Clean Code

BLOQUES 1, 2 Y 3

VICENTE MENA MICAELO

1º DAM | IES Alonso de Avellaneda

Enlace al repositorio de GitHub: https://github.com/VicenteM23/CleanCode1Ev_Vicente/

Vicente Mena Micaelo

Índice

Ejemplo 1

En el primer ejemplo que encontramos de código sucio, vemos que este no es autoexplicativo y además de ello no disponemos de ningún comentario que nos ayude a la hora de saber que es lo que está realizando el código.

```
    public static void main(String[] args) {
        Scanner scan = new Scanner(source: System.in);
            String i;
            System.out.print(s: "Inserta el número de aciertos en 20 preguntas :" );
            i = scan.nextLine();
            double j = Double.valueOf(s: i);
            double f = 20-j;
            System.out.println("Número de errores: " + f);
            double h = f / 2;
            System.out.println(x: h);
            double k = (f - h)/2;
            if(k < 5)
                System.out.println("Calificación: " + k + " Insuficiente");
            else if(k >= 5 && k <6)
                System.out.println("Calificación: " + k + " Suficiente");
            else if(k >= 6 && k <7)
                System.out.println("Calificación: " + k + " Bien");
            else if(k >= 7 && k <9)
                System.out.println("Calificación: " + k + " Notable");
            else if(k >= 9)
                System.out.println("Calificación: " + k + " Sobresaliente");
    }

}
```

Para poder mejorar y limpiar este código podemos realizar lo siguiente:

1. Modificar los nombres de las variables por nombres con significado, fáciles de pronunciar y que se puedan buscar:

```
String numeroAciertos;
double numeroErrores = 20-numAciertosDouble;
double notaResta = numeroErrores / 2;
double notaResta = numeroErrores / 2;
double notaFinal = (numAciertosDouble - notaResta)/2;
```

2. Añadimos ciertos comentarios considerados necesarios, a pesar de que el código ya es autoexplicativo:

```
public static void main(String[] args) {
    Scanner scan = new Scanner(source: System.in);
    //Realizamos el proceso para solicitar al usuario que nos indique cuantos aciertos ha obtenido en su examen:
        String numeroAciertos;
        System.out.print(s: "Inserta el número de aciertos en 20 preguntas :" );
        numeroAciertos = scan.nextLine();
        double numAciertosDouble = Double.valueOf(s: numeroAciertos); /*Necesitamos realizar el cambio de tipo,
        ya que, hemos solicitado un String y nosotros vamos a trabajar con double*/

    /*Creamos y visualizamos por pantalla la variable que contiene el número de errores
    que el usuario ha obtenido:*/
        double numeroErrores = 20-numAciertosDouble;
        System.out.println("Número de errores: " + numeroErrores);

    /*Debido a que cada error resta 0.5 a la nota, es necesario dividir sus errores entre 2
    para obtener cuanta nota total se le restará, dicho valor lo almacenamos en otra variable:*/
        double notaResta = numeroErrores / 2;
        System.out.println(x: notaResta);

    /*Creamos nuestra última variable, que nos servirá para realizar nuestro condicional if,
    este le mostrará al usuario su nota numérica además de indicarle si esta es Insuficiente,
    Suficiente, Bien, Notable o Sobresaliente*/
        double notaFinal = (numAciertosDouble - notaResta)/2;

        if(notaFinal < 5)
            System.out.println("Calificación: " + notaFinal + " Insuficiente");

        else if(notaFinal >= 5 && notaFinal <6)
            System.out.println("Calificación: " + notaFinal + " Suficiente");

        else if(notaFinal >= 6 && notaFinal <7)
            System.out.println("Calificación: " + notaFinal + " Bien");

        else if(notaFinal >= 7 && notaFinal <9)
            System.out.println("Calificación: " + notaFinal + " Notable");

        else if(notaFinal >= 9)
            System.out.println("Calificación: " + notaFinal + " Sobresaliente");
}
```

Ejemplo 2

En el segundo ejemplo que encontramos de código sucio, vemos que este tampoco es autoexplicativo y además de ello no disponemos de ningún comentario que nos ayude a la hora de saber qué es lo que está realizando el código.

```
package cleancodeproject;
class c {
    double x;
    double y;
}
/**
 *
 * @author Vicente Mena
 */
public class EjemploCodigoSucio2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        c P = new c();
        c Q = new c();
        c R = new c();
        c O = new c();
        P.x=1.1;
        P.y=2.2;
        Q.x=4.7;
        Q.y=7.4;
        R.x=0.5;
        R.y=3.5;
        O.x=0;
        O.y=0;
        double h = Math.sqrt((Math.pow(a: P.x, b: 2))+(Math.pow(a: P.y, b: 2)));
        double i = Math.sqrt((Math.pow(a: Q.x, b: 2))+(Math.pow(a: Q.y, b: 2)));
        double j = Math.sqrt((Math.pow(a: R.x, b: 2))+(Math.pow(a: R.y, b: 2)));
        System.out.printf(format: "P (%1.1f:%1.1f) - O (%1.1f:%1.1f) = %1.3f\n", args: P.x , args: P.y , args: O.x , args: O.y , args: h);
        System.out.printf(format: "Q (%1.1f:%1.1f) - O (%1.1f:%1.1f) = %1.3f\n", args: Q.x , args: Q.y , args: O.x , args: O.y , args: i);
        System.out.printf(format: "R (%1.1f:%1.1f) - O (%1.1f:%1.1f) = %1.3f\n", args: R.x , args: R.y , args: O.x , args: O.y , args: j);
        double PQ = Math.sqrt((Math.pow((P.x-Q.x),b: 2 ) + Math.pow((P.y-Q.y),b: 2 )));
        System.out.printf(format: "P: (%1.1f:%1.1f) - Q: (%1.1f:%1.1f) = %02.5f\n", args: P.x , args: P.y ,args: Q.x , args: Q.y , args: PQ);
        double PR = Math.sqrt((Math.pow((P.x-R.x),b: 2 ) + Math.pow((P.y-R.y),b: 2 )));
        System.out.printf(format: "P: (%1.1f:%1.1f) - R: (%1.1f:%1.1f) = %02.5f\n", args: P.x , args: P.y ,args: R.x , args: R.y , args: PR);
        double RQ = Math.sqrt((Math.pow((R.x-Q.x),b: 2 ) + Math.pow((R.y-Q.y),b: 2 )));
        System.out.printf(format: "P: (%1.1f:%1.1f) - Q: (%1.1f:%1.1f) = %02.5f\n", args: R.x , args: R.y ,args: Q.x , args: Q.y , args: RQ);

    }
```

