Logotipo, nombre de la empresa

Descripción generada automáticamenteUniversidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Primer Semestre 2023

LAB. INTROD. A LA PROGRAMACION Y COMPUTACION 1

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | | Registro Académico | |
| Javier Andrés Monjes Solórzano | | 202100081 | |
| Actividad | **Correlativo** | | **Fecha** |
| Tarea No.1 | 1 | | 07/ 02 / 2023 |

Sección E

En cualquier lenguaje de programación, el paso de parámetros es vital para modular el código fuente. El paso de parámetros permite capturar valores o introducirlos directamente en el código fuente en forma de parámetros.

¿Qué es el Paso de Parámetros en Java?

Java siempre usa el método de paso de valor para pasar parámetros o argumentos. Cada vez que se llama a un método, Java crea una copia de cada parámetro y se la pasa al método. Si el parámetro pasado es un tipo de datos primitivo, este parámetro copia el valor en la pila. Luego lo pasa al método llamado. Sin embargo, si el parámetro pasado es un tipo de datos no primitivo, como un objeto, apunta a una referencia o dirección en la pila de memoria donde residen los datos reales. En Java, todos los tipos de datos no primitivos suelen ser referencias con la dirección de datos real. Por lo tanto, pasar un objeto como parámetro sigue siendo técnicamente pasar un valor, pero el valor pasado es una referencia.

**Paso por valor en Java**

Pasar por valor pasa una copia de la variable como argumento a la función. Con este método, el valor de cada variable de llamada se duplica. También se copia en una nueva variable correspondiente a la variable que recibió la "llamada". Cada vez que se cambia la variable principal, no afecta a la variable llamada. Esto se debe a que el "receptor" tiene una copia del parámetro original, no el parámetro en sí.

**Paso por referencia en Java**

En la mayoría de los lenguajes de programación y teoría de compiladores, se pasa una referencia por referencia al parámetro llamado. En otras palabras, los receptores de "llamada" y "llamador" hacen referencia y utilizan las mismas variables de parámetro. Cuando esta variable de referencia cambia, todas las funciones que hacen referencia a esa variable también cambian.

**Paso por referencia vs Paso por valor**

El paso por valor y el paso por referencia describen cómo un lenguaje pasa las variables a las funciones.

En el paso por valor, el valor de un parámetro de la función se copia en otra variable. Por el contrario, en el paso por referencia, se pasan/copian los parámetros reales a la función.

A menudo, los requisitos de seguridad del código dictan si se debe usar el paso por referencia o el paso por valor.

|  |  |
| --- | --- |
| Pase por referencia | Paso por valor |
| * Pasa la referencia original o los parámetros reales a la función | * Copia la variable de función en otra variable ficticia |
| * Los cambios realizados dentro de la función afectan a las variables originales | * Los cambios realizados dentro de la función no actualizan las variables originales |
| * Requiere menos memoria | * Suele requerir más memoria |
| * Consume menos memoria y tiempo que el paso por valor | * Suele requerir más tiempo de compilación y ejecución |

Ejemplo:

Los lenguajes de programación generalmente tienen dos formas de pasar parámetros a métodos. Parámetros post-valor (hacer una copia de la variable) o parámetros post-referencia (dar una referencia a la variable original).

**Paso de parámetros por valor**

Lo primero que tenemos que ver es que para los datos primitivos en Java se realiza claramente una copia.

public void metodo(int p) {

p=3;

}

int p1=2;

metodo(p1);

System.out.println(p1); //p1 = 2

**Paso de parámetros «por referencia»: referencia de objetos**

Pero ahora pasemos al manejo de objetos como parámetros. Lo que sucede cuando trabajas con objetos en Java, las variables contienen referencias a objetos, por lo que cuando pasamos un objeto como parámetro, se crea una copia de la referencia. Entonces tenemos dos variables diferentes que apuntan al mismo objeto. Creamos una clase básica llamada MiClase:

public class MiClase {

public int valor;

}

Y ahora un método que modifica ese valor:

public static void metodo\_referencia (MiClase m) {

m.valor =3;

}

Veamos cómo se pasa por valor, aunque parece que hay una referencia:

MiClase m1 = new MiClase ();

m1. valor = 2;

System.out.println(m1. valor); // Devuelve 2

metodo\_referencia(m1);

System.out.println(m1. valor); // Devuelve 3

Texto

Descripción generada automáticamente