Journal of Yaşar University
http://joy.yasar.edu.tr

TEACHING OBJECT ORIENTED PROGRAMMING AT THE INTRODUCTORY LEVEL

Mehmet C. OKUR*

ABSTRACT

Teaching object oriented programming has become a rapidly expanding preference at various educational environments. However, teachers usually experience problems when introducing object oriented concepts and programming to beginners. How to teach the fundamentals of object oriented programming at an introductory level course is still a common subject for debate. In this paper, an evaluation of these problems is presented and some possible approaches for improving the quality and success of such courses are discussed.

Key Words : Object oriented language , procedural programming, abstraction, UML, algorithm, programming environment.

1. INTRODUCTION

Object oriented programming has become the dominant programming style in both the software industry and education over the last ten years or so. Software companies have released the object oriented versions of their products and educational institutions at every level have included object oriented programming in their curricula. Most secondary school and university courses on programming have moved from Pascal or C to object oriented languages such as C++, Smalltalk, Eiffel and lately Java and C#. The main reasons for this tendency are related to having advantages like; abstraction , inheritanca ,polymorphism and encapsulation. On the other hand, object oriented methods scale up very well with the growing sizes of real life software projects and they facilitate the creation of complex software products. It is now widely accepted that the object oriented style offers better tools for teaching and software development.

* Department of Computer Engineering Yasar University, Izmir, TÜRKİYE mehmet.okur@yasar.edu.tr

Switching from the older imperative procedural programming style to object oriented style, however, still remains to be a difficult task. Because, object orientation imposes a paradigm shift for the educators and in most cases, for the students. For most teachers and students, procedural style programming constitutes the first experience. It is a known fact that, learning object oriented programming is more difficult after getting used to procedural programming. According to observations, a considerably long period is needed for an average student to switch his or her mind from a procedural to an object oriented approach (Stroustrup,1994).Namely, in the transition period, an "unlearn" process takes place. This phenomenon suggests that, if we want to teach object oriented programming, we should start it as early as possible. Alongside with many others, our experience shows that starting with procedural style complicates teaching and learning object oriented programming at later stages.

Teaching object oriented programming ,on the other hand, is not an easy task. There are no pedagogically proven methodologies to guide the teachers and institutions. Educators in this area are still experimenting various styles and approaches to identify better and more effective ways of introducing object oriented concepts and techniques. Some important questions of interest that require decisions are:

- Which object oriented language should be used?
- What should be the teaching order of object oriented concepts?
- What should be the teaching environment?
- When and how deep the modeling and object oriented design elements should be included?

The choices, regarding these and similar questions, affect to a great extent the structure and success of a course on object oriented programming. It is sometimes argued that, object oriented technologies are too complex and full of difficulties for the earlier stages of programming education. This is partly true but experience from numerous courses demonstrates that, choosing effective educational pedagogies and careful design of the courses help overcome these difficulties. Object oriented software technologies are essentially based on the modeling of real life objects, It is therefore natural to place the description and identification of objects at the start of a course on object oriented programming .So if we want to teach object oriented programming, objects and their interactions should always be at the focal point. Modeling objects and their relationships, on the other hand, can best be realized

by including a design element into the course right from the beginning. However, modeling, design and implementation of objects should be integrated into the course in a coherent manner. Modern object oriented languages and teaching aids provide effective techniques and tools that facilitate such an integration.

2. DIFFICULTIES IN MOVING FROM PROCEDURAL TO OBJECT ORIENTED PROGRAMMING

Traditional imperative programming languages are mostly based on the von Neumann architecture. Their pedagogy is built upon sequence, selection and iteration. As a result, teaching popular languages such as FORTRAN, Pascal and C is governed by the design and implementation of common algorithms and data structures. This was best expressed by the famous quotation "Algorithms+ Data Structures=Programs." Here the main focus is to teach students programming through the implementation of algorithmic methods. This educational model necessitates teaching the syntax and basic structures of the language as early as possible. In this process, the peculiarities and syntactical details of the language dominate, even the earlier stages of teaching. All these tendencies are contradictory and even harmful in case of object oriented programming, where the main focus is problem solving, rather than language details. In order to escape these adverse effects, object oriented programming is recommended to be the first choice for teaching and learning programming (Kölling,1999).

