PROGRAMACIÓN II

Ejercicios adicionales: Programación Orientada a Objetos II

OBJETIVO GENERAL

Comprender y aplicar conceptos de Programación Orientada a Objetos en Java, incluyendo el uso de **this**, constructores, sobrecarga de métodos, encapsulamiento y miembros estáticos, para mejorar la modularidad, reutilización y diseño del código.

MARCO TEÓRICO

Concepto	Aplicación en el proyecto
Uso de this	Referencia a la instancia actual dentro de constructores y métodos
Constructores y sobrecarga	Inicialización flexible de objetos con múltiples formas de instanciación
Métodos sobrecargados	Definición de varias versiones de un método según los parámetros recibidos
toString()	Representación legible del estado de un objeto para visualización y depuración
Atributos estáticos	Variables compartidas por todas las instancias de una clase
Métodos estáticos	Funciones de clase invocadas sin instanciar objetos
Encapsulamiento	Ocultar la complejidad interna de un objeto, protegiendo sus datos mediante el uso de modificadores de acceso (private) y métodos públicos (getters y setters).

Caso Práctico 1

Sistema de Registro de Libros con Editorial

CLASE LIBRO

Atributos:

- **String titulo**
- **String autor**
- **static String editorial (por defecto “Independiente”)**

REQUERIMIENTOS

1. Uso de this:

   ○ Utilizar this en los constructores para distinguir parámetros de atributos.

2. Encapsulamiento:

   ○ Todos los atributos de la clase deben ser **`private`**. Proporcionar los métodos **`get`** y **`set`** públicos (`public`) necesarios para acceder o modificar los atributos, excepto aquellos que no deberían ser alterados desde fuera de la clase.

3. Constructores sobrecargados:

   - ○ Uno que reciba título y autor.
   - ○ Otro que reciba título, autor y editorial.

4. Métodos sobrecargados **actualizarTitulo:**

   - ○ Uno que reciba solo el nuevo título.
   - ○ Otro que reciba un prefijo y un nuevo título, generando: prefijo + " " + nuevoTitulo.
   - ○ Ejemplos de prefijo:
     - "Edición Especial" → Resultado: Edición Especial Cien Años de Soledad
     - "Versión Ilustrada" → Resultado: Versión Ilustrada Rayuela
     - "Colección Juvenil" → Resultado: Colección Juvenil El Principito
     - "Reimpresión 2024" → Resultado: Reimpresión 2024 Don Quijote

5. Método estático **cambiarEditorial(String nueva):** modifica la editorial común a todos los libros.

6. Método **toString():** mostrar título, autor y editorial.

TAREAS A REALIZAR

1. Crear libros con y sin editorial explícita.
2. Usar las dos variantes de actualizarTitulo.
3. Mostrar los libros con toString().
4. Cambiar la editorial global y volver a mostrar.

Caso Práctico 2

Sistema de Alumnos con Nota de Aprobación

CLASE ALUMNO

Atributos:

- **String nombre**
- **double promedio**
- **static double notaAprobacion (por defecto 6)**

REQUERIMIENTOS

1. Uso de this:
   - Utilizar this en los constructores para distinguir parámetros de atributos.
2. Encapsulamiento:
   - Todos los atributos de la clase deben ser **`private`**. Proporcionar los métodos **`get`** y **`set`** públicos (`public`) necesarios para acceder o modificar los atributos, excepto aquellos que no deberían ser alterados desde fuera de la clase.
3. Constructores sobrecargados:
   - Uno que reciba solo el nombre (promedio = 0).
   - Otro que reciba nombre y promedio.
4. Métodos sobrecargados **actualizarPromedio:**
   - Uno que reciba un nuevo promedio.
   - Otro que reciba un array de notas, y calcule el promedio.
5. Método aprobo(): retorna true si el promedio ≥ notaAprobacion.
6. Método estático cambiarNotaAprobacion(double nueva): cambia la nota mínima para aprobar.
7. Método toString(): mostrar nombre, promedio y si aprueba o no.

TAREAS A REALIZAR

1. Crear varios alumnos con ambos constructores.
2. Actualizar promedios con las dos variantes de método.
3. Imprimir resultados y verificar quién aprueba.

4. Cambiar la nota de aprobación y volver a probar.

Caso Práctico 3

Inventario de Productos con IVA

CLASE PRODUCTO

Atributos:

- **String nombre**
- **double precioBase**
- **static double IVA (0.21 = 21%)**

REQUERIMIENTOS

1. Uso de this:
   - Utilizar this en los constructores para distinguir parámetros de atributos.
2. Encapsulamiento:
   - Todos los atributos de la clase deben ser **`private`**. Proporcionar los métodos **`get`** y **`set`** públicos (`public`) necesarios para acceder o modificar los atributos, excepto aquellos que no deberían ser alterados desde fuera de la clase.
3. Constructores sobrecargados:
   - Uno que reciba nombre y precio base.
   - Otro que reciba solo el nombre (precio por defecto = 100).
4. Métodos sobrecargados **aplicarDescuento:**
   - Uno que reciba un porcentaje de descuento.
   - Otro que reciba un porcentaje y un precio mínimo, para no bajar de ese valor.
5. Método calcularPrecioFinal(): retorna el precio con IVA aplicado.
6. Método estático cambiarIVA(double nuevo): cambia el IVA para todos los productos.
7. Método toString(): mostrar nombre, precio base y precio final.

TAREAS A REALIZAR

1. Crear productos con ambos constructores.
2. Probar las dos variantes de aplicarDescuento.

3. Mostrar los productos con toString().
4. Cambiar el IVA global y recalcular precios finales.

Caso Práctico 4 - Banca Simple – Cuentas

CLASE CUENTA

Atributos:

- **int numero:** Número de cuenta autogenerado.
- **String titular:** Titular de la cuenta.
- **double saldo:** Saldo disponible.
- **static int ultimoNumero:** Base para autoincrementar la numeración. (por defecto 100)
- **static int totalCuentas:** Contador global de cuentas creadas. (por defecto 0)

REQUERIMIENTOS

1. Uso de this:
   - Utilizar this en los constructores para distinguir parámetros de atributos.
2. Encapsulamiento:
   - Todos los atributos de la clase deben ser **`private`**. Proporcionar los métodos **`get`** y **`set`** públicos (`public`) necesarios para acceder o modificar los atributos, excepto aquellos que no deberían ser alterados desde fuera de la clase.
3. Constructores sobrecargados:
   - Uno que reciba titular (saldo inicial 0).
   - Otro que reciba titular y saldoInicial.
   - Ambos asignan this.numero = ++ultimoNumero e incrementan totalCuentas.
4. Métodos sobrecargados **consultarSaldo:**
   - Uno que retorne el saldo actual en pesos (double).
   - Otro que reciba una cotización de dólar (double cotizacionDolar) y devuelva el saldo convertido a dólares.
   - Ejemplo:
     - Si el saldo es $2000 y la cotización del dólar es 1360 → consultarSaldo(1360) devuelve 1.47
5. Metodos depositar() y extraer(): aplicar validaciones para montos negativos o por fuera del límite disponible
6. Método estático mostrarTotalCuentas(): Retornar la cantidad total de cuentas creadas.
7. Método toString(): mostrar **número, titular, saldo** con dos decimales.

TAREAS A REALIZAR

1. Crear cuentas con ambos constructores.
2. Probar las dos variantes de consultarSaldo.
3. Mostrar las cuentas con toString().
4. Mostrar el total de cuentas generadas.