OBJECT ORIENTED METHODOLOGY

Theory- 4

Lecture Notes

3rd Semester CSE/IT

By

Nishita Kindo, Lecturer (CSE)

Bhubananda Odisha School of Engineering, Cuttack

CHAPTER-1
OBJECT ORIENTED PROGRAMMING (OOPS) CONCEPTS

1.1 Programming Languages

Computer program is a collection of instructions that can be executed by a computer to perform a specific **task**. A computer program is usually written by a computer programmer in a programming language.
Computer programming is the process of designing and building an executable computer program to accomplish a specific computing result or to perform a specific task.

- A programming language is a vocabulary and set of grammatical rules for instructing a computer or computing device to perform specific tasks.
- The term programming language usually refers to high-level languages, such **as BASIC, C, C++, COBOL, Java, FORTRAN, Ada, and Pascal.**
- Each programming language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.
- Regardless of what language you use, you eventually need to convert your program into machine language so that the computer can understand it. There are two ways to do this:
- 1) Compile the program.
  2) Interpret the program.

1) Compile the program.

To transform a program written in a high-level programming language from source code into object code. Programmers write programs in a form called source code. Source code must go through several steps before it becomes an executable program. The first step is to pass the source code through a compiler, which translates the high-level language instructions into object code.

2) Interpret the program

Interpreter is a program that executes instructions written in a high-level language. There are two ways to run programs written in a high-level language. The most common is to compile the program; the other method is to pass the program through an interpreter.

GENERATIONS OF PROGRAMMING LANGUAGE

Programming languages have been developed over the year in a phased manner. Each phase of developed has made the programming language more user-friendly, easier to use and more powerful. Each phase of improved made in the development of the programming languages can be referred to as a generation. The programming language in terms of their performance reliability and robustness can be grouped into **five different generations**,

1. First generation languages (1GL)
2. Second generation languages (2GL)
3. Third generation languages (3GL)
4. Fourth generation languages (4GL)
5. Fifth generation languages (5GL)

1. First Generation Language (Machine language)

The first-generation programming language is also called low-level programming language because they were used to program the computer system at a very low level of abstraction. i.e. at the machine level. The machine language also referred to as the native language of the computer system is the first-generation programming language. In the machine language, a programmer only deals with a binary number.

Advantages of first-generation language

- They are translation free and can be directly executed by the computers.
- The programs written in these languages are executed very speedily and efficiently by the CPU of the computer system.
- The programs written in these languages utilize the memory in an efficient manner because it is possible to keep track of each bit of data.

2. Second Generation language (Assembly Language)

The second-generation programming language also belongs to the category of low-level-programming language. The second-generation language comprises assembly languages that use the concept of mnemonics for the writing program. In the assembly language, symbolic names are used to represent the opcode and the operand part of the instruction.

Advantages of second-generation language

- It is easy to develop understand and modify the program developed in these languages are compared to those developed in the first-generation programming language.
- The programs written in these languages are less prone to errors and therefore can be maintained with a great case.

3. Third Generation languages (High-Level Languages)

The third-generation programming languages were designed to overcome the various limitations of the first and second-generation programming languages. The languages of the third and later generation are considered as a high-level language because they enable the programmer to concentrate only on the logic of the programs without considering the internal architecture of the computer system.

Advantages of third generation programming language

- It is easy to develop, learn and understand the program.
- As the program written in these languages are less prone to errors they are easy to maintain.
- The program written in these languages can be developed in very less time as compared to the first and second-generation language.

Examples: FORTRAN, ALGOL, COBOL, C++, C

4. Fourth generation language (Very High-level Languages)

The languages of this generation were considered as very high-level programming languages required a lot of time and effort that affected the productivity of a programmer. The fourth generation programming languages were designed and developed to reduce the time, cost and effort needed to develop different types of software applications.

Advantages of fourth-generation languages

- These programming languages allow the efficient use of data by implementing the various database.
- They require less time, cost and effort to develop different types of software applications.
- The program developed in these languages are highly portable as compared to the programs developed in the languages of other generation.

