An abstract class is a class that you cannot create an instance of. It can provide basic functionality, but in order for that functionality to be used, one or more other classes must derive from the abstract class. One of the major benefits of abstractclasses is that you can reuse code without having to retype it. That has a plethora of benefits, such as reducing bugs and making coding faster. A concrete example of an abstract class would be a class called Animal. You see many animals in real life, but there are only kinds of animals. That is, you never look at something purple and furry and say "that is an animal and there is no more specific way of defining it". Instead, you see a dog or a cat or a pig... all animals. The point is, that you can never see an animal walking around that isn't more specifically something else (duck, pig, etc.). The Animal is the abstract class and Duck/Pig/Cat are all classes that derive from that base class. Animals might provide a function called "Age" that adds 1 year of life to the animals. It might also provide an abstract method called "IsDead" that, when called, will tell you if the animal has died. Since IsDead is abstract, each animal must implement it. So, a Cat might decide it is dead after it reaches 14 years of age, but a Duck might decide it dies after 5 years of age. The abstract class Animal provides the Age function to all classes that derive from it, but each of those classes has to implement IsDead on their own.

Now, an interface is like an abstract class, except it does not contain any logic. Rather, it specifies an interface. So, there might be an interface called IFly. This might have the methods GoForward and GoDown. Those methods would not actually contain any logic... each class that implements interface IFly would have to implement those GoForward and GoDown methods. You could have classes Duck and Finch implement interface IFly. Then, if you want to keep a list of instances that can fly, you just create a list that contains items of type IFly. That way, you can add Ducks and Finches and any other instance of a class the implements IFly to the list.

So, abstract classes can be used to consolidate and share functionality, while interfaces can be used to specify what the common functionality that will be shared between different instances will be, without actually building that functionality for them. Both can help you make your code smaller, just in different ways. There are other differences between interfaces andabstract classes, but those depend on the programming language, so I won't go into those other differences here.