CASO PRÁCTICO EVALUABLE UNIDAD 3

ACLARACIÓN GENERAL

Ejercicio **1, 2 y 3** en un **pdf**. (Diagrama y explicaciones)

- **Diagrama UML**, (imagen) incrustada en este pdf, con los correspondientes atributos, métodos, etc. (según se indica).
- Especificar los que son privados, públicos o protegidos con notación adecuada en el diagrama y explicar el motivo de ponerlos así.
- Establecer **relaciones** entre las clases del diagrama **UML** y explicar

Ejercicio **4**.(Código Java)

- Pasar a código Java, estas clases, pero al pasarlas a código Java **de momento** las **trataremos como clases independientes** ya que **todavía no hemos visto la unidad 7** en la que aprenderemos a **establecer relaciones** entre clases(agregación, composición, herencia) **con código Java**.

Usar la **plantilla ejemplo que se encuentra en el foro "**Avisos**" en el hilo **"Pautas para realizar la tarea evaluable de la unidad 3"**

PLANTILLA PARA REALIZACIÓN EJERCICIO N°4

(en este ejercicio solo se entrega un archivo .java)

En esta unidad, **todavía no representaremos las relaciones entre clases con código Java y las trataremos como **clases independientes**, aunque en el **diagrama** **UML SI se DEBE** establecer algún tipo de relación de las vistas en la unidad **(Agregación, composición, herencia…)**.

En la **unidad 7** veremos cómo representar este tipo de relaciones con **código Java**.

Si hay alguna duda por favor, indicármelo.

Un saludo.
Maricarmen.

EJEMPLO PLANTILLA GUÍA

```
public class Main {
    // Clase interna 1
    static class Clase1 {
        private String atributo1;
        private int atributo2;
       //constructor sin parámetros
        public Clase1() {}
```

```
   //constructor con parámetros
    public Clase1(String atributo1, int atributo2) {
        this.atributo1 = atributo1;
        this.atributo2 = atributo2;
    }

    public String getAtributo1() {
        return atributo1;
    }

    public void setAtributo1(String atributo1) {
        this.atributo1 = atributo1;
    }



    public int getAtributo2() {
        return atributo2;
    }

    public void setAtributo2(int atributo2) {
        this.atributo2 = atributo2;
    }

    public void mostrarInfo() {
        System.out.println("Clase1 - Atributo1: " + atributo1 + ", Atributo2: " + atributo2);
    }
}

// Clase interna 2
static class Clase2 {
    private String atributo3;
    private double atributo4;

    public Clase2() {}

    public Clase2(String atributo3, double atributo4) {
        this.atributo3 = atributo3;
        this.atributo4 = atributo4;
```

```
    }

    public String getAtributo3() {
        return atributo3;
    }

    public void setAtributo3(String atributo3) {
        this.atributo3 = atributo3;
    }

    public double getAtributo4() {
        return atributo4;
    }




    public void setAtributo4(double atributo4) {
        this.atributo4 = atributo4;
    }

    public void mostrarInfo() {
        System.out.println("Clase2 - Atributo3: " + atributo3 + ", Atributo4: " + atributo4);
    }
}

// Método main
public static void main(String[] args) {
    //Clase1
   // Crear instancia de Clase1 usando constructor sin parámetros
    Clase1 firstOobjClass1 = new Clase1();
    firstOobjClass1.setAtributo1("Valor1");
    firstOobjClass1.setAtributo2(10);
   …
   …
    firstOobjClass1.mostrarInfo();

    // Crear instancia de Clase1 usando constructor con parámetros
    Clase1 secondObjClass1 = new Clase1("Valor2", 20);
    secondObjClass1.mostrarInfo();
```

```
        //Clase2
        // Crear instancia de Clase2 usando constructor sin parámetros
        …
         // Crear instancia de Clase2 usando constructor con parámetros
        …
        Y así con las clases indicadas
    }
}
```

Los **nombres de métodos, de atributos y tipos se deben adaptar a lo que se ha representado en el diagrama UML, el código Java **debe ser imagen fiel** de los métodos y atributos representados en el **diagrama UML**, **salvo** por lo comentado en el aspecto de las **relaciones** entre clases (que las **implementaremos** en Java cuando tratemos la **unidad 7**) aquí las trataremos como clases independientes(sin relación entre ellas).