Object Oriented Programming

Unit I

Objective

- To introduce object oriented programming features and its diagrammatic representation of its model components
- To understand concept of class, handling its features and the reusability concept in object oriented language.
- To understand the mechanism to make use of files and standard libraries.
- To introduce the exception handling mechanism and the MVC architecture along with web components to design the software solution.
- To introduce how to perform the event driven programming.

Contents

- Introduction to object oriented programming paradigm
- procedure oriented programming vs OOP
- features of OOP
- benefits of OOP
- defining class
- instantiating a class.
- UML diagrams to represent class, objects and various relationships.

Skills for Software Engineers

Coding
Software Testing
Knowledge of tools

Technical Skills

Logical and analytical thinking

Problem Solving Skills

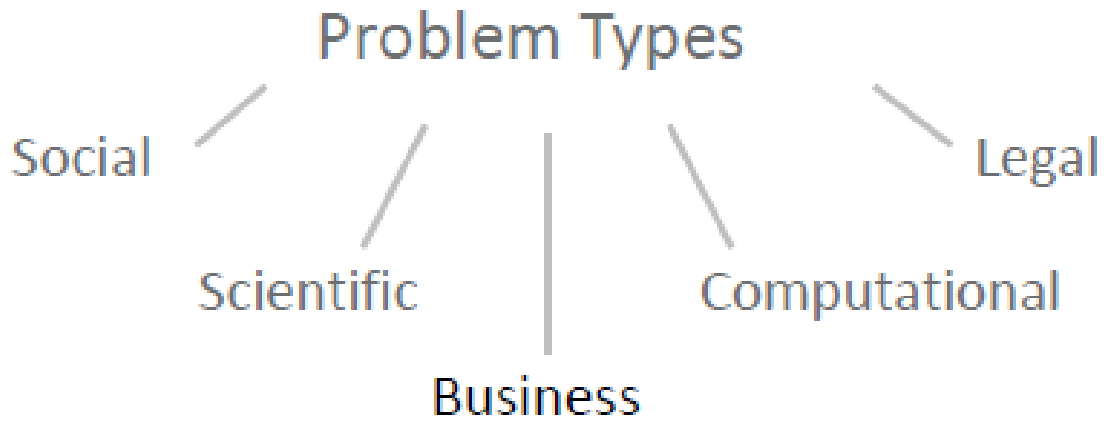
Communication
Teamwork

Soft Skills

What is a Problem?

