+ sobre "funciones"

UNRN
Universidad Nacional
de Río Negro

VI
2024

r15

Lo visto en estos días

El resumen de lo visto en estos días

Por Dredd

1

5

Una llamada a función en cada caso de prueba*

*a no ser que estén fuertemente conectadas (`DivisionLenta`)


¿Preguntas?

Detalles sobre

Pasaje de argumentos

Por un lado tenemos

```
public static int suma(int n, int m){
    n = n + m;
    return n;
}
```

¿Cómo quedan las variables en ‘test’?

```
public static int suma(int n, int m){
    n = n + m;
    return n;
}

public static void test(){
    int a = 10;
    int b = 20;
    int resultado = suma(a, b);
    System.out.println(resultado);
}
```

En Java, todo es por copia

Hasta acá, sin sorpresas

Y ahora

La suma de elementos de un arreglo

Suma de los elementos del arreglo

```
public static int suma(int[] arreglo){
    for(int i = arreglo.length-1; i>0; i--){
        arreglo[0] = arreglo[0] + arreglo[i];
    }
    return arreglo[0];
}
```

¿Es legal eso?

¿Se obtiene el resultado?

```
public static int suma(int[] arreglo){
    for(int i = arreglo.length-1; i>0; i--){
        arreglo[0] += arreglo[i];
    }
    return arreglo[0];
}

public static void test(){
    int[] areglo = {1,2,3,4}; // la suma debiera dar 10
    int resultado = suma(areglo);
    System.out.println(resultado);
}
```


¿Preguntas?

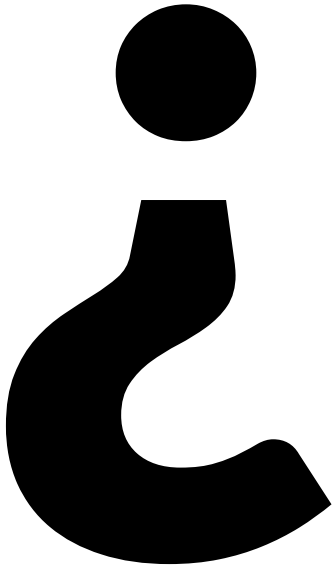
¿Cuales son los efectos secundarios?

Suma de los elementos del arreglo

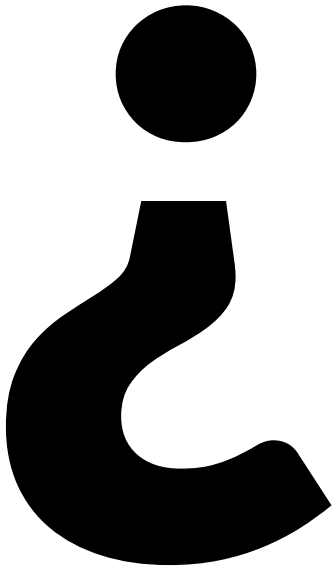
```
public static int suma(int[] arreglo){
    for(int i = arreglo.length-1; i>0; i--){
        arreglo[0] += arreglo[i];
    }
    return arreglo[0];
}
```

Suma de los elementos del arreglo

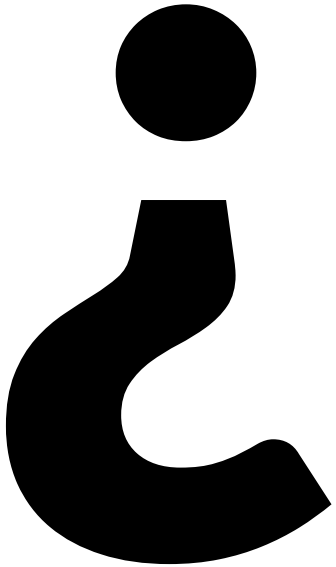
```
/**
 * Devuelve la suma de los elementos del arreglo
 * @param arreglo contiendo los valores que deseamos sumar
 * @return la suma de los elementos del arreglo
 * #PRE: el arreglo debe ser válido y contener por lo menos un valor
 * #POST: el arreglo debe quedar exactamente igual a como entró
 */
public static int suma(int[] arreglo){
    for(int i = arreglo.length-1; i>0; i--){
        arreglo[0] += arreglo[i];
    }
    return arreglo[0];
}
```



por que sucede esto



punteros



punteros

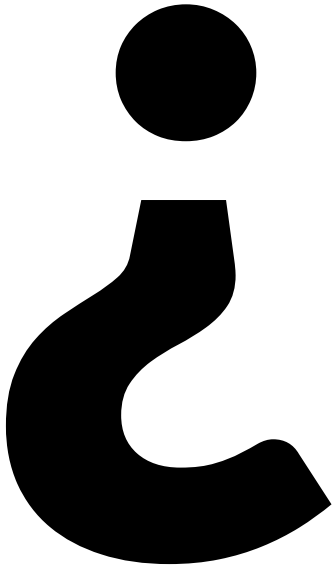

¿ punteros ?