An important negative carry-over effect of traditional programming is caused by the emphasis on the syntactic and structural details of the language. As a result , most teachers and students tend to spend their times on unnecessary details rather than concentrating on real life projects and applications. This is a fairly common observation for especially introductory level courses. Object oriented programming education is no exception. In most cases, the syntax and algorithmic elements of the object oriented language could shadow more important characteristics of the object oriented paradigm. A traceable reason is the use C++ as the teaching language for object oriented programming. It is a well known fact that C++ is a "hybrid" language and sometimes classified as an "object oriented C". Namely, C++ is based on and compatible with the procedural C language. Due to this proximity,changing the teaching and programming habits become very difficult if one tries to replace C by C++. So, the better approach is to start teaching object oriented programming with a so called "pure" object oriented language.

3.THE IMPORTANCE OF ABSTRACTION AND DESIGN IN OBJECT ORIENTED PROGRAMMING

Object oriented software products are based on objects and their relationships. In some sense, object oriented concepts have replaced the procedural structure of traditional programming languages. Therefore, object oriented programs should not be slightly different versions of programs in Pascal or C .The main framework should be problem solving through proper modeling and design of objects and classes. So, a "Hello World!" type program is definitely a bad start .Because the syntax is being put before the problem solving dimension of programming. A better and mostly recommended approach is to use abstraction and design right from the beginning Nguyen and Wong,2001). Although abstract thinking may be considered to be somehow difficult for many students to learn, it has now become an important component in object oriented programming education. Abstraction on the other hand, requires the modeling and design components to be introduced in even introductory level courses.

Teaching the fundamentals of object oriented programming at introductory level remains to be a serious pedagogical challenge. Even the most common concepts are inherently advanced and it is difficult to form complete programs without using them. Take for example the minimal signature of the main method, which is needed for every Java application or simple print statements which require reference to System Class.These concepts and frameworks could only be grasped after considerable experience in the Java language. So, how to overcome this kind of difficulties and make object oriented programming more manageable for the beginner?

Object oriented programming require some design effort before the actual coding starts. Because, finding out the objects and deciding on the task allocation among them is not a programming job. The work at this stage should be concentrated on finding answers to questions like:

- What will the program achieve when it is completed?
- What classes and objects will be needed?
- What will be responsibilities of the classes and objects?
- What will be the information content and functionalities of each object?

- How will the objects handle their responsibilities?
- How will the objects communicate?

A good and well received approach in this respect is “Objects first with design patterns” (Dock and Steegmans,2002) . The main ingredients of this approach are object identification, object modeling and object design. Coding decisions should be based on the realizations of this stage. Object oriented modeling and design can be facilitated by using simple graphical integrated graphical aids. A subset of the Unified Modeling Language (UML), or even a simple notation that uses boxes and arrows could be of great assistance. In most courses, the use of a home-made diagrammatic notation greatly facilitates apprehension of object-class-instance relationships. Emphasizing modeling and design right from the beginning and coupling code with design are now well established preferences in object oriented programming education (Ventura et.al.,2001).

4. CHOOSING THE OBJECT ORIENTED TEACHING LANGUAGE

Nowadays C++ and Java are two popular languages for teaching object oriented programming,C++ is mostly preferred in Computer Science and Computer Engineering curricula, because of its power as a programming language, backward compatibility with C and relative ease of implementing classical algorithms and data structures. However, C++ lacks many characteristics that should be possessed by a language for teaching object oriented programming. Among the problem areas are the “hybrid” nature of the language, unsafe typing, highly complex object model and lack of readability(Kölling 1999) classifies C++ to be one of the worst candidates as a teaching language for object oriented programming,
Java , on the other hand, is considered to be a “pure” object oriented language and does not exhibit many important disadvantages of C++. It is also by far the most common language for Internet environments and many network based programming tasks. These and other advantages ,such as platform independence and relative simplicity promote Java to be the natural choice for a teaching language at introductory levels. The market success of Java is also another positive factor to its favor (Kölling, 1999) concludes that, among various alternatives, Java appears to be a better choice to start object oriented programming.

5 . PROGRAMMING ENVIRONMENTS

A suitable programming environment is important for the success of an introductory course on object oriented programming. Traditional environments for programming include an editor and the DOS or UNIX operating system. These are sufficient for simple application development. Many commercial development systems are also available for common programming languages. The use of integrated graphical development systems in teaching is becoming more and more common. However these systems are essentially professional development environments and their suitability as teaching aides in an introductory course are questionable. Simply they are far more powerful and sophisticated than needed and their advanced features make them less convenient for such a course. This kind of environments are also criticized for their inability to reflect the object oriented concepts.

