

**¿Cómo mantengo un resultado más allá de la ejecución de un método?**

Two short, horizontal gray bars are positioned below the main text, one on the left and one on the right.

# Problema

## Sistema de Estadísticas para un Equipo de Fútbol.

Imaginemos que para cada partido, se registran ciertas estadísticas de los jugadores. Estas estadísticas son: número de goles, número de asistencias, número de tarjetas amarillas y número de tarjetas rojas. Necesitamos calcular la "calificación" de un jugador en base a estas estadísticas.

$$\text{calificación} = (\text{goles} \times 5) + (\text{asistencias} \times 3) - (\text{tarjetasAmarillas} \times 2) - (\text{tarjetasRojas} \times 5)$$

# Solución v1

```
public class CalculadoraDeCalificaciones {  
    public void v1(int g, int a, int ta, int tr, int cali) {  
  
        cali = g * 5 + a * 3 - ta * 2 - tr * 5;  
    }  
  
    public static void main(String[] args) {  
        int goles = 3;  
        int asistencias = 2;  
        int t_amarrillas = 0;  
        int t_rojas = 1;  
        int calificacion = 0;  
        CalculadoraDeCalificaciones cc = new CalculadoraDeCalificaciones();  
        cc.v1(  
            goles,  
            asistencias,  
            t_amarrillas,  
            t_rojas,  
            calificacion);  
        System.out.println(calificacion);  
    }  
}
```

# Solución v1

```
public static void main(String[]args) {  
    int goles = 3;  
    int asistencias = 2;  
    int t_amarrillas = 0;  
    int t_rojas =1;  
    int calificacion = 0;  
    CalculadoraDeCalificaciones cc = new CalculadoraDeCalificaciones();  
    cc.v1( // <--- se pasan los valores a v1 por valor dados que son primitivos  
        goles,  
        asistencias,  
        t_amarrillas,  
        t_rojas,  
        calificacion);  
    System.out.println(calificacion);  
}
```

---

# Solución v1

```
public class CalculadoraDeCalificaciones {  
    public void v1(int g, int a, int ta, int tr, int cali) {  
  
        /*  
         * En la stack de la ejecución del método se copian los valores  
         * // Stack:  
         // +-----+  
         // | cali: 0 |  
         // +-----+  
         // | tr: 1 |  
         // +-----+  
         // | ta: 0 |  
         // +-----+  
         // | a: 2 |  
         // +-----+  
         // | g: 3 |  
         // +-----+  
         *  
         * */  
        cali = g * 5 + a * 3 - ta * 2 - tr * 5;  
  
    }  
}
```

# Solución v1

```
public class CalculadoraDeCalificaciones {  
    public void v1(int g, int a, int ta, int tr, int cali) {  
        cali = g * 5 + a * 3 - ta * 2 - tr * 5;  
        // Luego de la ejecución de la línea del cálculo se actualiza el valor de cali  
        // perteneciente a la Stack de v1:  
        // +-----+  
        // | cali: 16 |  
        // +-----+  
        // | tr: 1    |  
        // +-----+  
        // | ta: 0    |  
        // +-----+  
        // | a:  2    |  
        // +-----+  
        // | g:  3    |  
        // +-----+  
    }  
}
```


# Solución v1

```
public class CalculadoraDeCalificaciones {  
    public void v1(int g, int a, int ta, int tr, int cali) {  
        cali = g * 5 + a * 3 - ta * 2 - tr * 5;  
        /*  
        * Dado que no quedan más instrucciones en el método v1 se limpia la Stack completamente  
        * y se retorna el control de donde fue llamada, es decir en la línea 20  
        * // Stack:  
        // +-----+  
        // |           |  
        // +-----+  
        *  
        * */  
    }
```

```
13     public static void main(String[] args) {  
14         int goles = 3;  
15         int asistencias = 2;  
16         int t_amarrillas = 0;  
17         int t_rojas = 1;  
18         int calificacion = 0;  
19         CalculadoraDeCalificaciones cc = new CalculadoraDeCalificaciones();  
20         cc.v1( // <--- se pasan los valores a v1 por valor dados que son primitivos  
21             goles,  
22             asistencias,  
23             t_amarrillas,  
24             t_rojas,  
25             calificacion);  
26         System.out.println(calificacion);  
27     }
```

# Solución v1

```
public static void main(String[]args) {  
    int goles = 3;  
    int asistencias = 2;  
    int t_amarrillas = 0;  
    int t_rojas =1;  
    int calificacion = 0;  
    CalculadoraDeCalificaciones cc = new CalculadoraDeCalificaciones();  
    cc.v1( // <--- se pasan los valores a v1 por valor dados que son primitivos  
        goles,  
        asistencias,  
        t_amarrillas,  
        t_rojas,  
        calificacion);  
    // como se pasaron copias de los valores por ser primitivos las modificaciones  
    //fueron locales al método y no sobre calificacion  
    System.out.println(calificacion);  
}
```