A problem is a puzzle that requires logical thought or mathematics to solve it

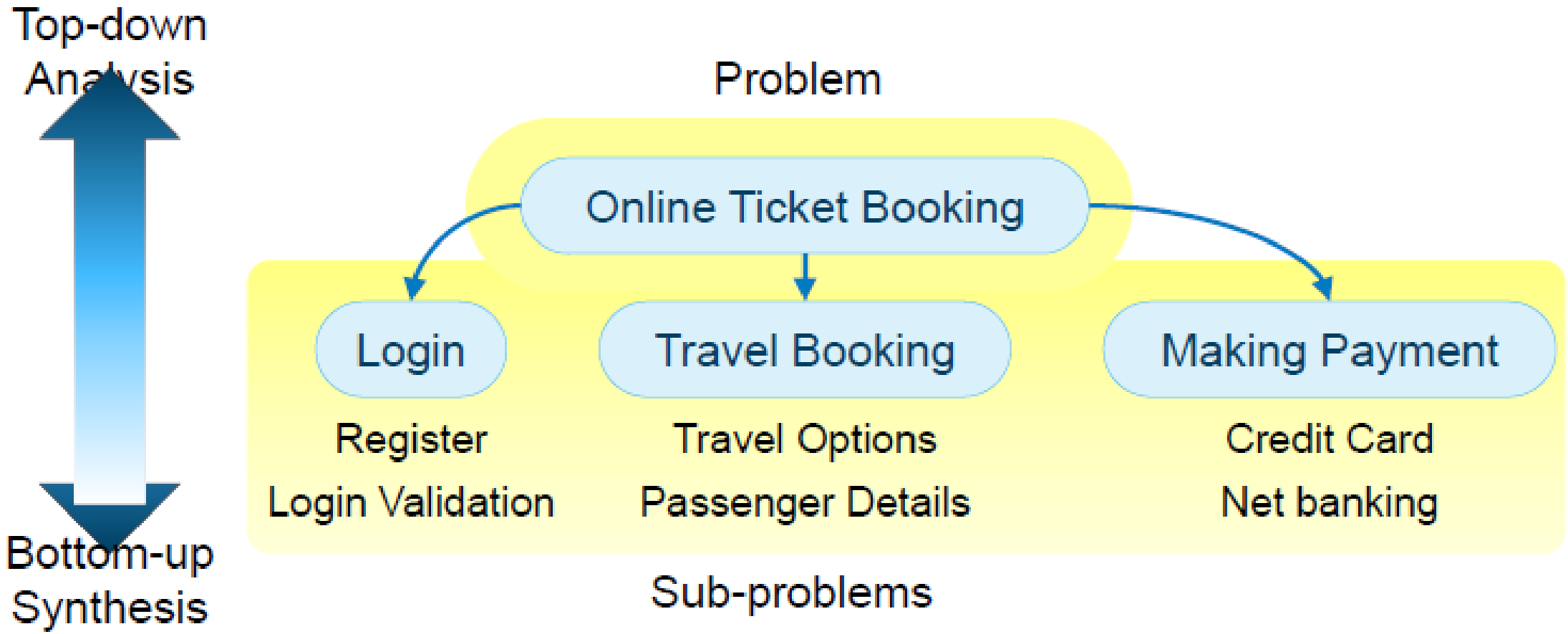
Source: Collins COBUILD Advanced Learner's English Dictionary



Computational Problem Solving

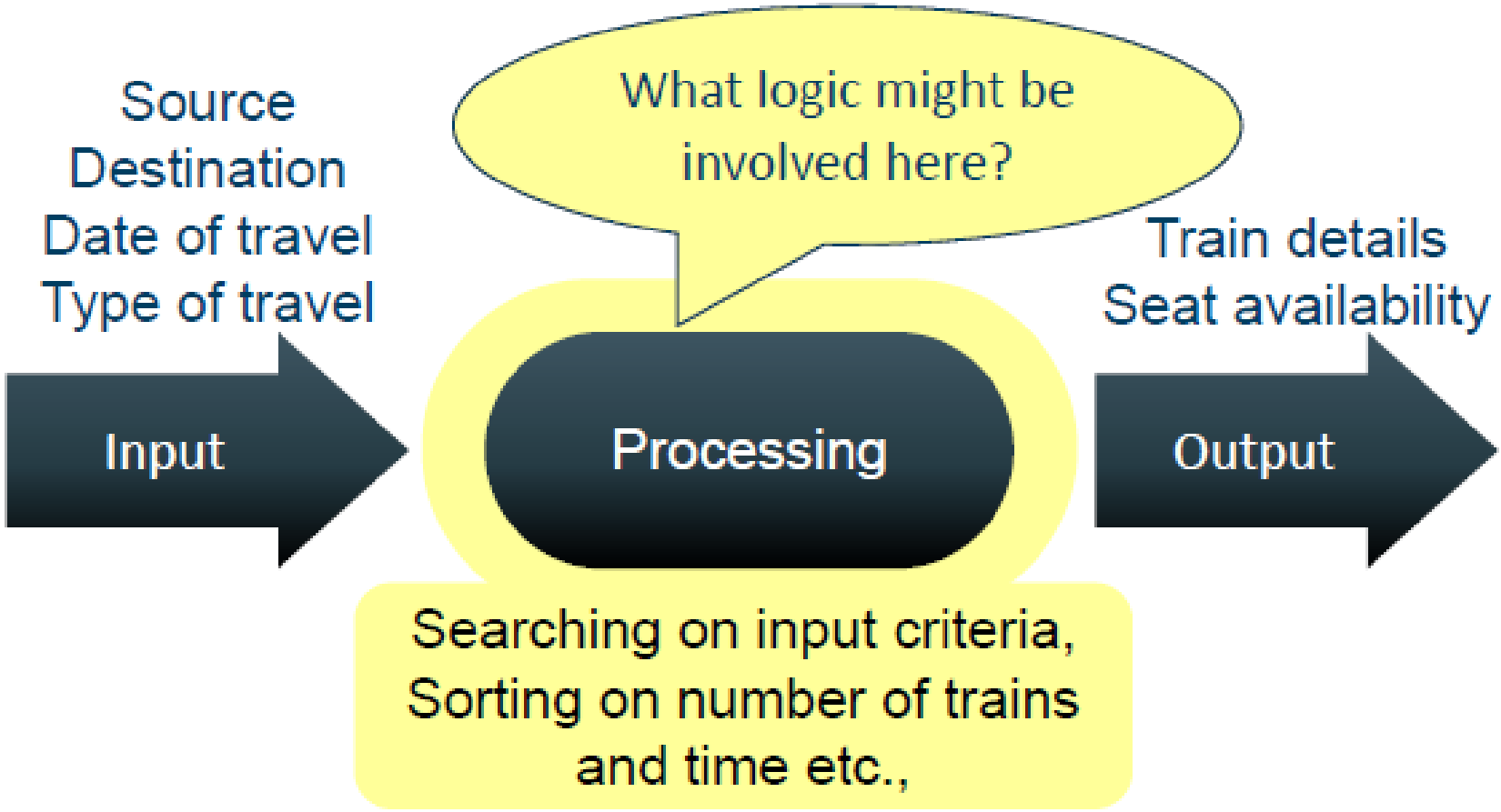
- Computers can be effectively used to solve many complex problems in all domains. However we will primarily focus on solving business problems
- Using computers to solve complex problems require computational thinking (way in which one can use computers to solve problems)
  - Computational thinking requires problem solving techniques that programmers use to analyze problems and design solutions

