1 OOP in Perl

Compared to traditional class-based OOP found in Ruby, Perl offers a system that is much more flexible and serves as middle-ground between class-based and prototype-based systems. OOP in Perl has no access control, all class fields and methods are public. Class methods can be called in any context during runtime such that methods can be shared between classes without inheritance or even used as a static method. Classes in Perl has no specific syntax for constructors, instead factory methods are used.

2 Scoping in C++ and Perl

Aside from the traditional local and global scoping found in C++, the Perl language also offers dynamic scoping which allows variables to be bound to the call-stack frame instead of the lexical block. In my 'correct.pl' program all of the variables are declared with 'my' except for 'creditLimit' which is global. This variable is shadowed by a 'local' version when 'premierClient' is called.

3 Dynamic Scoping

Dynamic scoping is a useful language feature that provides an execution context to code without the use of OOP, but much like global variables, they must be used with care because programmers cannot easily trace their contents by looking at the code. Personally I wouldn't say that this feature is required in a programming language because I could easily emulate its behaviour in C/C++ with 2 extra lines. (See 'local.cpp:premierClient')