2. Introduction of Object Oriented Programming

Syeda Jannatul Naim
Lecturer, Dept. of CSE, WUB

January, 2026

Content

- Introduction
- Paradigm Evolution
- Motivation
- Categories of languages that support OOP
- Concepts used extensively in OOP
- Class and Objects
- Encapsulation and Data abstraction
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing
- Conclusion
- References

Introduction

Object-oriented programming is a method of programming based on a hierarchy of **classes**, and well-defined as well as cooperating **objects**. OOP breaks up problems based on object.

"Class and object are building block of OOP"

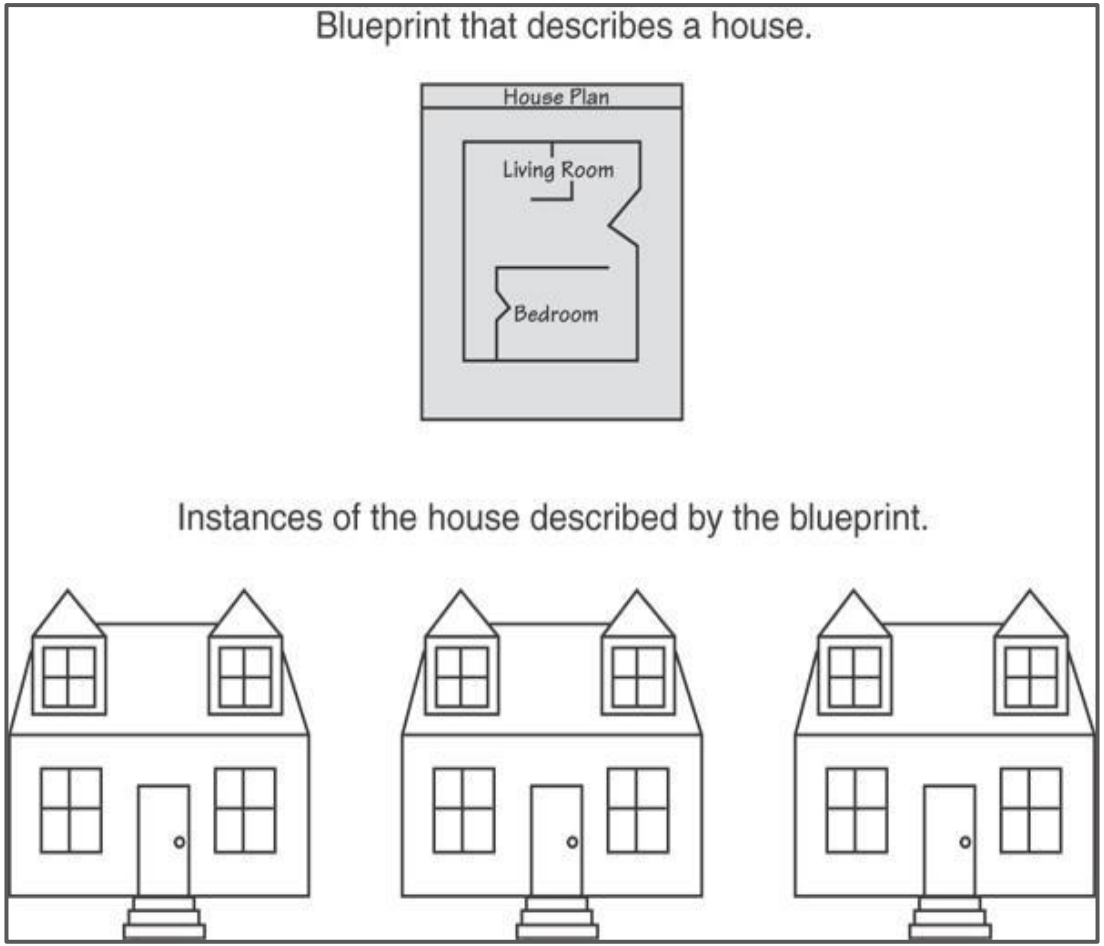
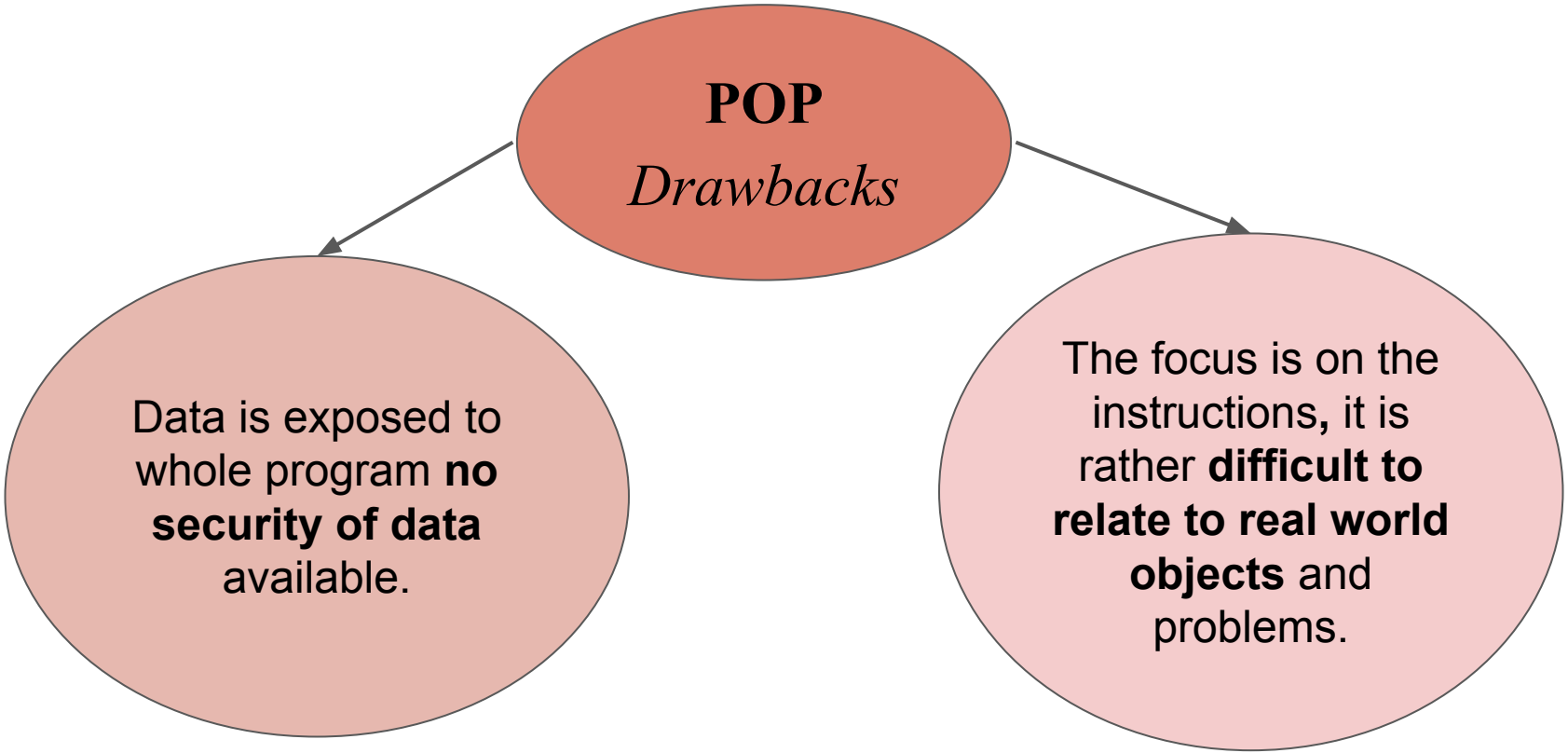