Solving complex business problem – Approach



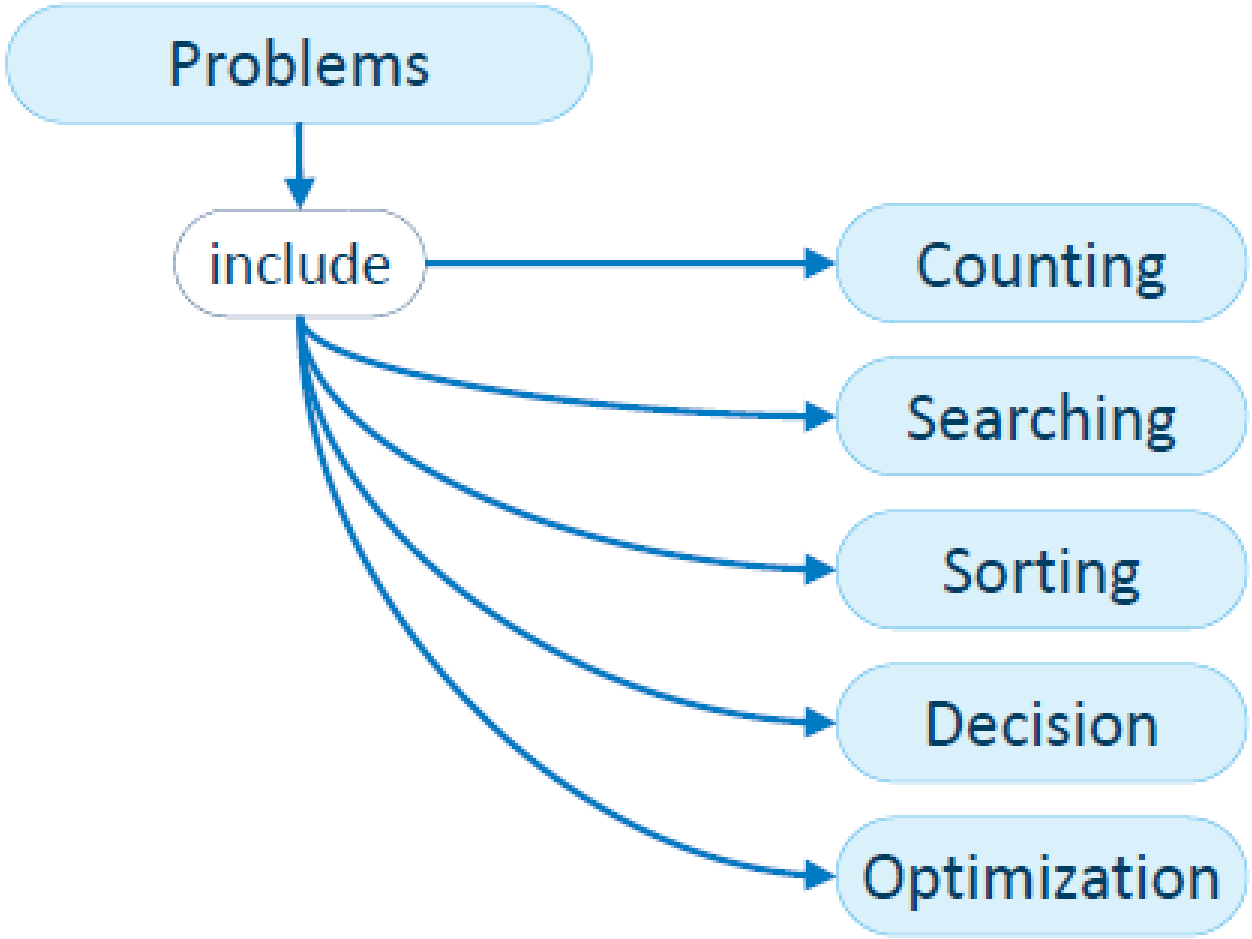
Solving complex business problem – Approach

