Ref() {
61        System.out.println();
62        System.out.println("      " + "-----");
63        System.out.println("      " + " Método refactorizado   ");
64        System.out.println("      " + "-----");
65        System.out.println();
66    }
67
68    double descuentoSource(double unidades, double precioUnitario) {
69        return unidades * 0.95 * precioUnitario;
70    }
71
72    static final double DESCUENTO_DIRECTO = 0.95;
73
74    double descuentoRefactor(double unidades, double precioUnitario) {
75        return unidades * DESCUENTO_DIRECTO * precioUnitario;
76    }
77
78    void test() {
79        System.out.println("Comparación de resultados:");
80        System.out.println();
81        System.out.println(descuentoSource(unidades, precioUnitario));
82        System.out.println(descuentoRefactor(unidades, precioUnitario));
83    }
84}
85

```

```
> nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main >
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio35'
```

Patrón de refactorización usual 35:

Uso de constantes:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ35

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Comparación de resultados:

-----  
Método no refactorizado  
-----

1235.0

-----  
Método refactorizado  
-----

1235.0

Comparación de resultados:

1235.0

1235.0

## Ejercicio 36



```
1 package Ejercicios;
2 public class Ejercicio36 {
3     private String[] asignaturas;
4
5     public Ejercicio36(String[] asignaturas) {
6         this.asignaturas = asignaturas;
7     }
8
9     public static void main(String[] args) {
10        try {
11            Ejercicio36 ejercicio36 = new Ejercicio36(new String[] {"Matemáticas", "Física", "Química"}); // Crear una instancia de Ejercicio36
12            ejercicio36.titulo(); // Imprimir el banner
13            ejercicio36.codigo(); // Ejecutar el método de prueba para comparar resultados
14        } catch (Exception e) {
15            printError();
16        }
17    }
18
19    // Métodos de funcionalidad principal
20    void codigo() {
21        titulo();
22        System.out.println("Comparación de resultados:");
23        System.out.println();
24        NoRef();
25        antesRefactorizar();
26        System.out.println();
27        Ref();
28        despuesRefactorizar();
29        System.out.println();
30    }
31
32    // Métodos de impresión de títulos y mensajes
33    private void titulo() {
34        System.out.println();
35        System.out.println("Patrón de refactorización usual 36:\n\nUso de copias para evitar modificación de originales:\n Refactoriza métodos para mejorar su claridad y mantenimiento");
36        System.out.println("RESOLUCIÓN DEL EJ36");
37        System.out.println();
38    }
39
40    private void titulo() {
41        System.out.println("      " + "-----");
42        System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
43        System.out.println("      " + "-----");
44    }
45
46    static void printError() {
47        System.out.println();
48        System.out.println("      " + "-----");
49        System.out.println("      " + " Salida forzada      ");
50        System.out.println("      " + "-----");
51        System.out.println();
52    }
53
54    void NoRef() {
55        System.out.println();
56        System.out.println("      " + "-----");
57        System.out.println("      " + " Método no refactorizado  ");
58        System.out.println("      " + "-----");
59        System.out.println();
60    }
61
62    void Ref() {
63        System.out.println();
64        System.out.println("      " + "-----");
65        System.out.println("      " + " Método refactorizado   ");
66        System.out.println("      " + "-----");
67        System.out.println();
68    }
69
70    // Getters y Setters
71    String[] getAsignaturas() {
72        return asignaturas;
73    }
74
75    void setAsignaturas(String[] var) {
76        asignaturas = var;
77    }
78
79    void setAsignatura(int donde, String titulo) {
80        asignaturas[donde] = titulo;
81    }
82
83    void antesRefactorizar() {
84        System.out.println("Resultado antes de refactorizar:");
85        imprimirAsignaturas(getAsignaturas());
86    }
87
88    void despuesRefactorizar() {
89        System.out.println("Resultado después de refactorizar:");
90        imprimirAsignaturas(getAsignaturasCopia());
91    }
92
93    /*
94     * Setter a partir de una copia.
95     * Preserva el original.
96     */
97    void setAsignaturasCopia(String[] var) {
98        asignaturas = new String[var.length];
99        for (int i = 0; i < var.length; i++) {
100            setAsignatura(i, var[i]);
101        }
102    }
103
104    /*
105     * En este getter creo una copia del array
106     * y lo devuelvo, con lo cual no se modifica
107     * el array original
108     */
109    String[] getAsignaturasCopia() {
110        String[] var = new String[asignaturas.length];
```

