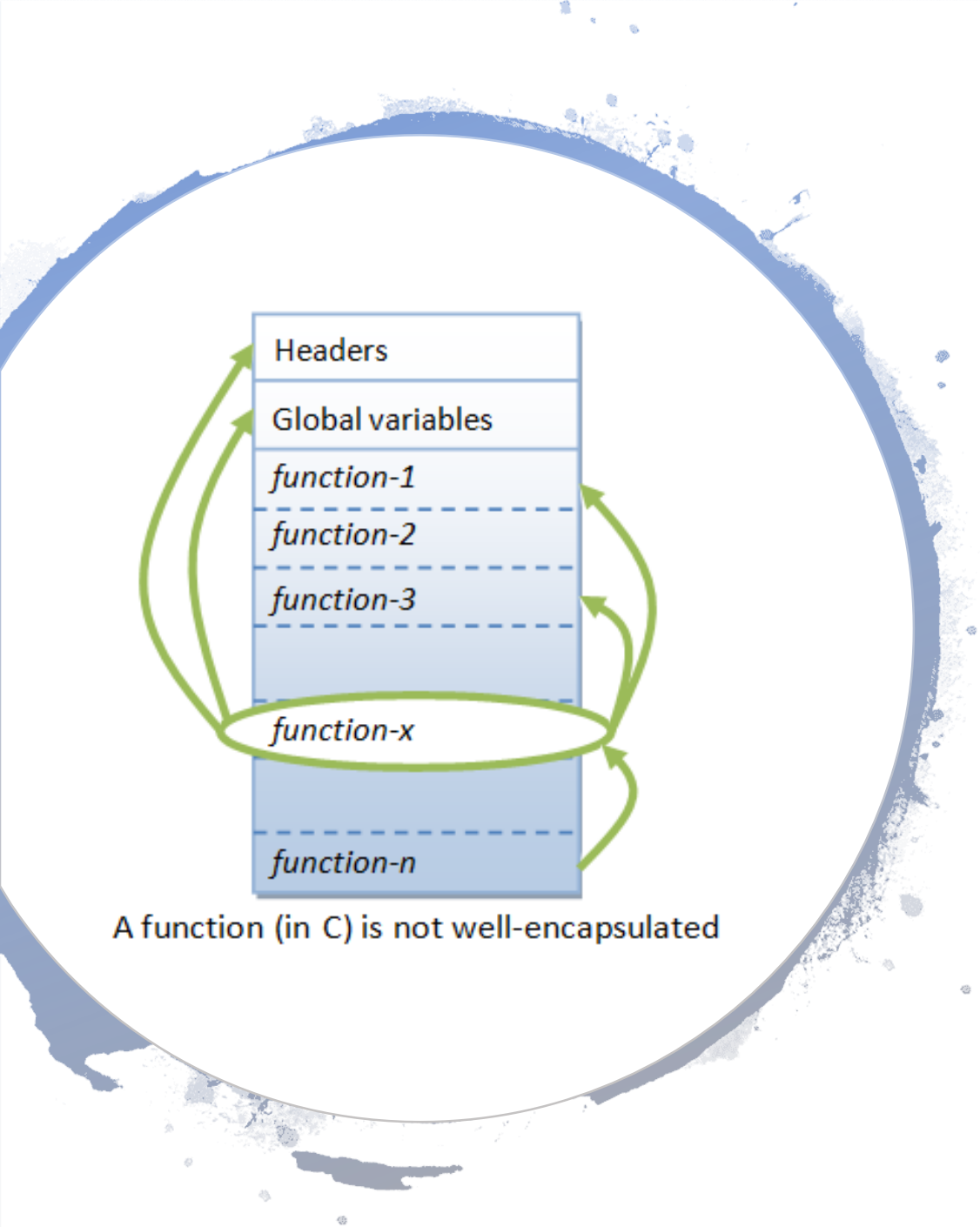
Object Oriented programming (OOP)