# Solución v2 Integer

```
public static void main(String[]args) {  
    int goles = 3;  
    int asistencias = 2;  
    int t_amarrillas = 0;  
    int t_rojas =1;  
    Integer calificacion = 0; // Integer es un objeto de tipo Wrapper.  
    CalculadoraDeCalificaciones cc = new CalculadoraDeCalificaciones();  
    cc.v1(  
        goles, // <-pasa por valor por ser primitivo, es decir se copia el valor  
        asistencias, // <-idem anterior  
        t_amarrillas, // <-idem anterior  
        t_rojas, // <-idem anterior  
        calificacion // <-Se pasa una copia de la referencia del objeto.  
    );  
    System.out.println(calificacion);  
}
```

# Solución v2 Integer

```
public class CalculadoraDeCalificaciones {  
    public void v1(int g, int a, int ta, int tr, Integer cali) {  
        // Stack:  
        // +-----+  
        // | cali:0x11f | <- cali ahora es una ref  
        // +-----+  
        // | tr: 1 |  
        // +-----+  
        // | ta: 0 |  
        // +-----+  
        // | a: 2 |  
        // +-----+  
        // | g: 3 |  
        // +-----+  
        cali = g * 5 + a * 3 - ta * 2 - tr * 5;  
    }  
}
```

# Solución v2 Integer

```
public class CalculadoraDeCalificaciones {  
    public void v1(int g, int a, int ta, int tr, Integer cali) {  
  
        cali = g * 5 + a * 3 - ta * 2 - tr * 5;  
        // Luego de la ejecución del cálculo la Stack queda así:  
        // +-----+  
        // | cali:0x00e| <- cali ahora es otra ref a otro Integer  
        // +-----+  
        // | tr: 1      |  
        // +-----+  
        // | ta: 0      |  
        // +-----+  
        // | a:  2      |  
        // +-----+  
        // | g:  3      |  
        // +-----+  
    }  
}
```

# Solución v2 Integer

```
public class CalculadoraDeCalificaciones {
    public void v1(int g, int a, int ta, int tr, Integer cali) {
        cali = g * 5 + a * 3 - ta * 2 - tr * 5;
        // Finalmente se libera la Stack:
        // +-----+
        // |           |
        // +-----+
    }

    public static void main(String[]args) {
        int goles = 3;
        int asistencias = 2;
        int t_amarrillas = 0;
        int t_rojas = 1;
        Integer calificacion = 0; // Integer es un objeto de tipo Wrapper.
        CalculadoraDeCalificaciones cc = new CalculadoraDeCalificaciones();
        cc.v1(
            goles, // <-pasa por valor por ser primitivo, es decir se copia el valor
            asistencias, // <-idem anterior
            t_amarrillas, // <-idem anterior
            t_rojas, // <-idem anterior
            calificacion // <-Se pasa una copia de la referencia del objeto.
        );
        // dado que el método v1 pisa la ref del objeto con una asignación
        //calificacion sigue siendo 0
        System.out.println(calificacion);
    }
}
```



# Solución v3 Objeto Resultado

```
public static void main(String[]args) {  
    int goles = 3;  
    int asistencias = 2;  
    int t_amarrillas = 0;  
    int t_rojas =1;  
    Resultado calificacion = new Resultado();// Resultado es un objeto  
    CalculadoraDeCalificaciones cc = new CalculadoraDeCalificaciones();  
    cc.v1(  
        goles, // <-pasa por valor por ser primitivo, es decir se copia el valor  
        asistencias, // <-idem anterior  
        t_amarrillas, // <-idem anterior  
        t_rojas, // <-idem anterior  
        calificacion // <-Se pasa una copia de la referencia del objeto por ej 0xd1.  
    );  
  
    System.out.println(calificacion.getRes());  
}
```

# Solución v3 Objeto Resultado

```
public class CalculadoraDeCalificaciones {  
    public void v1(int g, int a, int ta, int tr, Resultado cali) {  
        // Stack:  
        // +-----+  
        // | cali:0xd1 |  
        // +-----+  
        // | tr: 1 |  
        // +-----+  
        // | ta: 0 |  
        // +-----+  
        // | a: 2 |  
        // +-----+  
        // | g: 3 |  
        // +-----+  
        cali.setRes(g * 5 + a * 3 - ta * 2 - tr * 5);  
    }  
}
```