Examples: SOL, CSS, ColdFusion

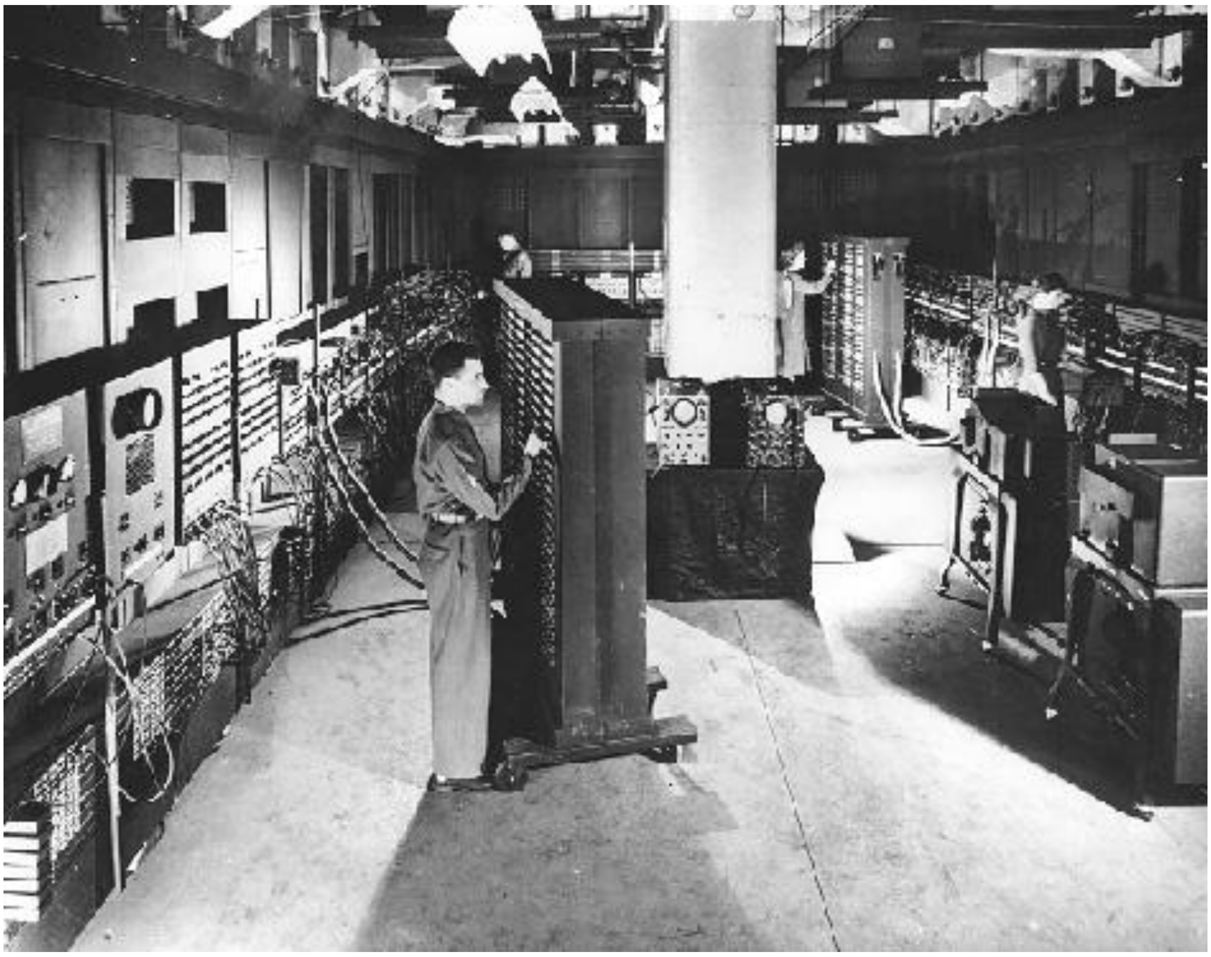
5. Fifth generation language (Artificial Intelligence Language)

The programming languages of this generation mainly focus on constraint programming. The major fields in which the fifth-generation programming language are employed are Artificial Intelligence and Artificial Neural Networks

Advantages of fifth-generation languages

- These languages can be used to query the database in a fast and efficient manner.
- In this generation of language, the user can communicate with the computer system in a simple and an easy manner.

Examples: mercury, prolog, OPS5



Can anyone Guess what is it???????????????

TYPES OF PROGRAMMING LANGUAGES

1. **MACHINE LANGUAGES:** Imagine them as the **"native tongue" of the computer**, the language closest to the hardware itself. Each unique computer has a unique machine language. A machine language program is **made up of a series of binary patterns** (e.g., 01011100) which represent simple operations that can be accomplished by the computer (e.g., add two operands, move data to a memory location). Machine language programs are *executable,* meaning that they can be run directly. Programming in machine language requires memorization of the binary codes and can be difficult for the human programmer.

2. **ASSEMBLY LANGUAGES:** They represent an effort to make programming easier for the human. **The machine language instructions are replaced with simple pneumonic abbreviations (e.g., ADD, MOV).** Thus, assembly languages are unique to a specific computer (machine). Prior to execution, an assembly language program requires translation to machine language. This translation is accomplished by a computer program known as an Assembler. Assemblers are written for each unique machine language.

3. **HIGH LEVEL LANGUAGES:** High-level languages, like **C, C++, JAVA** etc., are more English-like and, therefore, make it easier for programmers to "think" in the programming language. High-level languages also require translation to machine language before execution. This translation is accomplished by either a compiler or an interpreter. *Compilers* translate the entire source code program before execution. (E.g.: C++, Java) *Interpreters* translate source code programs one line at a time. (E.g.: Python) Interpreters are more interactive than compilers.

