**When we go for Abstract and Interface**

**In real time how the abstraction and interface uses when we go for those concept in real time.**

Both abstract classes and interfaces are used when there is a difference in behaviour among the sub-types extending the abstract class or implementing the interface.

When the sub-types behaviour is totally different then you use an interface, when the sub-types behaviour is partially common and different with respect to the supertype an abstract class is used. In an abstract class the partially common behaviour is given a concrete implementation. Since there is no common behaviour between an interface and a sub-type an interface does not have an implementation for any of its behaviour.

*Lets assume you are developing a framework for a 'killer' competition where the people who participate have to demonstrate their killing skills. Members participating in the competetion have their own ways of killing.*

It could be as different as 'using a gun to kill', 'throwing the victim up in the air and kicking his balls when he victim comes down,  'slitting his throat or behaeding the victim'

The expected behaviour here is to 'KILL', though all the members do have that behaviour 'KILL" each member manifests the behaviour in a different way. Since the behaviour manifestation varies from member to member this scenario calls for an interface.

Interface KilingCompetition{   
  Corpse kill();   
}

Each member will implement this interface and give an implementation in his/her own way.

Now, lets consider a karate competition(is it called sparring?). Here each member has to follow certain things as laid out by the organizers or the karate protocol. It may include things like bowing to the opponent before and after the fight starts, taking a stand etc. Now these behaviours are common and has to manifested in the same way by every paticipating member. But the behaviour 'fight' will differ from member to member. So now we have a set of common behaviour and also a behaviour which manifests differently from member to member. this calls for an abstract class where you give implementation to the common behaviour and make the differeing behaviour abstract and hence the class abstract.

public abstract class KarateFight{   
      public void bowOpponent(){   
         //implementation for bowing which is common   
         // for every participant   
      }

      public void takeStand(){   
         //implementation which is common   
         // for every participant   
      }

      public abstract boolean fight(Opponent op);   
      //this is abstract because it differs from   
      // person to person   
}

**Vinay.**

**What is the difference between interface and abstract class?**

\* interface contains methods that must be abstract; abstract class may contain concrete methods.   
\* interface contains variables that must be static and final; abstract class may contain non-final and final variables.   
\* members in an interface are public by default, abstract class may contain non-public members.   
\* interface is used to "implements"; whereas abstract class is used to "extends".   
\* interface can be used to achieve multiple inheritance; abstract class can be used as a single inheritance.   
\* interface can "extends" another interface, abstract class can "extends" another class and "implements" multiple interfaces.   
\* interface is absolutely abstract; abstract class can be invoked if a main() exists.   
\* interface is more flexible than abstract class because one class can only "extends" one super class, but "implements" multiple interfaces.   
\* If given a choice, use interface instead of abstract class.