```
111     System.arraycopy(asignaturas, 0, var, 0, asignaturas.length);
112 }
113 }
114
115 // Método para imprimir asignaturas
116 void imprimirAsignaturas(String[] asignaturas) {
117     for (String asignatura : asignaturas) {
118         System.out.println(asignatura);
119     }
120 }
121 }
122 }
```

```
> nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main
```

```
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio36'
```

Patrón de refactorización usual 36:

Uso de copias para evitar modificación de originales:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ36

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Comparación de resultados:

-----  
Método no refactorizado  
-----

Resultado antes de refactorizar:

Matemáticas

Física

Química

-----  
Método refactorizado  
-----

Resultado después de refactorizar:

Matemáticas

Física

Química

## **Ejercicio 37**

```

1 package Ejercicios;
2 public class Ejercicio37 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio37 ejercicio37 = new Ejercicio37(); // Crear una instancia de Ejercicio37
6             ejercicio37.titulo(); // Imprimir el banner
7             ejercicio37.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13 // Métodos de funcionalidad principal
14 void codigo() {
15     refactorTenista objeto37 = new refactorTenista();
16     titulo();
17     System.out.println("Demostración de resultados:");
18     System.out.println();
19     Ref();
20     objeto37.create(1);
21     objeto37.create(2);
22     NoRef();
23     objeto37.create(3);
24     System.out.println("");
25 }
26
27 // Métodos de impresión de títulos y mensajes
28 private void titulo() {
29     System.out.println();
30     System.out.println(
31         "Patrón de refactorización usual 37:\n\nUso de clases específicas para diferentes tipos:\nRefactoriza métodos para mejorar su claridad y mantenimiento.\n");
32     System.out.println("RESOLUCIÓN DEL EJ37");
33     System.out.println();
34 }
35
36 private void titulo() {
37     System.out.println("      " + "-----");
38     System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
39     System.out.println("      " + "-----");
40 }
41
42 static void printError() {
43     System.out.println();
44     System.out.println("      " + "-----");
45     System.out.println("      " + " Salida forzada   ");
46     System.out.println("      " + "-----");
47     System.out.println();
48 }
49
50 void Ref() {
51     System.out.println();
52     System.out.println("      " + "-----");
53     System.out.println("      " + " Método refactorizado   ");
54     System.out.println("      " + "-----");
55     System.out.println();
56 }
57
58 void NoRef() {
59     System.out.println();
60     System.out.println("      " + "-----");
61     System.out.println("      " + " Método no refactorizado   ");
62     System.out.println("      " + "-----");
63     System.out.println();
64 }
65
66 // Definición de clases
67 class tenista {
68     public final int AMATEUR = 1;
69     public final int SEMIPROFESIONAL = 2;
70     public final int PROFESIONAL = 3;
71     public final int _sexo = 2;
72
73     public int getSexo() {
74         return _sexo;
75     }
76 }
77
78 class refactorTenista {
79     private final int AMATEUR = 1;
80     private final int SEMIPROFESIONAL = 2;
81     private final int PROFESIONAL = 3;
82     private final int _tipo = 2;
83
84     public int getTipo() {
85         return _tipo;
86     }
87
88     public refactorTenista create(int tipo) {
89         switch (tipo) {
90             case AMATEUR:
91                 System.out.println("Se crea objeto amateur");
92                 return new amateur();
93             case SEMIPROFESIONAL:
94                 System.out.println("Se crea objeto semiprofesional");
95                 return new semiprofesional();
96             case PROFESIONAL:
97                 System.out.println("Se crea objeto profesional");
98                 return new profesional();
99             default:
100                 throw new IllegalArgumentException("tipo incorrecto");
101            }
102        }

```

