*Journal of Information Systems Education*
Volume 3, Number 1

# OBJECT-ORIENTED PROGRAMMING AND THE CIS CURRICULUM

**William P. Cain**
CIS Department - School Of Business
Manhattan College
Bronx, NY 10471

*ABSTRACT: Object-Oriented Programming (hereinafter referred to as OOP) is becoming very important in the data processing field. Computer science publications are beginning to contain many references to this topic. However, CIS journals contain very few OOP articles. This situation exists, in large measure, because COBOL, the primary language taught to CIS students, does not now support OOP. Even though CIS students can not write OOP programs as part of their courses, it is necessary that CIS students be exposed to OOP concepts. OOP techniques are being used in industry in graphics programming and in computer-aided software engineering. OOP technology also is used in the storage and retrieval of engineering drawings and in object-oriented databases. Besides being of importance in many application areas, OOP techniques are of value in that they demonstrate software engineering principles (e.g., information hiding and modularity) which CIS students will be expected to learn as part of their college curriculum.*

*This paper explains OOP concepts using terms and examples familiar to CIS students. A sample application is discussed without showing the actual programs to maintain language independence. This paper concludes with suggestions as to where OOP concepts can be introduced into the current CIS curriculum. Although OOP implementation details cannot be taught now because of the present level of the COBOL language, OOP can be presented using general terms and examples as is done in this paper. What is important is that CIS graduates, as computer professionals, need to know the underlying concepts and benefits of OOP technology.*

*KEYWORDS: Object-Oriented Programming, CIS Curriculum, Software Engineering*

## INTRODUCTION

Object-Oriented Programming (hereinafter referred to as OOP) is rapidly becoming one of the most frequently discussed topics in computer technology. The most popular languages supporting OOP today are Pascal and C++. Since these languages are not generally used in CIS courses, OOP does not find a natural niche in the CIS curriculum. The purpose of this paper is to explain OOP concepts using examples familiar to CIS students and to discuss how these concepts relate to the CIS curriculum.

As part of our research into OOP, we wrote several programs on the IBM PC using Borland's Turbo Pascal, Version 5.5 (1). This language product supports OOP techniques and is accompanied by manuals explaining Pascal and OOP concepts. The programs we wrote were all concerned with a school library application involving books and lending members. We will refer to these programs throughout this paper. However, because we are only interested in explaining OOP concepts and not in describing a specific Pascal implementation, we will not show the actual Pascal programs which we wrote.

## WHAT IS OOP?

OOP is a style of programming in which a system is broken into a collection of "objects" - an "object" is the representation of a person, place, or thing in the user's system to be designed. Faculty, buildings, telephones, and cities are all objects. In a system to handle orders for clothing items, the objects might be: items, customers, salespeople, orders. Objects have properties and objects can undergo certain operations. The telephone object has the properties of weight, color, and location - the telephone has the operators of making a call and receiving a call. A clothing item has the properties of item number, item name, selling price, and number on hand - an item has the operators of adding to inventory (for a shipment), withdrawing from inventory (for an order), and undergoing a price change. The collection of all telephones is called the telephone "object class" or "object type" - the individual telephone in my kitchen is a specific "instance" in the telephone class. An object is defined by the characteristics of the individual properties of the object as well as by the operators (called "methods") which are allowed to act upon the object.

The schematic representations of BOOK and LENDER, two objects in our sample library system, are shown in Figures 1 and 2.

In our library example we use Pascal statements to define the BOOK and LENDER object classes. Object properties are defined by declaring field names and characteristics. Procedural methods are coded using Pascal statements to list the parameters and write the program for each method. As an example, the LOAN_BOOK method is applied to an individual book and requires as a parameter the identification number of

**FIGURE 1: Schematic Representation of BOOK Object.**

Fields defined for each specific instance in the BOOK class:

| Name | Description |
|---|---|
| BOOKID | Book Identification Number |
| TITLE | Title of Book |
| AUTHOR | Author of Book |
| ONLOAN | "Y" If Book Is Currently On Loan - "N" If Book Is Not Currently On Loan |
| LENDERID | Identification Number of Lender Currently Having this Book |

Methods which can be used on a specific instance in the BOOK class:

| Name | Description |
|---|---|
| DISPLAY_BOOK | Display Title, Author, and Loan Data |
| ADD_NEW | Add a New Book to the Library |
| LOAN_BOOK | Put a Book On Loan |
| RETURN_BOOK | Allow a Book to be Returned from On Loan |
| DISP_CURR_LENDER | Display Name of Person to whom Book is On Loan |

**FIGURE 2: Schematic Representation of LENDER Object.**

Fields defined for each specific instance in the LENDER class:

| Name | Description |
|---|---|
| LENDERID | Lender Identification Number |
| LNAME | Lender Name |
| LADDRESS | Lender Address |

Methods which can be used on a specific instance in the LENDER class:

| Name | Description |
|---|---|
| DISPLAY_LENDER | Display Lender Name and Address |
| ADD_LENDER | Add a New Lender to the Library |

the lender of the book. The program for the LOAN_BOOK method consists of the following steps:

- Search the file of book records until the correct book is found.
- Read the book data fields into primary storage.
- Set the book ONLOAN field to "Y".
- Set the book LENDERID field to the id of the person borrowing the book.
- Rewrite the record for the book.

