PROGRAMMING METHODOLOGIES - INTRODUCTION

When programs are developed to solve real-life problems like inventory management, payroll processing, student admissions, examination result processing, etc. they tend to be huge and complex. The approach to analyzing such complex problems, planning for software development and controlling the development process is called **programming methodology**.

Types of Programming Methodologies

There are many types of programming methodologies prevalent among software developers –

Procedural Programming

Problem is broken down into procedures, or blocks of code that perform one task each. All procedures taken together form the whole program. It is suitable only for small programs that have low level of complexity.

Example – For a calculator program that does addition, subtraction, multiplication, division, square root and comparison, each of these operations can be developed as separate procedures. In the main program each procedure would be invoked on the basis of user's choice.

Object-oriented Programming

Here the solution revolves around entities or objects that are part of problem. The solution deals with how to store data related to the entities, how the entities behave and how they interact with each other to give a cohesive solution.

Example – If we have to develop a payroll management system, we will have entities like employees, salary structure, leave rules, etc. around which the solution must be built.

Functional Programming

Here the problem, or the desired solution, is broken down into functional units. Each unit performs its own task and is self-sufficient. These units are then stitched together to form the complete solution.

Example – A payroll processing can have functional units like employee data maintenance, basic salary calculation, gross salary calculation, leave processing, loan repayment processing, etc.

Logical Programming

Here the problem is broken down into logical units rather than functional units. **Example:** In a school management system, users have very defined roles like class teacher, subject teacher, lab assistant, coordinator, academic in-charge, etc. So the software can be divided into units depending on user roles. Each user can have different interface, permissions, etc.

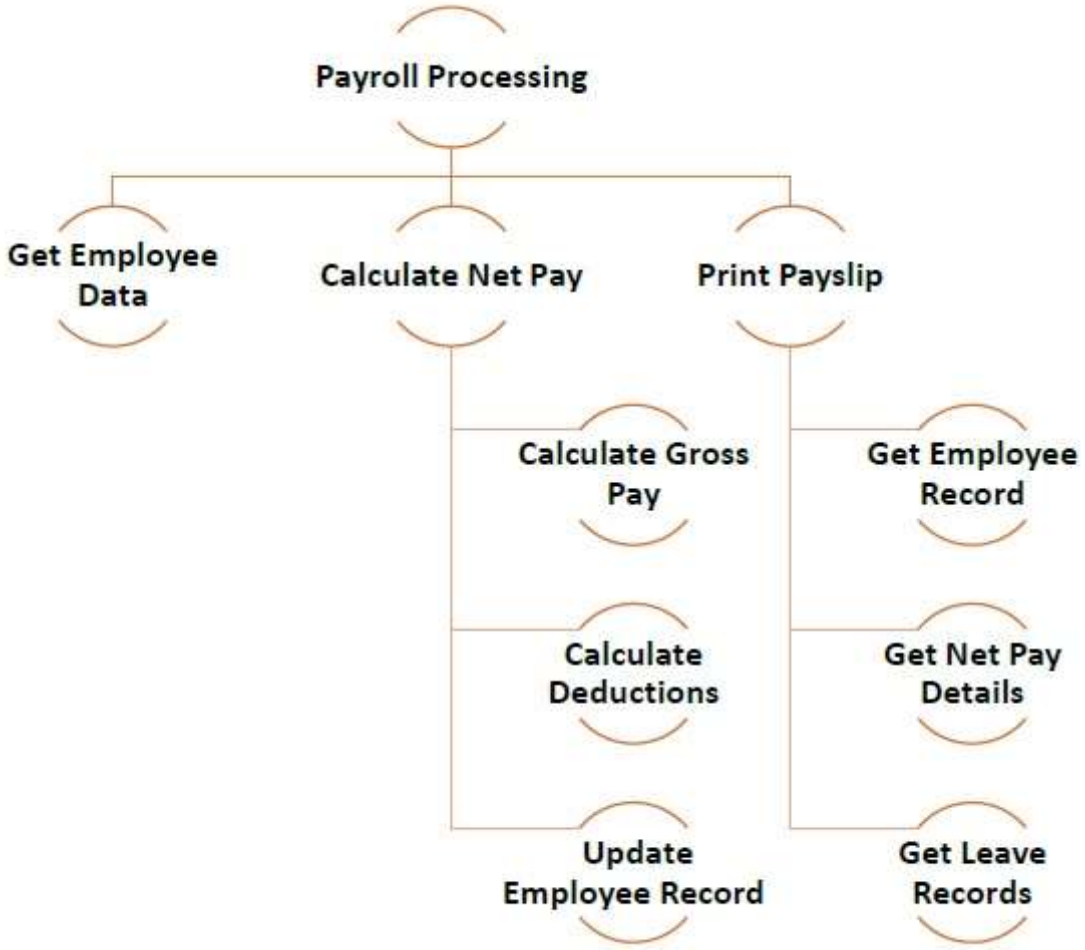
Software developers may choose one or a combination of more than one of these methodologies to develop a software. Note that in each of the methodologies discussed, problem has to be broken down into smaller units. To do this, developers use any of the following two approaches –

- Top-down approach
- Bottom-up approach

Top-down or Modular Approach

The problem is broken down into smaller units, which may be further broken down into even smaller units. Each unit is called a **module**. Each module is a self-sufficient unit that has everything necessary to perform its task.

