Object Oriented Programming – Java 1

Lecture 1

Computer Programming Overview

Dr. Obuhuma James

Description

This topic aims at introducing the course on Object Oriented Programming by first giving an overview of computer programming. The topic lays a foundation by dividing computer programming languages into four main categories, referred to as programming paradigms. These are procedural, functional, object-oriented, and logic programming paradigms. The key features of each of the paradigms with examples of programming languages per paradigm are discussed and clearly outlined.

Learning Outcomes

By the end of this topic, you will be able to:

- Understand the reasons behind the existence of different programming paradigms.
- Describe features of the main programming paradigms with examples of programming languages per paradigm.
- Describe the salient features of Object-Oriented Programming languages, that form the basis of this course.

Overview of Computer Programming

A computer is an electronic device that has found many applications in the real world. Devices that fall in the family of computers range from smart watches, smartphones, tablets, laptops, desktops, servers among others. Each of these devices can be perceived to be composed of two main elements, namely, hardware and software. The hardware represents physical, tangible part of the computer while software are the sets of instructions written in a programming language for purposes of controlling the functioning of the hardware and/or for performing specific tasks. Software is sometimes referred to as computer programs. Thus, computer programming is the process of writing or developing computer programs. This is achieved through the use of a specific computer programming language, a development environment, and a compiler.

There are many different programming styles, commonly referred to as programming paradigms. Each of the styles has a set of computer programming languages that embrace it. This course focuses on the Object-Oriented Programming style, with the Java computer programming language being used to demonstrate concepts.

Programming Paradigms

Computer programming paradigms refers to a style or method of used to write computer programs. Thus, the term paradigm is sometimes used interchangeably with terms like style or methodology. There are many different programming paradigms that classify computer programming languages based on the structure of program codes, key unique concepts, …. among others. Some of the most common computer programming paradigms include, functional, procedural, object-oriented, and logic programming paradigm. Each of these paradigms has its own salient features, thus, determining computer programming languages that align to it. There exist many computer programming languages, whose genealogy is as outlined in Figure 1.