Figure 1 : Example of class & object

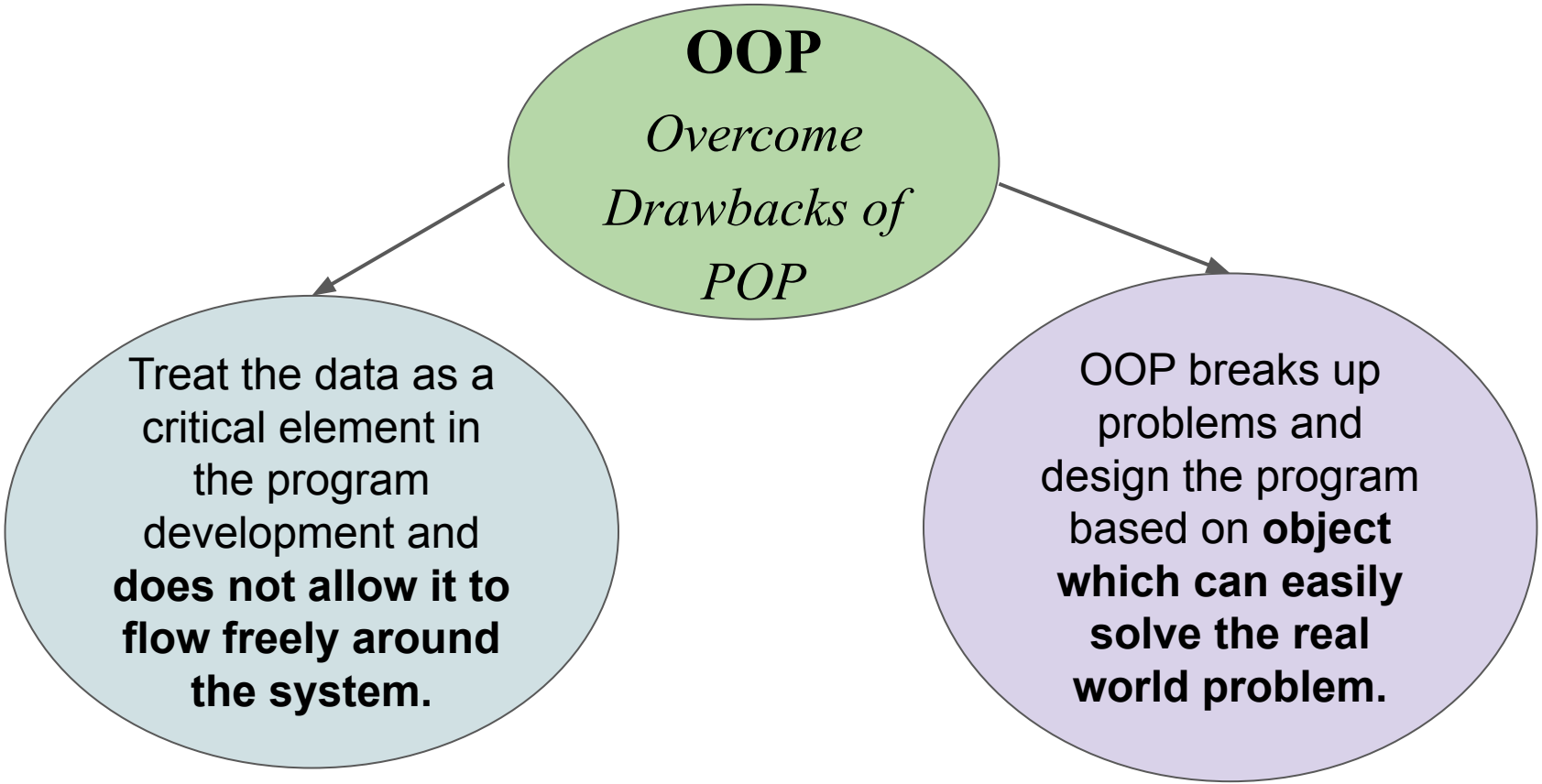
Paradigm Evolution

1. Procedural - 1950s-1970s (procedural abstraction)
2. Data-Oriented - early 1980s (data abstraction)
3. OOP - late 1980s (inheritance and dynamic binding)

Motivation


POP
Drawbacks
Data is exposed to whole program **no security of data** available.
The focus is on the instructions, it is rather **difficult to relate to real world objects** and problems.

Motivation


OOP
Overcome Drawbacks of POP
Treat the data as a critical element in the program development and **does not allow it to flow freely around the system.**
OOP breaks up problems and design the program based on **object which can easily solve the real world problem.**

Categories of languages that support OOP

- C++ (also supports procedural and data-oriented programming)
- Ada 95 (also supports procedural and data-oriented programming)
- CLOS (also supports functional programming)
- Scheme (also supports functional programming)
- Java (based on C++)
- Smalltalk (pure OOP)
- Python (also supports procedural programming) .....etc.

Concepts used extensively in OOP

- Class
- Object
- Data abstraction and Encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing

Class and Object

- **Class :** Classes are user-defined data types that act as the prototype or template or blueprint for individual objects.

