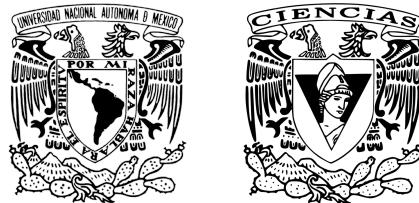


UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS



Práctica 02:  
**Tipos primitivos y bits**

*Pablo A. Trinidad Paz*

Trabajo presentado como parte del curso de **Introducción a Ciencias de la Computación**  
impartido por la profesora **Verónica Esther Arriola Ríos**.

27 de agosto de 2018

**Actividad 2.1** Revisa la documentación de las clases Byte, Short, Integer y Long de Java y revisa los atributos de clase que permiten acceder a esta información a) ¿Cuáles encuentras relevantes?

Intenta sumarle uno a `max`, imprime el resultado en base 10 y su representación en binario. b) Explica qué pasó.

- a) Todos los considero relevantes excepto por TYPE y BYTES porque creo que son muy redundantes. Respecto a BYTES, ya contamos con SIZE el cual menciona el número de bits y respecto a TYPE pues también siento muy redundante preguntar de qué clase viene el número si cuando lo inicializamos ya lo decimos explícitamente, por ejemplo:

```
double n = Double.MAX_VALUE;  
n.TYPE; // <--- j?
```

- b) El valor de `max` es  $2147483647_{10}$  en base 10 el cuál es equivalente a 31 1s precedidos por un 0 en base 2 ( $01111111111111111111111111111111_2$ ) donde el 0 indica que se trata de un entero positivo y al mismo tiempo se hace uso de los 32 bits. Al momento de sumar 1 al valor de `max`, la representación resultante en base 2 se vuelve  $10000000000000000000000000000000_2$  la cuál es interpretada dentro de Java como un entero negativo debido a que el primer dígito indica el signo, y a su vez, el valor de `max` es equivalente a  $-2147483648_{10}$  en base 10.

**Actividad 2.2** Utiliza la clase `ImpresoraBinario` para visualizar la representación interna de estos<sup>1</sup> valores especiales.

**Actividad 2.3** Imprime como se ve esto en binario ¿Cuánto vale permisos en base 10? (Sólo mándalo a imprimir Java tabaja por defecto en base 10)

## El código:

```
ImpresoraBinario printer = new ImpresoraBinario();
int permisos = Integer.parseInt("0754", 8);
printer.imprime(permisos);
System.out.println(permisos);
```

Genera la salida:

111101100  
492

---

<sup>1</sup>Referente a los valores descritos dentro de la descripción de la práctica.

**Actividad 2.4** Realiza pruebas con los operadores de corrimiento <<, >> y >>>. Recorre los números del inciso anterior por uno y tres bits.

El código:

```
System.out.println("\n\n===== Actividad 2.4 =====");

ImpresoraBinario printer = new ImpresoraBinario();
int permissions_b10 = 0754;

System.out.print("Original value: ");
printer.imprime(permissions_b10);

System.out.print("\n<< 1: ");
printer.imprime(permissions_b10 << 1);
System.out.print("<< 3: ");
printer.imprime(permissions_b10 << 3);

System.out.print("\n>> 1: ");
printer.imprime(permissions_b10 >> 1);
System.out.print(">> 3: ");
printer.imprime(permissions_b10 >> 3);

System.out.print("\n>>> 1: ");
printer.imprime(permissions_b10 >>> 1);
System.out.print(">>> 3: ");
printer.imprime(permissions_b10 >>> 3);
```

Genera la salida:

```
===== Actividad 2.4 =====
Original value: 111101100

<< 1: 1111011000
<< 3: 1111011000000

>> 1: 11110110
>> 3: 111101

>>> 1: 11110110
>>> 3: 111101
```

**Actividad 2.5** Ahora toma los permisos de la sección anterior: ¿Qué operaciones necesitas hacer para que todos los usuarios tengan permiso de escritura? Verifica tu respuesta imprimiendo las representaciones binarias.

Podemos hacer uso de la operación **or** y el valor 1 en la posición de los permisos de escritura (tercer carácter de cada grupo de usuarios) y usar el valor 0 para las demás posiciones y para no alterar los valores de otros permisos. Dicho de otra manera, le aplicamos la función **or** a cualquier permiso existente usando el valor  $010010010_2$  (222<sub>8</sub>).

Verificación:

```
System.out.println("\n\n===== Actividad 2.5 =====");

ImpresoraBinario printer = new ImpresoraBinario();
int original_permission = 0754;
int write_permissions = 0222;

System.out.print("Original value: ");
printer.imprime(permission);

System.out.print("Write permissions: ");
printer.imprime(write_permissions);

System.out.print("New permission (original OR write): ");
printer.imprime(permission | write_permissions);
```

Genera la salida:

```
===== Actividad 2.5 =====
Original value: 111101100
Write permissions: 10010010
New permission (original OR write): 111111110
```