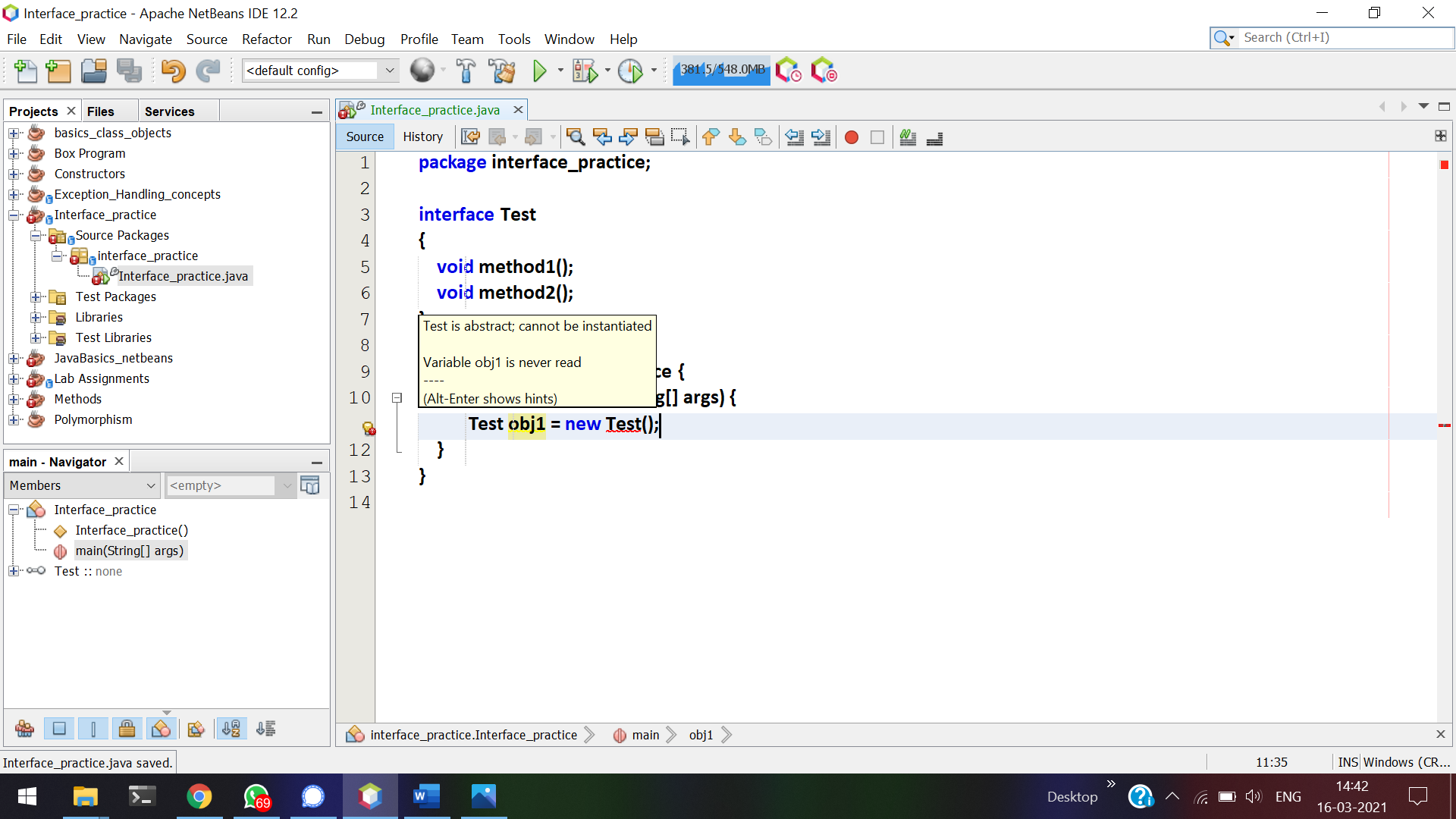
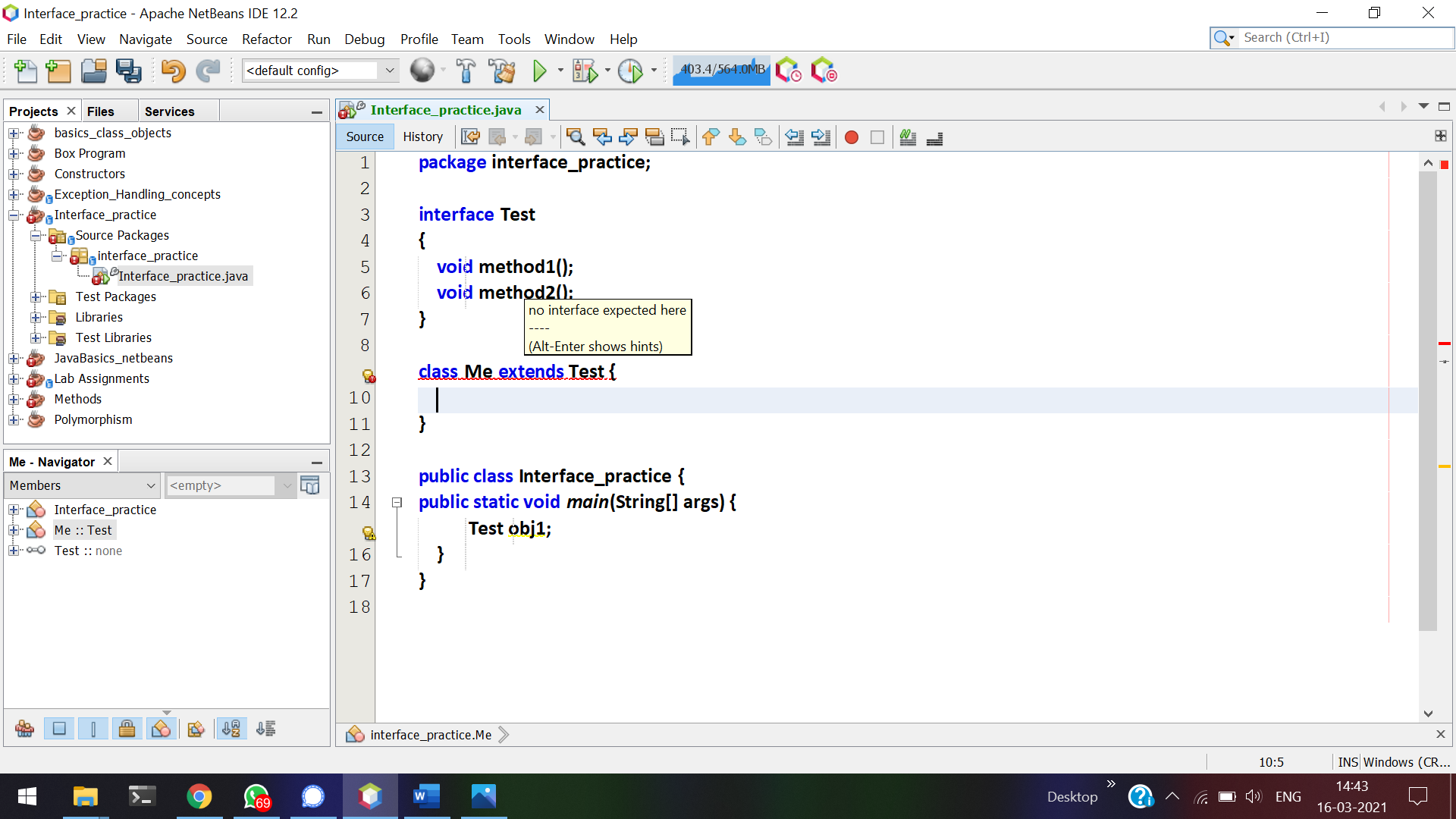
