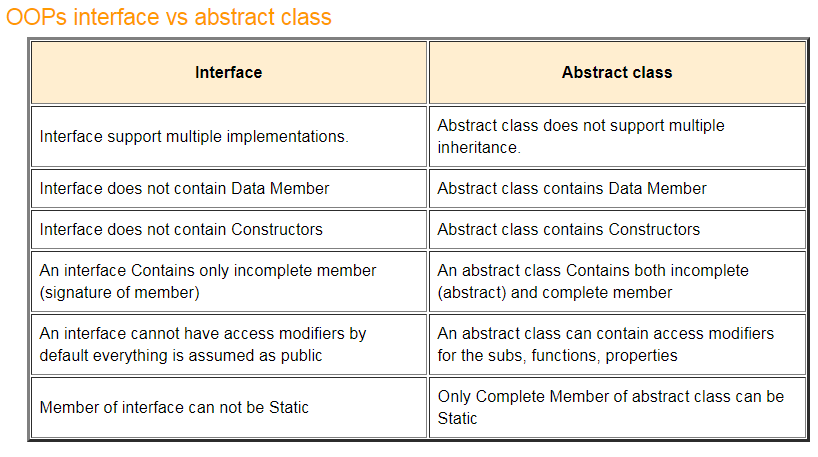
[](https://i.stack.imgur.com/Xf8Yz.png)Taken from:

<http://www.dotnetbull.com/2011/11/difference-between-abstract-class-and.html>

<http://www.dotnetbull.com/2011/11/what-is-abstract-class-in-c-net.html><http://www.dotnetbull.com/2011/11/what-is-interface-in-c-net.html>

[share](https://stackoverflow.com/a/18810652)[improve this answer](https://stackoverflow.com/posts/18810652/edit)

[edited Oct 26 at 5:55](https://stackoverflow.com/posts/18810652/revisions)

answered Sep 15 '13 at 8:59

[[https://www.gravatar.com/avatar/26efb73b8af0373305693dd6beddf3c0?s=32&d=identicon&r=PG](https://stackoverflow.com/users/2373500/vivek)](https://stackoverflow.com/users/2373500/vivek)

[Vivek](https://stackoverflow.com/users/2373500/vivek)

**3,782**2138

* 10

You need to say what language this applies to ("Abstract class does not support multiple inheritance" is far from being universally true) – [Ben Voigt](https://stackoverflow.com/users/103167/ben-voigt) [Mar 7 '14 at 4:16](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment33776339_18810652)

* Last comparison is confusing as per table! Methods in interface can't be static but variables are static final Implemented methods in abstract class can be static – [realPK](https://stackoverflow.com/users/853001/realpk" \o "599 reputation) [Mar 9 '14 at 4:29](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment33843454_18810652)
* 1

Typo it not Cunstructor .. Its Constructor.. – [Pra Jazz](https://stackoverflow.com/users/1487469/pra-jazz" \o "2,299 reputation) [Jul 18 '14 at 10:57](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment38536470_18810652)

* 6

Member of the interface must be static final . Last statement is wrong. – [Xar E Ahmer](https://stackoverflow.com/users/3496570/xar-e-ahmer" \o "22,172 reputation) [Aug 5 '14 at 8:21](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment39121757_18810652)

* 1

What is the targetted programming language here? C#? – [Peter Mortensen](https://stackoverflow.com/users/63550/peter-mortensen) [Nov 20 '16 at 14:26](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment68638121_18810652)

[show **1** more comment](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class)

up vote77down vote

An explanation can be found here: <http://www.developer.com/lang/php/article.php/3604111/PHP-5-OOP-Interfaces-Abstract-Classes-and-the-Adapter-Pattern.htm>

An abstract class is a class that is only partially implemented by the programmer. It may contain one or more abstract methods. An abstract method is simply a function definition that serves to tell the programmer that the method must be implemented in a child class.

An interface is similar to an abstract class; indeed interfaces occupy the same namespace as classes and abstract classes. For that reason, you cannot define an interface with the same name as a class. An interface is a fully abstract class; none of its methods are implemented and instead of a class sub-classing from it, it is said to implement that interface.

Anyway I find this explanation of interfaces somewhat confusing. A more common definition is: *An interface defines a contract that implementing classes must fulfill. An interface definition consists of signatures of public members, without any implementing code.*

[share](https://stackoverflow.com/a/1913112)[improve this answer](https://stackoverflow.com/posts/1913112/edit)

[edited Dec 16 '09 at 8:24](https://stackoverflow.com/posts/1913112/revisions)

answered Dec 16 '09 at 8:18

[[https://i.stack.imgur.com/iqUR6.jpg?s=32&g=1](https://stackoverflow.com/users/4574/konamiman)](https://stackoverflow.com/users/4574/konamiman)

[Konamiman](https://stackoverflow.com/users/4574/konamiman)

**42k**1496126

* 4

This is the most correct answer, since PHP interfaces differ from other languages in that PHP interfaces ARE abstract classes under the hood, whereas other languages' interfaces are signatures that classes must match. They behave the same as long as there are no errors though. – [Tor Valamo](https://stackoverflow.com/users/228936/tor-valamo) [Dec 16 '09 at 8:55](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment1818436_1913112)

* 1

True, for PHP it's the real best anwser. But it's harder to get from the text blob than from a simple snippet. – [e-satis](https://stackoverflow.com/users/9951/e-satis) [Dec 16 '09 at 12:15](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment1819529_1913112)

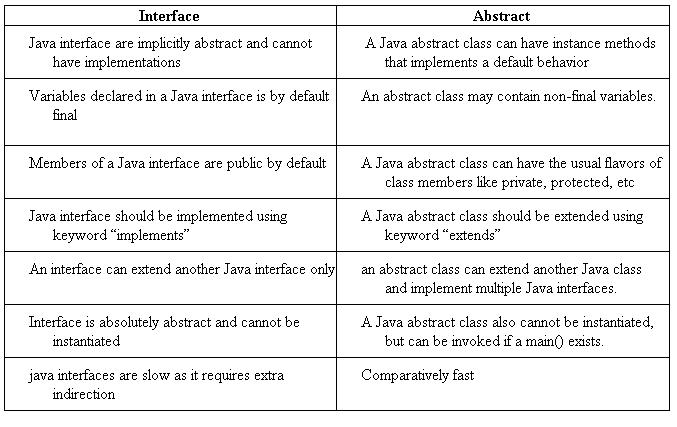
* From the definitions you provided, they look the same except for one detail: and interface is 100% abstract, while an abstract class is partially abstract and can have some method implementations (perhaps all methods can have implementations?). – [jww](https://stackoverflow.com/users/608639/jww" \o "51,706 reputation) [Aug 23 '14 at 4:02](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment39725918_1913112)

add a comment

up vote37down vote

Some important differences:

In the form of a table:



As [stated by Joe from javapapers](http://javapapers.com/core-java/abstract-and-interface-core-java-2/difference-between-a-java-interface-and-a-java-abstract-class/):

1.Main difference is methods of a Java interface are implicitly abstract and cannot have implementations. A Java abstract class can have instance methods that implements a default behavior.

2.Variables declared in a Java interface is by default final. An abstract class may contain non-final variables.

3.Members of a Java interface are public by default. A Java abstract class can have the usual flavors of class members like private, protected, etc..

4.Java interface should be implemented using keyword “implements”; A Java abstract class should be extended using keyword “extends”.

5.An interface can extend another Java interface only, an abstract class can extend another Java class and implement multiple Java interfaces.

6.A Java class can implement multiple interfaces but it can extend only one abstract class.

7.Interface is absolutely abstract and cannot be instantiated; A Java abstract class also cannot be instantiated, but can be invoked if a main() exists.

8.In comparison with java abstract classes, java interfaces are slow as it requires extra indirection.

