**1) Can abstract class have constructors in Java?**

Yes, abstract class can declare and define constructor in Java. Since you cannot create instance of abstract class, constructor can only be called during [constructor chaining](http://javarevisited.blogspot.com/2012/12/constructor-chaining-in-java-calling-another-constructor.html), i.e. when you create instance of concrete implementation class.

Now some interviewer, ask what is the purpose of constructor, if you cannot instantiate abstract class? Well, it can still be used to initialize common variables, which are declared inside abstract class, and used by various implementations.

Also even if you don’t provide any constructor, compiler will add [default no argument constructor](http://javarevisited.blogspot.com/2012/12/what-is-constructor-in-java-example-chainning-overloading.html) in an abstract class, without that your subclass will not compile, since first statement in any constructor implicitly calls super(), default super class constructor in Java.

**2) Can abstract class implements interface in Java? Do they require implementing all methods?**

Yes, abstract class can implement interface by using implements keyword. Since they are abstract, they don’t need to implement all methods. It’s good practice to provide an abstract base class, along with an interface to declare Type. One example of this is java.util.List interface and corresponding java.util.AbstractList abstract class. Since AbstractList implements all common methods, concrete implementations

Like [LinkedList](http://javarevisited.blogspot.com/2012/02/difference-between-linkedlist-vs.html) and [ArrayList](http://javarevisited.blogspot.com/2012/03/how-to-loop-arraylist-in-java-code.html) are free from burden of implementing all methods, had they implemented List interface directly. It’s best of both worlds; you can get advantage of interface for declaring type, and flexibility of abstract class to implement common behavior at one place. Effective Java has a nice chapter on how to use interface and abstract class in Java, which is worth reading.

**3) Can abstract class be final in Java?**

No, abstract class cannot be final in Java. Making them final will stop abstract class from being extended, which is the only way to use abstract class. They are also opposite of each other, abstract keyword enforces to extend a class, for using it, on the other hand, [final keyword](http://javarevisited.blogspot.com/2011/12/final-variable-method-class-java.html) prevents a class from being extended. In real world also, abstract signifies incompleteness, while final is used to demonstrate completeness. Bottom line is, you cannot make your class abstract and final in Java, at same time, it’s a compile time error.

**4) Can abstract class have static methods in Java?**

Yes, abstract class can declare and define [static methods](http://javarevisited.blogspot.com/2011/11/static-keyword-method-variable-java.html), nothing prevents from doing that. But, you must follow guidelines for making a method static in Java, as it’s not welcomed in a object oriented design, because [static methods cannot be overridden in Java](http://javarevisited.blogspot.com/2013/03/can-we-overload-and-override-static-method-java.html). It’s very rare, you see static methods inside abstract class, but as I said, if you have very good reason of doing it, then nothing stops you.

**5) Can you create instance of abstract class?**

No, you cannot create instance of abstract class in Java, they are incomplete. Even though, if you’re abstract class doesn’t contain any abstract method, you cannot create instance of it. By making a class abstract, you told compiler that, it’s incomplete and should not be instantiated. Java compiler will throw error, when a code tries to instantiate abstract class.

**6) Is it necessary for abstract class to have abstract method?**

No, it’s not mandatory for an abstract class to have any abstract method. You can make a class abstract in Java, by just using abstract keyword in class declaration. Compiler will enforce all structural restriction, applied to abstract class, e.g. now allowing creating any instance. By the way, it’s debatable whether you should have abstract method inside abstract class or interface. In my opinion, abstract class should have abstract methods, because that’s the first thing programmer assumes, when he see that class. That would also go nicely along principle of least surprise.

