Programación orientada a objetos

Clases y objetos

Elizabeth León Guzmán, Ph.D.
eleonguz@unal.edu.co

Jonatan Gómez Perdomo, Ph. D.
jgomezpe@unal.edu.co

Arles Rodríguez, Ph.D.
aerodriguezp@unal.edu.co

Camilo Cubides, Ph.D. (c)
eccubidesg@unal.edu.co

Carlos Andres Sierra, M.Sc.
casierrav@unal.edu.co

Research Group on Data Mining – Grupo de Investigación en Minería de Datos – (Midas)
Research Group on Artificial Life – Grupo de Investigación en Vida Artificial – (Alife)
Computer and System Department
Engineering School
Universidad Nacional de Colombia

Agenda

1. Introducción
2. Clases
   - Definición
   - Atributos y Métodos
   - Estructura básica
   - Constructor
3. Objetos
   - Definición
   - Instanciar un objeto
   - Mensajes
   - Múltiples Objetos
4. Problemas

Objetos (I)

Ejemplos

Imagen tomada de https://www.freepik.es/vector-gratis/muchacho-lindo-diversos-objetos-escuela_1175918.htm

Objetos (II)

Definición RAE

Objeto: Todo lo que puede ser materia de conocimiento o sensibilidad de parte del sujeto, incluso éste mismo.
https://dle.rae.es/objeto

MIDAS

Objetos (III)

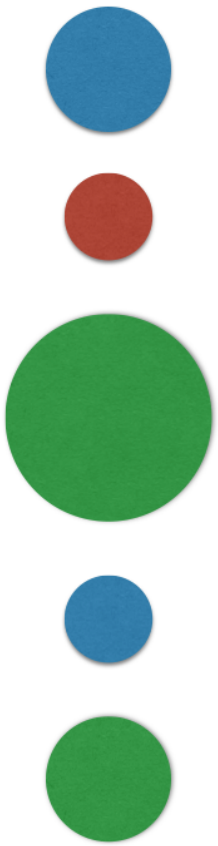
Ejemplos

Imagen tomada de https://www.freepik.es/vector-gratis/conjunto-objetos-escuela-personas-escuela_1046882.htm

Atributos (I)

Ejercicio

Volviendo a la escuela. Relacione objetos con sus atributos:



Color

V. Verde

A. Azul

R. Rojo

Tamaño

G. Grande

M. Mediano

P. Pequeño

Atributos (II)

Definición RAE

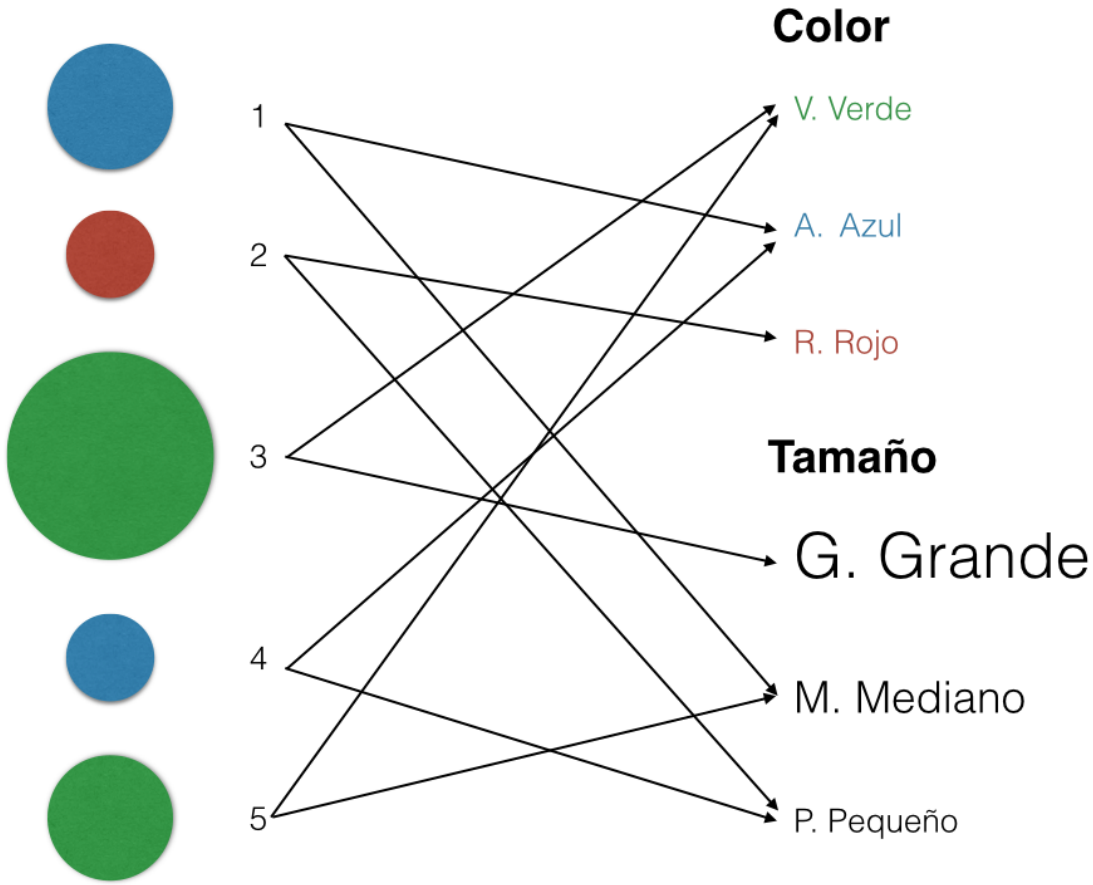
Atributo: Cada una de las cualidades o propiedades de un ser.
https://dle.rae.es/atributo