[share](https://stackoverflow.com/a/16579929)[improve this answer](https://stackoverflow.com/posts/16579929/edit)

[edited Oct 14 '13 at 19:53](https://stackoverflow.com/posts/16579929/revisions)

[[https://www.gravatar.com/avatar/52a2ffb44b1c11ab9001cb67e0a0021c?s=32&d=identicon&r=PG](https://stackoverflow.com/users/19679/brad-larson)](https://stackoverflow.com/users/19679/brad-larson)

[Brad Larson](https://stackoverflow.com/users/19679/brad-larson)**♦**

**161k**40363541

answered May 16 '13 at 5:55

[[https://www.gravatar.com/avatar/3f03fd88c58012533a4dfcc0215e8f05?s=32&d=identicon&r=PG&f=1](https://stackoverflow.com/users/1688090/softmage99)](https://stackoverflow.com/users/1688090/softmage99)

[softmage99](https://stackoverflow.com/users/1688090/softmage99)

**627**1812

* 3

I've edited your answer to provide correct attribution. You can't just drop a link at the bottom of your answer. You need to quote all of the language that was copied from another source, as well. Also, if that table was drawn from somewhere, you should clearly indicate where that is from. – [Brad Larson**♦**](https://stackoverflow.com/users/19679/brad-larson) [Oct 14 '13 at 19:54](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment28700329_16579929)

* Please mention for C++ as well.. though there is no keyword "interface" in C++ as such, but its a commonly asked Qn regd C++ as well. – [cbinder](https://stackoverflow.com/users/837515/cbinder" \o "774 reputation) [Jun 18 '14 at 4:52](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment37509521_16579929)
* @cbinder : There is no keyword 'interface' in c++. For difference in c++, please refer 1. [tutorialspoint.com/cplusplus/cpp\_interfaces.htm](http://www.tutorialspoint.com/cplusplus/cpp_interfaces.htm) 2. [tutorialspoint.com/cplusplus/cpp\_interfaces.htm](http://www.tutorialspoint.com/cplusplus/cpp_interfaces.htm) – [softmage99](https://stackoverflow.com/users/1688090/softmage99) [Jun 18 '14 at 9:34](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment37518352_16579929)
* @MageshBabu Perhaps defining a function in a class containing pure virtual function makes it an abstract class rather than the interface – [cbinder](https://stackoverflow.com/users/837515/cbinder" \o "774 reputation) [Jun 19 '14 at 9:52](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment37559260_16579929)
* 2

With Java 8, the differences are less now. Check updated differences here: [journaldev.com/1607/…](http://www.journaldev.com/1607/difference-between-abstract-class-and-interface-in-java) – [Pankaj](https://stackoverflow.com/users/926520/pankaj" \o "2,808 reputation)[Jul 15 '14 at 17:28](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment38425561_16579929)

[show **1** more comment](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class)

up vote35down vote

I don't want to highlight the differences, which have been already said in many answers ( regarding public static final modifiers for variables in interface & support for protected, private methods in abstract classes)

In simple terms, I would like to say:

*interface:* To implement a contract by multiple unrelated objects

*abstract class:* To implement the same or different behaviour among multiple related objects

From the Oracle [documentation](https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html)

*Consider using abstract classes* if :

1. You want to share code among several closely related classes.
2. You expect that classes that extend your abstract class have many common methods or fields, or require access modifiers other than public (such as protected and private).
3. You want to declare non-static or non-final fields.

*Consider using interfaces* if :

1. You expect that unrelated classes would implement your interface. For example,many unrelated objects can implement Serializable interface.
2. You want to specify the behaviour of a particular data type, but not concerned about who implements its behaviour.
3. You want to take advantage of multiple inheritance of type.

*abstract class establishes "is a" relation with concrete classes. interface provides "has a" capability for classes.*

If you are looking for Java as programming language, here are a few more updates:

Java 8 has reduced the gap between interface and abstract classes to some extent by providing a default method feature. *An interface does not have an implementation for a method* is no longer valid now.

Refer to this documentation [page](https://docs.oracle.com/javase/tutorial/java/IandI/defaultmethods.html) for more details.

Have a look at this SE question for code examples to understand better.

[How should I have explained the difference between an Interface and an Abstract class?](https://stackoverflow.com/questions/18777989/how-should-i-have-explained-the-difference-between-an-interface-and-an-abstract/34978606#34978606)

[share](https://stackoverflow.com/a/33957698)[improve this answer](https://stackoverflow.com/posts/33957698/edit)

[edited May 23 '17 at 12:34](https://stackoverflow.com/posts/33957698/revisions)

[[https://www.gravatar.com/avatar/a007be5a61f6aa8f3e85ae2fc18dd66e?s=32&d=identicon&r=PG](https://stackoverflow.com/users/-1/community)](https://stackoverflow.com/users/-1/community)

[Community](https://stackoverflow.com/users/-1/community)**♦**

**1**1

answered Nov 27 '15 at 12:42

[[https://i.stack.imgur.com/nAGmt.png?s=32&g=1](https://stackoverflow.com/users/4999394/ravindra-babu)](https://stackoverflow.com/users/4999394/ravindra-babu)

[Ravindra babu](https://stackoverflow.com/users/4999394/ravindra-babu)

**28.4k**5151134

add a comment

up vote25down vote

The main point is that:

* **Abstract is object oriented**. It offers the basic data an 'object' should have and/or functions it should be able to do. It is concerned with the object's basic characteristics: what it has and what it can do. Hence objects which inherit from the same abstract class share the basic characteristics (generalization).
* **Interface is functionality oriented**. It defines functionalities an object should have. Regardless what object it is, as long as it can do these functionalities, which are defined in the interface, it's fine. It ignores everything else. An object/class can contain several (groups of) functionalities; hence it is possible for a class to implement multiple interfaces.

[share](https://stackoverflow.com/a/26097117)[improve this answer](https://stackoverflow.com/posts/26097117/edit)

[edited Jun 30 '16 at 17:48](https://stackoverflow.com/posts/26097117/revisions)

[[https://www.gravatar.com/avatar/508065acda08f03db54a19db4ca2b6b9?s=32&d=identicon&r=PG&f=1](https://stackoverflow.com/users/6489306/ragingasiancoder)](https://stackoverflow.com/users/6489306/ragingasiancoder)

[ragingasiancoder](https://stackoverflow.com/users/6489306/ragingasiancoder)

**604**416

answered Sep 29 '14 at 9:54

[[https://www.gravatar.com/avatar/1b9fca2dd4c9125a356a881b222d2fbf?s=32&d=identicon&r=PG](https://stackoverflow.com/users/1074582/yusup)](https://stackoverflow.com/users/1074582/yusup)

[Yusup](https://stackoverflow.com/users/1074582/yusup)

**656**810

* Thank you now we are getting somewhere with a good high level response. Funny how far deep into the comments you have to go to find a more understanding-based response. – [Andrew](https://stackoverflow.com/users/3093731/andrew) [Jan 16 '17 at 22:51](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment70569750_26097117)
* The other answers are too technical. This is getting towards what I would feel is a 'right' answer. The whole point of OOP is semantics, and whether or not a private nested classes public getters are invoked through CPU expensive lookups is pretty much irrelevant here – [Sentinel](https://stackoverflow.com/users/442396/sentinel) [Mar 6 '17 at 21:41](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment72399752_26097117)

add a comment

up vote23down vote

When you want to provide polymorphic behaviour in an inheritance hierarchy, use abstract classes.

When you want polymorphic behaviour for classes which are completely unrelated, use an interface.

[share](https://stackoverflow.com/a/10779446)[improve this answer](https://stackoverflow.com/posts/10779446/edit)

[edited Nov 20 '16 at 14:21](https://stackoverflow.com/posts/10779446/revisions)

[[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](https://stackoverflow.com/users/63550/peter-mortensen)](https://stackoverflow.com/users/63550/peter-mortensen)

[Peter Mortensen](https://stackoverflow.com/users/63550/peter-mortensen)

**13.2k**1983111

answered May 28 '12 at 4:42

[[https://www.gravatar.com/avatar/c238de501c7a7e6868ea4d1b1253b843?s=32&d=identicon&r=PG](https://stackoverflow.com/users/1420976/sculptor)](https://stackoverflow.com/users/1420976/sculptor)

[sculptor](https://stackoverflow.com/users/1420976/sculptor)

**239**22

add a comment

up vote23down vote

I am constructing a building of 300 floors

The building's blueprint **interface**

* For example, Servlet(I)

Building constructed up to 200 floors - partially completed---**abstract**

* Partial implementation, for example, generic and HTTP servlet

Building construction completed-**concrete**

* Full implementation, for example, own servlet

Interface

* We don't know anything about implementation, just requirements. We can go for an interface.
* Every method is public and abstract by default
* It is a 100% pure abstract class
* If we declare public we cannot declare private and protected
* If we declare abstract we cannot declare final, static, synchronized, strictfp and native
* Every interface has public, static and final
* Serialization and transient is not applicable, because we can't create an instance for in interface
* Non-volatile because it is final
* Every variable is static
* When we declare a variable inside an interface we need to initialize variables while declaring
* Instance and static block not allowed

Abstract

* Partial implementation
* It has an abstract method. An addition, it uses concrete
* No restriction for abstract class method modifiers
* No restriction for abstract class variable modifiers
* We cannot declare other modifiers except abstract
* No restriction to initialize variables

