Abstract method & class :-

A method without body is called abstract method

```
abstract returntype name(---);
```

If a class containing any abstract method then the class is said to be abstract class and it should be declared as abstract.

```
abstract class name
{
   abstract returntype name(=);
}
```

```
public abstract class AbDemo
{
  void rose()
  {
    System.out.println("I am rose");
  }
  abstract void lilly();
}
public class AbDemoEx extends AbDemo
{
  void lilly()
  {
    System.out.println("I am lilly");
  }
  void flower()
  {
    System.out.println("I am flower");
  }
  public static void main(String[] args)
  {
   AbDemoEx a = new AbDemoEx();
   a.rose(); a.lilly(); a.flower();
  }
}
```

Differences between concrete and abstract class

concrete	abstract
1. It doesnot contain any abstract methods	1. It contains abstract methods.
2. Sub-class may or may not override superclass methods	2. Sub-class must override abstract methods of super class.
3. It can be instantiated	3. It cannot be instantiated

Similarities :-

1. Both are classes.
2. Both contains concrete methods
3. Both are extended by using extends.
4. We can't extend more than one abstract class as well as concrete class. i.e multiple inheritance is not possible in java through classes

Interface :-

```
interface name
{
    abstract method
}
```

* It contains only abstract methods. It shows functionalities and hides implementation.
* It is a complete future framework.
* ~~Subclass~~ It is implemented by using keywords implements.
* Subclass must implement abstract methods of interface.

* Interface cannot be instantiated.

```
public interface varun
{
    abstract void coke();
    abstract void pepsi();
}
public class varunC implements varun
{
  public void coke()
  {
    System.out.println("coke");
  }
  public void pepsi()
  {
    System.out.println("pepsi");
  }
  public void sting()
  {
    System.out.println("string");
  }
  public static void main(String[] args)
  {
    varunC v = new varunC();
    v.coke(); v.pepsi(); v.sting();
  }
}
```

* we can implement any number of interfaces i.e multiple inheritance is possible in java through interface.

```
public interface varun
{
    abstract void coke();
}
public interface Shiva
{
    abstract void pepsi();
}
public class VarunC implements varun, Shiva
{
    public void coke()
    {
        System.out.println("coke");
    }
    public void pepsi()
    {
        System.out.println("pepsi");
    }
    public void sting()
    {
        System.out.println("sting");
    }
    public static void main(String[] args)
    {
        VarunC v = new VarunC();
        v.coke(); v.pepsi(); v.sting();
```

Differences between concrete class and interface

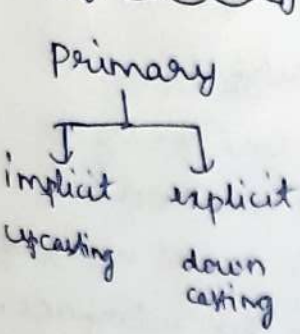
concrete class	Interface
1. It does not contain any abstract methods.	1. It contains only abstract methods.
2. Sub-class may or may not override superclass methods.	2. Sub-class must implement implemented interface.
3. we cannot extend more than one class. Multiple inheritance is not possible in java through classes	3. We can implement any no. of interfaces. i.e Mul Inheri is possible in java through interfaces.
4. public abstract is not default for methods	4. public abstract is default for methods.
5. public static final is not default for members.	5. public static final is default for members.
6. It can be instantiated	6. It cannot be instantiated.

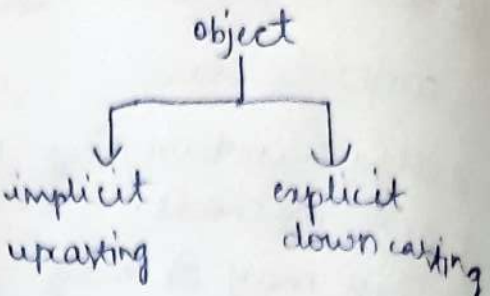
abstract class	Interface
1. It contains concrete as well as abstract methods.	1. It contains only abstract methods.
2. we cannot extend more than one abstract class. ie Multi. Inhe is not possible through abstract classes	2. we can implement any no. of interfaces. ie M.I is possible in java through interfaces.
3. public abstract is not default for methods	3. public abstract is default for methods.
4. public static final is not default for members.	4. public static final is default for members.

Similarities b/w abstract class & Interface

1. Both contains abstract methods.
2. Both cannot be instantiated.
3. Sub class must override abstract methods & declare subclass as abstract.
4. Both supports dynamic binding.

type casting






```
public interface Add
{
        void sub();
}
public class Sreeja implements Add
{
    public void sub();
    {
      System.out.println("Sreeja class");
    }
}
public class Siit implements Add
{
   public void sub();
   {
     system.out.println("Siit class");
   }
  public static void main (String[] args)
  {
      Add a;
      a = new Sreeja();
      a.sub();
      a = new Siit();
      a.sub();
  }
}
```