Programación orientada a objetos Individual - POO - Unidad 2
Actividad: Taller sobre el modificador **static** en Java
Tutor: JOHN CARLOS ARRIETA ARRIETA DOCENTE

Taller: El Modificador `static` en Java

Objetivo del Taller:

En este taller, los estudiantes aprenderán el uso correcto y la importancia del modificador `static` en Java. Se proporcionarán definiciones claras, se explicará cómo y cuándo utilizarlo, y se ofrecerán ejercicios prácticos para reforzar los conceptos. El taller está diseñado para ser pedagógico, utilizando un lenguaje claro y ejemplos prácticos que faciliten la comprensión.

Temario:

1. **Contextualización y Definición del Modificador `static`**
   - ¿Qué es el modificador `static`?
   - ¿Cómo se define y cuál es su comportamiento?
2. **Objetivos del Uso de `static`**
   - Compartir datos entre todas las instancias.
   - Definir métodos que no dependen de instancias.
3. **Cuándo Usar y Cuándo No Usar el Modificador `static`**
   - Casos recomendados y no recomendados.
4. **Ejemplos de Uso Correcto**
   - Ejemplos prácticos con atributos y métodos `static`.
5. **Ejemplos de Uso Incorrecto con Errores de Compilación**
   - Ejemplos que violan las reglas de `static` y no compilan.
6. **Ejemplos de Uso Incorrecto sin Generar Error de Compilación**
   - Ejemplos que son malas prácticas pero que no generan errores.
7. **Ejercicios Propuestos**
   - Ejercicios para poner en práctica el uso correcto e incorrecto del modificador `static`.

---

1. Contextualización y Definición del Modificador `static`

¿Qué es `static` en Java?

En Java, el modificador `static` se utiliza para definir miembros de una clase (atributos o métodos) que pertenecen a la clase en sí misma, en lugar de a las instancias (objetos) de la clase. Los miembros estáticos se comparten entre todas las instancias de la clase y se pueden acceder sin necesidad de crear un objeto.

¿Cómo se define `static`?

Se declara un miembro de la clase como `static` al colocar la palabra clave `static` antes de su tipo. Por ejemplo:

```
public class MiClase {
    static int contador = 0; // Atributo estático

    static void incrementarContador() { // Método estático
        contador++;
    }
}
```

En este caso, `contador` es un atributo compartido por todos los objetos de `MiClase`, y `incrementarContador` es un método que puede ser llamado sin crear una instancia de la clase.

2. Objetivos del Uso de `static`

El uso del modificador `static` tiene objetivos específicos y cumple funciones clave en la programación orientada a objetos:

- **Compartir Datos:** Permite que un atributo sea compartido entre todas las instancias de la clase. Por ejemplo, un contador que lleva la cuenta de cuántos objetos se han creado.
- **Métodos Independientes de Instancia:** Se utilizan para crear métodos que pueden ser llamados directamente desde la clase sin necesidad de crear un objeto, como los métodos de utilidad (por ejemplo, métodos matemáticos).

Visibilidad y Alcance de `static`

Los miembros `static` pertenecen a la clase, por lo que se accede a ellos utilizando el nombre de la clase, seguido de un punto (`.`). Por ejemplo:

```
MiClase.incrementarContador();
```

**3. Cuándo Usar y Cuándo No Usar el Modificador `static`

Cuándo Usar `static`:

- **Para Definir Constantes:** Las constantes (`final`) que son valores fijos a lo largo de la ejecución del programa y que pertenecen a la clase.
- **Métodos de Utilidad:** Métodos que no dependen de los atributos de una instancia (por ejemplo, métodos matemáticos como `Math.sqrt()`).
- **Contadores o Acumuladores:** Para llevar un registro compartido entre todas las instancias de una clase.

Cuándo No Usar `static`:

- **En Atributos Específicos de Instancias:** Cuando el valor de un atributo debe ser diferente para cada objeto.
- **En Métodos que Acceden a Datos de Instancia:** Los métodos que dependen de atributos no estáticos no pueden ser declarados como `static`.