Taken from DurgaJobs Website

[share](https://stackoverflow.com/a/24660764)[improve this answer](https://stackoverflow.com/posts/24660764/edit)

[edited Apr 1 at 9:35](https://stackoverflow.com/posts/24660764/revisions)

[[https://lh6.googleusercontent.com/-tKs0WJTpN5U/AAAAAAAAAAI/AAAAAAAAABU/evA61jYST_E/photo.jpg?sz=32](https://stackoverflow.com/users/6874531/chamod-pathirana)](https://stackoverflow.com/users/6874531/chamod-pathirana)

[Chamod Pathirana](https://stackoverflow.com/users/6874531/chamod-pathirana)

**97**10

answered Jul 9 '14 at 18:00

[[https://i.stack.imgur.com/1vnH1.jpg?s=32&g=1](https://stackoverflow.com/users/2857333/jaichander)](https://stackoverflow.com/users/2857333/jaichander)

[Jaichander](https://stackoverflow.com/users/2857333/jaichander)

**617**1921

* An abstract class can have a constructor – [vimal krishna](https://stackoverflow.com/users/2211208/vimal-krishna" \o "1,488 reputation) [Jul 26 '17 at 8:33](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment77604933_24660764)
* 2

I completely disagree with this view. The blueprint is a completely different concept to 'interface.' Blueprint is more analogous to a static model or design specification for a specific implementation. It is closer to 'class,' as the blueprint can be instantiated multiple times through its constructor, but even this is not close enough as the 'class' also contains the specification for how to construct (the ctor), and the means to do so. Interface as a concept is intended to represent some behavior, such as Heat Up/ Cool Down, that can be applied to arange of things, eg: buildings, ovens, etc – [Sentinel](https://stackoverflow.com/users/442396/sentinel) [Sep 18 '17 at 7:48](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment79510725_24660764)

add a comment

up vote18down vote

Let's work on this question again:

The first thing to let you know is that 1/1 and 1\*1 results in the same, but it does not mean that multiplication and division are same. Obviously, they hold some good relationship, but mind you both are different.

I will point out main differences, and the rest have already been explained:

Abstract classes are useful for modeling a class hierarchy. At first glance of any requirement, we are partially clear on what **exactly** is to be built, but we know **what to build.** And so your abstract classes are your base classes.

Interfaces are useful for letting other hierarchy or classes to know that what I am capable of doing. And when you say I am capable of something, you must have that capacity. Interfaces will mark it as compulsory for a class to implement the same functionalities.

[share](https://stackoverflow.com/a/10100207)[improve this answer](https://stackoverflow.com/posts/10100207/edit)

[edited Nov 20 '16 at 14:18](https://stackoverflow.com/posts/10100207/revisions)

[[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](https://stackoverflow.com/users/63550/peter-mortensen)](https://stackoverflow.com/users/63550/peter-mortensen)

[Peter Mortensen](https://stackoverflow.com/users/63550/peter-mortensen)

**13.2k**1983111

answered Apr 11 '12 at 5:18

[[https://i.stack.imgur.com/d1Hqh.jpg?s=32&g=1](https://stackoverflow.com/users/871726/dhananjay)](https://stackoverflow.com/users/871726/dhananjay)

[Dhananjay](https://stackoverflow.com/users/871726/dhananjay)

**2,674**21618

* Good answer, but the math metaphor is useless and made me waste roughly an equivalent amount of time reading it as writing this comment. Now multiply that by all the other people that have read this question. – [Andrew](https://stackoverflow.com/users/3093731/andrew) [Jan 16 '17 at 22:49](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment70569693_10100207)
* " math metaphor is useless" , why do you think so ? – [Dhananjay](https://stackoverflow.com/users/871726/dhananjay) [Mar 16 '17 at 17:01](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment72789972_10100207)

add a comment

up vote12down vote

It's pretty simple actually.

You can think of an interface as a class which is only allowed to have abstract methods and nothing else.

So an interface can only "declare" and not define the behavior you want the class to have.

An abstract class allows you to do both declare (using abstract methods) as well as define (using full method implementations) the behavior you want the class to have.

And a regular class only allows you to define, not declare, the behavior/actions you want the class to have.

One last thing,

In Java, you can implement multiple interfaces, but you can only extend one (Abstract Class or Class)...

This means inheritance of defined behavior is restricted to only allow one per class... ie if you wanted a class that encapsulated behavior from Classes A,B&C you would need to do the following: Class A extends B, Class C extends A .. its a bit of a round about way to have multiple inheritance...

Interfaces on the other hand, you could simply do: interface C implements A, B

So in effect Java supports multiple inheritance only in "declared behavior" ie interfaces, and only single inheritance with defined behavior.. unless you do the round about way I described...

Hopefully that makes sense.

[share](https://stackoverflow.com/a/24899565)[improve this answer](https://stackoverflow.com/posts/24899565/edit)

[edited Mar 17 '17 at 22:31](https://stackoverflow.com/posts/24899565/revisions)

answered Jul 22 '14 at 23:36

[[https://www.gravatar.com/avatar/1d68adfeb27962cc3c8bf842c8f4ccff?s=32&d=identicon&r=PG](https://stackoverflow.com/users/1500191/g00dnatur3)](https://stackoverflow.com/users/1500191/g00dnatur3)

[g00dnatur3](https://stackoverflow.com/users/1500191/g00dnatur3)

**828**612

add a comment

up vote10down vote

The comparison of interface vs. abstract class is wrong. There should be two other comparisons instead: 1) **interface vs. class** and 2) **abstract vs. final class**.

**Interface vs Class**

**Interface** is a contract between two objects. E.g., I'm a Postman and you're a Package to deliver. I expect you to know your delivery address. When someone gives me a Package, it has to know its delivery address:

interface Package {

String address();

}

**Class** is a group of objects that obey the contract. E.g., I'm a box from "Box" group and I obey the contract required by the Postman. At the same time I obey other contracts:

class Box implements Package, Property {

@Override

String address() {

return "5th Street, New York, NY";

}

@Override

Human owner() {

// this method is part of another contract

}

}

**Abstract vs Final**

**Abstract class** is a group of incomplete objects. They can't be used, because they miss some parts. E.g., I'm an abstract GPS-aware box - I know how to check my position on the map:

abstract class GpsBox implements Package {

@Override

public abstract String address();

protected Coordinates whereAmI() {

// connect to GPS and return my current position

}

}

This class, if inherited/extended by another class, can be very useful. But by itself - it is useless, since it can't have objects. Abstract classes can be building elements of final classes.

**Final class** is a group of complete objects, which can be used, but can't be modified. They know exactly how to work and what to do. E.g., I'm a Box that always goes to the address specified during its construction:

final class DirectBox implements Package {

private final String to;

public DirectBox(String addr) {

this.to = addr;

}

@Override

public String address() {

return this.to;

}

}

In most languages, like Java or C++, it is possible to have **just a class**, neither abstract nor final. Such a class can be inherited and can be instantiated. I don't think this is strictly in line with object-oriented paradigm, though.

Again, comparing interfaces with abstract classes is not correct.

[share](https://stackoverflow.com/a/24944708)[improve this answer](https://stackoverflow.com/posts/24944708/edit)

answered Jul 24 '14 at 21:57

[[https://www.gravatar.com/avatar/f9e28dc83496320c87c55b3cf988ab60?s=32&d=identicon&r=PG](https://stackoverflow.com/users/187141/yegor256)](https://stackoverflow.com/users/187141/yegor256)

[yegor256](https://stackoverflow.com/users/187141/yegor256)

**54.6k**86360500

add a comment

up vote9down vote

The only difference is that one can participate in multiple inheritance and other cannot.

The definition of an interface has changed over time. Do you think an interface just has method declarations only and are just contracts? What about static final variables and what about default definitions after Java 8?

Interfaces were introduced to Java because of [the diamond problem](https://en.wikipedia.org/wiki/Multiple_inheritance#The_diamond_problem) with multiple inheritance and that's what they actually intend to do.

Interfaces are the constructs that were created to get away with the multiple inheritance problem and can have abstract methods, default definitions and static final variables.

See [*Why does Java allow static final variables in interfaces when they are only intended to be contracts?*](http://www.quora.com/Why-does-Java-allow-static-final-variables-in-interfaces-when-they-are-only-intended-to-be-contracts).

