CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 8

DIAGRAMAS DE ESTRUCTURA

PRÁCTICA 01 [NO EVALUABLE]

DIAGRAMAS DE CLASES (I)

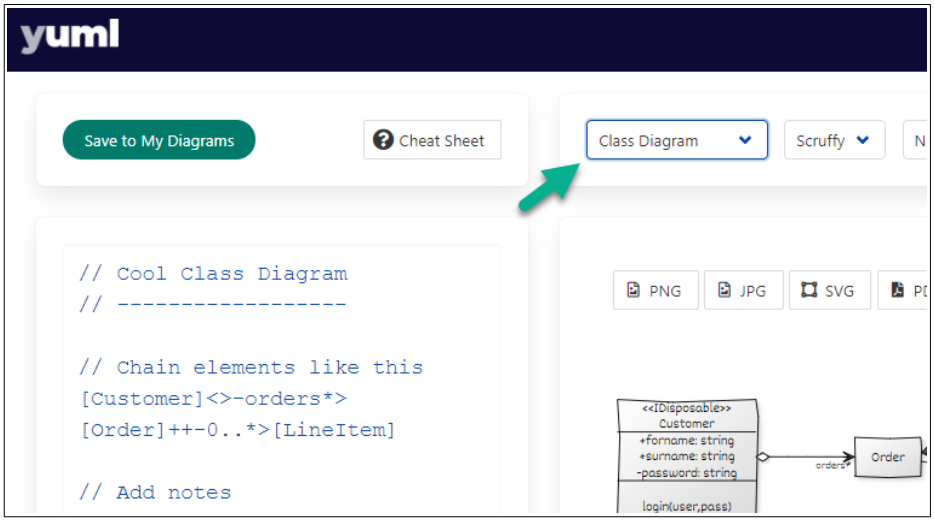
Revisado por:
Sergio Badal
Autores:
Cristina Álvarez, Fco. Javier Valero Garzón, M.ª Carmen Safont, Paco Aldarias
Fecha:
05/02/21

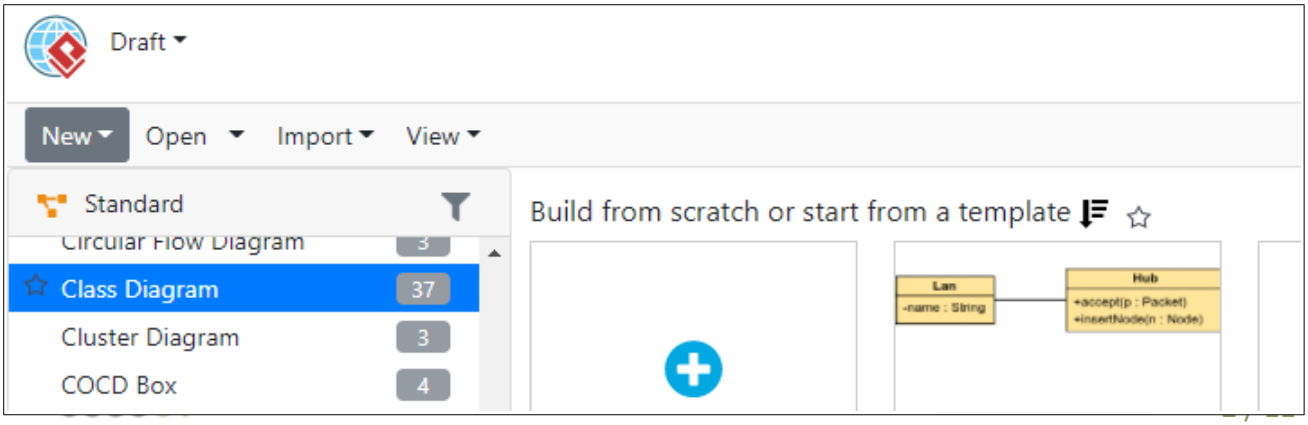
PRÁCTICA 01: DIAGRAMAS DE CLASES (I)	UD 08: DIAGRAMAS DE ESTRUCTURA
Contenidos: 1. Justificación de la práctica 2. Decálogo de recomendaciones 3. EJERCICIOS: 1. VEHÍCULO (CON SOLUCIÓN) 2. PERSONA (CON SOLUCIÓN) 3. EMPLEADO (CON SOLUCIÓN) 4. Bibliografía	
PRÁCTICA NO EVALUABLE	

