# Abstract Methods and Abstract Classes

- ***Non abstract method*** is methods which is having body and implementation

  **Example**

  ```
  int sum(int x,int y)  //non abstract method
  { int s;
    s=x+y;
  retrun s; }
  ```

- ***Null body method*** is a methods which is having body. But doesn't have any implementation.every null body method acts as non abstract method.but non abstract methods is not null body method

  **Example:**

  ```
  void sum(int x,int y)  //null body method
  { }
  ```

- ***Abstract method*** is method which is not having body, just it is a declaration

  **Example:**

  ```
  abstract void sum(int x,int y);  //abstract method
  ```

  **Examples:**

  ```
  int sum(int x,int y)  //non abstract method
  {
        int s;
        s=x+y;
        return s;
  }
  ```

```
void sub(int x,int y)  //null body mtd
        {  }

abstract void mul(int x,int y); //abstract mtd
```

- In the class we can define only non abstract methods, but not abstract methods

Example:

```
class Sample
{
        int sum(int x,int y)  //non abstract method
        {
                int s;
                s=x+y;
                return s;
        }

        void sub(int x,int y)  //null body mtd
                {  }
}
```

- Abstract class is the collection of both abstract or non abstract methods
- It is not compulsory to have an abstract method in abstract class
- In the abstract class we define only non abstract methods or we define only abstract methods or we can define both abstract or non abstract methods

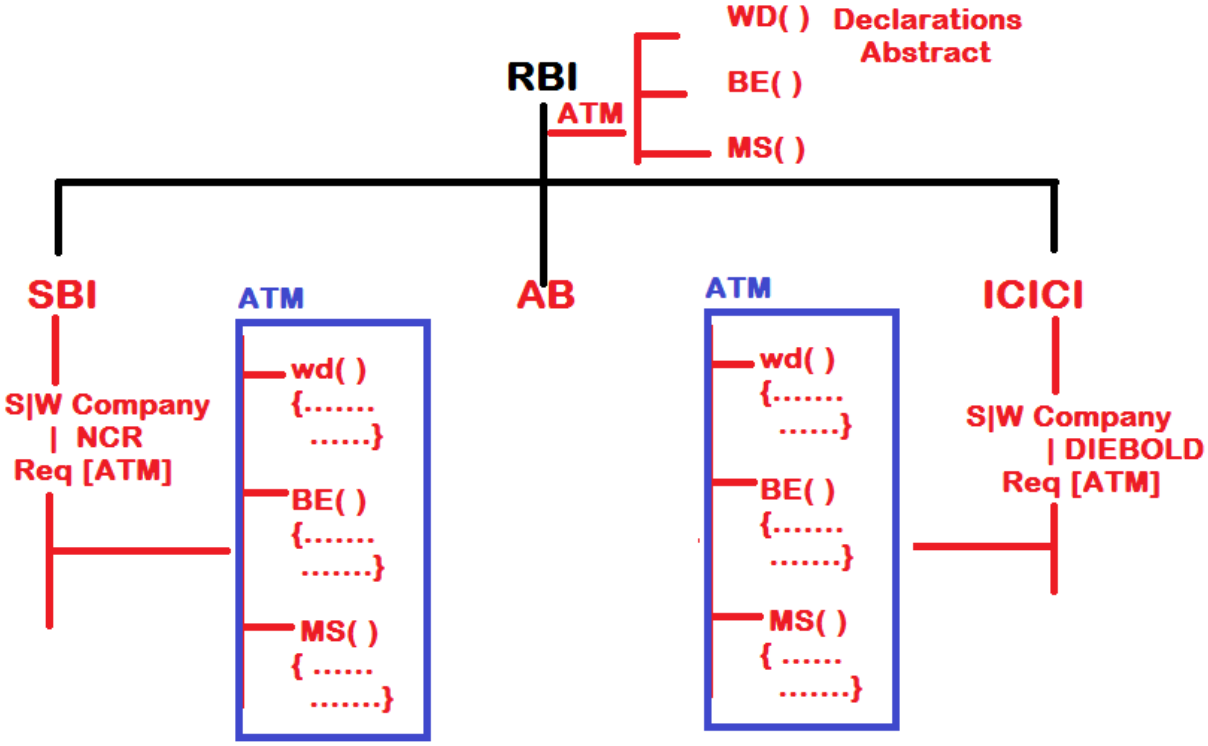
**Example:**

```
abstract class Sample
{
        int sum(int x,int y)  //non abstract method
        {
                int s;
                s=x+y;
                return s;
        }

        void sub(int x,int y)  //null body mtd
                {  }

        abstract void mul(int x,int y);
}
```

**When do we abstract methods and Abstract classes ?**

- **Whenever two or more sub classes required to fulfill the same role through different implementation**

**Example:**




**Samples:**

```
abstract class Super    //valid
{
        void method1()  //null body | non abstract mtd
        {}
}
```

Example 2:

```
//2.java
class Sample   //CE
{
        void method1()
        {}
        abstract void method2();
}
```

Example 3:

```
abstract class Sample  //valid
{
        abstract void method2();
}
```

**Example 4:**

```
abstract class Sample   //valid
{
        abstract void method2();
        void method1()
        {}
}
```

**Example 5:**

```
abstract class Sample
{
        abstract void method2();
        void method1()
        {}
}
class Testing extends Sample   //CE
{
}
```



**If any class extended by abstract class with abstract methods the we must declare sub class as an abstract otherwise we have to override all the abstract methods of super class in the sub class**

**Example:**

```
abstract class Sample  //valid
{
        abstract void method2();
        void method1()
        {}
}
class Testing extends Sample
{
        void method2( )
        { }
}
class Demo
{
  public static void main(String args[ ])
   { Sample s=new Sample( ); }
}
```

**Sample is abstract; cannot be instantiated**

```
        Sample s=new Sample();
```

**Note : Creating an object for an abstract is nothing but creating an object any of its Concrete class**

- **Concrete class is a class which is overridden all the abstract methods of its super class**
- **Every Concrete class is subclass , But Every subclass is not Concrete class**

```
abstract class Test
{
        void method1()  //non abstract mtd
        { System.out.println("M1 of Test "); }

        abstract void method2();
}
class Testing extends Test
{
        @Override
        void method2()
        { System.out.println("OR M2 of Test "); }

        void method3()
        { System.out.println("M3 of Testing "); }
}
class AbsEx1{
        public static void main(String args[]){
                Test t=new Testing();
                        t.method1();
                                t.method2();
                                //  t.method3();   CE

                Testing  t2=new Testing( );
                        t2.method3( );
                                t2.method2( );
                                t2.method1( );
        }
}
```

**Example 2:**

```
abstract class Shapes
{
        float dim1,dim2;  //instance fields

        void setShapes(float dim1,float dim2)
        { this.dim1=dim1; this.dim2=dim2; }

        abstract float findArea();
}
class Triangle extends Shapes
{
        @Override
        float findArea()
        { return (0.5f*dim1*dim2); }
}

class Rect extends Shapes{
        @Override
        float findArea()
        { return (dim1*dim2); }
}

class AbsMain
{
   public static void main(String args[])
        {
                Shapes s=new Triangle( );
                        s.setShapes(4.0f,4.0f);
                 float at=s.findArea( );
```

```
            System.out.println("Area of Traingle : "+at);

                s=new Rect( );
                        s.setShapes(5.0f,5.0f);
                float ar=s.findArea( );
                System.out.println("Area of Rect : "+ar);
        }
}
```