Tipos de Variables y Referencias de memoria

José Moris y Paolo Ferreira

18-03

**Introducción**

En este informe, exploraremos las variables locales y globales, también se hablará de las referencias de memoria. Estos conceptos son fundamentales en la programación y tienen diversas ventajas, diferencias y usos. Comenzaremos explicando los conceptos más simples y avanzaremos hacia los más complejos, brindando una comprensión clara de cada uno y su propósito.

Variables locales

Las variables locales son variables de tipo privadas usualmente puestas para un método, la cual tiene un alcance limitado, no puede ser usada desde fuera de la propia clase o bloque de Código y estas son borradas/reiniciadas, esto se hace para no generar problemas en las variables y mantener el código organizado.

Las variables locales son accesibles desde su declaraciónhasta el final del bloque de Código.

En el caso de java, un bloque de Código viene a ser determinado por las llaves

“{ }” .

Texto

Descripción generada automáticamente

En este primero ejemplo, se da a comprender como funciona una variable local. Como se puede ver en el metodo “operacionMatematica” se declara una variable llamada numero la cual contiene el valor entero 5. Esta misma opera dentro del método y solo dentro de este

Texto

Descripción generada automáticamenteEn el momento de intentar llamar a la variable desde una ubicación diferente del método, el propio compilador nos advierte que la variable no se puede encontrar, esto se debe a que la propia variable se “destruye” cuando el método no es llamado y por ende no existe mientras el método no sea llamado.

Variables globales

Las variables globales al contrario de las locales estas se declaran fuera de los métodos en una clase y pueden ser accedidas desde cualquier ubicacion del programa, aunque no es recomendable usar demasiadas variables de este estilo ya que puede generar choques entre variables además de hacerlo menos legible, especialmente en programas grandes.

El uso de variables globales es bastante desaconsejable, ya que estas faltan a un pilar importante de la programación orientada a objetos, el encapsulamiento de datos. Esto sucede ya que las variables globales pueden ser accedidas sin restricción alguna y generalmente no pueden ser controladas por métodos de acceso como lo serían las variables locales.

Ejemplo de variable global:

Texto

Descripción generada automáticamente

Como se puede apreciar, se creó una variable global llamada contador la cual equivale a 0, se puede saber que esta es una variable global ya que esta puede ser llamada desde cualquier método pudiendo ser modificada desde cualquier parte del código.

Texto, Carta

Descripción generada automáticamenteAquí hay otro ejemplo, donde se muestra como una variable global, puede ser modificada desde cualquier parte del código.

Referencias de memoria

Las referencias de memoria son direcciones que apuntan a una ubicación física o virtual de los datos en la memoria. Estas referencias permiten a los programas acceder, modificar y manipular los datos almacenados en la memoria del dispositivo, estas son fundamentales en algunos lenguajes de programación C, C++, Python, Java, entre otros, ya que utilizan la memoria para almacenar datos y trabajar de manera ordenada.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteTexto

Descripción generada automáticamentePor ejemplo:

Texto

Descripción generada automáticamente

Como se ve en este ejemplo, se crea una clase Persona con el atributo nombre y edad, en el método “main” se crean dos objetos “persona1” y “persona2”.

En este código, “persona1” y “persona2” son referencias de memoria que apuntan a los objetos de la clase Persona que están almacenados en la memoria. Al cambiar el nombre de “persona2”, no afecta a “persona1” porque son dos objetos diferentes con sus propias referencias de memoria. Esto ilustra cómo las referencias de memoria permiten acceder y manipular objetos que apunten a la misma clase.

Texto

Descripción generada automáticamenteOtro ejemplo:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En este ejemplo, creamos un arreglo de números enteros llamado “numeros”. Los elementos del arreglo son almacenados en la memoria y la variable “numeros” es una referencia que apunta a este arreglo en memoria.

Luego, imprimimos los elementos del arreglo usando un bucle “for”. Posteriormente, cambiamos el valor del primer elemento del arreglo y mostramos nuevamente todos los elementos del arreglo.

Aquí, “numeros” actúa como una referencia de memoria que nos permite acceder y manipular los elementos del arreglo en la memoria.

**Conclusión**

En resumen, este informe ha destacado las diferencias fundamentales entre las variables locales y globales, así como la importancia de comprender las referencias de memoria en el contexto de la programación. Al explorar estos conceptos, hemos podido apreciar cómo afectan la estructura y el comportamiento de un programa, así como su impacto en la organización y mantenimiento del código.

Es crucial reconocer que el uso adecuado de variables locales y globales puede influir significativamente en la legibilidad, la eficiencia y la seguridad de un programa. Las variables locales ofrecen un alcance limitado y promueven la encapsulación de datos al restringir su accesibilidad a un método o bloque de código específico. Por otro lado, las variables globales, aunque ofrecen acceso desde cualquier parte del programa, pueden generar problemas de legibilidad y mantenimiento, así como violar los principios de encapsulación.

Además, las referencias de memoria desempeñan un papel crucial en la gestión de datos y objetos en la memoria del dispositivo. Comprender cómo funcionan las referencias de memoria es esencial para evitar errores, así como para garantizar un comportamiento coherente y predecible del programa.