

3. Programación orientada a objetos¹

A manera de introducción planteamos una situación que iremos resolviendo conforme avancemos en los temas.

Imagina que tienes una veterinaria y vas a modelar un programa para interactuar con los animales que llegan a tu consultorio. Generalmente reciben solamente perros y gatos (indicador de qué vamos a modelar) y te interesa saber datos básicos de éstos como: edad, nombre, peso y nombre del dueño. Además, quieres automatizar ciertos cálculos como saber su edad en años humanos, e interactuar con ellos para alimentarlos y sacarlos a correr.

Gato
Nombre Edad Peso Noombre del dueño
Edad Humana Alimentar Intarctuar Imprime Informacion

No nos dan mucha información extra del problema, si no queda claro es mejor seguir preguntando (levantando requerimientos) hasta que quede claro todo y podamos modelar el problema. En este caso, sólo necesitamos las propiedades básicas de los animales y algunas funciones para interactuar con ellos y obtener información.

Un diagrama básico (UML) de lo que serían nuestros objetos sería el siguiente:

Perro
Nombre Edad Peso Nombre del dueño
Edad Humana Alimentar Interactuar Imprime Informacion

¹ Documento elaborado por el Ing. Arturo Nereu Núñez.

Donde la primera sección denota el nombre de la clase, la segunda los atributos de nuestra clase y por último los métodos que tendrá dicha clase.

Como verás, esta abstracción carece de los tipos de datos tanto para los atributos como los métodos, de modo que convendría poner estos valores en nuestro diagrama:

Perro	Gato
Nombre : String Edad : int Peso : float Nombre del dueño : string	Nombre : String Edad : int Peso : float Noombre del dueño : String
Edad Humana () : int Alimentar (int tipoComida) : void Interactuar() : String Imprime Informacion() : void	Edad Humana() : int Alimentar(int tipoComida) : void Interactuar() : String Imprime Informacion() : String

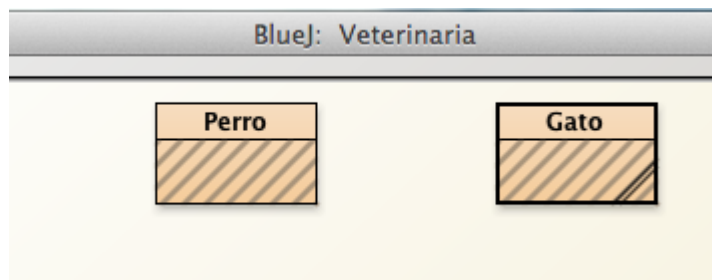
Estos diagramas nos dan más información sobre cómo debemos implementar nuestras clases y dado que nombramos de forma clara los atributos y métodos, también podemos saber qué hace cada uno.

Vamos pues a seguir con nuestro ejercicio y vayamos profundizando en otros temas.

Declaración de clases

Comencemos declarando los elementos de nuestra clase, como vimos anteriormente las clases se componen de un nombre (un identificador único de la clase), atributos y métodos.

Abre BlueJ y crea un nuevo proyecto llamado **Veterinaria**, dentro de tu proyecto crea dos clases. ¿Sabes cuáles? Claro, una llamada Perro y otra Gato.



Ahora vamos a abrir el archivo Perro y a darle estructura a nuestra clase, primero vamos a declarar los atributos:

Atributos – Variables

```
public class Perro
{
    private int edad;
    private String nombre;
    private float peso;
}
```

Nuestro código, hasta este momento queda así, solamente con sus atributos, veamos la estructura:

Primero, tienen el modificador de acceso que puede ser *public* o *private* (hay más pero concentrémonos en éstos). Los cuales indican si la variable de instancia (*atributo*) puede ser accesada desde dentro de la clase o también desde afuera, en este caso éstos son privados por lo que sólo los podemos acceder desde la clase misma y sus métodos.

Después viene el tipo de dato, que puede ser primitivo o compuesto (String es un tipo de dato no primitivo para representar texto).