As a compromise, environments that are specifically designed for object oriented language teaching have been developed .A popular example in this respect is the BLUE language.The C++ and Java versions of the BLUE provides a very simple and user friendly environment for creating and manipulating objects and methods. The system encourages the student to think a program in terms of objects, classes and their interactions. The Java version of the system, BLUEJ is now available as the accompanying component of a textbook (Barnes and Kölling 2005). Depending on the next step in object oriented programming education, this kind of Integrated environment can be chosen as the basic teaching aid.

6. OBJECT ORIENTED COURSE CONTENTS

Another critical success factor in teaching object oriented programming is the choice and order of the topics that should be presented. The fact that, even the simplest object oriented programs include fairly advanced concepts makes this choice more important. As we have emphasized, the aim is not only teaching a programming language, but to integrate a considerable amount of abstraction and design element into the course. To some extent, object oriented programming is about building and implementing abstractions. Therefore the main goal should be to teach program structuring and design through objects and classes. The algorithmic details and data structures should all be based on the modeling and design work. In structuring an introductory course, we should keep in mind that object oriented paradigm incorporates three concepts : object, class and inheritance. The course contents should reflect this incorporation. A possible priority list of topics could be as follows:

- Object
- Class, subclass and superclass
- Data and variables
- Methods
- Constructors and method overloading
- Encapsulation and visibility modifiers
- Object interactions
- Inheritance and software reuse

As indicated in the list, objects and classes should be introduced from the very beginning of the course. Starting with very simple real life objects and classes, major modeling and design steps of software development cycle should be followed. The fundamental stages of this cycle constitute a natural framework for this purpose. Easily recognized objects such as person, student, car, house, school, bank, traffic light etc. could be chosen for illustration. From a design point of view, an object should be considered as a software element that contributes to the solution of the problem at hand. After introducing the fundamentals of object modeling and design, formal class definitions and instantiations can be implemented using a minimal amount of code detail. Basic elements of the language could be introduced after this point. Main topics that should be covered include data values and types, variables expressions, method invocations and parameter passing.

Next the relationships among the objects and classes should be covered by emphasizing the use of libraries, packages, reference variables and various data structures. Full extent of Inheritance and related concepts of software reuse and streams can not be covered in an introductory course So, the priority should be to make a sound introduction to the fundamental concepts and leave more advanced concepts to a higher level course. It should be stressed again that, at an introductory level course, the details of the language should not dominate teaching so as to obscure the fundamental object oriented concepts.

7 .CONCLUDING GUIDELINES

The following is a set of of concluding remarks based on experiences:

- Use a pure object oriented language in teaching.
- Start teaching by introducing objects and classes first.
- Demonstrate the basic concepts by referring to simple real life objects.
- Require modeling and design as fundamental components at every stage. Use a simple graphical notation.
- Do not start coding without a proper class and object design.
- Follow a simple problem-design-implementation pattern consistently.
- Develop the solution to a new problem by including as many elements as possible from a previous solution..
- Introduce, object associations and inheritance as early as possible but to a manageable extent.
- Use simple algorithms and data structures that do not require advanced features of the language.

REFERENCES

Barnes, D.J,, Kölling, M.(2005) : Objects First with Java. A Practical Introduction Using BLUEJ. Prentice Hall.

Docks, van D., Steegmans, E.(2002):A new pedagogy for Programming. Sixth Workshop on Pedagogies and Tools for Learning Object Oriented Concepts, ECOOP,Malaga Spain.

Kölling, M. (1999). The problem of teaching Object-Oriented Programming, Part 1: Languages.Journal of Object-OrientedProgramming, 1(8): 8-15.

Kölling M, (1999).:The Problem of teaching Object-Oriented Programming, Part 2:Environments.Journal of Object-OrientedProgramming, 11(9): 6-12.

Nguyen, D., Wong, S(2001):OOP in Introductory CS: Better Students through Abstraction. Fifth Workshop on Pedagogies and Tools for Assimilating Object Oriented Concepts. OOPSLA,Umea,Sweden.

Nino J., Hosch, (2001): Programming and Object Oriented Design Using Java. Wiley,

Stroustrup, B.(1994.): The Design and Evolution of C++.Addison-Wesley.

Ventura, P.R., et. Al.(2001) : Teaching OOD and OOP through Java and UML in CS1 and CS2.OOPSLA,Umea,Sweden.