**RESUMEN AYED**

Una clase java es un bloque de código o un molde, que describe el estado y el comportamiento que tendrán los objetos que con ella se creen. agregarContacto • Un archivo origen java debe tener como mínimo:- en la primera línea la palabra clave package seguida del nombre del paquete.- la palabra clave class seguida del nombre de la clase. • Un archivo origen java debe guardarse con el mismo nombre que la clase (y con extensión .java). Se deben respetar las mayúsculas

**Para agregar estado y comportamiento debemos incluir:**

•**Variables de instancia**: constituyen el estado de un objeto. Normalmente, las variables de instancia se declaran private, lo que significa que se puede acceder a ellas directamente sólo desde la clase donde se definen.

•**Métodos de instancia**: definen las operaciones que pueden realizar los objetos de un tipo de clase. Un método es un bloque de código, similar a lo que es una función o procedimiento en los lenguajes procedurales.

**En java hay 2 categorías de tipos de datos: tipo primitivo y tipo referencial o de una clase particular:**

• **Tipos primitivos**: las variables de tipo primitivo mantienen valores simples y NO son objetos. Existen 8 tipos de datos primitivos:

**Entero**: byte, short, int, long

**Punto flotante:** float y double

**Un carácter de texto**: char

**Lógico:** boolean.

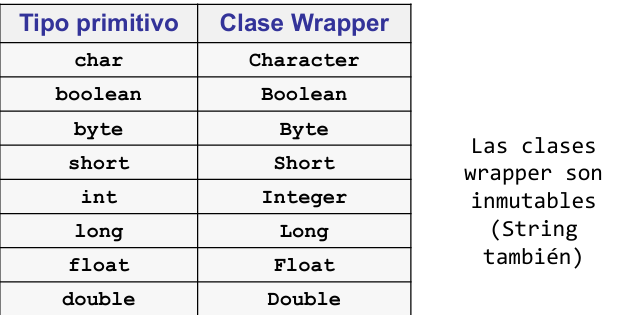
**Tipos de una clase**: las variables que referencian a un objeto son llamadas variables referencias y contienen la ubicación (dirección de memoria) de objetos en memoria.

**Clases wrappers:**

• Java no considera a los tipos de datos primitivos como objetos. Los datos numéricos, booleanos y de caracteres se tratan en su forma primitiva por razones de eficiencia.

• Java proporciona clases wrappers para manipular a los datos primitivos como objetos. Los datos primitivos están envueltos ("wrapped") en un objeto que se crea en torno a ellos.

• Cada tipo de datos primitivo de Java, posee una clase wrapper correspondiente en el paquete java.lang. Cada objeto de la clase wrapper encapsula a un único valor primitivo.



**Autoboxing** Es la conversión automática que realiza el compilador de Java entre los tipos primitivos y sus clases wrappers correspondientes. Por ejemplo, convertir un int en un Integer, un double en un Double, etc.

**Character c = 'a';**

**Integer i = 7**

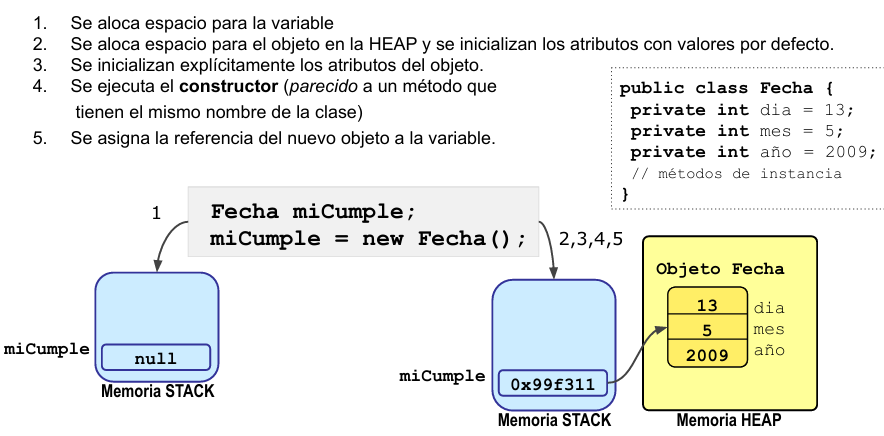
**Unboxing** Es la conversión es al revés, es decir conversión de wrapper a un primitivo. Por ejemplo de un Character a char o Double a double.

**char c1 = c;**

**int i1 = i**

¿Cómo se crean instancias de una clase?

\Para instanciar una clase, es decir, para crear un objeto de una clase, se usa el operador new. La creación e inicialización de un objeto involucra los siguientes pasos: \

****

Los constructores son piezas de código -sintácticamente similares a los métodos- que permiten definir un estado inicial específico de un objeto, en el momento de su creación.

Se diferencian de los métodos tradicionales porque:

•Deben tener el mismo nombre que la clase. La regla de que el nombre de los métodos debe comenzar con minúscula, no se aplica a los constructores.

•No retornan valor.

•Son invocados automáticamente

**La inicialización está garantizada**: cuando un objeto es creado, se aloca almacenamiento en la memoria HEAP y se invoca al constructor.

- La expresión new retorna una referencia al objeto creado recientemente, pero el constructor no retorna un valor.

- Java siempre llama automáticamente a un constructor cuando crea un objeto (antes de que el objeto sea usado). De esta forma la inicialización del objeto está garantizada.

**-------------------------------------------------------------------------------------------------------------------------------**

***Sobrecarga de Constructores:*** La sobrecarga de métodos permite que el mismo nombre de método sea usado con distintos tipos y cantidad de argumentos.

**Variables de instancia y variables locales:**

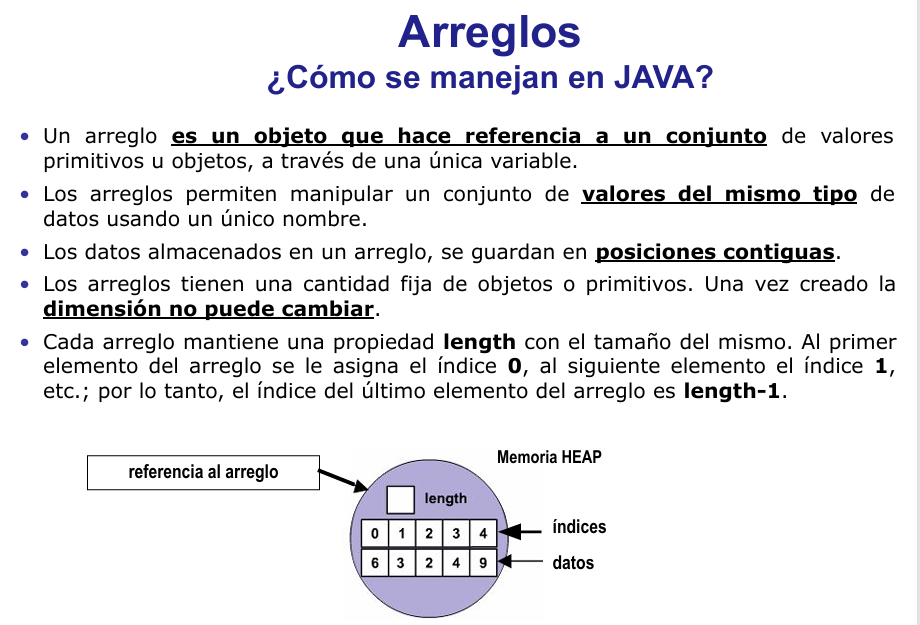
Las variables pueden declararse en dos lugares diferentes (siempre adentro de la clase):

• **afuera de cualquier método**. Son las variables de instancia que son creadas cuando el objeto es construido usando el new(). Estas variables existen, mientras exista el objeto.

• **adentro de cualquier método.** Estas variables son llamadas variables locales y deben inicializarse antes de ser usadas. Los parámetros de los métodos también son variables locales y las inicializan el código que llama al método. Estas variables son creadas cuando el método comienza a ejecutar y son destruidas cuando el método finaliza su ejecución

**La palabra clave static:** La palabra clave static declara atributos (variables) y métodos asociados con la clase en lugar de asociados a cada una de las instancias de la clase.