```
102     }
103
104     class amateur extends refactorTenista {
105         public amateur() {
106     }
107 }
108
109     class semiprofesional extends refactorTenista {
110         public semiprofesional() {
111     }
112 }
113
114     class profesional extends refactorTenista {
115         public profesional() {
116     }
117 }
118 }
119 }
120 }
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) >
Entornos de Desarrollo > main >
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio37'
```

Patrón de refactorización usual 37:

Uso de clases específicas para diferentes tipos:

Refactoriza métodos para mejorar su claridad y mantenimiento.

#### RESOLUCIÓN DEL EJ37

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Demostración de resultados:

-----  
Método refactorizado  
-----

Se crea objeto amateur  
Se crea objeto semiprofesional

-----  
Método no refactorizado  
-----

Se crea objeto profesional

## Ejercicio 38

```
1 package Ejercicios;
2 public class Ejercicio38 {
3     // Instancia única de la clase Singleton
4     private static Ejercicio38 instancia = null;
5
6     // Constructor privado para evitar la instantiación directa
7     private Ejercicio38() {
8         // Este método existe solamente para evitar la instantiación directa.
9     }
10
11    // Método estático para obtener la instancia de la clase
12    public static Ejercicio38 getInstance() {
13        // Si la instancia es nula, creamos una nueva instancia
14        if (instancia == null) {
15            instancia = new Ejercicio38();
16        }
17        // Devolvemos la instancia única
18        return instancia;
19    }
20
21    public static void main(String[] args) {
22        try {
23            Ejercicio38 ejercicio38 = new Ejercicio38(); // Crear una instancia de Ejercicio38
24            ejercicio38.titulo(); // Imprimir el banner
25            ejercicio38.codigo(); // Ejecutar el método de prueba para comparar resultados
26        } catch (Exception e) {
27            printError();
28        }
29    }
30
31    // Métodos de funcionalidad principal
32    void codigo() {
33        titulo1();
34        System.out.println("Comparación de resultados:");
35        System.out.println();
36        NoRef();
37        antesRefactorizar();
38        System.out.println();
39        Ref();
40        despuesRefactorizar();
41        System.out.println();
42    }
43
44    // Métodos de impresión de títulos y mensajes
45    private void titulo() {
46        System.out.println();
47        System.out.println("Patrón de refactorización usual 38:\n\nUso de Singleton:\n Refactoriza métodos para mejorar su claridad y mantenimiento");
48        System.out.println("RESOLUCIÓN DEL EJ38");
49        System.out.println();
50    }
51
52    private void titulo1() {
53        System.out.println("-----");
54        System.out.println("----- Prueba de métodos refactorizados y no refactorizados: ");
55        System.out.println("-----");
56    }
57
58    static void printError() {
59        System.out.println();
60        System.out.println("-----");
61        System.out.println("----- Salida forzada -----");
62        System.out.println("-----");
63        System.out.println();
64    }
65
66    void NoRef() {
67        System.out.println();
68        System.out.println("-----");
69        System.out.println("----- Método no refactorizado -----");
70        System.out.println("-----");
71        System.out.println();
72    }
73
74    void Ref() {
75        System.out.println();
76        System.out.println("-----");
77        System.out.println("----- Método refactorizado -----");
78        System.out.println("-----");
79        System.out.println();
80    }
81
82    public static void antesRefactorizar() {
83        System.out.println("Método antes de refactorizar de la clase Singleton");
84    }
85
86    public static void despuesRefactorizar() {
87        System.out.println("Método después de refactorizar de la clase Singleton");
88    }
89 }
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main >
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio38'
```

Patrón de refactorización usual 38:

Uso de Singleton:

Refactoriza métodos para mejorar su claridad y mantenimiento.

## RESOLUCIÓN DEL EJ38

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Comparación de resultados:

-----  
Método no refactorizado  
-----

Método antes de refactorizar de la clase Singleton

-----  
Método refactorizado  
-----

Método después de refactorizar de la clase Singleton

# Ejercicio 39

