UNED
Centro Asociado de Cádiz

RESUMEN DE CONCEPTOS BASICOS DE PROGRAMACION JAVA

1. OBJETOS

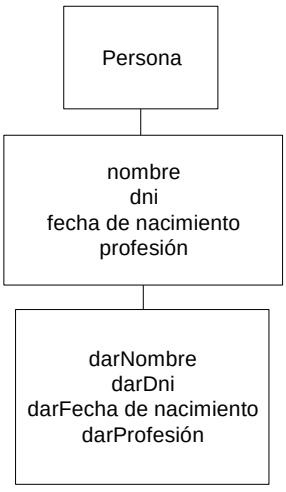
Cualquier elemento del programa es un objeto.
Un programa es un conjunto de objetos que se comunican entre sí (mensajes).
Cada objeto está compuesto de otros objetos, sus atributos.
Cada objeto es de una clase o tipo determinado. La clase determina cómo está compuesto el objeto (atributos) y cómo se comporta (operaciones que puede realizar).

2. CLASE

Una clase agrupa objetos con los mismos atributos e idénticas operaciones.
Los atributos son los datos internos del objeto.
Las operaciones o métodos, representan los mensajes que puede recibir y procesar de otros objetos.
La clase equivale a la definición de la estructura global y los objetos son instancias de la clase o casos concretos de utilización.
Un programa o aplicación, se compone de clases de las que se han creado objetos que se comunican entre sí mediante mensajes, es decir, llamando a los métodos de los objetos.

Por ejemplo, para representar mediante objetos a 2 personas por su nombre, dni, fecha de nacimiento y profesión, dentro de un programa que sea capaz de realizar consultas a estos objetos sobre los 4 datos indicados, habrá que proceder de la siguiente forma:

Se crea la clase persona
Se crean atributos nombre, dni, fecha de nacimiento y profesión
Se crean métodos darNombre, darDni, darFecha de nacimiento, dar Profesión



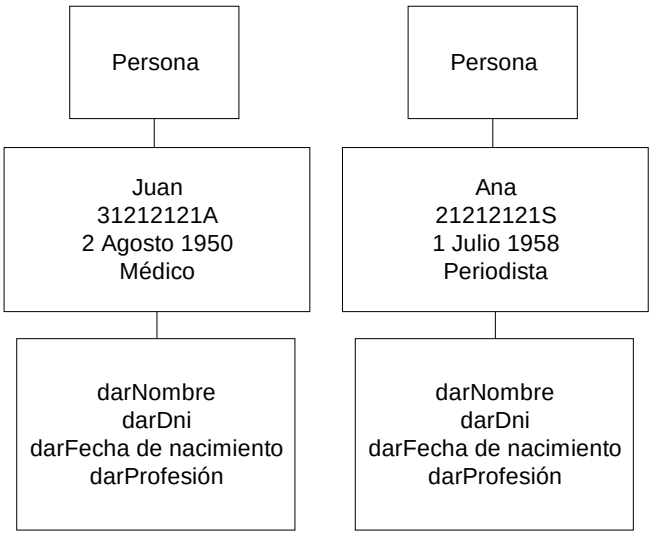
La clase es el nombre de la estructura global, los atributos indican los datos internos que contendrá la clase y las operaciones muestran los mensajes que será capaz de recibir y responder.

Una instancia u objeto es una utilización concreta de esta clase, es decir:

Juan, 31212121A, 2 Agosto 1950, Médico

Un segundo objeto es otra utilización de la clase con un nuevo caso concreto:

Ana, 21212121S, 1 Julio 1958, Periodista



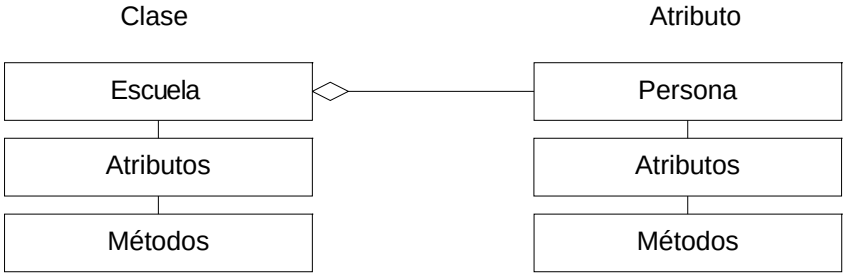
Una clase, por tanto, se construye mediante 3 bloques que son:

NOMBRE – ATRIBUTOS – METODOS

Un objeto es la utilización específica de la estructura construida en la clase.