- **Object :** Objects are instances or variables of a class created with specifically defined data. Objects can correspond to real-world objects or an abstract entity.

"Class and object are building block of OOP"

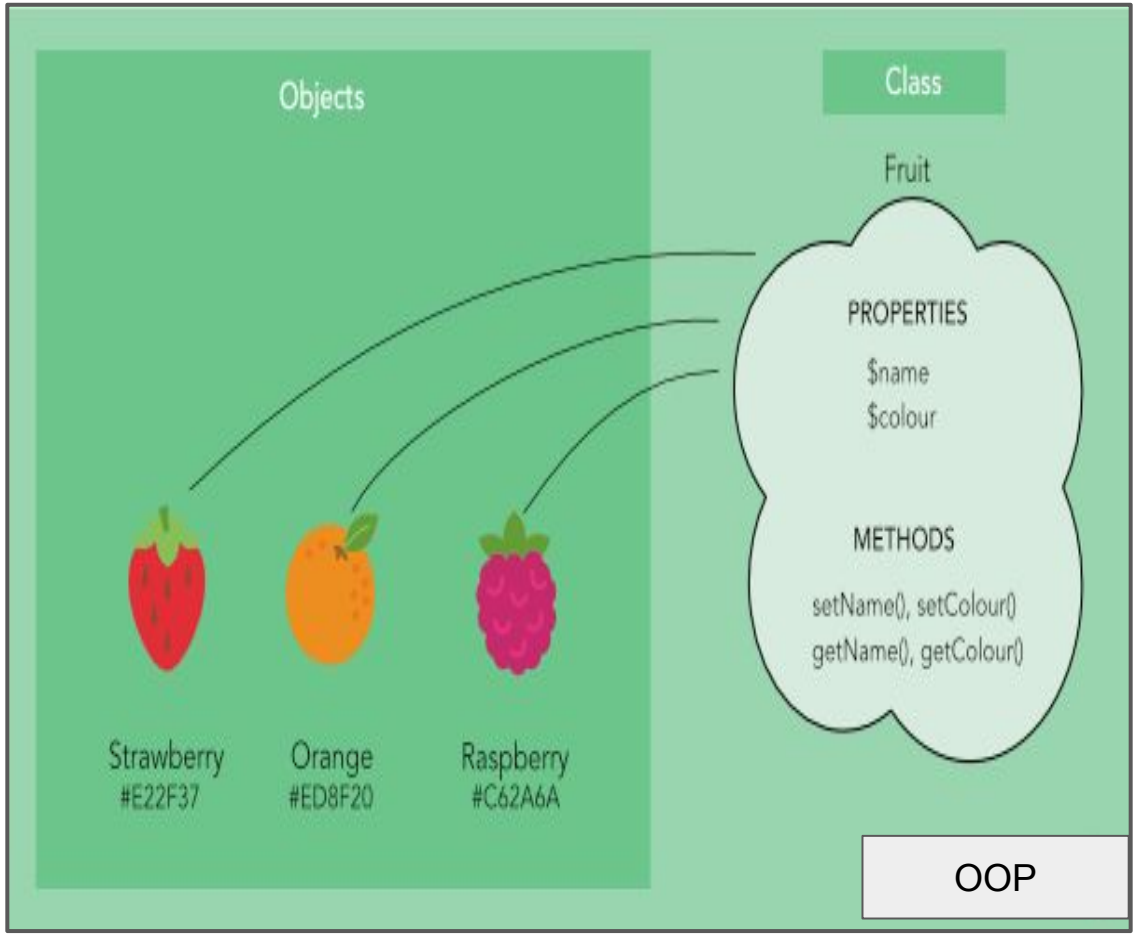


Figure 2 : Example of class & object

Encapsulation and Data-abstraction

Classes use the concept of data abstraction and encapsulation. Classes also known as ADT.

➢ **Encapsulation :** The wrapping up the data(variables/attributes) and associated functions(methods) into a single unit is known as encapsulation.

