Abstraction in Java:

->Abstraction is a process of hiding the implementation details and showing only functionality to the user.

->Another way, it shows only essential things to the user and hides the internal details,

for example, sending SMS where you type the text and send the message.

You don't know the internal processing about the message delivery.

Abstract class in Java:

->A class which is declared with the 'abstract' keyword is known as an abstract class in Java.

->It can have abstract and non-abstract methods (method with the body).

->It needs to be extended and its method implemented. It cannot be instantiated.

Points to Remember:

->An abstract class must be declared with an abstract keyword.

->It can have abstract and non-abstract methods.

->It cannot be instantiated.

->It can have constructors and static methods also.

->It can have final methods which will force the subclass not to change the body of the method.

Abstract Method in Java:

->A method which is declared as abstract and does not have implementation is known as an abstract method.

Example of abstract method:

---> abstract void printStatus();//no method body and abstract

Which should you use, abstract classes or interfaces?

Consider using abstract classes if any of these statements apply to your situation:

1.You want to share code among several closely related classes.

2.You expect that classes that extend your abstract class have many common methods or fields,

or require access modifiers other than public (such as protected and private).

3.You want to declare non-static or non-final fields.

This enables you to define methods that can access and modify the state of the object to which they belong.

Consider using interfaces if any of these statements apply to your situation:

1.You expect that unrelated classes would implement your interface.

For example, the interfaces Comparable and Cloneable are implemented by many unrelated classes.

2.You want to specify the behavior of a particular data type,

but not concerned about who implements its behavior.

3.You want to take advantage of multiple inheritance of type.