7.G. Anexos de ejercicios resueltos.

4. Anexo IV.- Contextos del modificador final.

Distiintos contextos en los que puede aparecer el modificador final	
Lugar	**Funciõn**
Como modificador de clase.	La clase no puede tener subclases.
Como modificador de atributo.	El atributo no podrá ser modificado una vez que tome un valor. Sirve para definir constantes.
Como modificador al declarar un método	El método no podrá ser redefinido en una clase derivada.
Como modificador al declarar una variable referencia.	Una vez que la variable tome un valor referencia (un objeto), no se podrá cambiar. La variable siempre apuntará al mismo objeto, lo cual no quiere decir que ese objeto no pueda ser modificado internamente a través de sus métodos. Pero la variable no podrá apuntar a otro objeto diferente.
Como modificador en un parámetro de un método.	El valor del parámetro (ya sea un tipo primitivo o una referencia) no podrá modificarse dentro del código del método.

Veamos un ejemplo de cada posibilidad:

1. **Modificador de una clase.**

```
public final class ClaseSinDescendencia {     // Clase “no heredable”
    …
}
```

2. **Modificador de un atributo.**

```
public class ClaseEjemplo {
   // Valor constante conocido en tiempo de compilación
   final double PI= 3.14159265;

   // Valor constante conocido solamente en tiempo de ejecución
   final int SEMILLA= (int) Math.random()*10+1;
   …
}
```

3. **Modificador de un método**.

```
public final metodoNoRedefinible (int parametro1) {   // Método “no redefinible”
    …
}
```

4. **Modificador en una variable referencia.**

```
// Referencia constante: siempre se apuntará al mismo objeto Alumno recién creado,
// aunque este  objeto pueda sufrir modificaciones.
final Alumno PRIMER_ALUMNO= new Alumno (“Pepe”, “Torres”, 9.55);   // Ref. constante

// Si la variable no es una referencia (tipo primitivo), sería una constante más
// (como un atributo constante).
final int NUMERO_DIEZ= 10;     // Valor constante (dentro del ámbito de vida de la variable)
```

5. **Modificador en un parámetro de un método.**

```
void metodoConParametrosFijos (final int par1, final int par2) {
    // Los parámetros “par1” y “par2” no podrán sufrir modificaciones aquí dentro
    …
}
```