**Tutorial interactivo sobre Métodos y Variables Static en Java**

¡Bienvenido a este tutorial interactivo sobre **métodos y variables static en Java**! En este recorrido aprenderás a utilizar y comprender el concepto de miembros estáticos (static) en clases de Java. Este tipo de miembros es ampliamente utilizado en la programación orientada a objetos, y te ayudará a optimizar y organizar mejor tu código. A lo largo del tutorial, participarás con preguntas, tareas y ejercicios prácticos que te permitirán aplicar lo aprendido. ¡Comencemos!

**Paso 1: ¿Qué significa static en Java?**

En Java, la palabra clave static indica que un miembro de una clase (una variable o un método) pertenece a la clase en lugar de a las instancias de la clase (objetos). Esto significa que puedes acceder a ellos sin crear un objeto de esa clase.

**Ejemplo:**

java

Copiar código

class Calculadora {

static int sumar(int a, int b) {

return a + b;

}

}

En este caso, el método sumar es **estático**, lo que significa que no necesitas crear un objeto Calculadora para usarlo.

**Pregunta 1:**

¿Qué ocurre si intentas llamar al método sumar() sin crear un objeto de la clase Calculadora?

**Responde:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Paso 2: Variables static**

Las variables declaradas con static pertenecen a la clase y no a una instancia específica. Es decir, hay una única copia de la variable que es compartida por todos los objetos de la clase.

**Ejemplo:**

java

Copiar código

class Contador {

static int cuenta = 0;

Contador() {

cuenta++;

}

}

Cada vez que creas una nueva instancia de Contador, la variable cuenta se incrementa, ya que es compartida por todas las instancias de la clase.

**Tarea 1:**

Escribe un programa en Java que defina una clase con una variable static. Crea varios objetos y muestra cómo se comparte la variable entre ellos.

**Paso 3: Métodos static**

Un método static pertenece a la clase en su conjunto, y no a instancias específicas. Esto significa que no puede acceder a variables de instancia (no estáticas) directamente.

**Ejemplo:**

java

Copiar código

class Utilidades {

static String convertirMayusculas(String texto) {

return texto.toUpperCase();

}

}

Puedes llamar a este método así:

java

Copiar código

Utilidades.convertirMayusculas("hola mundo");

**Pregunta 2:**

¿Es posible acceder a un método no estático desde un método estático? ¿Por qué?

**Responde:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Tarea 2:**

Crea una clase Matematica con varios métodos static que realicen operaciones matemáticas simples, como sumar, restar, multiplicar y dividir. Usa estos métodos sin crear un objeto.

**Paso 4: Variables y Métodos static en acción**

El uso de variables y métodos static puede ser muy útil en diversas situaciones, como cuando deseas mantener un contador global o cuando necesitas una función auxiliar que no dependa de un objeto específico.

**Ejemplo:**

java

Copiar código

class Banco {

static double tasaInteres = 3.5;

static void mostrarTasaInteres() {

System.out.println("La tasa de interés es: " + tasaInteres + "%");

}

}

Puedes acceder tanto a la variable tasaInteres como al método mostrarTasaInteres() sin crear una instancia de Banco:

java

Copiar código

Banco.mostrarTasaInteres();

**Pregunta 3:**

¿Qué ocurriría si cambiamos el valor de la variable tasaInteres en algún punto del programa?

**Responde:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Tarea 3:**

Crea una clase Juego que tenga una variable static para almacenar el número total de jugadores. Cada vez que se cree un nuevo jugador, esta variable debe incrementarse.

**Paso 5: Bloques static**

En Java, también es posible definir **bloques de código static**, que se ejecutan cuando se carga la clase, es decir, antes de que se cree ningún objeto de la clase. Se utilizan comúnmente para inicializar variables static.

**Ejemplo:**

java

Copiar código

class Ejemplo {

static int x;

static {

x = 10; // Inicializa x antes de que se cree ningún objeto

System.out.println("Bloque estático ejecutado.");

}

}

El bloque static se ejecuta una vez, cuando se carga la clase.

**Pregunta 4:**

¿Cuándo se ejecuta un bloque static?

**Responde:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Tarea 4:**

Escribe un programa que use un bloque static para inicializar una variable static con un valor predefinido y muestre un mensaje cuando se cargue la clase.

**Paso 6: Acceso y restricciones de static**

Los métodos y variables static tienen las siguientes características:

1. **Acceso:** Se puede acceder a ellos usando el nombre de la clase, sin necesidad de crear una instancia.
2. **Restricción:** No pueden acceder directamente a métodos o variables no estáticas. Esto se debe a que los miembros static existen sin depender de una instancia específica, mientras que los no estáticos requieren una instancia.

**Ejemplo:**

java

Copiar código

class Persona {

String nombre;

static String especie = "Humano";

static void mostrarEspecie() {

System.out.println(especie);

}

void mostrarNombre() {

System.out.println(nombre); // Necesita una instancia para acceder a nombre

}

}

En este ejemplo, especie es accesible desde un método static, pero nombre no lo es, ya que pertenece a una instancia específica.

**Pregunta 5:**

¿Por qué los métodos static no pueden acceder a miembros no estáticos directamente?

**Responde:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Tarea 5:**

Crea una clase Empresa con una variable static que almacene el nombre de la compañía. Agrega un método static que imprima este nombre y un método no estático que imprima el nombre del empleado de la compañía.

**Paso 7: Aplicación práctica**

**Ejercicio final:**

1. Crea una clase ContadorPersonas con una variable static que cuente cuántas personas se han creado. Cada vez que crees una nueva instancia de ContadorPersonas, este número debe incrementarse. Además, define un método static que imprima cuántas personas han sido creadas.
2. Crea varias instancias de la clase y verifica cómo el contador aumenta en cada creación.

**Conclusión:**

¡Felicidades! Has aprendido cómo funcionan las variables y métodos static en Java. Recuerda que estos miembros pertenecen a la clase y no a las instancias individuales, lo que los hace útiles para valores compartidos y métodos que no necesitan modificar el estado de un objeto específico.

**Respuestas a las Preguntas:**

1. Puedes llamar al método sumar() sin crear un objeto de la clase Calculadora, ya que es un método static y pertenece a la clase, no a sus instancias.
2. No, no es posible. Un método static no tiene acceso a miembros no estáticos porque los métodos static pertenecen a la clase en sí, y los miembros no estáticos pertenecen a instancias específicas.
3. Si cambias el valor de tasaInteres, el nuevo valor será visible para todas las instancias y llamadas que utilicen la clase, ya que es una variable static compartida por todas.
4. Un bloque static se ejecuta una vez, cuando la clase es cargada por primera vez, antes de que se cree cualquier instancia de la clase.
5. Los métodos static no pueden acceder a miembros no estáticos directamente porque estos miembros pertenecen a instancias específicas y no existen cuando se accede a través de la clase (sin instancias).