Práctico 3: Introducción a la Orientación a Objetos

Objetivo:

Comprender los fundamentos de la Programación Orientada a Objetos, incluyendo clases, objetos, atributos y métodos, para estructurar programas de manera modular y reutilizable en Java.

Resultados de aprendizaje:

1. Modelado de Clases y Objetos: Identificar y definir clases, objetos, atributos y métodos en Java para modelar entidades del mundo real, diferenciando entre clases y objetos, y comprendiendo el estado e identidad de los objetos.

2. Implementación de Clases y Métodos: Desarrollar clases y objetos en Java, aplicando atributos, métodos y principios básicos de encapsulación con buenas prácticas de estructuración de código.

3. Gestión del Estado e Identidad de Objetos: Manipular el estado de los objetos mediante la asignación y modificación de atributos, asegurando su correcta gestión en el programa.

4. Definición del Comportamiento de los Objetos: Implementar métodos en Java para definir acciones que los objetos pueden realizar, estructurando su comportamiento dentro del código.

5. Encapsulamiento y Diseño Modular: Aplicar el encapsulamiento para ocultar detalles internos de implementación, protegiendo el estado de los objetos y favoreciendo un diseño modular y seguro.

6. Uso de Modificadores de Acceso: Implementar y diferenciar los modificadores public, private y protected para controlar la visibilidad y el acceso a atributos y métodos dentro de las clases.

7. Manejo de Getters y Setters: Crear métodos get y set para acceder y modificar atributos privados, garantizando encapsulamiento y preservando la integridad de los datos.

¿Qué es una Kata y cómo se utiliza en programación?

Una kata es un ejercicio de programación diseñado para mejorar habilidades de codificación mediante la repetición y el aprendizaje progresivo. El término proviene de las artes marciales, donde las katas son secuencias de movimientos que se practican repetidamente para perfeccionar la técnica.

En programación, las katas ayudan a los programadores a reforzar conceptos, mejorar la comprensión del código y desarrollar buenas prácticas. **Se recomienda resolver una kata varias veces, intentando mejorar el código en cada iteración, utilizando mejores estructuras, nombres más claros y principios de diseño.**

Importante:

Intentar resolver cada kata sin mirar la solución.

Comprobar la solución y corregir errores si es necesario.

Repetir 2 o 3 veces para mejorar la comprensión, lógica y el código.

Experimentar con diferentes valores para reforzar el aprendizaje.

A continuación, te presento 5 enunciados de katas en Java para fortalecer los conceptos de clases, objetos, encapsulamiento, estado, identidad, modificadores de acceso y getters/setters.

Kata 1: Registro de Estudiantes (Nivel Básico)

Enunciado

Imagina que estás desarrollando un sistema de gestión de estudiantes para una plataforma de cursos.
Debes modelar la clase Estudiante con los siguientes atributos y métodos:

Atributos:

nombre (String)
apellido (String)

curso (String)
calificacion (double)

Métodos:

mostrarInfo(): Muestra la información del estudiante (apellido+nombre+curso+calificación)
subirCalificacion(double puntos): Aumenta la calificación del estudiante en la cantidad de puntos especificada (máximo 10).
bajarCalificacion(double puntos): Disminuye la calificación del estudiante (mínimo 0).

🎯 Tarea a realizar

- Crear una instancia de la clase Estudiante con un apellido, nombre, curso y calificación
- Mostrar la información del estudiante: apellido, nombre, curso y calificación
- Aumentar y disminuir la calificación con subirCalificacion() y bajarCalificacion().

Kata 2: Registro de Mascotas (Nivel Básico)

Enunciado

Imagina que estás desarrollando un sistema de gestión de mascotas para un refugio de animales.
Debes modelar la clase Mascota con los siguientes atributos y métodos:

- **Atributos:**

  nombre
  especie
  edad

- **Métodos:**

  mostrarInfo()
  cumplirAnios().