NOPE!
Chuck Testa

tenemos

referencias

Que son parecidos pero no iguales



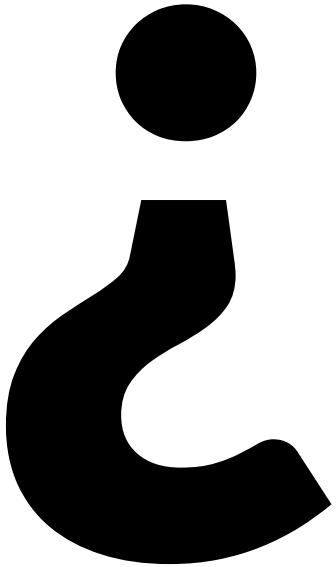
Qué operaciones definen a los punteros

en punteros

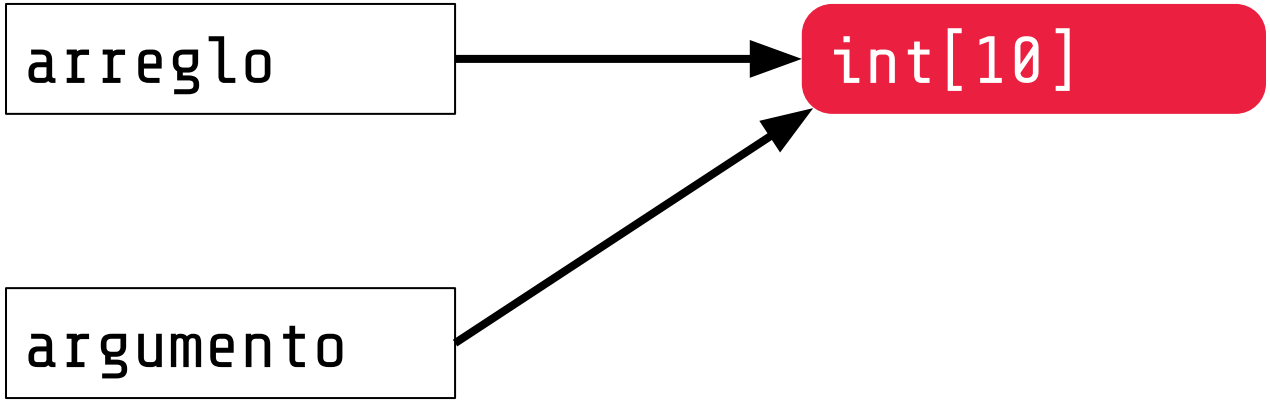
aritmética

en referencias

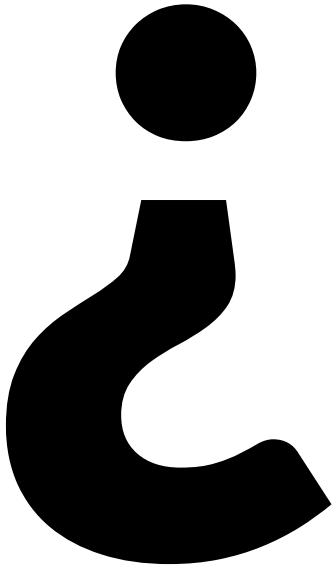
~~aritmética~~



Que quiere decir esto



Ambos van a parar al mismo arreglo



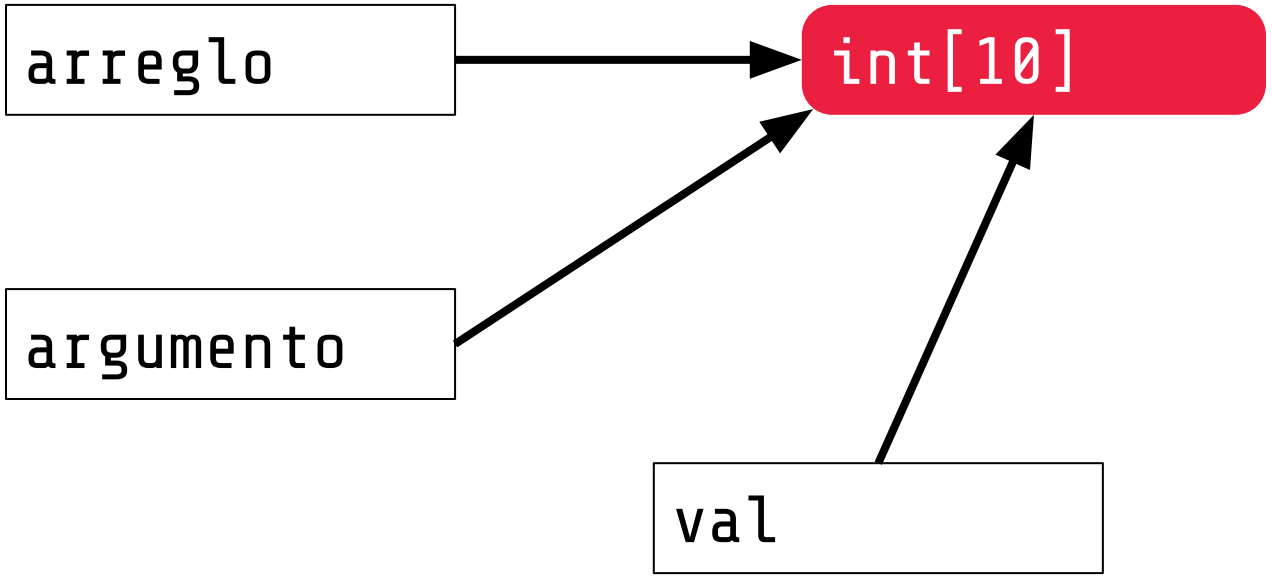
Como lo podemos evitar

Y ahora, ¿cómo queda el arreglo fuera?

```