[share](https://stackoverflow.com/a/25491625)[improve this answer](https://stackoverflow.com/posts/25491625/edit)

[edited Dec 5 '16 at 16:44](https://stackoverflow.com/posts/25491625/revisions)

answered Aug 25 '14 at 17:53

[Vivek Vermani](https://stackoverflow.com/users/2789764/vivek-vermani)

**1,583**1035

add a comment

up vote8down vote

In short the differences are the following:

Syntactical Differences Between **Interface** and **Abstract Class**:

1. Methods and members of an abstract class can have any visibility. All methods of an **interface**must be **public**. *//Does not hold true from Java 9 anymore*
2. A concrete child class of an **Abstract** Class must define all the abstract methods. An **Abstract**child class can have abstract methods. An *interface* extending another interface need not provide default implementation for methods inherited from the parent interface.
3. A child class can only extend a single class. An **interface** can extend multiple interfaces. A class can implement multiple interfaces.
4. A child class can define abstract methods with the same or less restrictive visibility, whereas class implementing an **interface** must define all interface methods as public.
5. **Abstract Classes** can have constructors but not **interfaces**.
6. Interfaces from Java 9 have private static methods.

In Interfaces now:

public static - supported  
public abstract - supported  
public default - supported  
private static - supported  
private abstract - compile error  
private default - compile error  
private - supported

[share](https://stackoverflow.com/a/44771142)[improve this answer](https://stackoverflow.com/posts/44771142/edit)

[edited Oct 27 '17 at 5:18](https://stackoverflow.com/posts/44771142/revisions)

answered Jun 27 '17 at 1:43

[[https://i.stack.imgur.com/TgQ5O.png?s=32&g=1](https://stackoverflow.com/users/1475228/pritam-banerjee)](https://stackoverflow.com/users/1475228/pritam-banerjee)

[Pritam Banerjee](https://stackoverflow.com/users/1475228/pritam-banerjee)

**10.4k**64263

add a comment

up vote7down vote

Interface: Turn ( Turn Left, Turn Right.)

Abstract Class: Wheel.

Class: Steering Wheel, derives from Wheel, exposes Interface Turn

One is for categorizing behavior that can be offered across a diverse range of things, the other is for modelling an ontology of things.

[share](https://stackoverflow.com/a/42636200)[improve this answer](https://stackoverflow.com/posts/42636200/edit)

answered Mar 6 '17 at 21:46

[[https://www.gravatar.com/avatar/2c7ec89489bfcad5a94119f7bd9a1abf?s=32&d=identicon&r=PG](https://stackoverflow.com/users/442396/sentinel)](https://stackoverflow.com/users/442396/sentinel)

[Sentinel](https://stackoverflow.com/users/442396/sentinel)

**2,147**2031

add a comment

up vote4down vote

Not really the answer to the original question, but once you have the answer to the difference between them, you will enter the when-to-use-each dilemma: [*When to use interfaces or abstract classes? When to use both?*](https://stackoverflow.com/questions/1231985/when-to-use-interfaces-or-abstract-classes-when-to-use-both)

I've limited knowledge of OOP, but seeing interfaces as an equivalent of an adjective in grammar has worked for me until now (correct me if this method is bogus!). For example, interface names are like attributes or capabilities you can give to a class, and a class can have many of them: ISerializable, ICountable, IList, ICacheable, IHappy, ...

[share](https://stackoverflow.com/a/1913233)[improve this answer](https://stackoverflow.com/posts/1913233/edit)

[edited May 23 '17 at 12:02](https://stackoverflow.com/posts/1913233/revisions)

[[https://www.gravatar.com/avatar/a007be5a61f6aa8f3e85ae2fc18dd66e?s=32&d=identicon&r=PG](https://stackoverflow.com/users/-1/community)](https://stackoverflow.com/users/-1/community)

[Community](https://stackoverflow.com/users/-1/community)**♦**

**1**1

answered Dec 16 '09 at 8:48

[[https://www.gravatar.com/avatar/dc2af1ae87ff35c5dd8fdc9d15007304?s=32&d=identicon&r=PG](https://stackoverflow.com/users/28581/azkotoki)](https://stackoverflow.com/users/28581/azkotoki)

[azkotoki](https://stackoverflow.com/users/28581/azkotoki)

**2,047**11925

add a comment

up vote4down vote

If you have some common methods that can be used by multiple classes go for abstract classes. Else if you want the classes to follow some definite blueprint go for interfaces.

Following examples demonstrate this.

Abstract class in Java:

abstract class animals

{

// They all love to eat. So let's implement them for everybody

void eat()

{

System.out.println("Eating...");

}

// The make different sounds. They will provide their own implementation.

abstract void sound();

}

class dog extends animals

{

void sound()

{

System.out.println("Woof Woof");

}

}

class cat extends animals

{

void sound()

{

System.out.println("Meoww");

}

}

Following is an implementation of interface in Java:

interface Shape

{

void display();

double area();

}

class Rectangle implements Shape

{

int length, width;

Rectangle(int length, int width)

{

this.length = length;

this.width = width;

}

@Override

public void display()

{

System.out.println("\*\*\*\*\n\* \*\n\* \*\n\*\*\*\*");

}

@Override

public double area()

{

return (double)(length\*width);

}

}

class Circle implements Shape

{

double pi = 3.14;

int radius;

Circle(int radius)

{

this.radius = radius;

}

@Override

public void display()

{

System.out.println("O"); // :P

}

@Override

public double area()

{

return (double)((pi\*radius\*radius)/2);

}

}

Some Important Key points in a nutshell:

1. The variables declared in Java interface are by default final. Abstract classes can have non-final variables.
2. The variables declared in Java interface are by default static. Abstract classes can have non-static variables.
3. Members of a Java interface are public by default. A Java abstract class can have the usual flavors of class members like private, protected, etc..

[share](https://stackoverflow.com/a/46538339)[improve this answer](https://stackoverflow.com/posts/46538339/edit)

answered Oct 3 '17 at 6:25

[[https://i.stack.imgur.com/aE3So.jpg?s=32&g=1](https://stackoverflow.com/users/7145177/pransh-tiwari)](https://stackoverflow.com/users/7145177/pransh-tiwari)

[Pransh Tiwari](https://stackoverflow.com/users/7145177/pransh-tiwari)

**709**717

add a comment

up vote3down vote

Inheritance is used for two purposes:

* To allow an object to regard parent-type data members and method implementations as its own.
* To allow a reference to an objects of one type to be used by code which expects a reference to supertype object.

In languages/frameworks which support generalized multiple inheritance, there is often little need to classify a type as either being an "interface" or an "abstract class". Popular languages and frameworks, however, will allow a type to regard one other type's data members or method implementations as its own even though they allow a type to be substitutable for an arbitrary number of other types.

Abstract classes may have data members and method implementations, but can only be inherited by classes which don't inherit from any other classes. Interfaces put almost no restrictions on the types which implement them, but cannot include any data members or method implementations.

There are times when it's useful for types to be substitutable for many different things; there are other times when it's useful for objects to regard parent-type data members and method implementations as their own. Making a distinction between interfaces and abstract classes allows each of those abilities to be used in cases where it is most relevant.

[share](https://stackoverflow.com/a/18596865)[improve this answer](https://stackoverflow.com/posts/18596865/edit)

answered Sep 3 '13 at 16:09

[[https://www.gravatar.com/avatar/135a97c4150691234931a93f3a27a52f?s=32&d=identicon&r=PG](https://stackoverflow.com/users/363751/supercat)](https://stackoverflow.com/users/363751/supercat)

[supercat](https://stackoverflow.com/users/363751/supercat)

**55.4k**2114145

add a comment

up vote3down vote

Key Points:

* Abstract class can have property, Data fields ,Methods (complete / incomplete) both.
* If method or Properties define in abstract keyword that must override in derived class.(its work as a tightly coupled functionality)
* If define abstract keyword for method or properties in abstract class you can not define body of method and get/set value for properties and that must override in derived class.
* Abstract class does not support multiple inheritance.
* Abstract class contains Constructors.
* An abstract class can contain access modifiers for the subs, functions, properties.
* Only Complete Member of abstract class can be Static.
* An interface can inherit from another interface only and cannot inherit from an abstract class, where as an abstract class can inherit from another abstract class or another interface.

