Referencias y Operador punto

Material de clase elaborado por Sandra Victoria Hurtado Gil

Tipos de variables en Java

En un programa orientado a objetos lo primero que debe hacerse es definir las clases que se necesitarán. Sin embargo, después de crear las clases, y para que el programa logre los objetivos deseados, es muy importante crear objetos de esas clases y trabajar con sus valores y métodos.

Para poder crear y utilizar los objetos en un lenguaje de programación como Java, es necesario tener variables. Java tiene dos grandes grupos de tipos de datos para poder crear variables:

- Los primitivos (o básicos): corresponden a valores simples y son: *boolean*, *byte*, *short*, *int*, *long*, *float*, *double* y *char*.
- Los **referenciados (u objeto**s): corresponden a las clases, y es el tema que se presentará a cotinuación.

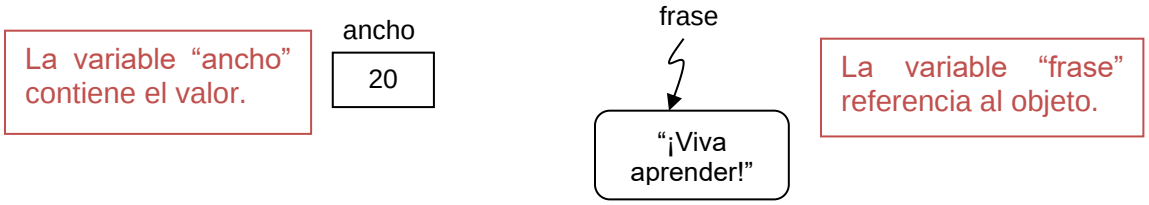
1. Referencias

Cuando se trabaja con tipos primitivos las variables **contienen** un dato o valor, pero cuando se trabaja con objetos las variables **referencian** el objeto, es decir, no lo contienen, sino que permiten llegar al objeto para usarlo.

Por ejemplo, se definirán dos variables: una primitiva y otra referenciada, cada una con un valor inicial.

```
int ancho = 20;
String frase = new String("¡Viva aprender!");
```

Una representación de cómo queda la memoria puede ser:



Las variables de tipo primitivo se usan directamente en las expresiones, mientras que las variables referenciadas por lo general no se usan directamente, sino que a través de ellas se pueden llamar los métodos del objeto. Por ejemplo:

```
int dobleAncho = ancho * 2;
```

Se usa directamente

```
String palabraInicial = frase.substring(1,5);
```

Permite usar el método "*substring*"

Al igual que las variables de tipo primitivo, las variables referenciadas pueden declararse antes de usarlas, pero en ese caso todavía no están referenciando ningún objeto, de la misma forma que una variable primitiva que no se ha inicializado no tiene ningún valor.

Para poder usar estas variables es necesario asignarles algún valor o referencia. A una variable referenciada se le puede asignar un objeto que ya existe, o uno nuevo, usando la instrucción *new*. Cuando a una variable referenciada se le asigna otra variable, lo que se hace es que referencia **el mismo objeto** (NO crea una copia).

Por ejemplo, se tiene la clase Cuenta:

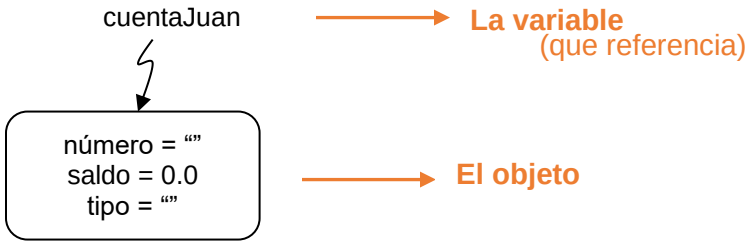
Cuenta
número: String saldo: double tipo: String
consultarSaldo(): double consignar(cantidad: double): void retirar(cantidad: double): boolean

Para crear un objeto de esta clase en Java, la instrucción es:

```
Cuenta cuentaJuan = new Cuenta();
```

Lo que se está haciendo es: creando un objeto de la clase Cuenta, que es **referenciado** por la variable “cuentaJuan”.

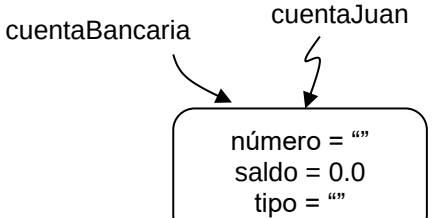
Una forma de representar esto es:



Ahora, se define otra variable de tipo Cuenta, y se le asigna el objeto que se acabó de crear, así:

```
Cuenta cuentaBancaria = cuentaJuan;
```

Una representación de cómo queda la memoria puede ser:



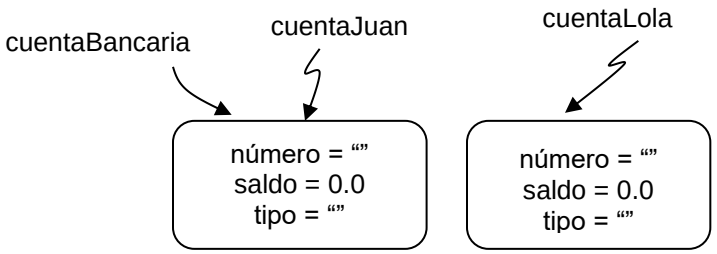
NO se ha creado un nuevo objeto.
La variable “cuentaBancaria” referencia **el mismo objeto** anterior.

