International Journal of Engineering, Applied and Management Sciences Paradigms, Vol. 01, Issue 01, February 2013
ISSN (Online): 2320-6608
www.ijeam.com

The Use of Object Oriented Programming Techniques in Industry

Dr. R.K. Chauhan

Former Professor, DU New Delhi

Abstract

The OOPs programming is very much important in many fields. It is very much important for researchers and also for industrialists. The concept of information hiding, encapsulation and other related features of this are used in many implementations. The main aim of this paper is to understand some features of OOPs and apply them to the implementation by important programming techniques.

Keywords: *OOPs Features, Methodologies of OOPs, IH Techniques.*

Introduction

OOPs is the programming in which program is seen as a collection of subroutines. Each object can be viewed as an independent "machine" with a distinct role or responsibility. The actions (or "methods") on these objects are closely associated with the object. An object-oriented program will usually contain different types of objects, each type corresponding to a particular kind of complex data to be managed or perhaps to a real-world object or concept such as a bank account, a hockey player, or a bulldozer. A program might well contain multiple copies of each type of object, one for each of the real-world objects the program is dealing with. For instance, there could be one bank account object for each real-world account at a particular bank. Each copy of the bank account object would be alike in the methods it offers for manipulating or reading its data, but the data inside each object would differ reflecting the different history of each account.

Methodologies

Information Hiding

This is the concept used when the design may be changed in future and we have to make a blueprint of the system. Information hiding is the ability to prevent certain aspects of a class or software component from being accessible to its clients. This provides the system an interface which protects the system from the implementation. This concept may be used in hardware and software. The concept can be used with less data and if we are having the model in our mind so that we can change it in future.

Encapsulation

Sometimes information hiding and encapsulation are used interchangeably, but the difference between these two is that information hiding is principle and the encapsulation is application to implement the principle. The idea of encapsulation is more general how it is applied in the concept of object oriented programming. For Example, a relational database is

encapsulated in the sense that its only public interface is a Query language (SQL for example), which hides all the internal machinery and data structures of the database management system.

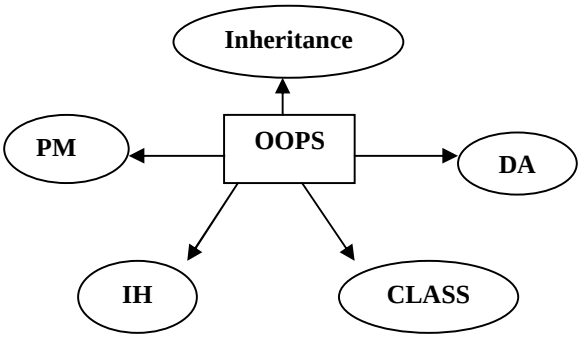

Figure 1: The OOPs

Implementation

The engineers design the car by dividing the task up into pieces of work which are assigned to teams. Each team then designs their component to a particular standard or interface which allows the sub-team flexibility in the design of the component while at the same time ensuring that all of the components will fit together.

Motor vehicle manufacturers frequently use the same core structure for several different models, in part as a cost-control measure. Such a "platform" also provides an example of information hiding, since the floorpan can be built without knowing whether it is to be used in a sedan or a hatchback.

The concept of OOPs is used in the following program:

```
class SquareSumComputer:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def transform(self, x):
        return x * x

    def inputs(self):
        return range(self.a, self.b)

    def compute(self):
        return sum(self.transform(value) for value in
self.inputs())

class CubeSumComputer(SquareSumComputer):
    def transform(self, x):
        return x * x * x
```

The concept of Reusability shown by program:

```
class A
{ public:
  void DoSomethingALike() const {}
};

class B : public A
{ public:
  void DoSomethingBLike() const {}
};

void UseAnA(A const& some_A)
{
  some_A.DoSomethingALike();
}

void SomeFunc()
{
  B b;
  UseAnA(b); // b can be substituted for an A.
}
```

Conclusion

The concepts of OOPs are very much useful for programmers and also for other persons of the industry. The features are used in many different designing techniques. The person with the knowledge

of OOPs techniques can use it even in car designing and other designs. So we can conclude that this concept is multidisciplinary and only the concept of computer science. So more and more people are interested in the study of object oriented techniques and make their designs.

References

[1] Hoare, C. A. (Nov 1965). "Record Handling". ALGOL Bulletin (21): 39–69.doi:10.1145/1061032.1061041

[2] Kay, Alan. "The Early History of Smalltalk". Retrieved 13 September 2007.

[3] 1995 (June) Visual FoxPro 3.0, FoxPro evolves from a procedural language to an object-oriented language. Visual FoxPro 3.0 introduces a database container, seamless client/server capabilities, support for ActiveX technologies, and OLE Automation and null support. Summary of Fox releases

[4] FoxPro History web site: Foxprohistory.org

[5] 1995 Reviewers Guide to Visual FoxPro 3.0: DFpug.de

[6] Armstrong, The Quarks of Object-Oriented Development. In descending order of popularity, the "quarks" are: Inheritance, Object, Class, Encapsulation, Method, Message Passing, Polymorphism, Abstraction

[7] Pierce, Benjamin (2002). Types and Programming Languages. MIT Press. ISBN 0-262-16209-1., section 18.1 "What is Object-Oriented Programming?"

[8] John C. Mitchell, Concepts in programming languages, Cambridge University Press, 2003,ISBN 0-521-78098-5, p.278

[9] Michael Lee Scott, Programming language pragmatics, Edition 2, Morgan Kaufmann, 2006,ISBN 0-12-633951-1, p. 470 vikas