

Comparison Object Oriented Programming in Objective-C with GO

Technische
Hochschule
Rosenheim
Technical University of Applied Sciences



Thomas Martin Randl

Fakultät für Informatik

WS 2020/21

Abstract

ToDo

Keywords: Go, Objective-C, Object Oriented Programming

Table of Contents

Abstract	1
List of Abbreviations	3
1. Introduction	4
2. History of OOP	4
3. Object Oriented Programming	4
3.1 Classification	5
3.2 Encapsulation	5
3.3 Inheritance	5
3.4 Polymorphism	6
3.5 Persistence	6
4. Object Oriented Programming in Objective-C	6
5. Object Oriented Programming in Go	6
6. Differences between Go and Objective-C regarding OOP	6
6. Summary	7
References	8
List of Figures	9

List of Abbreviations

Go Programming Language Go

OOP Object Oriented Programming

1. Introduction

2. History of OOP

The first approach for Object Oriented Programming (OOP) was more than 40 years ago with the programming language Simula 67. However the first language to be really successful with this new approach in programming was Smalltalk. These languages fulfilled the basics needed for OOP. With a slow and steady development OOP became a more and more used concept in new developed programming languages [\[KOP09\]](#).

Since the teaching of programming languages like Java took place in a wide range of university classes, OOP became more and more relevant to companies. This lies in the fact that graduates from all over the world tend to know how to use at least one OOP language. The fact that more and more companies made use of OOP languages in their software solutions pushed their popularity even further.

Nowadays OOP is the most spread concept between all programming languages. Besides their teaching all over the globe their popularity can be derived from the fact that problems from the real world can easily be transferred into the programming world. It is simple to imagine a button in a graphical interface to be an object with certain properties. Even non OOP languages nowadays tend to use object similar approaches as you can see with the programming language Go in [Chapter 5. Object Oriented Programming in Go](#) [\[KOP09\]](#). Sometimes this works out well, but the OOP approach turns out to be the better way in most of this cases.

3. Object Oriented Programming

As we learned in [chapter 2. History of OOP](#), OOP is a widely spread concept in modern programming languages. It takes proceedings from the real world into a programming context. This is because of the fact that you can declare every instance from the real world as an object. These objects on the other hand can be put in context and interact with each other. These traits of OOP are called classification and inheritance. With that in mind every object needs his own class with its own methods to interact with each other [\[KOP09\]](#). In the following chapter we will learn how a class in OOP is structured and what base concepts OOP uses to set classes in context with each other and how they are visible outside of the object. The extend to which a programming language uses these concepts defines how much of an OOP language it is.

3.1 Classification

Classification is a technique used to classify objects due to their skills. During the process objects are structured (chapter [3.3 Inheritance](#)) and the skills are assigned to the objects class.

A class describes and implements a new type of object. Objects in OOP are derived from that class. Every class consists of several attributes and methods. An object kann be created using the class as kind of a datatype with parameters. The added parameters are assigned to the attributes in the constructor of the class. The created object interacts independently from other objects with the environment using the methods provided by the base class.

The methods of a class are declared in an interface. This can be used by other classes to interact with the classes objects using these methods [\[KOP09\]](#).

3.2 Encapsulation

Encapsulation is used for hiding the objects information from the environent. It is uesed to protect the attributes values of the object from manipulation. This is done by declaring the attributes visibility. If an attribute is declared as not visible from outside the object it can only be accessed by using methods declared in the inteface of the objects class. Methods can also be declared in different visibility states depending on the programming language.

An encapsulated object can only be interacted with via the methods declared in the interface. Every attribute or method not mentioned in the interface is neither accessible nor visible from outside [\[KOP09\]](#).

3.3 Inheritance

As in chapter [3.1 Classification](#) mentioned, Objects in OOP are structured by their skills and abilities. In OOP every Object is derived from a parent object. The resulting child object is able to use the functionality of the parent and its own. This property is called inheritance.

With the use of inheritance it is possible to reuse the previous defined structures or to make use of an abstract object which can be specified in the child object. This helps the programmer to reduce the complexitiy and effort of the programm.

Especially the use of an abstract class can be very useful. This is the case if a program uses different object that contain several equal attributes. In this case the abstract class defines the base attributes and the child objects inherit them and implement their functionality for them [\[KOP09\]](#).

3.4 Polymorphism

A synonym for polymorphism is diversity. Taking this into the context of programming languages, polymorphism is the approach to accept and return values of more than one datatype.

OOP uses this functionality with its inheritance (chapter [3.3 Inheritance](#)). This allows the use of different objects as parameters and/or return values.

3.5 Persistence

Persistence stands for the lifetime an object exists in the program after it is created. There are some different approaches depending on the used programming language. While in C++ the user is responsible for deleting the created objects after their use expired, languages like Java use a so called "garbage collector".

Having this in mind, languages with a garbage collector are for no use in safety related software, because a fast reaction to a problem can not be guaranteed if the garbage collector interrupts the program at the exact moment of an emergency. An automatic memory management on the other hand is much less bug prone due to the lower complexity [\[KOP09\]](#).

4. Object Oriented Programming in Objective-C

5. Object Oriented Programming in Go

6. Differences between Go and Objective-C regarding OOP

6. Summary

References

- OC209** Stephen G. Kochan, Objective-C 2.0, 2009, Addison-Wesley
- DEP15** Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns - Entwurfsmuster als Elemente wiederverwendbarer objektorientierter Software, 2015, Mitp
- KOP09** Arnd Poetzsch-Heffter, Konzepte objektorientierter Programmierung, Rev. 2, 2009, Springer
- OOT15** Suad Alagic, Object-Oriented Technology, 2015, Springer
- OOA15** Brahma Dathan, Sarnath Ramnath, Object-Oriented Analysis, Design and Implementation - An Integrated Approach, Rev. 2, 2015, Springer
- POL20** Website, [duden.de](https://www.duden.de), called 2020-12-02,
<https://www.duden.de/rechtschreibung/Polymorphismus>

List of Figures

[\[img-go_example_call\]](#) Method call syntax in Go