public static int suma(int[] arreglo){
    int[] val = arreglo;
    for(int i = val.length - 1; i>0; i--){
        val[0] += val[i];
    }
    return val[0];
}
```


¿ Funciona ?

nope



Los tres van a parar al mismo arreglo

Solo creamos una nueva referencia al arreglo


¿Preguntas?

¿Y entonces?
¿Qué tenemos que hacer?

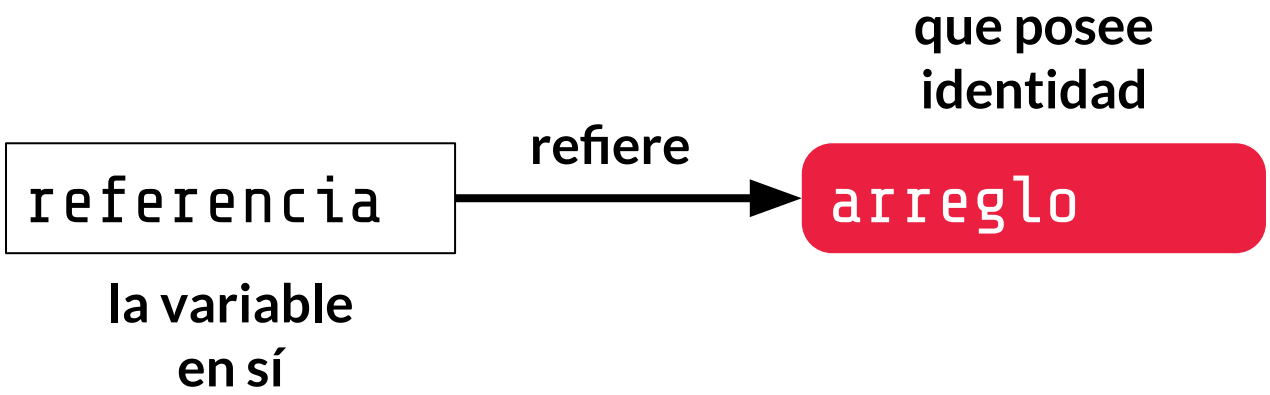
Copiar los elementos en un arreglo nuevo

```
int[] arr2 = new int[arr.length]
for (int i = 0; i < arr.length; i++){
    arr2[i] = arr[i]
}
```

