**Interface**: It is a blueprint of a class which defines the rules that a class has to implement.

Interface is an entity through which we can achieve 100% abstract.

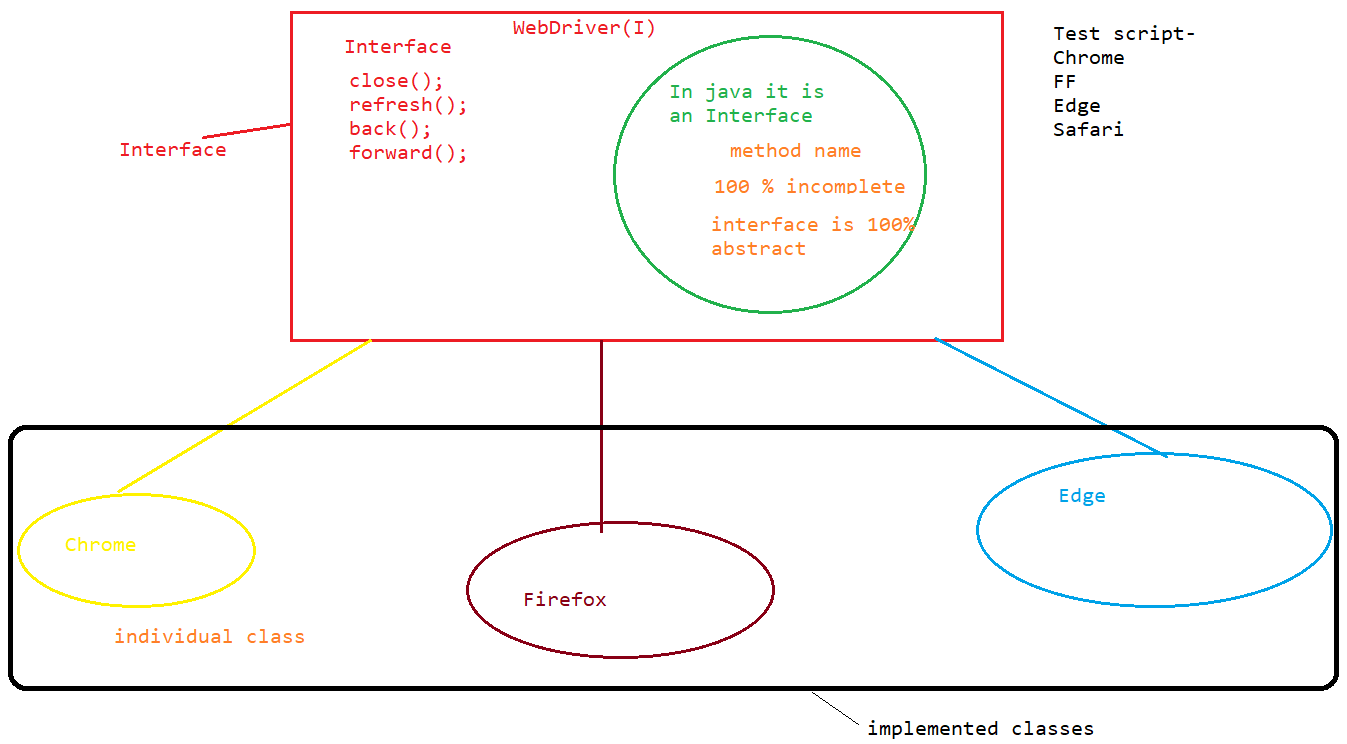
To provide the implementation inside a class we have to write implements keyword.

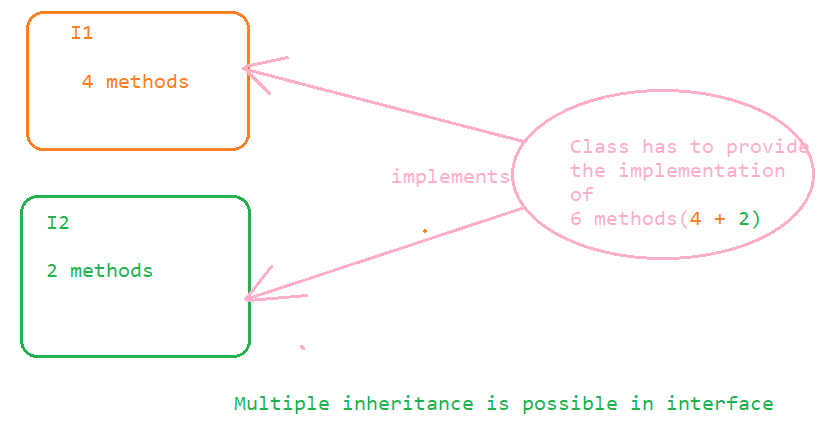
We cannot create an object of interface. But we can create a reference variable of interface.

Example:

Interface1 i1 = **new** Browser1();

A class can implements multiple interface at the same time i.e multiple inheritance is possible in terms of interface.





Through this diamond ambiguity problem gets resolve after using interface.

Example:

**public** **interface** Interface1 {

**void** close();

**public** **abstract** **void** back();

**public** **void** refresh();

**public** **interface** Interface2 {

**public** **void** m1();

**public** **void** m2();

**public** **void** m3();

**public** **void** m4();

}**public** **class** Test2 **implements** Interface2, Interface1 {

**public** **void** close() {

}

**public** **void** back() {

}

**public** **void** refresh() {

}

**public** **void** m1() {

}

**public** **void** m2() {

}

**public** **void** m3() {

}

**public** **void** m4() {

}

}

**Methods in an Interface:**

1. Every method(non-static) inside an interface is by default public and abstract whether we declare it or not.

Example:

**package** interfacediscussion;

**public** **interface** Interface1 {

**void** close();

**public** **abstract** **void** back();

**public** **void** refresh();

// all the above methods are having the same category i.e public and abstract.

}**public** **class** Browser1 **implements** Interface1 {

**public** **void** close() {

System.***out***.println("close method of browser 1");

}

**public** **void** back() {

System.***out***.println("back method of browser 1");

}

**public** **void** refresh() {

System.***out***.println("refresh method of browser 1");

}

}

2. Inside an interface we can only have complete static method inside it. All the static method inside an interface is by default public and concrete.

Example:

**public** **interface** Interface1 {

**static** **void** m1()

{

System.***out***.println("M1 static method");

}

**public** **static** **void** main(String[] args) {

*m1*();

Interface1.*m1*();

}

}

In the above example of static method it is by default public whether we declare it or not.

Variables inside an interface:

By default all the variables inside an interface are public static and final whether we define them or not.

Example:

**public** **interface** VariablesInInterface {

**public** **static** **final** String ***s*** = "abc";// static String s = "abc" // final String s = "abc"

// Above variables are the same one

**boolean** ***k*** = **false**;

**public** **static** **void** main(String[] args) {

System.***out***.println(***s***);

}

Note: In the above example s variable is public static and final whether we declare it or not.