**---------------------------------------------------------------------------------------------------------------------🡪**



**Pasaje de parámetros en Java**

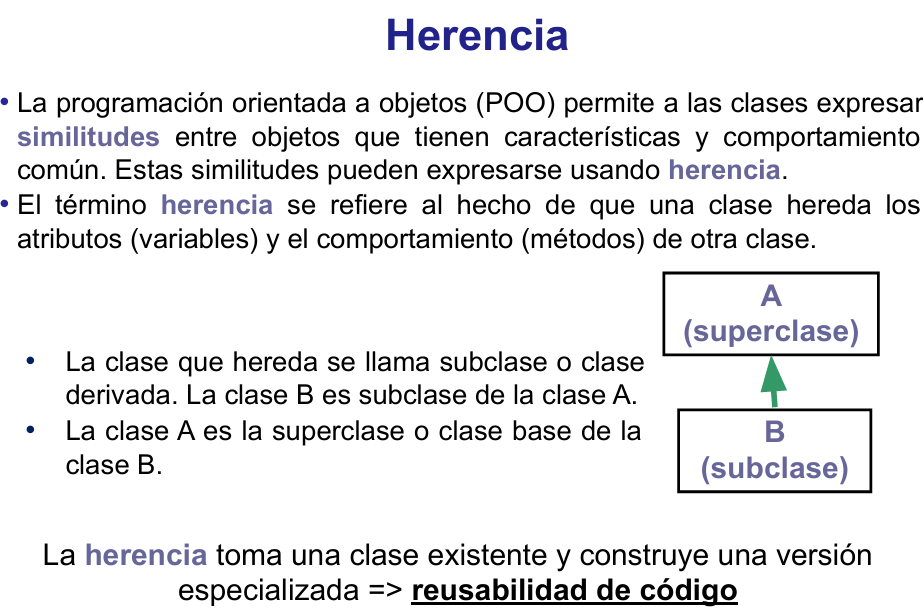
En Java los parámetros se pasan por valor. Pasaje por valor significa que cuando se invoca a un método, se pasan como argumentos al método una copia de cada parámetro actual.

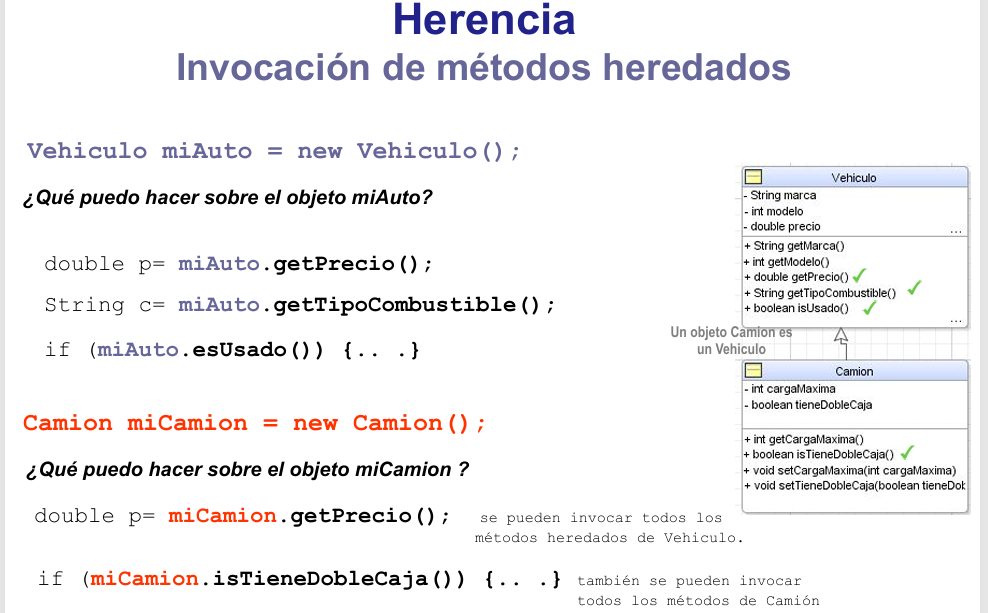
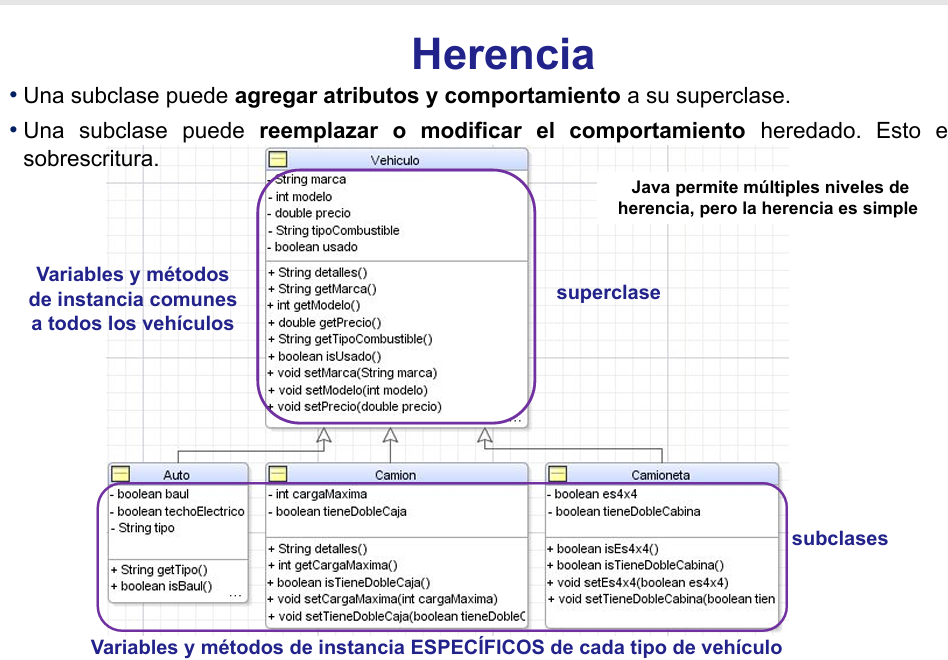
**Tipo de dato primitivo :**

Cuando Java llama a un método, hace una copia de sus parámetros reales y envía las copias al método. Cuando el método finaliza, esas copias se descartan y los valores de las variables en el código principal son los mismos que antes.

En Java, podemos pasar como parámetro una referencia a un objeto y con ella cambiar el estado (valor de sus variables) del objeto, pero no la referencia.