MIDAS

Atributos (III)

Ejemplo

Volviendo a la escuela. Objetos con sus atributos:



Comportamientos/Métodos (I)

Ejercicio

Volviendo a la escuela. Relacione objetos con sus comportamientos:

1

2

3

4

5

A. Cazar

B. Ronronear

C. Ladrar

D. Aullar

E. Maullar

F. Roer

Comportamientos/Métodos (II)

Definición RAE

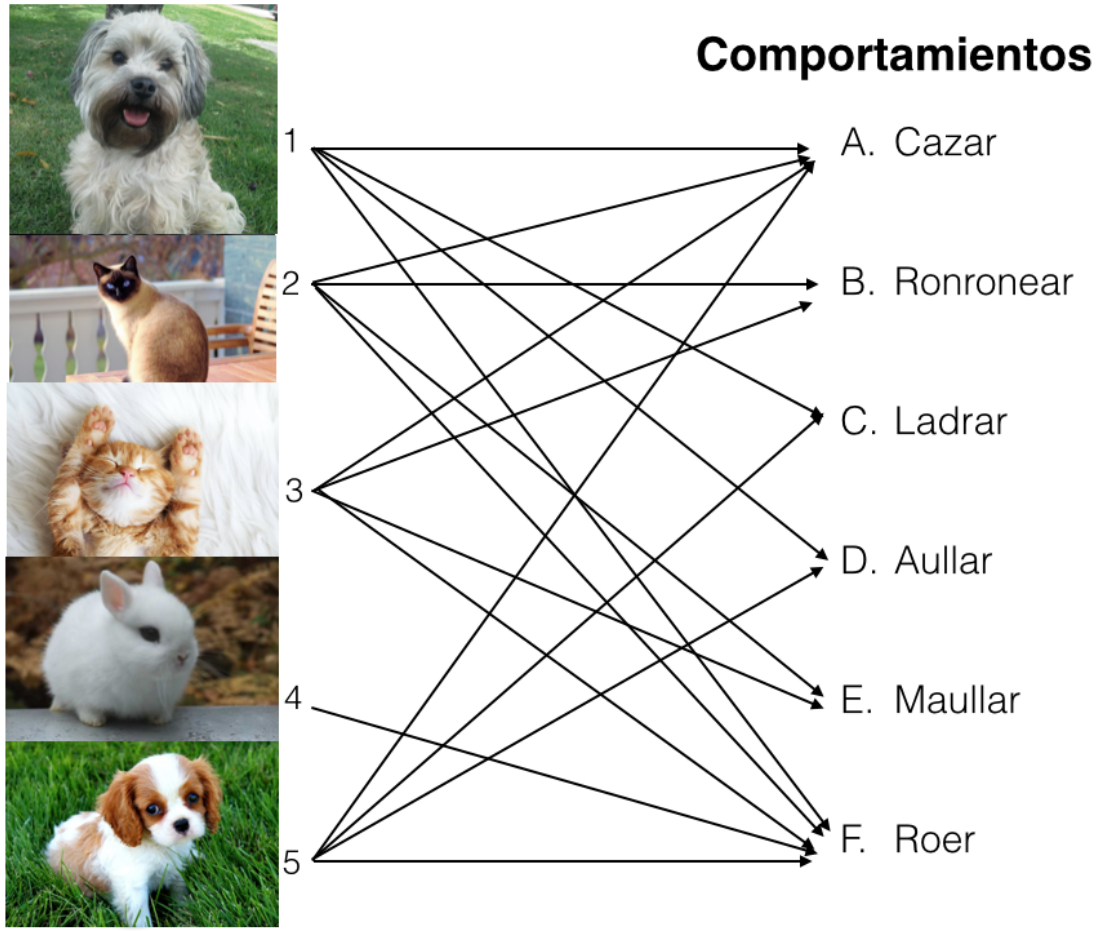
Comportamiento: Manera de comportarse.
https://dle.rae.es/comportamiento.

