PREFACE

The SIG CS@e-Learning committee sincerely appreciates the dedication and contributions of the educators from Jabatan Sains Komputer & Matematik (JSKM), UiTM Penang Branch, in bringing the 9th edition to fruition. This edition received 30 scholarly articles, all of which met the required criteria and were accepted. Authors are encouraged to further refine their research with additional insights and discussions for potential publication in high-impact journals indexed by SCOPUS, WOS, or ERA.

The theme for the ninth volume, "Beyond Boundaries: The Multidimensional Horizons of E-Learning," reflects the continuous evolution of digital learning. Over the past few decades, e-learning has proven to be a transformative force in education, demonstrating exceptional adaptability and effectiveness. The widespread use of mobile technology has expanded its reach, making e-learning an essential component not only in higher education and vocational training but also in primary and secondary education. Emerging trends such as artificial intelligence (AI), micro-credentials, big data, virtual and augmented reality, blended learning, cloud-based platforms, gamification, mobile learning, the Internet of Things (IoT), and online video are reshaping the digital learning landscape.

SIG CS@e-Learning remains dedicated to fostering academic excellence through impactful publications. With continuous commitment and innovation, we aspire for JSKM to attain recognition in esteemed academic journals, further advancing the frontiers of e-learning.

Ts. Jamal Othman
Chief Editor
SIG CS@e-LEARNING
Beyond Boundaries : The Multidimensional Horizons of E-Learning
Vol. 9, 24 March 2025

e-ISBN : 978-629-98755-5-0
SIG : e-Learning@CS
https://appspenang.uitm.edu.my/sigcs/
Publication Date: 24 – Mac - 2025

STRUCTURAL DIFFERENCES OF CONSTRUCTOR METHODS IN OBJECT-ORIENTED PARADIGMS: A COMPARATIVE STUDY USING C++, JAVA & PYTHON

*Jamal Othman[1], Syarifah Adilah Mohamed Yusoff[2], Arifah Fasha Rosmani[3] and Mohd Saifulnizam Abu Bakar[4]
**jamalothman@uitm.edu.my [1], syarifah.adilah@uitm.edu.my [2], arifah840@uitm.edu.my[3], mohdsaiful071@uitm.edu.my[4]*

[1,2,3,4]Jabatan Sains Komputer & Matematik (JSKM),
Universiti Teknologi MARA Cawangan Pulau Pinang, Malaysia

**Corresponding author*

ABSTRACT

Structure of a class in Object Oriented Programming involves the attributes and methods. The methods of a class normally consist of constructors, mutators, retrievers, processors and printers. Among these, constructor methods play a vital role in initializing objects and are essential in class construction within the Object-Oriented paradigm. Constructor method is the most important method in constructing a class. This article will compare the structure of constructor methods in three (3) different programming languages such as the C++, Java and Python, providing a clear comparison of how each language handles object initialization. These three programming languages support the Object-Oriented Paradigms. The comparative will be focusing on the syntaxes which are applied in each programming language to construct the constructor methods. Constructor methods can be categorized as default, normal and copy constructors and comparison of the three programming languages will be shown to compare the structure for each category of constructor methods. Understanding these differences is crucial for programmers who work with multiple languages and need to adapt their coding approaches accordingly.

Keywords: object-oriented, object, class, constructors, instantiate

Introduction

Object-oriented Programming is one of the programming paradigms classifications besides the logic, functional, imperative or structured and scripting programming paradigms. Object-oriented has special features such as the class, inheritance, polymorphism, encapsulation, information hiding and abstraction. The basic element of object-oriented is the class which consists of the attributes and methods. Object-oriented encourages the practitioners to create a new abstract data type (ADT) which is called class. Once the class has been created, several objects can be instantiated to the same class. The only difference among objects is the behaviours of each object (Wikipedia, 2025).

Methods of a class may consist of constructors, mutators, retrievers, processors and printers. The most important method is the constructor. Constructor method is important before an object can be instantiated to a class. Without the constructor of a class, the objects cannot be created or instantiated. Constructor is a method which has the same name as the class and no return statement is applied in the function or method (Queenskisivuli, 2025).

