Tema 3

PARADIGMA DE PROGRAMACIÓN ORIENTADA A OBJETOS

1. INTRODUCCIÓN

- La programación orientada a objetos es un paradigma de programación totalmente diferente al método clásico de programación, el cual utiliza objetos y su comportamiento para resolver problemas y generar programas y aplicaciones informáticas.
- Con la POO aumenta la modularidad de los programas y la reutilización de los mismos.
- Además, se diferencia de la programación clásica porque utiliza técnicas nuevas como el **polimorfismo**, el **encapsulamiento**, la **herencia**, etc.

2. INTRODUCCIÓN AL CONCEPTO DE OBJETO

- Los programas realizados mediante el paradigma de la POO solamente tienen **objetos**.
- Todos los objetos pertenecen a una **clase**, por ejemplo, mi loro Felipe pertenecería a la clase pájaro.
- Todos los objetos de la clase pájaro se identificarán, entre otros atributos, con un nombre, un color de plumaje, una edad y si son domésticos o no.

3. CLASES

- Las clases son los moldes de los cuales se generan los objetos.
- Los objetos se instancian y se generan.
- Instancia=Objeto.
- Por ejemplo, Felipe será un objeto concreto de la clase pájaro.

- *En un símil con la costura, las clases son los patrones y los objetos son las prendas.*

3. CLASES

- Cuando se escribe un programa o aplicación OO, lo que se hace es definir las clases de objetos dotándolas de estado y comportamiento y cuando se ejecute el programa se crearán los objetos.

- Cuando se programa, las clases se escriben en ficheros ASCII con el mismo nombre que la clase y extensión .java.

3. CLASES

- Las clases tienen una estructura parecida a la siguiente:

```
[algo_1] class nombre_clase [algo_2] {
        [Atributos]
        [Métodos]
}
```

- En [] se han etiquetado los elementos opcionales de la clase.
- [algo_1] y [algo_2] contendrán palabras reservadas que se estudiarán en próximos capítulos.

3. CLASES

- Es muy común ver definiciones de clases como:

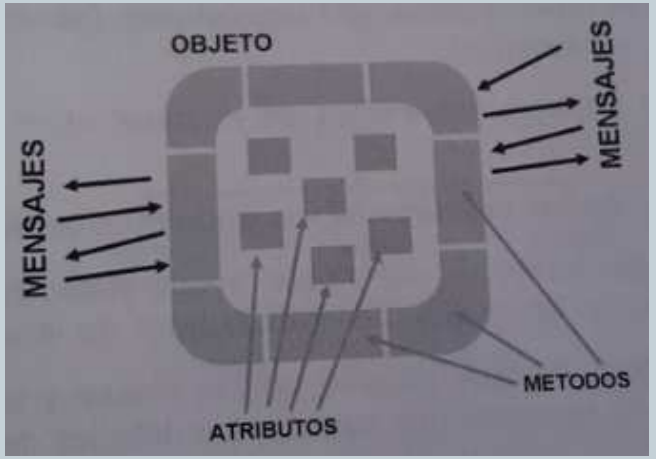
  public class nombre_clase
- La palabra reservada public indica que la clase puede ser accedida por cualquier clase que necesite de su utilización.
- Los atributos dentro de una clase pueden ser desde cero a muchos.
- Una clase puede tener cero o muchos métodos y se corresponden con los procedimientos o funciones de otros lenguajes de programación.

4. OBJETOS

- Un objeto tiene una serie de características:
  - **Identidad:** cada objeto es único y diferente de otro objeto. Ejemplo: Mi loro Felipe es diferente a otros loros.
  - **Estado:** el estado serán los valores de los atributos del objeto, en el caso de los objetos de clase pájaro serían nombre, color, edad, doméstico, etc.
  - **Comportamiento:** el comportamiento serían los métodos o procedimientos que realiza dicho objeto. Dependiendo de tipo o clase de objeto, éstos realizarán unas operaciones u otras (volar, cantar, hablar, etc)

4. OBJETOS

- Los programas o aplicaciones OO están compuestas por objetos, los cuales interactúan unos con otros a través del paso de **mensajes**.
- Cuando un objeto recibe un mensaje, lo que hace es ejecutar el método asociado.
- Por lo tanto, los métodos son los procedimientos que ejecuta el objeto cuando recibe un mensaje vinculado a ese método concreto.



4. OBJETOS

- En un programa OO primero se crean los objetos y entre ellos se envían mensajes procesándose la información para luego destruirse y liberar la memoria que estaban ocupando.

4.1 PROPIEDADES DE UN OBJETO

- Datos = Propiedades = Atributos.
- Referencian a las variables de una clase.
- Pueden ser tipos primitivos (char, int, boolean, etc) o bien pueden ser objetos de otra clase.
- Por ejemplo, un objeto de la clase coche puede tener un objeto de clase motor.

