PROGRAMACION II MARKETING

LECTURAS 5 SESION

http://www.utn.edu.ec/reduca/programacion/poo/atributos.html

http://www.mundojava.net/definicion-de-atributos-de-una-clase.html

Atributos

Cada objeto tiene unas características que le identifican como tal, así:

OBJETO	CARACTERÍSTICAS	OBJETO	CARACTERÍSTICAS
rectángulo	lados	**carro**	placa
	ángulos		color
			marca
persona	nombre	**factura**	nroFactura
	cédula		fecha
	fecha Nacimiento		totalPago
libro	título	**detalleFactura**	nroFactura
	autor		codProducto
	año		cantidad

Para identificar las características de un objeto, se puede utilizar la palabra TIENE. Ejemplo:

Un rectángulo tiene: lados y ángulos;

Una persona tiene cédula, nombre, fecha de nacimiento;

Un libro tiene: título, autor, año;

Un carro tiene: placa, color, modelo;

Una factura tiene: nroFactura, fecha, totalPago

Un detalle tiene: nroFactura, producto, cantidad

A estas características se les denomina también propiedades y técnicamente se conocen como ATRIBUTOS del objeto.

LECTURA 2

Los atributos de una clase son definidos según esta sintaxis:

[modifVisibilidad] [modifAtributo] tipo nombreVariable [= valorInicial] ;

Donde nombreVariable es el nombre que daremos a la variable, siendo un nombre válido según las normas del lenguaje:

- por convención, en Java, los nombres de las variables empiezan con una letra minúscula (los nombres de las clases empiezan con una letra mayúscula).
- Un nombre de variable Java: debe ser un identificador legal de Java comprendido en una serie de caracteres Unicode. Unicode es un sistema de codificación que soporta texto escrito en distintos lenguajes humanos. Unicode permite la codificación de 34.168 caracteres. Esto le permite utilizar en sus programas Java varios alfabetos como el Japonés, el Griego, el Ruso o el Hebreo. Esto es importante para que los programadores pueden escribir código en su lenguaje nativo.
- no puede ser el mismo que una palabra clave
- no deben tener el mismo nombre que otras variables cuyas declaraciones aparezcan en el mismo ámbito.

Es el tipo de la variable, pudiendo ser un tipo básico o un objeto de una clase o de un interfaz. También puede ser una matriz o vector.

Veamos ahora que tipos de datos básicos existen en el lenguaje y sus características:

Tipo	Formato	Descripción
(Números enteros)		
byte	8-bit complemento a 2	Entero de un Byte.
short	16-bit complemento a 2	Entero corto.
int	32-bit complemento a 2	Entero.
long	64-bit complemento a 2	Entero largo.
(Números reales)		
float	32-bit IEEE 754	Coma flotante de precisión simple.
double	64-bit IEEE 754	Coma flotante de precisión doble.
(otros tipos)		
char	16-bit Carácter	Un sólo carácter (Unicote).
boolean	true o false	Un valor booleano (verdadero o falso).

Tenemos un ejemplo del uso de los tipos de datos en Tipos.java

modifVisibilidad indica desde que parte del código se puede acceder a la variable:

- public: indica que es un atributo accesible a través de una instancia del objeto.
- private: indica que a través de una instancia no es accesible el atributo. Al heredar el atributo se convierte en inaccesible.
- protected: indica que a través de una instancia no es accesible el atributo. Al heredar si se puede usar desde la clase derivada.
- Sin especificar: indica visibilidad de paquete, se puede acceder a través de una instancia, pero sólo desde clases que se encuentren en el mismo paquete.

valorInicial permite inicializar la variable con un valor.

Se permite definir más de una variable, separándolas por coma, por ejemplo:

```
public int a = 5, b, c = 4;
```

modifAtributos son características específicas del atributo, son:

- static: El atributo pertenece a la clase, no a los objetos creados a partir de ella.
- final: El atributo es una constante, en ese caso debe de tener valor inicial obligatoriamente. Por convenio en java las constantes se escriben en mayúsculas.
- transient: Marca al atributo como transitorio, para no ser serializado. Lo emplearemos en java beans.
- volatile: es un atributo accedido de forma asíncrona mediante hilos, con este atributo se lo notificamos a java.

