An Abstract class doesn't have to implement abstract methods

An abstract class that extends another abstract class has some flexibility.

- It can implement all of the parent's abstract methods.
- It can implement some of them.
- Or it can implement none of them.
- It can also include additional abstract methods, which will force subclasses to implement both Animal's abstract methods, as well as Mammal's.

Why use an abstract class?

In truth, you may never need to use an abstract class in your design, but there are some good arguments for using them.

An abstract class in your hierarchy forces the designers of subclasses to think about, and create unique and targeted implementations, for the abstracted methods.

It may not always make sense to provide a default, or inherited implementation of a particular method.

An abstract class can't be instantiated, so if you're using abstract classes to design a framework for implementation, this is definitely an advantage.

Why use an abstract class?

In our example, we don't really want people creating instances of Animals or Mammals.

We used those classes to abstract behavior at different classification levels.

All Animals have to implement the move and makeNoise methods, but only Mammals needed to implement shedHair, as I demonstrated.