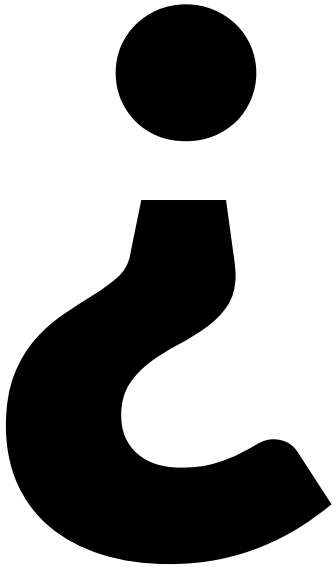
O usar
`Arrays.copyOf`


¿Preguntas?

En sintesis


que posee
identidad
refiere
referencia
arreglo
la variable
en sí


¿Preguntas?



Y String

La tercera vía

```
public static String concatenar(String dst, String src){
    dst = dst + src;
    return dst;
}
```

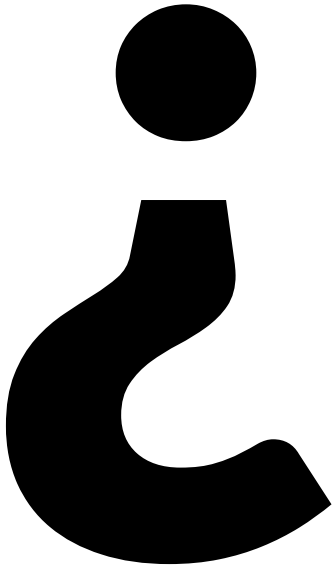
La tercera vía

```
public static String concatenar(String dst, String src){
    dst = dst + src;
    return dst;
}

public static void testCadenas(){
    String uno = "CADENA";
    String dos = "cadena";
    String resultado = concatenar(uno, dos);
    System.out.println(resultado);
}
```

Los String son inmutables

La identidad de cada cadena es diferente



cuando tenemos que tener en cuenta esto

cuando cambien valores que no deberían de cambiar


¿Preguntas?

Solo nos queda

La referencia no válida y su valor por defecto

¿Que contiene arreglo?

```
int[] arreglo;
for(int i = 0; i < arreglo.length; i++){
    System.out.print(arreglo[i]);
}
```

¡No hay nada!

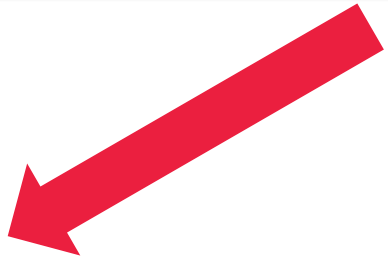
¿Que contiene arreglo?


```
int[] arreglo = null;
for(int i = 0; i < arreglo.length; i++){
    System.out.print(arreglo[i]);
}
```

Nos vamos a encontrar con


```
java.lang.NullPointerException
```

¿Y ahora?



```
int[] arreglo = new int[10];
for(int i = 0; i < arreglo.length; i++){
    System.out.print(arreglo[i]);
}
```

En donde debiera estar un new, hay una referencia

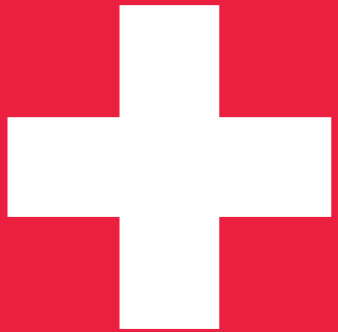

¿Preguntas?

1

6

Usar argumentos como variables solo si no cambia su significado

Un plus



Más números

¡No primitivos!

`java.math.BigInteger`

`java.math.BigDecimal`

¡Precisión arbitraria!

pero...

No funcionan los operadores

Para usar correctamente esto falta

```
import java.math.BigInteger;

// class y función

BigInteger uno = new BigInteger("100000000");
BigInteger dos = uno.pow(1000);
BigInteger tres = uno + dos;
BigInteger tres = uno.add(dos);
```

Necesario para lo que está fuera de java.lang

No funcionan los operadores


¿Preguntas?

unrn.edu.ar

UNRN Universidad Nacional de Río Negro

| unrionegro