3. COMPOSICION DE CLASES

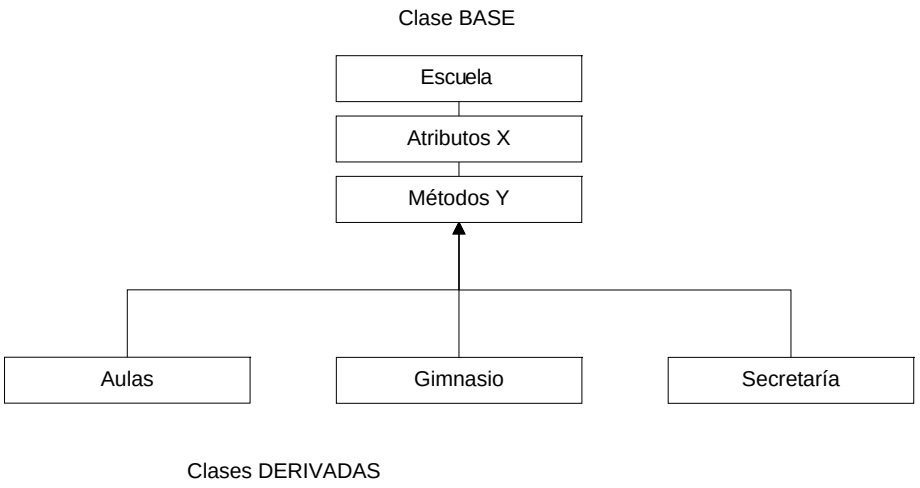
Se pueden crear clases complejas agregando objetos ya creados de otras clases definidas con anterioridad. Por ejemplo, si creamos la nueva clase “Escuela”, uno de sus atributos podría ser la clase definida anteriormente como “Persona”. Se representa en el diagrama mediante una línea de conexión que termina en rombo.



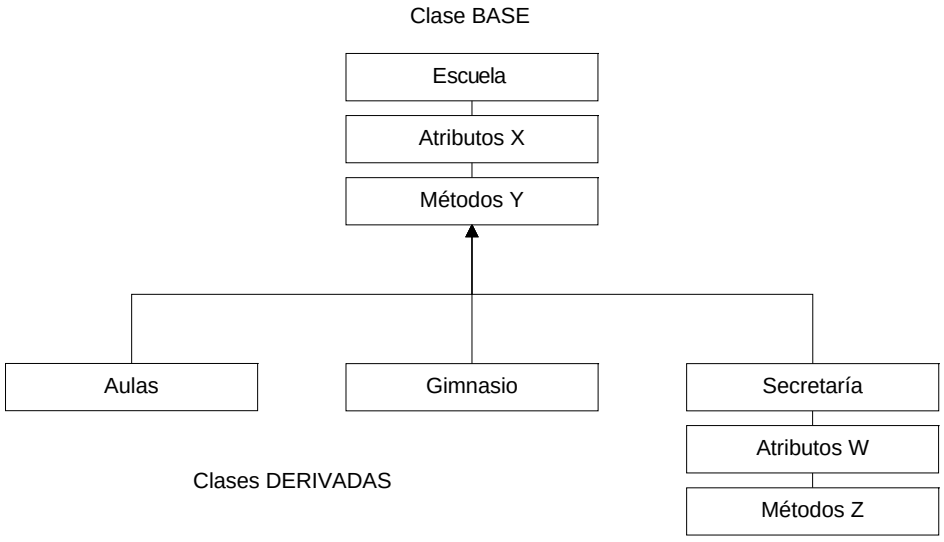
4. HERENCIA DE CLASES

La herencia define similitud entre clases, de forma que se enlazan gráficamente una clase base y otras clases derivadas.

La clase base, tiene unos atributos y métodos que va a transmitir a las clases derivadas. Las clases que han heredado, tienen todos los atributos y el interfaz de la clase base.



Cualquier clase derivada puede añadir nuevos atributos o nuevos métodos a los ya heredados. No serán parte de la clase base sino una extensión.



También se puede sobrescribir algún método, escribiendo de nuevo en la clase derivada el método que aparece en la clase base. De esta forma, la implementación es diferente.

