Apunte – De UML a Código en Java

Objetivo de la actividad

En esta actividad consolidamos todo lo aprendido en Programación Orientada a Objetos (POO). El reto consiste en **pasar de la teoría a la práctica**, aplicando el análisis, el diseño y la implementación en Java.

Paso 1: Detectar clases y relaciones en un problema real

Cuando enfrentamos un enunciado, el primer paso es **identificar los sustantivos** (posibles clases) y los **verbos** (posibles métodos).

Ejemplo:
"Un estudiante se inscribe en una materia y rinde exámenes."

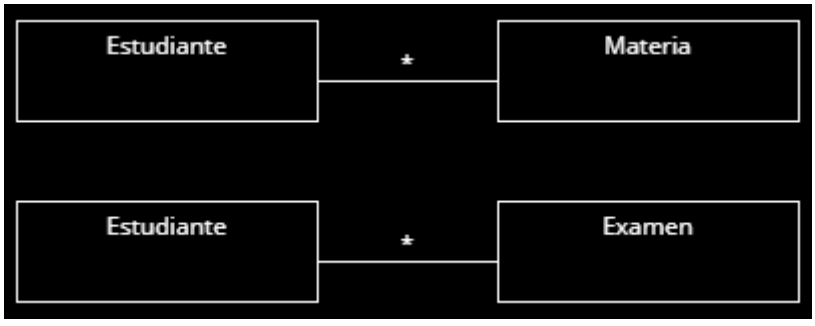
De aquí podemos extraer:

- **Clases:** Estudiante, Materia, Examen.
- **Relaciones:**
    - Un estudiante cursa una o varias materias.
    - Una materia puede tener varios exámenes.
    - El estudiante rinde exámenes.

Paso 2: Representar en UML

El **diagrama UML** nos permite ver la estructura antes de programar.

Ejemplo en UMLetino:



Paso 3: Pasar de UML a Código en Java

Una vez definido el diagrama, podemos escribir las clases en Java.

Ejemplo:

```
public class Estudiante {

    private String nombre;
    private String legajo;

    public Estudiante(String nombre, String legajo) {
        this.nombre = nombre;
        this.legajo = legajo;
    }

    public String getNombre() {
        return nombre;
    }
}

public class Materia {

    private String nombre;

    public Materia(String nombre) {
        this.nombre = nombre;
    }

    public String getNombre() {
        return nombre;
    }
}

public class Examen {

    private String tema;
    private int nota;

    public Examen(String tema, int nota) {
        this.tema = tema;
        this.nota = nota;
    }

    public int getNota() {
        return nota;
    }
}
```

Paso 4: Crear instancias y probar el programa

En una clase Main, probamos nuestro diseño:

```
public static void main(String[] args) {
    Estudiante el = new Estudiante("Ana", "2023-001");
    Materia ml = new Materia("Programación");
    Examen exl = new Examen("POO", 9);

    System.out.println(el.getNombre() + " rindió " + exl.getNota() + " en " + ml.getNombre());
}
```

Salida esperada:

Ana rindió 9 en Programación

Buenas prácticas

- **Detectar primero las clases**: evita programar sin modelo.
- **Aplicar encapsulamiento**: atributos privados + getters/setters cuando corresponda.
- **Mantener bajo acoplamiento y alta cohesión**.
- **Iterar y refactorizar**: si el código no refleja bien la realidad, vuelve al UML.