

# Interface Contract, Class Object?

[Ask Question](#)

▲ Is contract to interface as object is to class?

20 ▼

★

6

class interface object contract

asked Oct 20 '08 at 18:39

[Adron](#)

1,501

7

21

30

I see the difference now, thx for the answers. I've used them for years, albeit well and accurately from what I've seen, but I never really stopped to actually think the definitions through. — [Adron](#) Oct 20 '08 at 19:23

## 6 Answers

[Home](#)[PUBLIC](#)[Stack Overflow](#)[Tags](#)[Users](#)[Jobs](#)

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

**Teams**

Q&amp;A for work

[Learn More](#)

22



An interface is an abstract class (in languages like Java where there is no multiple inheritance, sometimes there are other restrictions, such as a separate data type) that is intended to be used as a common base to access a number of similarly-behaving objects. Conceptually, there is no requirement for abstractness, but usually, an interface will have at least one abstract method. An interface is a method for your program to communicate with a number of similar classes, each with different semantics but the same general purpose.

## Contract

A contract is the implicit agreement you make between users and implementers of a class or interface. For instance, preconditions and postconditions (invariants are usually a contract within the class' implementation - generally, things like the relation between internal members don't need to be exposed). The specification for a return value or an argument can also be part of the contract. It basically represents how to use the function/class/interface, and isn't generally fully representable in any language (some languages, like Eiffel, allow you to put explicit contracts in, but even these can't always fully flesh out the requirements). When you implement an interface or derive from a class, you are always having to meet the interface requirements, or, when overriding a non-abstract class, to behave similar enough that an external viewer doesn't notice the difference (this is the Liskov Substitution Principle; a derived object should be capable of replacing the base with no difference in behavior from the outside perspective).

## Class

A class doesn't need a lot of going over, since you clearly have used them before. A class is the data type, and in some languages is a superset of interfaces (which have no formal definition, as in C++), and in others is independent (such as in Java).

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

An object is an instance of a class type (or of any non-class type, usually). The exact definition of an object is very specific to a language, but the general definition is the actual thing referred to by multiple references/pointers to the same thing - for instance, in some languages like Java, `==` compares whether two variables are the same object, not necessarily whether they are semantically the same. Objects are independent from classes or interfaces - they represent a single instance. Another way of thinking of it is that class or interface is the mold, and the object is the physical object that comes out of the mold (a rather bad analogy, but it's the best I can come up with right now).

answered Oct 20 '08 at 19:08




[coppo](#)

12.6k 3 51 69

---

Not sure about other languages, but in C# `interface` and `abstract class` are similar, however are different concepts; they are not the same and cannot always be used interchangeably. – [iliketocode](#) Jul 7 '18 at 6:45

---

I would not still omit the "implicit" - that doesn't necessarily describe the nature of what I understand as contract. Generally a type system allows the programmer to express constraints on implementations. As such, a type definition - such as an interface in Java - can very well be viewed an explicit contract which is fulfilled by a type implementing it. – [IARI](#) Apr 18 at 10:56 

---

See for example this cseducators post:  
[cseducators.stackexchange.com/questions/544/...](https://cseducators.stackexchange.com/questions/544/...) – [IARI](#) Apr 18 at 11:09

---

7

No, not really. A class is a template that you define. Each object that instantiates that class follows the template. They're not really redundant terms, because the two things are not identical. You can think of a class as a user-defined data type. Classes and objects are different from each other in the exact same way that the primitive data type `int` is different from the literal value 3.

An interface defines a set of methods that all implementing classes must support. The interface itself is the contract that you define for the implementing classes. It just says that any class that implements the interface, must have that interface's set of public methods.

edited Oct 20 '08 at 19:04

answered Oct 20 '08 at 18:58



[Bill the Lizard](#)

**300k** 159 501 795

1

Well I guess... if an interface specifies a contract than a class specifies an (or multiple) instance(s) of a particular object.

Terminology is less important than application though.

answered Oct 20 '08 at 18:57



[Quibblesome](#)

**21.3k** 10 52 94

1

Actually, an interface is a contract, when an object is an instance of a class - they are different things that don't have too much in common.

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

For example, you can have two classes implementing the same interface/contract, but do totally different things (even though the meaning of doing them may be the same).

Take the IDisposable interface for example: Each object can release the resources that it uses, but it can do it in many different ways, it can choose not to release anything. It's the object's choice.

At least this would be the POV in .NET

edited Oct 20 '08 at 19:14

answered Oct 20 '08 at 18:57



Bogdan Maxim

3,411 3 19 33



To complete the previous answers, a word about interfaces:

1

If the class is more than a template for an object (because of its global characteristics independent of any instances), the interface can be also be described as a ***point of view***



A class implementing several interface:

- complete the contract it needs to respect
- allow the user to see any instances of that class from the *point of view* of the one represented by the implemented interface.

"Point of view" means you can using an object by focusing solely on the contract define by that interface.

It is in that aspect an interface is an "abstract class", as in an

an interface leaves actually a lot out, since it can only be applied to define contract for instance, not for static methods or functions.

edited Oct 20 '08 at 19:43

answered Oct 20 '08 at 19:29



VonC

867k

310

2773

3345



0



"Class" and "Object" represent two different things; they are related, but what they represent IS different, quite strongly.

The best way to describe this is to look at Static. A class can have static members, which are completely separate from any INSTANCE of that class. Objects of that class may or may not use those static members; but the instance of the object of that class is completely separate from any static uses of that class (or should be, at the very least).

Or think of the singleton pattern. Storing an instance of the class object in a static class accessor is a common practice, and shows the difference. You refer to the **class static accessor** to get the **object instance** of a singleton class; if the **class static member** does not have an **object instance** to refer to, the **class** creates the **instance** of the **object**.

Put another way; an object **is** an instance of a class; but a class can be **more** than just a template from which objects are instantiated. Static members of classes have a representation in memory that is completely independent of object instances of those classes.

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

**33.5k**

3

66

108

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).