1.2 What is object-oriented programming (OOP)?

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behaviour. *Or* **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects.

What is Object?

Object means a real-world entity such as a pen, chair, table, computer, watch, etc.

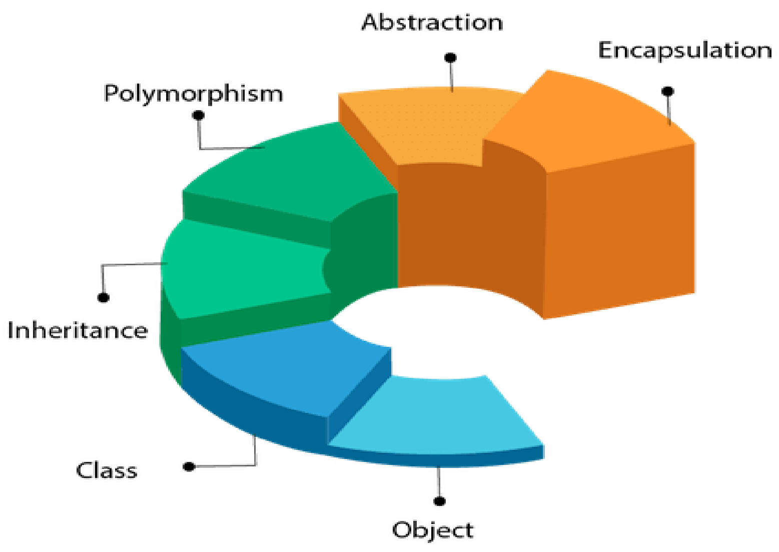
Why OOP?

OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained.

1.3 OOP Concepts and Terminology

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation
- Message Passing
- Dynamic Binding

OOPs (Object-Oriented Programming System)



1. Object

Any entity that has state and behaviour is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

Example: A dog is an object because it has states like colour, name, breed, etc. as well as behaviour like wagging the tail, barking, eating, etc.

2. Class

Collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

3. Inheritance

When one object acquires all the properties and behaviours of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

4. Polymorphism

"Now Speak!"
"Quack"
"Woof"
"Meow"

If **one task is performed in different ways**, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

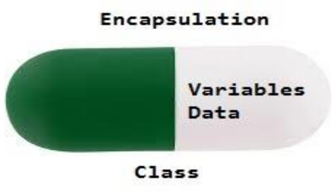
Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

5. Abstraction

Hiding internal details and showing functionality is known as abstraction. For example, phone call, we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction.

6. Encapsulation



Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

7. Message Passing:

Objects communicate with one another by sending and receiving information to each other. A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results. Message passing involves specifying the name of the object, the name of the function and the information to be sent.

8. Dynamic Binding

Dynamic binding also called dynamic dispatch is the process of linking procedure call to a specific sequence of code (method) at run-time. It means that the code to be executed for a specific procedure call is not known until run-time. Dynamic binding is also known as **late binding or run-time binding**.

1.4 Benefits of OOPs

- It is easy to model a real system as real objects are represented by programming objects in OOP. The objects are processed by their field (member data) and methods (functions). **It is easy to analyse the user requirements.**
- **With the help of inheritance, we can reuse the existing class to derive a new class** such that the redundant code is eliminated and the use of existing class is extended. This saves time and cost of program.
- In OOP, data can be made private to a class such that only member functions of the class can access the data. **This principle of data hiding helps the programmer to build a secure program** that cannot be invaded by code in other part of the program.
- **With the help of polymorphism, the same function or same operator can be used for different purposes.** This helps to manage software complexity easily.
- **Large problems can be reduced to smaller and more manageable problems**. It is easy to partition the work in a project based on objects.
- **It is possible to have multiple instances of an object to co-exist without any interference** i.e. each object has its own separate field (member data) and methods (functions).

DIFFERENCE BETWEEN OBJECT ORIENTED PROGRAMMING AND PROCEDURAL ORIENTED PROGRAMMING

OOP	**POP**
Object oriented.	Structure oriented.
Program is divided into objects.	Program is divided into functions.
Bottom-up approach.	Top-down approach.
Inheritance property is used.	Inheritance is not allowed.
It uses access specifier.	It doesn’t use access specifier.
Encapsulation is used to hide the data.	No data hiding.
Concept of virtual function.	No virtual function.
C++, Java.	C, Pascal.

1.5 Main application areas of OOP:

- User interface design such as windows, menu.
- Real Time Systems
- Simulation and Modelling
- Object oriented databases
- AI and Expert System
- Neural Networks and parallel programming
- Decision support and office automation systems etc.