1. JUSTIFICACIÓN DE LA PRÁCTICA

El diagrama de clases es vital para la definición del sistema. Las clases representan entidades de forma estática. Una clase puede contener atributos, propiedades y métodos.

En esta práctica vamos a realizar diagramas de clases en Visual Paradigm, YUML y DRAW.IO, **aunque puedes usar cualquier otro editor**.





2. DECÁLOGO DE RECOMENDACIONES

[clases]

1. Nombra las clases con sustantivos en singular, UpperCamelCase y en cursiva si son abstractas.
2. No incluyas las clases que no representen una entidad del sistema como “main”, “test”

[atributos/campos/propiedades]

3. Nombra los atributos con sustantivos lowerCamelCase y en cursiva si son abstractos.
4. Los atributos de una clase suelen ser privados.
5. Los tipos de datos de los atributos suelen ser opcionales (diseño o implementación).

[métodos/operaciones/funciones]

6. Nombra los métodos con verbos lowerCamelCase y solo en cursiva si son abstractos.
7. No incluyas setters, getters, constructores ni destructores, salvo si te los piden.
8. Los métodos suelen ser públicos y los parámetros opcionales (diseño o implementación).

[relaciones/asociaciones]

9. Marca las asociaciones con un rombo relleno (composición) o vacío (agregación).
10. Etiqueta las asociaciones solo cuando sea necesario, con una, dos o hasta tres etiquetas.

		Mismo paquete		Otro paquete	
		Subclase	Otra	Subclase	Otra
-	private	*no*	*no*	*no*	*no*
#	protected	*sí*	*sí*	*sí*	*no*
+	public	*sí*	*sí*	*sí*	*sí*
~	*package*	*sí*	*sí*	*no*	*no*

3. EJERCICIOS

PTCA 1. EJERCICIO 1: VEHÍCULO	ENUNCIADO

Se pide realizar el diagrama de clases **EN VISUAL PARADIGM** correspondiente **a partir de un enunciado**, siguiendo las recomendaciones del decálogo salvo que te indiquen lo contrario.

Queremos obtener un sistema de información que almacene datos de:

- **Vehículos**: color, número de puertas y modelo.
- **Autos**: Los de vehículo y, además, descapotable (si/no) y las funciones de subir y bajar capota.
- **Camionetas**: Los de vehículo y, además, la tara (en kilos), carga (en kilos) y el método de cargar, con los kilos como parámetro de entrada.

Queremos un diagrama de clases algo especial, con todo el detalle posible:

- Incluye todos los setters, getters, y constructores.
- Ninguno de los atributos puede ser visible fuera de las clases, serán protegidos los de Vehículo y privados el resto.
- Los métodos serán todos públicos, excepto los constructores de Auto y Camioneta, que tendrán visibilidad de paquete.
- Indica los tipos de datos, pensando que el sistema se implementará en Java.

PTCA 1. EJERCICIO 2: PERSONA	ENUNCIADO

Se pide realizar el diagrama de clases **EN YUML** correspondiente **a partir de un código en Java**, incluyendo los setters y getters que estén indicados en el código, así como los tipos de datos y los parámetros de los métodos.

