PROGRA❤AMANDO

holamundo.co

Paso por referencia

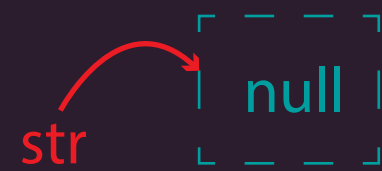
Es cuando se envía la dirección en memoria de una variable a una función por medio de sus parámetros.

Se debe tener cuidado ya que si se modifica el valor de la variable, el cambio se ve reflejado en el contexto desde donde fue llamada la función.

Valor null

Cuando se declara un tipo de dato que puede ser una referencia, es posible asignarle un valor que representa “nada”, y con este valor es posible empezar a usar la variable e operaciones.

```
StringBuilder str = null;
```



Recuerde que el valor null es temporal, para realizar otras operaciones es necesario inicializar la variable para evitar una excepción de NullPointerException.

NullPointerException

Se presenta cuando se intenta acceder a una función de un tipo de dato referencia, cuyo valor es NULL. Para que no se produzca la excepción se debe garantizar que la variable ha sido inicializada, la mayoría de los casos esto se hace por medio de la palabra reservada new y/o el operador de asignación “=” .

```
StringBuilder str = null;
str = new StringBuilder(”Prueba”);

String str2 = “Prueba 2”;
```

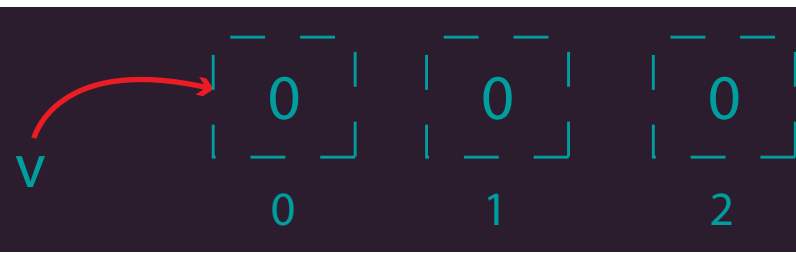
Arrays

Este tipo de dato puede ser una combinación de dos tipos diferentes: datos primitivos para las componentes y el contenedor en si, es de tipo referencia. Es posible asignar un valor de NULL a su variable.

Una vez creado el array usando new, cada posición del arreglo toma un valor por defecto que depende del tipo de dato.

```
float v[];
```

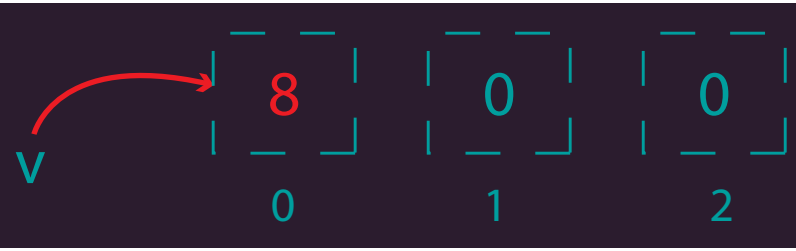

```
v = new float[3];
```



Componentes de un array

Para inicializar cada componente del array es necesario asignar un dato del mismo tipo del array, se puede hacer un array de datos tipo referencia o de un array tipo primitivo.

v[0] = 8;



v[1] = -2;

v[2] = 30;