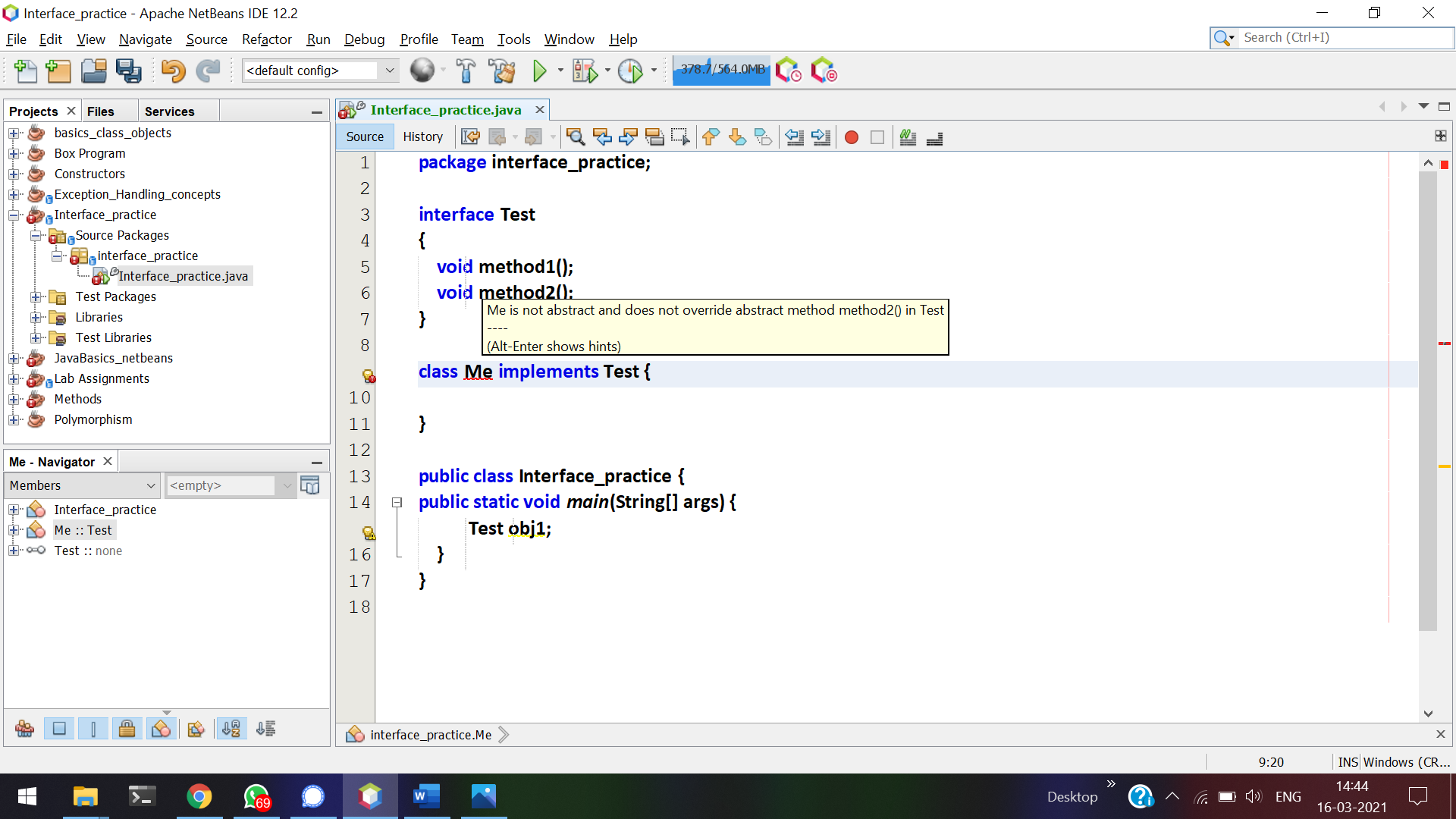


