Diferencias entre Java y C++

En Java los punteros son expícitos

En C++ el uso de punteros es común para manejar memoria dinámica.

En Java no se programa con punteros, porque Java siempre utiliza punteros explícitos (direcciones de memoria).

En Java "todo son punteros" -más o menos-

Referencias

- Diferencias entre tipos primitivos y Objetos

```
int  var1;
MiClase objeto1;
```

En la dirección de memoria var1 se almacena un valor entero

La dirección de memoria objeto1 no apunta a los datos del objeto, sino que es una referencia al propio objeto (null si no se le ha asignado un objeto)

Referencias

En C++

```
MiClase objeto1;
```

crea el objeto y su memoria

"Una referencia es un puntero con la sintaxis de una variable"

Asignación

- En C++

ob1=ob2;

copia los datos de un objeto ob2 a otro objeto ob1.

En Java se copia la referencia

ob1.met();

ob2.met();

Aplica el método met() al mismo objeto

El operador new

- Todo objeto en Java se crea con new, que devuelve una referencia al objeto.
- En C++ new devuelve un puntero.

- En Java no podemos corromper unos datos accediendo a su dirección de memoria.

Borrado de la memoria

- En C++ se debe borrar con delete
- En Java no hay que preocuparse de borrar la memoria, pues el recolector de basuras (garbage collector) borra los objetos no referenciados automáticamente.

Argumentos

- En C++ se puede pasar un puntero como argumento de una función para no copiar los datos de un objeto.
- En Java todos los objetos se pasan por referencia (evitando copiar objetos) y los tipos primitivos se pasan por valor (se crea una variable y se copia el valor).

Operador ==

- En C++ el oreador

ob1==ob2 indica si los datos de los distintos objetos son iguales

- En Java el operador == indica si las referencias son iguales, es decir, si son el mismo objeto.

Sobrecarga de operadores

- En C++ se puede sobreescribir algunos operadores como +, * para que sea dsitinto en los objetos de alguna clase
- En Java no hay sobrecarga de operadores

Herencia Múltiple

- En C++ las clases soporta herencia simple y múltiple
- Las clases en Java sólo soportan herencia simple (aunque las interfaces si soportan herencia múltiple)