Para poder mejorar y limpiar este código podemos realizar lo siguiente:

1. Modificar el nombre de la clase, objetos y variables por nombres con sentido, fáciles de pronunciar y buscar:

```
class Coordenadas {
    double x;
    double y;
}
Coordenadas PuntoP = new Coordenadas();
Coordenadas PuntoQ = new Coordenadas();
Coordenadas PuntoR = new Coordenadas();
Coordenadas PuntoO = new Coordenadas();
```

```
PuntoP.x=1.1;
PuntoP.y=2.2;

PuntoQ.x=4.7;
PuntoQ.y=7.4;

PuntoR.x=0.5;
PuntoR.y=3.5;

PuntoO.x=0;
PuntoO.y=0;
double distanciaPOrigen
double distanciaQOrigen
double distanciaROrigen
```

2. Añadir ciertos comentarios aclarativos, a pesar de que el código es autoexplicativo:

```
public static void main(String[] args) {
    Coordenadas PuntoP = new Coordenadas();
    Coordenadas PuntoQ = new Coordenadas();
    Coordenadas PuntoR = new Coordenadas();
    Coordenadas PuntoO = new Coordenadas();

    //Asignamos Valores
    PuntoP.x=1.1;
    PuntoP.y=2.2;

    PuntoQ.x=4.7;
    PuntoQ.y=7.4;

    PuntoR.x=0.5;
    PuntoR.y=3.5;

    PuntoO.x=0;
    PuntoO.y=0;

    //Calculamos la distancia al origen de cada punto:
    double distanciaPOrigen = Math.sqrt((Math.pow(a: PuntoP.x, b: 2))+(Math.pow(a: PuntoP.y, b: 2))); //Distancia a origen del punto P
    double distanciaQOrigen = Math.sqrt((Math.pow(a: PuntoQ.x, b: 2))+(Math.pow(a: PuntoQ.y, b: 2))); //Distancia a origen del punto Q
    double distanciaROrigen = Math.sqrt((Math.pow(a: PuntoR.x, b: 2))+(Math.pow(a: PuntoR.y, b: 2))); //Distancia a origen del punto R

    //Imprimimos por pantalla con formato los resultados de cada punto:
    System.out.printf(format: "Distancia desde PuntoP (%1.1f:%1.1f) al Punto de Origen (%1.1f:%1.1f) = %1.3f\n", args: PuntoP.x , args: PuntoP.y , args: PuntoO.x , args: PuntoO.y , args: distanciaPOrigen);
    System.out.printf(format: "Distancia desde PuntoQ (%1.1f:%1.1f) al Punto de Origen (%1.1f:%1.1f) = %1.3f\n", args: PuntoQ.x , args: PuntoQ.y , args: PuntoO.x , args: PuntoO.y , args: distanciaQOrigen);
    System.out.printf(format: "Distancia desde PuntoR (%1.1f:%1.1f) al Punto de Origen (%1.1f:%1.1f) = %1.3f\n", args: PuntoR.x , args: PuntoR.y , args: PuntoO.x , args: PuntoO.y , args: distanciaROrigen);

    //Hacemos la operación para obtener la distancia entre los puntos P y Q:
    double resultadoPQ = Math.sqrt((Math.pow((PuntoP.x-PuntoQ.x),b: 2 ) + Math.pow((PuntoP.y-PuntoQ.y),b: 2 )));
    System.out.printf(format: "La distancia entre PuntoP: (%1.1f:%1.1f)y PuntoQ: (%1.1f:%1.1f) es : %02.5f\n", args: PuntoP.x , args: PuntoP.y ,args: PuntoQ.x , args: PuntoQ.y , args: resultadoPQ);

    //Hacemos la operación para obtener la distancia entre los puntos P y R:
    double resultadoPR = Math.sqrt((Math.pow((PuntoP.x-PuntoR.x),b: 2 ) + Math.pow((PuntoP.y-PuntoR.y),b: 2 )));
    System.out.printf(format: "La distancia entre PuntoP: (%1.1f:%1.1f)y PuntoR: (%1.1f:%1.1f) es : %02.5f\n", args: PuntoP.x , args: PuntoP.y ,args: PuntoR.x , args: PuntoR.y , args: resultadoPR);

    //Hacemos la operación para obtener la distancia entre los puntos R y Q:
    double resultadoRQ = Math.sqrt((Math.pow((PuntoR.x-PuntoQ.x),b: 2 ) + Math.pow((PuntoR.y-PuntoQ.y),b: 2 )));
    System.out.printf(format: "La distancia entre PuntoP: (%1.1f:%1.1f)y PuntoQ: (%1.1f:%1.1f) es : %02.5f\n", args: PuntoR.x , args: PuntoR.y ,args: PuntoQ.x , args: PuntoQ.y , args: resultadoRQ);

}
```