Ejercicios de Relaciones entre clases

Programación — DAW

Ricardo Pérez López
IES Doñana

Curso 2025/2026

1. Dibujar un diagrama de clases UML que represente el modelo estático de cada uno de los sistemas del boletín de ejercicios de *Programación orientada a objetos*, incluyendo las clases, sus atributos, sus métodos y las relaciones entre las clases. Indicar de qué tipo son las relaciones (dependencia, agregación o composición) que se establecen entre las clases.
2. Usando herencia, crear las clases `ColeccionItems` y `ColeccionConexiones`, que sean subclases de `Coleccion`. Usarlas donde corresponda en lugar de `Coleccion`.
3. Diseñar la clase `Hora`, que representa un instante de tiempo compuesto por la hora (de 0 a 23) y los minutos. Dispone de los métodos:
   - `__init__(`*hora*`,` *minutos*`)`: construye un objeto con los datos pasados como argumentos.
   - `inc()`: incrementa el instante en un minuto y no devuelve nada.
   - `set_minutos(`*valor*`)`: asigna un valor (si es válido) a los minutos. Devuelve `True` o `False` según se haya podido modificar los minutos o no.
   - `set_hora(`*valor*`)`: asigna un valor (si está comprendido entre 0 y 23) a la hora. Devuelve `True` o `False` según se haya podido cambiar la hora o no.
   - `__str__()`: devuelve una cadena con la representación de la hora.
4. A partir de la clase `Hora` diseñada en el ejercicio anterior, implementar la clase `HoraExacta`, que incluye en la hora los segundos. Además de los métodos heredados desde la clase `Hora`, dispondrá de:
   - `__init__(`*hora*`,` *minutos*`,` *segundos*`)`: construye un objeto con los datos pasados como argumentos.

- `set_segundos(`*valor*`)`: asigna un valor (si está comprendido entre 0 y

  59) a los segundos. Devuelve `True` o `False` según se haya podido cambiar los segundos o no.

- `inc()`: incrementa la hora en un segundo.

5. Añadir a la clase `HoraExacta` un método que compare si dos horas (la invocante y otra pasada como argumento al método) son iguales o distintas. ¿Cómo debería llamarse ese método?

6. Crear la clase abstracta `Instrumento` que almacena en una lista las notas musicales de una melodía (dentro de una misma octava). El método concreto `add` añade nuevas notas musicales. La clase también dispone del método abstracto `interpretar` con la siguiente signatura:

   ```
   interpretar() -> None
   ```

   que, en cada subclase que herede de `Instrumento`, mostrará por la salida las notas musicales según las interprete. Las notas serán constantes estáticas definidas dentro de la clase `Nota`, de la siguiente forma:

   ```
   class Nota:
       DO = 'do'
       RE = 're'
       MI = 'mi'
       FA = 'fa'
       SOL = 'sol'
       LA = 'la'
       SI = 'si'
   ```

7. Crear la clase `Piano` como subclase de la clase `Instrumento` del ejercicio anterior.