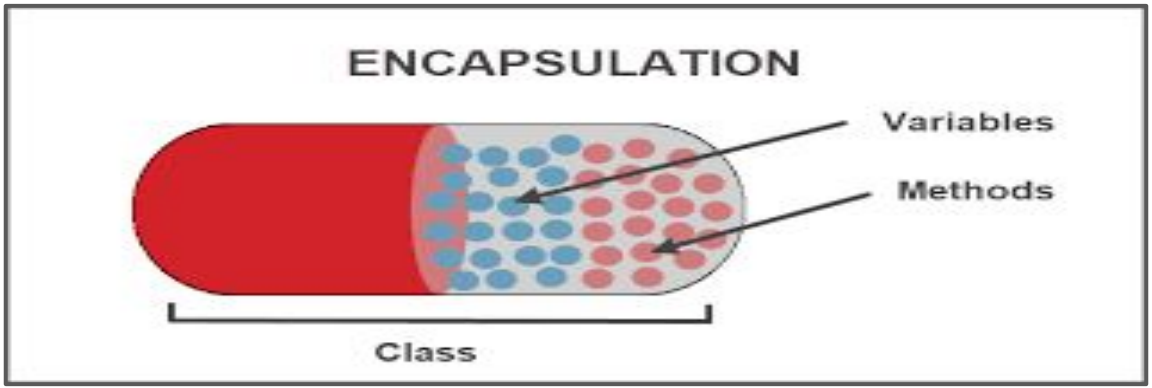


Figure 3 : Encapsulation

➢ **Data-abstraction :** Abstraction refers to the act of representing essential features without including details or background. Class helps us to group data members and member functions using available access specifiers. A Class can decide which data member will be visible to outside world and which is not.

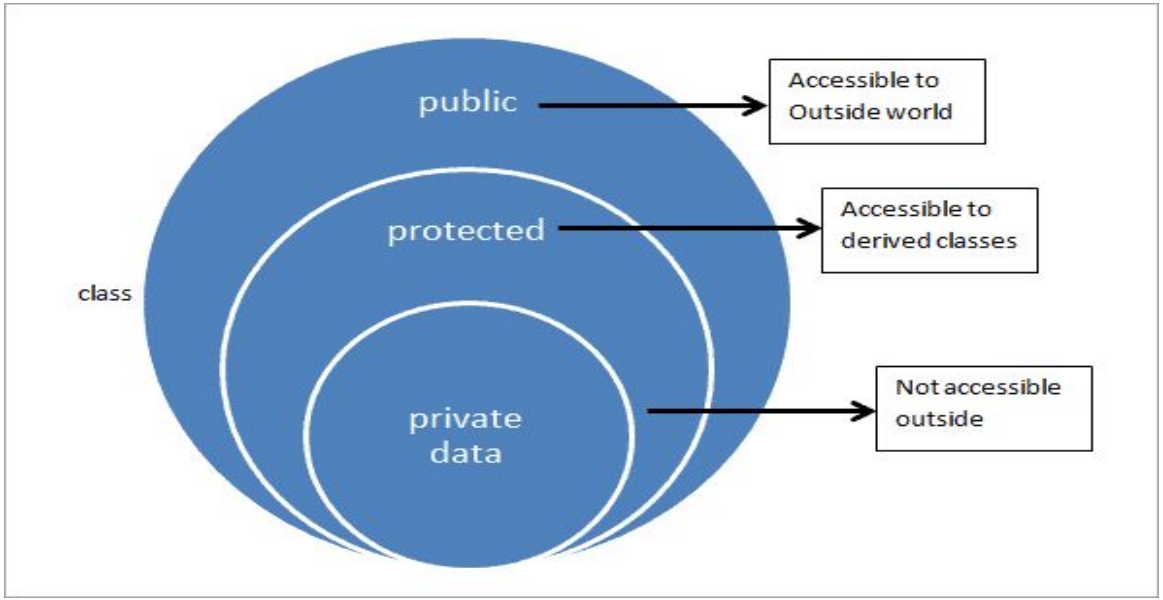


Figure 4 : Data Abstraction

Inheritance

Figure 5 : Example of Inheritance

- Inheritance is the process by which object of one class acquire the properties of objects of another class.It supports the concept of hierarchical classification.
- It provides the idea of reusability, this means that we can add extra features to an existing class without modifying it by deriving a new class from the class. The new class will have the combined features of both classes.

