APRENDERAPROGRAMAR.COM

ASIGNACIÓN DE IGUALDAD EN JAVA. REFERENCIAS A OBJETOS. DIFERENCIA ENTRE IGUALDAD E IDENTIDAD. (CU00663B)

Sección: Cursos
Categoría: Curso “Aprender programación Java desde cero”

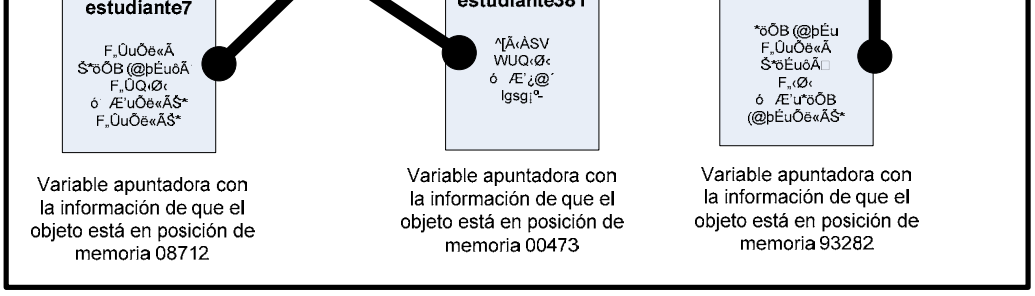
Fecha revisión: 2029

Resumen: Entrega nº63 curso Aprender programación Java desde cero.

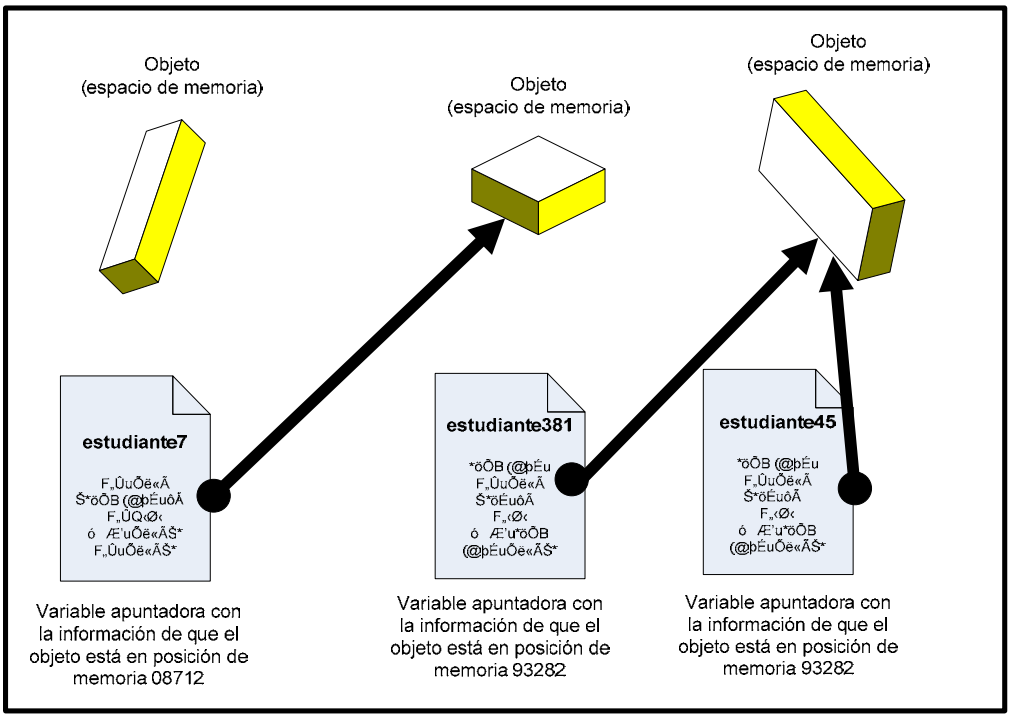
Autor: Alex Rodríguez

ASIGNACIÓN DE IGUALDAD CON TIPOS PRIMITIVOS Y OBJETOS.

Si como hemos explicado anteriormente los tipos objeto no se almacenan directamente sino mediante referencias al objeto, ¿Qué efecto tiene hacer una asignación de tipo = entre objetos? En primer lugar, hay que tener en cuenta que para establecer asignaciones de este tipo entre objetos debe haber coincidencia de tipos. Es decir, tratar de establecer *estudiante478 = deposito3;* nos llevará a un mensaje de error del compilador de tipo "*Incompatible types – found Deposito but expected Estudiante*".



Ahora supongamos que establecemos: estudiante381 = estudiante45. El significado de esta instrucción es que la variable apuntadora de estudiante381 se convierte en idéntica a la variable apuntadora estudiante45. Es decir, ahora ambas variables referencian al mismo objeto. Gráficamente:



Si nos fijamos en este último gráfico, comprobaremos una cosa: dos variables tienen la misma información (apuntan al mismo objeto) y un objeto ha dejado de estar apuntado, se ha quedado "flotando". ¿Cómo podemos acceder a ese objeto? La respuesta es que ya no podemos acceder. Este tipo de situaciones en principio no son deseables, ya que supone dejar "información" a la que no podemos acceder y esto no puede considerarse adecuado o razonable. Al hacer una asignación como estudiante381 = estudiante45 lo que hacemos es tener dos variables referenciadoras con la misma información, lo que nos lleva a que solo podamos acceder a un único objeto. Después de hecha la asignación ambas variables nos referencian al mismo objeto, podríamos decir que son redundantes: dos nombres para la misma cosa. Dos variables que apuntan al mismo objeto se dice que son "alias" de ese objeto.