# Solución v3 Objeto Resultado

```
public class CalculadoraDeCalificaciones {  
    public void v1(int g, int a, int ta, int tr, Resultado cali) {  
        cali.setRes(g * 5 + a * 3 - ta * 2 - tr * 5);  
        // Luego de ejecutar el método setRes la stack queda igual.  
        // Al utilizar "." sobre una ref, le estamos indicando a java que  
        // queremos acceder a las definiciones del objeto que se encuentra en la ref.  
        // En este caso es un set de un valor y modifica el estado del objeto  
        // +-----+  
        // | cali: 0xd1|  
        // +-----+  
        // | tr: 1    |  
        // +-----+  
        // | ta: 0    |  
        // +-----+  
        // | a:  2    |  
        // +-----+  
        // | g:  3    |  
        // +-----+  
    }  
}
```

# Solución v3 Objeto Resultado

```
public class CalculadoraDeCalificaciones {  
  
    public void v1(int g, int a, int ta, int tr, Resultado cali) {  
        cali.setRes(g * 5 + a * 3 - ta * 2 - tr * 5);  
        // Finalmente se libera la Stack:  
        // +-----+  
        // |           |  
        // +-----+  
    }  
  
    public static void main(String[]args) {  
        int goles = 3;  
        int asistencias = 2;  
        int t_amarrillas = 0;  
        int t_rojas =1;  
        Resultado calificacion = new Resultado();// Resultado es un objeto  
        CalculadoraDeCalificaciones cc = new CalculadoraDeCalificaciones();  
        cc.v1(  
            goles, // <-pasa por valor por ser primitivo, es decir se copia el valor  
            asistencias, // <-idem anterior  
            t_amarrillas, // <-idem anterior  
            t_rojas, // <-idem anterior  
            calificacion // <-Se pasa una copia de la referencia del objeto por ej 0xd1.  
        );  
        // dado que el método v1 utiliza un método que cambia el estado del objeto  
        //calificacion es 16  
        System.out.println(calificacion.getRes());  
    }  
}
```





# Parámetros de métodos en JAVA

En JAVA cuando invocamos a métodos y le pasamos parámetros, el tipo de pasaje de parámetros que usamos se llama “Pasaje de parámetros por valor o por copia”: **el método invocado recibe una copia del parámetro enviado.**

## ¿Cómo funciona el pasaje de parámetros por valor en JAVA?

**-Si los parámetros son datos primitivos** (int, double, float, char, boolean) los cambios que se hagan del parámetro en el método invocado sólo se reflejan dentro del método y cuando el método retorna estos cambios se pierden.

**-Si los parámetros son objetos** los cambios que se hagan en los valores de los objetos dentro del método (sus variables de instancia) se mantendrán cuando el método retorne. En este caso, el parámetro del método contiene una copia de la “referencia” (puntero) del parámetro enviado. Ambos “referencias” apuntan al mismo objeto.

# Solución v4 return

```
public class CalculadoraDeCalificaciones {  
    public int v1(int g, int a, int ta, int tr) {  
        return g * 5 + a * 3 - ta * 2 - tr * 5;  
    }  
    public static void main(String[] args) {  
        int goles = 3;  
        int asistencias = 2;  
        int t_amarrillas = 0;  
        int t_rojas = 1;  
  
        CalculadoraDeCalificaciones cc = new CalculadoraDeCalificaciones();  
        int calificacion = cc.v1(  
            goles, // <-pasa por valor por ser primitivo, es decir se copia el valor  
            asistencias, // <-idem anterior  
            t_amarrillas, // <-idem anterior  
            t_rojas // <-idem anterior  
        );  
        System.out.println(calificacion);  
    }  
}
```



# Solución v5 variable de instancia

```
public class CalculadoraDeCalificaciones {  
    private int calificacion = 0;  
    public void v1(int g, int a, int ta, int tr) {  
        this.setCalificacion(g* 5 + a * 3 - ta * 2 - tr * 5);  
    }  
    public static void main(String[] args) {  
        int goles = 3;  
        int asistencias = 2;  
        int t_amarrillas = 0;  
        int t_rojas = 1;  
  
        CalculadoraDeCalificaciones cc = new CalculadoraDeCalificaciones();  
        cc.v1(  
            goles, // <-pasa por valor por ser primitivo, es decir se copia el valor  
            asistencias, // <-idem anterior  
            t_amarrillas, // <-idem anterior  
            t_rojas // <-idem anterior  
        );  
        //el uso de este tipo de solución tiene que tener sentido lógico en la implementación  
        //donde se declara la variable de instancia, al ser una calculadora lo tiene  
        System.out.println(cc.getCalificacion());  
    }  
}
```