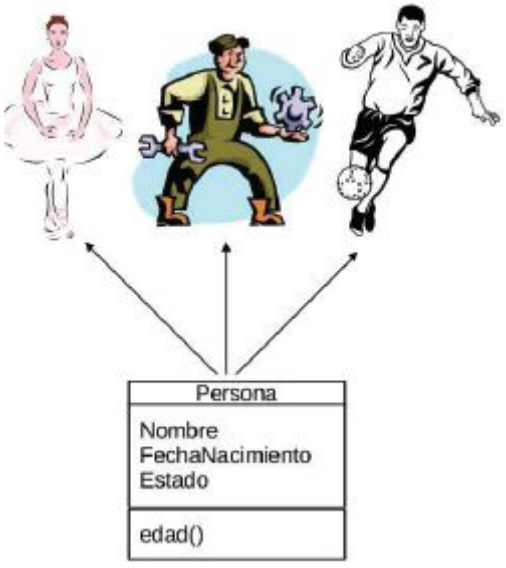
U7 – INTRODUCCIÓN A LA ORIENTACIÓN A OBJETOS.

7.1. INTRODUCCIÓN A POO.

La unidad fundamental de programación en Java es la clase, donde un programa está formado por un conjunto de clases.

Pero, ¿Qué es una clase?



Una clase es una "plantilla" que describe un conjunto de objetos con atributos y comportamientos similares o comunes a un cierto tipo de instancias. En sí es una entidad autónoma y autocontenida.

Una vez tenemos definida una clase podremos "instanciar" objetos de dicha clase en nuestro programa.

Pero, ¿Qué es un objeto?

En Java, un objeto es básicamente una instancia de una clase.

Es un ejemplar de una clase que utilizaremos en nuestro algoritmo y será independiente de otro objeto, aunque sea de la misma clase y tenga un comportamiento similar, representa otro objeto.

Ahora bien**, ¿Qué entendemos programación orientada a objetos?**

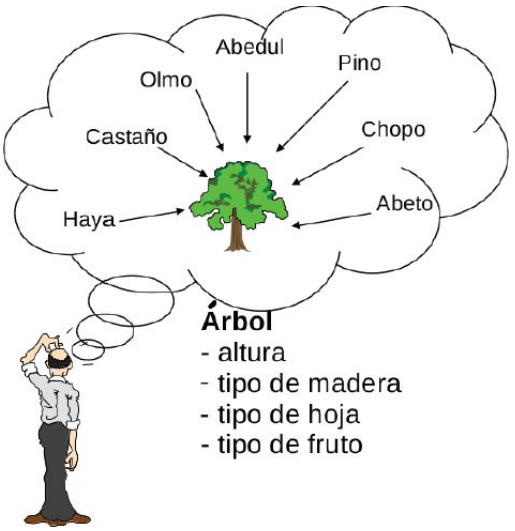
La programación orientada a objetos o **POO**, trata de descomponer un problema en un pequeño número de entidades o "clases" relacionadas pero independientes de forma que cada entidad puede ser utilizada por separado.

Se basa en la idea general que consiste en la supresión intencionada u ocultamiento de algunos detalles de un proceso con la finalidad de destacar de manera más clara otros aspectos, detalles o estructuras. Buscando la capacidad de centrarse en las características esenciales: **Divide y vencerás** (dividir la información en componentes aislados pero asociados a un concepto).

Así pues, si una entidad es demasiado compleja se podría aplicar nuevamente el principio de descomposición creando entidades más simples, fáciles de manejar y que agrupen/definan un tipo de instancias u objetos.

EJEMPLO: Trabajamos en un vivero y queremos catalogar y definir cada uno de los árboles en la tienda.

Ante el problema de querer definir un pino, olmo, abeto.. se observa que todos ellos se pueden definir con unos atributos comunes y podrían tener un comportamiento o métodos comunes, por lo que podríamos crear una clase genérica llamada "Árbol". Utilizaremos la clase árbol como una plantilla donde se define sus características principales, modelando todos los árboles deseados con la misma clase.



