THE BASICS OF TEACHING OF OBJECT-ORIENTED PROGRAMMING

Aleksandar Kupusinac[1], PhD;

[1] University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia, sasak@uns.ac.rs

Abstract: *The one of the major paradigms in the software industry is object-oriented (OO). Due to the great expansion and rapid development of the software industry, properly learning of fundamental terms meanings, such as object and class, has an important role in programmer's education. The challenge for the teachers of OO programming is to teach the application of mathematical and logical concepts, algorithmic thinking, and good programming practice. This paper considers the way of teaching fundamentals of OO programming by using conceptual definitions of the object and class.*

Keywords: *Object-oriented programming, Teaching, Programming paradigm.*

1. INTRODUCTION

Information technologies have great popularity in the world and there is a growing need for highly qualified and professionally educated programmers. The learning of object-oriented (OO) paradigm has an important role, since it is one of the major paradigms in the software industry. The idea of object programming was created at the end of the 1950s and early 1960s at the MIT for the needs of artificial intelligence, with the goal of describing the real world in a formal way. Officially, the first OO programming language Simula 67 was made in the 1960s at the Norwegian Computing Center in Oslo [1]. Simula 67 introduced classes and objects, and influenced to making popular OO languages such as C++ and Java. The first OO programming language that strongly influenced software industry in the 1980s was C++ [2].

The study of a scientific and professional area starts with the introduction of basic concepts and terms, that is, by studying their definitions. Unfortunately, object oriented programming has evolved in an evolutionary way, in response to the increasing demands of the software market, on the one hand, on the other hand, due to the very protraction of science and the way of thinking. Therefore, during the construction of theory and practice, a factual state is established in which there are different views on the fundamental concepts of object methodology - object and class. This paper considers basics of the teaching OO programming by using conceptual definitions of the class and object.

The paper is structured as follows. The Section 2 describes the terminology in OO paradigm and the current problems. The Section 3 considers the way of OO paradigm teaching. The paper ends with conclusions.

2. TERMINOLOGY IN OBJECT-ORIENTED PARADIGM

Understanding terminology of OO paradigm is a problem, since often there are several terms for the same thing (synonyms) and the unreasonable use of terms that are taken from other scientific fields with wrong association [3]. A typical example is the term *object,* which usually means a thing or a correlate to a subject [4]. There are several different **object** definitions [5]:

- The object represents an individual, a identifiable detail, a unit or entity, real or abstract, with a well-defined role in the issue of the problem [6],
- The object was anything with sharp and clearly defined boundaries [7],
- The object is a thing that can be identified and that plays a role in relation to the request for executing operations [8],
- The object is a collection of operations that share the state [8],
- The object is an encapsulation of a set of operations that can be externally aroused and states that remember the effects of applying the methods [8],
- The object is an instance of a class at run time [9],
- The object is an entity model that has identity, state, and behavior, where the entity implies the existence of something [8] [10].

and several **class** definitions [8]:

- Class serves for objects grouping,
- Class has the nature of a schema or template that define the structure and behavior of all objects belonging to it,
- Class serves as a generator of objects.

In the OO program development, the analysis and modeling have basic role. Programming is a thought procedure, where programmers deal with thoughts about participants in the information system, i.e. the observation unit is treated through thoughts about it, or more precisely, through thoughts about its essential characteristics. The thought about the essential characteristics of a thing is concept [11]. For example, designers of the information system of the bank do not deal with real clients, instead they use the thoughts about them, i.e. thoughts about essential characteristic or concept.

The previous definitions have a free form, and a disadvantage that the object and class cannot be defined independently of each other. The better solution is a definition that object is an entity model, but it raises the question of what the terms entity and model are and how to use them, since they are already present in terminology of philosophy. In other words, this definition has one obvious problem, the programmers do not work with concrete observation units (entities), but with thoughts about them. The best solution are conceptual definitions of a class and object [12], which allow the objects and classes to be defined independently of each other and give a more realistic insight of the modeling process based on essential characteristics of the observation units:

- A class of objects (a shorter class) is a software model of a class concept.
- The object is a software model of an individual concept.