The constructor consists of two types of methods which are the default constructor and parameterized or normal constructor. Both constructors are needed in any class. The default constructor

initializes each attribute in the class, while the parameterized constructor initializes the object's attributes (MindStick, 2025). Constructor methods support the overloading which means the class may have a similar method name but holds a different number of parameters or order of parameters types in each constructor method. The appropriate constructor will be used based on the number, types, and order of the parameters provided (Logicmojo, 2025).

The main purpose of this article is to show and present the construction of constructors in three different programming languages, namely C++, Java, and Python. By examining the implementation of constructors in these three widely-used programming languages, the article aims to provide a comprehensive understanding of how object initialization is handled in each case. The comparison will focus primarily on the structure and syntax used to define and utilize constructors, highlighting the similarities and differences among C++, Java and Python.

By analysing these distinctions, the article will provide valuable insights into how each programming language approaches object-oriented design and initialization. Additionally, this comparison will help programmers better understand the nuances of each language, enabling them to write more efficient and maintainable code.

Constructor Methods in C++ Programming Language

C++ programming language is one of the programming languages which support object-oriented programming (OOP) but it is not a pure OOP. C++ is normally used for writing procedures or functions to perform processing on data. The following program fragment shows how to write the constructor methods in C++ programming language.

```
#include <iostream>
using namespace std;
class Employee {
	public:
		//attributes:
		int empID;
		string empName;
		float salary;

		//default constructor
		Employee()
		{
			empID = 0;
			empName = "";
			salary = 0.0;
		}

		//normal constructor
		Employee(int eID, string eName, float sal)
		{
			empID = eID;
			empName = eName;
			salary = sal;
		}

};
```

```
int main()
{
        Employee emp1 (123, "MOHD AMIN BIN ABU BAKAR",4500);
        Employee emp2 (128, "MAZLINA ABDUL SHUKOR",6750);

        cout << "\n Employee ID         : " << emp1.empID;
        cout << "\n Employee Name       : " << emp1.empName;
        cout << "\n Employee Salary RM : " << emp1.salary;

        return 0;
}
```

Figure 1: Sample of codes with Constructors in C++ Programming Language

The above program fragment shows a class named `Employee` which consists of three (3) attributes such as the employee ID, name and salary. Two constructors are included in the class named default and normal constructors. Both constructors have a similar name as the class name. The default constructor has no parameters received by the method, while the normal constructor receives the values from the object through the parameters. The access modifier of the class has been set as `public`, so that the main program is allowed to access methods and attributes from the class. C++ supports the overloading for constructors.

Constructor Methods in JAVA Programming Language

Java programming language is classified as Object-oriented Programming (OOP) which means the class becomes the dominant in constructing the codes in Java (Othman, 2010). Similar to C++, the constructor's name in Java must match with the name of the class and the constructor method cannot have a return value. The constructor's method is applied when the object is instantiated to the class. The following program fragment shows how to write the constructor methods in Java programming language.

```
public class Employee {
        //attributes
        int empID;
        String empName;
        float salary;

        //default constructor
        Employee()
        {       empID=0;
                empName ="";
                salary=0;
        }

        //normal constructor
        Employee(int empID, String empName, float salary)
        {       this.empID = empID;
                this.empName = empName;
                this.salary = salary;
        }
}
```

```
public class empApp {
	public static void main(String[] args)
	{
		Employee emp1 = new Employee (123, "MOHD AMIN BIN ABU BAKAR",4500);
		Employee emp2 = new Employee (128, "MAZLINA ABDUL SHUKOR",6750);

		System.out.println("Employee ID     : "+emp1.empID);
		System.out.println("Employee name   : "+emp1.empName);
		System.out.println("Employee salary : RM "+emp1.salary);
	}
}
```

Figure 2: Sample of codes with Constructors in Java Programming Language

The program fragment in figure 2 shows the implementation of the default and normal constructor in the class named `Employee`. Both in C++ and Java codes, the default constructor looks similar. The normal constructor uses the keyword `this` because the attributes and the parameters name are similar. The keyword `this` is referring to the class name and substitute to each attribute of the `Employee` class. The parameters received the values from the object named `emp1` which instantiated to the `Employee` class as shown in the application program (class named `empApp`). Java supports the overloading for constructors.