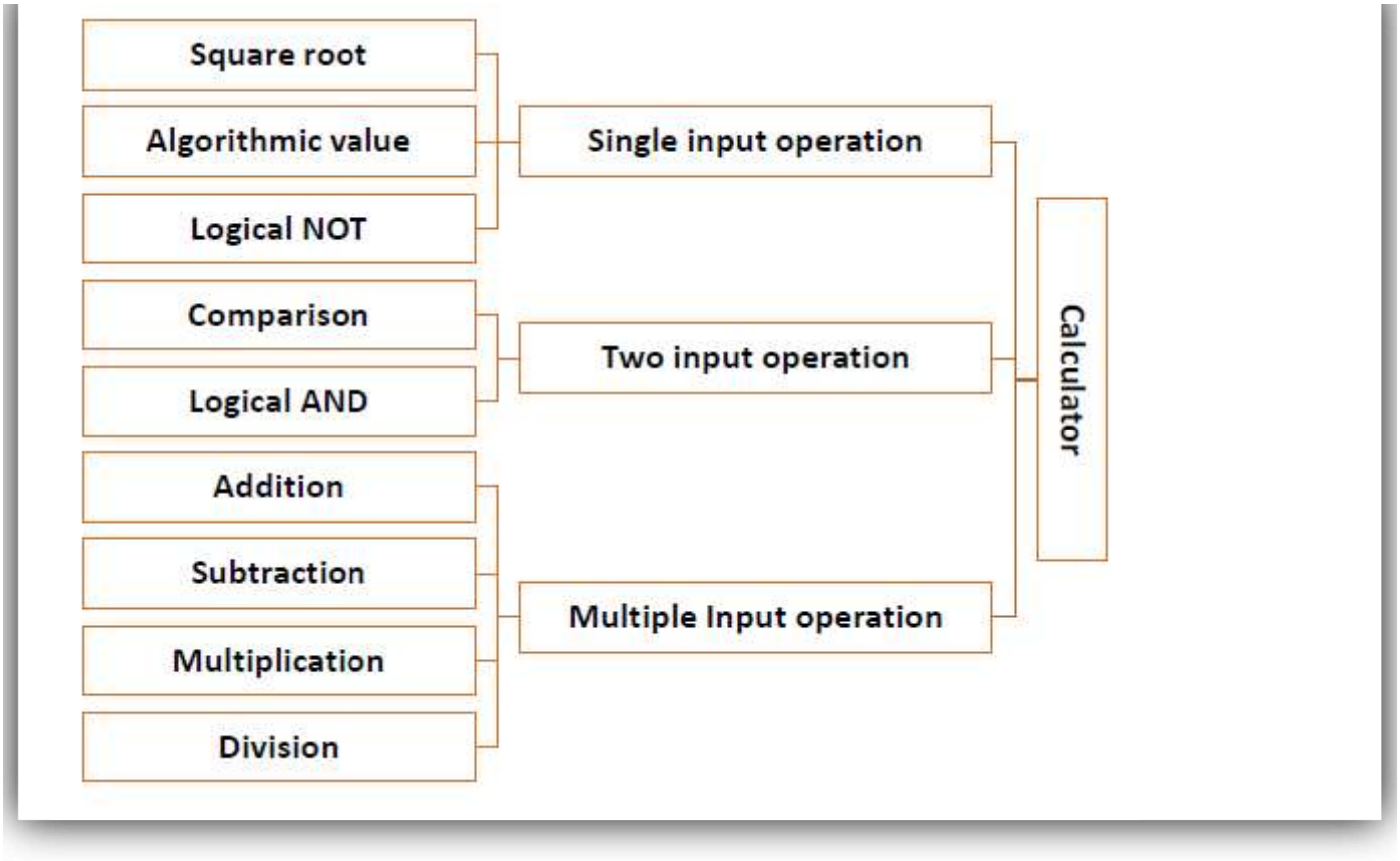
The following illustration shows an example of how you can follow modular approach to create different modules while developing a payroll processing program.



Bottom-up Approach

In bottom-up approach, system design starts with the lowest level of components, which are then interconnected to get higher level components. This process continues till a hierarchy of all system components is generated. However, in real-life scenario it is very difficult to know all lowest level components at the outset. So bottoms up approach is used only for very simple problems.

Let us look at the components of a calculator program.


Square root
Algorithmic value
Logical NOT
Comparison
Logical AND
Addition
Subtraction
Multiplication
Division
Single input operation
Two input operation
Multiple Input operation
Calculator