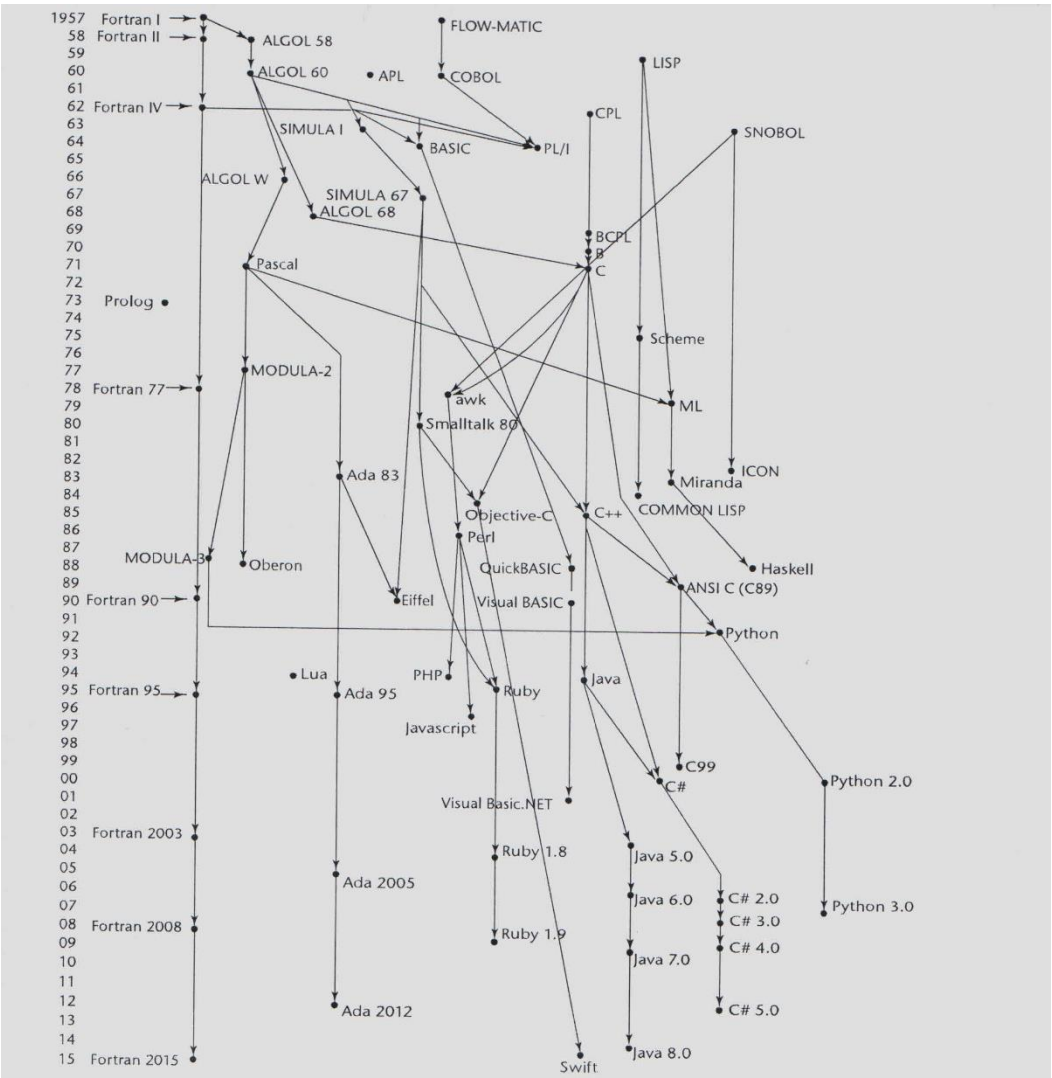

Figure 1. Genealogy of Computer Programming Languages [3]

1. Procedural

Also known as imperative programming paradigm, is the oldest and most popular paradigm that consists of actions to effect changes in the state of a machine. This happened through assignment operations or side effects with well-defined control-flow statements. The control-flow is implemented through conditional and unconditional branching and loops. Procedural programming languages are also characterized by the concepts of declaration and definition of procedures and functions as a way of modularizing program codes. According to [3], the design of procedural programming languages is founded on the Von Neumann architecture, with efficiency being the primary concern rather than the suitability of the language for software development. Thus, the chief features for the procedural paradigm are variables, assignment statements, and iteration [3].

Some examples of procedural programming languages include Fortran, Algol, Cobol, PL/I, Pascal, Modula-2, Ada, C, Java, among others.

2. Functional

Also known as applicative programming paradigm are based on mathematical functions. This makes functional programming languages to embrace a solid theoretical approach that is closer to the user, but relatively unconcerned with the architecture of the machine running the program [3]. Functional programming languages focus on evaluation of functions. Thus, they avoid updates, assignments, mutable (changeable) states, and side effects.

Some examples of functional programming languages include Lisp, Scheme, Standard ML, Haskell, F#, among others.

Some notable differences between procedural and functional programming languages include:

- Operations in procedural languages are carried out with their results stored in variables that could be accessed and used at a later stage [3]. Thus, variable management is a concern and source of complexity. On the contrary, variables are not necessary in functional programming languages. Furthermore, the evaluation of functions always produces similar results given the same set of parameters [3].

- Some of the characteristics of procedural languages include efficient execution, complex semantics and syntax, concurrency control defined by the programmer, among others. On the other hand, characteristics of functional languages include simple semantics and syntax, less efficient execution, concurrency can be automated, among others.

According to [3], purely functional programming languages portray advantages over procedural programming languages. However, they are less commonly used. Some procedural programming languages now incorporate support for functional programming, thus making them multiparadigm.

3. Object Oriented

Object-oriented programming languages portray three major language features, namely, support for abstract data types, inheritance, and polymorphism [3]. Abstract data types are implemented through definition of classes and objects. Inheritance permits definition of new classes in terms of other existing classes, in a manner that allows them to inherit common attributes. Polymorphism on the other hands allows methods to be expressed in different forms. Object oriented programming languages visualizes everything as an object and/or class where an object is a real-world thing while a class is a group of related objects.

Some examples of Object-oriented programming languages include Simula, Smalltalk, C++, Modula-3, Java, C#, Ruby, among others.

The first programming language that had full support for object-oriented concepts was Smalltalk [3]. Thus, Smalltalk forms an important part of any discussion of the evolution of programming languages. Smalltalk pioneered the graphical user interface design concepts featured in the Object-Oriented Programming approach [3].

4. Logic

Logic programming is based on predicate logic, targeted at proving of theorems, automated reasoning, database applications among others. Programs under this paradigm are expressed in form of symbolic logic [3] which happens in a declarative rather than procedural approach.

As such, only specification of results is stated without detailed procedures for producing the results. In most cases, statements are expressed as facts, rules, or goals.

Some examples of logic programming languages include Prolog, among others.

5. Other Paradigms

Other programming paradigms worth mentioning include markup languages, concurrent, scripting languages and multiparadigm. Markup languages are commonly used in the design of web pages. They are characterized by the use of tag or elements. Examples of markup languages for the web include XML and HTML. Concurrent programming paradigm permits for execution of statements simultaneously. This is one of the features that seems to be trending in modern-day programming languages. On the other hand, scripting languages are lightweight programming languages that borrow programming constructs from heavyweight programming languages. Examples of scripting languages include PHP, ASP, Jscript, JavaScript, VB Script, Perl, Python, Ruby, among others.

