What is Marker interface in Java and why required? Answer

The marker interface in Java interfaces with no field or methods or, in simple words **empty interface in java is called a marker interface**. Example of marker interface is Serializable, Cloneable, and Remote interface. If the marker interface doesn't have any field or method, or behavior, why would Java need it? This was asked to one of my friends in one of his core java interviews, and then I thought to touch base on it. In this article, we will see a couple of reasons for **what marker interfaces do in Java** and **the use of marker interfaces in Java**. A marker interface is also called a **tag interface in Java**.

What is Marker interfaces in Java, and why required

Now, let's understand the marker interface in Java, some common examples of marker interfaces, and what benefits they provide.

1. What Marker or Tag interface do in Java?

Looking carefully at marker interfaces in Java, like, **Serializable, Cloneable, and Remote,** it looks they are **used to indicate something to compiler or JVM**. So if JVM sees a Class is Serializable, it has done some special operation on it, a similar way, if JVM sees one Class is implement Cloneable, it performs some operation to support cloning.

The same is true for **RMI and Remote interface**. So, in short, the Marker interface indicates a signal or a command to Compiler or JVM.

And, If you are new to the Java world, then I also recommend you go through **these Java programming courses** to learn Java in a better and more structured way. This is one of the best and up-to-date courses to learn Java online.

This is a pretty standard answer to the question about the *marker interface. Once* you give this answer, most of the time interviewee definitely asked, "**Why this indication can not be done using a flag inside a class**?" this makes sense, right?

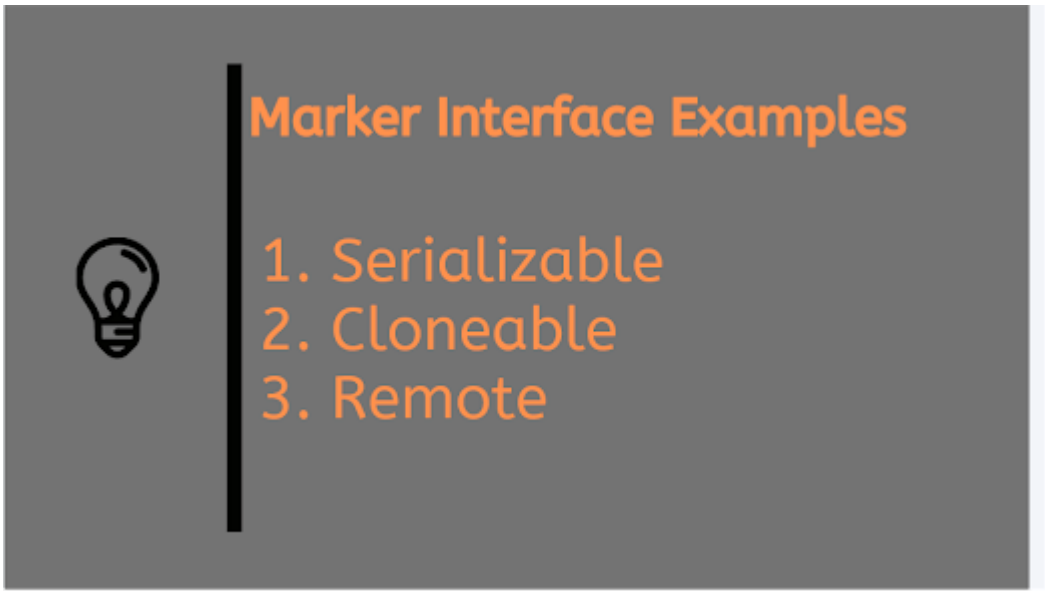
Yes, this can also be done by using a boolean flag or a String. Still, it doesn't mark a class like Serializable or Cloneable, makes it more readable, and it also allows

one to take advantage of Polymorphism in Java.

Where Should I use the Marker interface in Java?

Apart from using a built-in marker interface for making a class Serializable or Cloneable. One can also develop his own marker interface. The marker interface is a good way to classify code. You can create a marker interface to logically divide your code and if you have your own tool, you can perform some pre-processing operations on those classes. Particularly useful for developing API and frameworks like Spring or Struts.

After introducing Annotation on Java5, Annotation is a better choice than the marker interface, and JUnit is a perfect example of using Annotation, e.g**., @Test, for specifying a Test Class**. The same can also be achieved by using a Test marker interface.



Another use of marker interface in Java

One more use of marker interface in Java can be commenting. A marker interface called ThreadSafe can communicate to other developers that classes implementing this marker interface give a thread-safe guarantee, and any modification should not violate that. The marker interface can also help code coverage or a code review tool to find bugs based on specified behavior of marker interfaces.

Again Annotations are a better choice `@ThreadSafe` looks a lot better than implementing the ThraedSafe marker interface.

In summary, the **marker interface in Java is used to indicate something to the compiler, JVM,** or any other tool, but **Annotation** is a better way of doing the same thing.