Why multiple inheritances are not supported in Java

Recently one of my friends appeared for an interview and after few so-called easy questions, he was asked **"Why multiple inheritance is not supported in Java"**, though he has a brief idea that in Java we can support multiple inheritance in java via interface but the interviewer was kept pressing on why part, maybe he was just read any blog post about it :). So after the interview, my friend comes to me and in a usual talk, he told me about these questions and ask me the answer.

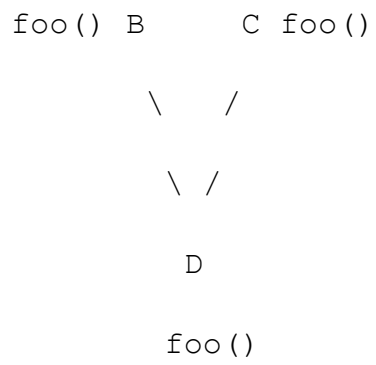
Well, this is a very classical question like **Why String is immutable in Java**; the similarity between these two questions is they are mainly driven by design decisions taken by java's creator or designer. Though following two reasons make sense to me on Why Java doesn't support multiple inheritances:

Why Java doesn't support multiple inheritance

1) First reason is **ambiguity around the Diamond problem**, consider a class A has `foo()` method and then B and C derived from A and has their own foo() implementation, and now class D derives from B and C using multiple inheritance and if we refer just `foo()` compiler will not be able to decide which `foo()` it should invoke.

This is also called the Diamond problem because the structure on this inheritance scenario is similar to 4 edge diamond, see below

```
A foo()
 / \
/   \
```



In my opinion, even if we remove the top head of diamond class A and allow multiple inheritances we will see this problem of ambiguity.

Sometimes if you give this reason to the interviewer he asks if C++ can support *multiple inheritances* then why not Java. hmmmmm in that case I would try to explain to him the second reason which I have given below that it's not because of technical difficulty but more to maintainable and clearer design was driving factor though this can only be confirmed by any java designer and we can just speculate. Wikipedia link has some good explanation on how different language address problem arises due to diamond problem while using multiple inheritances.

2. Second and more convincing reason to me is that **multiple inheritances does complicate the design and creates problem during casting, constructor chaining etc** and given that there are not many scenarios on which you need multiple inheritances its wise decision to omit it for the sake of simplicity.

> Also, java avoids this ambiguity by supporting single inheritance with interfaces. Since the interface only has a method declaration and doesn't provide any implementation there will only be just one implementation of a specific method hence there would not be any ambiguity.