Consider building the **Travel Options** module



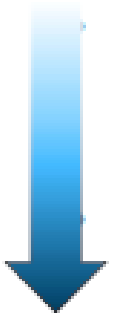
These are some fundamental problems encountered in real world problems

Frequently encountered problems



Analysis and Synthesis - Steps

Top-down Analysis

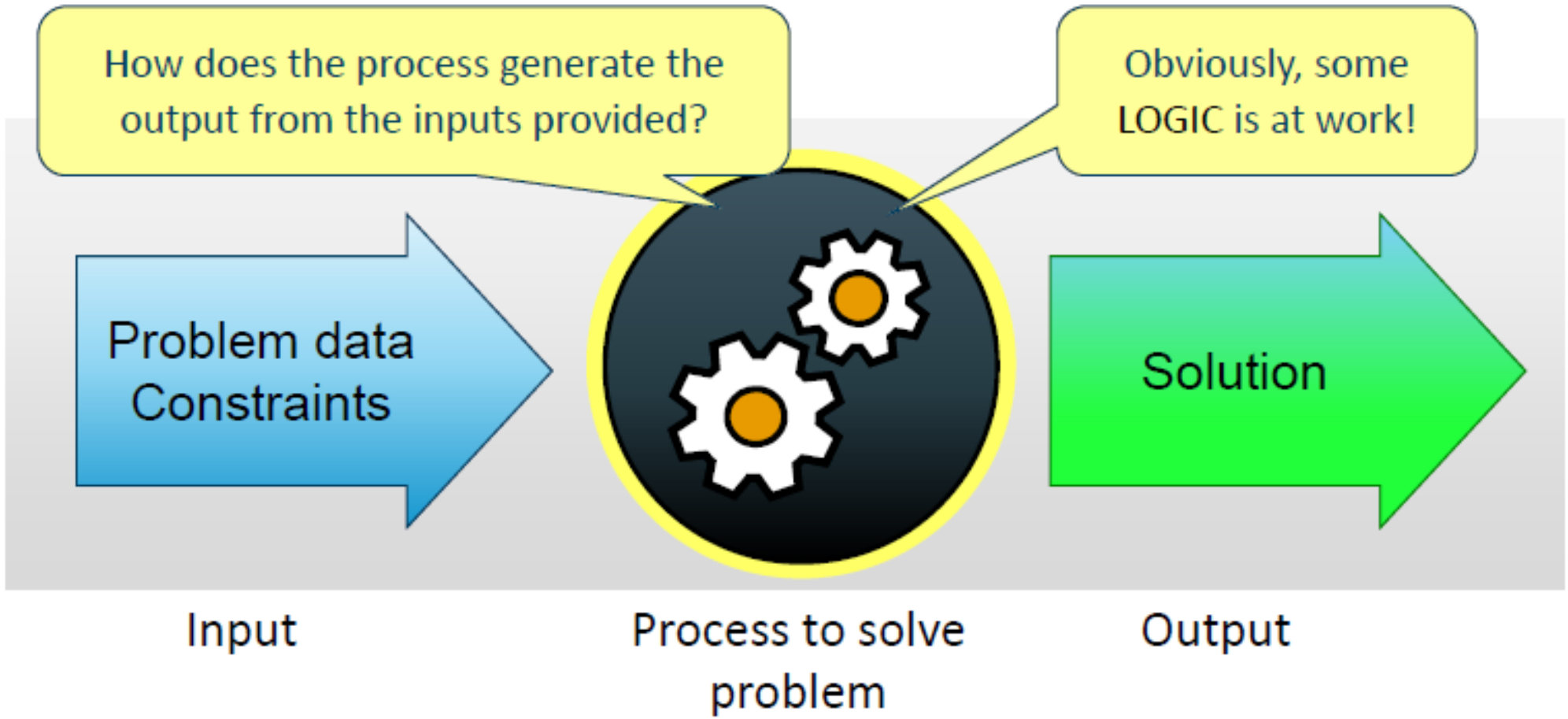


- The problem is broken down into smaller sub problems
- Each sub problem is successively refined till we recognize the solution

Bottom-up Synthesis

- Each sub problem solution is built as a unit or module
- Modules are successively put together to make the entire solution

Problem solving requires identification and understanding of:



Logic

Logic is a method of reasoning that involves a series of statements, each of which must be true, if the statement before it is true.

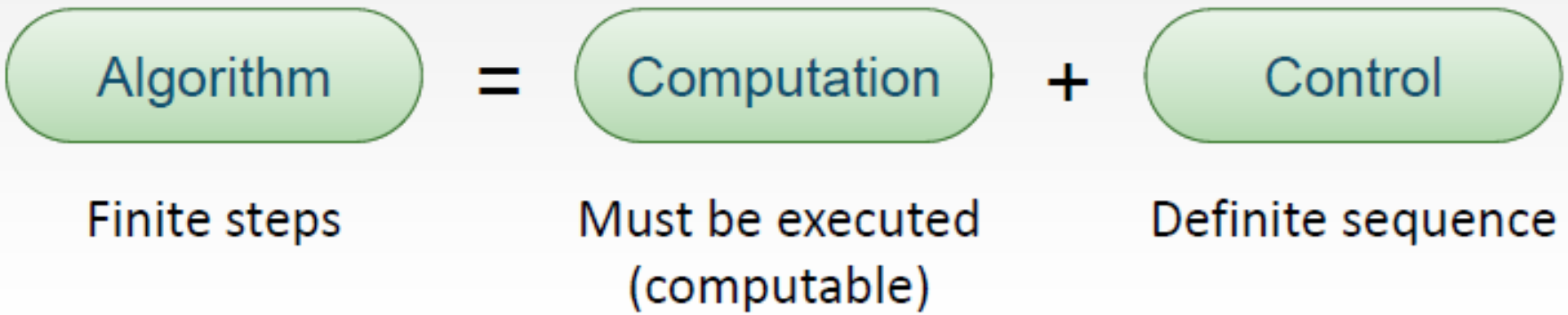
Source : Collins COBUILD Advanced Learner's English Dictionary

- Logic requires thinking in a step by step manner about how a problem can be solved
- Logic is a mental model required to solve a problem
- The logic is represented as an ALGORITHM