```
● ● ●
1 package Ejercicios;
2 public class Ejercicio39 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio39 ejercicio39 = new Ejercicio39(); // Crear una instancia de Ejercicio39
6             ejercicio39.titulo(); // Imprimir el banner
7             ejercicio39.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13 // Métodos de funcionalidad principal
14 void codigo() {
15     Paisaje paisaje = new Paisaje();
16
17     titulo1();
18     System.out.println("Demostración de resultados:");
19     System.out.println();
20     Ref();
21     System.out.println(paisaje.mostrar());
22 }
23
24 // Métodos de impresión de títulos y mensajes
25 private void titulo() {
26     System.out.println();
27     System.out.println("Patrón de refactorización usual 39:\n\nUso de clases específicas para diferentes tipos de escenas:\n Refactoriza métodos para mejorar su claridad y mantenimiento.\n");
28     System.out.println("RESOLUCIÓN DEL EJ39");
29     System.out.println();
30 }
31
32 private void titulo1() {
33     System.out.println("      " + "-----");
34     System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
35     System.out.println("      " + "-----");
36 }
37
38 static void printError() {
39     System.out.println();
40     System.out.println("      " + "-----");
41     System.out.println("      " + " Salida forzada      ");
42     System.out.println("      " + "-----");
43     System.out.println();
44 }
45
46 void Ref() {
47     System.out.println();
48     System.out.println("      " + "-----");
49     System.out.println("      " + " Método refactorizado  ");
50     System.out.println("      " + "-----");
51     System.out.println();
52 }
53
54 // Definición de clases
55 class Escena {
56     void dibujar() {}
57 }
58
59 class EscenaCiudad extends Escena {}
60
61 class EscenaMontaña extends Escena {}
62
63 class Paisaje {
64     String mostrar() {
65         return "Mostrar Escenas";
66     }
67 }
68 }
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo main
> c:; cd 'c:\Users\zhiya\Desktop\Primer Año\David\Entornos de Desarrollo (90 horas)\Entornos de Desarrollo'; & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio39'
```

Patrón de refactorización usual 39:

Uso de clases específicas para diferentes tipos de escenas:

Refactoriza métodos para mejorar su claridad y mantenimiento.

#### RESOLUCIÓN DEL EJ39

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Demostración de resultados:

-----  
Método refactorizado  
-----

Mostrar Escenas

-----  
Método no refactorizado  
-----

Mostrar Escenas

## **Ejercicio 40**

```

1 package Ejercicios;
2 public class Ejercicio40 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio40 ejercicio40 = new Ejercicio40(); // Crear una instancia de Ejercicio40
6             ejercicio40.titulo(); // Imprimir el banner
7             ejercicio40.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13 // Métodos de funcionalidad principal
14 void codigo() {
15     titulo();
16     System.out.println("Comparación de resultados:");
17     System.out.println();
18     NoRef();
19     antesRefactorizar();
20     System.out.println();
21     Ref();
22     despuesRefactorizar();
23     System.out.println();
24 }
25
26 // Métodos de impresión de títulos y mensajes
27 private void titulo() {
28     System.out.println();
29     System.out.println("Patrón de refactorización usual 40:\n\nUso del patrón Decorator:\n Refactoriza métodos para mejorar su claridad y mantenimiento");
30     System.out.println("RESOLUCIÓN DEL EJ40");
31     System.out.println();
32 }
33
34 private void titulo1() {
35     System.out.println("      " + "-----");
36     System.out.println("      " + " Prueba de métodos refactorizados y no refactorizados: ");
37     System.out.println("      " + "-----");
38 }
39
40 static void printError() {
41     System.out.println();
42     System.out.println("      " + "-----");
43     System.out.println("      " + " Salida forzada   ");
44     System.out.println("      " + "-----");
45     System.out.println();
46 }
47
48 void NoRef() {
49     System.out.println();
50     System.out.println("      " + "-----");
51     System.out.println("      " + " Método no refactorizado  ");
52     System.out.println("      " + "-----");
53     System.out.println();
54 }
55
56 void Ref() {
57     System.out.println();
58     System.out.println("      " + "-----");
59     System.out.println("      " + " Método refactorizado  ");
60     System.out.println("      " + "-----");
61     System.out.println();
62 }
63
64 // Método antesRefactorizar - sin usar el patrón Decorator
65 public static void antesRefactorizar() {
66     // Crear instancias de diferentes paquetes
67     Pack pack1 = new PackBasico();
68     Pack pack2 = new PackCuerpo();
69     Pack pack3 = new PackLujo();
70     // Imprimir descripciones y precios de los paquetes
71     System.out.println(pack1.getDescripcion() + " cuesta " + pack1.getPrecio());
72     System.out.println(pack2.getDescripcion() + " cuesta " + pack2.getPrecio());
73     System.out.println(pack3.getDescripcion() + " cuesta " + pack3.getPrecio());
74 }
75
76 // Método despuesRefactorizar - usando el patrón Decorator
77 public static void despuesRefactorizar() {
78     // Crear instancias de paquetes y decorarlas
79     Pack pack1 = new PackBasico();
80     Pack pack2 = new CuerpoDecorator(new PackBasico());
81     Pack pack3 = new XenonDecorator(new LunasDecorator(new CuerpoDecorator(new PackLujo())));
82     // Imprimir descripciones y precios de los paquetes decorados
83     System.out.println(pack1.getDescripcion() + " cuesta " + pack1.getPrecio());
84     System.out.println(pack2.getDescripcion() + " cuesta " + pack2.getPrecio());
85     System.out.println(pack3.getDescripcion() + " cuesta " + pack3.getPrecio());
86 }
87
88 // Definición de clases
89 abstract static class Pack {
90     protected String descripcion = "Paquete desconocido";
91
92     public String getDescripcion() {
93         return descripcion;
94     }

```