We cannot create objects for abstract classes.



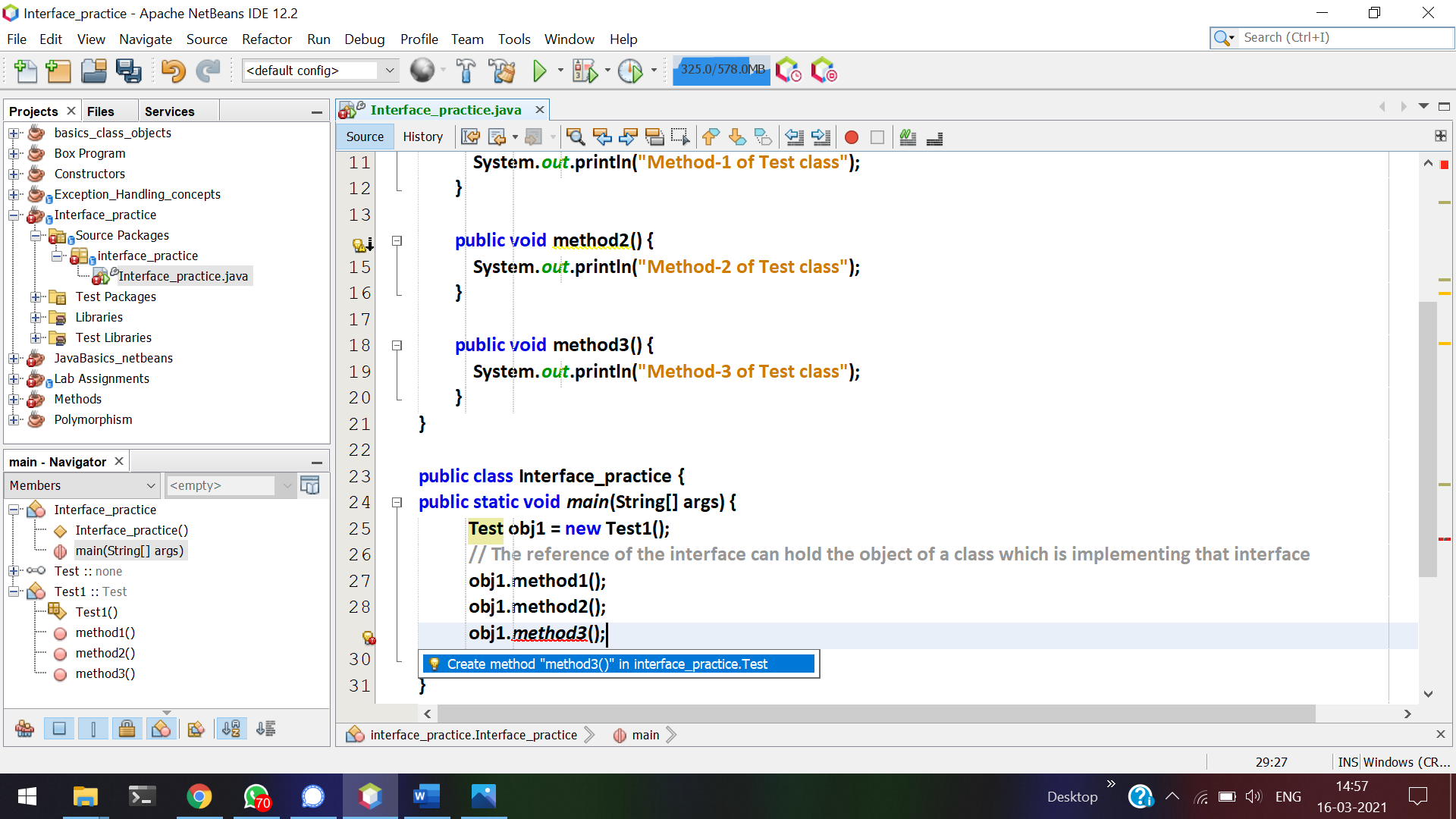


Interface should be implemented and not extended.