Comportar: Dicho de una cosa: Funcionar o actuar.
https://dle.rae.es/comportar

Comportamientos/Métodos (III)

Ejemplo

Volviendo a la escuela. Objetos con sus comportamientos:



Clases (I)

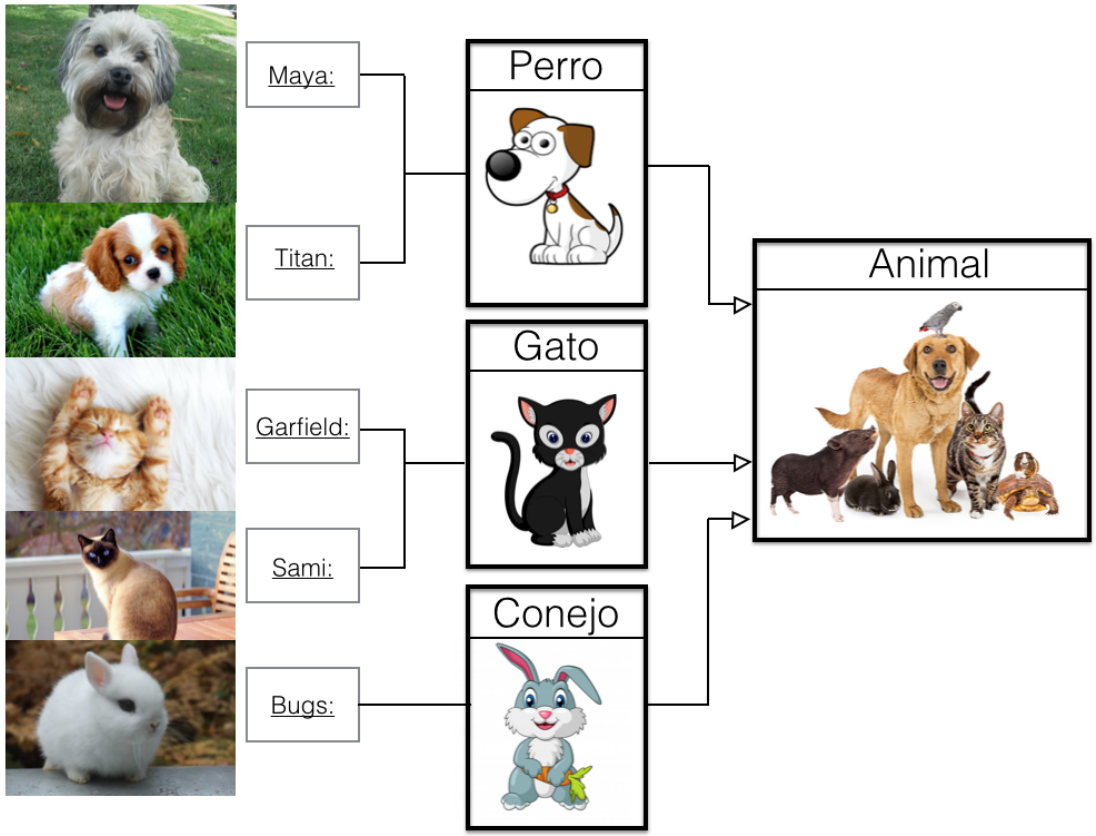
Definición RAE

Clase: Conjunto de elementos con carácteres comunes.
https://dle.rae.es/clase

Clases (II)

Ejemplo (I)

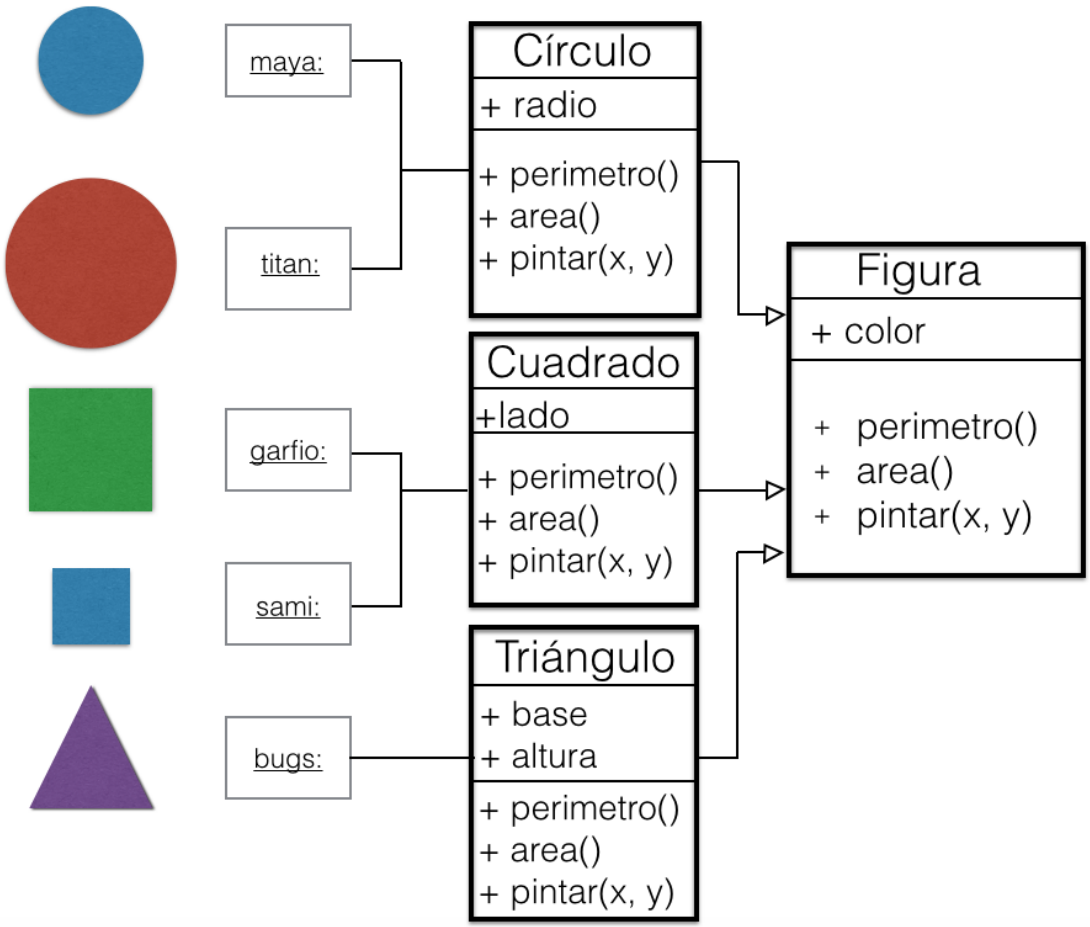
Volviendo a la escuela. Objetos con sus clases:



Clases (III)

Ejemplo (II)

Volviendo a la escuela. Objetos con sus clases:



Programación orientada a objetos POO (I)

Definición

La **Programación Orientada a Objetos** (POO) básicamente define una serie de conceptos y técnicas de programación para representar acciones o cosas de la vida real basada en objetos. Estos objetos se usan como instancias de clases y sus interacciones para diseñar aplicaciones y programas.

La programación orientada a objetos trabaja de esta manera: todo el programa está construido con base a diferentes componentes (objetos), cada uno tiene un rol específico en el programa y todos los componentes pueden comunicarse entre ellos de formas predefinidas.

Programación orientada a objetos POO (II)

Definición

- La POO vincula diferentes conceptos tales como clases, objetos, métodos, propiedades, estados, instancias, etc.
- La POO hace uso de técnicas tales como la abstracción, herencia, modularidad, polimorfismo y encapsulamiento.
- Usualmente los lenguajes orientados a objetos se usan en combinación con la programación imperativa. Ejemplos: Simula 67 (1967), Smalltalk (1972 a 1980), Ada, BASIC, Lisp, C++, Java (1991), Python, R, etc.

Agenda

1 Introducción

2 Clases
- Definición
- Atributos y Métodos
- Estructura básica
- Constructor

3 Objetos
- Definición
- Instanciar un objeto
- Mensajes
- Múltiples Objetos

4 Problemas

Agenda

1 Introducción

2 Clases
- Definición
- Atributos y Métodos
- Estructura básica
- Constructor

3 Objetos
- Definición
- Instanciar un objeto
- Mensajes
- Múltiples Objetos

4 Problemas

Definición

Clases

Las **clases** son los pilares fundamentales dentro de la POO. Consisten en la **representación abstracta** de cosas de la realidad. Se pueden pensar como plantillas que nos ayudan a modelar algo en específico.

Ejemplo

Pensemos en un carro. Podemos definir una plantilla "genérica" sobre la que todos los carros se basan para ser construidos.

La clase Perro

Ejemplo (Un Perro)

Podemos imaginar que todos los perros tienen los mismos **atributos** y, por tanto, definir una plantilla sobre la que todos los perros serán construidos.

MIDAS

Porqué usar clases

El objetivo principal de las clases es definir un conjunto de **atributos y métodos** que representen un modelo. Los atributos se definen como las **características** del modelo, mientras que los métodos se definen como su **comportamiento**.

Agenda

1 Introducción

2 Clases
- Definición
- Atributos y Métodos
- Estructura básica
- Constructor

3 Objetos
- Definición
- Instanciar un objeto
- Mensajes
- Múltiples Objetos

4 Problemas

Atributos

Los **atributos** de una clase son el conjunto de datos que representan adecuadamente el modelo que estamos definiendo. Estos datos son almacenados como variables. Comúnmente se atribuyen a aspectos como los *rasgos físicos*, *edad*, *nombres*, *características*, etc.

Ejemplo

En el ejemplo del `Perro` los atributos pueden ser:

- `Color de ojos`.
- `Altura`.
- `Peso`.
- `Nombre`.

Métodos

Los **métodos** de una clase son el conjunto de acciones que puede realizar nuestro modelo. Normalmente definen el comportamiento de los modelos y cómo interactúan entre sí.

Comúnmente se atribuyen a operaciones o actividades que nuestro modelo puede ejecutar, tales como *caminar*, *comer*, *saltar*, etc.

Ejemplo

En el ejemplo del `Perro` los métodos pueden ser:

- `Dormir`.
- `Comer`.
- `Jugar`.
- `Ladrar`.

Agenda

1 Introducción

2 Clases
- Definición
- Atributos y Métodos
- Estructura básica
- Constructor

3 Objetos
- Definición
- Instanciar un objeto
- Mensajes
- Múltiples Objetos

4 Problemas

Estructura básica

En Java, las clases se definen con la siguiente estructura:

```
public class Perro {
    // Atributos
    public int nombre;
    // Constructor
    public Perro(int nombre) {
        this.nombre = nombre;
    }
    // Métodos
    public void ladrar() {
        System.out.println("Guau guau");
    }
}
```

Y esas palabras reservadas ¿qué?