Advantage:

* It is a kind of contract that forces all the subclasses to carry on the same hierarchies or standards.
* If various implementations are of the same kind and use common behavior or status then abstract class is better to use.
* If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly.
* Its allow fast execution than interface.(interface Requires more time to find the actual method in the corresponding classes.)
* It can use for tight and loosely coupling.

find details here... <http://pradeepatkari.wordpress.com/2014/11/20/interface-and-abstract-class-in-c-oops/>

[share](https://stackoverflow.com/a/27034440)[improve this answer](https://stackoverflow.com/posts/27034440/edit)

[edited Dec 5 '14 at 10:00](https://stackoverflow.com/posts/27034440/revisions)

answered Nov 20 '14 at 7:55

[[https://i.stack.imgur.com/QLwU0.jpg?s=32&g=1](https://stackoverflow.com/users/3211873/pradeep-atkari)](https://stackoverflow.com/users/3211873/pradeep-atkari)

[Pradeep atkari](https://stackoverflow.com/users/3211873/pradeep-atkari)

**300**1413

add a comment

up vote3down vote

The shortest way to sum it up is that an interface is:

1. Fully abstract, apart from default and static methods; while it has definitions (method signatures + implementations) for default and static methods, it only has declarations (method signatures) for other methods.
2. Subject to laxer rules than classes (a class can implement multiple interfaces, and an interface can inherit from multiple interfaces). All variables are implicitly constant, whether specified as public static final or not. All members are implicitly public, whether specified as such or not.
3. Generally used as a guarantee that the implementing class will have the specified features and/or be compatible with any other class which implements the same interface.

Meanwhile, an abstract class is:

1. Anywhere from fully abstract to fully implemented, with a tendency to have one or more abstract methods. Can contain both declarations and definitions, with declarations marked as abstract.
2. A full-fledged class, and subject to the rules that govern other classes (can only inherit from one class), on the condition that it cannot be instantiated (because there's no guarantee that it's fully implemented). Can have non-constant member variables. Can implement member access control, restricting members as protected, private, or private package (unspecified).
3. Generally used either to provide as much of the implementation as can be shared by multiple subclasses, or to provide as much of the implementation as the programmer is able to supply.

Or, if we want to boil it all down to a single sentence: An interface is what the implementing class *has*, but an abstract class is what the subclass *is*.

[share](https://stackoverflow.com/a/35072896)[improve this answer](https://stackoverflow.com/posts/35072896/edit)

answered Jan 28 '16 at 21:58

[[https://www.gravatar.com/avatar/0b3a1b8c1a623eaffa736e29cf37411f?s=32&d=identicon&r=PG&f=1](https://stackoverflow.com/users/5386374/justin-time)](https://stackoverflow.com/users/5386374/justin-time)

[Justin Time](https://stackoverflow.com/users/5386374/justin-time)

**2,952**1328

add a comment

up vote3down vote

Many junior developers make the mistake of thinking of interfaces, abstract and concrete classes as slight variations of the same thing, and choose one of them purely on technical grounds: *Do I need multiple inheritance? Do I need some place to put common methods? Do I need to bother with something other than just a concrete class?* This is wrong, and hidden in these questions is the main problem: **"I"**. When you write code for yourself, by yourself, you rarely think of other present or future developers working on or with your code.

Interfaces and abstract classes, although apparently similar from a technical point of view, have completely different meanings and purposes.

Summary

1. An interface **defines a contract** that some implementation will fulfill *for you*.
2. An abstract class **provides a default behavior** that *your implementation* can reuse.

Alternative summary

1. An interface is for defining public APIs
2. An abstract class is for internal use, and for defining SPIs

On the importance of hiding implementation details

A concrete class does the actual work, in a very specific way. For example, an ArrayList uses a contiguous area of memory to store a list of objects in a compact manner which offers fast random access, iteration, and in-place changes, but is terrible at insertions, deletions, and occasionally even additions; meanwhile, a LinkedList uses double-linked nodes to store a list of objects, which instead offers fast iteration, in-place changes, and insertion/deletion/addition, but is terrible at random access. These two types of lists are optimized for different use cases, and it matters a lot how you're going to use them. When you're trying to squeeze performance out of a list that you're heavily interacting with, and when picking the type of list is up to you, you should carefully pick which one you're instantiating.

On the other hand, high level users of a list don't really care how it is actually implemented, and they should be insulated from these details. Let's imagine that Java didn't expose the List interface, but only had a concrete List class that's actually what LinkedList is right now. All Java developers would have tailored their code to fit the implementation details: avoid random access, add a cache to speed up access, or just reimplement ArrayList on their own, although it would be incompatible with all the other code that actually works with List only. That would be terrible... But now imagine that the Java masters actually realize that a linked list is terrible for most actual use cases, and decided to switch over to an array list for their only List class available. This would affect the performance of every Java program in the world, and people wouldn't be happy about it. And the main culprit is that implementation details were available, and the developers assumed that those details are a permanent contract that they can rely on. This is why it's important to hide implementation details, and only define an abstract contract. This is the purpose of an interface: define what kind of input a method accepts, and what kind of output is expected, without exposing all the guts that would tempt programmers to tweak their code to fit the internal details that might change with any future update.

An abstract class is in the middle between interfaces and concrete classes. It is supposed to help implementations share common or boring code. For example, AbstractCollection provides basic implementations for isEmpty based on size is 0, contains as iterate and compare, addAll as repeated add, and so on. This lets implementations focus on the crucial parts that differentiate between them: how to actually store and retrieve data.

APIs versus SPIs

Interfaces are low-cohesion **gateways** between different parts of code. They allow libraries to exist and evolve without breaking every library user when something changes internally. It's called *Application Programming****Interface***, not Application Programming Classes. On a smaller scale, they also allow multiple developers to collaborate successfully on large scale projects, by separating different modules through well documented interfaces.

Abstract classes are high-cohesion **helpers** to be used when implementing an interface, assuming some level of implementation details. Alternatively, abstract classes are used for defining SPIs, Service Provider Interfaces.

The difference between an API and an SPI is subtle, but important: for an API, the focus is on who **uses** it, and for an SPI the focus is on who **implements** it.

Adding methods to an API is easy, all existing users of the API will still compile. Adding methods to an SPI is hard, since every service provider (concrete implementation) will have to implement the new methods. If interfaces are used to define an SPI, a provider will have to release a new version whenever the SPI contract changes. If abstract classes are used instead, new methods could either be defined in terms of existing abstract methods, or as empty throw not implemented exceptionstubs, which will at least allow an older version of a service implementation to still compile and run.

A note on Java 8 and default methods

Although Java 8 introduced default methods for interfaces, which makes the line between interfaces and abstract classes even blurrier, this wasn't so that implementations can reuse code, but to make it easier to change interfaces that serve both as an API and as an SPI (or are wrongly used for defining SPIs instead of abstract classes).

Which one to use?

1. Is the *thing* supposed to be **publicly used** by other parts of the code, or by other external code? Add an interface to it to hide the implementation details from the public abstract contract, which is the general behavior of the thing.
2. Is the *thing* something that's supposed to have multiple implementations with a lot of code in common? Make both an interface and an abstract, incomplete implementation.
3. Is there ever going to be only one implementation, and nobody else will use it? Just make it a concrete class.
   1. "ever" is long time, you could play it safe and still add an interface on top of it.

A corollary: the other way around is often wrongly done: when using a *thing*, always try to use the most generic class/interface that you actually need. In other words, don't declare your variables as ArrayList theList = new ArrayList(), unless you actually have a very strong dependency on it being an **array** list, and no other type of list would cut it for you. Use List theList = new ArrayListinstead, or even Collection theCollection = new ArrayList if the fact that it's a list, and not any other type of collection doesn't actually matter.

[share](https://stackoverflow.com/a/37905560)[improve this answer](https://stackoverflow.com/posts/37905560/edit)

answered Jun 19 '16 at 9:02

[[https://www.gravatar.com/avatar/31a241046afc0213e3fca220b0c0edc3?s=32&d=identicon&r=PG](https://stackoverflow.com/users/620249/sergiu-dumitriu)](https://stackoverflow.com/users/620249/sergiu-dumitriu)

[Sergiu Dumitriu](https://stackoverflow.com/users/620249/sergiu-dumitriu)

**9,393**32557

add a comment

up vote3down vote

By definition, interfaces cannot have an implementation for any methods, and member variables cannot be initialized.

However, abstract classes can have methods implementated and member variables initialized.

Use abstract classes when you expect changes in your contract, i.e., say in future you might need to add a new method.

In this situation, if you decide to use an interface, when the interface is changed to include interface, your application will break when you dumped the new interface dll.