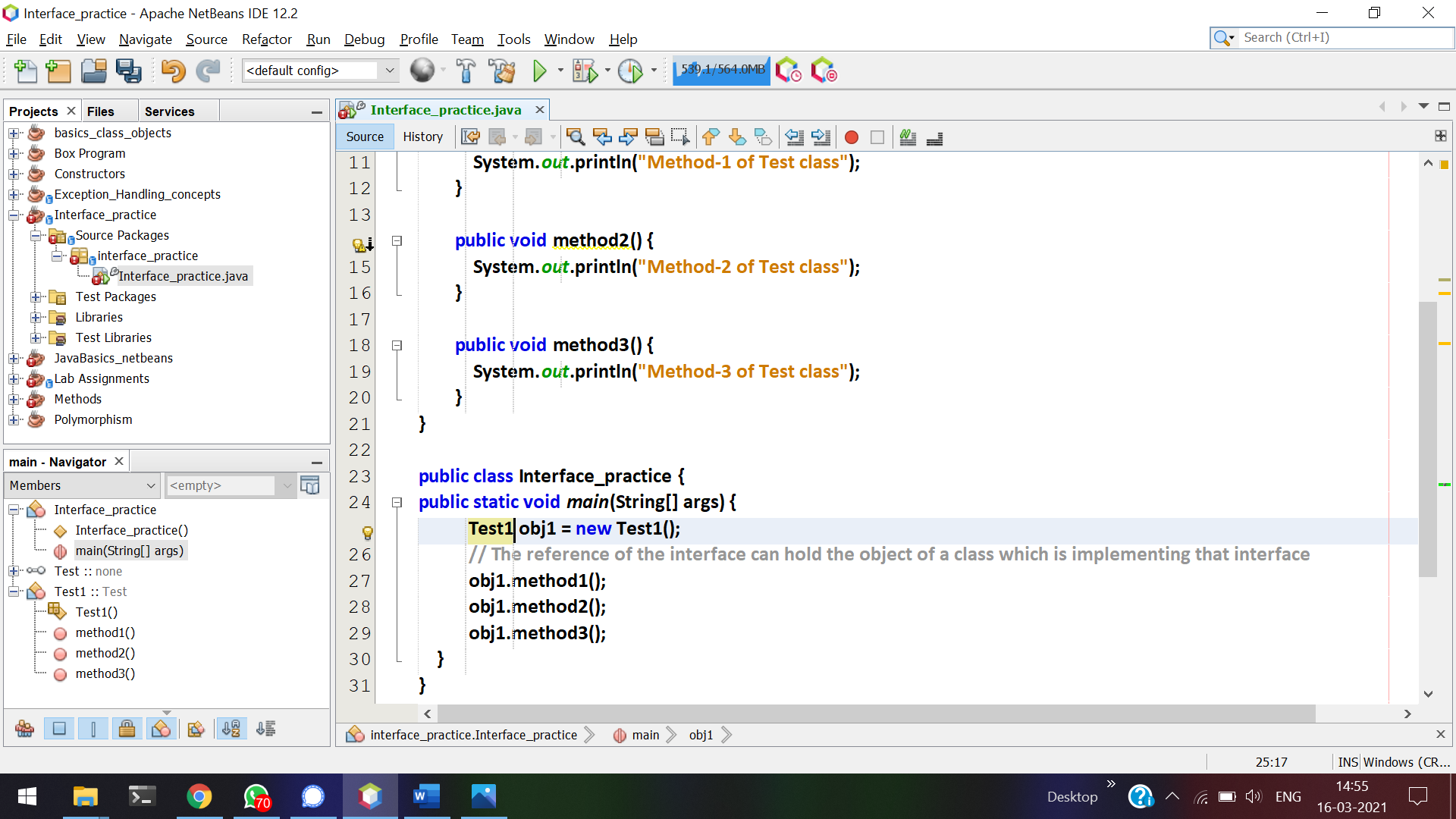


When-ever a class implements a interface, Unless and un-till method1() and method2() are over-written, class Me becomes abstract class.

The reference of the interface can hold the object of a class which is implementing that interface.



method3() is not declared in the interface.  
Without making Test1 as a interface we cannot call method3().

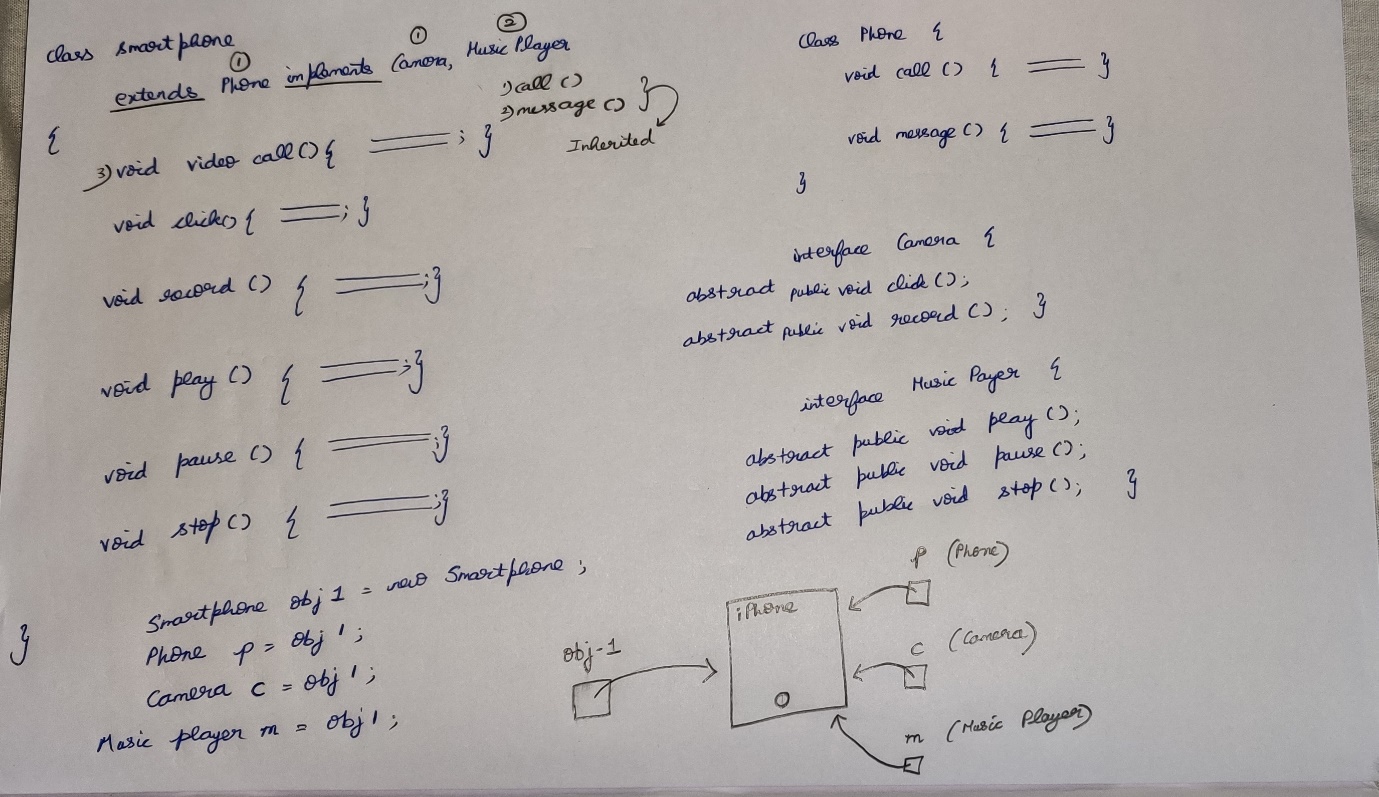


If I make Test1 as a reference, then method3() can be called.