Array de referencias

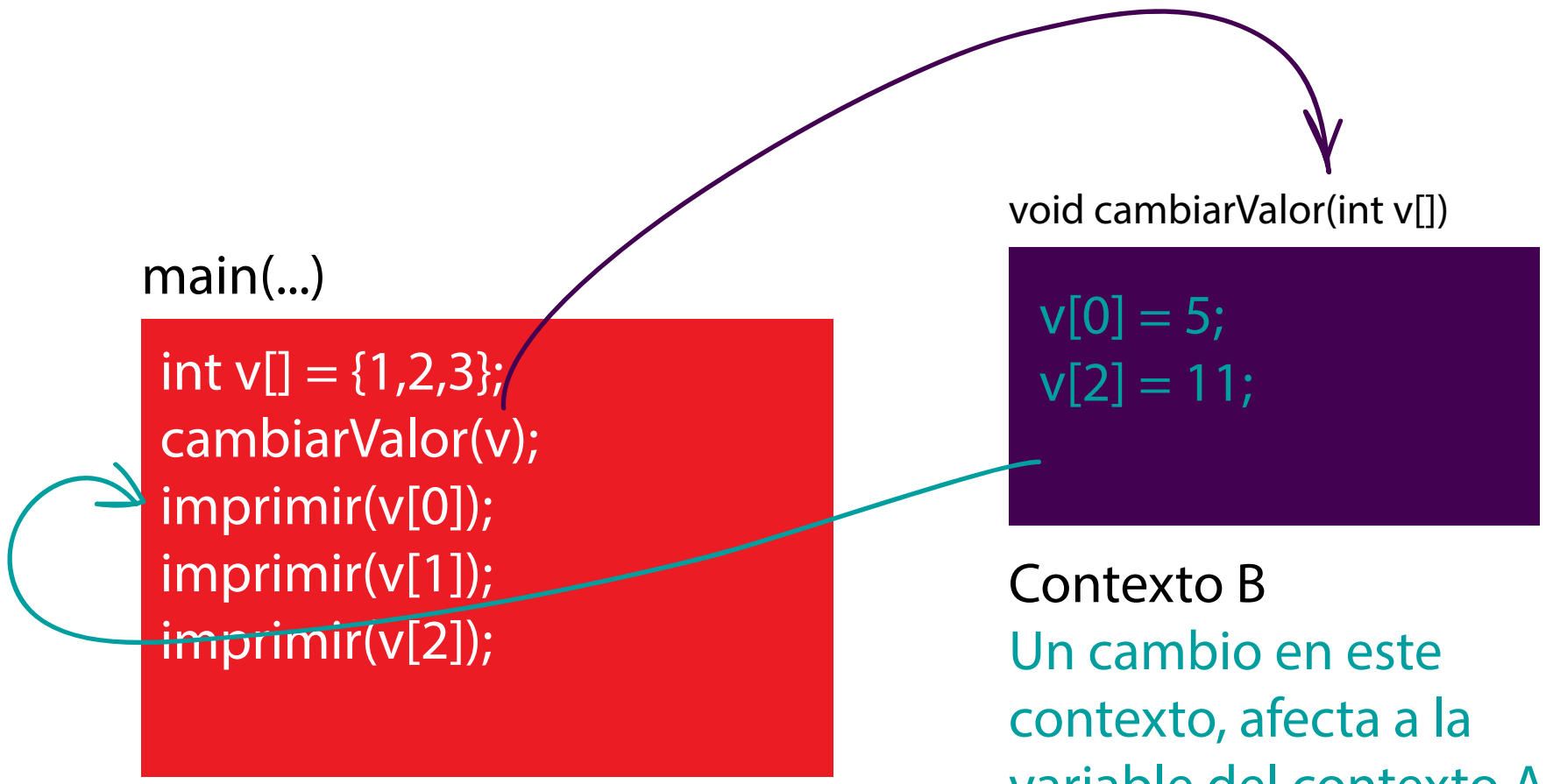
Se usa el operador new para inicializar el array y luego se usa nuevamente para inicializar cada componente.

```
StringBuilder str[] = new StringBuilder[2];
```

```
str[0] = new StringBuilder("Hola");
```

```
str[1] = new StringBuilder("Mundo");
```

Paso por referencia



Contexto A

V[0]:5, V[1]:2,V[2]:11 Se cambia la variable del contexto A, porque se pasa por referencia

Contexto B

Un cambio en este contexto, afecta a la variable del contexto A, porque hay paso por referencia

Función void con retorno

Si en la firma de una función se define el valor de retorno void, quiere decir que no se usa la palabra reservada return y por tanto no se retorna un dato al exterior de la función.

Sin embargo hay una forma de retornar un valor, de manera implícita en el parámetro de la función. NO SE CONSIDERA UNA BUENA PRÁCTICA, PERO ES ALGO QUE DEBE SABER ;)

Hay algunas referencias que se pasan como una copia del valor, por tanto es mejor usar el tipo de retorno explícito, para cambiar una variable por fuera del contexto de la función.

Función void con retorno

Declarar explícitamente el retorno de una función es una buena práctica, pues es más sencillo realizar debug en caso que la función presente un error, si el retorno es implícito, puede ser más complicado realizar pruebas.

Recuerde además que el propósito de una función, es hacer bien solo una cosa, ser especializada.

Función void con retorno

```
public static void retornarValor(char c[]){
        c[0] = -2;
}
public static void main(String[] args) {
        char c[] = new char[]{3,5,6};
        retornarValor(c);
        System.out.println(Arrays.toString(c));
}
```

La función retornarValor por usar un dato que se pasa por referencia, puede afectar el contenido de una función externa, como se pudo observar en el ejemplo anterior.

Recolección de basura

Java dispone de un mécanismo por defecto encargado de optimizar el uso de la memoría, llamado Garbage Collector (GC). Se encarga de liberar espacios de memoría que no se usan.

```
boolean v[] = new bolean(){true, false, true};
v = null; //El GC dispone de la memoria usada
```

Al asignar a una referencia el valor nulo, hace que el contenido que hacía referencia la variable, sea "reciclado" por el GC.

```
v = new bolean(){true, false, true}; //Se reasigna el valor
```

Errores frecuentes

Errores en tiempo de compilacón al tratar de realizar operaciones con variables inicializadas en NULL, estos errores no son detectables en tiempo de compilación.

No usar estructuras try-catch-finally, para controlar errores en tiempo de ejecución.

Utilizar el retorno implícito para más de una variable, ya que si se excede en esta práctica, puede perder el control del programa.

Usar valores primitivos para realizar un operación por referencia, el programa compila, pero el resultado no es el esperado.

La mayoría de variables tipo referencia, se pasan por copia.

Ejercicios

Realice una función que:

Lea cada una de las componentes de un array de números enteros, que se pasa por referencia.

Cree referencias del tipo StringBuilder a partir de una variable vacía que le llega como referencia. ¿Qué pasa si se imprime el valor antes y después de esta operación?

Reciba un StringBuilder y borre las vocales.

Retorne una referencia a un array de n números enteros aleatorios teniendo en cuenta un parámetro que llega como copia.