To read in detail, visit [difference between abstract class and a interface](http://rangahc.blogspot.in/2014/06/difference-between-abstract-class-and-interface-in-CSharp.html)

[share](https://stackoverflow.com/a/28020949)[improve this answer](https://stackoverflow.com/posts/28020949/edit)

[edited Jun 30 '16 at 16:27](https://stackoverflow.com/posts/28020949/revisions)

[[https://www.gravatar.com/avatar/508065acda08f03db54a19db4ca2b6b9?s=32&d=identicon&r=PG&f=1](https://stackoverflow.com/users/6489306/ragingasiancoder)](https://stackoverflow.com/users/6489306/ragingasiancoder)

[ragingasiancoder](https://stackoverflow.com/users/6489306/ragingasiancoder)

**604**416

answered Jan 19 '15 at 8:59

[[https://i.stack.imgur.com/iwxfv.jpg?s=32&g=1](https://stackoverflow.com/users/833265/ranganatha)](https://stackoverflow.com/users/833265/ranganatha)

[Ranganatha](https://stackoverflow.com/users/833265/ranganatha)

**690**519

add a comment

up vote3down vote

I'd like to add one more difference which makes sense. For example, you have a framework with thousands of lines of code. Now if you want to add a new feature throughout the code using a method enhanceUI(), then it's better to add that method in abstract class rather in interface. Because, if you add this method in an interface then you should implement it in all the implemented class but it's not the case if you add the method in abstract class.

[share](https://stackoverflow.com/a/41170330)[improve this answer](https://stackoverflow.com/posts/41170330/edit)

answered Dec 15 '16 at 17:47

[[https://www.gravatar.com/avatar/69b17d899bd7ccb9113eeeb73dd062f3?s=32&d=identicon&r=PG&f=1](https://stackoverflow.com/users/5668588/toothless)](https://stackoverflow.com/users/5668588/toothless)

[Toothless](https://stackoverflow.com/users/5668588/toothless)

**125**312

add a comment

up vote3down vote

**Differences between abstract class and interface on behalf of real implementation.**

**Interface**: It is a keyword and it is used to define the template or blue print of an object and it forces all the sub classes would follow the same prototype,as for as implementation, all the sub classes are free to implement the functionality as per it's requirement.

Some of other use cases where we should use interface.

Communication between two external objects(Third party integration in our application) done through **Interface** here Interface works as Contract.

**Abstract Class:** Abstract,it is a keyword and when we use this keyword before any class then it becomes abstract class.It is mainly used when we need to define the template as well as some default functionality of an object that is followed by all the sub classes and this way it removes the redundant code and **one more use cases where we can use abstract class**, such as we want no other classes can directly instantiate an object of the class, only derived classes can use the functionality.

**Example of Abstract Class:**

public abstract class DesireCar

{

//It is an abstract method that defines the prototype.

public abstract void Color();

// It is a default implementation of a Wheel method as all the desire cars have the same no. of wheels.

// and hence no need to define this in all the sub classes in this way it saves the code duplicasy

public void Wheel() {

Console.WriteLine("Car has four wheel");

}

}

\*\*Here is the sub classes:\*\*

public class DesireCar1 : DesireCar

{

public override void Color()

{

Console.WriteLine("This is a red color Desire car");

}

}

public class DesireCar2 : DesireCar

{

public override void Color()

{

Console.WriteLine("This is a red white Desire car");

}

}

**Example Of Interface:**

public interface IShape

{

// Defines the prototype(template)

void Draw();

}

// All the sub classes follow the same template but implementation can be different.

public class Circle : IShape

{

public void Draw()

{

Console.WriteLine("This is a Circle");

}

}

public class Rectangle : IShape

{

public void Draw()

{

Console.WriteLine("This is a Rectangle");

}

}

[share](https://stackoverflow.com/a/45708929)[improve this answer](https://stackoverflow.com/posts/45708929/edit)

[edited Aug 18 '17 at 5:56](https://stackoverflow.com/posts/45708929/revisions)

answered Aug 16 '17 at 8:43

[[https://i.stack.imgur.com/IAlpU.jpg?s=32&g=1](https://stackoverflow.com/users/5736534/sheo-dayal-singh)](https://stackoverflow.com/users/5736534/sheo-dayal-singh)

[Sheo Dayal Singh](https://stackoverflow.com/users/5736534/sheo-dayal-singh)

**513**55

add a comment

up vote3down vote

You can find clear difference between **interface** and **abstract class.**

**Interface**

* Interface only contains abstract methods.
* Force users to implement all methods when implements the interface.
* Contains only final and static variables.
* Declare using interface keyword.
* All methods of an interface must be defined as public.
* An interface can extend or a class can implement multiple other interfaces.

**Abstract class**

* Abstract class contains abstract and non-abstract methods.
* Does not force users to implement all methods when inherited the abstract class.
* Contains all kinds of variables including primitive and non-primitive
* Declare using abstract keyword.
* Methods and members of an abstract class can be defined with any visibility.
* A child class can only extend a single class (abstract or concrete).

[share](https://stackoverflow.com/a/45863762)[improve this answer](https://stackoverflow.com/posts/45863762/edit)

answered Aug 24 '17 at 14:05

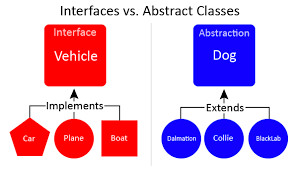
[[https://i.stack.imgur.com/TlzS6.jpg?s=32&g=1](https://stackoverflow.com/users/7333306/rahul-chauhan)](https://stackoverflow.com/users/7333306/rahul-chauhan)

[Rahul Chauhan](https://stackoverflow.com/users/7333306/rahul-chauhan)

**405**49

add a comment

up vote2down vote

[](https://i.stack.imgur.com/OpMyR.png)

Here is a very basic understanding over interface vs abstract class.

[share](https://stackoverflow.com/a/34946694)[improve this answer](https://stackoverflow.com/posts/34946694/edit)

answered Jan 22 '16 at 12:24

[[https://i.stack.imgur.com/faXaW.jpg?s=32&g=1](https://stackoverflow.com/users/5730313/santosh)](https://stackoverflow.com/users/5730313/santosh)

[Santosh](https://stackoverflow.com/users/5730313/santosh)

**297**313

* 11

How does this explain the difference? Why couldn't Car/Plane/Boat *extend* an abstract Vehicle class? – [aioobe](https://stackoverflow.com/users/276052/aioobe" \o "318,323 reputation) [Aug 30 '16 at 6:58](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment65779386_34946694)

* 3

I don't understand, I even don't agree. It contradicts the answer with maximum votes – [whiteletters in blankpapers](https://stackoverflow.com/users/1276136/whiteletters-in-blankpapers" \o "2,869 reputation) [Apr 22 '17 at 11:04](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment74167968_34946694)

* 1

How can this answer get 6 upvotes? – [Pritam Banerjee](https://stackoverflow.com/users/1475228/pritam-banerjee) [Sep 17 '17 at 6:58](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment79486290_34946694)

add a comment

up vote2down vote

An abstract class is a class whose object cannot be created or a class which cannot be instantiated. An abstract method makes a class abstract. An abstract class needs to be inherited in order to override the methods that are declared in the abstract class. No restriction on access specifiers. An abstract class can have constructor and other concrete(non abstarct methods ) methods in them but interface cannot have.

An interface is a blueprint/template of methods.(eg. A house on a paper is given(interface house) and different architects will use their ideas to build it(the classes of architects implementing the house interface) . It is a collection of abstract methods , default methods , static methods , final variables and nested classes. All members will be either final or public , protected and private access specifiers are not allowed.No object creation is allowed. A class has to be made in order to use the implementing interface and also to override the abstract method declared in the interface. An interface is a good example of loose coupling(dynamic polymorphism/dynamic binding) An interface implements polymorphism and abstraction.It tells what to do but how to do is defined by the implementing class. For Eg. There's a car company and it wants that some features to be same for all the car it is manufacturing so for that the company would be making an interface vehicle which will have those features and different classes of car(like Maruti Suzkhi , Maruti 800) will override those features(functions).

Why interface when we already have abstract class? Java supports only multilevel and hierarchal inheritance but with the help of interface we can implement multiple inheritance.

[share](https://stackoverflow.com/a/39602068)[improve this answer](https://stackoverflow.com/posts/39602068/edit)

[edited Sep 20 '16 at 19:19](https://stackoverflow.com/posts/39602068/revisions)

answered Sep 20 '16 at 19:12

[[https://i.stack.imgur.com/wSeP6.jpg?s=32&g=1](https://stackoverflow.com/users/4957686/tutu-kumari)](https://stackoverflow.com/users/4957686/tutu-kumari)

[Tutu Kumari](https://stackoverflow.com/users/4957686/tutu-kumari)

**119**16

add a comment

up vote2down vote

To give a simple but clear answer, it helps to set the context : you use both when you do not want to provide full implementations.

The main difference then is an interface has no implementation at all (only methods without a body) while abstract classes can have members and methods with a body as well, i.e. can be partially implemented.