La palabra reservada `this`, hace referencia al objeto que está en construcción (en otros métodos será el objeto que está ejecutando la acción).

Las palabras reservadas `public`, `protected` y `private` hacen referencia al nivel de acceso a los métodos y atributos de un objeto (quien puede comunicarse con el objeto por medio del atributo o método). En este caso, `public` indica que cualquier otro objeto puede acceder al (comunicarse por medio del) atributo o método etiquetado público.

Agenda

1 Introducción

2 Clases
- Definición
- Atributos y Métodos
- Estructura básica
- Constructor

3 Objetos
- Definición
- Instanciar un objeto
- Mensajes
- Múltiples Objetos

4 Problemas

Constructor de una clase

El **constructor** de una clase es una función especial de cada clase que permite inicializar los atributos con algún valor en específico al momento de crear el objeto. Nos permite definir qué valores tomarán los atributos inicialmente al crear nuestro objeto.

Posee el mismo nombre que la clase, y recibe como entrada los valores que inicializarán el objeto.

```
public Perro(int nombre) {
    this.nombre = nombre;
}
```

MIDAS

Agenda

1. Introducción
2. Clases
   - Definición
   - Atributos y Métodos
   - Estructura básica
   - Constructor
3. Objetos
   - Definición
   - Instanciar un objeto
   - Mensajes
   - Múltiples Objetos
4. Problemas

Agenda

1 Introducción

2 Clases
- Definición
- Atributos y Métodos
- Estructura básica
- Constructor

3 Objetos
- Definición
- Instanciar un objeto
- Mensajes
- Múltiples Objetos

4 Problemas

Definición

Un **objeto** es una **instancia** de una clase. Una vez se haya construido una clase, se pueden crear objetos a partir de dicha clase. Se pueden crear múltiples instancias, todas basadas en la misma clase, y por lo tanto, todas tendrán la misma estructura.

Debemos pensar que los objetos son *"seres vivos"* dentro de nuestro programa, seres a los que les podemos pedir (les enviamos un mensaje) que hagan cosas (se comporten, ejecuten un método) de acuerdo a su estado (valores de sus atributos).

Ejemplo

Podemos crear un perro llamado "`Toby`" a partir de la clase `Perro`.

Aclaración

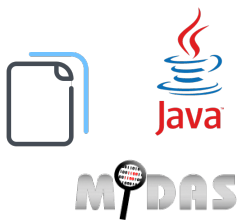
Es importante mencionar que una **Clase** y un **Objeto** no se refieren a lo mismo. Una clase es la plantilla (o estructura) que se define para todos; mientras que un objeto es usar dicha plantilla para instanciar un ente en particular (que tendrá características particulares para ese objeto).

MIDAS

Clase Perro I

Se definió la siguiente clase Perro:

```
public class Perro {
    // Atributos
    public int edad;
    public String nombre;
    public String colorOjos;
    // Constructor
    Perro(int edad, String nombre, String colorOjos) {
        this.edad = edad;
        this.nombre = nombre;
        this.colorOjos = colorOjos;
    }
```



Clase Perro II

Se definió la siguiente clase Perro (cont..):

```
    // Métodos
    public void ladrar() {
        System.out.println("Guau Guau");
    }

    public void saludar() {
        System.out.println("Hola, mi nombre es " + nombre);
    }
}
```

Agenda

1 Introducción

2 Clases
- Definición
- Atributos y Métodos
- Estructura básica
- Constructor

3 Objetos
- Definición
- Instanciar un objeto
- Mensajes
- Múltiples Objetos

4 Problemas

Instanciar un objeto

Ejemplo

Usando la clase anterior, se quiere instanciar un objeto a partir de ella. La manera de hacerlo en Java, por ejemplo en el método principal es la siguiente:

```
public static void main(String[] args) {
    Perro miPerro = new Perro(5, "Toby", "Azul");
}
```

Donde `Perro` es la clase que se definió anteriormente, y `miPerro` es el objeto que se acaba de instanciar.

Agenda

1 Introducción

2 Clases
- Definición
- Atributos y Métodos
- Estructura básica
- Constructor

3 Objetos
- Definición
- Instanciar un objeto
- Mensajes
- Múltiples Objetos

4 Problemas

Comunicación

Los objetos en un programa se comunican entre ellos para resolver un problema. La forma de comunicarse con un objeto es usar el nombre que tiene el objeto en el programa, seguido de un punto (.) y el mensaje que queremos enviarle:

```
NombreObjeto.Atributo
NombreObjeto.NombreMetodo(parametros)
```

Recordemos, depende del nivel de acceso del atributo o método:

`public`: Cualquier otro objeto puede acceder al atributo o método.

`protected`: Sólo objetos de la misma clase, subclase, o del mismo paquete (package) pueden acceder al atributo o método.

`private`: Sólo objetos de la misma clase pueden acceder al atributo o método.

Acceder a los atributos I

Cuando se define un objeto, se puede acceder a sus atributos individualmente con la siguiente estructura

```
NombreObjeto.NombreAtributo
```

Donde `NombreObjeto` es el nombre del objeto y `NombreAtributo` es el nombre del atributo al que se desea acceder.