**Herencia**

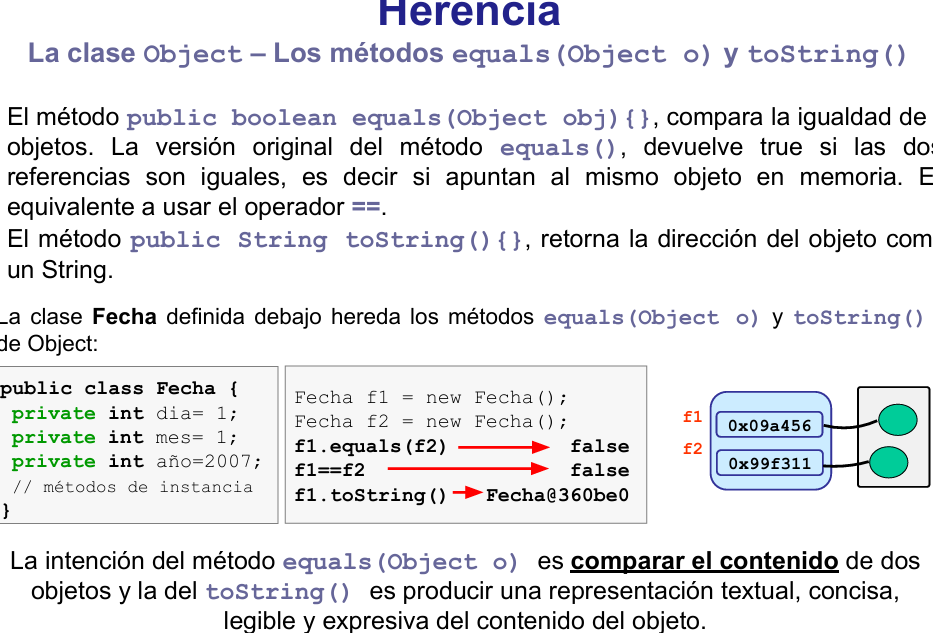
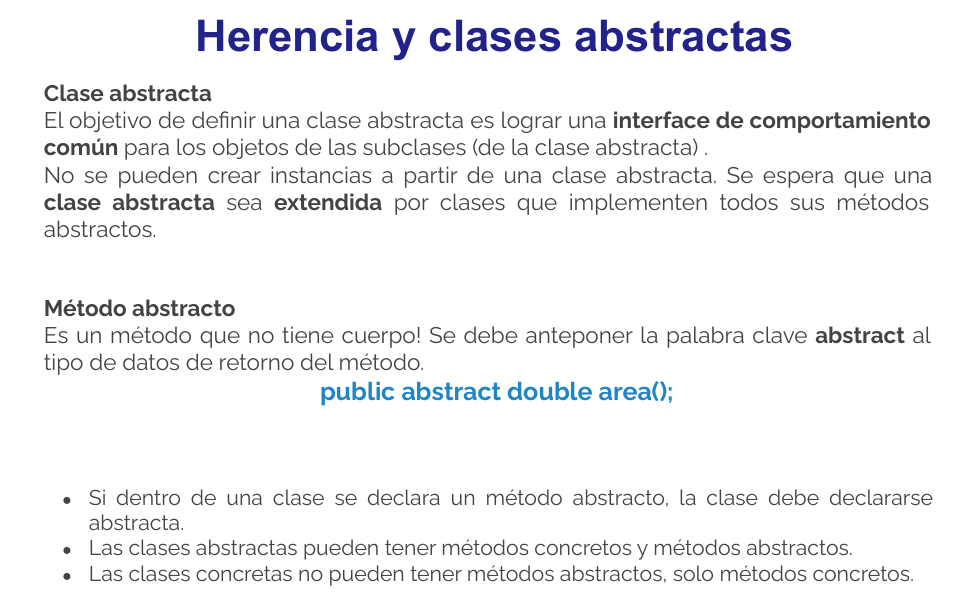
**Upcasting – Downcasting:**

Tratar a una referencia de la clase derivada como una referencia de la clase base, se denomina upcasting. Con el upcasting, la conversión es hacia arriba en la jerarquía de herencia y se pierde el tipo específico del objeto. Para recuperar el tipo del objeto, se debe mover hacia abajo en la jerarquía y a esto se lo llama downcasting.

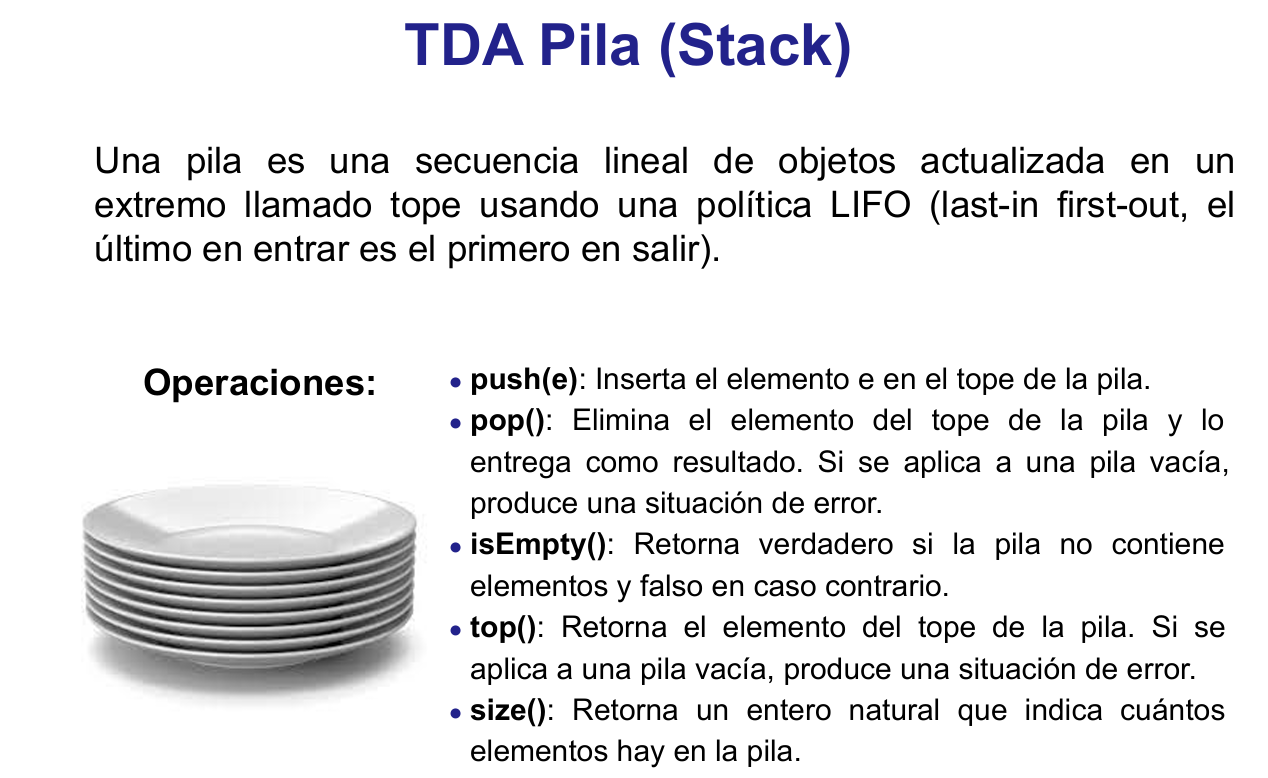
El upcasting es seguro, la clase base tiene una interface que es igual o es un subconjunto de la clase derivada. Pero, en el downcasting no ocurre lo mismo.