[share](https://stackoverflow.com/a/44800387)[improve this answer](https://stackoverflow.com/posts/44800387/edit)

answered Jun 28 '17 at 10:49

[user3775501](https://stackoverflow.com/users/3775501/user3775501)

**79**29

* Since you answered it just now, your answer does not take into account the default keyword in Java 8 with which you can define concrete methods in interfaces as well. – [philantrovert](https://stackoverflow.com/users/7224597/philantrovert" \o "6,187 reputation) [Jun 28 '17 at 11:19](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class#comment76581767_44800387)

add a comment

up vote1down vote

In an interface all methods must be only definitions, not single one should be implemented.

But in an abstract class there must an abstract method with only definition, but other methods can be also in the abstract class with implementation...

[share](https://stackoverflow.com/a/18434143)[improve this answer](https://stackoverflow.com/posts/18434143/edit)

[edited Nov 20 '16 at 14:25](https://stackoverflow.com/posts/18434143/revisions)

[[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](https://stackoverflow.com/users/63550/peter-mortensen)](https://stackoverflow.com/users/63550/peter-mortensen)

[Peter Mortensen](https://stackoverflow.com/users/63550/peter-mortensen)

**13.2k**1983111

answered Aug 25 '13 at 21:56

[[https://www.gravatar.com/avatar/66b514555394d0831afc567f7f0fdc15?s=32&d=identicon&r=PG](https://stackoverflow.com/users/757007/shamim-ahmed)](https://stackoverflow.com/users/757007/shamim-ahmed)

[Shamim Ahmed](https://stackoverflow.com/users/757007/shamim-ahmed)

**716**21526

add a comment

up vote1down vote

I read a simple yet effective explanation of Abstract class and Interface on [php.net](http://php.net/manual/en/language.oop5.abstract.php#82111)

Which is as follows.

An Interface is like a protocol. It doesn't designate the behavior of the object; it designates how your code tells that object to act. An interface would be like the English Language: defining an interface defines how your code communicates with any object implementing that interface.

An interface is always an agreement or a promise. When a class says "I implement interface Y", it is saying "I promise to have the same public methods that any object with interface Y has".

On the other hand, an Abstract Class is like a partially built class. It is much like a document with blanks to fill in. It might be using English, but that isn't as important as the fact that some of the document is already written.

An abstract class is the foundation for another object. When a class says "I extend abstract class Y", it is saying "I use some methods or properties already defined in this other class named Y".

So, consider the following PHP:

<?php

class X implements Y { } // this is saying that "X" agrees to speak language "Y" with your code.

class X extends Y { } // this is saying that "X" is going to complete the partial class "Y".

?>

You would have your class implement a particular interface if you were distributing a class to be used by other people. The interface is an agreement to have a specific set of public methods for your class.

You would have your class extend an abstract class if you (or someone else) wrote a class that already had some methods written that you want to use in your new class.

These concepts, while easy to confuse, are specifically different and distinct. For all intents and purposes, if you're the only user of any of your classes, you don't need to implement interfaces.

[share](https://stackoverflow.com/a/50484500)[improve this answer](https://stackoverflow.com/posts/50484500/edit)

answered May 23 at 9:19

[[https://www.gravatar.com/avatar/0f7fb0d32971ab7afc81e2003c9ab8d6?s=32&d=identicon&r=PG&f=1](https://stackoverflow.com/users/4464204/vatsal-shah)](https://stackoverflow.com/users/4464204/vatsal-shah)

[Vatsal Shah](https://stackoverflow.com/users/4464204/vatsal-shah)

**339**214

add a comment

1 [2 next](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class?page=2&tab=votes#tab-top)

**protected** by [Stefano Borini](https://stackoverflow.com/users/78374/stefano-borini) Mar 27 '13 at 18:20

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 [reputation](https://stackoverflow.com/help/whats-reputation) on this site (the [association bonus does not count](https://stackoverflow.com/help/privileges/new-user)).   
  
Would you like to answer one of these [unanswered questions](https://stackoverflow.com/unanswered?fromProtectedNotice=true) instead?

Not the answer you're looking for? Browse other questions tagged [oop](https://stackoverflow.com/questions/tagged/oop" \o "show questions tagged 'oop') [interface](https://stackoverflow.com/questions/tagged/interface) [abstract-class](https://stackoverflow.com/questions/tagged/abstract-class) or [ask your own question](https://stackoverflow.com/questions/ask).

|  |  |
| --- | --- |
| asked | 8 years, 11 months ago |
| viewed | 1,111,707 times |
| active | [14 days ago](https://stackoverflow.com/questions/1913098/what-is-the-difference-between-an-interface-and-abstract-class?lastactivity) |

[42 People Chatting](http://chat.stackoverflow.com/)

[SO Close Vote Reviewers](http://chat.stackoverflow.com/rooms/41570)

1 hour ago - [EJoshuaS](http://chat.stackoverflow.com/users/4032703)

[](http://chat.stackoverflow.com/users/4032703)[https://i.stack.imgur.com/7cmCt.png?s=128&g=1&g&s=32](http://chat.stackoverflow.com/users/3735529)[https://www.gravatar.com/avatar/c2773fec98b1630ec087b575482dc08a?s=23&d=identicon&r=PG](http://chat.stackoverflow.com/users/563532)[https://www.gravatar.com/avatar/921df025d1d7c99fad4553ac83bd8bdf?s=23&d=identicon&r=PG](http://chat.stackoverflow.com/users/441757)[https://i.stack.imgur.com/erE0E.jpg?s=128&g=1&g&s=32](http://chat.stackoverflow.com/users/6373379)[https://www.gravatar.com/avatar/16b6783e9b99b1199c75761eaf5c0685?s=23&d=identicon&r=PG](http://chat.stackoverflow.com/users/522444)[https://i.stack.imgur.com/lStUs.jpg?s=128&g=1&g&s=32](http://chat.stackoverflow.com/users/5781745)

[SOBotics](http://chat.stackoverflow.com/rooms/111347)

1 hour ago - [Jammy](http://chat.stackoverflow.com/users/5675570)

[https://i.stack.imgur.com/1Ab8s.png?s=128&g=1&g&s=32](http://chat.stackoverflow.com/users/7829893)[https://i.stack.imgur.com/58jhk.jpg?s=128&g=1&g&s=32](http://chat.stackoverflow.com/users/6294609)[https://i.stack.imgur.com/aMUMt.jpg?s=128&g=1&g&s=32](http://chat.stackoverflow.com/users/6817005)[https://www.gravatar.com/avatar/921df025d1d7c99fad4553ac83bd8bdf?s=23&d=identicon&r=PG](http://chat.stackoverflow.com/users/441757)[https://i.stack.imgur.com/IuV0f.jpg?s=128&g=1&g&s=32](http://chat.stackoverflow.com/users/7240793)[https://i.stack.imgur.com/7cmCt.png?s=128&g=1&g&s=32](http://chat.stackoverflow.com/users/3735529)[https://i.stack.imgur.com/erE0E.jpg?s=128&g=1&g&s=32](http://chat.stackoverflow.com/users/6373379)

Linked

[6](https://stackoverflow.com/q/15857982?lq=1" \o "Vote score (upvotes - downvotes))

[Difference Between Interface and Class](https://stackoverflow.com/questions/15857982/difference-between-interface-and-class?noredirect=1&lq=1)

[5](https://stackoverflow.com/q/3577096?lq=1" \o "Vote score (upvotes - downvotes))

[what's the difference between abstract and interface in php?](https://stackoverflow.com/questions/3577096/whats-the-difference-between-abstract-and-interface-in-php?noredirect=1&lq=1)

[3](https://stackoverflow.com/q/4607081?lq=1" \o "Vote score (upvotes - downvotes))

[What are the advantages of interfaces and abstract classes?](https://stackoverflow.com/questions/4607081/what-are-the-advantages-of-interfaces-and-abstract-classes?noredirect=1&lq=1)

[2](https://stackoverflow.com/q/27777641?lq=1" \o "Vote score (upvotes - downvotes))

[Why should we use interface if we can simply override methods of the superclass or use abstract classes?](https://stackoverflow.com/questions/27777641/why-should-we-use-interface-if-we-can-simply-override-methods-of-the-superclass?noredirect=1&lq=1)

[6](https://stackoverflow.com/q/29675297?lq=1" \o "Vote score (upvotes - downvotes))

[Advantages of Abstraction and Polymorphism in Java](https://stackoverflow.com/questions/29675297/advantages-of-abstraction-and-polymorphism-in-java?noredirect=1&lq=1)

[-7](https://stackoverflow.com/q/18119521?lq=1" \o "Vote score (upvotes - downvotes))

