'method' vs. 'message' vs. 'function' vs. '???'

Asked 16 years, 3 months ago Modified 10 years, 11 months ago

Viewed 11k times

34

I recently asked a question about what I called "method calls". The answer referred to "messages". As a self-taught hobby programmer trying to phrase questions that don't make me look like an idiot, I'm realizing that the terminology that I use reveals a lot about how I learned to program.

Is there a distinction between the various terms for methods/messages/etc. in OO programming? Is this a difference that comes from different programming languages using different terminology to describe similar concepts?

I seem to remember that in pre-OO languages, a distinction would sometimes be made between "subroutines" and "functions" based on whether a return value was expected, but even then, was this a language-by-language distinction?

language-agnostic terminology

Share

asked Sep 4, 2008 at 13:39

Improve this question

Follow

Nathan Fritz

2,328 ● 3 ● 20 ● 14

11 Answers

Sorted by: Highest score (default)

18

I've found this to be a language and programming-paradigm thing. One paradigm — OOP — refers to objects with member methods, which conceptually are how you send messages to those objects (this view is reflected in UML, for example).

Another paradigm — functional — may or may not involve classes of objects, but functions are the atomic unit of work.

In structured programming, you had sub-routines (notice that the prefix "sub" implies structure).

In imperative programming (which overlaps with structured quite a lot, but a slightly different way of looking at things), you have a more formulaic view of the world, and so 'functions' represent some operation (often mathematical).

All you have to do to not sound like a rube is to use the terminology used by the language reference for the language you're using.

Share Improve this answer

Follow

edited Jan 14, 2014 at 17:47

answered Sep 4, 2008 at 13:45

Ben Collins
20.7k ● 18 ● 128 ● 190

7 If functions are atomic, does that make a statement a neutron? An identifier a quark? And where's the Higgs boson? – Scottie T Mar 6, 2009 at 14:55

3 The Higgs boson are the exceptions. An inevitable part of excistence, but darn hard to find when you are looking for them. – JonC Nov 8, 2011 at 22:06

Just to avoid confusion: Atomic operations in concurrent programming are program operations that run completely, in one go, without the risk of interference from other processes. For instance a int assignment in java is atomic whereas a long assignment isn't guaranteed to be. In this sense functions are NOT, in general, atomic. – Tore Rudberg Jan 14, 2014 at 9:42

11

Message!=Method!=function

in OOP different *objects* may have different *methods* bound to the same *message*.

for example: the message "rotate left n degrees" would be implemented diffrently by diffrent objects such as shape, circle, rectangle and square.

Messages: Objects communicate through messages.

-Objects send and recieve messages.

-the response to a message is executing a method.

-the method to use is determine be the reciever at run-time.

In C++ Methods and Messages are called **function members.**

Share Improve this answer

Follow

edited Jun 14, 2009 at 16:09

brian d foy

132k ● 31 ● 211 ● 604

answered Mar 6, 2009 at 14:50

user74722

1 C++ doesn't really use message passing. Objects have member functions that are sometimes called methods and you call those directly rather than passing a message symbol. – Chuck Oct 13, 2009 at 23:46

It's not about message passing or having member functions, it's about logic flow. You don't do any permutations of data within messages, you only send information from one object to another, hence "message". When you want to mess with the data/object you should be using methods. Function is a basic programming term, both Messages/Methods are functions, hell even a constructor is a function. Usually to keep your logic flow clean and understandable, you send messages when data needs to be worked/objects need to be updated. The end receiver of the message performs methods to complete the action. – Mike Jan 14, 2014 at 17:53

I should add that obviously Methods are objects member functions, and that messages are also usually member functions but with the sole purpose of being an informant or passing data. Methods (or member functions) should be used when altering or creating new data within that object.

> Encapsulation is the reasoning we use messages and it's why we don't call them methods, because they are not the method of that object/class, they are just informing the class/object that should be working with this information.
> – Mike Jan 14, 2014 at 17:57

4

In Object Oriented implementations like C#, the concept of a "message" does not really exist as an explicit language construct. You can't look at a particular bit of code and say "there's the message."