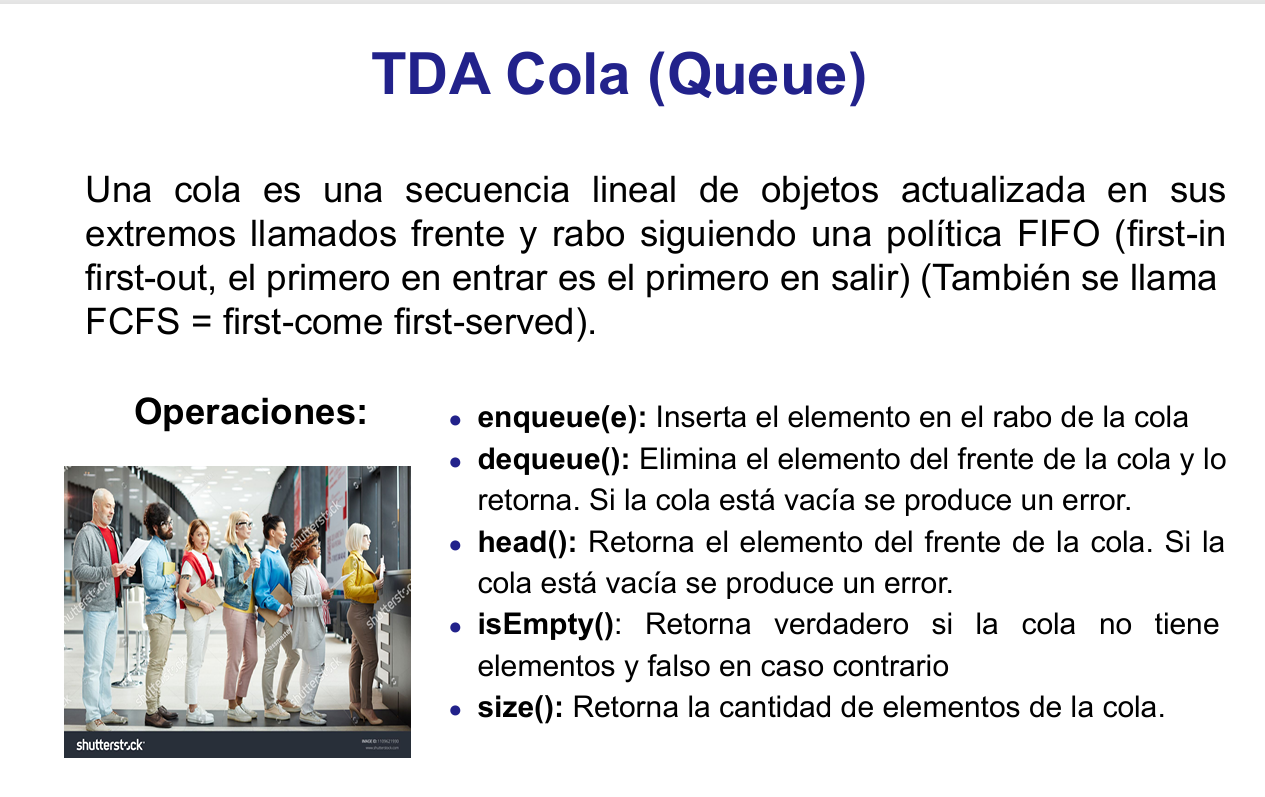
**La clase Object:**

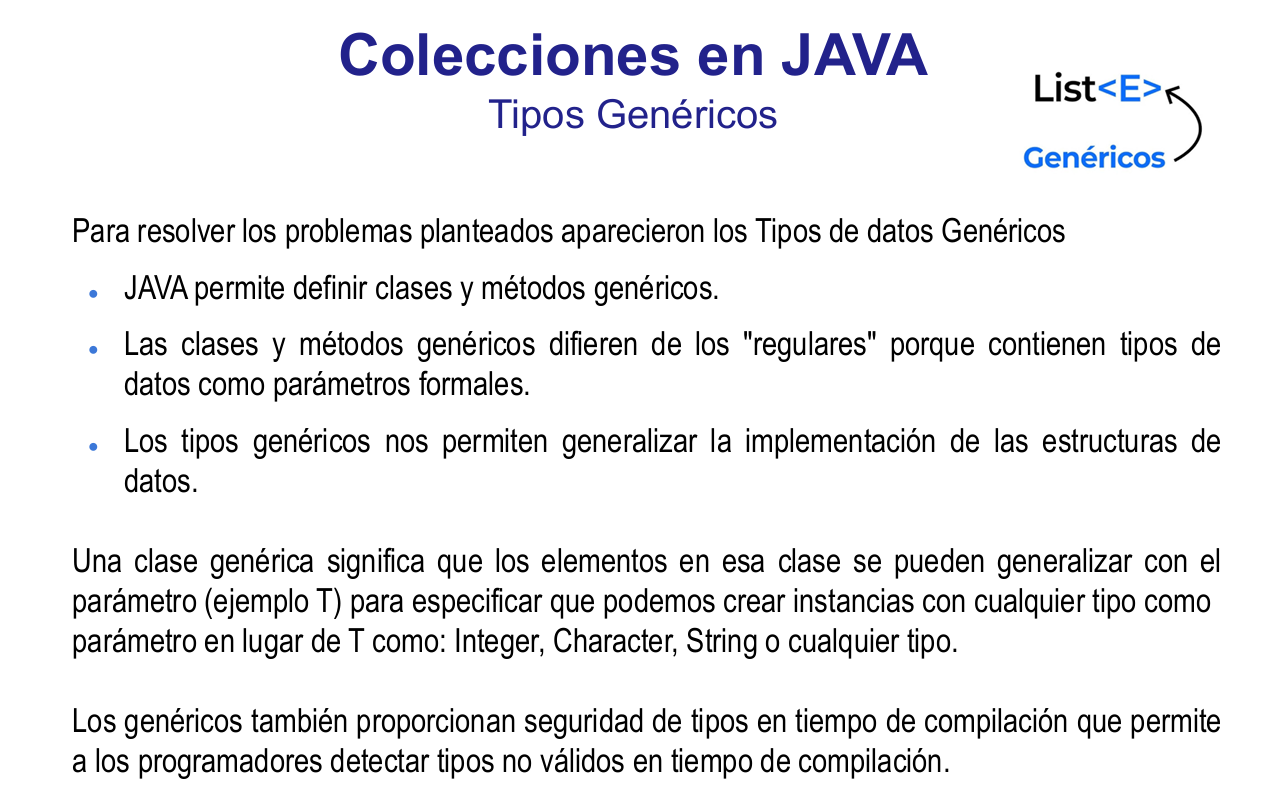
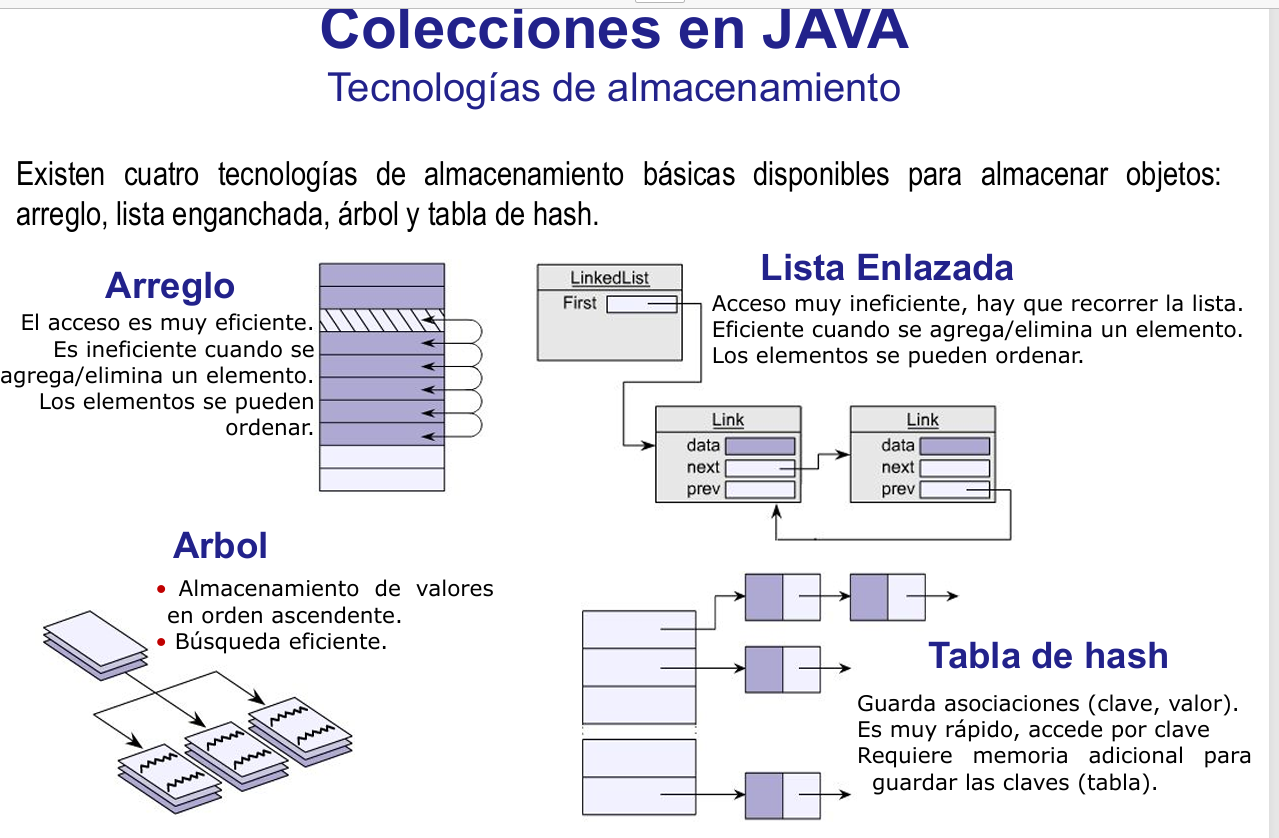
La clase Object es la raíz de todas las clases JAVA y está ubicada en el paquete java.lang •Cuando se declara una clase sin usar la palabra clave extends el compilador JAVA implícitamente agrega el código extends Object a la declaración de la clase.