Las dos variables están referenciando al mismo objeto.

Ahora se crea un nuevo objeto:

```
Cuenta cuentaLola = new Cuenta();
```

Lo cual se representa en memoria:



Se tienen **dos** objetos Cuenta.

2. Operador punto

Después de crear los objetos, se pueden usar los servicios que ofrecen, es decir, **llamar** a los métodos para realizar alguna operación.

Siguiendo con el ejemplo de la clase Cuenta, se observa que tiene el método “consultarSaldo”. Usando este método es posible saber el saldo o cantidad de dinero de una cuenta, pero antes por lo general se consigna o se retira algún valor, para lo cual se pueden usar los métodos “consignar” y “retirar”.

Por ejemplo, para obtener y mostrar el saldo de una cuenta en la cual se consigna $150.000 y se retiran $50.000 y $20.000, el código en Java es:

```
Cuenta cuenta = new Cuenta();
cuenta.consignar(150000);
boolean pudoPrimerRetiro = cuenta.retirar(50000);
boolean pudoSegundoRetiro = cuenta.retirar(20000);
double saldoFinal = cuenta.consultarSaldo();
System.out.println("El saldo es: " + saldoFinal);
```

Como se puede ver en el ejemplo anterior, la forma de usar (o llamar) los métodos de un objeto es con el operador punto (**.**).

3. Valor *null*

Cuando se definen variables dentro de un método es recomendable asignarles un valor inicial. Esto ayuda a evitar errores en la ejecución del programa.

Por ejemplo, el siguiente código es erróneo:

```
int suma;
suma = suma + 1;  //error
```

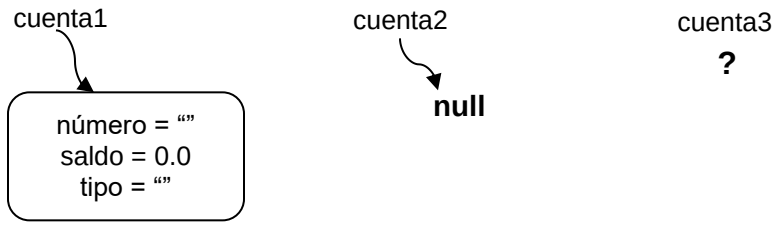
Al tratar de ejecutarlo, como la variable “suma” no tiene un valor inicial, no se puede ejecutar la segunda instrucción.

Lo mismo sucede para las **variables de tipo objeto o referenciadas**: Si no se le asigna ningún valor queda con “basura”, lo cual puede llevar a errores. Por este motivo, en Java se definió un valor especial para indicar que una variable de este tipo no está referenciando a ningún objeto: este valor es ***null***.

Por ejemplo, se tiene el siguiente código:

```
Cuenta cuenta1 = new Cuenta();
Cuenta cuenta2 = null;
Cuenta cuenta3;
```

En el anterior código se tienen tres variables de tipo Cuenta, de las cuales la primera (cuenta1) está referenciando un objeto, la segunda (cuenta2) tiene valor *null* y la última (cuenta3) no tiene ningún valor definido, así:



Posteriormente se desea saber la cantidad de dinero (o el saldo) que tiene cada cuenta – lo cual solo funcionará para la variable que está referenciando un objeto.

Es decir, en el siguiente código solo es correcta la primera instrucción, las dos siguientes generan errores al compilar o al ejecutar el programa, porque las variables no están referenciando ningún objeto:

```
System.out.println(cuenta1.consultarSaldo());
System.out.println(cuenta2.consultarSaldo());        //ERROR
System.out.println(cuenta3.consultarSaldo());        //ERROR
```

Para evitar estos errores se modificará el código. De esta manera, antes de preguntar por el saldo, se validará si la variable está referenciando un objeto, lo cual se hace preguntando si es diferente de ***null***, así:

a) Para cuenta1:

```
if (cuenta1 != null)
{
      System.out.println(cuenta1.consultarSaldo());
}
```

Este código funciona apropiadamente y muestra el saldo que tiene la cuenta.

b) Para cuenta2:

```
if (cuenta2 != null)
{
      System.out.println(cuenta2.consultarSaldo());
}
```

Este código también funciona apropiadamente, y como la variable está con valor *null*, no entra al *if*. No se muestra el mensaje, pero no sale error y el programa continúa normalmente.

c) Para cuenta3:

```
if (cuenta3 != null)
{
      System.out.println(cuenta3.consultarSaldo());
}
```

Este código **genera un error**. Como la variable cuenta3 no tiene un valor definido –ni siquiera *null*– no se puede usar.

En resumen, con respecto al valor *null*:

- Se usa el valor *null* para inicializar una variable de tipo objeto (referenciada) cuando todavía no tiene un objeto asociado.
- Al comparar una variable de tipo objeto con el valor *null* se puede saber si tiene asociado un objeto o no.
- Si trata de usar una variable referenciada que no tiene ningún valor –ni siquiera null– se generará un error.
- Se pueden usar las variables que tengan valor *null,* pero solo para validar si están referenciando o no a un objeto.
- **Solo se pueden usar los métodos en variables que referencien un objeto**. No se pueden usar en variables que no tengan valor o que sean *null* porque se generaría un error.