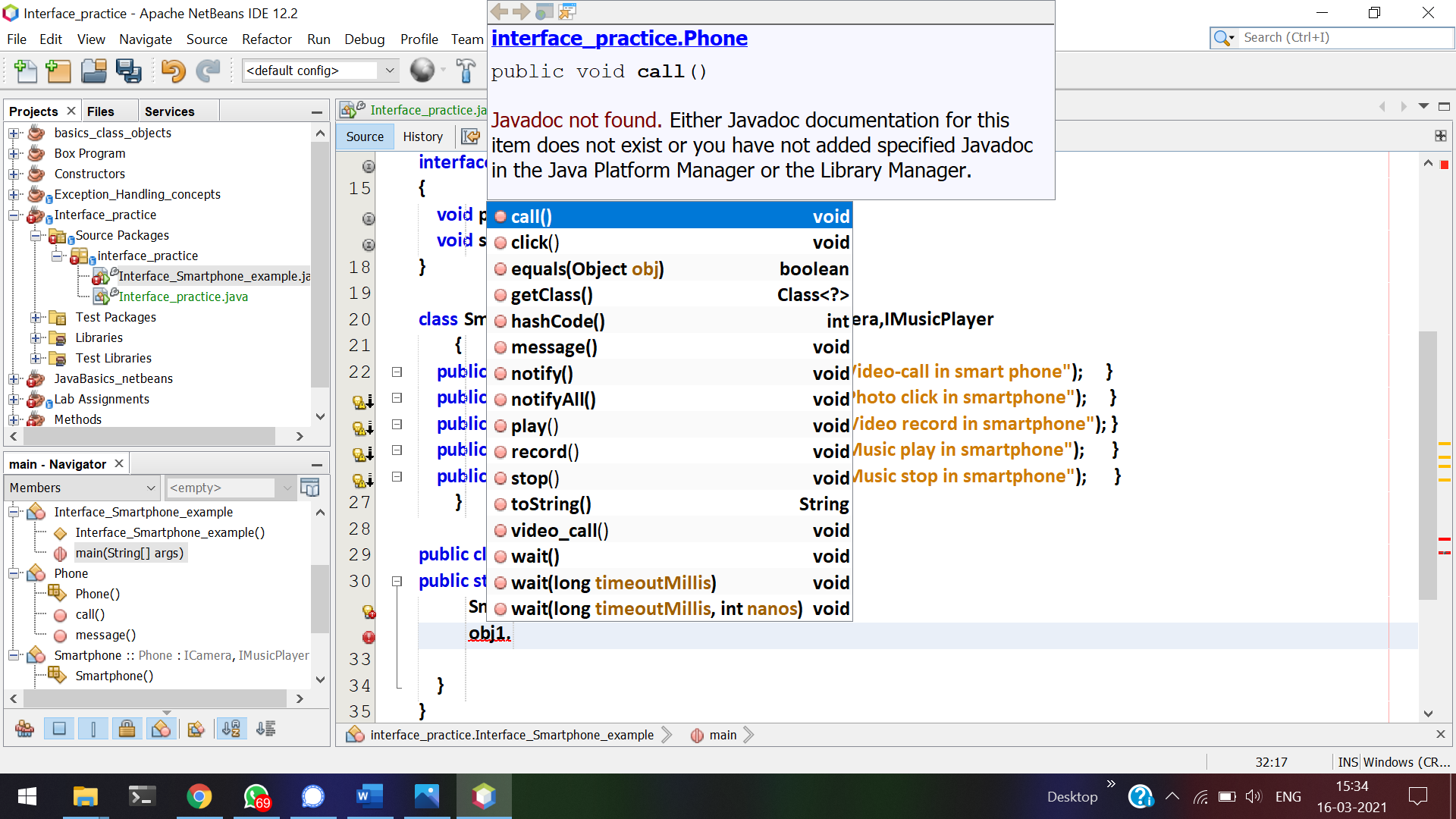
We achieved dynamic dispatch method (i.e) run-time polymorphism. Interfaces are meant to achieve run-time polymorphism.  
Here method1() and method2() are over-ridden.

# **Example for interface**

camera can be interface and also class.  
camera as a class 🡪 DSLR camera.  
camera as a interface 🡪 camera in mobile-phones and tabs.

