1. **Difference between Abstract class and interface.**

|  |  |
| --- | --- |
| **Interface** | **Abstract class** |
| if we don’t know anything about method implementation we just know requirement specification then we should go for interface | if we know partially method implementation (not completely ) then we should go for abstract class |
| Inside interface every methods are public and abstract whether we are declaring or not | Inside abstract class every methods need not be public and abstract. |
| In interface we can take concrete method but it need to be default or static modifier . | In abstract class we can take concrete method without default or static modifier. |
| Inside interface every variable are public static final whether we are declaring or not and we cannot change its value. | Inside abstract class variable need not be public static final and we can change its value |
| inside interface Variable compulsory need to be initialized | Variable need not to be initialized |
| Inside interface we can not declare instance and static block | Inside abstract class we can declare instance and static block |
| We cannot create interface object | We can create constructor inside abstract class it will be executed at the runtime of object creation |
| We can implement N number of interface in implementation class.  as java support multiple inheritance for interface | We can extend only one class as java dose not support multiple inheritance for classes |

Interview Questions :

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

* Yes, abstract class can declare and define constructor in Java. Since you can not 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.

1. **what is the purpose of constructor, if you can not instantiate abstract class?**

* it can still be used to initialize common variables, which are declared inside abstract class, and used by various implementation**.**
* whenever we are creating child class object parent class constructor will be executed but parent class object wont be created
* if we are not writing constructor inside parent class then child class is the responsible to initialize parent class variable.

e.g

**abstract** **class** Abstarct1 {

String a;

**public** **abstract** **void** m1();

**public** **int** m8(**int** a) {

**return** a;

}

}

**public** **class** AbstractImpl **extends** Abstarct1{

**public** AbstractImpl(String s) {

**this**.a=s;

}

@Override

**public** **void** m1() {

System.***out***.println(a);

}

**public** **int** m8(){

**return** 1;

}

**public** **static** **void** main(String[] ar){

AbstractImpl b =**new** AbstractImpl("Shailesh");

b.m1();

}

}

o\p : Shailesh

* in above example we are having only one variable in side parent class so it is possible to initialize parent class variable inside child class constructor.
* But consider we are having 100 variable inside parent class and we are not having Constructor inside abstract class and we are having 1000 of chilled classes having 10 variable in each extending to the same parent class which is having 100 variable. In this child class is the responsible to initialize parent class variable. Now we need to write 100 variable initiation line inside every child class constructor it menace we are rewriting all the 100 lines inside each child so there will be code redundancy issue. To avoid this issue we need Constructor in side abstract class to initializes 100 variable by doing this the child Constructor will pass the 100 values to the abstract class to initialize its variable and code redundancy will be removed.

### Can abstract class implements interface in Java? does they require to implement 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.Listinterface 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 Listinterface directly. It’s best of both world, you can get advantage of interface for declaring type, and flexibility of abstract class to implement common behavior at one place. Effective Java hasa nice chapter on how to use interface and abstract class in Java, which is worth reading.

### Can abstract class be final in Java?

No, abstract class can not 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 can not make your class abstract and final in Java, at same time, it’s a compile time error.

### 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 can not 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.

1. **When we can have static method inside abstract class?**
2. **Can you create instance of abstract class?**

No, you can not create instance of abstract class in Java, they are incomplete. Even though, if your abstract class don’t contain any abstract method, you can not 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.

1. **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 to create 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.

1. **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 it’s 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.

1. **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.

1. **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.

1. **In What Scenarios Will You Use Abstract Class vs Interface**

If you want to increase reusability in inheritance then abstract classes are good.

If you want to implement or force some methods across classes must be for uniformity you can use an interface.

So to increase reusability via inheritance use the abstract class as it is nothing but a base class and to force methods to use interfaces.