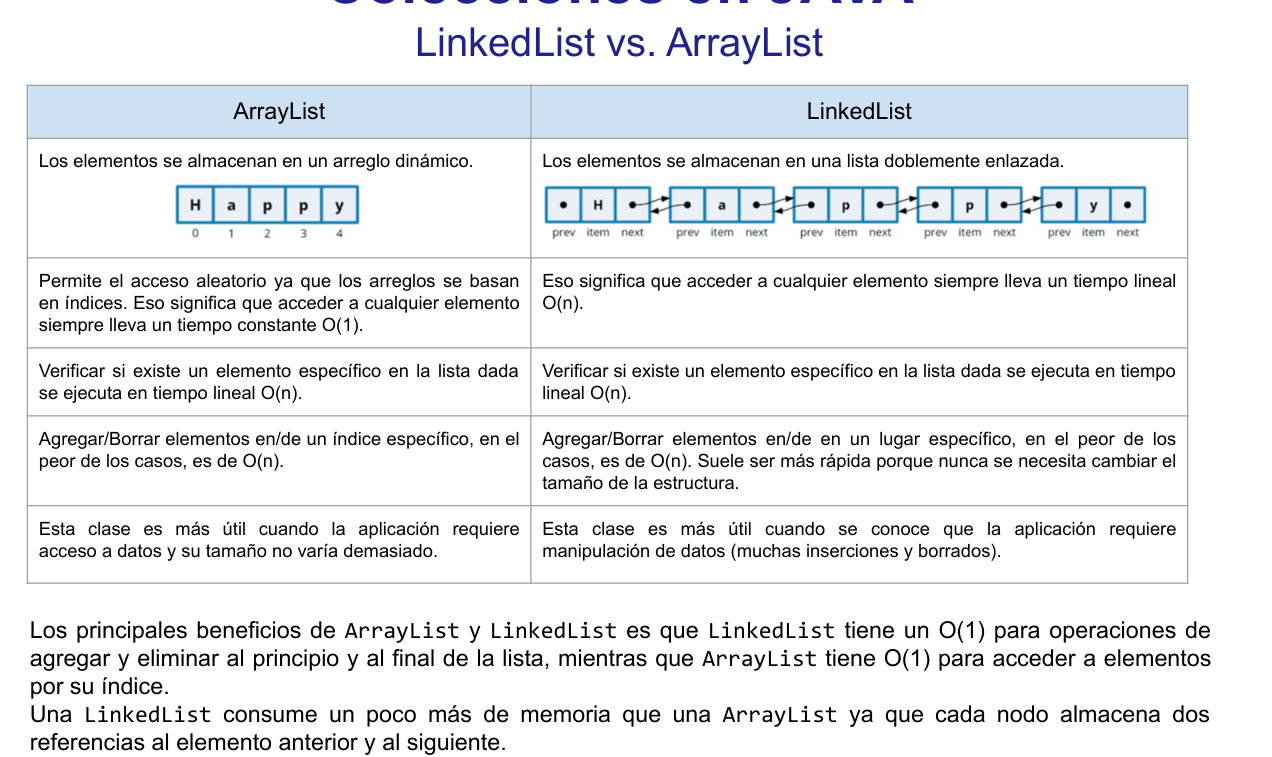
** **

***Tipo de dato abstracto (TDA) Listas, Pilas y Colas:*** tipo definido solamente en términos de sus operaciones y de las restricciones que valen entre las operaciones.

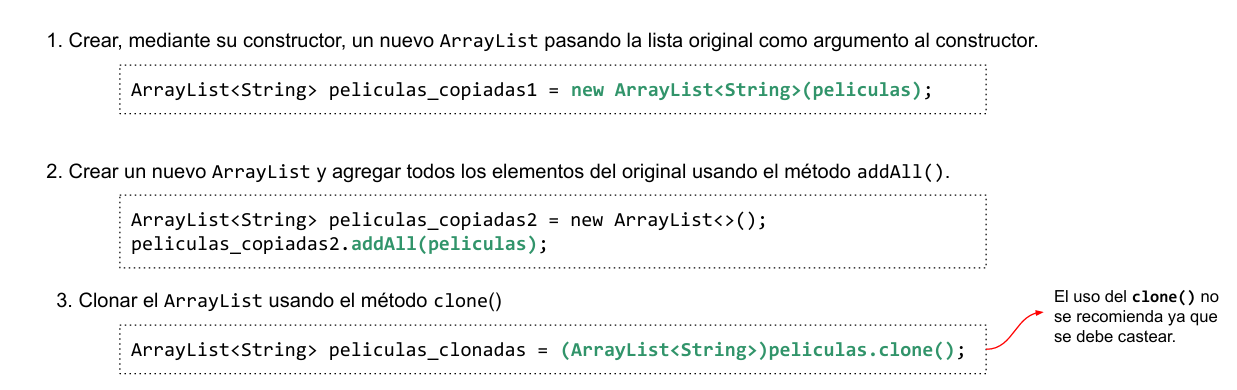
******



Qué es ArrayList? Es una clase, que incluye la API de Java que implementa una lista con un arreglo de tamaño variable. Está dentro del paquete java.util y pertenece al framework de colecciones. Podríamos usar también LinkedList, que implementa una lista con nodos enlazados



**Métodos para copiar colecciones**

****

**Iteradores:** patrón de diseño de comportamiento que permite el recorrido secuencial por una estructura de datos sin exponer sus detalles internos.

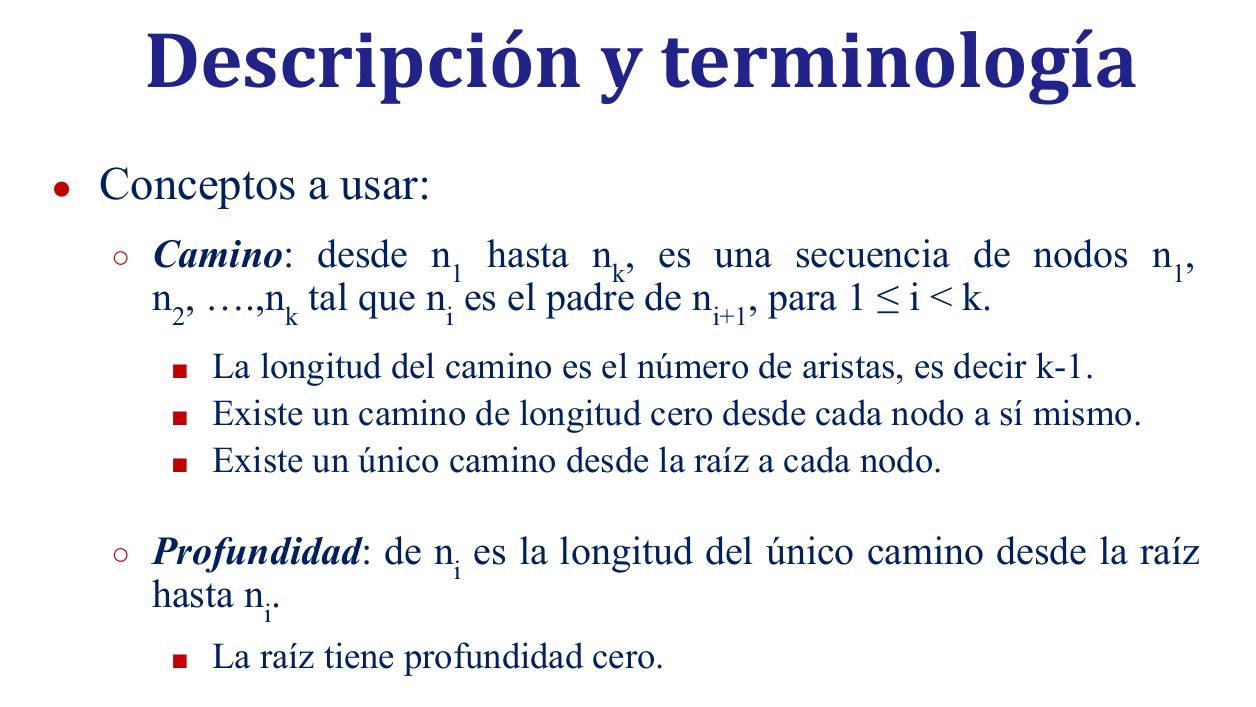
TDA Iterable: Una colección es iterable si dispone del método iterator() para iterar la colección -debe implementar la interface java.util.Iterable-.