Constructor Methods in Python Programming Language

Both Python and Java are categorized as Object-oriented Programming (OOP) languages. Everything in Python is an object, with its properties and methods. The constructor in Python can be recognized through the built-in `__init__()` function. The object created in Python will be instantiated to the `__init__()` function or constructor. The following program fragment shows how to write the constructor methods in Python programming language.

```
class Employee:

    #default constructor
    def __init__(self):
        self.empID = 0
        self.empName = ""
        self.salary = 0

    #default constructor
    def __init__(self, empID, empName, salary):
        self.empID = empID
        self.empName = empName
        self.salary = salary

def main():
    emp1 = Employee(123, "MOHD AMIN BIN ABU BAKAR",4500)
    emp2 = Employee(128, "MAZLINA ABDUL SHUKOR",6750)

    print("Employee ID     : ",emp1.empID)
    print("Employee Name   : ",emp1.empName)
    print("Employee Salary : RM ",emp1.salary)
main()
```

Figure 3: Sample of codes with Constructors in Python Programming Language

The constructor indicator as shown in figure 3 is recognized through the `__init__()` identifier. It has two structures of constructor which the default constructor has one parameter named `self` which means it refers to the name of the class. While the normal constructor in Python is similar to in C++ and Java programming which received the values from the object which instantiated to the `Employee` class in the application program. The constructor in Python has no explicit overloading as applied in C++ or Java programming languages.

Discussion

In C++, constructors are typically named after the class, ensuring that they are easily identifiable and closely associated with the class they initialize. This naming convention enhances code readability and maintainability. C++ also supports constructor overloading, allowing multiple constructors with different parameters to coexist within the same class. This feature provides flexibility, as objects can be initialized in various ways depending on the arguments passed during instantiation. Additionally, C++ includes a copy constructor, which is used to create a new object as a copy of an existing one, enabling precise control over object copying.

Java, on the other hand, follows a similar naming convention where the constructor is named after the class, maintaining consistency and clarity. However, Java extends the concept with additional features such as constructor chaining, which allows one constructor to call another within the same class using the `this()` keyword. This promotes code reuse and reduces redundancy by enabling a sequence of constructor calls, each adding incremental initialization logic. Java also provides the `super()`

keyword to invoke the parent class's constructor, ensuring proper initialization in inheritance hierarchies.

Python, known for its simplicity and readability, takes a different approach by using the `__init__` method as its constructor. Unlike C++ and Java, Python's constructor is not named after the class but is standardized as `__init__(self)`, which is consistent across all classes. This design choice enhances uniformity and reduces confusion. Although Python does not support constructor overloading in the traditional sense, it allows default parameter values and flexible argument handling, which can achieve similar functionality. Python's emphasis on simplicity and explicitness is evident in its constructor design, making it more approachable for beginners.

Conclusion

In conclusion, while C++, Java, and Python all provide mechanisms for object initialization through constructors, they do so in ways that reflect their underlying philosophies and design goals. C++ prioritizes control and efficiency with its explicit overloading and copy constructors. Java emphasizes consistency and robust inheritance handling with constructor chaining and `super()` calls. Python, in contrast, opts for simplicity and readability with its `__init__` method and flexible argument handling. Understanding these differences is crucial for developers working across these languages, as it enables them to effectively leverage each language's unique strengths.

References:

Logicmojo. (n.d.). *Constructor in OOPs*. Logicmojo. Retrieved February 13, 2025, from https://logicmojo.com/constructor-in-oops#4

MindStick. (n.d.). Understanding constructors in object-oriented programming (OOP). MindStick. Retrieved February 13, 2025, from https://www.mindstick.com/articles/334044/understanding-constructors-in-object-oriented-programming-oop

Othman, J. (2010), Fundamentals of Programming: With Examples in C, C++ and Java, UPENA UiTM, pp 11 – 15, ISBN 978-967-363-110-0.

Queenskisivuli. (n.d.). How to use class and object constructors in OOP. Medium. Retrieved February 13, 2025, from https://medium.com/@queenskisivuli/how-to-use-class-and-object-constructors-in-oop-5212f7f03c39

Wikipedia contributors. (n.d.). Constructor (Object-Oriented programming). Wikipedia. Retrieved February 13, 2025, from https://en.wikipedia.org/wiki/Constructor_(object-oriented_programming)

SIG CS@e-Learning
Unit Penerbitan
Jabatan Sains Komputer & Matematik
Kolej Pengajian Pengkomputeran, Informatik & Matematik
Universiti Teknologi MARA Cawangan Pulau Pinang

e-ISBN : 978-629-98755-5-0

Design of the cover powered by FPPT.com