**Abstraction** is a process of hiding the implementation details and showing only functionality to the user. Another way, it shows only important things to the user and hides the internal details

**Example** : when you consider the case of e-mail, complex details such as what happens as soon as you send an e-mail, the protocol your e-mail server uses are hidden from the user. Therefore, to send an e-mail you just need to type the content, mention the address of the receiver, and click send.

A class which contains the **abstract** keyword in its declaration is known as **abstract** class.

* Abstract classes **may or may not** contain *abstract methods*, i.e., methods without body (public void get(); )
* But, if a class has at least one abstract method, then the class **must** be declared abstract.
* If a class is declared abstract, it cannot be instantiated. i.e. **we** **cannot create an object**.
* To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
* If you inherit(**Extend**) an abstract class, you have to provide implementations to **all the abstract** methods in it.
* A method defined abstract must always be redefined in the subclass, thus making [overriding](http://write.geeksforgeeks.org/overriding-in-java/) compulsory OR make subclass itself abstract.
* An abstract class can have parameterized constructors and default **constructor** is always present in an abstract class. Abstract class constructor will be invoked when child class object will be created
* ABSTRACT method can **have** **return** type

Key Word Abstract should come **first** in class and method declaration in order to let the compiler know that it is abstract. Ex. : abstract class aa { … } OR abstract void add() { … }