Por último, viene el nombre de la propiedad que sigue las reglas para cualquier identificador en Java. Para repasar este tema visita:

<http://www.codexion.com/tutorialesjava/java/javaOO/variables.html>

Asignar valores a los atributos

El objetivo de tener una variable, es que podemos modificar su valor para cambiar el estado de los objetos, una buena práctica es inicializar las variables para que al instanciar los objetos tengan un valor por defecto.

Así, inicializadas quedaría algo como:

```
public class Perro
{
    private int edad = 0;
    private String nombre = "Perro";
    private float peso = 1.0f;
}
```

Tenemos nuestras variables declaradas y asignadas, estos valores cambiarán conforme la ejecución del programa, pero ahora tienen un valor al inicio.

Métodos de la clase

Ya con nuestra clase con sus atributos procederemos a implementar los métodos en el diagrama:

```
public class Perro
{
    private int edad = 0;
    private String nombre = "Perro";
    private float peso = 1.0f;

    //Calcula la edad del perro en años humanos, supongamos que 1 año de perro es igual a 7 años humanos
    public int edadHumana()
    {
        //multiplicamos la edad del animal utilizando su atributo edad por el número 7 y lo regresamos
        return edad * 7;
    }
}
```

Como vimos en la unidad anterior, declaramos nuestro método con la estructura de: modificador, valor de retorno, nombre del método y parámetros(en este caso no hay).

Si nuestros métodos son claros, basta con leer su declaración para saber qué hacen, pero en ocasiones es necesario poner comentarios. En JAVA, para poner un comentario de línea, se debe comenzar poniendo “//” doble diagonal.

Para los métodos, los modificadores de acceso funcionan igual que las variables, en este caso, vamos a permitir que ese método sea invocado desde afuera de la clase.

Implementando los demás métodos:

```
public void alimentar(int tipoDeComida)
{
    if(tipoDeComida == 0)
    {
        System.out.println("Soy un perro comiendo carne");
    }
    else
    {
        System.out.println("Soy un perro comiendo croquetas");
    }
}
```

Este método no regresa ningún valor, pero recibe un parámetro, que es un tipo de dato entero. Con este parámetro hará una comparación, si se alimentó con un valor de “0” entonces imprimirá un mensaje a consola de que comió carne, si recibe cualquier otro valor, dirá que comió croquetas. Veremos más sobre control de flujo en unidades posteriores.

```
//Al interactuar con el perro, regresa un mensaje con su onomatopeya del ladrido.
public String Interactuar()
{
    return "Guau";
}

public void imprimeInformacion()
{
    System.out.println("Soy un perro con " + edad + " años, y me llamo: " + nombre);
}
```

Instanciación de objetos

Por ahora ya tenemos nuestra clase Perro, misma que está lista para utilizarse; para tener orden vamos a crear una nueva clase que sirva de contenedor para el método *main*. Crea una nueva clase llamada

Programa.



El código para la instanciación quedaría de la siguiente manera en el *main*:

```
public class Programa
{
    public static void main(String[] args)
    {
        Perro miPerro = new Perro();
    }
}
```

Al ejecutarse el programa (*main*) vamos a instanciar un nuevo objeto del tipo Perro, es decir un objeto concreto a partir de la clase Perro que definimos, la sintaxis para esto es:

Modificador de acceso de la variable, por default (si no lo especificamos) es privado.

Tipo de dato: Perro, un tipo de dato que definimos.

Asignación, que es el símbolo de “=” seguido por el método constructor.

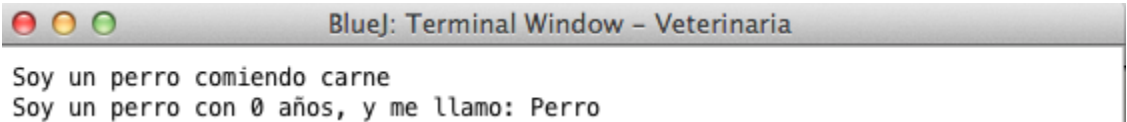
El método constructor es un método de la clase que sirve para crear nuevos objetos en a partir de una clase, y tiene el exacto mismo nombre que la clase. Si el programador no lo define, Java automáticamente asigna uno sin parámetros.

