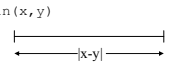
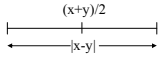
Problema
Escribir un método (función) que reciba dos enteros y entregue el mayor de los dos (sin usar Math.max). Ej: int m = mayor(a,b);

Solución




```
static public int mayor(int x, int y){
  return Math.min(x,y) + Math.abs(x-y);
}
```

Solución 2




```
static public int mayor(int x, int y){
  return (x + y + Math.abs(x-y))/2;
}
```

Solución 3 (más natural)

```
static public int mayor(int x, int y)
{
  if( x > y )
    return x;
  else
    return y;
}
```

¿Significado?

si x es mayor que y,
entonces entregar el valor de x,
si no, es decir si x es menor o igual que y, devolver el valor de y

Instrucción if-else

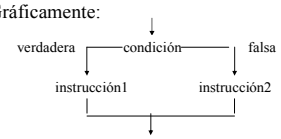
Sintaxis

```
if( condición )
   instrucción1;
else
   instrucción2;
```

Instrucción if-else

Semántica

- Si condición se cumple (es verdadera), ejecutar instrucción1
- Si condición no se cumple (es falsa), ejecutar instrucción2
- Gráficamente:



Condición (condición simple o comparación)

sintaxis: expresión1 operador-de-relación expresión2
operador de relación (comparación): <, >, <=, >=, ==, !=

semántica:
- evaluar expresiones
- comparar resultados de expresiones
- si condición se cumple entregar valor verdadero (true)
- si condición no se cumple entregar valor falso (false)

ejemplo

```
if( b*b >= 4*a*c )
    U.println("raices reales");
else
    U.println("raices complejas");
```

Caso especial 1: cada instrucción puede ser otro if-else

```
static public int mayor(int x,int y,int z)
{
   if( x >= y )

      if( x >= z )
        return x;
      else
        return z;

   else

      if( y >= z )
        return y;
      else
        return z;
}
```

Caso especial 2: else puede omitirse

```
static public int mayor(int x,int y,int z)
{
   int aux=x;

   if( y > aux) aux=y;

   if( z > aux) aux=z;

   return aux;
}
```

Caso especial 3:
bloque {...} para agrupar varias instrucciones

```
   …
   if( a>=b)
   {
      mayor=a;
      menor=b;
   }
   else
   {
      mayor=b;
      menor=a;
   }
```

Condiciones compuestas

```
static public int mayor(int x,int y,int z)
{
   if( x >= y  &&  x >= z )
     return x;
   else
     if( y >= z )
       return y;
     else
       return z;
}
```

Con indentación (uso de márgenes) que evidencia selección múltiple:

```
static public int mayor(int x,int y,int z)
{
  if( x >= y  &&  x >= z )
    return x;
  else if( y >= z )
    return y;
  else
    return z;
}
if(cond1)
  inst1
else if(cond2)
  inst2
…
else
  inst
```

Condición compuesta
sintaxis

- condición1 operador-lógico condición2 . . .
- operador lógico:
  &&: y, and, conjunción
  ||: o, or, disyunción
  !: no, not, negación (operador unario)

semántica

c1	c2	c1 && c2	c1 || c2	!c1
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V
		V si ambos V	V si alguno V	V si F

Nota. c2 se evalúa sólo si es necesario. Por ej
`if(x>=y && x>=z)`... si x<y entonces x>=z no se evalúa

prioridades de operadores
(orden de evaluación)

1	+ - ! (unarios)
2	(tipo) coerción
3	* / %
4	+ -
5	< > <= >=
6	== !=
7	&&
8	||

Problema. Escribir los métodos iguales y main

```
class Programa{
//iguales(x,y,z): cantidad de números iguales (3,2, o 0)
//ej:iguales(1,2,3)=0,iguales(1,2,1)=2,iguales(1,1,1)=3
static public int iguales(double x,double y,double z){
…
}
static public void main(String[]arg)throws IOException{
…
}
}
```

Diálogo del programa principal:

```
Tipo de triángulo de lados a,b,c
a? __
b? __
c? __
equilátero,isósceles,o,escaleno
```

```
int iguales(double x,double y,double z){
  if(x==y && x==z)
    return 3;
  else if(x==y || x==z || y==z)
    return 2;
  else
    return 0;
}
```

Solución 2. Con if sin else

```
int iguales(double x,double y,double z){
  if(x==y && x==z) return 3;
  if(x==y || x==z || y==z) return 2;
  return 0;
}
```

Solución 3. "negando" la 2ª condición

```
int iguales(double x,double y,double z){
  if(x==y && x==z) return 3;
  if(x!=y && x!=z && y!=z) return 0;
  return 2;
}
```

```
U.println("Tipo de triángulo de lados a,b,c");
double
  a=U.readDouble("a?"),
  b=U.readDouble("b?"),
  c=U.readDouble("c?");
int n=iguales(a,b,c);
if( n == 3 )
  U.println("equilátero");
else if( n == 2 )
  U.println("isósceles");
else
  U.println("escaleno");
```

Tipo boolean

constantes: **`true`** (verdadero) y **`false`** (falso)

variables: **`boolean`** nombre;

expresiones: condiciones

asignación: variable=condición;

ejemplos:

```
boolean p ;
p = a>=b && a>=c;
if( p ) //equivalencia: if( p==true )
   U.println("mayor="+a);
```

Funciones

ejemplo:

```
static public boolean par(int x){
  return x%2==0;
}
```

equivalencia:

```
static public boolean par(int x){
  if(x%2==0) return true; else return false;
}
```

uso

```
if(par(n))… ;else …;
```

sintaxis

```
static public boolean nombre(parámetros){
  instrucciones;
  return condición (exp de tipo boolean);
}
```