Before OOP



Procedural-Oriented languages

- Procedural programming is a programming paradigm
- derived from structured programming
- based upon the concept of the procedure call.
  - Procedures, also known as routines, subroutines, or functions, simply contain a series of computational steps to be carried out.
  - Any given procedure might be called at any point during a program's execution, including by other procedures or itself

Procedural-Oriented languages

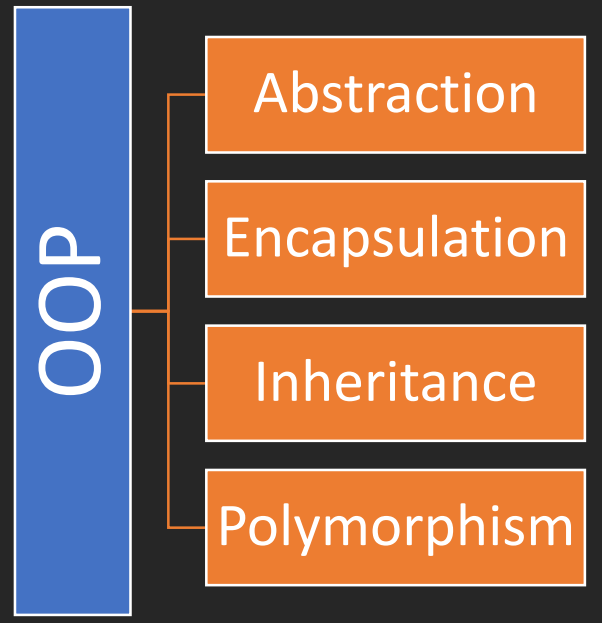
- Traditional procedural-oriented languages (such as C and Pascal) suffer some notable drawbacks in creating reusable software components:
  1. The programs are made up of functions. Functions are often not reusable.
     - It is very difficult to copy a function from one program and reuse in another program because the function is likely to reference the headers, global variables and other functions.
     - In other words, functions are not well-encapsulated as a self-contained reusable unit.
  2. The procedural languages are not suitable of high-level abstraction for solving real life problems.

     For example, C programs uses constructs such as if-else, for-loop, array, function, pointer, which are low-level and hard to abstract real problems such as a Customer Relationship Management (CRM) system or a computer soccer game. (Imagine using assembly codes, which is a very low level code, to write a computer soccer game. C is better but no much better.)

What OOP

- Object-oriented programming (OOP) is a software programming model/paradigm/style constructed around objects and associated with core concepts like;

1. Data hidden (Data Abstraction)
2. Encapsulation
3. Inheritance
4. Polymorphism



Features of OOP

- OOP features include the following:
  - Ease in software design as you could think in the problem space rather than the machine's bits and bytes. You are dealing with high-level concepts and abstractions. Ease in design leads to more productive software development.
  - Ease in software maintenance: object-oriented software are easier to understand, therefore easier to test, debug, and maintain.
  - Reusable software: you don't need to keep re-inventing the wheels and re-write the same functions for different situations. The fastest and safest way of developing a new application is to reuse existing codes - fully tested and proven codes.

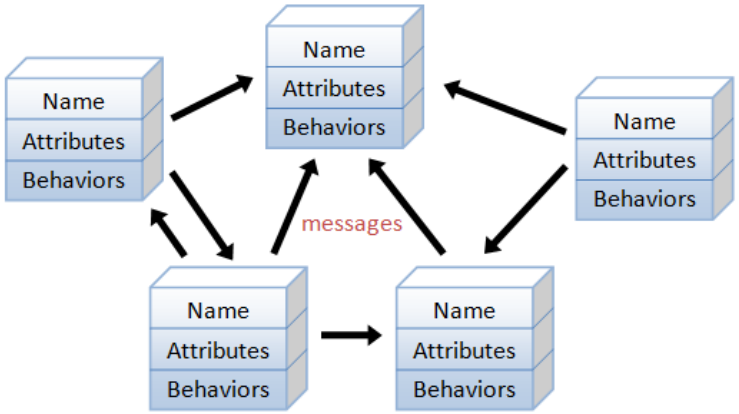
Why OOP

- Object-oriented programming (OOP) languages are designed to overcome these problems.

