ERRORES FRECUENTES EXÁMENES DAW:

DIAGRAMAS DE CLASES

1. Nomenclatura[1]:
   a) Nombres de clase en plural
   b) No comienza por mayúscula
   c) Nombres de clase con caracteres ilegales: espacios en blanco, acentos, eñes, cmienza por cifra o contiene sólo cifras.
   d) No separa correctamente palabras[2]
   e) Cometer errores similares en los nombres de atributo y método, que deben además comenzar por minúscula.
2. No elegir correctamente las clases, poner atributos como clases o no poner todas las clases.[3]
3. Elegir mal los símbolos de relación:
   a) Poner agregación o composición (rombos blanco y negro) cuando se trata de una generalización ("es un"). [4]
   b) No poner nombre a una relación genérica (flecha o línea continua normal, sin rombo, sin triángulo). [5]
   c) Poner la relación en sentido opuesto al correcto, es decir, poner el rombo o el triángulo o la flecha en el extremo contrario.
   d) No usar correctamente la relación de dependencia[6]
4. No poner las multiplicidades en las relaciones o ponerlas en un sólo extremo o ponerlas en StarUML en un apartado que no es "Multiplicity" dentro de "Properties".
5. Poner nombre a las relaciones de Generalización, Agregación, Composición o Dependencia.[7]
6. Poner nombres de atributos con verbos, en lugar de con sustantivos
7. Poner nombres de métodos con sustantivos en lugar de con verbos.
8. Poner métodos en la casilla de atributos y viceversa.
9. No poner los tipos (Integer, Boolean, ...) de los atributos.
10. Poner varios atributos en lugar de un array.[8]
11. No poner tipo de retorno de un método que lo requiere.[9]
12. No poner argumentos a un método.[10]

---

1 La nomenclatura es importante porque del diagrama de clases, StarUML y el resto de herramientas UML generan automáticamente un esquema del programa JAVA con la definición de las clases. Si se ponen nombres ilegales, el código generado dará errores de compilación.

2 Cada palabra en un nombre de clase debe comenzar por mayúscula: EJ: **AnimalDeTiro**

3 En el enunciado los sustantivos serán nombres de clase o nombres de atributo. Será una clase cuando podemos deducir para ella unos atributos y métodos y además se ve que la clase va a tener varias instancias. La clase **Perro**, tendrá varias instancias, cada una con valores de atributo diferentes: Perro.nombre puede valer: "Toby", "Cuqui", "Chispa", ... para cada instancia de **Perro**. Los sustantivos serán atributos cuando sólo puede atribuírseles un valor: nombre de **Perro**, por ejemplo, sólo puede tener valores atómicos como los anteriores.

4 Un **Presidente** de una Comunidad de vecinos "es un" ***Cargo*** o "es una" **Persona**. Es una relación de herencia o generalización. Si se pone rombo, se estaría diciendo (mal) que las partes físicas en que se descompone la clase "***Cargo***" son las clases "**Presidente**", "**Vocal**" y "**Administrador**", cuando, en todo caso, las partes de un **Cargo** son **Cabeza, Tronco y Extremidades** ¿no? Éstas últimas sí llevarían rombo negro (composición).

5 Si la relación no tiene un símbolo concreto, se debe poner una relación genérica **con su nombre.**

6 La dependencia se pone cuando una clase genera o crea una instancia de la clase dependiente. Un Administrador convoca reuniones, luego habrá un método en **Administrador** que será `convocarReunión(fecha:String)` que, cuando se ejecute, generará una nueva instancia de la clase **Reunion**.

7 Estas relaciones no tienen nombre. Sus nombres son siempre implícitamente: "es un", "se compone de" o "Genera una instancia de", el nombre de estas relaciones se presupone siempre el mismo.

8 color1: String; color2: String; color3:String debe ponerse colorSemaforo:String[3];

9 EstaAbierto() debe retornar True o False, luego debe definirse como estaAbrierto(): Boolean

10 cambiaColorSemaforo(color:String) es lo correcto. Incorrecto definir tres métodos: cambiaColorVerde(); cambiaColorRojo(); cambiaColorAmarillo();

DIAGRAMAS DE SECUENCIA y DE COLABORACIONES

1. No usar las clases definidas en el diagrama de clases[11]
2. No poner los actores externos.
3. No elegir el nombre de método del desplegable (teclearlo a mano).[12]
4. Dirigir el mensaje con origen incorrecto.[13]
5. En un diagrama de instancia (no genérico):
   a) Poner condiciones o iteraciones genéricas.[14]
   b) Poner variables y no poner valores concretos ("rojo", "amarillo", ...) en los argumentos de un mensaje enviado.
   c) Poner variables y no poner valores de retorno concretos (True, False, ...) para un mensaje.[15]

DIAGRAMAS de ESTADO

1. No entender el concepto de Estado[16]
2. No usar los métodos y atributos de la clase para indicar transiciones (triggers) y acciones (Effects). [17]

11 Al poner una instancia de clase en el diagrama, se le da un nombre a la instancia (variable) y se elige a qué clase pertenece (no se teclea el nombre) pulsando en Classifier en la ventana Properties.

12 Si se hace el punto anterior correctamente, al dibujar un mensaje debe elegirse el método de un desplegable que aparece pulsando en el botón "=". Además esto evita usar de modo incorrecto métodos que no son de la Clase a la que se dirige el mensaje.

13 El origen del mensaje debe tener lógica. Es un programa Java. Hay que elegir bien quién es la clase o actor externo que pide, por ejemplo, abrir la barrera de un paso a nivel. No tiene sentido que la instancia de Barrera envíe un mensaje a la instancia de Semaforo para cambiar el color del semáforo. Una barrera no controla a un semáforo, es más propio que sea una clase encargada del control en general como PasoNivel la que envíe ese mensaje a semáforo.

14 Un diagrama de instancia representa una ejecución de un escenario con valores concretos. Puede ponerse una iteración, pero sin condiciones. Si se ejecuta un mensaje tres veces se pone *[3], pero no *[mientras t >0]. Las condiciones genéricas [t > a] no se ponen nunca en un diagrama de instancia.

15 Usar variables, condiciones e iteraciones condicionales sólo se pone en los diagramas genéricos, que tienen que considerar todos los posibles casos que pueden suceder.

16 Un estado, para una clase concreta, es un momento en el que sus atributos tienen unos valores típicos. Para un **Semaforo**, el estado abierto tiene el atributo color con un valor típico que es "verde".

17 Normalmente si un semáforo de un paso a nivel pasa del estado abierto a cerrado es porque alguien ejecuta un método. El trigger (hace que se cambie a otro estado) suele ser un método concreto, por ejemplo detectaTren(). La acción es poner el semáforo en rojo, que suele ser otro método cambiaColor("rojo"). Ambos deben estar definidos previamente en el diagrama de clases. Hay que usarlos.