Abstraction – hiding the implementation and showing functionality

Two ways to achieve abstraction –

1. Using the abstract class

2. Using interface

Abstract class – a class which is declared with the keyword "abstract" is called abstract class

- Must be declared with abstract keyword
- Can have abstract and non-abstract method
- It cannot be instantiated
- It can have the constructor  n static method
- It can have final method so subclass cannot make changes in it

```java
package OOPS;


abstract class UI{

	abstract void m1();

	static void m2() {

		System.out.println("I am Static method");
	}
	final void m3() {
		System.out.println("I am final method");
	}
}
public class Abstraction extends UI {

	// hiding the implementation & showing only functionality

	//1. abstract class
	//2. interface
	@Override
	void m1() {
		System.out.println("Safe Data");

	}


	public static void main(String[] args) {
		Abstraction a = new Abstraction();
		a.m1();
		m2();
		a.m3();

	}
}



package OOPS;

abstract class TCS{

	abstract void telecalling();
}

class C1 extends TCS{

	void telecalling() {
		System.out.println("Jio");
	}

}

class C2 extends TCS{

	void telecalling() {
		System.out.println("Airtel");
	}
}

class C3 extends TCS{

	void telecalling() {
		System.out.println("BSNL");
	}

}



public class Client {

	public static void main(String[] args) {
		C1 obj = new C1();
		obj.telecalling();

		C2 obj1 = new C2();
		obj1.telecalling();
		C3 obj2 = new C3();
		obj2.telecalling();

	}

}
```

If you are inheriting a class inside a class or an interface inside an interface then you have to use extends keyword
And if you are inheriting an interface inside a class then you have to use implements keyword

If you are inheriting an abstract class then you must implement the abstract method declared in that abstract class.

```
package OOPS;

abstract class whatsappUI{

	public whatsappUI() {
		System.out.println("Welcome to the whatsap");
	}

	abstract void Sendmsg();

	static void profile() {

		System.out.println("Profile pic");

	}

	final void message() {

		System.out.println(" Type message");

	}

	void data() {
		System.out.println("Whatsap images, docs");
	}
}


public class Whatsapp  extends whatsappUI{
	void Sendmsg() {

		System.out.println(" The message is being sent through a 
SFTP protocol");
	}

	public static void main(String[] args) {


	}

}
```

Abstraction using an interface –

```
package OOPS;

public interface Instagram {

	 void story();
	 void post();
	 void reels();
	 package OOPS;

public class LogicInsta implements Instagram {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

	}

	@Override
	public void story() {
		// TODO Auto-generated method stub

	}

	@Override
	public void post() {
		// TODO Auto-generated method stub

	}

	@Override
	public void reels() {
		// TODO Auto-generated method stub
```

```
    }

}

     static void profile() {

     }



}
```

Using interface you can achieve 100% abstraction because there is no implementation of any method.

If you are inheriting the interface in an abstract class then you won’t need to implement the unimplemented methods compulsorily.
Example –

```
package OOPS;

public interface InterFA {

    void f1();
    abstract void f2();

}
package OOPS;

public abstract class AbsA implements InterFA{

    public static void main(String[] args) {

    }

}
```

But if you are inheriting the interface in a normal class then you must implement the unimplemented methods.

```
Interface calculator --→  abstract class CalC --→ CalCLogic
Add, sub,mul, div              add                      sub, mul, div
```

If an interface is having 4 methods and then the interface is inherited by an abstract method then it is not necessary that it implements the method, it can and also if does want then it can leave.

But if then this abstract method is being inherited by any other class then that class must implement the unimplemented methods.

```java
package OOPS;

public interface Calculator {

	void addition();
	void substraction();
	void multiplication();
	void division();

}


package OOPS;

public abstract class  CalC implements Calculator{

	public void addition() {
		// TODO Auto-generated method stub

	}


	public static void main(String[] args) {

	}

}

package OOPS;

public class CalCLogic extends CalC {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
	}


	@Override
	public void substraction() {


	}

	@Override
	public void multiplication() {
		// TODO Auto-generated method stub

	}

	@Override
	public void division() {
		// TODO Auto-generated method stub

	}
}
```