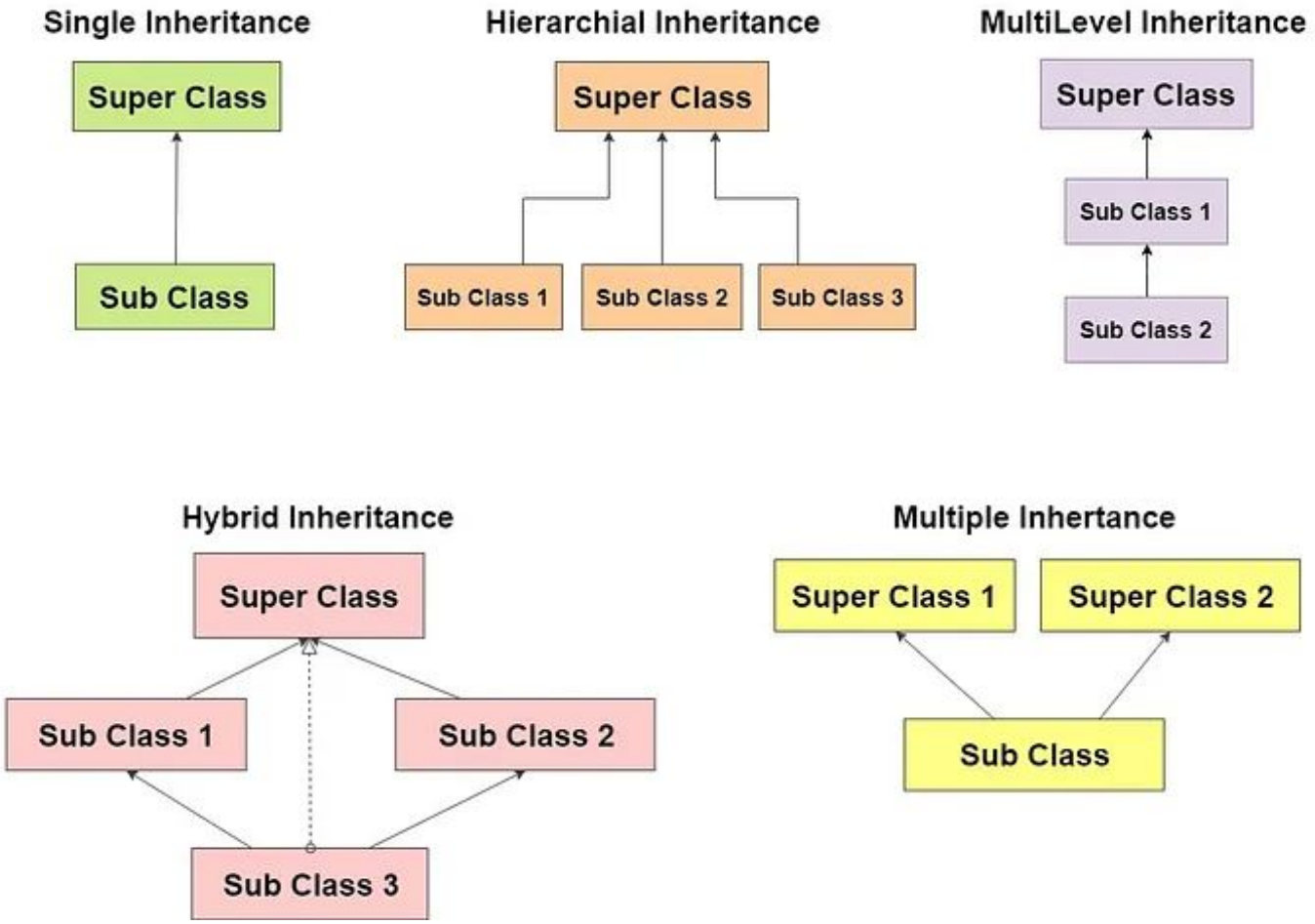


Figure 6 : Types of Inheritance

Polymorphism

- Polymorphism a greek term means, the ability to take more than one form. An operation may represent different behaviors in different instances.

- It allow objects having different internal structure to share the same external interface. This means that a general class of operations may be accessed in the same manner even though specific actions associated with each operation may differ.