## HOW ARE OBJECTS USED IN A PROGRAM?

In order to use variables in the BOOK and LENDER classes, a program which makes use of methods is written. Instances of BOOK are defined by declaring one or more variables to be in the BOOK class. Instances of LENDER are defined by declaring one or more variables to be in the LENDER class. By declaring a variable to be in a certain object class, all properties (data fields) and methods declared for that object class can be used with that variable. It is important to note that only the methods declared for an object class can be used with a variable in that class - no OTHER methods are allowed. When we have a variable in an object class and we wish to apply a specific method to that variable, the OOP terminology says that we "send a message" to that method. In more traditional programming terminology, we simply "call" that method for the variable in that object class. As an example of how a program uses methods, one of the programs for our library application allows an individual lender to borrow a specific book. The program which invokes that method has the following steps:

- Ask the user at the terminal to enter the id of a lender.
- Send a message to DISPLAY_LENDER to list that lender's name and address.
- Ask the user at the terminal to enter the id of the book to be borrowed.
- Send a message to DISPLAY_BOOK to list that book's title and author.
- Send a message to LOAN_BOOK to put that book on loan to the indicated lender.

Note that the program as outlined above never explicitly accesses the specific data fields of a variable in an object class. The object's data fields are accessed ONLY through the methods invoked by the program.

## WHAT ARE APPLICATIONS FOR OOP?

A major reason for OOP's importance is that it finds extensive use in graphics applications. As an example, the software for the icon screen interfaces used by Apple's Macintosh computers, Microsoft's Windows, and IBM's Presentation Manager all use OOP techniques. Each kind of icon is declared to be in an object class. By defining specific methods for each object class, the systems software which deals with these icons manipulates them by sending appropriate messages to the correct methods.

As a simple example showing how OOP is used for graphics applications, the OOP manual written by Borland (1) uses objects of class LOCATION, POINT, CIRCLE, and ARC. Methods are defined allowing object variables to be initialized, to be made visible at a specific screen position, to be moved to a new position, and to be made invisible. Thus a relatively complex graphics application can be written by declaring variables to be in an object class (POINT, CIRCLE, etc.) and then causing shapes to be drawn, moved, and erased by sending messages to the methods which are defined for the object classes to which the variables belong.

OOP technology has grown well beyond being used only for graphics. Many articles in today's literature show widespread usage of OOP. Examples are:

- Martin (2) describes OOP as used in computer-aided engineering to manipulate objects which are complex shapes that interact with each other. He also discusses the use of OOP in computer-aided software engineering to handle objects which are graphical symbols representing software design components.
- Bochenski (3) describes a banking example using checking accounts and savings accounts as object classes. Similar methods such as "DEPOSIT" and "WITHDRAW" are defined for both object classes but unique methods such as "CALCULATE_INTEREST" are defined for the savings account object class only.
- Ten Dyke and Kunz (4) describe the use of OOP with IntelliCorp's Knowledge Engineering Environment, a system used to develop complex knowledge systems. The system can be used, for example, with objects that are financial instruments, such as shares of common stock and bonds. Some methods (e.g., buying and selling), are common to all objects while others (e.g., posting of dividends), are applicable to the stock object class, but not to the bond class.
- C.J. Date's database text (5) discusses recent research in which object-oriented database systems are used to solve problems which are inherent to relational databases. As an example, Date cites the use of object-oriented databases for large document storage and retrieval systems.

## WHY IS OOP IMPORTANT? WHAT ARE ITS ADVANTAGES?

OOP has its historical beginnings with computer graphics because of OOP's easy-to-use and natural techniques for manipulating shapes. However, the real

importance of OOP lies in its ability to help in the writing of more reliable and more easily understandable programs.

Software engineering is the study of ways to produce programs that are free from error, easily maintained, and understandable. Texts on this subject (for example, Software Engineering With Ada, by Booch (6) ) discuss how principles such as information hiding and modularity help the goals of software engineering to be met. Two of OOP's properties are encapsulation and inheritance. These help achieve information hiding and modularity - because of this, OOP is an important technology in aiding programmers to write code that is correct and understandable.