Programación con Java

Comentarios:

/** … */
/* … */
// … (hasta final de línea)
* … para líneas internas entre /* y */

Constantes y variables:

byte (entero con signo, ±128)
short (entero con signo, ±32768)
int (entero con signo, por defecto en enteros)
long (entero con signo) (también se emplea una L después del dato)
float (real, simple precisión) (también se emplea una F después del dato)
double (real, doble precisión, por defecto) (se puede usar D después del dato)
char (carácter, 'a', 'b', …)
boolean (false/true)
string ("texto …") (para unir 2 textos se usa +)
final (se pone delante para definir una constante) (las constantes se escriben en mayúsculas)

```
33L
3.66F
double radio = 88.8
final double PI = 3.1415926536
```

Operaciones:

Unarias: cambio de signo
+ , - , * , / , % (resto de la division entera)
++ , -- (incremento, decremento) (++a preincremento, a++ postincremento)
> , < , >= , <=
== , != (operadores de igualdad, desigualdad)
= , += , -= , *= , /= , %= (operadores de asignación)
! (booleano, negación)
&& (Y lógico, AND)
|| (O lógico, OR)
Orden: unarias, multiplicativos, aditivos, relación, asignación

Ejemplo:

```
public class Hola {
 public static void main (String[] args) {
  System.out.println ("Hola");
  }
}
```

Clase:

```
Class Nombre { … }
```

Definición de objetos:

```
Nombre objeto1;
Nombre objeto2;
```

Creación del objeto:

```
objeto1 = new Nombre()
objeto2 = new Nombre()
```

Definición y creación se pueden unir en unir en la misma línea.
`null` (si no hay nada que asignar)
`objeto1 = objeto2` (alias, un objeto con 2 referencias)

Atributos:

Definen el estado de un objeto. Son de clase o de objeto.
`static` (atributos de clase)
Definición: < acceso > < tipo > < nombre > = Valor inicial

```
string nombre = null
```

Métodos:

Pueden devolver un resultado, pero ese valor debe ser del tipo declarado.
`void` (se pone delante del nombre cuando no se devuelve resultado)
`string` (devuelve string)

Cabecera: public / tipo del valor a devolver / nombre (parámetros) excepciones

```
public void imprime(){ System.out.println ( … ) }
```

Invocación desde fuera de la clase: `objeto.metodo()`
Si es un método de clase, static: `class.metodo()`
Si está en la misma clase: `metodo()`
Si no es `void`, se requiere asignación: `z = nombre…()`
Si es `void`, no hace falta asignación
`Public static void main()` es el método principal de ejecución.

```
class Nombre {
  string nombre;
  string apellidos;
  int dni;
  char horario;
   …
  System.out.println ("…" + objeto.componente)
   …
}
```

Caracteres especiales:

El tipo de datos carácter, entre comillas simples ‘a’
El tipo de datos string, entre comillas dobles “a”
`\b` retroceso
`\t` tabulador
`\n` salto de línea (línea1 \n línea2 \n línea3 …)
`\r` cambio de línea
`\”` comillas dobles
`\’` comillas simples
`\\` barra atrás

Acceso a los atributos de una clase:

a) Desde el método `main`
   `objeto.atributo=`…
   (`objeto1.dni=33333333, objeto1.nombre=”Juan”`)
b) Desde un método que no es main
   `return atributo`
   (`return dni, return nombre`)
c) Desde un método que no es main con autoreferencia
   `this.atributo`
   (`this.dni, this.nombre`)

No se puede acceder a ningún atributo o método de una referencia que vale null.
Antes de usar una referencia hay que comprobar que no es null:

```
if (objeto != null) …
```

Un objeto puede tener varias referencias, que se llaman alias.

```
clase objeto1
objeto2=objeto1 (se copia la referencia, no el objeto)
```

Java inicializa variables de tipo primitivo que son atributos de clase (0 para números y caracteres y false para boolean) pero no para variables locales definidas dentro de un método. Para inicializar objetos hace falta un programa constructor.

Estructuras de control:

if (condicion) { … } else { … }

```
if (x<6) {System.out.println(“menor que 6”);}
else {System.out.println(“x mayor que 6”);}
```

Se admite el operador condicional ?
(condicion) ? primer valor si es true : segundo valor si es false
Se admite anidación:

```
if (...) { ..... }
     else if (...) { ..... }
     else if (...) { ..... }
else { ..... }
```

while (condicion) { … }

```
while (x<6) {
  System.out.println(“la variable x es: “ + x);
  x++;
}
```

do { … } while (condicion)

```
do {
  System.out.println(“la variable x es: “ + x);
  x++;
}
while (x<6)
```

for (inicializacion;condicion;actualizacion) { … }

```
for (int x=0; x<6; x++) {
  System.out.println(“la variable x es: “ + x);
}
```