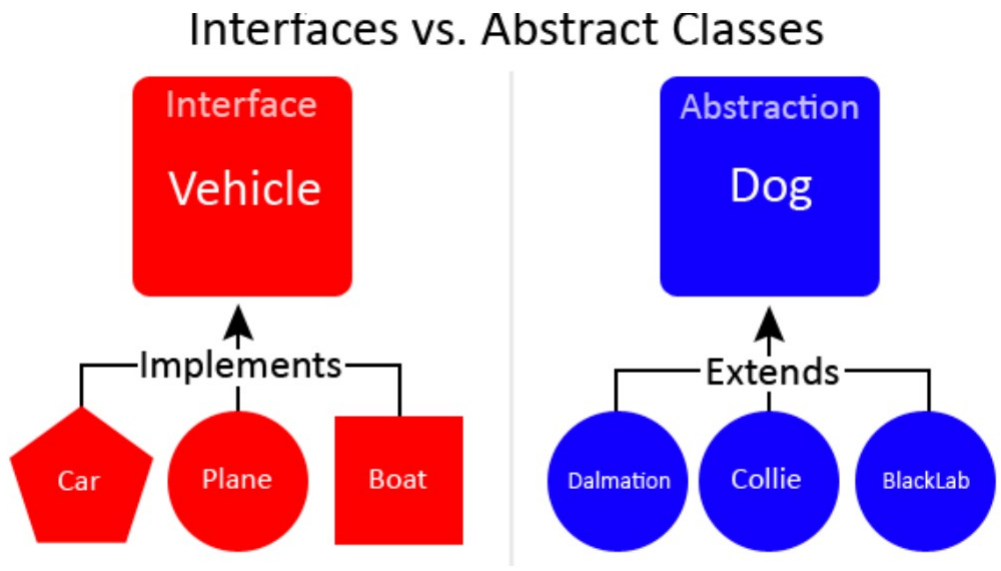

Entrambe permettono di definire un concetto senza dover conoscere i dettagli di una classe, dettagli che saranno implementati dai figli.

- Una classe astratta, oltre a metodi astratti, può contenere dati e metodi non astratti. Una interfaccia no
- Una classe astratta può avere un costruttore. Una interfaccia no

Se in una classe astratta tutti i metodi sono astratti, siamo in presenza di un'interfaccia.

Una classe astratta che dichiara almeno un metodo astratto, deve essere dichiarata astratta anch'essa. La classe Veicolo per esempio, è astratta perché non sappiamo come definire i suoi metodi accelera e decelera dato che non sappiamo di che veicolo stiamo parlando. Una tale classe è viene definita per essere estesa da sottoclassi concrete, per esempio la classe Auto, la classe Nave, e la classe Aereo, ognuna delle quali fornirà un'implementazione diversa ai metodi accelera e decelera con degli override.

A livello di progettazione quindi le classi astratte costituiscono uno strumento fondamentale. Infatti, dichiarare una classe abstract, significa voler indirizzare lo sviluppo verso la creazione di sottoclassi della classe astratta.

Questo non solo ci permetterà di risparmiare codice che avremmo dovuto mettere nelle sottoclassi e che invece possiamo mettere a fattor comune nella superclasse astratta, ma soprattutto ci darà la possibilità di trattare gli oggetti delle sottoclassi Auto, Nave, e Aereo, come fossero veicoli. Pertanto, non ha senso dichiarare una classe astratta se non si ha intenzione di estenderla.

Esempio di Interface

```
interface Animale
{
   public void verso();
}

class Gatto implements Animale
{
   public void verso()
    {
      System.out.println("miao");
    }
}

class Cane implements Animale
{
    public void verso()
     {
        System.out.println("bau");
     }
}
class interfaccia
{
   public static void main(String[] args)
   {
      Gatto gattoObj = new Gatto();
      gattoObj.verso();
      Cane caneObj = new Cane();
      caneObj.verso();
   }
}
```