Object-Oriented Programming in Computer Science

Rahime Yilmaz
Istanbul University, Turkey

Anil Sezgin
Yildiz Technical University, Turkey

Sefer Kurnaz
Istanbul Esenyurt University, Turkey

Yunus Ziya Arslan
Istanbul University, Turkey

INTRODUCTION

In computer science, a program is composed of a series of commands, which runs within a computer or an electronic circuit, producing information for users. Programming is a that can help programmers while writing a program. Computer programming is the process of writing an algorithm and, it is also the encoding of the algorithm into a notation that can produce and provide information to the users. It can be classified into two groups, that is, system programming and application programming. System programming is a sub branch of the general programming that is composed of low level instructions which are used to operate and handle computer hardware. Application programming is considered as the improved version of the computer programs which can perform specific tasks for the users. One of the application programming types is the object oriented programming (OOP) which is about how information is represented in human mind.

As a computer programming approach, OOP is useful such that it provides easy modeling in designing and developing real entities. This approach is intended to model the entities and, also, the relationships existing between them. OOP allows programmers to define the required classes to create the objects and to apply modifications (manipulations) on them. It can also supply inheritance, polymorphism and encapsulation features to the developers. With these capabilities, the processed data can be isolated from other redundant applications. Because of its abilities that are readily available to the users, OOP is preferred much more than other available programming languages. The inherent properties of OOP, which do not exist in other application programming, can be stated as modularity, extensibility and reusability. This chapter provides a substantial survey of OOP in computer science.

In this chapter, we have highlighted a number of explanations and reviews that are generally accepted and are in common use in OOP. We explain the heart of this chapter OOP concept in section 1, Object Oriented Programming Features, making up the largest section. Main topic of OOP which are included Inheritance, Polymorphism, Abstraction and Encapsulation titles are explained with details in the subtitles of section 1. In section 2, you can find an example of OOP implementation in Java. There are many kinds of OOP languages in use but in this study, Java was given as a strong example to OOP language (Harel, Marron, & Weiss, 2010). The following section is Future Research Directions which include future works related OOP. The

DOI: 10.4018/978-1-5225-2255-3.ch650

Copyright © 2018, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

chapter ends with Conclusion section that gives a brief information about this study.

BACKGROUND

Programming paradigm is a fundamental style of computer programming which classifies programming languages. Different programming paradigms were developed by considering the concepts and abstraction which are used to represent the elements of a program, and steps that compose a computation. Some of the programming languages are designed to support one paradigm and some of them support multiple paradigms. Before OOP languages, there are also some other paradigms such as classic programming, modular programming and structural programming (Bista, Bajracharya, & Dongol, 2015). These programming technics were helped the programmers while solving their problems. Depending on improving technology, new structure of OOP has emerged so one of these paradigms is OOP which changed radically the programming paradigm continued until the day it appeared. Software methodology used before OOP referred to by the name of the procedural programming. This methodology was based on advancing codes in a particular direction and calling the common function that is used to reduce the workload. This methodology, used in the software world for a long time, has some difficulties. First of all, application of the procedural programming developed as a whole cannot be divided. So, each developer working on the application has to know almost every building of application. Due to its building as a whole, it is hard to make changes on the application. The reason of these difficulties is that procedural programming is an abstract that is not able to model the real world. The real world can be simulated by programmers thanks to OOP by using objects and classes. Object-oriented approach enables to divide a complex system into smaller parts and manageable modules which makes development process easier to grasp and share among members of a developer team and easier to communicate to users who are needed to provide requirements and confirm how well the system meets the requirements throughout the process (Dennis, Wixom, & Tegarden, 2015). Thus, OOP enables modularity and abstraction with increased code understanding, maintenance and expansion. OOP is formed by the collection of objects which communicate with each other in order to perform tasks. This communication is based on messages in OOP, and objects are created from classes.

Class is a blueprint which is used to define objects describing the contents of the objects itself. It is a user-defined prototype for an object that defines a set of attributes and methods which characterize any object of the class. These attributes are data members or variables (static attributes), and methods are dynamic behaviors, also called member functions. Data members and member functions are called class members. Attributes, which are attached to the classes, store information about the object (Robson, 1981). Data member has a name and a type; it holds a value of that type. Member function receives parameters from the caller (if it's required), performs the tasks which are defined in the function body and returns result or void to the caller.

In most of the programming languages, class keyword is used to define a class. Class declaration must contain the name of the class which programmer declares. Basic class declaration looks like this:

```
Class NameofClass {
...
}
```

The other term, object, helps users to understand the object oriented notion. Objects, also called instances of a class, are modeled on real world entities. All the instances of a class have similar properties. Basically, objects have 2 characteristics, state and behavior. State is a well-defined condition of an item which captures the relevant aspects of an object. Behavior is the observable

9 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/object-oriented-programming-in-computer-science/184444

Related Content

Culture and Internet Banking Technology: Long-Term Orientation Over the Acceptance
Leelien Ken Huang (2019). *Handbook of Research on the Evolution of IT and the Rise of E-Society (pp. 239-259).*
www.irma-international.org/chapter/culture-and-internet-banking-technology/211618

The Past, Present, and Future of UML
Rebecca Plattand Nik Thompson (2018). *Encyclopedia of Information Science and Technology, Fourth Edition (pp. 7481-7487).*
www.irma-international.org/chapter/the-past-present-and-future-of-uml/184445

Chaotic Map for Securing Digital Content: A Progressive Visual Cryptography Approach
Dhiraj Pandeyand U. S. Rawat (2016). *International Journal of Rough Sets and Data Analysis (pp. 20-35).*
www.irma-international.org/article/chaotic-map-for-securing-digital-content/144704

A Model Based on Data Envelopment Analysis for the Measurement of Productivity in the Software Factory
Pedro Castañedaand David Mauricio (2020). *International Journal of Information Technologies and Systems Approach (pp. 1-26).*
www.irma-international.org/article/a-model-based-on-data-envelopment-analysis-for-the-measurement-of-productivity-in-the-software-factory/252826

Teacher Presence
Caroline M. Crawford (2018). *Encyclopedia of Information Science and Technology, Fourth Edition (pp. 7922-7934).*
www.irma-international.org/chapter/teacher-presence/184488