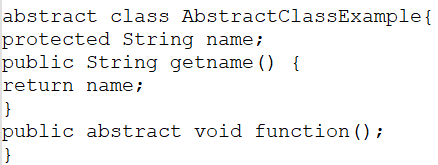
**1)What is an abstract class?**

A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

* An abstract class must be declared with an abstract keyword.
* It can have abstract and non-abstract methods.
* It cannot be instantiated.
* It can have [constructors](https://www.javatpoint.com/java-constructor) and static methods also.

Example of Abstract class:

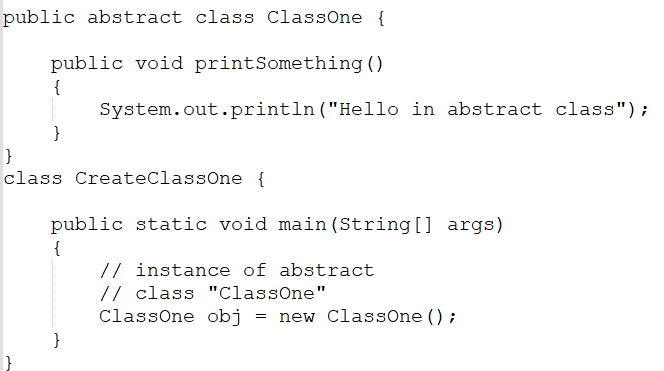


**2)Can you make an instance of an abstract class?**

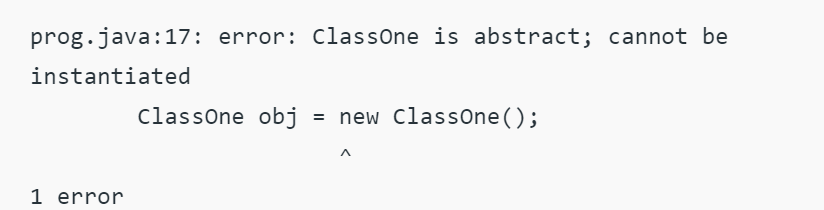
Abstract class are classes which can have abstract methods and it can’t be instantiated. We cannot instantiate an abstract class in Java because it is abstract, it is not complete, hence it cannot be used.

No! You cannot make an instance of an abstract class. An abstract class has to be sub-classed.  
If you have an abstract class and you want to use a method which has been implemented, you may  
need to subclass that abstract class, instantiate your subclass and then call that method.

Example:

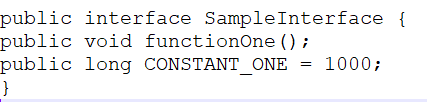


OUTPUT:



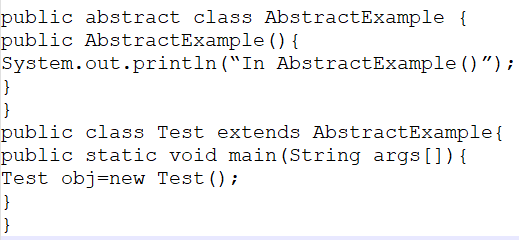
**3)How to define an Interface?**

In Java Interface defines the methods but does not implement them. Interface can include constants.  
A class that implements the interfaces is bound to implement all the methods defined in Interface.  
Example of Interface:



**4)Can Abstract Class have constructors?**

Abstract class’s can have a constructor, but you cannot access it through the object, since you cannot instantiate abstract class. To access the constructor create a sub class and extend the abstract class which is having the constructor.

Example  


**5)Can interfaces have constructors?**

No, you cannot have a constructor within an interface in Java.

* You can have only public, static, final variables and, public, abstract, methods as of Java7.
* From Java8 onwards interfaces allow default methods and static methods.
* From Java9 onwards interfaces allow private and private static methods.

Moreover, all the methods you define (except above mentioned) in an interface should be implemented by another class (overridden). But, you cannot override constructors in Java.

Still if you try to define constructors in an interface it generates a compile time error.

Example

In the following Java program, we are trying to define a constructor within an interface.

public interface MyInterface{

   public abstract MyInterface();

   /\*{

      System.out.println("This is the constructor of the interface");

   }\*/

   public static final int num = 10;

   public abstract void demo();

}

Compile time error

On compiling, the above program generates the following error

Output

MyInterface.java:2: error: expected

   public abstract MyInterface();

^

1 error