4.1 PROPIEDADES DE UN OBJETO

- Ejemplo:

```
class pajaro
{   //*** atributos o propiedades ****
    private char color; //propiedad o atributo color
    private int edad; //propiedad o atributo edad
    //*** métodos de la clase ****
    public void setedad(int e){edad = e;}
    public void printedad(){System.out.println(edad);}
    public void setcolor(char c){color=c;}
    public void printcolor(){
          switch(color){
          //Los pájaros son verdes, amarillos, grises, negros o blancos
          //No existen pájaros de otros colores
                case 'v`: System.out.println("verde");break;
                case 'a`: System.out.println("amarillo");break;
                case 'g`: System.out.println("gris");break;

               case 'n`: System.out.println("negro");break;
               case 'b`: System.out.println("blanco");break;
               default: System.out.println("color no establecido");
        }
}
 }
```

4.1 PROPIEDADES DE UN OBJETO

- Ejemplo:

```
class test
{
    public static void main(String[] args) {
            pajaro p;
            p=new pajaro();
            p.setedad(5);
            p.printedad();
    }
}
```

4.1 PROPIEDADES DE UN OBJETO

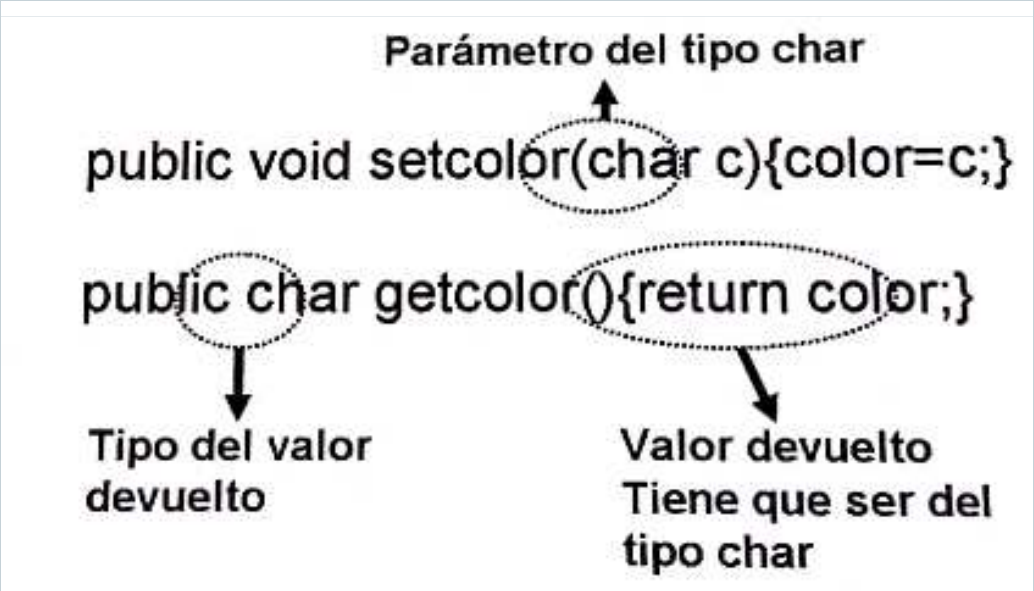
- En el código anterior existen dos clases diferentes: *pájaro* y *test* las cuales estarán en dos ficheros diferentes (pajaro.java y test.java).
- En la clase test se crea un objeto de la clase pájaro y se llama a los métodos para actualizar la edad y mostrarla por pantalla.
- En ningún momento la clase test puede acceder a los métodos o atributos **private** de la clase pájaro, sólo puede acceder a los métodos públicos (public) de la clase → **Abstracción**.

5. PARÁMETROS Y VALORES DEVUELTOS

- Los métodos pueden permitir que se los llame especificando una serie de valores.
- A estos valores se les denomina **parámetros**.
- Los parámetros pueden tener un tipo básico (char, int, boolean, etc) o bien ser un objeto.
- Además, los métodos (excepto el constructor) pueden retornar un valor o no (en este caso se pone **void**)

5. PARÁMETROS Y VALORES DEVUELTOS

- Ejemplo:
  - Dos métodos para la clase pájaro: setcolor que admite un parámetro y getcolor que devuleve un valor de tipo char.



6. CONSTRUCTORES Y DESTRUCTORES DE OBJETOS

- En Java, existen unos métodos especiales que son los constructores y destructores del objeto.
- Estos métodos son opcionales, es decir, no es obligatorio programarlos salvo que los necesites.
- El **constructor** del objeto es un procedimiento llamado automáticamente cuando se crea un objeto de esa clase. Si el programador no los declara, Java genera uno por defecto.
- La función del constructor es inicializar el objeto.

6. CONSTRUCTORES Y DESTRUCTORES DE OBJETOS

- El **destructor** del objeto se ejecutará automáticamente siempre que se destruye un objeto de dicha clase.
- Los destructores, generalmente, se utilizan para liberar recursos y cerrar flujos abiertos (realiza una limpieza final).
- Los destructores no reciben parámetros.
- En Java **NO** hay métodos destructores como en C++.