**7) Difference between abstract class and interface in Java?**

This is the most important and one of the classic Java Interview question. I don’t know, how many times I have seen this question at all most all levels of Java interviews. One reason, which makes this question interesting, is ability to produce example. It’s easy to answers questions on core OOPS concepts

like [Abstraction](http://javarevisited.blogspot.com/2010/10/abstraction-in-java.html), [Encapsulation](http://javarevisited.blogspot.com/2012/03/what-is-encapsulation-in-java-and-oops.html), [Polymorphism](http://javarevisited.blogspot.com/2011/08/what-is-polymorphism-in-java-example.html) and [Inheritance](http://javarevisited.blogspot.com/2012/10/what-is-inheritance-in-java-and-oops-programming.html), but when it comes to subtle points like this, candidate more often fumbled. You can see this post for all syntactical difference between abstract class and interface, but it deserves a post on its own.

**8) When do you favor abstract class over interface?**

This is the follow-up of previous interview questions on abstract class and interface. If you know syntactical difference, you can answer this question quite easily, as they are the one, which drives the decision. Since it’s almost impossible to add a new method on a published interface, it’s better to use abstract class, when evolution is concern. Abstract class in Java evolves better than interface. Similarly, if you have too many methods inside interface, you are creating pain for all its implementation; consider providing an abstract class for default implementation. This is the pattern followed in Java collection package; you can see AbstractList provides default implementation for List interface.

**9) What is abstract method in Java?**

An abstract method is a method without body. You just declare method, without defining it and use abstract keyword in method declaration.  All method declared inside [Java Interface](http://javarevisited.blogspot.com/2012/04/10-points-on-interface-in-java-with.html) are by default abstract. Here is an example of abstract method in Java

public void abstract printVersion();

Now, In order to implement this method, you need to extend abstract class and [override](http://javarevisited.blogspot.com/2011/12/method-overloading-vs-method-overriding.html) this method.

**10) Can abstract class contains main method in Java ?**

Yes, abstract class can contain [main method](http://javarevisited.blogspot.sg/2011/12/main-public-static-java-void-method-why.html), it just another static method and you can execute Abstract class with main method, until you don’t create any instance.

**11) Abstract class must have only abstract methods. True or false?**

False. Abstract methods can also have concrete methods.

**12) Is it compulsory for a class which is declared as abstract to have at least one abstract method?**

Not necessarily. Abstract class may or may not have abstract methods.

**13) Can we use “abstract” keyword with constructor, Instance Initialization Block and Static Initialization Block?**

No. Constructor, Static Initialization Block, Instance Initialization Block and variables cannot be abstract.

**14) Why final and abstract cannot be used at a time?**

Because, final and abstract are totally opposite in nature. A final class or method cannot be modified further whereas abstract class or method must be modified further. “final” keyword is used to denote that a class or method does not need further improvements. “abstract” keyword is used to denote that a class or method needs further improvements.

**15) Can we instantiate a class which does not have even single abstract methods but declared as abstract?**

No, we can’t instantiate a class once it is declared as abstract even though it does not have abstract methods.

**16) Can we declare abstract methods as private? Justify your answer?**

No. Abstract methods cannot be private. If abstract methods are allowed to be private, then they will not be inherited to sub class and will not get enhanced.

17**) We can’t instantiate an abstract class. Then why constructors are allowed in abstract class?**

It is because, we can’t create objects to abstract classes but we can create objects to their sub classes. From sub class constructor, there will be an implicit call to super class constructor. That’s why abstract classes should have constructors. Even if you don’t write constructor for your abstract class, compiler will keep default constructor.

**18) Can we declare abstract methods as static?**

No, abstract methods cannot be static.

**19) Can a class contain an abstract class as a member?**

Yes, a class can have abstract class as its member.

**20) Abstract classes can be nested. True or false?**

True. Abstract classes can be nested i.e. an abstract class can have another abstract class as it’s member.

**21) Can we declare abstract methods as synchronized?**

No, abstract methods cannot be declared as synchronized. But methods which override abstract methods can be declared as synchronized.

**22) Can we declare local inner class as abstract?**

Yes. Local inner class can be abstract.

**23) Can abstract method declaration include throws clause?**

Yes. Abstract methods can be declared with throws clause.