

28 May 2006

By: Vlad Tarko, Senior Editor, Sci-Tech News



## [The Metaphysics of Object Oriented Programming](#)

*Plato would have been proud of today's computer programmers*

I'm not really a very good programmer, because I lack patience and I'm easily pissed off by bugs impossible to find. But, nonetheless, I find the *idea* of programming very attractive, and I'm curious to know the inner workings of computer programs and web sites. Moreover, my curiosity often goes off track into awful philosophical considerations ... such as this one. Unlike traditional programming that was an endless combination of loops and conditions, object oriented programming (OOP) is much more fun. (Actually, we should note: programmers have always hoped their programs don't contain any literally *endless* loops - that's what a so called program crash is - when it crashes, the program doesn't stop working, on the contrary, it gets stuck in a loop without exit - but unfortunately for them Godel's theorems prove it's impossible to demonstrate whether a program contains endless loops.) Anyway, although at ground level OOP is still actually made of loops and conditions, it sure looks different from a programmer's point of view. In case of OOP you have to deal with so called *objects* (such as a button or a window or a menu or whatever you invent) and these objects have properties, events and methods. Traditional programming was causal, while OOP is intentional. Traditionally, you would tell the computer something like: "Do this, then that, if the user does that go that way, if (s)he does that other thing go the other way, then enter this loop and continuously check and check and check whether the user has done anything you have plans for, and God forbid (s)he does something unexpected, and so on." Usually, you would have to predict any possible action the user could do and devise a certain response. In OOP you are not programming in such a causal manner, you are actually setting various purposes or functions to various objects. You say "The purpose of this button over here is to open up that window", "The purpose of this text box is to get some text from the user." And so on. This allows you to create much more complex programs in a more bug-free manner. A *property* is how the object looks like. For example, it has a certain color, or a certain text imprinted on it, or has a certain position inside a window and so on. When you change the value of some property, the object miraculously changes its appearance. You're just saying "I want you to be like that", and it becomes like that. An *event* is something that can happen to the object. For example a button can be clicked. Or the mouse can go over it. A text can be changed. Etc. You, as a programmer, can say to an object: "If this event ever happens to you, this is how I want you to react." A *method* is something that the object can do. While events are things inflicted on objects, methods are the things the objects themselves do. For instance, a text box can undo a text change or can copy the selected text to Clipboard. What's fun from a philosophical point of view is that this OOP creates a very platonic virtual world. Plato's most famous idea is that everything in the real world, everything that we see and ourselves included, is only a sort of copy of some "archetypal" or ideal object. In the same way, every object OOP deals with is created from a so-called *class*. A class is the ideal, platonic object. A class consists in a set of properties without a specified value, having empty events and methods. To create a particular virtual object (that you can actually see on screen or with what you can actually do anything) you assign certain values to the ideal object (to the class), and write some code for its events and methods. The class defines in a sort of abstract manner what all the possible objects of that type could be like. A class doesn't define a particular button, but defines the concept of *any* button. Translating Plato into OOP language: he believed the true world is the world of classes only, while the world of objects is only apparent (he used the following metaphor: objects are like the shadows of the classes, and we are only seeing these shadows, wrongfully taking them for real). In other words, he believed God was a computer

programmer that did the job in a divine Visual Basic. The philosopher most famous for debunking Plato's vision of the world is Wittgenstein. Wittgenstein's most famous idea is that objects in the real world are not the outcome of some platonic classes, but that they come in "families". A family is a set of real objects that resemble each other from various points of view. For example, the family "chair" is a set of all the objects that we call by that name. But we group together these objects in one single family, not because they share some common essence, but out of historic accidents. For example, in Romanian a toilet seat is called a "toilet chair" (because Romanians read sitting on the toilet more often than the English). To Romanians the toilet seat is part of the "chair" family, while the English have a more restrictive concept of "chair". For describing his idea, Wittgenstein used the metaphor of a rope: a rope is made of many fibers, but there's no fiber that has the same length as the entire rope. In the same way, object A resembles object B, which resembles object C, and which, in turn, resemble object D, and all these objects may be part of the same family, but nonetheless, object A may not resemble object D in any dramatic manner. In other words, there's no common thread through all the objects called by a certain name - i.e. belonging to a certain family. I've been wondering how a Wittgensteinian programming language would look like. Suppose the buttons and all the windows you could encounter in this virtual world would not arise from certain classes, but they would belong to a family. I think the virtual world would be much more diverse and fun, and, probably, any programmer's nightmare. In order to create a new object, you would have to take an existing object and modify it in various ways. But there would not be any predetermined class to tell you in what ways you may modify it - a predetermined concept of "all" objects of that type. I guess such a Wittgensteinian standard would be great for the open source programs.

*Cartoon by: Rich Tennant*