Variables y asignaciones

Sobre concepto de variable en web oficial de oracle

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html

o en wiki san clemente (más o menos una traducción de lo anterior)
http://manuais.iessanclemente.net/index.php/Elementos_esenciais_da_linguaxe_Java#Variables

Por el momento puedes obviar la clasificación de variables de estos enlaces y pensar simplemente en el concepto que exponemos a continuación.

CONCEPTO DE VARIABLE

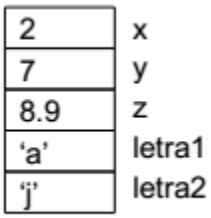
Vemos una visión ultra simple del concepto de variable y poco a poco, en unidades posteriores, la iremos enriqueciendo.

La gestión de memoria que hace el sistema operativo y un compilador/intérprete es muy compleja. Como primer punto de partida, sin rigor, imagina la memoria RAM que utiliza tu programa organizada en posiciones. Las posiciones puedes imaginarlas como "cajones" o celdas de una tabla.

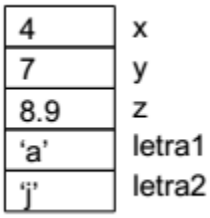
Estas posiciones almacenan valores con los que trabaja nuestro programa.

2
7
8.9
'a'
'j'

A estas posiciones le podemos cambiar de valor y por eso les llamamos "variables". Java nos permite acceder a estas posiciones de valor variable a través de un nombre que declaramos en nuestro programa java y que representamos en este gráfico a la derecha de la posición.



Si se ejecuta la sentencia
x=4;
La memoria cambiará de forma que x contendrá un nuevo valor



LOS TÉRMINOS SENTENCIA E INSTRUCCIÓN

Ambos son una especie de "frase" del lenguaje de programación. Incluso informalmente podemos verlas como "órdenes" del lenguaje.
A veces estos términos se emplean indistintamente, pero más finamente, en la actualidad, con el término instrucción nos referimos a una instrucción de ensamblador (bajo nivel) y las instrucciones de los lenguajes de alto nivel pasan a llamarse sentencias de forma que al traducir el código de alto nivel normalmente implica que una sentencia se traduce a varias instrucciones de bajo nivel.

LA SENTENCIA DE ASIGNACIÓN

En una instrucción del tipo
x=2;
Decimos que estamos asignando el valor 2 a x. Es intuitivo su funcionamiento, pero es tal su importancia que a pesar de su simpleza debemos reflexionar en este proceso. En el libro piensa en java nos indica (se resume): *La asignación se realiza con el operador =. Su significado es: toma el valor del lado derecho y cópialo en el lado izquierdo. En el lado derecho nos podemos encontrar una constante, una variable o una expresión que genere un valor. El lado izquierdo debe ser siempre una variable.*

Tras analizar los ejemplos que siguen, debes de entender la frase al 100%

Por tanto, lo primero que hace java es "calcular" o "evaluar" el lado derecho, una vez que tiene un valor, lo asigna a la variable del lado izquierdo. Por ejemplo:

- x=2;evaluar el lado derecho es inmediato ya que se trata de una constante. a x se le asigna el valor 2 o dicho de otra forma *en la variable x se copia el valor 2*. Observa que copiar un valor en una variable implica "machacar" (sobreescribir) su contenido precedente.
- x=2+3; se suma 2 y 3, se obtiene 5 y se le asigna a x el valor 5
- 2=5; el lado izquierdo debe ser una variable, esto no es posible
- x=b; supongamos que b almacena el valor 1. java accede a la variable b y lee su contenido, el valor uno, y lo asigna a x

- x=b+2; supongamos que b es una variable que almacena el valor 1. java accede a la variable b y lee su contenido, el valor uno, a este valor le suma 2 y por tanto a x le asigna 3
- x=x+2; supongamos que x almacena el valor 1. java accede a la variable x y lee su contenido, el valor "1", a este valor le suma 2 y por tanto a x se le asigna finalmente 3. aquí se ve la importancia del orden con el que procede java: primero se evalúa el lado derecho por lo que es posible, y muy habitual, que en la expresión del lado derecho aparezca la propia variable a la que queremos asignarle un nuevo valor.
- 2=x; el lado izquierdo debe ser una variable, esto no es posible.

Ejemplo: ejecútalo y entiende su resultado

```
public class Unidad1 {
    public static void main(String[] args) {
        int edad; //declaramos (definimos) una variable indicando su tipo
        edad=60; //le damos un valor a una variable
        int peso=75;  //al mismo tiempo que se define una variable se le puede asignar un valor
        System.out.println(edad);
        System.out.println(peso);
        //el valor de la variable se puede imprimir junto a cadenas de caracteres
        System.out.println("mi edad es: "+edad);
        peso=80;
        System.out.println(peso);
        //puedo hacer operaciones con las variables
        System.out.println("el peso de los años ..." + peso*edad);
        //el resultado de una operacion puede almacenarse en otra variable
        int salud;
        salud=peso*edad;
        System.out.println("la variable salud contiene el valor: " + salud);
        //el valor que se le asigna a una variable deber ser compatible con su tipo
        //edad="jamones para todos";
    }
}
```

Ejercicio: descomenta la última instrucción y observa el error del compilador. Me indica que hay incompatibilidad de tipos ya que tras evaluar el lado derecho obtiene un String pero el lado izquierdo es un entero. No se puede asignar un string a una variable entera por ello me indica que se requiere un int pero se encontró con un string.