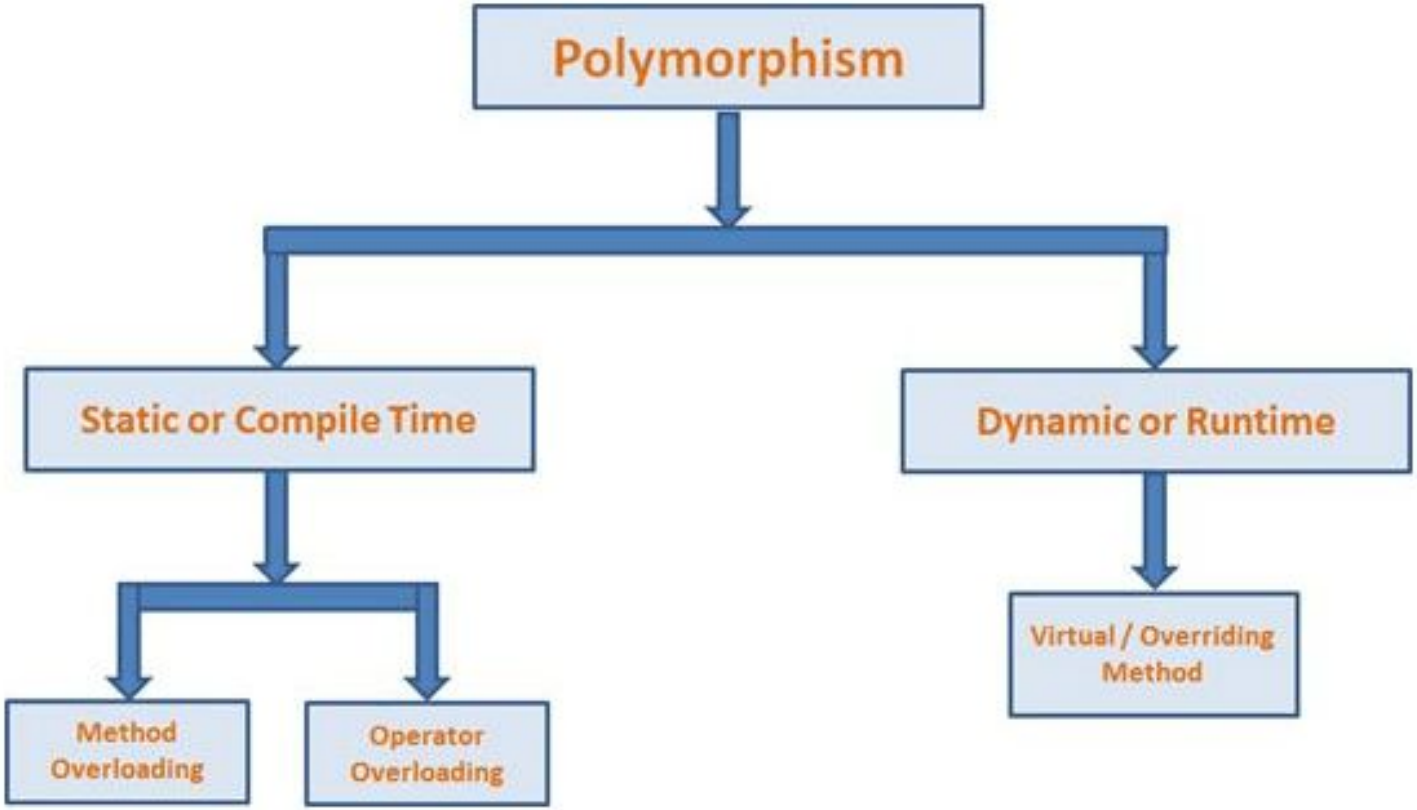


Figure 7 : Types of Polymorphism

Dynamic Binding & Message Passing

- **Dynamic Binding :** Dynamic Binding also known as **late binding** means that the code associated with a given procedure call is not known until the time of the call at run time. It is associated with the *inheritance and polymorphism*.
- **Message Passing :** A message for an object is a request for execution of a procedure(function) and therefore will invoke a function in the receiving order that generates the desired result. A message passing involves specifying the name of the object, name of the function(message) and the information to be sent.

Conclusion

- Programs are divided into what are known as objects
- Data structures are designed such that they characterize the objects
- Functions are operate on the data of an object are tied together in the data structure
- Data is hidden and can not be accessed by external functions
- Object may communicate with each other through functions
- New data and functions can be easily added whenever necessary
- Follows bottom-up approach in program design

END
of
Chapter 2

***Reference**:* E. Balaguruswamy; Object Oriented Programming with C++; *chapter-1, chapter-2, chapter-5;*