Instead, a method of an object's class implies the idea that other objects can send a type of message which trigger the behaviour within that method. So you end up just specifying the method directly, rather than sending a message.

With other implementations like Smalltalk, you can see the message being passed, and the receiving object has the ability to do with that message what it will.

There are libraries which sit on top of languages such as C# which attempt to restore the explicit message passing feel to the language. I've been cooking up one of my own for fun here: http://collaborateframework.codeplex.com/

Share Improve this answer

Follow

answered Jun 2, 2011 at 23:12

goofballLogic

40.3k ● 9 ● 46 ● 63

I believe *message* is used in smalltalk.

Java, C# etc. tend to use *method* or *instance method*.

3

Share Improve this answer

Follow

answered Sep 4, 2008 at 13:45

Daren Thomas

70.2k ● 42 ● 155 ● 205

I am pretty sure (but a quick Wikipedia check seems to confirm this) that the `message passing' terminology comes from the Smalltalk community. I think it is more or less equivalent to a method call.

3

Share Improve this answer

Follow

answered Sep 4, 2008 at 13:46

lindelof

35.2k ● 31 ● 102 ● 144

The "Message" term can refer to sending a message to an object, which is supported in some programming languages and not others.

2

If the object supports the message, then it will execute some code. Otherwise it will just ignore it. This is a more dynamic approach than an explicit function/method call where the object must support that function.

Objective-c, I believe, uses this messaging approach.

Share Improve this answer

Follow

answered Sep 4, 2008 at 13:53

John Sibly

23k ● 7 ● 64 ● 82

0

I'm not sure about origin of *message* terminology. Most ofter I encounter *messages* in UML design. Objects (Actors in UML terminology) can communicate with each other by means of *messages*. In real-world code *message* is just a function call usually. I think of message as of attempt to communicate with some object. It can be a real message (like messages in OS) or function calls.

Share Improve this answer

Follow

answered Sep 4, 2008 at 13:42

aku

124k ● 33 ● 176 ● 203

3 The origin is an old design of Smalltalk. Alan Kay briefly toyed with the actor model of concurrency and made every single object its own actor running in its own thread. In that system, method calls really *were* asynchronous message sends. This design was quickly reverted, the terminology wasn't. – Jörg W Mittag Jan 14, 2009 at 19:57

0

Usually, "Method" seems to be the proper name for Functions. However, each language has it's own keywords. Delphi for example even makes a difference between Methods that return something ("Functions") and Methods that return Nothing ("Procedures") whereas in C-Type languages, there is no difference.

Share Improve this answer

Follow

answered Sep 4, 2008 at 13:44

Michael Stum

181k ● 119 ● 407 ● 541

0

Here's some simplified definitions:

methods/subroutines/voids: perform an action

functions: perform an action and return a value

events: are called when an object is acted upon

handlers: are the functions/methods that handle the events

PS: this is a perfect example of why SO should support DL/DT/DD tags.

Share Improve this answer

Follow

answered Sep 4, 2008 at 13:49

travis

36.4k ● 21 ● 72 ● 97

0

I believe that it is a matter of preference at this point. The words that you mention are basically synonyms in today's languages and for the most part people will understand what you mean if you say either "method" or "function". If you use "message", which is only used really in OOP, then you may confuse what you are attempting to convey.For example: "I need to create a message to send an email message." Other terms that could be

synonymous, and this isn't a complete list, are subroutine, action, procedure, operation (although usually mathematical in nature), subprogram, command...

Share Improve this answer

Follow

answered Sep 4, 2008 at 13:58

martinatime

2,487 ● 1 ● 17 ● 23

0

method : similar to function in traditional languages

message : similar to parameter passing in traditional language

Share Improve this answer

Follow

answered Oct 13, 2009 at 23:23

user189485