Nota: Observa que en println() se trabaja con una variedad de valores como cadenas de caracteres y números enteros y que el operador "+" lo que hace es unir(concatenar) todos estos valores para imprimirlos. Más adelante entenderás mejor el funcionamiento del operador "+".

LA DECLARACIÓN DE VARIABLES.

Si consultas el enlace que explica lo que es una variable, observa que hay un tipo de variable llamada *variable local*. En este boletín sólo nos referimos a las variables locales. Como ya vimos en el ejemplo anterior, en java antes de usar una variable hay que declararla indicando su tipo.

```
public class Unidad1 {
    public static void main(String[] args) {
```

```
        int edad; //declaramos (definimos) una variable
        edad=60; //le damos un valor a una variable
    }
}
```

Observa el error de compilación si intentamos usar una variable sin declarar. Compila por ejemplo

```
public class Unidad1 {
    public static void main(String[] args) {
            edad=60; //le damos un valor a una variable sin declarar
    }
}
```

LA INICIALIZACIÓN DE VARIABLES LOCALES.

Una variable "nace" en la memoria de nuestro programa cuando se declara, pero inicialmente no tiene valor alguno, debemos de inicializarla nosotros en el programa antes de usarla. ¿Qué es inicializar una variable?. Darle un valor por primera vez.
Se puede inicializar en el momento de declararla o más tarde, pero antes de usarla debe de tener un valor.
Observa como da error de compilación intentar usar la variable edad si no tiene valor.

```
public class Unidad1 {
    public static void main(String[] args) {
        int edad;
        System.out.println("edad: "+edad);
    }
}
```

Si ahora compilamos con edad inicializada no hay error

```
public class Unidad1 {
    public static void main(String[] args) {
        int edad;
        edad=60;
        System.out.println("edad: "+edad);
    }
}
```

o lo que es equivalente, al mismo tiempo que declaramos una variable la inicializamos.

```
public class Unidad1 {
    public static void main(String[] args) {
        int edad=60;
        System.out.println("edad: "+edad);
    }
}
```

EL PUNTO Y COMA

Un programa es un conjunto de sentencias. A java le indicamos cuando termina una sentencia (y por tanto empieza otra, si es el caso) a través del carácter ";"
Lo más habitual es escribir cada sentencia en una línea física del editor, por ejemplo:

```
class Unidad1 {
    public static void main(String[] args) {
        int edad;
        edad=60;
        int peso=75;
```

```
      System.out.println("edad: "+edad);
      System.out.println("peso:"+peso);
   }
}
```

Pero ya que lo que realmente separa una sentencia de otra es el punto y coma lo podríamos haber escrito, por ejemplo:

```
public class Unidad1 {
   public static void main(String[] args) {
      int edad; edad=60;int peso=75;
      System.out.println("edad: "+edad); System.out.println("peso:"+peso);
   }
}
```

ESPACIOS EN BLANCO

Observa que en el código java aparecen separando palabras, números y símbolos "espacios en blanco". Algunos son obligatorios por ejemplo entre la palabra reservada "class" y el nombre de la clase tiene que haber al menos un espacio. Otros espacios son opcionales y se suelen incluir para que sea más legible el código, por ejemplo, una sentencia de asignación tiene 3 elementos:

- parte izquierda
- operador =
- parte derecha

Pero entre las partes puede haber espacios,por ejemplo
edad=60;
Se puede escribir
edad = 60;
O también
edad= 60;
antes, entre y después de cada elemento puede haber cualquier número de espacios en blanco. Los espacios en blanco son por tanto desde el punto de vista de la ejecución del programa irrelevantes pero son importantes para la legibilidad del programa.

Ejemplo: Ejecuta los siguientes códigos equivalentes y piensa cual de ellos "lees mejor"

```
public class Unidad1 {
   public static void main(String[] args) {
      int edad;
      edad=60;
      int peso=75;
      System.out.println("edad: "+edad);
      System.out.println("peso:"+peso);
   }
}
```

```
public class Unidad1 {
   public static void main(       String[]          args) {
      int edad;                    edad=                                        60;
                  int peso=75                        ;
                           System.out.println("edad: "+edad);
System.out.println("peso:"+peso);
                           }
                                 }
```

```
public class Unidad1 {
   public static void main(String[] args)                    {
      int edad; edad=60;
      int peso=75;  System.out.println("edad: "+edad); System.out.println("peso:"+peso);
   }
```

```
                    }
```

Fíjate que como programador, tienes que comunicarte correctamente con:

- Javac: si no cumples sus normas te da error
- Con otros programadores: si no cumples convenciones de estilo de escritura, aunque tu código sea ok para javac, otros programadores pueden estar incómodos leyéndolo y pueden entenderlo mal. Algo parecido a las lenguas manuscritas, si escribes con mala letra, desorden, tachones, … aunque todo lo que escribas sea conceptualmente correcto ...
- Contigo mismo : el principal lector de tu código java vas a ser tu mismo. Siguiendo el ejemplo anterior, , si no te entiendes la letra ni tu mismo pues ...