1. The basic unit of OOP is a class, which encapsulates both the static attributes and dynamic behaviors within a "box", and specifies the public interface for using these boxes.
   - Since the class is well-encapsulated (compared with the function), it is easier to reuse these classes.
   - In other words, OOP combines the data structures and algorithms of a software entity inside the same box.

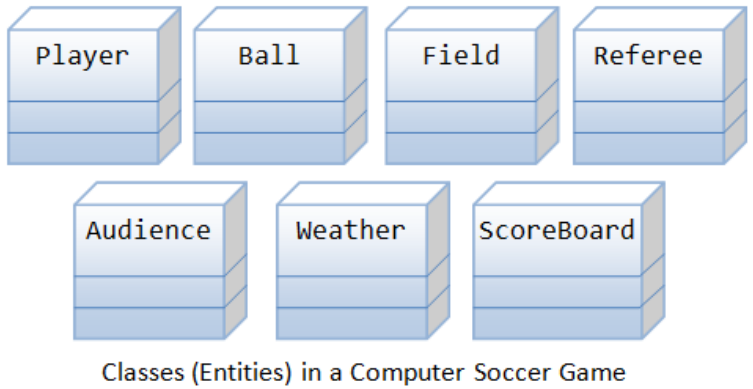
Why OOP

- OOP languages permit higher level of abstraction for solving real-life problems.
  - The traditional procedural language (such as C and Pascal) forces you to think in terms of the structure of the computer (e.g. memory bits and bytes, array, decision, loop) rather than thinking in terms of the problem you are trying to solve.
  - The OOP languages (such as Java, C++, C#) let you think in the problem space, and use software objects to represent and abstract entities of the problem space to solve the problem.



An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages

Example



Classes (Entities) in a Computer Soccer Game

- suppose you wish to write a computer soccer games. It is quite difficult to model the game in procedural-oriented languages. But using OOP languages, you can easily model the program accordingly to the "real things" appear in the soccer games.
  - Player:
    - attributes include name, number, location in the field, and etc;
    - operations include run, jump, kick-the-ball, and etc.
  - Ball:
  - Reference:
  - Field:
  - Audience:
  - Weather:
- Most importantly, some of these classes (such as Ball and Audience) can be reused in another application, e.g., computer basketball game, with little or no modification.

Procedure Oriented Programming vs. Object Oriented Programming

	Procedure Oriented Programming	Object Oriented Programming
Divided Into	In POP, program is divided into small parts called functions.	In OOP, program is divided into parts called objects.
Importance	In POP,Importance is not given to data but to functions as well as sequence of actions to be done.	In OOP, Importance is given to the data rather than procedures or functions because it works as a real world.
Approach	POP follows Top Down approach.	OOP follows Bottom Up approach.
Access Specifiers	POP does not have any access specifier.	OOP has access specifiers named Public, Private, Protected, etc.
Data Moving	In POP, Data can move freely from function to function in the system.	In OOP, objects can move and communicate with each other through member functions.
Expansion	To add new data and function in POP is not so easy.	OOP provides an easy way to add new data and function.
Data Access	In POP, Most function uses Global data for sharing that can be accessed freely from function to function in the system.	In OOP, data can not move easily from function to function,it can be kept public or private so we can control the access of data.
Data Hiding	POP does not have any proper way for hiding data so it is less secure.	OOP provides Data Hiding so provides more security.
Examples	Example of POP are : C, VB, FORTRAN, Pascal.	Example of OOP are : C++, JAVA, VB.NET, C#.NET.

- Encapsulation: This makes the program structure easier to manage because each object’s implementation and state are hidden behind well-defined boundaries.
- Polymorphism: This means abstract entities are implemented in multiple ways.
- Inheritance: This refers to the hierarchical arrangement of implementation fragments.