7.2. BENEFICIOS DE POO.

POO es una nueva forma de pensar acerca del software que se basa en abstracciones del mundo real.

La POO ofrece rendimiento, flexibilidad y funcionalidad para implementaciones prácticas y que permite la reutilización de componentes de software. Beneficios:

- Beneficios por la reutilización.
    - Reduce el tiempo de desarrollo = mejora productividad.
    - Aumenta fiabilidad y eficiencia, minimizando el esfuerzo requerido en el mantenimiento.
- Desarrollo rápido de aplicaciones
- Aplicaciones efectividad y eficiencia, pudiendo aislar el funcionamiento o puntos de fallo.
- Portabilidad. Clase utilizable en diversos proyectos.
- Reduce costes y facilita el mantenimiento.

7.3. ¿CÓMO CREAR UNA CLASE?

Las clases son un tipo de datos definido mediante una estructura o atributos concretos y unas operaciones que se pueden utilizar sobre esos atributos.

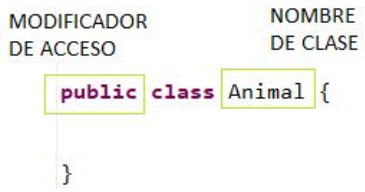
La clase es la que nos dice los componentes del ejemplar que vamos a crear, es decir, una clase contiene los atributos y los métodos que conformarán al ejemplar o instancias, de este modo al momento de crear una clase en Java, debemos especificar el tipo y el nombre (como mínimo) de los atributos y adicionalmente debemos especificar (si existen) los métodos o funciones, el tipo de dato que retornan, el nombre y los parámetros que reciben dichos métodos.

7.3.1. PARTES DE UN CLASE:

- Cabecera.
- Campos o atributos:
    - Variables
    - Contantes.
- Métodos:
    - Funciones. Acciones que permite.
    - Constructores. Se utiliza para inicializar nuevos objetos.

```java
public class Animal {
    private String raza;
    private String nombre;
    private int edad;
    private boolean tieneChip;


    public Animal() {
        System.out.println("Nuevo animal ingresado sin información ingresada. Usamos datos por defecto.");
        this.raza="None";
        this.nombre="None";
        this.edad = 0;
        this.tieneChip = false;
    }

    public Animal(String nombre, String raza, int edad, boolean tieneChip) {
        System.out.println("Nuevo animal ingresado con toda la información necesaria");
        this.raza = raza;
        this.nombre = nombre;
        this.edad = edad;
        this.tieneChip = tieneChip;
    }

    public String getRaza() {
        return raza;
    }

    public void setRaza(String raza) {
        this.raza = raza;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public boolean isTieneChip() {
        return tieneChip;
    }

    public void setTieneChip(boolean tieneChip) {
        this.tieneChip = tieneChip;
    }

}
```

1. **CABECERA:**



2. **ATRIBUTOS:**

```
private String raza;
private String nombre;
private int edad;
private boolean tieneChip;
```

3. **MÉTODOS:**

3.1. CONSTRUCTOR / ES

```
public Animal() {
    System.out.println("Nuevo animal ingresado sin información ingresada. Usamos datos por defecto.");
    this.raza="None";
    this.nombre="None";
    this.edad = 0;
    this.tieneChip = false;
}

public Animal(String nombre, String raza, int edad, boolean tieneChip) {
    System.out.println("Nuevo animal ingresado con toda la información necesaria");
    this.raza = raza;
    this.nombre = nombre;
    this.edad = edad;
    this.tieneChip = tieneChip;
}
```

3.2. FUNCIONES.

```java
public String getRaza() {
    return raza;
}

public void setRaza(String raza) {
    this.raza = raza;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public int getEdad() {
    return edad;
}

public void setEdad(int edad) {
    this.edad = edad;
}

public boolean isTieneChip() {
    return tieneChip;
}

public void setTieneChip(boolean tieneChip) {
    this.tieneChip = tieneChip;
}
```

