Ejercicio 2.1 Empleados

```
public class EmpleadoTemporario {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    public double horasTrabajadas = 0;
    public int cantidadHijos = 0;
    // ......

        public double sueldo() {
            return this.sueldoBasico
                + (this.horasTrabajadas * 500)
                + (this.cantidadHijos * 1000)
                - (this.sueldoBasico * 0.13);
        }
}

public class EmpleadoPlanta {
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    public int cantidadHijos = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico
                + (this.cantidadHijos * 2000)
                - (this.sueldoBasico * 0.13);
    }
}

public class EmpleadoPasante {
```

```
    public String nombre;
    public String apellido;
    public double sueldoBasico = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico - (this.sueldoBasico * 0.13);
    }
}
```

Paso uno: Análisis

- Las tres clases comparten las variables: **nombre**, **apellido**, y **sueldoBasico**. Esto junto al nombre de clase parecido que tienen hace pensar que se podría crear una superclase que generalice algunos de los datos → Extract Superclass, Pull Up Field
- Además, en las tres clases se define un método **sueldo()**. Este método está planteado diferente en cada clase, pero en todos los casos se mantiene igual la parte "*this.sueldoBasico - (this.sueldoBasico * 0.13)*" → Slide Statements, Extract Function, Replace Inline Code with Function Call, Pull Up Method
- Otra cosa que se podría hacer es crear otra jerarquía para aquellos empleados que tengan la variable cantidadHijos, pero creo con esto basta.

Paso dos: Refactoring

1. **Code Smell:** Código Repetido → Las tres clases comparten tres variables
    a. **Refactoring a aplicar:** Extract Superclass → Crear una clase abstracta Empleado. Volverla clase padre de las tres clases
    b. **Refactoring a aplicar:** Pull Up Field → Escribir las variables repetidas en la clase Empleado (declararlas como *protected*). Borrar las variables de cada subclase
2. **Code Smell:** Código Repetido → Las tres clases comparten un método que hace lo mismo en una parte

a. **Refactoring a aplicar:** Slide Statements → Poner juntas las sentencias que tengan que ver con la resta, como por ejemplo en **EmpleadoTemporario** y **EmpleadoPlanta**.

b. **Refactoring a aplicar:** Extract Function → Crear una nueva función que lleve el nombre de **sueldoBasico()**. Copiar dentro "*return this.sueldoBasico - (this.sueldoBasico * 0.13);*"

   // El nombre es discutible

c. **Refactoring a aplicar:** Replace Inline Code with Function Call → Reemplazar la sentencia copiada anteriormente de cada método con una llamada al nuevo método **sueldoBasico()**

d. **Refactoring a aplicar:** Pull Up Method → Subir el método a la superclase (y declararlo protected). Borrar el método de las subclases

e. Crear un método abstracto sueldo() en la superclase

   //No es un refactoring en sí

Resultado final

```
public abstract class Empleado {
   protected String nombre;
   protected String apellido;
   protected double sueldoBasico = 0;

   public abstract double sueldo();

   protected double sueldoBasico() {
      return this.sueldoBasico - (this.sueldoBasico * 0.13);
   }
}
//-------------------------------------------------------
public class EmpleadoPasante extends Empleado {
   // ......
```

```
    public double sueldo() {
        return this.sueldoBasico();
    }
}
//-------------------------------------------------------
public class EmpleadoPlanta extends Empleado {
    public int cantidadHijos = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico()
                + (this.cantidadHijos * 2000);
    }
}
//-------------------------------------------------------
public class EmpleadoTemporario extends Empleado {
    public double horasTrabajadas = 0;
    public int cantidadHijos = 0;
    // ......

    public double sueldo() {
        return this.sueldoBasico()
            +  (this.horasTrabajadas * 500)
            +  (this.cantidadHijos * 1000);
    }
}
```