[Difference between Interface And Abstract Class](https://stackoverflow.com/questions/18119521/difference-between-interface-and-abstract-class?noredirect=1&lq=1)

[2](https://stackoverflow.com/q/10720258?lq=1" \o "Vote score (upvotes - downvotes))

[PHP, OOP interface and abstraction](https://stackoverflow.com/questions/10720258/php-oop-interface-and-abstraction?noredirect=1&lq=1)

[-2](https://stackoverflow.com/q/10185658?lq=1" \o "Vote score (upvotes - downvotes))

[What is the difference between an interface and an abstract class in Java?](https://stackoverflow.com/questions/10185658/what-is-the-difference-between-an-interface-and-an-abstract-class-in-java?noredirect=1&lq=1)

[-5](https://stackoverflow.com/q/15026591?lq=1" \o "Vote score (upvotes - downvotes))

[What's the difference between interface and abstract and how about Dependency Injection](https://stackoverflow.com/questions/15026591/whats-the-difference-between-interface-and-abstract-and-how-about-dependency-in?noredirect=1&lq=1)

[1](https://stackoverflow.com/q/20580748?lq=1" \o "Vote score (upvotes - downvotes))

[what is the behaviour abstract class and interface in c#?](https://stackoverflow.com/questions/20580748/what-is-the-behaviour-abstract-class-and-interface-in-c?noredirect=1&lq=1)

[see more linked questions…](https://stackoverflow.com/questions/linked/1913098?lq=1)

Related

[870](https://stackoverflow.com/q/54867?rq=1" \o "Vote score (upvotes - downvotes))

[What is the difference between old style and new style classes in Python?](https://stackoverflow.com/questions/54867/what-is-the-difference-between-old-style-and-new-style-classes-in-python?rq=1)

[4789](https://stackoverflow.com/q/100003?rq=1" \o "Vote score (upvotes - downvotes))

[What are metaclasses in Python?](https://stackoverflow.com/questions/100003/what-are-metaclasses-in-python?rq=1)

[2831](https://stackoverflow.com/q/136097?rq=1" \o "Vote score (upvotes - downvotes))

[What is the difference between @staticmethod and @classmethod?](https://stackoverflow.com/questions/136097/what-is-the-difference-between-staticmethod-and-classmethod?rq=1)

[743](https://stackoverflow.com/q/318064?rq=1" \o "Vote score (upvotes - downvotes))

[How do you declare an interface in C++?](https://stackoverflow.com/questions/318064/how-do-you-declare-an-interface-in-c?rq=1)

[461](https://stackoverflow.com/q/372042?rq=1" \o "Vote score (upvotes - downvotes))

[Difference between abstract class and interface in Python](https://stackoverflow.com/questions/372042/difference-between-abstract-class-and-interface-in-python?rq=1)

[1358](https://stackoverflow.com/q/391483?rq=1" \o "Vote score (upvotes - downvotes))

[What is the difference between an abstract function and a virtual function?](https://stackoverflow.com/questions/391483/what-is-the-difference-between-an-abstract-function-and-a-virtual-function?rq=1)

[2062](https://stackoverflow.com/q/576169?rq=1" \o "Vote score (upvotes - downvotes))

[Understanding Python super() with \_\_init\_\_() methods](https://stackoverflow.com/questions/576169/understanding-python-super-with-init-methods?rq=1)

[1240](https://stackoverflow.com/q/761194?rq=1" \o "Vote score (upvotes - downvotes))

[Interface vs Abstract Class (general OO)](https://stackoverflow.com/questions/761194/interface-vs-abstract-class-general-oo?rq=1)

[983](https://stackoverflow.com/q/1549801?rq=1" \o "Vote score (upvotes - downvotes))

[What are the differences between type() and isinstance()?](https://stackoverflow.com/questions/1549801/what-are-the-differences-between-type-and-isinstance?rq=1)

[401](https://stackoverflow.com/q/18777989?rq=1" \o "Vote score (upvotes - downvotes))

[How should I have explained the difference between an Interface and an Abstract class?](https://stackoverflow.com/questions/18777989/how-should-i-have-explained-the-difference-between-an-interface-and-an-abstract?rq=1)

[Hot Network Questions](https://stackexchange.com/questions?tab=hot)

* [Uncertainty principle for a sitting person](https://physics.stackexchange.com/questions/440399/uncertainty-principle-for-a-sitting-person)
* [What is the purpose of the Evocation wizard's Sculpt Spells feature?](https://rpg.stackexchange.com/questions/135233/what-is-the-purpose-of-the-evocation-wizards-sculpt-spells-feature)
* [All the single eights](https://codegolf.stackexchange.com/questions/175724/all-the-single-eights)
* [Will these two transistors (BC558A and BC558B) function in the same way?](https://electronics.stackexchange.com/questions/406230/will-these-two-transistors-bc558a-and-bc558b-function-in-the-same-way)
* [Why isn’t ‘because’ a logical connective in propositional logic?](https://math.stackexchange.com/questions/2993855/why-isn-t-because-a-logical-connective-in-propositional-logic)
* [What are the benefits of becoming an academic administrator?](https://academia.stackexchange.com/questions/119646/what-are-the-benefits-of-becoming-an-academic-administrator)
* [Deploy Salesforce Shield components in production sandbox](https://salesforce.stackexchange.com/questions/238988/deploy-salesforce-shield-components-in-production-sandbox)
* [How to prevent my child from being indoctrinated with Christianity?](https://parenting.stackexchange.com/questions/35218/how-to-prevent-my-child-from-being-indoctrinated-with-christianity)
* [Should I volunteer for a pay cut?](https://workplace.stackexchange.com/questions/122412/should-i-volunteer-for-a-pay-cut)
* [Fear of web-app not being "future-proof"](https://softwareengineering.stackexchange.com/questions/381047/fear-of-web-app-not-being-future-proof)
* [How to respond to hiring manager unexpectedly acting dismissive](https://workplace.stackexchange.com/questions/122241/how-to-respond-to-hiring-manager-unexpectedly-acting-dismissive)
* [I'm supposedly eligible for an inheritance from a distant relative. Offer appears to be legitimate](https://money.stackexchange.com/questions/101935/im-supposedly-eligible-for-an-inheritance-from-a-distant-relative-offer-appear)
* [How to handle lecturer who doesn't let me use my phone?](https://academia.stackexchange.com/questions/119373/how-to-handle-lecturer-who-doesnt-let-me-use-my-phone)
* [Where do extra planes come from?](https://aviation.stackexchange.com/questions/56957/where-do-extra-planes-come-from)
* [Should I ask for a pay raise for employees I manage?](https://workplace.stackexchange.com/questions/122351/should-i-ask-for-a-pay-raise-for-employees-i-manage)
* [Keeping my ears warm while biking in the cold with a helmet](https://bicycles.stackexchange.com/questions/57992/keeping-my-ears-warm-while-biking-in-the-cold-with-a-helmet)
* [humans born insane](https://scifi.stackexchange.com/questions/198325/humans-born-insane)
* [Is my paladin/warlock multiclass a viable tank?](https://rpg.stackexchange.com/questions/135325/is-my-paladin-warlock-multiclass-a-viable-tank)
* [Compute count of unique words using ES6 sets](https://codereview.stackexchange.com/questions/207429/compute-count-of-unique-words-using-es6-sets)
* [Notation of Derivatives](https://math.stackexchange.com/questions/2994700/notation-of-derivatives)
* [Why does the vacuum even have permeability and permittivity?](https://physics.stackexchange.com/questions/440007/why-does-the-vacuum-even-have-permeability-and-permittivity)
* [What are the rules for PC-PC seduction attempts?](https://rpg.stackexchange.com/questions/134899/what-are-the-rules-for-pc-pc-seduction-attempts)
* [Was it a cat I saw?](https://puzzling.stackexchange.com/questions/75011/was-it-a-cat-i-saw)
* [«θύων τε γὰρ φανερὸς ἦν πολλάκις μὲν οἴκοι» (Xenophon, Memorabilia, 1.1.2)](https://latin.stackexchange.com/questions/7514/%ce%b8%cf%8d%cf%89%ce%bd-%cf%84%ce%b5-%ce%b3%e1%bd%b0%cf%81-%cf%86%ce%b1%ce%bd%ce%b5%cf%81%e1%bd%b8%cf%82-%e1%bc%a6%ce%bd-%cf%80%ce%bf%ce%bb%ce%bb%ce%ac%ce%ba%ce%b9%cf%82-%ce%bc%e1%bd%b2%ce%bd-%ce%bf%e1%bc%b4%ce%ba%ce%bf%ce%b9-xenophon-memorabilia-1-1-2)