**Ejercicio U1\_B6\_E1:**

Una mala solución, sin %

class Unidad1{

public static void main(String[] args){

int x=7;

if(x==1 || x==3 || x==5 || x==7 || x == 9){

System.out.println("x es impar");

}else{

System.out.println("x es par");

}

}

}

La solución anterior sólo funciona para los 10 primeros números. el operador % me permite escribir una solución que funciona para cualquier número entero.

class Unidad1{

public static void main(String[] args){

int x =3;

int resto=x%2;

if(resto==0){

System.out.println(x + " es par");

}else{

System.out.println(x + " es impar");

}

}

}

también se puede escribir la lógica contraria

class Unidad1{

public static void main(String[] args){

int x =3;

int resto=x%2;

if(resto!=0){

System.out.println(x + " es impar");

}else{

System.out.println(x + " es par");

}

}

}

Podemos prescindir de la variable resto

class Unidad1{

public static void main(String[] args){

int x =3;

if(x%2==0){

System.out.println(x + " es par");

}else{

System.out.println(x + " es impar");

}

}

}

La mejor forma de escribir es con la que te sientas más cómod@, a medida que uses java, la tendencia es a escribir el código con estilo compacto, pero no hay prisa ya que es algo que llega sólo, de forma natural.

**Ejercicio U1\_B6\_E2:**

class Unidad1{

public static void main(String[] args){

int x =13;

int y=2;

if(x%y==0){

System.out.println(x + " es multiplo de "+y);

}else{

System.out.println(x + " NO es multiplo de "+y);

}

}

}

**Ejercicio U1\_B6\_E3:**

class Unidad1{

public static void main(String[] args){

int x =15;

int y=200;

if(x\*x>100){

System.out.println("valor de x: "+ x);

System.out.println("valor de y: "+ y);

y++;

System.out.println("Nuevo valor de y: "+ y);

}else{

}

}

}

Cuando el else es vacío, se prefiere no escribir, lo más sencillo suele ser mejor:

class Unidad1{

public static void main(String[] args){

int x =15;

int y=100;

if(x\*x>100){

System.out.println("valor de x: "+ x);

System.out.println("valor de y: "+ y);

y++;

System.out.println("Nuevo valor de y: "+ y);

}

}

}

**Ejercicio U1\_B6\_E4:**

class Unidad1{

public static void main(String[] args){

int x=8;

System.out.println(x%2==0?x+" es par": x + " es impar");

}

}

podríamos escribirlo menos compacto

class Unidad1{

public static void main(String[] args){

int x=8;

String stringPar=x+" es par";

String stringImpar=x+" es impar";

System.out.println(x%2==0?stringPar:stringImpar);

}

}

y respecto al primer operando podríamos enriquecerlo o embarullarlo, según se mire, con paréntesis. Observa en todo caso que no hicieron falta paréntesis porque

% tiene mayor precedencia que == y este que ?:

**Ejercicio U1\_B6\_E5:**

class Unidad1{

public static void main(String[] args){

int x =16;

int y=3;

System.out.println(x + (x%y==0?" es multiplo de ":" NO es multiplo de ")+y);

}

}

**Ejercicio U1\_B6\_E6:**

class Unidad1{

public static void main(String[] args){

int x =15;

int y=100;

System.out.println(x\*x>100?"valor de x: "+ x + "\nvalor de y: "+ y +"\nNuevo valor de y: "+ ++y :"");

}

}

Este caso tiene menos sentido para el operador condicional ya que si una expresión es vacía de los tres operadores uno “sobra”. El if sería más simple y claro.

**Ejercicio U1\_B6\_E7:**

class Unidad1{

public static void main(String args[]){

System.out.println("4\*1="+ (1<<2));//1\*4

System.out.println("4\*2="+ (2<<2));//2\*4

System.out.println("4\*3="+ (3<<2));//3\*4

System.out.println("4\*4="+ (4<<2));

System.out.println("4\*5="+ (5<<2));

System.out.println("etc...");

}

}

**Ejercicio U1\_B6\_E8:**

class Unidad1{

public static void main(String[] args){

int x = 4;

int y =5;

System.out.println("x:"+ x +" y:"+ y);

System.out.println("Multiplicación lógica: "+ Integer.toBinaryString(x)+ " and " +Integer.toBinaryString(y)+" = "+Integer.toBinaryString(x&y));

System.out.println("Multiplicación aritmética: "+ Integer.toBinaryString(x)+ " \* " +Integer.toBinaryString(y)+" = "+Integer.toBinaryString(x\*y));

}

}

**Ejercicio U1\_B6\_E9:**

class Unidad1{

public static void main(String[] args){

int x=10;//1010 en binario

//int x=0B1010 Equivalente al anterior

//set a 1 del bit 3

int mascara=1<<2;

int x2=x|mascara;

System.out.println("x: "+Integer.toBinaryString(x));

System.out.println("Máscara: "+Integer.toBinaryString(mascara));

System.out.println("x2: "+Integer.toBinaryString(x2));

}

}

**Ejercicio U1\_B6\_E10:**

class Unidad1{

public static void main(String[] args){

//int numero= 0B00000000000000000000000001010101;

//es más cómodo en decimal

int numero=85;

//int colocadorCero= 0B1111111111111111111111111111011;

//más cómodo calculando

int colocadorCero=~(1<<2);

int resultado=numero&colocadorCero;

String resultadoEnStringBinario=Integer.toBinaryString(resultado);

System.out.println(resultadoEnStringBinario);

}

}

fijáte que si hubiera hecho

int colocadorCero= 0b11111011;

ya que es un int se almacena

int colocadorCero= 0B00000000000000000000000011111011;

y ciertas situaciones pueden producirse efectos no deseados

**Ejercicio U1\_B6\_E11:**

class Unidad1{

public static void main(String[] args){

int i=10;

System.out.println("Valor inicial de i: "+i);

i%=3;//i=i%3;

System.out.println("Valor de i tras i%=3: "+i);

i>>=1;

System.out.println("Valor de i tras i>>=1: "+i);

i|=1;

System.out.println("Valor de i tras i|=1: "+i);

}

}