Java Bitwise

Algoritmos y estructuras de datos (AyEdD)

Jesús Parrado Alameda

03 de Marzo de 2022

Operaciones de bitwise en Java.

URLs de Referencia

Bitwise and Bit Shift Operators

`https://docs.oracle.com/javase/tutorial/java/nutsandbolts/op3.html`.

Operador Bitwise en Java

El lenguaje de programación Java proporciona operadores que realizan operaciones bit a bit y de cambio de bit en tipos enteros.

Aunque este tipo de operadores se usan con menos frecuencia, es bueno saber que existen y cómo utilizarlos.

Por ejemplo, el siguiente programa, BitDemo, utiliza el operador AND bit a bit para imprimir el número 2 en la salida estándar.

```java
class BitDemo {
    public static void main(String[] args) {
        int bitmask = 0x000F;
        int val = 0x2222;
        // prints "2"
        System.out.println(val & bitmask);
    }
```

También se pueden combinar las operaciones con asignación de valor.

```java
public class bwop {
    public static void main(String[] args)
    {
        // Valores iniciales
        int a = 5;
        int b = 7;

        // también se puede combinar con asignaciones de valor
        // a=a&b
        a &= b;
        System.out.println("a= " + a);
```

```
12      }
13  }
```

Tipos de Operadores Bitwise

En la siguiente tabla podemos ver todos los tipos de operadores bit a bit que tenemos disponibles Java.

operador	símbolo	expresión
AND	&	op1 & op2
OR	|	op1 | op2
XOR	^	op1 ^ op2
Complemento	~	~ op1
Desplazamiento a izquierda	<<	op1 <<op2
Desplazamiento a derecha	>>	op1 >>op2
Desplazamiento a derecha sin signo	>>>	>>>op

Pero veamos como se utilizan uno a uno.

Bitwise AND (op1 & op2)

El operador & bitwise realiza una operación AND bit a bit.

```
public class bwop {
    public static void main(String[] args)
    {
        // Valores iniciales
        int a = 5;
        int b = 7;

        // bitwise and
        // 0101 & 0111=0101 = 5
        System.out.println("a&b = " + (a & b));

    }
}
```

Bitwise OR (op1 | op2)

El operador | realiza una operación OR inclusiva bit a bit.

```
public class bwop {
    public static void main(String[] args)
    {
        // Valores iniciales
        int a = 5;
        int b = 7;

        // bitwise or
        // 0101 | 0111=0111 = 7
        System.out.println("a|b = " + (a | b));

    }
}
```

Bitwise XOR (op1 ^ op2)

El operador ^ realiza una operación OR exclusiva bit a bit.

```
public class bwop {
    public static void main(String[] args)
    {
        // Valores iniciales
        int a = 5;
        int b = 7;

        // bitwise xor
```

```
        // 0 ^ 0 = 0
        // 1 ^ 0 = 1
        // 0 ^ 1 = 1
        // 1 ^ 1 = 0
        // 0101 ^ 0111=0010 = 2
        System.out.println("a^b = " + (a ^ b));

    }
}
```

Bitwise Complemento (~ op2)

El operador complemento invierte un patrón de bits; se puede aplicar a cualquiera de los tipos enteros, haciendo de cada 0 un 1 y de cada 1 un 0.

Por ejemplo, un byte contiene 8 bits; aplicar este operador a un valor cuyo patrón de bits sea 00000000 cambiaría su patrón a 11111111.

```
public class bwop {
    public static void main(String[] args)
    {
        // Valores iniciales
        int a = 5;
        int b = 7;

        // bitwise not
        // ~0101=1010
        // will give 2's complement of 1010 = -6
        System.out.println("~a = " + ~a);

    }
}
```

Bitwise Desplazamiento a izquierda (op1 <<op2)

El operador de desplazamiento a la izquierda con signo desplaza un patrón de bits a la izquierda.

El patrón de bits viene dado por el operando de la izquierda y el número de posiciones a desplazar por el operando de la derecha.

```
public class bwop {
    public static void main(String[] args)
    {
        // Valores iniciales
        int a = 5;
        int b = 7;
```

```
        // bitwise << left shift
        // 0101 <<2 10100
        System.out.println("a<<2");
        System.out.println(a<<2);

    }
}
```

Bitwise Desplazamiento a derecha (op1 >>op2)

El operador de desplazamiento a la derecha con signo desplaza un patrón de bits a la derecha.

El patrón de bits viene dado por el operando de la izquierda y el número de posiciones a desplazar por el operando de la derecha.

```
public class bwop {
    public static void main(String[] args)
    {
        // Valores iniciales
        int a = 5;
        int b = 7;

        // bitwise >> right shift
        // 0101 >>2 0001
        System.out.println("a>>2");
        System.out.println(a>>2);

    }
}
```

Bitwise Desplazamiento a derecha sin signo (>>>op)

El operador de desplazamiento a la derecha sin signo desplaza un cero a la posición más a la izquierda, mientras que la posición más a la izquierda depende de la extensión del signo.

```
public class bwop {
    public static void main(String[] args)
    {
        // Valores iniciales
        int a = 5;
        int b = 7;

        // bitwise >>> unsigned right shift
        // 0101 >>>2 0001
        System.out.println("a>>>2");
```

```
11          System.out.println(a>>>2);
12
13      }
14  }
```

bwop.java

Ejemplos de operaciones bit wise en java.

```java
public class bwop {
    public static void main(String[] args)
    {
        // Valores iniciales
        int a = 5;
        int b = 7;

        // bitwise and
        // 0101 & 0111=0101 = 5
        System.out.println("a&b = " + (a & b));

        // bitwise or
        // 0101 | 0111=0111 = 7
        System.out.println("a|b = " + (a | b));

        // bitwise xor
        // 0 ^ 0 = 0
        // 1 ^ 0 = 1
        // 0 ^ 1 = 1
        // 1 ^ 1 = 0
        // 0101 ^ 0111=0010 = 2
        System.out.println("a^b = " + (a ^ b));

        // bitwise not
        // ~0101=1010
        // will give 2's complement of 1010 = -6
        System.out.println("~a = " + ~a);

        // bitwise << left shift
        // 0101 <<2 10100
        System.out.println("a<<2");
        System.out.println(a<<2);

        // bitwise >> right shift
        // 0101 >>2 0001
        System.out.println("a>>2");
        System.out.println(a>>2);

        // bitwise >>> unsigned right shift
        // 0101 >>>2 0001
        System.out.println("a>>>2");
        System.out.println(a>>>2);

        // también se puede combinar con asignaciones de valor
        // a=a&b
```

```
        a &= b;
        System.out.println("a &= b ");
        System.out.println("a= " + a);


    }
}
```

Keep Calm and Learn Java Bitwise