7.4. ENCAPSULADO. VISIBILIDAD DE UNA CLASE.

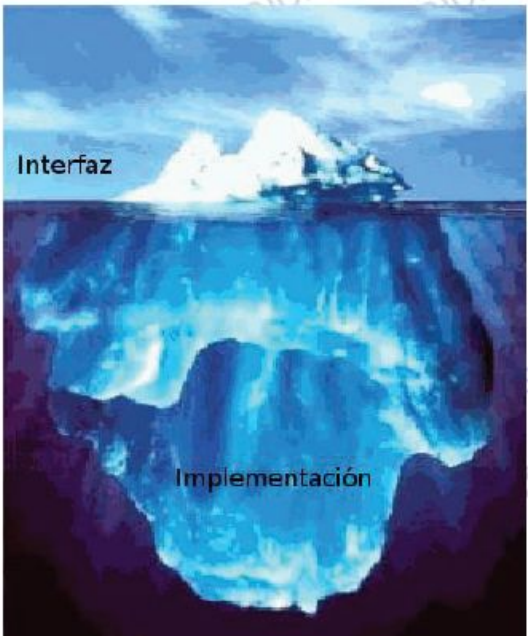
Aunque creemos un objeto de una clase no todos los elementos de dicha clase son visibles y accesibles.

7.4.1. ENCAPSULADO:

Encapsulado se define como: “proceso de almacenar en un mismo comportamiento los elementos de una abstracción que constituyen su estructura y su comportamiento”.

Consiste en ocultar atributos de un objeto al exterior, de forma que los datos y su funcionamiento están protegidos de acciones de otros objetos.

7.4.2. VISIBILIDAD:



Se distingue una parte de implementación y una parte de interfaz.

7.3.2.1. PARTE VISIBLE:

Las clases ofrecen una parte que es visible a otras entidades que se denomina “interfaz”. La interfaz se compone de las operaciones que definen el comportamiento e interacción con el objeto de dicha clase.

7.3.2.2. PARTE INVISIBLE:

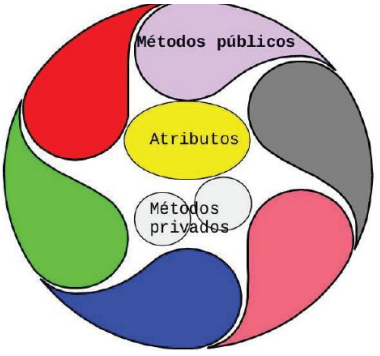
Las clases poseen una parte privada que solamente es visible dentro de la propia clase. Por lo tanto, no es accesible desde

el objeto que se crea sino para el funcionamiento interno de dicho objeto. Por lo tanto, los detalles de implementación quedan ocultos.

7.4.3. ¿COMO AFECTA AL USO DE LOS OBJETOS?

Utilizaremos los modificadores de acceso para definir la accesibilidad o visibilidad de cada uno de los atributos y métodos:

- VISIBLE Y ACCESIBLE: **public.**
- INVISIBLE Y NO ACCESIBLE DIRECTAMENTE: **private.**



Por lo general todos los atributos se deberán definir como **privados (invisibles)**, es decir, no son accesibles directamente.

Entonces, ¿cómo podría acceder a un atributo o modificarlo? Para ello en toda clase debemos definir una serie de métodos denominados **"Getters & Setters":** Estos métodos serán públicos (visibles) y accesibles y permitirán modificar o acceder a cada uno de los atributos dado un objeto.

7.5. ¿CÓMO INSTANCIAR UN OBJETO DE UNA CLASE?

Un objeto es una instancia de una clase creada en el tiempo de ejecución. Mantendrá una estructura de datos formada por tantos atributos como tiene la clase y tantas funciones como métodos públicos tenga definido.

7.6. ESTADO DE UN OBJETO.

El estado de un objeto viene dado por el valor de los atributos en un momento dado. El cual puede variar/evolucionar con el paso del tiempo.