Encapsulation

By encapsulation we mean that the data fields and the methods for manipulating the fields are defined together. Data fields for a variable in the BOOK object class can be accessed only by sending a message to a method defined for the BOOK object. A BOOK's data fields cannot be changed without using BOOK methods. The BOOK methods are used by a calling program without that calling program needing to know either how the individual object fields are declared or what specific coding details are used in the methods.

This property of encapsulation allows data and method details to be changed without the necessity of changing the using program. Using our school library example, if new data fields (e.g., book category), are added to an object's structure, the calling program often is unaffected. If the details in the method to put a book on loan are changed (e.g., details in the algorithm to calculate the expected date of return), the using program operates exactly the same as it did before the changes - namely, a message is sent to the LOAN_BOOK method to put a specific book on loan. Encapsulation results in "information hiding" - the details of one part of a system are inaccessible to another part of the system when these details need not be known.

Encapsulation, besides eliminating many program changes, allows errors to be detected at the time of compilation rather than at execution time. Compared to run-time errors, compile-time errors are much easier to diagnose and correct. Using the school library example, we can have, in addition to the BOOK object, a REFERENCE object for those library materials which can not be removed from the library. Many of the data fields in the REFERENCE object (BOOKID, TITLE, AUTHOR) would be the same as data fields in the BOOK object. However, there is no method defined which allows us to put REFERENCE objects on loan. If a program mistakenly calls the LOAN_BOOK method for a variable in the REFERENCE object class, that incorrect call will be trapped and diagnosed at the time of compilation. In traditional programming methodology, this type of error would not be noted by the compiler and unpredictable run-time data errors would probably result.

***...OOP is an important technology in aiding programmers to write code that is correct and understandable.***

Inheritance

By inheritance we mean that an object class may be defined as a descendant of another object class. A descendant will have the same data fields and methods as the original object plus the descendant can have additional data fields and methods. A method that exists for the original object can be used without alteration by the descendant object or the same method may be programmed differently for the original and the descendant objects.

Again using our library example, LIBRARY_OBJECT may be the root object and that object can contain data fields for ID, TITLE, and AUTHOR. REFERENCE can be a direct descendant of LIBRARY_OBJECT and can have the same data fields as the root LIBRARY_OBJECT class. CIRCULATION_BOOK can be a descendant of REFERENCE and have the data fields of REFERENCE plus the ONLOAN and LENDERID fields. LIBRARY_OBJECT, REFERENCE, and CIRCULATION_BOOK can all share the same LIST_TITLE method to display the title of an object - LIST_TITLE in this case is defined at the LIBRARY_OBJECT level. ADD_NEW can be a method used by both REFERENCE and CIRCULATION_BOOK, but the programming of this method can differ for the two object classes. The CIRCULATION_BOOK object can have a LOAN_BOOK method which is not defined for the REFERENCE ancestor object.

Inheritance allows us to define hierarchies of objects without having to repeat declarations for data fields and methods at each level of the hierarchy. Inheritance also allows us to extend our system without major reprogramming. If our library starts to loan videos, a VIDEO object can be declared as a descendant of CIRCULATION_BOOK. VIDEO can have the data fields of CIRCULATION_BOOK plus additional fields (e.g., RUNNING_TIME). VIDEO can use the methods defined for CIRCULATION_BOOK (e.g., LOAN_BOOK) with no changes. In fact, a programmer can add the VIDEO object and use the CIRCULATION_BOOK methods for VIDEO variables without even knowing the details of the CIRCULATION_BOOK data fields and method definitions.

## WHAT ARE DIFFICULTIES WITH USING OOP?

OOP technology helps produce reliable programs and is naturally suited to certain classes of applications. Yet, there are several difficulties with OOP concepts. Some of these are:

- OOP is not easily learned by programmers trained in the "old style". OOP requires that the problem be modeled in terms of objects or entities while traditional programming places more emphasis on the actions to be performed.
- COBOL does not now support OOP. Standard COBOL has several features that make OOP implementation very difficult, if not impossible. Perhaps the most important restriction is that COBOL, unlike Pascal, does not allow users to define their own variable types. However, major vendors are working on OOP extensions to COBOL (like C++ for C) that should be available sometime in 1992 or 1993.
- In the school library programs written as part of this research, we found it difficult to keep track of all of our methods and programs. By its nature, OOP results in many methods being defined for each object class. Even with our small sample application, we became aware that proper use of OOP techniques requires that the benefits from writing small, reusable modules be balanced against the burdens of managing and documenting a large collection of such modules.
- In our sample application, we wanted the actual data for instances of our objects to be kept on external files. Our total design called for two root object classes, LIBRARY_OBJECT and LENDER. We wanted REFERENCE as a descendant of LIBRARY_OBJECT; CIRCULATION_BOOK and PERIODICAL as descendants of REFERENCE; and VIDEO as descendant of CIRCULATION_BOOK. We wanted one external file to contain the data values for all instances of class LIBRARY_OBJECT and of all its descendants (REFERENCE, CIRCULATION_BOOK, PERIODICAL, VIDEO). In similar manner, we wanted a second external file to contain data about the LENDER object and its descendants STUDENT, FACULTY, STAFF, and OUTSIDER. As we attempted to implement our design, we found that our version of Pascal did not allow us to set up and use our files in the straightforward manner we had envisioned - it was necessary to use software techniques that were not intuitive.