The conceptual definitions have two great advantages - they are based on concepts with well-known meanings, and class and object are equal in a semantic sense, i.e. class is not defined over the object nor vice versa. Individual concepts with common characteristics belong to the same class, and the thought of a given class is class concept. For example, the individual concepts DOSTOEVSKY and TOLSTOY are thoughts about writers that really lived in 19. century in Russia. These individual concepts belong to class concept RUSSIAN WRITERS. The class concept RUSSIAN WRITERS belongs to more general class concept WRITERS, and on such a way we obtain a class concept hierarchy, that is a basis for inheritance OO paradigm. The individual concepts may and may not be real, while class concepts are not real. For example, the class concept RUSSIAN WRITERS is not real, while the individual concept

DOSTOEVSKY is real (Russian writer Fyodor Mikhailovich Dostoevsky existed in the period 1821-1881). However, both the individual and class concepts SQUARE are not real.

Each observation unit in its widest sense possesses certain characteristics that we can divide into essential and irrelevant. The thought of any characteristic is called a tag. The thoughts of relevant characteristics for a specific domain are called the relevant tags. The procedure for selecting the final number of relevant tags is called software modeling, which produces a software model with final set of relevant tags. In the software model, descriptive tags are called fields, while operational tags are called methods. For example, the class term CAR has descriptive tags - color, mass, etc., but also operational tags - the ability to move, the ability to shift gears, etc.

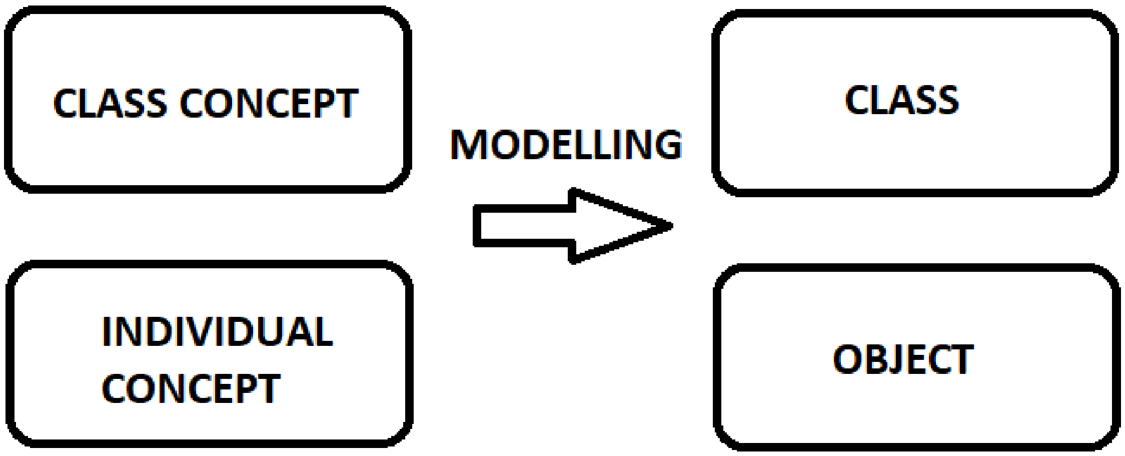

Figure 1. Modelling process.

We can observe that the modeling process is the way to obtain a simplified picture of a concept in a given domain of the problem, whereby this procedure is not significantly determined, i.e. the same concept can be modeled in many different ways. For example, the concept CITIZEN can be modeled with the final tag set {name, surname, address, phone number}, but it could also be modeled by the final tag set {name, surname, passport number}. The class designer decides how to model some concept, with the recommendation that model must be simple, but also enough descriptive.

The relationships between class concepts will be modeled as relationship between classes. The case when more complex class concept contains another class concept (fragment) will be modeled as an owner-component relationship. For example, the class concept TRIANGLE has three fragments VERTICES. The case when a class concept is derived from another, as a specialization, will be modeled as inheritance relationship. For example, the class concept CHIEF is derived from class concept EMPLOYEE.

3. INTRODUCING OO PARADIGM

The teacher as a responsible person in the teaching process is a coordinator, mentor and motivator whose duty is to provide a comfortable working environment [13]. There are high demands on teachers, because of popularity of information technologies and a growing need for highly qualified and professionally educated programmers. The teacher plays a role in selecting valuable information in knowledge transferring, with constant monitoring society changes as well as the student interest. Programming teaching should cover good programming practice, algorithmic thinking and applying mathematical and logical concepts [13].

