Ejercicios de Programación en Java

Ejercicio 1: Gestión Simple de Empleados

Enunciado: Crea una clase `Empleado` con propiedades para `nombre`, `salario` y `idEmpleado`. Implementa un método para mostrar la información del empleado. Luego, en el método `main`, crea un objeto `Empleado`, asígnale valores y muestra su información. Utiliza una estructura condicional para verificar si el salario es mayor a 2000 y, si lo es, aplica un bono del 10%.

Ejercicio 2: Calculadora Básica

Enunciado: Diseña una clase `Calculadora` con métodos para `sumar`, `restar`, `multiplicar` y `dividir` dos números. Cada método debe imprimir el resultado. En el método `main`, crea una instancia de `Calculadora` y realiza al menos dos operaciones diferentes. Añade una estructura condicional para evitar la división por cero.

Ejercicio 3: Control de Estudiantes

Enunciado: Crea una clase `Estudiante` con propiedades para `nombre`, `edad` y `notaPromedio`. Implementa un método que determine si el estudiante ha "Aprobado" (nota promedio >= 7) o "Reprobado". En el método `main`, crea varios objetos `Estudiante` y utiliza un bucle `for` para iterar sobre ellos y mostrar su estado de aprobación.

Ejercicio 4: Tienda de Productos

Enunciado: Define una clase `Producto` con propiedades `nombre`, `precio` y `cantidadEnStock`. Agrega un método para `vender` una cierta cantidad de productos (resta la cantidad del stock) y otro para `reponer` el stock. Asegúrate de que no se pueda vender más productos de los que hay en stock usando una estructura condicional. En el `main`, simula algunas ventas y reposiciones.

Ejercicio 5: Cuenta Bancaria Simple

Enunciado: Crea una clase `CuentaBancaria` con propiedades para `numeroCuenta`, `titular` y `saldo`. Implementa métodos para `depositar` y `retirar` dinero. El método `retirar` debe verificar que hay suficiente saldo antes de realizar la operación. En el `main`, realiza depósitos y retiros y muestra el saldo después de cada operación.

Ejercicio 6: Gestión de Tareas Pendientes

Enunciado: Crea una clase `Tarea` con propiedades `descripcion` y `completada` (un booleano). Agrega un método para `marcarComoCompletada()`. En el `main`, crea una lista de objetos `Tarea`. Utiliza un bucle `while` para permitir al usuario marcar tareas como completadas hasta que decida salir. Muestra el estado de todas las tareas después de cada interacción.

Ejercicio 7: Conversor de Unidades (Temperatura)

Enunciado: Diseña una clase `ConversorTemperatura` con métodos estáticos para convertir de Celsius a Fahrenheit y de Fahrenheit a Celsius. Los métodos deben recibir la temperatura como parámetro y retornar el valor convertido. En el `main`, pide al usuario una temperatura y la unidad de origen, y luego muestra la conversión.

Ejercicio 8: Juego de Adivinanza de Números

Enunciado: Crea una clase `JuegoAdivinanza` que genere un número aleatorio entre 1 y 100. El juego debe pedir al usuario que adivine el número. Utiliza un bucle `do-while` para que el juego continúe hasta que el usuario adivine correctamente. Proporciona pistas ("mayor" o "menor") después de cada intento incorrecto.

Ejercicio 9: Biblioteca de Libros

Enunciado: Crea una clase `Libro` con propiedades `titulo`, `autor` y `disponible` (booleano). Implementa métodos para `prestarLibro()` y `devolverLibro()`. El método `prestarLibro` debe verificar si el libro está disponible. En el `main`, crea varios objetos `Libro` y simula algunas operaciones de préstamo y devolución, mostrando el estado de disponibilidad de los libros.

Ejercicio 10: Gestión de Vehículos (Concesionario)

Enunciado: Diseña una clase `Vehiculo` con propiedades `marca`, `modelo` y `anioFabricacion`. Agrega un método para `mostrarDetalles()` del vehículo. En el `main`, crea una lista de al menos 5 objetos `Vehiculo`. Luego, utiliza un bucle `for-each` para recorrer la lista y mostrar los detalles de cada vehículo. Añade una estructura condicional para identificar y mostrar solo los vehículos fabricados después de 2020.