Abstract class and abstract methods :

A class that does not provide complete implementation (partial implementation) is defined as an abstract class.

An abstract method is a common/generic method which is used to provide easiness to the programmer because the programmer faces complexity to remember the method name.

An abstract method observation is very simple because every abstract method contains abstract keyword, abstract method does not contain any method body and at the end there must be a terminator i.e ; (semicolon)

In java, whenever action is common but implementations are different then we should use abstract method, Generally we declare abstract method in the super class and its implementation must be provided in the sub classes.

If a class contains at least one method as an abstract method then we should compulsory declare that class as an abstract class.

Once a class is declared as an abstract class we can't create an object for that abstract class.

*All the abstract methods declared in the super class must be overridden in the sub classes otherwise the sub class will become as an abstract class hence object can't be created for the sub class as well.

In an abstract class we can write all abstract methods or all concrete methods or combination of both the method.

It is used to acheive partial abstraction that means by using abstract classes we can acheive partial abstraction(0-100%).

*An abstract class may or may not have abstract method but an abstract method must have abstract class.

Note :- We can't declare an abstract method as final, private and static (illegal combination of modifiers)

We can't declare an abstract class as a final.

//Program :

```
abstract class Shape
{
    public abstract void draw();
}
class Circle extends Shape
{
    @Override
    public void draw()
    {
        IO.println("Drawing Circle");
    }
}
class Square extends Shape
{
    @Override
    public void draw()
    {
        IO.println("Drawing Square");
    }
}

public class AbstractDemo1
{
    public static void main(String[] args)
    {
        Shape s;

        s = new Circle(); s.draw();
        s = new Square(); s.draw();
    }
}
```

IQ :

```
abstract class Car
{
    protected int speed = 120;

  public Car()
    {
        IO.println("Car class constructor");
    }

  public void getCarDetails()
    {
        IO.println("Car has one engine 4 wheels");
    }

    public abstract void run();
}

class BMW extends Car
{
  @Override
  public void run()
    {
        IO.println("BMW Car is running");
    }
}

public class AbstractDemo2IQ
{
    public static void main(String[] args)
    {
        Car car = new BMW();
        IO.println("BMW speed is :"+car.speed);
        car.getCarDetails();
        car.run();

    }
}
```

Note : abstract class constructor will be executed through sub class object by using super()

IQ :

What is the advantage of writing instance variable(Object properties/ NSF), constructor and non static block inside an abstract class, If we can't create an object for abstract class ?

Yes, We can't create an object for abstract class but still we can write instance variable, constructor and non static block that means
abstract class also contains Object properties and behavior but these properties will initialize through SUB CLASS OBJECT -using super() so we can write object properties, non static block as well as constructor inside an abstract class.

-WAP to show that overriding is compulsory for abstract method.

```
package com.ravi.linked_list;

abstract class Alpha
{
    public abstract void show();
    public abstract void demo();
}

abstract class Beta extends Alpha
{
    @Override
    public void show() //= demo();
    {
        IO.println("show method is overridden in Beta class");
    }
}

class Gamma extends Beta
{
    @Override
    public void demo()
    {
        IO.println("demo method is overridden in Gamma class");
    }
}

public class Main
{
    public static void main(String[] args)
    {
        Gamma gamma = new Gamma();
        gamma.demo();
        gamma.show();
    }

}
```

Note : All the abstract methods declared in the super class must be overridden in sub classes

class final -> Inheritance is not possible
Method final -> Overriding is not possible
Field is final -> re-assignemnt is not possible

Offline class

a) interface
b) Array
c) String
d) Exception Handling
e) Collections Framework + Stream API
f) Multithreading
g) HAS-A Relation

Online Class (Sunday 9:30AM to 11:00AM)

1) Object class and its methods [11 methods]
2) final keyword
3) Inner classes in java
4) Input and Output + file Handling

* Serialization (Input & Output)
* Generics (Collection)
* Java 8 features [Functional Interface]