BAD SMELLS IN CODE

"If it stinks, change it."

..:: What we talk about today? ::..

Principles in Refactoring

- Defining Refactoring
- Why Should You Refactor?
- When Should You Refactor?
- Refactoring and Design.
- Refactoring and Performance

Bad Smells in Code

- Comments
- Variables
- Duplicated Code
- Long Method
- Long Parameter List
- Switch Statement

- No se refiere a bugs o errores en el código.
- Se refiere a problemas en un mal diseño que afectan el desempeño, aumentan la complejidad de lectura y por lo tanto del mantenimiento e incluso riesgos a errores en el futuro.
- Desarrollar nuestro sentido olfativo de código es algo que probablemente todos los desarrolladores hacemos sin darnos cuenta, y vamos adoptando lo que creemos son nuestras "mejores prácticas", que si bien es cierto nos funcionan, pero probablemente no son la mejor manera de lograr una solución aun problema.

Veámos un ejemplo…

Analicemos este código…

```
public class Empleado {


    String nombreDelEmpleado; //variable nombre del empleado
    int edadDelEmpleado; // variable edad del empleado
    String puestoDelEmpleado ; // variable puesto del empleado
    double bonoPorPuntualidadDelEmpleado;// variable bono por puntualidad
    int ht;// variable horas trabajadas del empleado
    int pxhe; //variable pago por hora extra del empleado
    double sb;// variable sueldo base
    double st; // variable sueldo total

    //Este metodo calcula el sueldo total del empleado
    public double calcularSueldoTotal(double sueldoBase, int horasTrabajadas, int pagoPorHoraExtra,double bono){

        if (horasTrabajadas >40) {
            int horasExtras = horasTrabajadas - 40;
            st = sueldoBase + horasExtras*pagoPorHoraExtra+bono;
            return st;
        }// end if
        else
        {
            st = sueldoBase +bono;
            return st;
        }//end else
```

Una posible solución sería…

```
public class TestEmpleado {

    String nombre;
    String puesto;

    public double calcularSueldoTotal(double sueldoBase, int horasTrabajadas, int pagoExtra,double bono){

        double sueldoExtra = this.setSueldoExtra(horasTrabajadas, pagoExtra);
        double sueldoTotal = sueldoBase + sueldoExtra + bono;

        return sueldoTotal;

        }

    public double setSueldoExtra(int horasTrabajadas, int pagoExtra){

        int horasExtra = horasTrabajadas - 40;
        double sueldoExtra = (horasTrabajadas > 40) ? horasExtra * pagoExtra : 0;

        return sueldoExtra;
    }
```

Y qué con los Switch Statement?

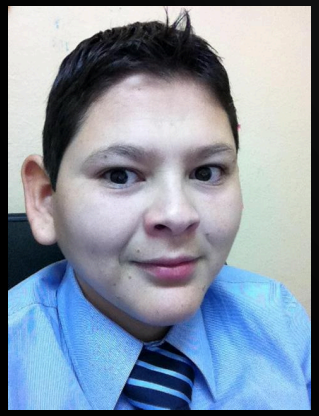
```
public static void main(String[] args) {
    int day = 8;

    switch (day) {
        case 1:
            System.out.println("Lunes");
        case 6:
            System.out.println("Sábado");
            break;
        case 7:
```

Una posible solución sería…

```
public class TestDay {
    public static void main(String[] args) {

        System.out.println( getDay(5) );

    }

    public static String getDay(int day){
        String days[] = {"Lunes", "Martes","Miercoles","Jueves","Viernes","Sábado","Domingo"};
        return days[day-1];
    }
}
```

```
            System.out.println("Viernes");
            break;
```

Gracias!!



Jesús Gabriel Pérez Pérez
jgperez@iswug.net
@gabrielp83

www.happyjavadevs.com