**Encapsulation**

It’s basically about hiding the state of object with the help of modifiers like private, public, protected etc. we expose the state thru public methods only if require.

Encapsulation is a strategy used as part of abstraction. Encapsulation refers to the state of objects - objects encapsulate their state and hide it from the outside; outside users of the class interact with it through its methods, but cannot access the classes’ state directly. So the class abstracts away the implementation details related to its state.

**Abstraction:**

The process of abstraction in Java is used to hide certain details and only show the essential features of the object. In other words, it deals with the outside view of an object (interface). The only good example i see for this across different sites is interface

is a more generic term, it can also be achieved by (amongst others) subclassing. For example, the interface List in the standard library is an abstraction for a sequence of items, indexed by their position, concrete examples of a List are an ArrayList or a LinkedList. Code that interacts with a List abstracts over the detail of which kind of a list it is using.

Abstraction is often not possible without hiding underlying state by encapsulation - if a class exposes its internal state, it can't change its inner workings, and thus cannot be abstracted

**Conclusion**: In simple words: You do abstraction when deciding what to implement. You do encapsulation when hiding something that you have implemented.