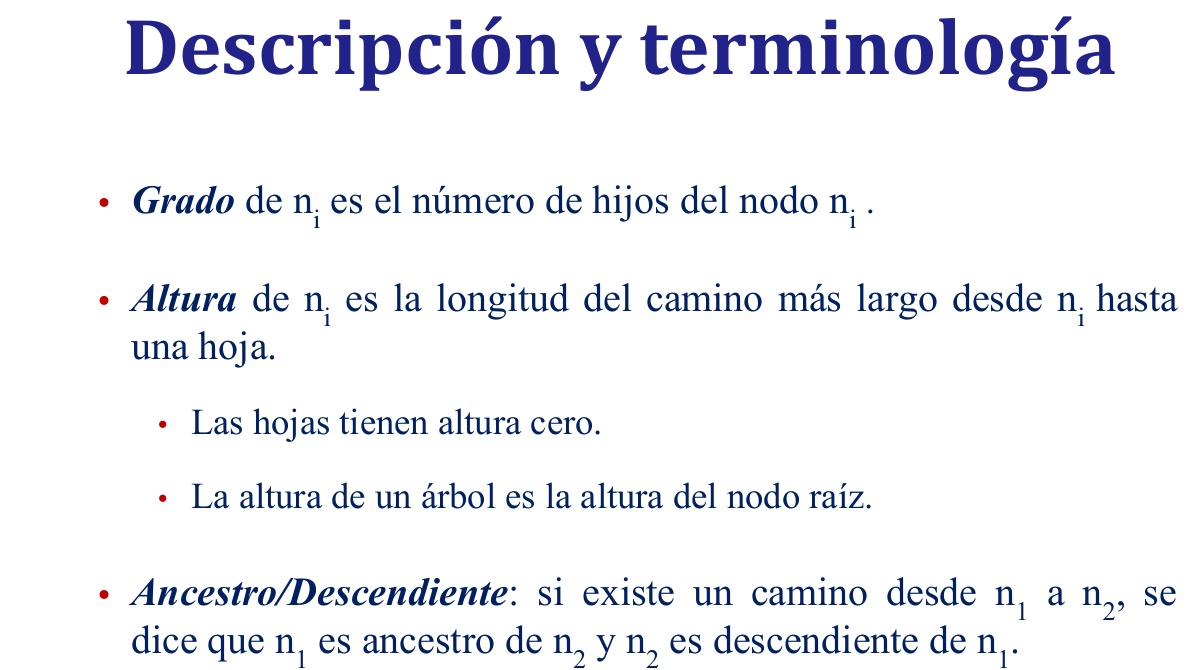
TDA Iterador: La interface java.util.Iterator brinda los siguientes métodos: - hasNext(): Testea si hay elementos para recorrer en el iterador - next(): Retorna el siguiente elemento del iterador

CLASE 3.1

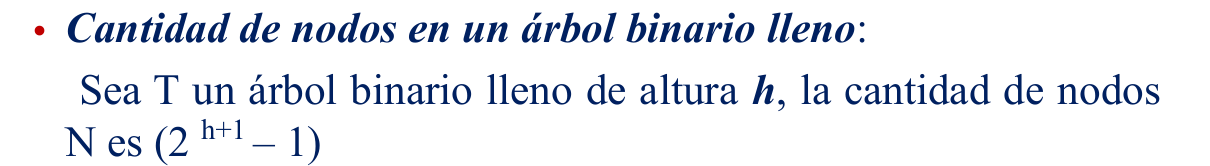
**Árboles Binarios :** Un árbol binario es una colección de nodos, tal que puede estar vacía, puede estar formada por un nodo distinguido R, llamado raíz y dos sub-árboles T1 y T2 , donde la raíz de cada subárbol Ti está conectado a R por medio de una arista.

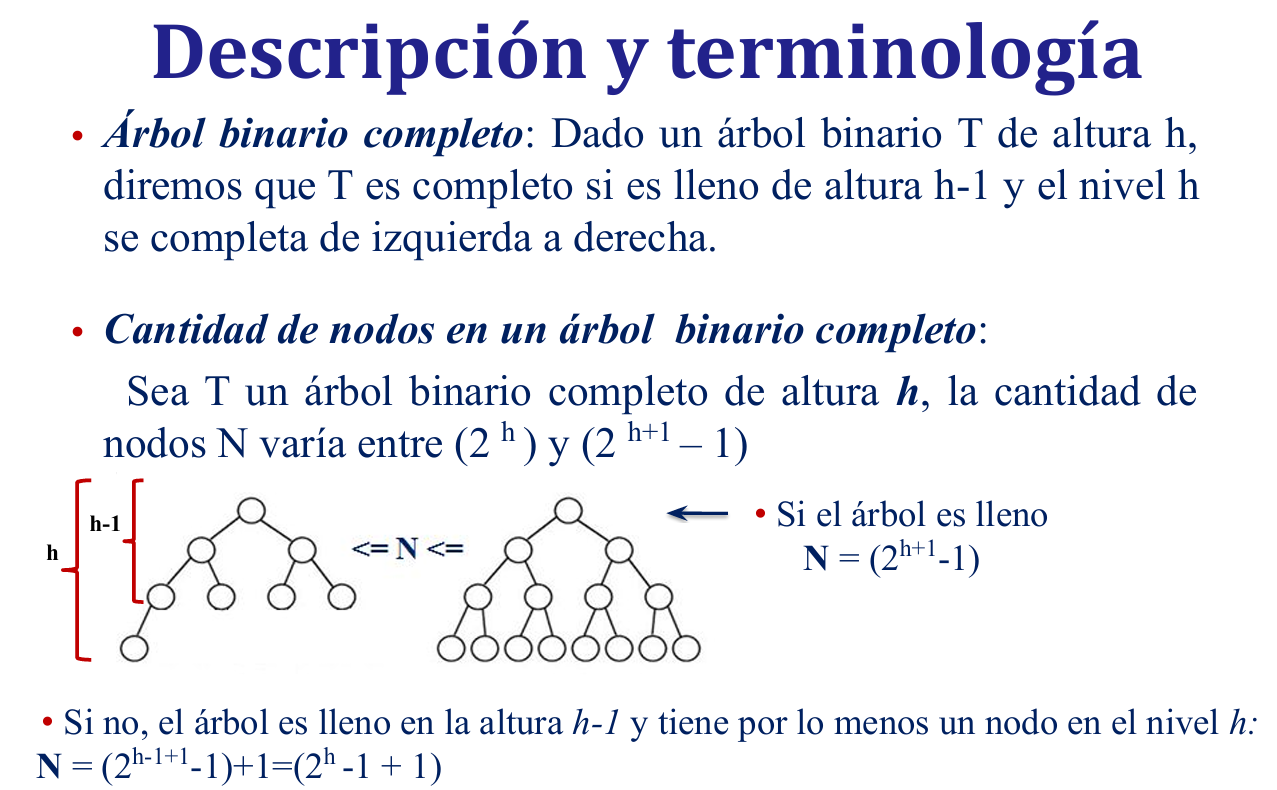
**Descripción y terminología: Cada nodo puede tener a lo sumo dos nodos hijos, Cuando un nodo no tiene ningún hijo se denomina hoja,** **Los nodos que tienen el mismo nodo padre se denominan hermanos.**

****

****

Árbol binario lleno: Dado un árbol binario T de altura h, diremos que T es lleno si cada nodo interno tiene grado 2 y todas las hojas están en el mismo nivel (h).