```

95     public abstract double getPrecio();
96 }
97
98 // Implementaciones concretas del Pack
99 static class PackBasico extends Pack {
100     public PackBasico() {
101         descripcion = "Paquete básico";
102     }
103
104     @Override
105     public double getPrecio() {
106         return 100.0;
107     }
108 }
109
110 static class PackCuero extends Pack {
111     public PackCuero() {
112         descripcion = "Paquete con cuero";
113     }
114
115     @Override
116     public double getPrecio() {
117         return 150.0;
118     }
119 }
120
121 static class PackLujo extends Pack {
122     public PackLujo() {
123         descripcion = "Paquete de lujo";
124     }
125
126     @Override
127     public double getPrecio() {
128         return 200.0;
129     }
130 }
131
132 // Decorator abstracto
133 abstract static class PackDecorator extends Pack {
134     public abstract String getDescripcion();
135 }
136
137 // Decoradores concretos
138 static class FantasDecorador extends PackDecorator {
139     Pack pack;
140
141     public FantasDecorador(Pack pack) {
142         this.pack = pack;
143     }
144
145     @Override
146     public String getDescripcion() {
147         return pack.getDescripcion() + ", con llantas de fantasía";
148     }
149
150     @Override
151     public double getPrecio() {
152         return pack.getPrecio() + 50.0;
153     }
154 }
155
156 static class LunasDecorador extends PackDecorator {
157     Pack pack;
158
159     public LunasDecorador(Pack pack) {
160         this.pack = pack;
161     }
162
163     @Override
164     public String getDescripcion() {
165         return pack.getDescripcion() + ", con lunas polarizadas";
166     }
167
168     @Override
169     public double getPrecio() {
170         return pack.getPrecio() + 30.0;
171     }
172 }
173
174 static class CueroDecorador extends PackDecorator {
175     Pack pack;
176
177     public CueroDecorador(Pack pack) {
178         this.pack = pack;
179     }
180
181     @Override
182     public String getDescripcion() {
183         return pack.getDescripcion() + ", con tapizado de cuero";
184     }
185
186     @Override
187     public double getPrecio() {
188         return pack.getPrecio() + 70.0;
189     }
190 }
191
192 static class XenonDecorador extends PackDecorator {
193     Pack pack;

```

