Abstraction:

1. A **Class** which is declared with the **Abstract keyword** is known as an **Abstract class.**
2. It can have **abstract and non-abstract Methods.** (method with the body).

3. Inside **Abstract Class** we can take **Static** and **Instance Blocks**, but **Static Block** is executed in **Abstract Class**, but **Instance Block** is **not executed** because **we cant create instance** in **Abstract Class.**

4. Inside **Abstract Class** we can take **Constructor**, but you **Cannot Instantiate** it.

5. In Abstract Class we don't have any body, Just we have only **Method Signature.**

6. In Abstract Class can declare **Instance Methods** but **instance Creation** is not possible, but static methods are possible and callable in Abstract Class

7. By default **abstract methods are public abstract**

8. In Abstract class we can have **instance variables, static variables and local variables,** but static variables and local variables are executed in abstract class but not instance variables

1. In Abstract Class we can declare **multiple abstract methods**, but we can also provide **only single method implementation of sub class** and all other remaining implementations we can take in **another class by using multiple inheritance**

2. We can **extend non abstract class** from **abstract class** using **multiple inheritance**

3. only **public, protected, abstract, default, synchronized, static and strictfp methods are permitted** in abstract class

4. Inside abstract class we can have **all non abstract method** but implementation is required in sub class

5. We cant make **abstract method as static**

6. We cant declare **abstract method as private**

Inside abstract class we can have abstract methods and non abstract methods

```
abstract class A {
public void m1() {
System.out.println("M1 Method of A");
}

//By default abstract methods are public
public abstract void m2();
}

class B extends A{
@Override
public void m2() {
System.out.println("M2 Method of B");
}
}

public class Client {
public static void main(String[] args) {
A  a = new B();
a.m1();
a.m2();
}
}
```

Inside abstract class we can have abstract methods and non abstract methods

```
public abstract class Profile {

public void m1() {
System.out.println("M1 Instance Method ");
}

public static void m2() {
System.out.println("M2 Static Method");
}

public int m3() {
System.out.println("M3 Primitive Type
Method");
return 0;
}

public String m4() {
System.out.println("M4 String Method");
return null;
}

public abstract void m5();
public abstract void m6();

}
```

```
public class ProfileImpl extends Profile {

@Override
public void m5() {
System.out.println("Abstract M5 Method");
}

@Override
public void m6() {

System.out.println("Abstract M6 Method");
}

public static void main(String[] args) {

ProfileImpl impl = new ProfileImpl();
impl.m5();
impl.m6();
impl.m1();
ProfileImpl.m2();
impl.m3();
impl.m4();
}
}
```

```
Abstract M5 Method
Abstract M6 Method
M1 Instance Method
M2 Static Method
M3 Primitive Type
Method
M4 String Method
```

Inside Abstract Class we can take Static and Instance Blocks, but Static Block is executed in Abstract Class,
but Instance Block is not executed because we cant create instance in Abstract Class.

```
public class Client {

static {
System.out.println("Static Block");
}

{
System.out.println("Instance Block");
}

public static void main(String[] args) {

System.out.println("Main Method");

}
}
```

In Abstract class we can have **instance variables, static variables and local variables,** but static variables and local variables are executed in abstract class but not instance variables

```
package com.dl.nine;

public abstract  class Client {

int i = 10;
int j = 20;

static int x = 100;
static int y = 200;

public static void main(String[] args) {

int a = 1000;
int b = 2000;
System.out.println("Local Variables: " + (a + b));
System.out.println("Static Variables: " + (x + y));

//System.out.println(new Client().i); // Cannot instantiate the type Client
//System.out.println(new Client().j); // Cannot instantiate the type Client
}
}
```

```
Local Variables: 3000
Static Variables: 300
```

Inside Abstract Class we can take Constructor, but you Cannot Instantiate it.

```
abstract class A {

public A() {
System.out.println("Constructor of Class A");
}
}

class B extends A {

public B() {
System.out.println("Constructor of Class B");
}
}

public class Client {
public static void main(String[] args) {

new B();
}
}
```

```
Constructor of Class A
Constructor of Class B
```

In Abstract Class we can declare **multiple abstract methods**, but we can also provide **only single method implementation of sub class** and all other remaining implementations we can take in **another class by using multiple inheritance**

```
abstract class A {

public abstract void m1();

public abstract void m2();

}
```

```
abstract class B extends A {

public void m1() {
System.out.println("M1 Mehtod of Class B");
}

}
```

```
class C extends B {

@Override
public void m2() {
System.out.println("M2 Method of Class C");
}

}
```

```
public class Client {

public static void main(String[] args) {

A a = new C();
a.m1();
a.m2();
}

}
```

```
M1 Mehtod of Class B
M2 Method of Class C
```

We can **extend non abstract class** from **abstract class** using **multiple inheritance**

```
class A {

public void m1() {
System.out.println("M1 Method of Class A");
}

}
```

```
abstract class B extends A {

public void m2() {
System.out.println("M2 Method of Class B");
}

public abstract void m3();
}
```

```
class C extends B {

@Override
public void m3() {
System.out.println("M3 Method of Class C");
}

}
```

```
public class Client {

public static void main(String[] args) {

C c = new C();
c.m1();
c.m2();
c.m3();
}
}
```

```
M1 Method of Class A
M2 Method of Class B
M3 Method of Class C
```

We can declare private variables and private methods in Abstract class, but only static variable and method are implemented

```
public abstract class  Client {
private int a = 10;
private void m1() {
System.out.println("Non Abstrcat Method M1 in Abstract Class");
}

private static int b = 20;
private static void m2() {
System.out.println("Non Abstrcat Method M2 in Abstract Class ");
}

public static void main(String[] args) {

int c = 30;
System.out.println("Local Variables in Abstract Class: " + c);

System.out.println("Private Variables in Abstract Class: " + Client.b);
Client.m2();

// new Client().m1(); // Cannot instantiate the type Client
// System.out.println(new Client().a); // Cannot instantiate the type Client
}
}
```

```
public abstract class Product {

static int productPrice;
static int discount;

public void getDiscount(int offer) {
discount = productPrice - offer;
System.out.println("After discount Product Price is: " + discount);
}

public void getUserDetails(String fName, String lName, String productName) {
System.out.println("User FirstName: " + fName);
System.out.println("User LastName: " + lName);
System.out.println("Product Name : " + productName);
}
}
```

```
public class ProductImpl extends Product {
public static void main(String[] args) {

ProductImpl impl = new ProductImpl();
impl.getUserDetails("Sai", "Kiran", "Samsung");
ProductImpl.productPrice = 12000;
impl.getDiscount(2000);
}
}
```