Difference between Abstract class and Interface in Java 8? Answer

Ever since JDK 8 has allowed concrete (non-abstract) methods on the interface like default and static methods, many of my readers have asked me how should they answer the classical abstract class vs interface questions. Earlier, an interface cannot have any concrete methods and that was the main difference between abstract class and interface but now that is not the case. In this post, I'll revisit this hugely popular Java interview question in light of Java 8 changes. This is also a popular Java interview question and knowing the difference will help you to answer this question in a real interview.

As I said, before JDK 8, the level of abstraction was the clear-cut difference between abstract class and interface like interface was the purest form of Abstraction which only defines what interface is supposed to do without specifying how they should be implemented, in other words, it only declares API methods and leaves implementation to its subclasses.

But, the main **difference between an abstract class and an interface in Java 8** is the fact that an abstract class is a class and an interface is an interface. A class can have a state which can be modified by non-abstract methods but an interface cannot have the state because they can't have instance variables.

The second difference is that an interface cannot have a constructor even in Java 8 but you may remember that abstract class can also have a constructor in Java. All methods of an interface were abstract but since Java 8 you can define non-abstract methods in the form of default and static methods inside the interface in Java.

This brings many questions to come to mind, which we'll discuss in the next section, btw, if you are new to the Java world and just started learning Java, I suggest you use a course

that is up-to-date like **The Complete Java MasterClass** on Udemy because Java is changing very fast. In just a year we have moved from Java 9 to Java 12. Thankfully this course is up-to-date and the author tries to update it with every Java release.

Difference between Abstract class vs Interface in Java 8

Prima facia, **in Java 8, an interface looks like an abstract class** and one can reason about, *can we use an interface with default methods in place of an abstract class in Java?*

Well, I believe they are for two different purposes and we will learn more once we start using Java 8 regularly, but following the semantics difference between abstract class and interface with default method will guide you further :

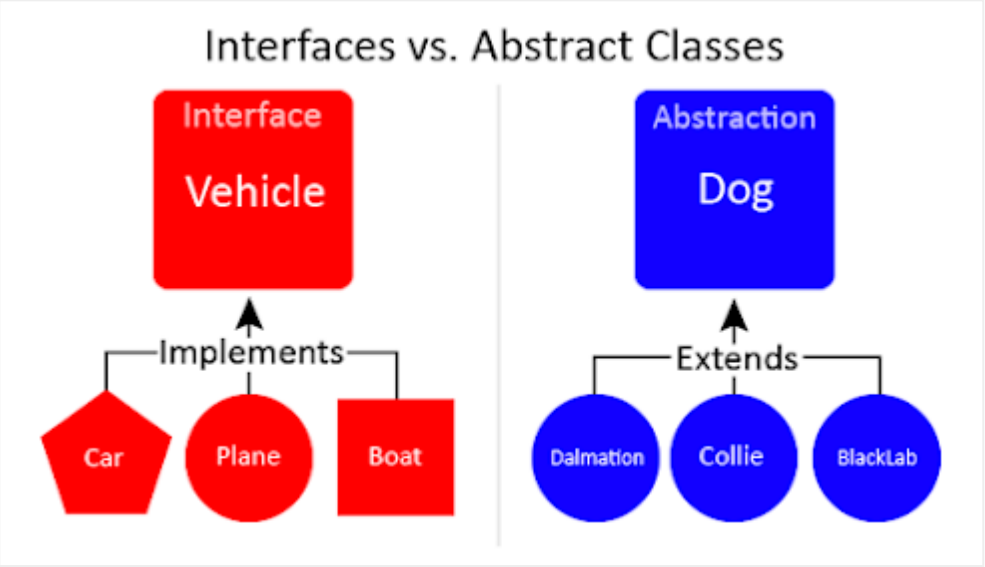
1) Abstract classes are **classes**, so they are not restricted to other restrictions of the interface in Java, like abstract class can have the `state`, but you cannot have the state on the interface in Java.

2) Another semantic difference between an interface with default methods and an abstract class is that you **can define constructors inside an abstract class**, but you cannot define constructors inside an interface in Java.

In reality, default or defender methods are introduced to maintain backward compatibility and the same time making Collection API more suitable to be used inside key Java 8 features like lambda expressions.

Without adding default methods, it wasn't possible to declare any new method on the existing interface in Java without breaking all classes which implement it, but because of the default method, you can now better evolve your API.

They defend your code against implementing new methods hence they are also called defender methods. If you want to know more about default methods or new changes in Java 8 in general, I suggest you check out these Java 8 to Java 13 courses from sites like Udemy and Pluralsight.



That's all about the **difference between an Abstract class and an Interface in Java 8**. Though I certainly agree that the difference between abstract class and the interface has *reduced* with the introduction of default methods, as well as allowing static methods inside the interface and their usage will evolve once Java 8 becomes a mainstream Java development version, but you must remember that an abstract class is a class and an interface is an interface.