```
194     pack.pack;
195
196     public XenonDecorator(Pack pack) {
197         this.pack = pack;
198     }
199
200     @Override
201     public String getDescripcion() {
202         return pack.getDescripcion() + ", con luces de xenon";
203     }
204
205     @Override
206     public double getPrecio() {
207         return pack.getPrecio() + 40.0;
208     }
209 }
210 }
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio40'
```

Patrón de refactorización usual 40:

Uso del patrón Decorator:

Refactoriza métodos para mejorar su claridad y mantenimiento.

RESOLUCIÓN DEL EJ40

-----  
Prueba de métodos refactorizados y no refactorizados:  
-----

Comparación de resultados:

-----  
Método no refactorizado  
-----

Paquete básico cuesta 100.0

Paquete con cuero cuesta 150.0

Paquete de lujo cuesta 200.0

-----  
Método refactorizado  
-----

Paquete básico cuesta 100.0

Paquete básico, con tapizado de cuero cuesta 170.0

Paquete de lujo, con tapizado de cuero, con lunas polarizadas, con luces de xenon cuesta 340.0

# Ejercicio 41

```
1 package Ejercicios;
2 public class Ejercicio41 {
3     public static void main(String[] args) {
4         try {
5             Ejercicio41 ejercicio41 = new Ejercicio41(); // Crear una instancia de Ejercicio41
6             ejercicio41.titulo(); // Imprimir el banner
7             ejercicio41.codigo(); // Ejecutar el método de prueba para comparar resultados
8         } catch (Exception e) {
9             printError();
10        }
11    }
12
13 // Métodos de funcionalidad principal
14 void codigo() {
15     Sensor objeto41A = new Sensor();
16     RefactorSensorEstado objeto41B = new RefactorSensorEstado();
17
18     titulo();
19     System.out.println("Comparación de resultados:");
20     System.out.println();
21     NoRef();
22     objeto41A.mide();
23     System.out.println();
24     Ref();
25     objeto41B.mide();
26     System.out.println();
27 }
28
29 // Métodos de impresión de títulos y mensajes
30 private void titulo() {
31     System.out.println();
32     System.out.println("Patrón de refactorización usual 41:\n\nUso de estados específicos para diferentes tipos de sensores:\n Refactoriza métodos para mejorar su claridad y mantenimiento");
33     System.out.println("RESOLUCIÓN DEL EJ41");
34     System.out.println();
35 }
36
37 private void titulo() {
38     System.out.println(" " + "-----");
39     System.out.println(" " + " Prueba de métodos refactorizados y no refactorizados: ");
40     System.out.println(" " + "-----");
41 }
42
43 static void printError() {
44     System.out.println();
45     System.out.println(" " + "-----");
46     System.out.println(" " + " Salida forzada ");
47     System.out.println(" " + "-----");
48     System.out.println();
49 }
50
51 void NoRef() {
52     System.out.println();
53     System.out.println(" " + "-----");
54     System.out.println(" " + " Método no refactorizado ");
55     System.out.println(" " + "-----");
56     System.out.println();
57 }
58
59 void Ref() {
60     System.out.println();
61     System.out.println(" " + "-----");
62     System.out.println(" " + " Método refactorizado ");
63     System.out.println(" " + "-----");
64     System.out.println();
65 }
66
67 // Definición de clases
68 class Sensor {
69     void monitorizar() {
70         Sensor estado = new Sensor();
71         estado.mide();
72     }
73
74     void mide() {
75         System.out.println("Medidor de estado");
76     }
77 }
78
79 class RefactorSensorEstado {
80     void mide() {
81         System.out.println("Medidor de estado");
82     }
83 }
84
85 class Apagado extends RefactorSensorEstado {}
86
87 class EnReposo extends RefactorSensorEstado {}
88
89 class Activo extends RefactorSensorEstado {}
90 }
91
```

```
nazhida@Nia ~ > Desktop > IMF > Primer Año > David > Entornos de Desarrollo (90 horas) > Entornos de Desarrollo > main
> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zhiya\AppData\Roaming\Code\User\workspaceStorage\9d13d2fb19366679bfa14d295ae3052c\redhat.java\jdt_ws\Entornos de Desarrollo_674e4e33\bin' 'Ejercicios.Ejercicio41'
```

Patrón de refactorización usual 41:

Uso de estados específicos para diferentes tipos de sensores:  
Refactoriza métodos para mejorar su claridad y mantenimiento.

RESOLUCIÓN DEL EJ41

-----  
Prueba de métodos refactorizados y no refactorizados:

Comparación de resultados:

-----  
Método no refactorizado

Medidor de estado

-----  
Método refactorizado

Medidor de estado