****

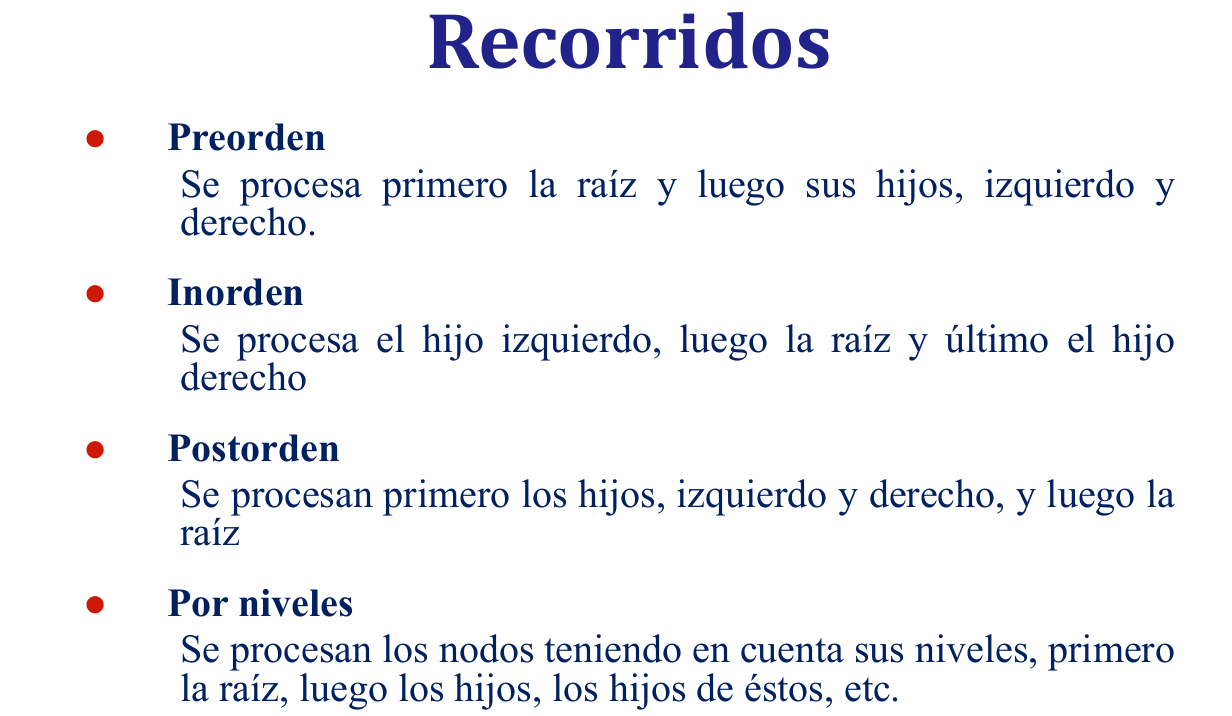
****

**Cada nodo tiene:**

•Información propia del nodo

•Referencia a su hijo izquierdo

•Referencia a su hijo derecho

****

Recorrido: Preorden

public void preorden(){

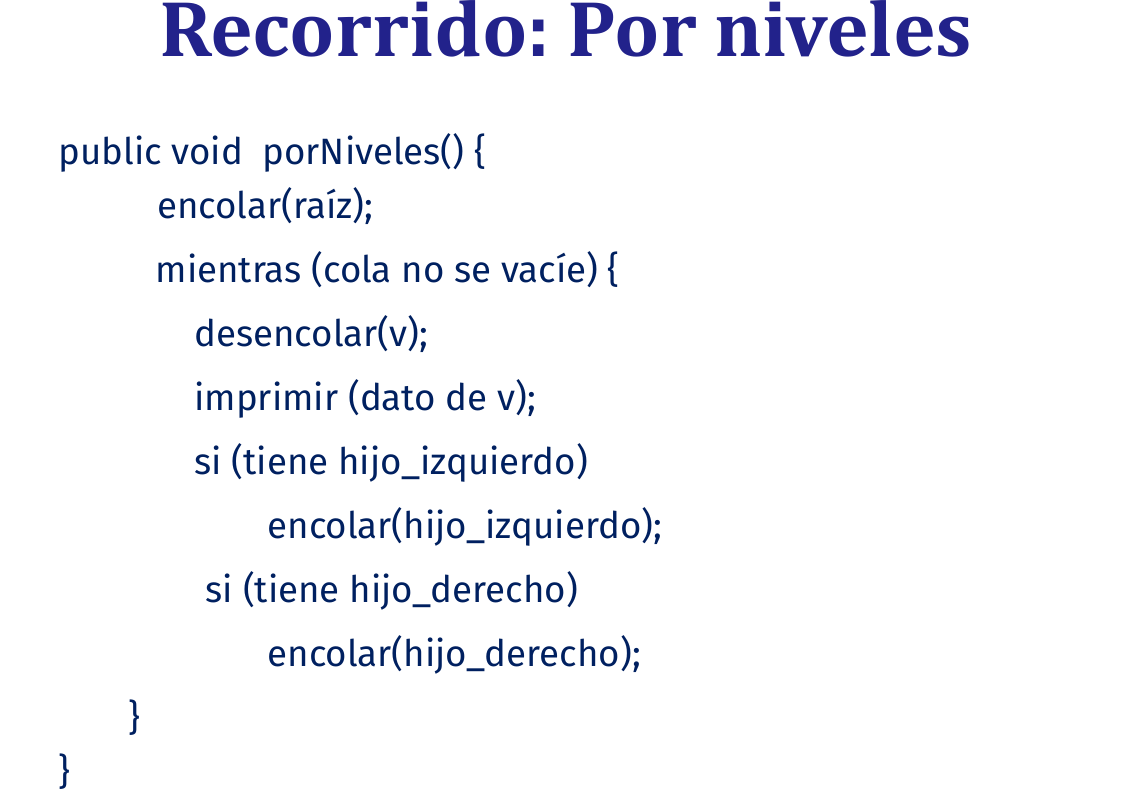
imprimir (dato);

si (tiene hijo\_izquierdo)

hijoIzquierdo.preorden();

si (tiene hijo\_derecho)

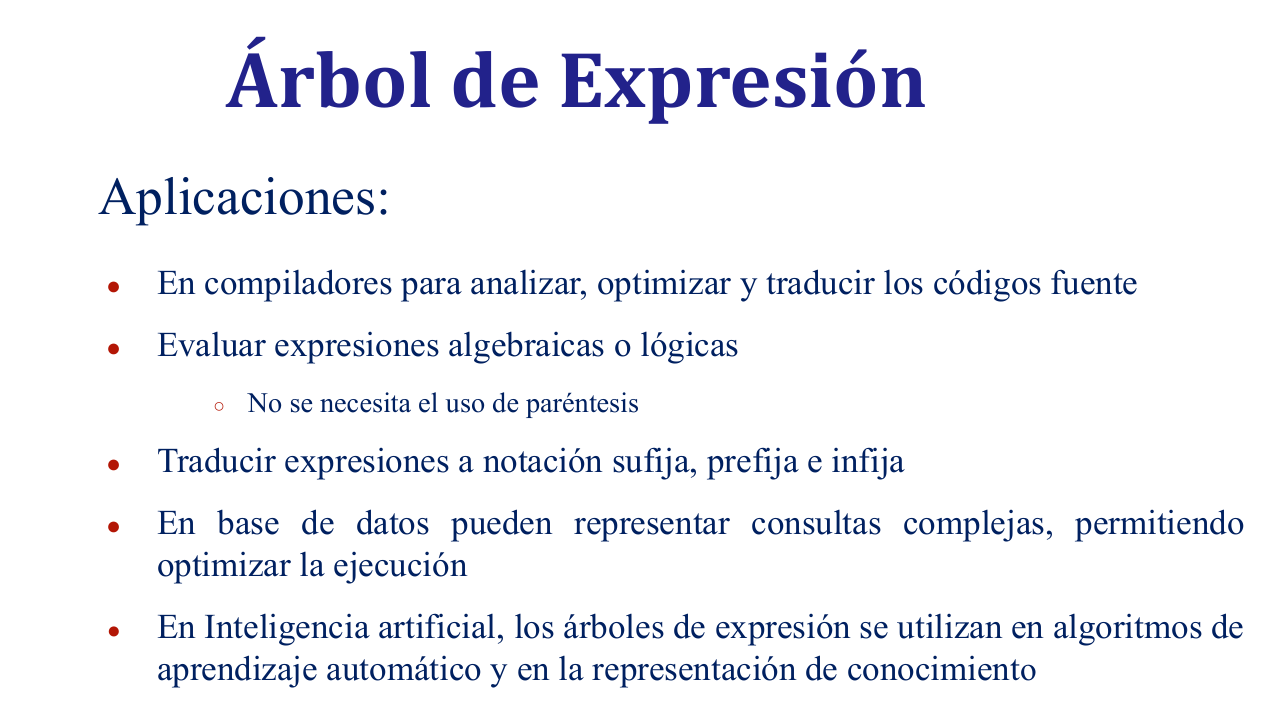
hijoDerecho.preorden();



