Abstract V/s Interface

Abstract class well these are very similar to interfaces, you cannot instantiate them and they may contain a mix of methods declared with or without an implementation. However, with abstract classes you can declare fields that are not static and final and define public protected and private concrete methods, your abstract class can extend only one parent class, however it can implement multiple interfaces. Now when an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. However if it doesn't, then the subclass must also be declared abstract.

When abstract class is used??

You want to use an abstract class when firstly you want to share code among several closely related classes animal for example with fields, name, age and so on.

Or if you expect classes that extend your abstract class to have many common methods or fuels or required access modifiers other than public, so protected in private may also want to use an abstract class when you want to declare non static or non-final fields, for example a name and age so this then enables you to define methods that can access and modify the state of an object, you know get names set name and so forth.

Also you may want to use an abstract class when you've got a requirement for your base class to provide a default implementation of certain methods but other methods should be open to being overridden by child classes. And so, general summary the purpose of an abstract class is to provide a common definition of a base class that multiple derived classes can share.

Interfaces so an interface is just the declaration of methods of a class, it's not the implementation You know in an interface, we defined what kind of operation an object can perform. now these operations are defined by the classes that implement the interface, so interfaces form a contract between the class and the outside world and this contract is enforced at build time by the Java compiler. Now you cannot instantiate interfaces and they may contain a mix of methods declared with or without an implementation. Now all methods and interfaces are automatically public and abstract. Now an interface can extend another interface. Interfaces are more flexible and can deal with a lot more stress on the design of your program than the implementation.

interfaces primary purpose is abstraction decoupling the what from the hell and just a couple of general notes here, since Java 8 interfaces can now contain default methods, so in other words methods with implementation.

When to use Interface?

Firstly when you expect that unrelated classes will implement your interface, for example, the interface is comparable and cloneable are implemented by many unrelated classes. Now another reason to use an interface is when you want to specify the behavior of a particular datatype but you're not concerned about who implements its behavior, still another reason for using an interface if you want to separate different behavior.