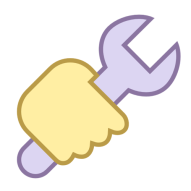
**CFP 8**

**Módulo 2:**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Unidad 5:**

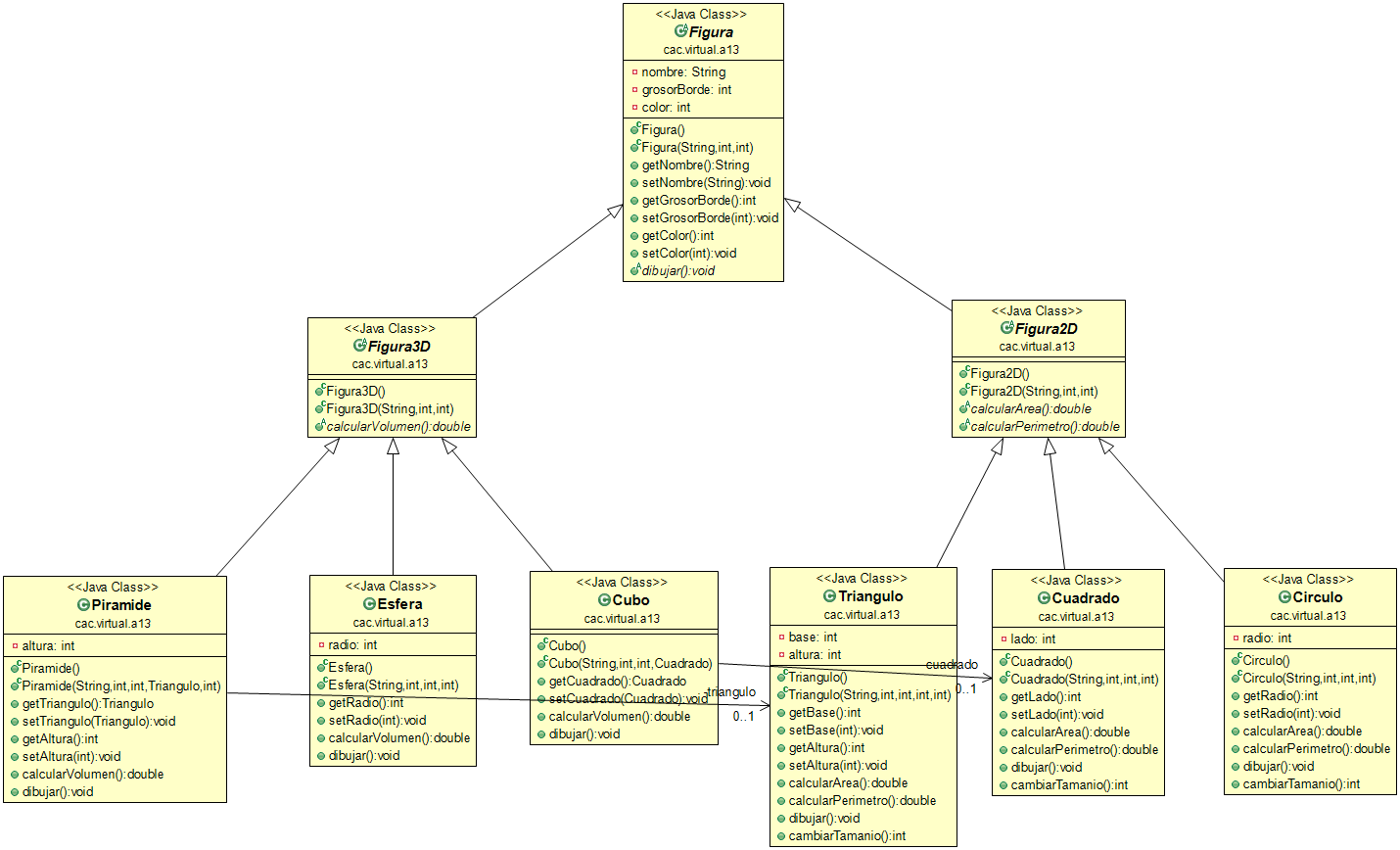
**POLIMORFISMO**

****

## ACTIVIDADES PRÁCTICAS

### Polimorfismo, Diferencia entre polimorfismo y sobrecarga, Clases abstractas, Métodos abstractos, Interfaz, Herencia múltiple

Considere la siguiente jerarquía de clases java representada por este diagrama UML para los **ejercicios 1 a 4**:

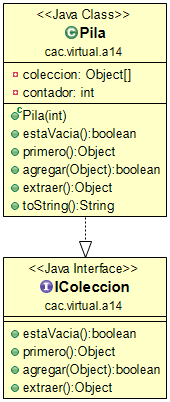


1. Se hará una librería de clases que represente figuras tridimensionales y bidimensionales, y su respectiva jerarquía de clases. Las clases deben ser capaces de tener funcionamiento bastante básico, como obtener áreas, volúmenes y perímetros de la figura correspondiente. La representación de la jerarquía sería como la representada en el diagrama. Para este ejercicio deberá crear la **clase abstracta** Figura según el diagrama, los atributos y los métodos que ahí figuran.
2. Crear las **clases abstractas** Figura3D y Figura2D considerando el diagrama de clases y métodos que ahí figuran.
3. Crear las clases Cuadrado, Círculo y Triángulo considerando el diagrama de clases y métodos que ahí figuran.
4. Crear las clases Esfera, Pirámide y Cubo considerando el diagrama de clases y métodos que ahí figuran.
5. Escribe una **interfaz**, llamada IColeccion, que declare los siguientes métodos:

* estaVacia(): devuelve true si la colección está vacía y false en caso contrario.
* extraer(): devuelve y elimina el primer elemento de la colección.
* primero(): devuelve el primer elemento de la colección.
* añadir(): añade un objeto, y devuelve true si se ha añadido y false en caso contrario.

**Nota:** en la interfaz se declaran todos los métodos sin implementar ninguno.

1. Crea una clase llamada Pila, que implemente la **interfaz** IColeccion del ejercicio anterior, utilizando para ello un ArrayList de Object. Para la clase se utilizan como atributos un ArrayList de Object. Tenga en cuenta la siguiente jerarquía al momento de implementar la interfaz y resolver el resto de los ejercicios.



1. Métodos estaVacia y primero de la clase Pila:
2. Escribir el código para el método estaVacia() que comprueba si el contador (método size del ArrayList) es 0 y devuelve true o false según corresponda.
3. Escribir el código para el método primero() que si la Pila está vacía, devuelve null, si no, devuelve el elemento que está indicado por el contador.
4. Escribir el código para el método añadir() de la clase Pila que comprueba que cabe el elemento, y si es así, lo añade en la celda que indica el contador, posteriormente se incrementa el valor del contador. Devuelve true o false, según si se ha podido añadir o no.
5. Escribir el código para el método extraer() que si la Pila está vacía, devuelve null, si no decrementa el contador y devuelve el elemento que está en la celda indicada por el contador, después de decrementarse.
6. **Sobrescribir** el método toString() de la Clase Pila.