if you declare a interface we don't require to explicitly mention methods as abstract. they are by default abstract

Interface :- It is a collection of final static variables and public abstract methods.

any class implementing an interface must override abstract methods else class becomes abstract class.

with the help of interface we can ~~~~ implement multiple inheritance.

Java supports default and static methods from JDK 1.8

an Interface can extend another interface like class.

```
interface i1 {
    void sum(int a, int b);
}
class Isum implements i1 {
    Public void sum(int a, int b) {
        Sop(a+B);
    }
    Public Static Void main(String[] args) {
        ISum obj = new ISum();
        obj.sum(10,6);
    }
}
```

```
interface i1 {
    void sum(int a, int B);
}

interface i2 {
    void div(int a, int b)
}
```

JFrame is in Swing

app by JFrame


list updation in 3rd by Thread class

Problem: we can't extend JFrame, Thread together.

Solution: we have runnable interface.

```
class ISum2 implements i1, i2 {
    Public void Sum(int a, int b) {
        SOP(a+b);
    }
    Public void div(int a, int b) {
        SOP(a/1);
    }
    Public static void main(String[] arg) {
        ISum2 obj = new ISum2();
        obj.Sum(10,6);
        obj.div(10,5);
    }
}
```

So, class F extend JFrame implements runnable

★ Functional Interface: An Interface with only one method is called Functional interface.

★ Marker Interface: An Interface with no method is called marker Interface.

★ Interface vs Class:

- With class we can only implement single inheritance but with interface we can achive multiple inheritance
- In class only one class property get inherited with multilevel inheritance but with interface we have several interface properties

always use interface over class
↳ it is better

default keyword is not the default access modifier

* • When we create anonomius class with functional interface using lamdba expression no byte code is genrated. but with abstract class byte code genrated (B$1.class, B$2.class)

*Default method in interface : (default method don't need to be overrided but we can if we want to)

```
interface Id1 {
    default void show() {
        S.O.P.("Default method of interface");
    }
    void display();
}

class Id1Demo implements Id1 {
    Public void display() {
        SOP.("display overrid");
    }
    public Static void main(String[] args) {
        Id1Demo obj = new Id1Demo();
        obj.show();
        obj.display();
    }
}
```

(optional)

```
Public void show() {
    SOP("Show overrided");
}
```

can add it here if we want.

we can declare default methods in interface using default keyword. Any class implementing an interface with default method either override the default method or leave it unoverrided.

* i extends i;
c implements i;

```
interface i1 {
    void show();
}

interface i2 extends i1 {
    void display();
}

class DemoI2 implements i2 {
    Public void show() {
        S.o.p.("Show From i1");
    }
    Public void display() {
        S.o.p.("Display from i2");
    }
    Public Static void main(String S[]) {
        DemoZI obj = new DemoZI();
        obj.show();
        obj.display();
    }
}
```

* Static methods in Interface:

We don't override static methods as both methods will get different reference in memory.

```
interface Smi {

    Public static void max(int a, int b) {
        if(a>b)
            S.O.p("max = "+a);
        else
            S.O.p("max = "+b);
    }
}
```

Smi

```
Class DemoSmi implements Smi {
    Public static void main(String[] args) {
        Smi.max(10,7);
        DemoSmi.max(10,6);
    }
```

We can access of interface with interface name as well as the class name which implements that interface in which static method is present.