**Marker Interface**

It is an empty interface (no field or methods). Examples of marker interface are Serializable, Cloneable and Remote interface. All these interfaces are empty interfaces.

public interface Serializable

{

// nothing here

}

Examples of Marker Interface which are used in real-time applications : 

**Cloneable interface :** Cloneable interface is present in java.lang package. There is a method clone() in [Object](https://www.geeksforgeeks.org/object-class-in-java/) class. A class that implements the Cloneable interface indicates that it is legal for clone() method to make a field-for-field copy of instances of that class.   
Invoking Object’s clone method on an instance of the class that does not implement the Cloneable interface results in an exception CloneNotSupportedException being thrown. By convention, classes that implement this interface should override Object.clone() method.   
Refer [here](https://www.geeksforgeeks.org/clone-method-in-java-2/) for more details.

**Program:**

package JavaConceptsMarkerInterface;

class A implements Cloneable{

int i;

String s;

// A class constructor

public A(int i,String s)

{

this.i = i;

this.s = s;

}

// Overriding clone() method

// by simply calling Object class

// clone() method.

@Override

protected Object clone()

throws CloneNotSupportedException

{

return super.clone();

}

}

public class MCloneableInterfaceEx1 {

public static void main(String[] args) throws CloneNotSupportedException {

A a = new A(20, "GeeksForGeeks");

// cloning 'a' and holding

// new cloned object reference in b

// down-casting as clone() return type is Object

A b = (A)a.clone();

System.out.println(b.i);

System.out.println(b.s);

}

}

**Serializable interface** : Serializable interface is present in java.io package. It is used to make an object eligible for saving its state into a file. This is called [Serialization](https://www.geeksforgeeks.org/serialization-in-java/).   
Classes that do not implement this interface will not have any of their state serialized or deserialized. All subtypes of a serializable class are themselves serializable.

package JavaConceptsMarkerInterface;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

class AB implements Serializable {

int i;

String s;

// A class constructor

public AB(int i, String s) {

this.i = i;

this.s = s;

}

}

public class SerializaleInterfaceEx2 {

public static void main(String[] args) throws IOException, ClassNotFoundException {

AB a = new AB(20,"GeeksForGeeks");

// Serializing 'a'

FileOutputStream fos = new FileOutputStream("xyz.txt");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(a);

// De-serializing 'a'

FileInputStream fis = new FileInputStream("xyz.txt");

ObjectInputStream ois = new ObjectInputStream(fis);

AB b = (AB)ois.readObject();//down-casting object

System.out.println(b.i+" "+b.s);

// closing streams

oos.close();

ois.close();

}

}