MIDAS

Acceder a los atributos II

Ejemplo

Se desea acceder a la edad de `miPerro` usando su atributo:

```
public static void main(String[] args) {
    Perro miPerro = new Perro(5, "Toby", "Blue");
    System.out.println(miPerro.edad);
}
```

Acceder a los métodos I

De la misma manera, se puede hacer uso de cada uno de sus métodos con la siguiente estructura:

```
NombreObjeto.NombreMetodo(parametros)
```

Donde `NombreObjeto` es el nombre del objeto y `NombreMetodo` es el nombre del método al que se desea acceder. Es importante pasarle los parámetros necesarios que tenga definidos el método.

MIDAS

Acceder a los métodos II

Ejemplo

Podemos usar el método `saludar()` del objeto `miPerro`

```
public static void main(String[] args) {
    Perro miPerro = new Perro(5, "Toby", "Azul");
    miPerro.saludar();
}
```

Agenda

1 Introducción

2 Clases
- Definición
- Atributos y Métodos
- Estructura básica
- Constructor

3 Objetos
- Definición
- Instanciar un objeto
- Mensajes
- Múltiples Objetos

4 Problemas

Múltiples Objetos I

Podemos usar varios objetos en un mismo programa, haciendo referencia al objeto deseado por el nombre que le dimos al momento de crearlo. Es decir, si tenemos los objetos `miPerro1`, `miPerro2` y `miPerro3`, podemos acceder a la edad de `miPerro2`, mediante las siguientes instrucciones

```
public static void main(String[] args) {
    Perro miPerro1 = new Perro(5, "Toby", "Azul");
    Perro miPerro2 = new Perro(7, "Vainilla", "Negro");
    Perro miPerro3 = new Perro(5, "Bony", "Amarillo");

    System.out.println(miPerro2.edad);
}
```

Múltiples Objetos II

Una de las ventajas de trabajar con múltiples objetos es la posibilidad de usarlos para **interactuar** entre sí. Esta interacción puede darse de múltiples formas, tales como interacciones entre atributos, métodos o clases.

Múltiples Objetos III

Para facilitar la interacción entre objetos, es posible recibir como parámetro de entrada de un método una instancia de una clase.

```
public class Perro {
    // Métodos
    public void quienEsMayor(Perro otroPerro) {

    }
}
```

Múltiples Objetos IV

Esa característica de POO nos permite definir interacciones personalizadas, usando los atributos y los métodos de cada uno de los objetos para algo en específico.

Ejemplo

Se pretende crear un método que compare las edades de dos perros. Dicho método comparará la edad de un perro en específico que invoca, contra la edad de otro perro cualquiera.

Múltiples Objetos V

Ejemplo

Un ejemplo del método anteriormente definido es el siguiente

```
public void quienEsMayor(Perro otroPerro) {
  if (this.edad > otroPerro.edad) {
    System.out.println("Soy mayor que "+otroPerro.nombre);
  } else if (this.edad == otroPerro.edad) {
    System.out.println("Tenemos la misma edad");
  } else {
    System.out.println(otroPerro.nombre+" es mayor que yo");
  }
}
```

Múltiples Objetos VI

Ejemplo

El método anteriormente visto está asociado a la clase `Perro`. Esto permite usarlo de la siguiente manera

```
public static void main(String[] args) {
    Perro miPerro1 = new Perro(5, "Toby", "Azul");
    Perro miPerro2 = new Perro(7, "Vainilla", "Negro");
    Perro miPerro3 = new Perro(5, "Bony", "Amarillo");

    miPerro1.quienEsMayor(miPerro2);
}
```

Múltiples Objetos VII

Se puede definir un saludo personalizado que tenga en cuenta el estado (valor de los atributos) del objeto.

```
public class Perro {
    // Métodos
    public void saludarOtroPerro(Perro otroPerro) {
        System.out.println
            ("Hola " + otroPerro.nombre + ", yo soy " +
            this.nombre);
    }
}
```

Múltiples Objetos VIII

Ejemplo

Luego se puede usar dicho método de la siguiente manera:

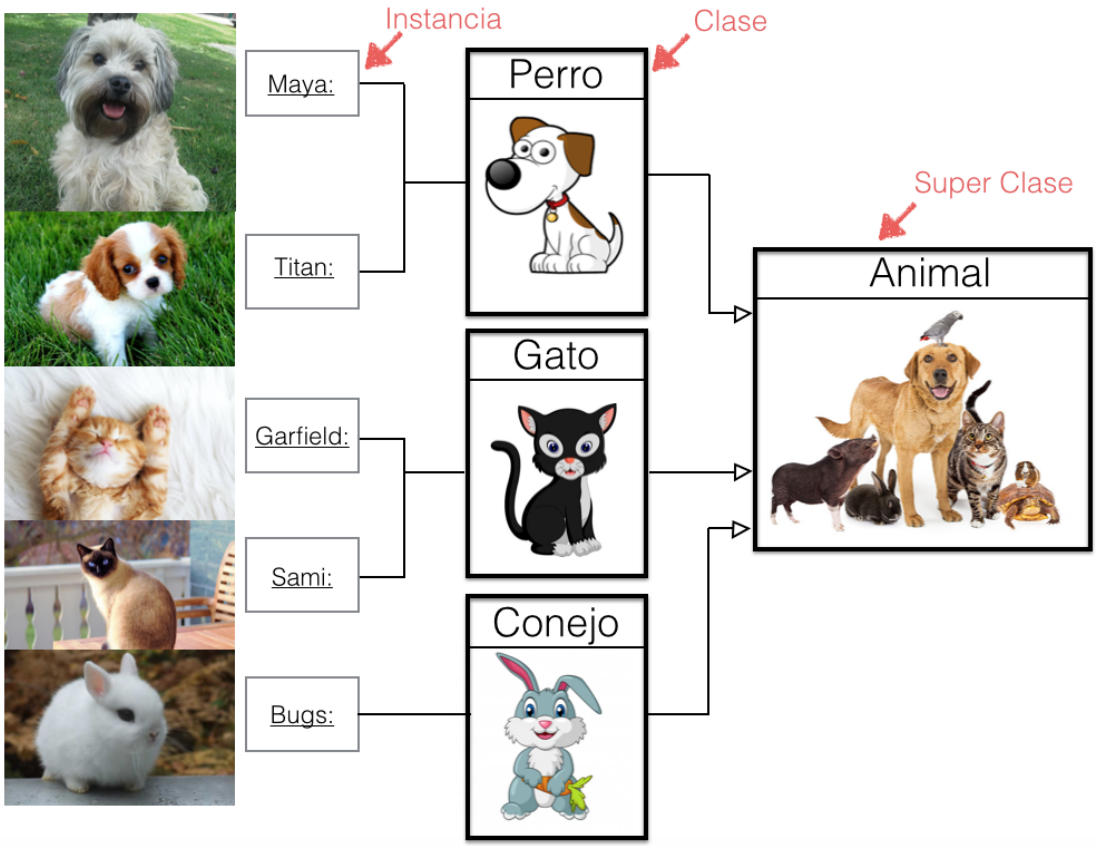
```
public static void main(String[] args) {
    Perro miPerro1 = new Perro(5, "Toby", "Azul");
    Perro miPerro2 = new Perro(7, "Vainilla", "Negro");
    Perro miPerro3 = new Perro(5, "Bony", "Amarillo");

    miPerro1.saludarOtroPerro(miPerro2);
}
```

Clases (IV)

Ejemplo (III)

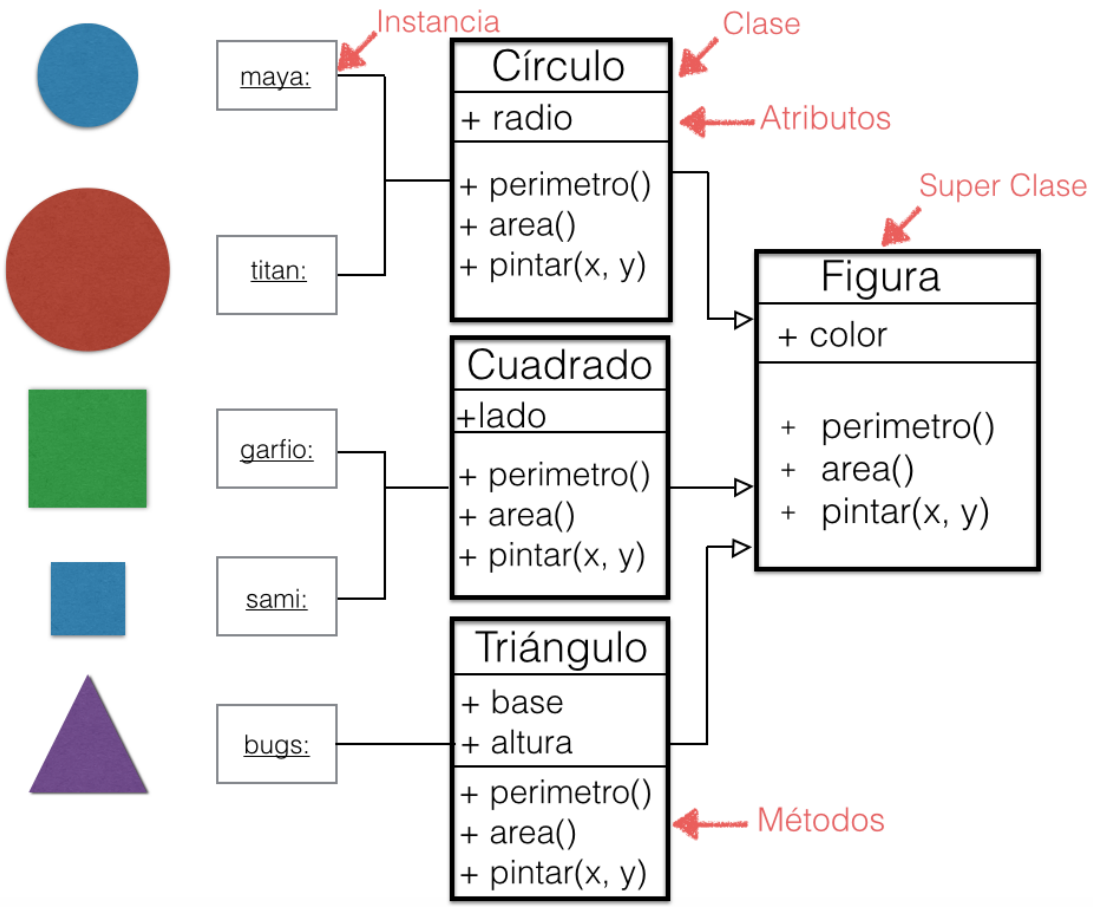
Volviendo a la escuela con POO. Objetos con sus clases:



Clases (V)

Ejemplo (IV)

Volviendo a la escuela con POO. Objetos con sus clases:



Agenda

1 Introducción

2 Clases
- Definición
- Atributos y Métodos
- Estructura básica
- Constructor

3 Objetos
- Definición
- Instanciar un objeto
- Mensajes
- Múltiples Objetos

4 Problemas

Problemas I

Problemas

1. Crear la clase `Persona` que tenga como atributos `Nombre`, `Edad`, `Genero` (`H` hombre, `M` mujer), `peso` y `altura`. El constructor debe recibir todos los parámetros para su inicialización.
   La clase debe tener los siguientes métodos:

   `calcularIMC()`: Calculará y retornará el Indice de Masa Corporal del objeto.

   `esMayorDeEdad()`: Devolverá `true` si es mayor de edad, `false` en caso contrario.

   `toString()`: Devolverá toda la información del objeto en un `String`.

MIDAS

Problemas II

Problemas (continuación)

2. Crear la clase `Contraseña` que tenga como atributos `Longitud` (por defecto será 8) y `Contraseña` (`String`). El constructor debe recibir la longitud y generar una contraseña aleatoria con dicha longitud. La clase debe tener los siguientes métodos:

   `esFuerte()`: Devuelve un booleano si es fuerte o no. Una contraseña se considera fuerte si tiene al menos una mayúscula, una minúscula, y más de cinco números.

   `cambiarContraseña(String nuevaContraseña)`: Cambiará la contraseña actual por la indicada en los parámetros.

   `toString()`: Devolverá toda la información del objeto en un `String`.

MIDAS

Problemas III

Triki I

Problemas (continuación)

3. El objetivo será crear una clase `Triki` con el que se pueda jugar. Considere la siguiente matriz

a	a	a
a	a	a
a	a	a

Cree la clase `Triki` que tenga como único atributo la matriz anteriormente contemplada.

El constructor no recibirá parámetros de entrada, y allí se inicializará la matriz con a's que indican que la casilla en cuestión está vacía.

Problemas IV

Triki II

Problemas (continuación)

La clase tendrá los siguientes métodos:

`marcarCasilla(String simbolo, int fila, int columna)`: Marcará el símbolo indicado (lo ideal es que en el parámetro `simbolo` se pase un único caracter (`X` ó `O`). Dicho caracter se escribirá en la posición de la matriz (fila, columna).

`verificarGanador()`: Verificará si hay un ganador. Retornará el ganador (si lo hay) del juego, devolviendo el caracter que usó para ganar. Si no hay ganador aún se devolverá una "a" indicando que no hay un ganador.

`verificarCasilla(int fila, int columna)`: Devolverá el caracter que hay en la casilla que está en la posición (`fila`, `columna`) de la matriz.

Problemas V

PacMan I

Problemas (continuación)

4. Considere el videojuego PacMan (1980). Se creará una simulación del personaje y el tablero. Para ello es necesario crear dos clases:
   - Clase `PacMan`: Tendrá la información del personaje.
   - Clase `Tablero`: Tendrá la información del tablero.

Problemas VI

PacMan II

Problemas (continuación)

Para la clase `PacMan` se deben definir como atributos el `color` (por defecto "`Amarillo`"), la `puntuacion` (por defecto `0`) y las `vidasRestantes` (por defecto `3`). El constructor debe recibir estos tres parámetros e inicializarlos.

Dicha clase tendrá los siguientes métodos:

`sumarPuntuacion()`: Cada vez que se llame este método, se sumará `+1` a la puntuación.

`obtenerPuntuación()`: Retornará la puntuación actual del personaje.

`restarVida()`: Cada vez que se llame este método, se restará `-1` a las vidas.

`sigueVivo()`: Devolverá un booleano indicando si aún le quedan vidas para seguir jugando.

Problemas VII

PacMan III

Problemas (continuación)

Para la clase `Tablero` se requerirán dos atributos: el `personaje` (será un objeto de tipo `PacMan`) y el `nivel`. El constructor debe recibir el `personaje` (el cual debe haber sido creado anteriormente y pasarse como un parámetro en el constructor de la clase) e inicializar el `nivel` en `0`.

Dicha clase tendrá el siguiente método:

`comprobarNivelActual()`: El funcionamiento es simple: cada 25 puntos del personaje equivalen a 1 nivel en el `tablero`. Este método obtendrá la puntuación del `PacMan` y fijará el nivel actual dependiendo de la cantidad de puntos que haya encontrado. (Por ejemplo, 57 puntos equivalen al nivel 2, 75 puntos al nivel 3, etc).