Una función esencial de los constructores, es la de asignar valores de inicio a los objetos para poner un estado inicial.

Veamos qué imprime la consola con el siguiente código:

```
public class Programa
{
    public static void main(String[] args)
    {
        Perro miPerro = new Perro();

        miPerro.alimentar(0);
        miPerro.interactuar();
        miPerro.imprimeInformacion();
    }
}
```



Vemos dos resultados impresos en consola, y en el caso del método *imprimeInformación*, nos dice que tiene 0 años y su nombre es Perro, pero sabemos que no todos los perros tendrán esa información, así que vamos a crear un constructor que reciba de parámetro su edad y nombre.

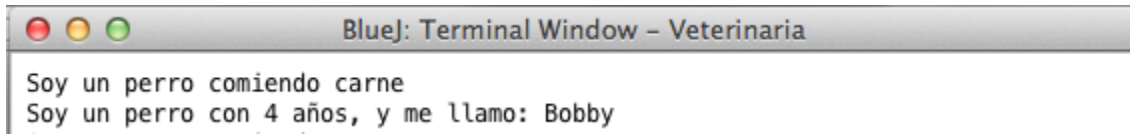
```
Perro(int pEdad, String pNombre, float pPeso)
{
    edad = pEdad;
    nombre = pNombre;
    peso = pPeso;
}
```

En nuestro constructor vamos a pedir tres parámetros, mismos que utilizaremos para inicializar el valor de nuestros objetos y que no tengan siempre el mismo valor. Así, cuando el veterinario da de alta un nuevo Perro, podrá asignar sus valores:

```
public class Programa
{
    public static void main(String[] args)
    {
        Perro miPerro = new Perro(4, "Bobby", 6.5f);

        miPerro.alimentar(0);
        miPerro.interactuar();
        miPerro.imprimeInformacion();
    }
}
```

El resultado en consola ahora queda:



```
BlueJ: Terminal Window - Veterinaria
Soy un perro comiendo carne
Soy un perro con 4 años, y me llamo: Bobby
```

Ahora vemos que con el constructor el objeto creado ya cambia su estado y el valor de sus atributos, por lo que los mensajes en consola cambian.

Packages

En el ejemplo anterior vimos que podemos hacer interactuar dos clases (objetos) en archivos diferentes, esto es porque dichas clases están declaradas en el mismo **package**(paquete).

Pero imagina que estás trabajando en una clase *Calculadora* para operaciones aritméticas con un método *Suma(int a, int b)* y por otro lado, alguien en tu equipo de desarrollo trabaja en otra clase llamada *Calculadora* con un método *Suma(int a, int b)*. Tu función además, de calcular el resultado también lo imprime a consola, la otra solamente calcula el resultado y lo regresa.

Usando packages, ambas clases y funciones podrán existir sin causar problemas, esto nos permite hacer el código modular y también reutilizable.

Para crear un paquete, simplemente al inicio de nuestro archivo se escribe la palabra clave `package` seguido del nombre del paquete.

```
package com.utel.matematicas;
```

Ahora, en el archivo que vamos a utilizar la clase importamos el paquete al inicio de la clase:

```
import com.utel.matematicas.*;
```

Consola de E/S

Desde la primera unidad hemos estado haciendo uso la consola de Java mediante BlueJ. Esto solamente para mostrar información(Salida) cuando utilizamos la función `System.out.println("");`

Vamos a modificar nuestra clase programa para pedir al usuario que ingrese el nombre de su perro:

```
//Importamos la clase Console de la libreria de JAVA
import java.io.Console;

public class Programa
{
    public static void main(String[] args)
    {
        Console console = System.console();

        System.out.println("Ingresa el nombre de tu perro");
        String nombre = console.readLine();

        Perro miPerro = new Perro(5, nombre, 6.5f);

        miPerro.alimentar(0);
        miPerro.interactuar();
        miPerro.imprimeInformacion();
    }
}
```

Referencias

Variables en JAVA: <http://www.codexion.com/tutorialesjava/java/javaOO/variables.html>

Tutorial BlueJ: <http://www.bluej.org/tutorial/tutorial-spanish-1.pdf>