IDENTIFICADORES

Los nombres que el programador “inventa” en su programa para nombrar a una variable, un método, una clase etc. se llaman identificadores.

LOS IDENTIFICADORES DE VARIABLES

Lee sobre esto en
http://manuais.iessanclemente.net/index.php/Elementos_esenciais_da_linguaxe_Java#Nomes_de_variables

Ejemplo:

```
class Unidad1 {
   public static void main(String[] args) {
      int x=45;
      int x2= x*2;
      int variable_x=23;
      int x88=0;
      int variableMuyBonita=19;
      int  areaDeCuadrado=25;

         System.out.println("El valor de variableMuyBonita es:"+ variableMuyBonita);


       }
}
```

Modifica el programa anterior para comprobar que se produce error de compilación en los siguientes casos:

- cambio x2 por 2x => el nombre de variable no puede empezar por un dígito
- cambio x88 por nombre void =>el nombre de variable no puede ser una palabra reservada. En el siguiente apartado veremos cuál son las palabras reservadas en Java
- cambio x88 por class=>idem caso anterior
- cambio x88 por ?x=>el primer caracter tiene que ser una letra o bien un “$” o “ _”

Comprueba que no da error:

- cambio x88 por $88
- cambio x88 por _88

Palabras reservada del lenguaje java

Un identificador, además de tener una combinación de caracteres permitida, no puede coincidir con una palabra reservada del lenguaje java
Palabras reservadas del lenguaje java:
http://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

Ejercicio U1_B3_E1: En una clase AreaCuadrado escribe un programa que guarde en una variable entera la longitud del lado de un cuadrado, con dicha variable calcule el área del cuadrado y almacene este valor en otra variable. Imprime por pantalla el valor de esta última variable.

Ejercicio U1_B3_E2: En este ejercicio tenemos que intercambiar el valor de dos variables. Es importante entenderlo bien pues se utiliza mucho y en todo tipo de programas

```
class IntercambioVariables{
  public static void main(String[] args){
    int x=10;
    int y=20;
    int z=0;
    System.out.println("ANTES. x vale "+ x +  " y vale "+y );
    ……..instrucciones que realizan el intercambio de valores
    System.out.println("DESPUES. x vale "+ x +  " y vale "+y );
  }
}
```

añadir instrucciones que intercambien los valores de x e y, pero de forma que la solución sea genérica para cualesquiera valores que contenga x e y en un momento dado. Para ello debes utilizar una tercera variable z.

CONVENCIONES DE IDENTIFICADOR DE VARIABLES

Las convenciones a la hora de escribir código son normas no obligatorias pero recomendables. Si todos las cumplimos, leemos más fácilmente el código de otros e incluso mismamente el nuestro.
Vimos anteriormente las reglas que definen cuales son identificadores legales e ilegales de variables. Dentro de los legales, se recomienda seguir una serie de convenciones. En el caso de las variables las recomendaciones son las siguientes:

- Poner nombre descriptivos.
  Por ejemplo si estoy calculando el área de un rectángulo

```
int area;
int alto=10;
int ancho=20;
area=alto*ancho;
```

  Es más legible que

```
int x;
int y=10;
int z=20;
x=y*z;
```

- Usar sólo letras minúsculas

```
int area;
```

  Mejor que

```
int Area;
```

- Para nombres largos de variables que incluyen varias palabras, empezar con mayúscula cada nueva palabra

  ```
  int areaCirculo;
  ```

  Mejor que

  ```
  int AreaCirculo;
  ```

  y que

  ```
  int areacirculo;
  ```

VARIABLE CONSTANTES

A las variables que una vez inicializadas no queremos que se les cambie el valor se llaman variables constantes o simplemente constantes. Parece en principio una contradicción pero tiene su uso y su sentido.
Para declarar una variable constante se usa la palabra reservada ***final***. Esta palabra tiene varios usos en java y declarar variables constantes es una de ellas. Otra forma de llamar a las constantes en el mundo java es el de *variables finales*.

```
final int VELOCIDAD_LUZ=300000;
```

Las normas para elegir identificadores son las mismas que para cualquier otra variable pero cambia la convención: escribir todo en mayúsculas y si el identificador consta de varias palabras concatenar con guión bajo

Ejercicio: comprueba que una vez que se inicializa una constante no se le puede cambiar de valor

```
final int VELOCIDAD_LUZ=300000;
VELOCIDAD_LUZ=20;//¡error!
```