En java definir un atributo de un tipo básico o tipo String significa que podemos acceder a dichas variables de forma directa (ejemplo PruebaVariable1.java):

```
int a = 25;
a = 34;
```

en cambio intentemos definir y emplear una variable del tipo, por ejemplo, Thread:

```
Socket c=null;
c.close();
```

si tratamos de compilarlo (fichero PruebaVariable2.java) todo irá bien, pero al ejecutarlo aparecerá un error:

```
Exception in thread "main" java.lang.NullPointerException
```

esto nos quiere indicar que hemos intentado emplear una variable que no apuntaba a un objeto válido, sino a null, y hemos intentado llamar a una función del objeto inexistente.

Es decir, en java las variables de tipo básico son el nombre de una zona de memoria en la cuál podemos almacenar valores, pero que en cambio, las variables de tipo objeto son en realidad referencias (punteros o alias) de objetos.

Una variable de tipo objeto no es un objeto completo, sino tan sólo almacena la situación del objeto en la memoria del equipo. Esto es muy similar a lo que ocurre con las casas y las direcciones de dichas casas: la dirección calle Alcalá 790 es una dirección válida, pero no podemos mandar cartas a dicha dirección porque es un descampado!!!

Lo mismo sucede con los objetos, podemos tener una variable para referirnos a objetos, pero la variable puede que no apunte a ningún objeto y por tanto no la puedo emplear para intentar acceder a un método o a un atributo del objeto referenciado por la variable, sencillamente porque no existe el objeto referenciado.

Una variable que no apunta a un objeto se asume que tiene un valor especial llamado null, e incluso podemos asignar el valor null a la variable:

```
Thread t = null;
```

Es por ello que se deben de construir objetos y asignarselos a las referencias, usando la palabra clave new. new permite crear un objeto a partir de la descripción de la clase que le pasamos como argumento, por ejempo:

```
new Persona()
```

Conseguimos crear un objeto de la clase Persona, los paréntesis permiten especificar que constructor estamos llamando al crear al objeto (veremos constructores más adelante).

Pero al crear un objeto persona como en el código anterior lo estamos creando como un objeto anónimo, es decir sin asignar el objeto a una variable de tipo referencia, desde la cuál poder referirnos al objeto y poder llamar a sus métodos y atributo, por ello lo más habitual sería asignar el objeto a una variable como en:

```
Persona p = new Persona();
```

y así poder acceder al objeto Persona recien creado:

```
p.nombre = "Alberto";
```

Práctica:
Vamos a construir la clase Persona, pero no estará completa hasta que no la completemos con características que iremos viendo en capítulos posteriores. Fabricaremos la clase Persona poco a poco, para ello:

- Construir el fichero Persona.java

- Agreguemos la clase dentro del fichero:

```
public class Persona
{
}
```

- Agreguemos a la clase los atributos de persona (en este caso seleccionaremos dos atributos):

```
private String nombre = null;
private int edad;
```

- Como resultado final obtendremos:

```
public class Persona
{
    private String nombre = null;
    private int edad;
}
```

Ahora vamos a crear una clase que emplearemos única y exclusivamente para que tenga la función main, y así poder escribir código que queremos ejecutar. Llamaremos a dicha clase Arranque, por tanto:

- Creamos el fichero Arranque.java
- Agreguemos la clase y el método main en su interior:

```
public class Arranque {
    public static void main (String arg[]){
    }
}
```

- Ahora agregaremos dentro de la función main el siguiente código:

```
Persona per1 = new Persona();
//per1.nombre = "Luis";
```

```
        //System.out.println( per1.nombre);
```

en el cuál declaramos una variable de tipo Persona y la damos el valor de un nuevo objeto de la clase Persona, en la segunda línea asignamos al atributo nombre del objeto per1 el valor Luis, y en la tercera línea mostramos por pantalla (gracias al método System.out.println) el valor de dicha variable. La segunda y tercera líneas están comentadas, debido a que el atributo nombre está definido como privado en la clase Persona.