En circunstancias normales (en una empresa) serás tú quién tenga que interpretar ese código con, probablemente, poca o nula documentación asociada por lo que te recomendamos que intentes hacer el diagrama **SIN LEER ESTE TEXTO DE AYUDA EN CURSIVA.**

Por tanto:

1. Lee el código fuente e intenta entender qué hace.
2. No es necesario que crees un proyecto en Java y lo ejecutes pero, si lo quieres hacer para entender el código mejor, recuerda crear un proyecto, un paquete y una clase (un archivo .java) por cada clase (incluída la clase test).
3. Extrae un boceto del diagrama de clases (si quieres hazlo en papel)
4. Vuelve a este punto y lee el texto de ayuda
5. Crea el diagrama de clases definitivo en la aplicación que más te guste
6. Consulta la solución

TEXTO DE AYUDA:

Revisemos tu borrador juntos:

- *Básicamente tenemos la clase base persona, de la cual hereda la subclase empleado.*
- *La clase persona tiene dos atributos privados de tipo cadena de texto (String en Java).*
- *La clase empleado dispone de un atributo propio protegido sueldoBase y tres métodos propios que son getSueldo y el setter y getter correspondiente a ese atributo.*
- *De la (sub)clase empleado hereda la (sub)clase encargado que redefine el método getSueldo de su superclase empleado (para aplicar un aumento por ser encargado).*

```
public class Persona {
            private String nombre;
            private String puesto;
            public void setNombre(String nom) {
                        nombre = nom;
            }
            public String getNombre() {
                        return nombre;
            }
            public void setPuesto(String p) {
                        puesto = p;
            }
}
```

```
public class Empleado extends Persona {
            protected int sueldoBase;
            public void setSueldoBase(int s) {
                        sueldoBase = s;
            }
            public int getSueldo() {
                        return sueldoBase;
            }
}
```

```
public class Encargado extends Empleado {
            public int getSueldo() {
                        Double d = new Double(sueldoBase * 1.1);
                        return d. intValue();
            }
}
```

```
public class Test {
            public static void main(String args[]) {
                        //Empleado emp = new Empleado();
                        Encargado enc = new Encargado();
                        enc.setNombre("Juan Pérez");
                        enc.setSueldoBase(500);
                        enc.setPuesto("Jefe de Almacén");
                        System.out.println(enc.getSueldo());
            }
}
/* EJECUCIÓN :
550
*/
```

PTCA 1. EJERCICIO 3: EMPLEADO	ENUNCIADO

Se desea un diagrama de clases **EN DRAW.IO** para diseñar una aplicación en Java que tenga la clase abstracta empleado la cual contiene el método abstracto setSueldo, que será implementado por las clases hijas Gerente y Operario, de forma diferente como indica el código que te proporcionamos.

Todos los métodos deben aparecer en el diagrama para verificar su operación, excepto los constructores.

empleado.java es la clase base, los otros son las clases derivadas más la clase Test, que no es necesario que aparezca.

```
public abstract class Empleado {
            protected int sueldo;
            public int getSueldo()
            {
                        return sueldo;
            }
            public abstract void setSueldo();
}
```

```
public class Operario extends Empleado {
            public void setSueldo (){
                        this.sueldo = 5;
            }
}
```

```
public class Gerente extends Empleado {
            public void setSueldo(){
                        sueldo = 10;
            }
}
```

```
public class Test {
            public static void main(String args[]) {
                        Operario o = new Operario();
                        Gerente g = new Gerente();
                        o.setSueldo();
                        g.setSueldo();
                        System.out.println(o.getSueldo());
                        System.out.println(g.getSueldo());
            }
}
/* EJECUCIÓN : 510 */
```

4. BIBLIOGRAFÍA Y ENLACES

- Aldarias, F. (2012): “Entornos de desarrollo”, CEEDCV
- Casado, C. (2012):Entornos de desarrollo, RA-MA, Madrid
- Ramos, A.; Ramos, MJ (2014):Entornos de desarrollo, Garceta, Madrid
- Visual Paradigm,www.visual-paradigm.com