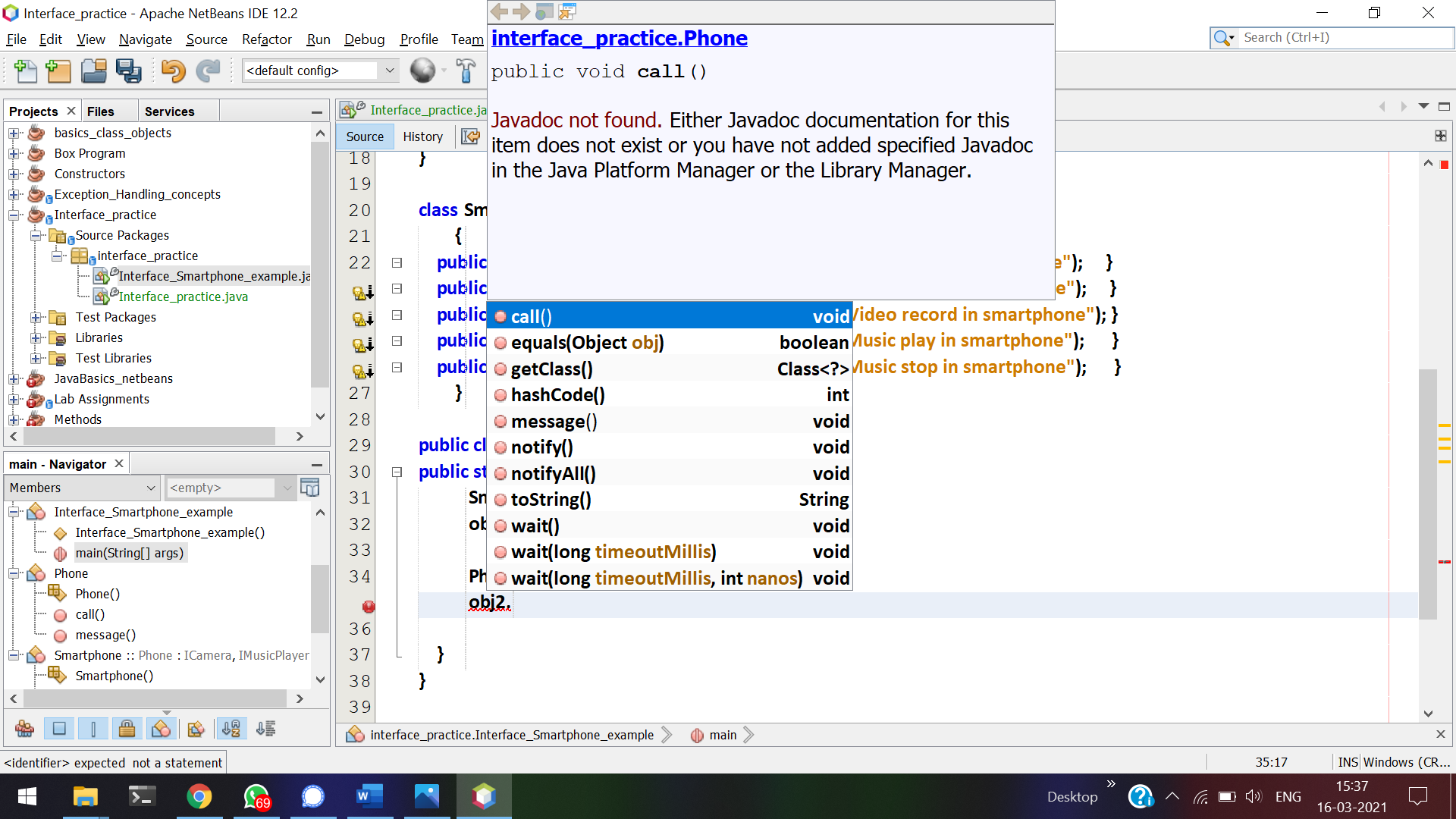


Your friend is asking your camera(interface) to take a snap in your mobile-phone.  
Then he takes snap/recording a video and returning back to you.  
He is not allowed to make a call/play music, since his reference is only camera.