**Árbol de Expresión Es un árbol binario asociado a una expresión aritmética :**

● Nodos internos representan operadores

● Nodos externos (hojas) representan operandos

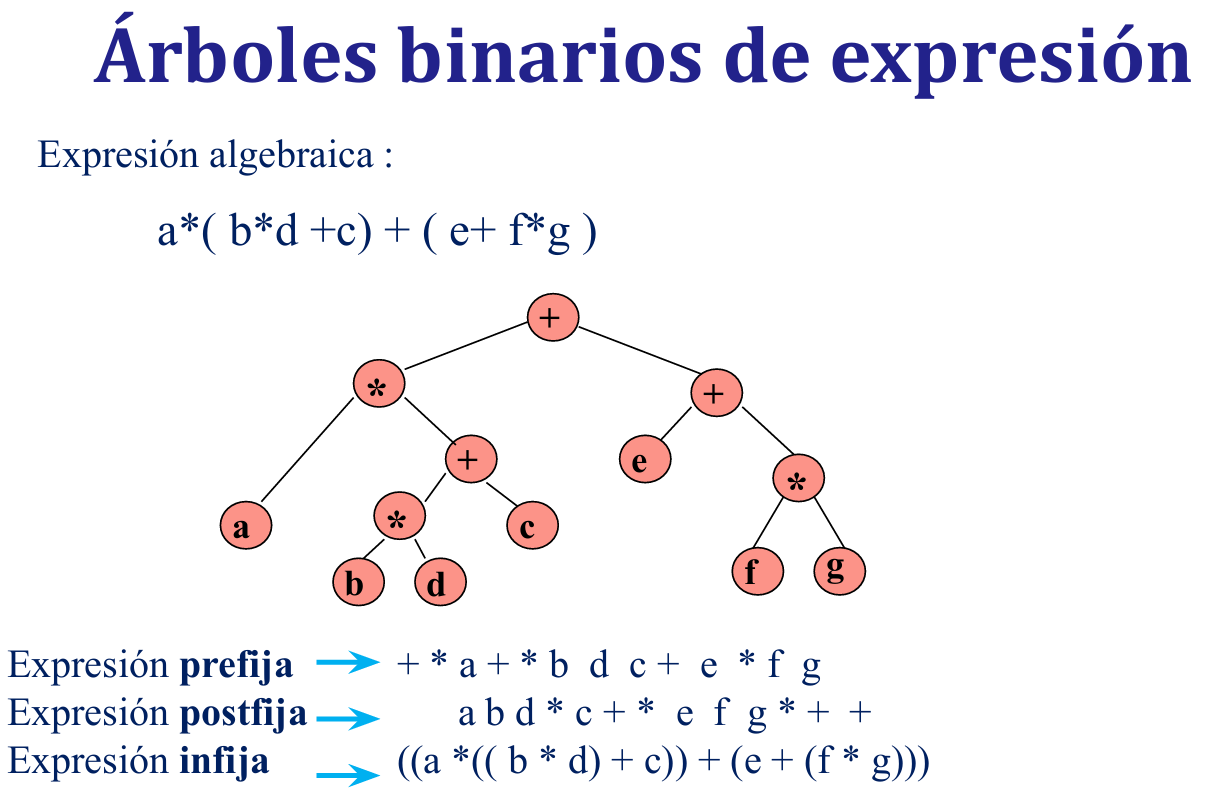


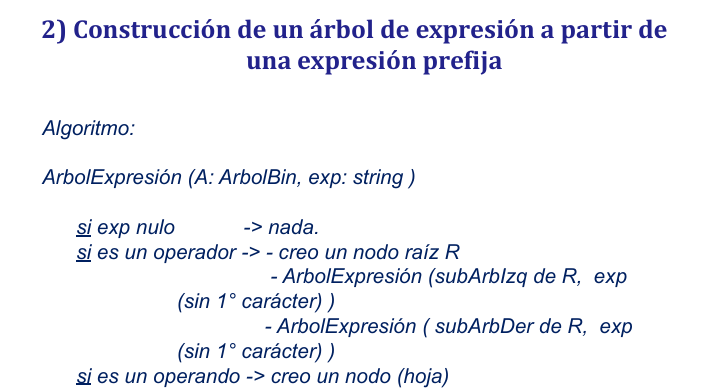
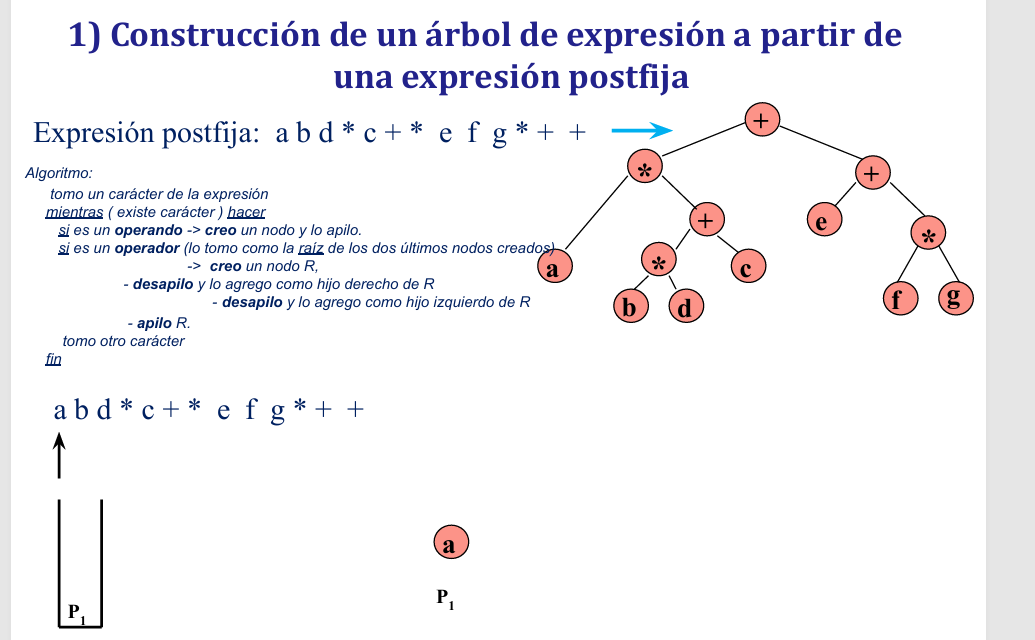
Construcción de un árbol de expresión A partir de una:

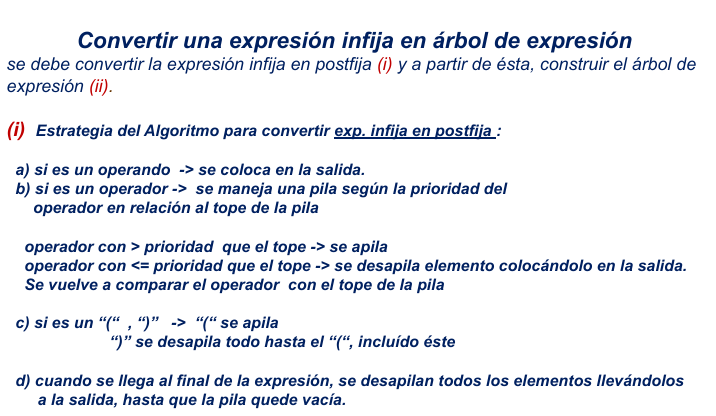
1) Expresión postfija

2) Expresión prefija

3) Expresión infija







Operadores ordenados de mayor a menor según su prioridad: ˄ \*, / (potencia) (multiplicación y división) +, - (suma y resta) Los “ ( “ siempre se apilan como si tuvieran la mayor prioridad y se desapilan sólo cuando aparece un “ ) ”