¿Qué ocurre con los tipos primitivos? En los tipos primitivos, al no tener la configuración apuntador <-> contenido, podemos usar las asignaciones = sin ningún problema. Por ejemplo: int i = 3; int j = 5; i = j; supone que la variable i toma el valor de la variable j, es decir, i pasa a valer 5.

Dentro de los objetos tenemos el caso que ya hemos citado en algunas ocasiones que es un tanto especial de los tipo String. En este caso, sí es posible usar el operador de asignación = para transferir el contenido de unas variables a otras porque como ya hemos indicado, se trata de objetos con un comportamiento un tanto singular.

Volvamos a los tipos objeto. Si no podemos usar el operador = ¿Cómo podemos asignar a un objeto el mismo contenido que otro objeto pero manteniendo que ambas variables referencien a objetos independientes? Vamos a citar, sin entrar en profundidad, algunas maneras:

1) Partimos de un primer objeto con un contenido. Creamos otro segundo objeto usando la instrucción *new*. Seguidamente, establecemos el contenido de campos del segundo objeto usando los métodos set y asignando como contenidos los valores devueltos por los métodos get del primer objeto. Por ejemplo: *estudiante381.setNombre (estudiante45.getNombre());*.

2) Buscamos formas para clonar objetos en el API de Java. Una posible vía para crear una copia de un objeto en otro objeto sería usar el método *clone*. No vamos a explicar su uso ahora.

3) Definimos un constructor que nos permita pasar como parámetro un objeto y cuyo código tenga las instrucciones precisas para que el objeto que se cree tenga el mismo contenido que el objeto que se pasa como parámetro. La creación del objeto con el mismo contenido la lograríamos invocando ese constructor, por ejemplo: *estudiante 381 = new Estudiante (estudiante45);*

4) Otras vías.

A modo de conclusión hay que retener algo importante: no podemos usar el operador de asignación = entre objetos de la misma forma en que lo hacemos para tipos primitivos.

REPASO Y EJEMPLOS SOBRE IGUALDAD, IDENTIDAD Y MÉTODO EQUALS

Consideremos que un objeto ocupa un espacio de memoria. Ahora diremos que hay una o varias variables que apuntan a ese objeto. Supongamos la clase Persona y dos objetos creados como:

Persona p1 = new Persona("José Ramírez Mota", 32);
Persona p2 = new Persona("José Ramírez Mota", 32);

Ahora nos podemos plantear lo siguiente: ¿p1 == p2? Esta pregunta en Java se traduce así: ¿Es el objeto al que apunta p1 igual al objeto al que apunta p2? Es decir, ¿son el mismo objeto los objetos apuntados por p1 y p2, y por tanto existe una relación de identidad entre ambos? La respuesta es que no, no son el mismo objeto, pues cada vez que usamos el constructor, cada vez que usamos new, creamos un nuevo objeto. Seguramente no queríamos preguntarnos si las variables apuntadoras apuntaban al mismo objeto, sino si los objetos eran "iguales". Ahora bien, la igualdad entre tipos primitivos sí es evaluable con relativa facilidad porque no distinguimos entre identidad e igualdad, en tipos primitivos ambas cosas podemos decir que son lo mismo: p.ej. ¿Es 32 == 21 ó el carácter 'f' =='F'? Obviamente no. Pero **la igualdad entre objetos en Java se basa en que exista una definición previa de cuál es el criterio de igualdad,** y esta definición es la que dé el método equals aplicable al objeto (todo objeto en principio tendrá un método equals). ¿Qué ocurre si hacemos p1 = p2? En este caso un objeto deja de estar apuntado por variable alguna. Las dos variables pasan a apuntar al mismo objeto y por tanto p1 == p2 devuelve true.

Recordar que los String, Integer, etc. son objetos y que por tanto no se pueden comparar usando ==. Consideremos este caso:

Persona p1 = new Persona("Andrés Hernández Suárez", 32, "54221893-D", "Economista);
Persona p2 = new Persona("Andrés Hernández Suárez", 32, "54221893-D", "Abogado");

¿Cómo establecemos si dos personas son iguales? El concepto de igualdad entre objetos, en Java, tiene que ser definido para que el compilador sepa a qué atenerse si se comparan dos objetos, en este caso a dos personas. La comparación usará la sintaxis:

if (objeto1.equals(objeto2)) { Instrucciones si se cumple el criterio de igualdad….. } else { Instrucciones si no se cumple el criterio de igualdad }

Por defecto, Java permite el uso del método equals para todo objeto, pero el criterio por defecto de Java no nos va a ser útil para clases creadas por nosotros: tendremos que definir nuestro criterio de igualdad dentro de nuestra clase. Veremos más adelante cómo se define un criterio de igualdad especificado por el programador para poder comparar objetos.

EJERCICIO

Considera una clase Java que se denomina TripulacionAvion y que tiene como atributos a tres objetos de tipo Persona: Persona piloto; Persona copiloto1; Persona copiloto2.

a) ¿Sería posible que al crear un objeto TripulacionAvion se produjera que piloto, copiloto1 y copiloto2 apuntaran a un mismo objeto, es decir, que existiera una relación de identidad entre los tres atributos?

b) ¿Existiría relación de identidad cuando creemos un objeto TripulacionAvion entre los tres atributos si no se inicializaran en el constructor?

c) ¿Cuál sería el contenido de los atributos si no se inicializan en el constructor y creamos un objeto de tipo TripulacionAvion?

Puedes comprobar si tus respuestas son correctas consultando en los foros aprenderaprogramar.com.

Próxima entrega: CU00664B

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188