4. Ejemplos de Uso Correcto

Ejemplo Correcto 1: Atributo `static` Compartido entre Instancias

```
public class Contador {
    static int contadorGlobal = 0; // Atributo estático

    public Contador() {
        contadorGlobal++; // Incrementa el contador global cada vez que se crea un objeto
    }

    public static void mostrarContador() {
        System.out.println("Contador global: " + contadorGlobal);
    }
}

public class Main {
    public static void main(String[] args) {
        Contador c1 = new Contador();
        Contador c2 = new Contador();
        Contador c3 = new Contador();

        // Se accede al método estático usando el nombre de la clase
        Contador.mostrarContador(); // Muestra: Contador global: 3
    }
}
```

Explicación: Aquí, `contadorGlobal` es un atributo `static` que se comparte entre todas las instancias de `Contador`. No importa cuántos objetos se creen, el valor de `contadorGlobal` es el mismo para todas las instancias.

Ejemplo Correcto 2: Método de Utilidad `static`

```
public class Calculadora {
    public static int sumar(int a, int b) {
        return a + b;
    }
}

public class Main {
    public static void main(String[] args) {
        int resultado = Calculadora.sumar(5, 7); // Llama al método estático sin crear una instancia
        System.out.println("Resultado: " + resultado); // Muestra: Resultado: 12
    }
}
```

Explicación: El método `sumar` es `static`, por lo que se puede llamar directamente desde la clase `Calculadora` sin crear un objeto.

Programación orientada a objetos Individual - POO - Unidad 2
Actividad: Taller sobre el modificador **static** en Java
Tutor: JOHN CARLOS ARRIETA ARRIETA DOCENTE

---

5. Ejemplos de Uso Incorrecto con Errores de Compilación

Ejemplo Incorrecto 1: Intentar Acceder a Atributos No Estáticos desde un Método `static`

```
public class Persona {
    private String nombre;

    public static void mostrarNombre() {
        System.out.println("Nombre: " + nombre); // Error de compilación: No se puede acceder a 'nombre' desde un contexto estático
    }
}
```

Explicación: Aquí, el método `mostrarNombre` es `static` e intenta acceder al atributo `nombre`, que no es `static`. Esto genera un error de compilación porque los métodos `static` solo pueden acceder a otros miembros `static`.

6. Ejemplos de Uso Incorrecto sin Generar Error de Compilación

Ejemplo Incorrecto 2: Declarar Métodos `static` que Modifican Datos de Instancia

```
public class Banco {
    public static double tasaInteres = 0.05; // Atributo estático para la tasa de interés

    public static void modificarTasa(double nuevaTasa) {
        tasaInteres = nuevaTasa; // Se modifica un atributo estático
    }

    public static void cambiarSaldo(double saldo) {
        // Intentar cambiar saldo no tiene sentido en un contexto estático
        saldo = saldo * (1 + tasaInteres); // No hay un atributo 'saldo' definido
    }
}
```

Explicación: Aunque el código compila, intentar modificar un atributo que no existe o realizar cálculos basados en datos de instancia en un método `static` es una mala práctica.

7. Ejercicios Propuestos

Ejercicio 1: Clase `Coche` con Contador de Instancias

1. Crea una clase `Coche` con los atributos `marca`, `modelo` y un atributo `static` llamado `contadorCoches`.
2. Define un constructor que incremente `contadorCoches` cada vez que se cree un objeto de la clase.
3. Crea un método `static` para mostrar cuántos coches se han creado.

Ejercicio 2: Clase `Matematicas` con Métodos `static`

1. Define una clase `Matematicas` que tenga métodos `static` para realizar las operaciones básicas (suma, resta, multiplicación, división).
2. Crea una clase de prueba que llame a estos métodos y muestre los resultados.

Ejercicio 3: Uso Incorrecto de `static`

1. Intenta crear un método `static` en una clase que intente acceder y modificar un atributo no estático. Observa los errores de compilación y explica por qué suceden.
2. Corrige el código para que sea correcto.

Conclusión del Taller:

Este taller ha proporcionado una base sólida sobre el modificador de acceso `static`, explicando su propósito, cuándo usarlo, y cómo definir miembros que pertenezcan a la clase en lugar de a las instancias. Además, se han discutido errores comunes y se han ofrecido ejemplos prácticos para reforzar el aprendizaje.