

*Mercado Salinas Fhernando*

- *Máster Universitario en Sistemas Embebidos*
  - Mondragon Unibertsitatea, Basque Country, Spain
- *Ingeniería en Sistemas Computacionales*
  - Tecnológico de Estudios Superiores de Jocotitlán

---

# Programación Lógica y Funcional

---

v 0.0, Advanced Level



# Expresión lambda como retorno de métodos



```
1  /*
2  ...   Interfaz Funcional
3  */
4
5  public interface CalculadoraInt{
6      public int calculate(int x, int y);
7  }
8
```

The screenshot shows an IDE window with three tabs: 'CalculadoraInt.java', 'CalculadoraLong.java', and 'Principal.java'. The 'CalculadoraInt.java' tab is active, displaying the following Java code:

# Expresión lambda como retorno de métodos

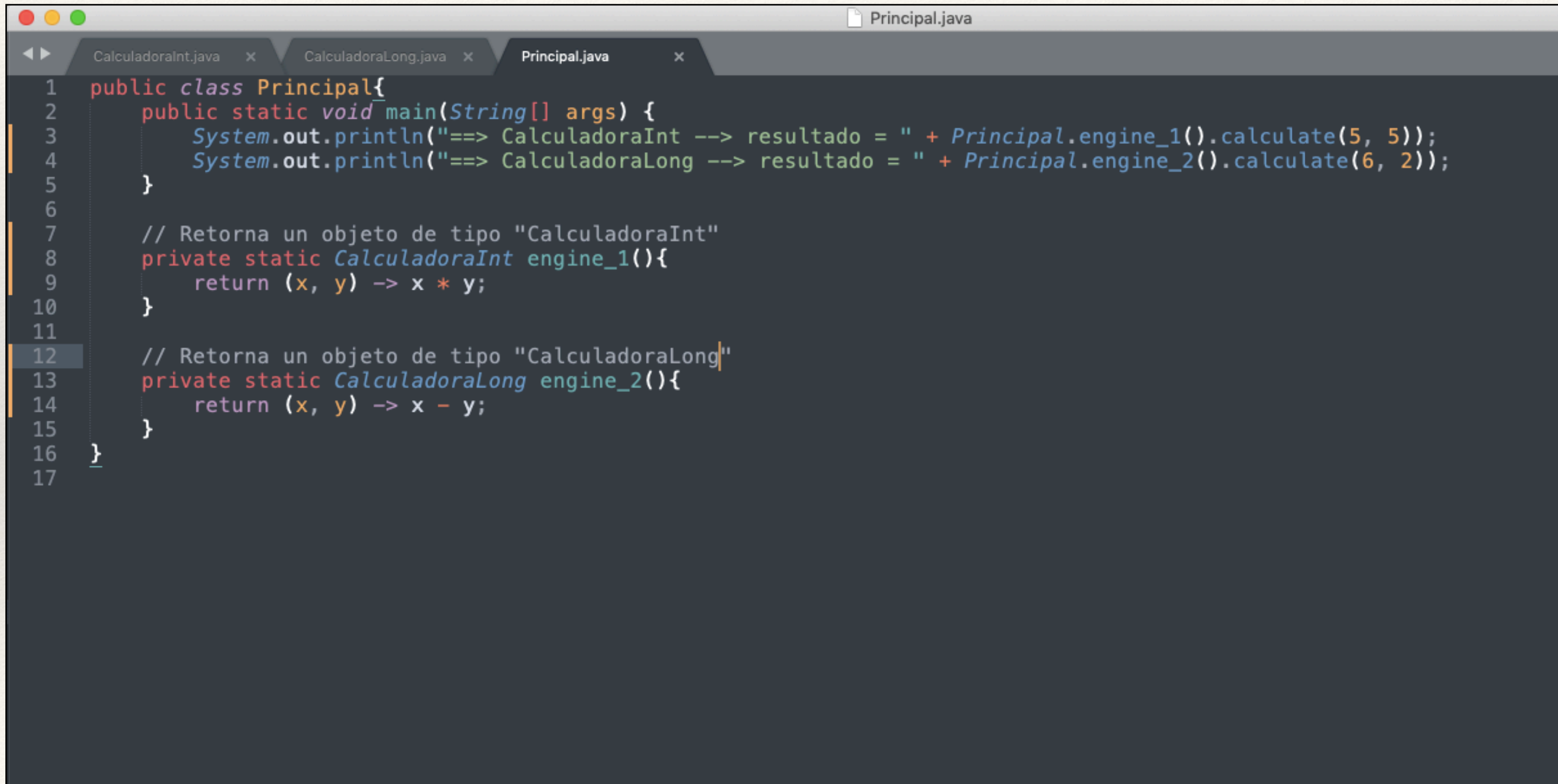
A screenshot of an IDE window titled 'CalculadoraLong.java'. The window shows a Java interface definition. The code is as follows:

```
1  /*  
2     Interfaz Funcional  
3  */  
4  
5  public interface CalculadoraLong{  
6      public long calculate(long x, long y);  
7  }  
8
```