6. CONSTRUCTORES Y DESTRUCTORES DE OBJETOS

- Ejemplo de constructores para la clase pájaro:

```
class pajaro
{   //*** atributos o propiedades ****
    private char color; //propiedad o atributo color
    private int edad; //propiedad o atributo edad
    //*** métodos de la clase ****
    pajaro(){color = 'v'; edad = 0;} //constructor de la clase pájaro
    pajaro(char c, int e){color = c; edad = e;} // constructor de la clase pájaro
    /* Aquí irán los demás métodos de la clase */
    public static void main(String[] args) { //método main
          pajaro p1,p2;
          p1=new pajaro();
          p2=new pajaro('a',3);
    }
}
```

- El constructor de la clase pájaro está sobrecargado ya que se pueden crear objetos de la clase pájaro de distintas formas

7. MÉTODOS ESTÁTICOS Y DINÁMICOS

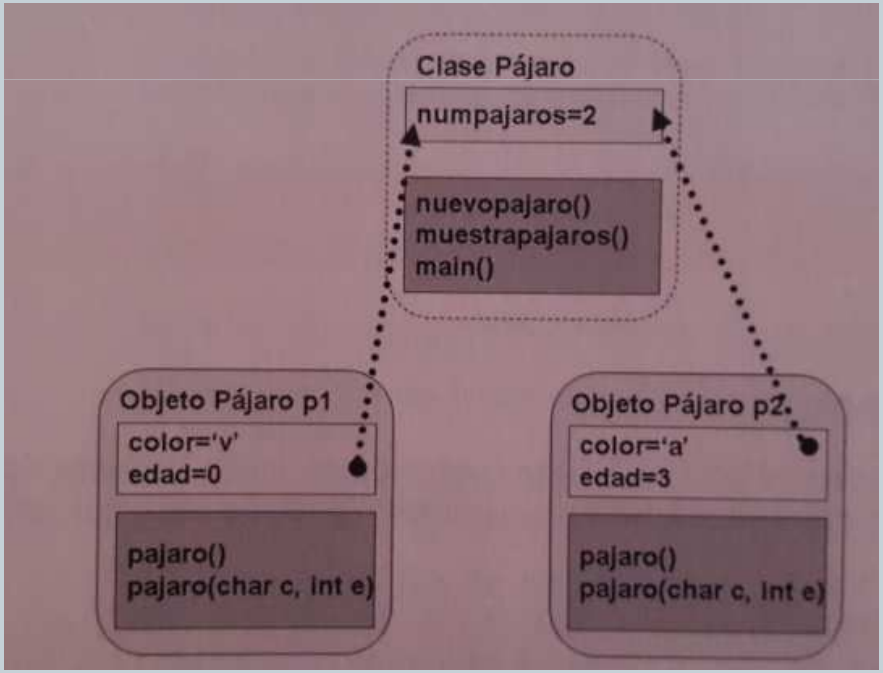
- Cuando un método o atributo se define como **static**, quiere decir que se va a crear para esa clase solo una instancia de ese método o atributo.
- En el siguiente ejemplo, se ve como se ha creado un atributo numpajaros que contará el número de pájaros que se van generando.
- Si ese atributo fuese no fuese estático, sería imposible contar los pájaros, puesto que en cada instancia del objeto se crearía una variable numpajaros.
- De la misma manera, los métodos *nuevopajaro()*, *muestrapajaro()* y *main()* son estáticos.

7. MÉTODOS ESTÁTICOS Y DINÁMICOS

```
package pajaro;

public class Pajaro {

    private static int numpajaros=0;
    private char color;
    private int edad;

    public Pajaro(){
        color='v';
        edad=0;
        nuevoPajaro();
    }

    public Pajaro(char c, int e){
        color=c;
        edad=e;
        nuevoPajaro();
    }

    static void nuevoPajaro(){
        numpajaros++;
    }

    static void muestraPajaros(){
        System.out.println(numpajaros);
    }

    public static void main(String[] args) {
        Pajaro p1, p2;
        p1=new Pajaro();
        p2= new Pajaro('a',3);

        Pajaro.muestraPajaros();
    }
}
```

7. MÉTODOS ESTÁTICOS Y DINÁMICOS

El atributo numpajaros y los métodos nuevopajaro(), muestrapajaro() y main() se comparten por todos los objetos creados de la clase pájaro.



8. LIBRERÍAS DE OBJETOS (PAQUETES)

- Un paquete o package es un conjunto de clases relacionadas entre sí.
- Gracias a los paquetes es posible organizar las clases en grupos.
- Para poder utilizar una clase contenida en un paquete es necesario utilizar la sentencia **import**.
- Es posible importar una clase individual. Ejemplo:

import java.lang.System; //Se importa la clase System

8. LIBRERÍAS DE OBJETOS (PAQUETES)

- O importar todas las clases de un paquete. Ejemplo:

import java.awt.;* //Se importa la clase System

...

Frame fr = new Frame("Ejemplo");

- Tambien es posible utilizar la clase sin utilizar la sentencia import:

java.awt.Frame fr = new java.awt.Frame("Ejemplo")

9. Normas de codificación

- http://javafoundations.blogspot.com/2010/07/java-estandares-de-programacion.html
- https://amap.cantabria.es/amap/bin/view/AMAP/CodificacionJava