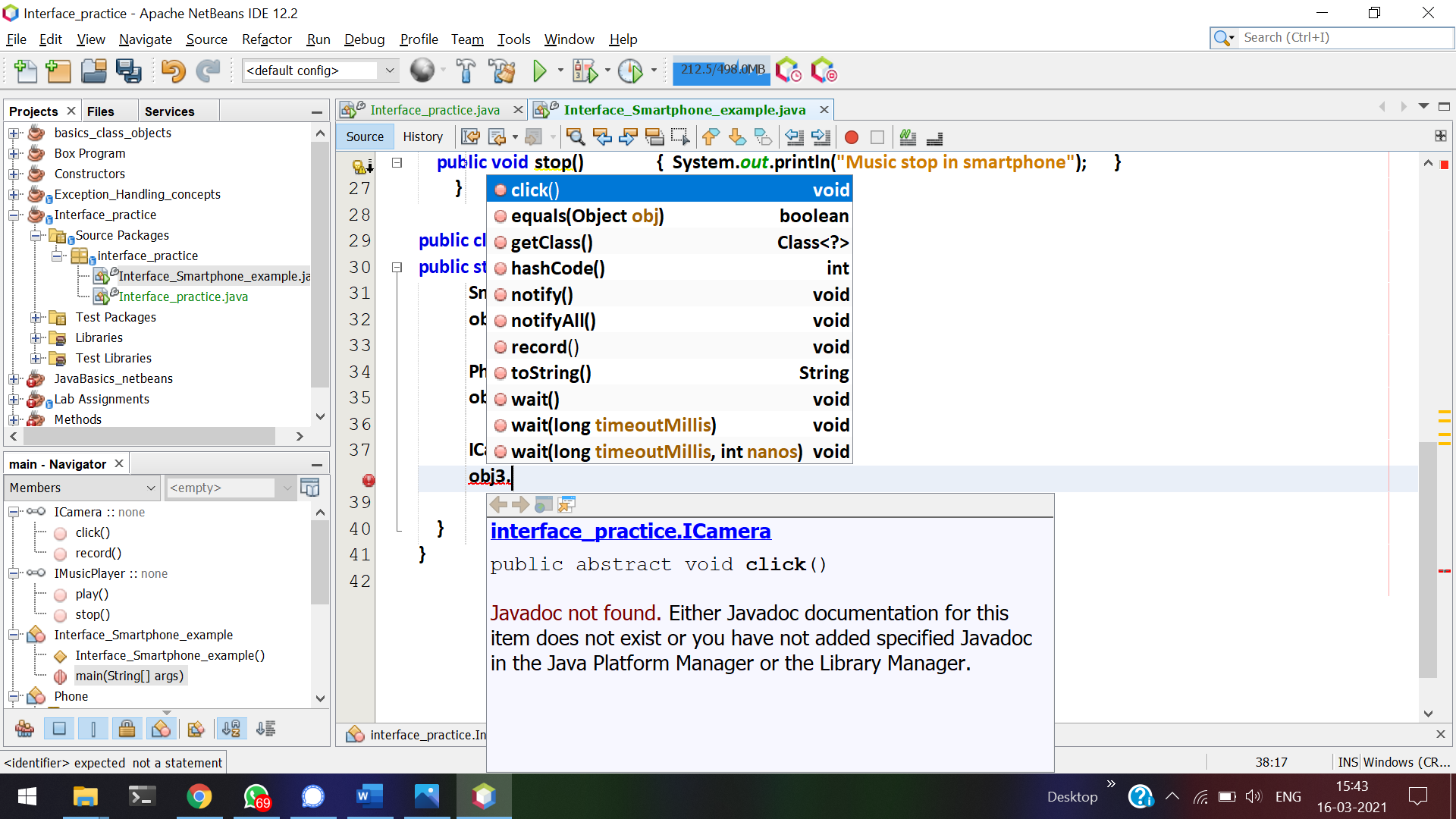


With Smartphone reference, I can able to call all these methods.   
When my friend asks my smartphone, he is allowed to use all the functions available in my smartphone.





With **Phone** reference, I can able to call only call() and message().   
So if my friend asks my phone to make a call/sms, he is allowed to do only these.



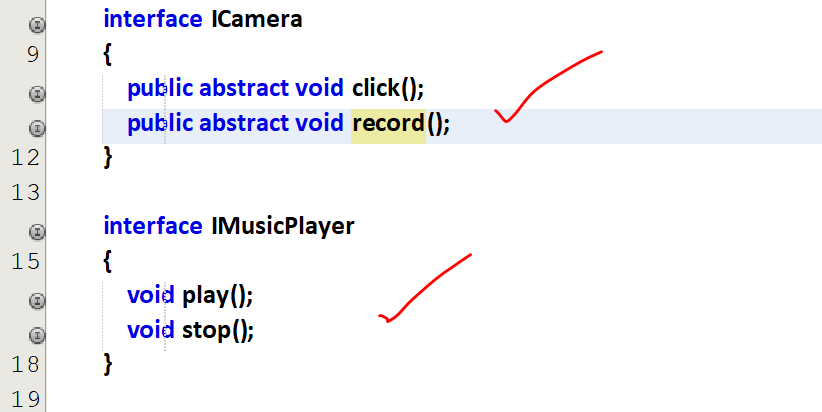


With **ICamera** reference, I can able to call only click() and record().   
So if my friend asks my phone to click a picture/ record a video, he is allowed to do only these.

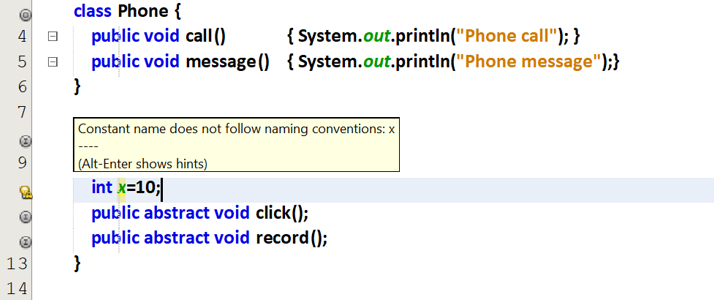
With a single smartphone, I can able to use it as a Phone(class) and ICamera, IMusicPlayer(interface)

# **Do’s and Dont’s of Interfaces**

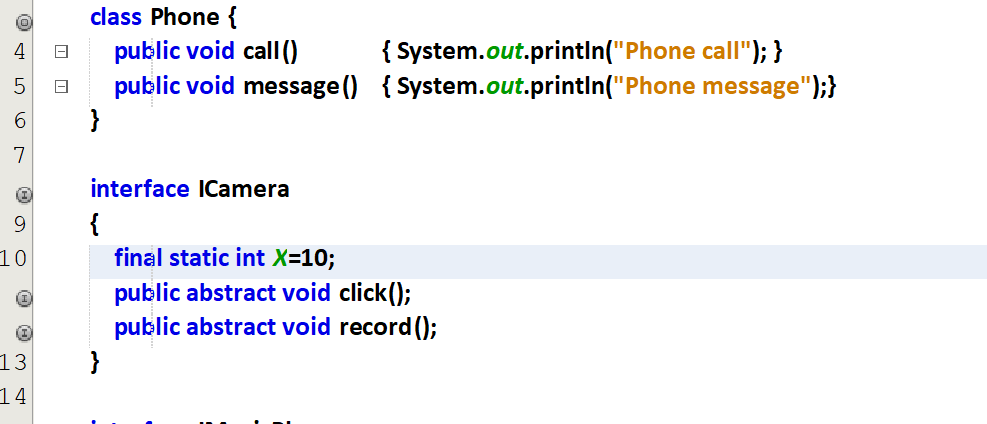
1. By default all the methods in interfaces are public and abstract



1. We can have identifiers in the interface, and by-default they are final and static.

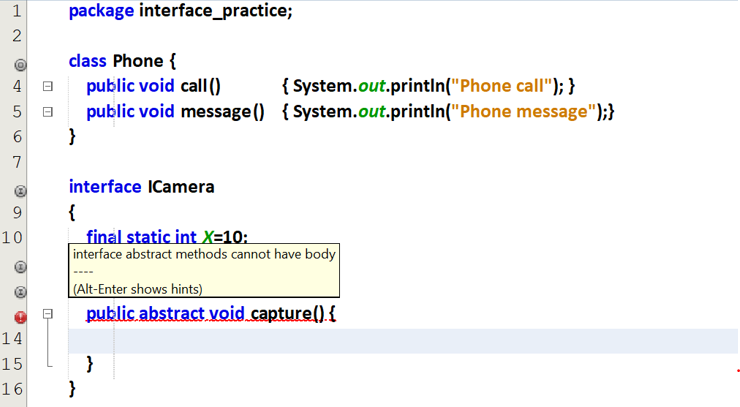


Small letters should not be used as identifiers.

