Multiple inheritance of state includes ability **to inherit instance fields from multiple classes.**

Multiple inheritance of type includes ability **to implement multiple interfaces and/or ability to extend from multiple classes.**

**Interfaces, classes and enums are all *types***. Java allows a class to implement multiple interfaces. In this way, Java supports multiple inheritances of types.

***State***, on the other hand, **is represented by instance fields**. Only a class can have instance fields and therefore, only a class can have a state (Fields defined in an interface are always implicitly static, even if you don't specify the keyword static explicitly. Therefore, an interface does not have any state.) **Since a class is allowed to extend only from one class at the most, it can inherit only one state**. Thus, **Java does not support multiple inheritance of state**.

This is an important concept and is explained in more detail here: <https://docs.oracle.com/javase/tutorial/java/IandI/multipleinheritance.html>

Java supports **multiple type inheritance but not multiple state inheritances**.

Java allows a class to implement multiple interfaces. An interface is a "type" and does not contain any state. This implies that Java supports multiple type inheritance.

A class contains state and extending a class means inheriting the state. Since Java does not allow a class to extend from multiple classes, it means that Java does not support multiple state inheritances.

This is an important concept and is explained in more detail here: <https://docs.oracle.com/javase/tutorial/java/IandI/multipleinheritance.html>

Which method declarations will enable a class to be run as a standalone program?

- **final** public static void main(String [ ] array)

- public static void main(String args[ ])

**final** only means that subclasses cannot hide (in case of static methods, and static or non-static fields) or override (in case of instance methods) that method.