🎯 Tarea a realizar

- Crear una instancia de la clase Mascota con un nombre, especie y edad inicial.
- Mostrar la información de la mascota:nombre, especie y edad inicial.
- Llamar a cumplirAnios() para aumentar la edad en 1 año.
- Mostrar la información actualizada.

Kata 3: Primeros Pasos en Encapsulamiento (Nivel Intermedio)

Enunciado

Desarrolla una clase Libro para gestionar libros en una biblioteca.
Para evitar cambios incorrectos, implementa encapsulamiento.

- **Atributos privados:**

  título
  autor
  añoPublicacion.

- **Métodos públicos:**

  getTitulo()
  getAutor()
  getAñoPublicacion().
  Método **setAñoPublicacion(int nuevoAño)**, con validación: **No se puede modificar si el año es menor a 1900 o mayor al año actual.**

🎯 Tarea a realizar

1. Crear un objeto de la clase Libro.
2. Intentar modificar el año de publicación con un valor inválido y otro válido.
3. Mostrar la información del libro:: título, autor, añoPublicacion.

Kata 4: Estado e Identidad de los Objetos (Nivel Intermedio-Avanzado)

En esta actividad, vas a programar un sistema para gestionar gallinas en una granja digital, registrando su producción de huevos y envejecimiento.

Requisitos del modelo

Cada **gallina** tendrá los siguientes atributos:

- idGallina ⟶ Identificador único.
- edad ⟶ Representa la edad de la gallina en años.
- huevosPuestos ⟶ Cantidad total de huevos que ha puesto.

Además, contará con los siguientes métodos:

- ponerHuevo() ⟶ Incrementa en 1 la cantidad de huevos puestos por la gallina.
- envejecer() ⟶ Aumenta en 1 su edad.
- mostrarEstado() ⟶ Muestra en pantalla la información actual de la gallina (idGallina, edad, huevosPuestos).

🎯 Tarea a realizar

1. **Crear dos gallinas** diferentes, asignando un identificador único a cada una.
2. Hacer que **cada gallina ponga al menos un huevo**.
3. Hacer que **cada gallina envejezca un año**.
4. **Mostrar el estado** de cada gallina después de estas acciones.

Kata 5: Comportamiento de los Objetos (Nivel Avanzado)

Enunciado

- Imagina que trabajas en una **agencia espacial** y necesitas programar un simulador de **naves espaciales**.
- Debes desarrollar una clase **NaveEspacial** con un sistema de **combustible limitado**, lo que obliga a gestionar eficientemente los recursos.

Especificaciones

Atributos:

- nombre (String) ⟶ Nombre de la nave espacial.
- combustible (int) ⟶ Cantidad actual de combustible disponible.

Métodos:

- despegar(): Reduce 10 unidades de combustible al despegar. No puede despegar si hay menos de 10 unidades.
- avanzar(int distancia): Consume 1 unidad de combustible por cada unidad de distancia. No puede avanzar si no hay suficiente combustible.
- recargarCombustible(int cantidad): Aumenta la cantidad de combustible en la nave. No puede superar el límite máximo de 100 unidades.
- mostrarEstado(): Muestra el nombre de la nave y la cantidad de combustible actual.

Reglas:

✔️ No puede **despegar** con menos de 10 unidades de combustible.
✔️ No puede **avanzar** si el combustible no es suficiente para la distancia requerida.
✔️ No puede **sobrecargar combustible** más allá del límite de **100 unidades**.

🎯 Tarea a realizar

1. **Intentar resolver la kata sin mirar la solución**.

2. Crear una instancia de la clase **NaveEspacial** con un nombre y **50 unidades de combustible**.
3. Intentar **avanzar 60 unidades** sin recargar (debe fallar por falta de combustible).
4. Recargar **40 unidades** de combustible.
5. Intentar **avanzar 60 unidades nuevamente** (ahora debe funcionar).
6. Mostrar el **estado actual de la nave**.
7. Comparar con la solución y **ajustar el código** si es necesario.
8. **Repetir el ejercicio** con diferentes valores para reforzar la comprensión.