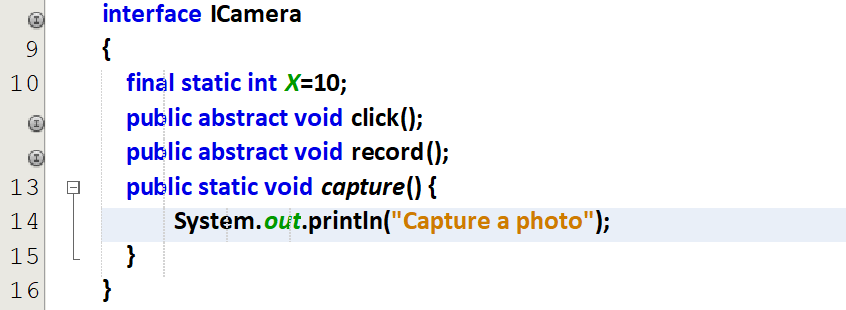


By-default all the variables in ICamera are final and static.

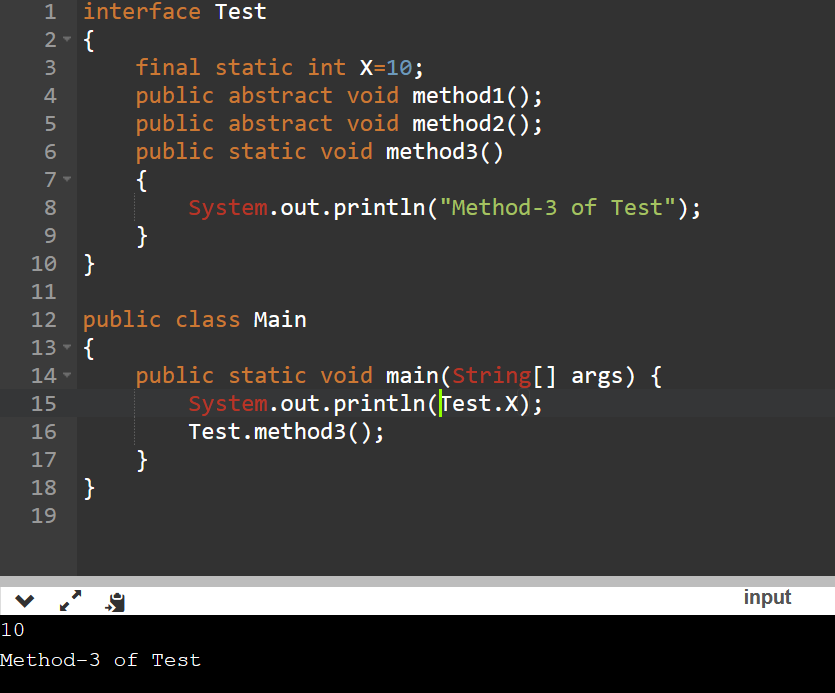
1. Methods inside the interface cannot have body.



With public static void function\_name(), we can define a function inside the interface.

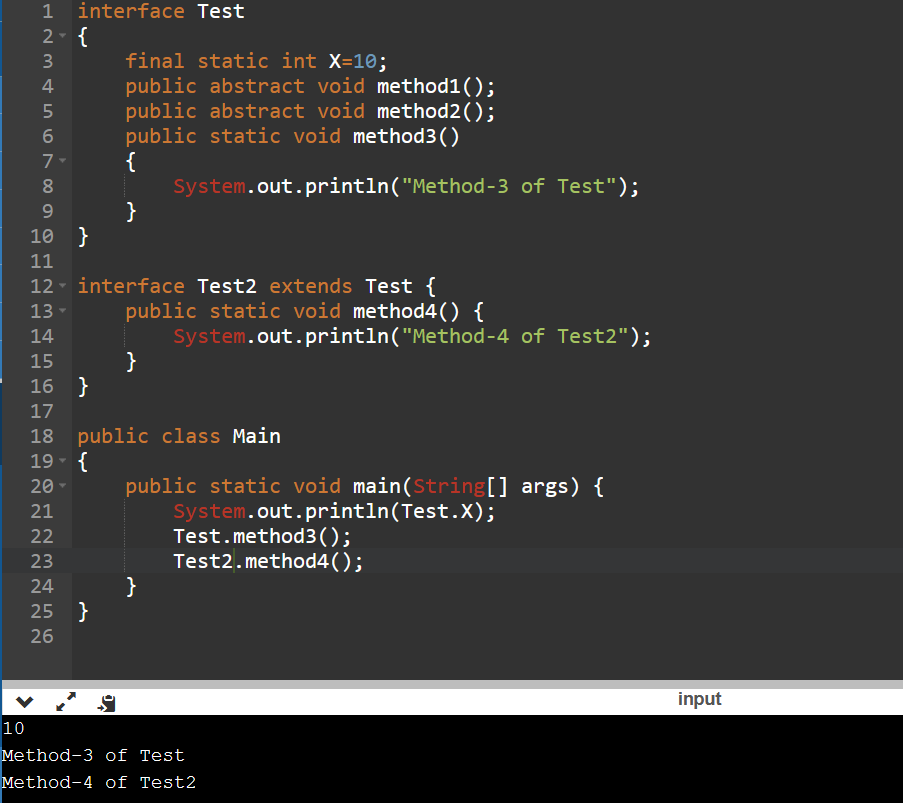






We cannot able to create objects for interface. With class name, we can call the variables and methods.

1. One interface can inherit(i.e extends) another interface



1. Interface can have default methods also