It is worth noting that some programming languages possess features that align to more than one programming style. Such languages are hence referred to as multiparadigm. This seems to be the norm for most of the modern-day programming languages. For instance, Java a procedural, object-oriented and concurrent programming language while Python and Ruby are Object-oriented scripting languages. These and many other languages can be categorized as multiparadigm.

Despite the variances in the programming styles as covered in the different programming paradigms, there are some concepts that unify the programming languages. These includes data types, expressions, functions/procedures, and commands.

Figure 2 outlines a summary of the programming paradigms with some examples of programming languages in each case.

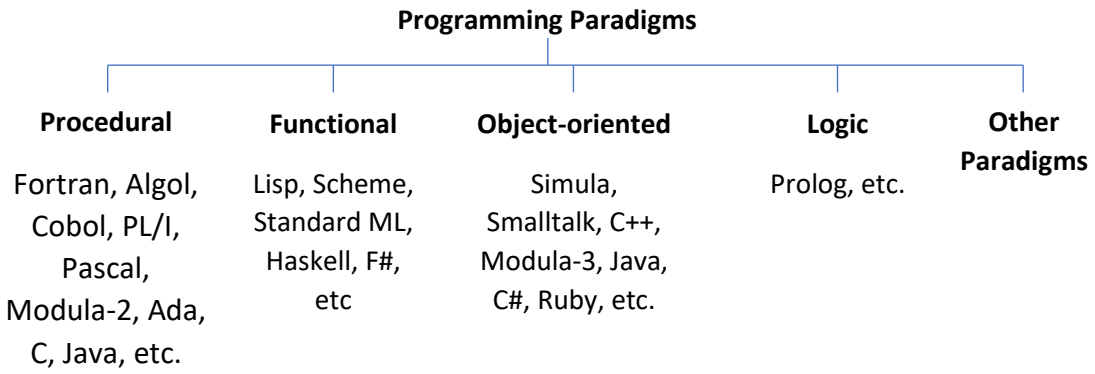

Figure 2. Programing Paradigm

The Course Focus

This course will focus on a thorough exploration of Object-Oriented Programming paradigm. The Java programming language will used to demonstrate concepts throughout the course. Thus, the first topics will lay a foundation to the Java programming languages. Afterwards, emphasis per topic will be on outlining and demonstrating object-oriented programming concepts, without necessarily focusing on the learning the Java programming language. The main assumption is that learners taking this course have already been introduced to programming, preferably a procedural programming language. The main topics that will be covered in the course are:

1. Basic elements of Java programming
2. Methods
3. Classes and Objects
4. Control Structures: Decision-making
5. Control Structures: Looping
6. Arrays
7. Inheritance
8. Encapsulation
9. Polymorphism
10. Exception Handling
11. File Input and Output

The Java compiler is the main tool required for demonstration of concepts. Hence, an Integrated Development Environment (IDE) that supports Java programming should be setup

in advance. These includes NetBeans, IntelliJ, Eclipse, among others. The NetBeans IDE will be used for demonstration purposes.

Summary

The topic has introduced the course by covering an overview of computer programming. Key features of four main programming paradigms have been discussed. These are the procedural, object-oriented, functional and logic programming paradigms. Furthermore, other programming paradigms have been briefly discussed, including, markup languages, scripting languages, concurrent and multiparadigm. Finally, an outline of the topics to be covered in the course has been provided where the first topics will focus on the Java programming languages which later transitions to topics on object-oriented programming concepts.

Check Points

1. Narrate your understanding of the term programming paradigm.
2. List the four main programming paradigms.
3. Describe the differences among the four main programming paradigms.
4. Outline examples of programming languages that fall under each of the four paradigms.
5. Discuss features of programming languages categorized as multiparadigm.
6. Briefly state and explain features of any other paradigms not covered in this topic.

Core Textbooks

1. Joyce Farrell, Java Programming, 7th Edition. Course Technology, Cengage Learning, 2014, ISBN-13 978-1-285-08195-3.
2. Malik, Davender S. JavaTM Programming: From Problem Analysis to Program Design, International Edition, 5th Edition, Cengage Learning.

Other Resources

3. Daniel Liang, Y. "Introduction to Java Programming, Comprehensive." (2011).
4. Malik, Davender S. JavaTM Programming: From Problem Analysis to Program Design, International Edition, 4th Edition, Cengage Learning, 2011.
5. Shelly, Gary B., et al. Java programming: comprehensive concepts and techniques. Cengage Learning, 2012.

References

[1] Farrell, J., Java Programming, 7th Edition. Course Technology, Cengage Learning, 2014, ISBN-13 978-1-285-08195-3.

[2] Malik, D. S., JavaTM Programming: From Problem Analysis to Program Design, International Edition, 5th Edition, Cengage Learning.

[3] Sebester, R. W., Concepts of Programming Languages, 12th Edition, Pearson, 2018, ISBN 0-321-49362-1.