The line numbers 1 through 8 are visible on the left side of the editor. The code is color-coded: keywords like 'public' and 'interface' are in red, 'long' is in blue, and 'calculate' is in green. The interface name 'CalculadoraLong' is in orange. The method signature 'calculate(long x, long y);' is in green. The comment '/\* Interfaz Funcional \*/' is in white. The line numbers 1 through 8 are in white. The IDE window has a standard macOS-style title bar with red, yellow, and green buttons. There are three tabs open: 'CalculadoraInt.java', 'CalculadoraLong.java' (active), and 'Principal.java'.



# Expresión lambda como retorno de métodos



```
1 public class Principal{
2     public static void main(String[] args) {
3         System.out.println("==> CalculadoraInt --> resultado = " + Principal.engine_1().calculate(5, 5));
4         System.out.println("==> CalculadoraLong --> resultado = " + Principal.engine_2().calculate(6, 2));
5     }
6
7     // Retorna un objeto de tipo "CalculadoraInt"
8     private static CalculadoraInt engine_1(){
9         return (x, y) -> x * y;
10    }
11
12    // Retorna un objeto de tipo "CalculadoraLong"
13    private static CalculadoraLong engine_2(){
14        return (x, y) -> x - y;
15    }
16 }
17
```

# Expresión lambda como retorno de métodos

The image shows a Java IDE with three tabs: `CalculadoraInt.java`, `CalculadoraLong.java`, and `Principal.java`. The `Principal.java` file is open, displaying the following code:

```
1 public class Principal{
2     public static void main(String[] args) {
3         System.out.println("==> CalculadoraInt --> resultado = " + Principal.engine_1().calculate(5, 5));
4         System.out.println("==> CalculadoraLong --> resultado = " + Principal.engine_2().calculate(6, 2));
5     }
6
7     // Retorna un objeto de tipo CalculadoraInt
8     private static CalculadoraInt engine_1() {
9         return (x, y) -> x * y;
10    }
11
12    // Retorna un objeto de tipo CalculadoraLong
13    private static CalculadoraLong engine_2() {
14        return (x, y) -> x - y;
15    }
16 }
17
```

Below the code editor, a terminal window titled `_Ejemplo_005_ — fm5@Air-Fhm5-5 — .._Ejemplo_005_ — -zsh — 95x26` shows the execution of the program:

```
└─[fm5 Air-Fhm5-5] - [~/Documents/_reposGit_github/_Programacion_Logica_y_Funcional/_Ejemplo_005_] - [Tue Mar 03, 16:02]
└─[$] <> javac Principal.java
└─[fm5 Air-Fhm5-5] - [~/Documents/_reposGit_github/_Programacion_Logica_y_Funcional/_Ejemplo_005_] - [Tue Mar 03, 16:02]
└─[$] <> java Principal
==> CalculadoraInt --> resultado = 25
==> CalculadoraLong --> resultado = 4
└─[fm5 Air-Fhm5-5] - [~/Documents/_reposGit_github/_Programacion_Logica_y_Funcional/_Ejemplo_005_] - [Tue Mar 03, 16:02]
└─[$] <> |
```



---

# Expresión lambda como retorno de métodos

---

¿Cómo podríamos modificar el ejercicio para poder hacerlo con Sobrecarga de Métodos?