***COBOL does not now support OOP...However, major vendors are working on OOP extensions to COBOL (like C++ for C) that should be available sometime in 1992 or 1993.***

Files are a very important construct in business programming. In order for a programming concept to gain widespread acceptance by the CIS community, that concept must be shown to be able to operate with data files in an understandable and easy-to-use manner.

## CONCLUSION. EFFECT OF OOP ON THE CIS CURRICULUM.

### Why Should OOP Be Taught To CIS Students?

OOP is a widely discussed programming topic. Because of the importance of software engineering and graphics, OOP concepts will continue to grow in relevance. CIS students, as data processing professionals, should know what OOP is and should know the underlying benefits of OOP technology.

CIS students need to understand the principles of software engineering and software development. Industry is turning more and more to CASE (Computer-Aided Software Engineering) tools and successful use of these products requires an understanding of concepts such as modularization and encapsulation. In explaining software engineering techniques and CASE tools to CIS students, OOP is a very important topic because:

- OOP is a current programming technology that can be used to illustrate the meaning of various software engineering principles.
- The CASE tools now marketed present a graphical interface to users. This interface relies on OOP techniques both in terms of its use and in terms of how it is implemented by the vendor.
- Design tools now ask users to think of their systems in terms of objects as well as the more traditional entity-relationship model. (See, for example, Sagawa's description of the Repository Manager for the new IBM AD/ Cycle products (7) ).

### In What Courses Should OOP Be Taught To CIS Students?

OOP concepts can be introduced into the CIS curriculum without using explicit Pascal programs. A less detailed treatment, as is done in this paper, can be used to show OOP terminology and benefits. Several CIS courses offer opportunities to teach OOP:

- The software engineering course, if one is offered, provides a natural setting for the presentation of OOP concepts.
- The systems design and analysis course can introduce OOP, CASE tools, and related software engineering issues.
- The COBOL course can introduce OOP terminology while teaching related COBOL topics (e.g., variable types and subprograms).
- The database course can introduce OOP concepts. The research ideas discussed earlier can be presented

when considering the strengths and weaknesses of the relational model and when discussing database directions.

## SUMMARY

OOP technology has arrived. CIS students should be familiar with OOP because of its growing importance in a variety of data processing areas. This paper has explained the underlying principles using terms and examples familiar to CIS students. Although OOP cannot be neatly packaged into one CIS course, OOP examples and concepts can be naturally introduced throughout the CIS curriculum.

## REFERENCES/FURTHER READINGS

1. *Turbo Pascal*, Version 5.5. Scotts Valley, CA: Borland International, 1988.
2. James Martin. Series of four weekly articles on object-oriented programming, *PC Week*. Volume 6, Number 35, September 4, 1989, thru Volume 6, Number 38, September 25, 1989.
3. Barbara Bochenski. "Object-Oriented Cells Bring New Life To DBMS", *Software Magazine*. Volume 9, Number 8, June 1989, pp. 60-71.
4. R.P. Ten Dyke and J.C. Kunz. "Object-Oriented Programming", *IBM Systems Journal*. Volume 28, Number 3, 1989, pp. 465-478.
5. C.J. Date. *An Introduction To Database Systems*, Volume I, 5th ed., Chapter 25. Reading, MA: Addison-Wesley, 1990.
6. Grady Booch. *Software Engineering With Ada*, 2nd ed. Menlo Park, CA: Benjamin/Cummings, 1987.
7. J.M. Sagawa. "Repository Manager Technology", *IBM Systems Journal*. Volume 29, Number 2, 1990, pp. 209-227.

## *AUTHOR'S BIOGRAPHY*

*William P. Cain has taught at Manhattan College in New York City in the Computer Information Systems department since September, 1987. His primary area of research interest is the use and programming of database applications in a business environment. Before joining Manhattan, he worked in industry for twenty-seven years as a market forecaster, a manager, a computer programmer, and an educator.*


Information Systems & Computing
Academic Professionals


## STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.



ISSN 1055-3096