The natural way to learn programming is to start with the procedural paradigm because it is closest to the beginners perception about how the computer performs an algorithm. The procedural paradigm is close to the human way of thinking, because it implies direct implementation of the algorithm by using collection of variables, data structures, and subroutines. While records, modules, procedures and procedure calls represent mechanisms of procedural paradigm, object-oriented paradigm uses objects, classes, methods and messages, respectively.

Array sorting can be used as a typical introductory example. In procedural paradigm the sorting array can be written in a pseudocode:

```
int i, j, temp;
for i:=0 to n-2 do
      for j:=i+1 to n-1 do
          if a[i]>a[j] then
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
          end;
      end;
end;
```

, where we assume that a given array `a` has `n` elements. In OO paradigm, the same problem will be solved in the following way:

```
objArray.sort();
```

, where `objArray` is an object of the class `Array`, that containes a method `sort()`. The method `sort()` implements a sorting algorithm.

The class concept STUDENT is formed in the our mind based on many individual concepts, i.e. the thoughts of each concrete student that we met in the past. The class Student is obtained by software modeling of the class concept STUDENT, depending on the problem that we solve, e.g. class Student in the case of student administration office will contain the following fields:

```
class Student {
      String name;
      String surname;
      String indexNumber;
      int passedExamsNumber;
      double averageMark;
      …
};
```

, while in the case of student ambulance it will contain:

```
class Student {
      String name;
      String surname;
      String indexNumber;
      double height;
      double weight;
      …
};
```

The objects of the class `Student` contain components `name` and `surname`, that are objects of the class `String`. This example illustrates owner-component relationship. Class `PhDStudent` can be obtained as a specialization of the class `Student`, with additional field `graduationExamData`. This example illustrates inheritance relationship:

```
class PhDStudent extends Student {
      String graduationExamData;
      …
};
```

4. CONCLUSION

This paper presented the way of using conceptual definitions of the object and class in the OO programming teaching. There is a growing need for highly qualified and professionally educated programmers and the learning of OO paradigm has an important role, since it is one of the major paradigms in the software industry.

ACKNOWLEDGEMENTS

This work was partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia within the projects: ON 174026 and III 044006.

REFERENCES

[1] J. R. Holmevik, "Compiling Simula: A historical study of technological genesis", IEEE Annals of the History of Computing, 16 (4): 25–37, 1994. doi:10.1109/85.329756.

[2] B. Stroustrup, "The C++ Programming Language" (1st ed.), Addison–Wesley, 1985.

[3] D. Malbaški, „Objektno orijentisano programiranje kroz programski jezik C++", Novi Sad: Fakultet tehničkih nauka, 2008.

[4] M. Marković, „Filozofski osnovi nauke", Beograd: Prosveta, 1994.

[5] D. Malbaški i D. Obradović, „On Some Basic Concepts in Object Orientation", 6th Balcan Conference on Operational Research, Thessaloniki, 2002.

[6] M. Smith i S. Tockey, „An Integrated Approach to Software Requirements Definition Using Objects", Scattle WA: Boeing Commercial Airplane Suport Division, 1988.

[7] B. Cox, „Object Oriented Programming: An Evolutionary Approach", Reading MA, Addison-Wesley, 1986.

[8] E. Ellmer, „Object-Orientation–an Overwiev", University of Viena, www.ifs.univie.ac.at/ISOO/overview.html .

[9] B. Meyer, „Object-Oriented Software Construction", Prentice Hall, 1988.

[10] D. Malbaški, „Objekti i objektno programiranje kroz programske jezike C++ i Pascal", Novi Sad: Fakultet tehničkih nauka, 2006.

[11] G. Petrović, „Logika", Zagreb: Školska knjiga, 1981.

[12] D. Malbaški i A. Kupusinac, Konceptualna definicija objekta i klase. International Symposium on Industrial Electronics (INDEL) 8; Banja Luka; 2010.

[13] O. Hrnjaković, M. Tot, A. Kupusinac i R. Doroslovački, „Različiti pristupi u metodici nastave osnova programiranja u srednjoj školi i na fakultetu", XXV Skup TRENDOVI RAZVOJA: "KVALITET VISOKOG OBRAZOVANJA", Kopaonik 11-14.02.2019.