Algorithms - Definition

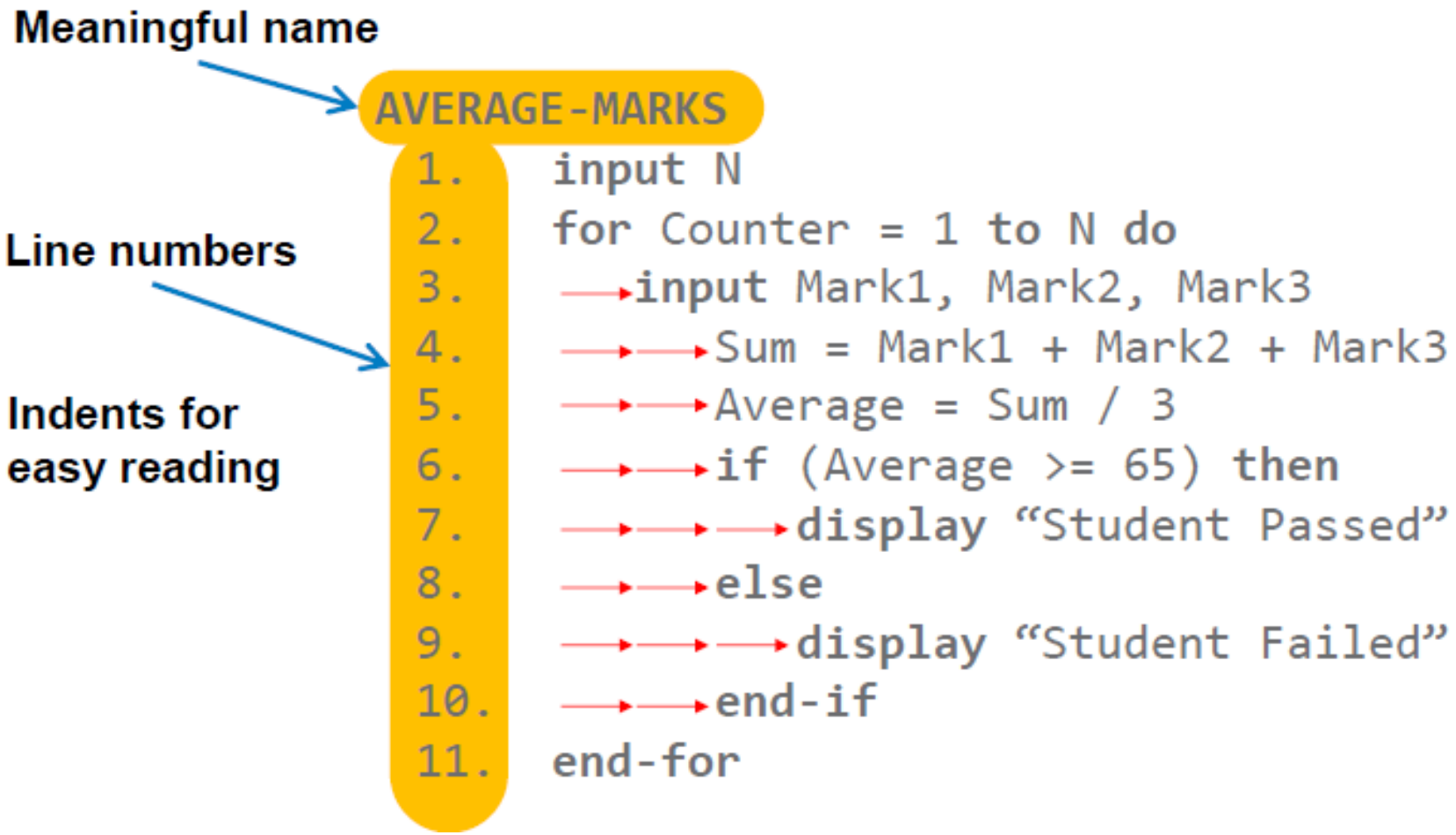
Demo: Algorithms – Assignment 2

Algorithm is a **finite set of steps** to accomplish a task / solve a problem. The steps **must be executed** in some **definite sequence** to achieve the expected result.

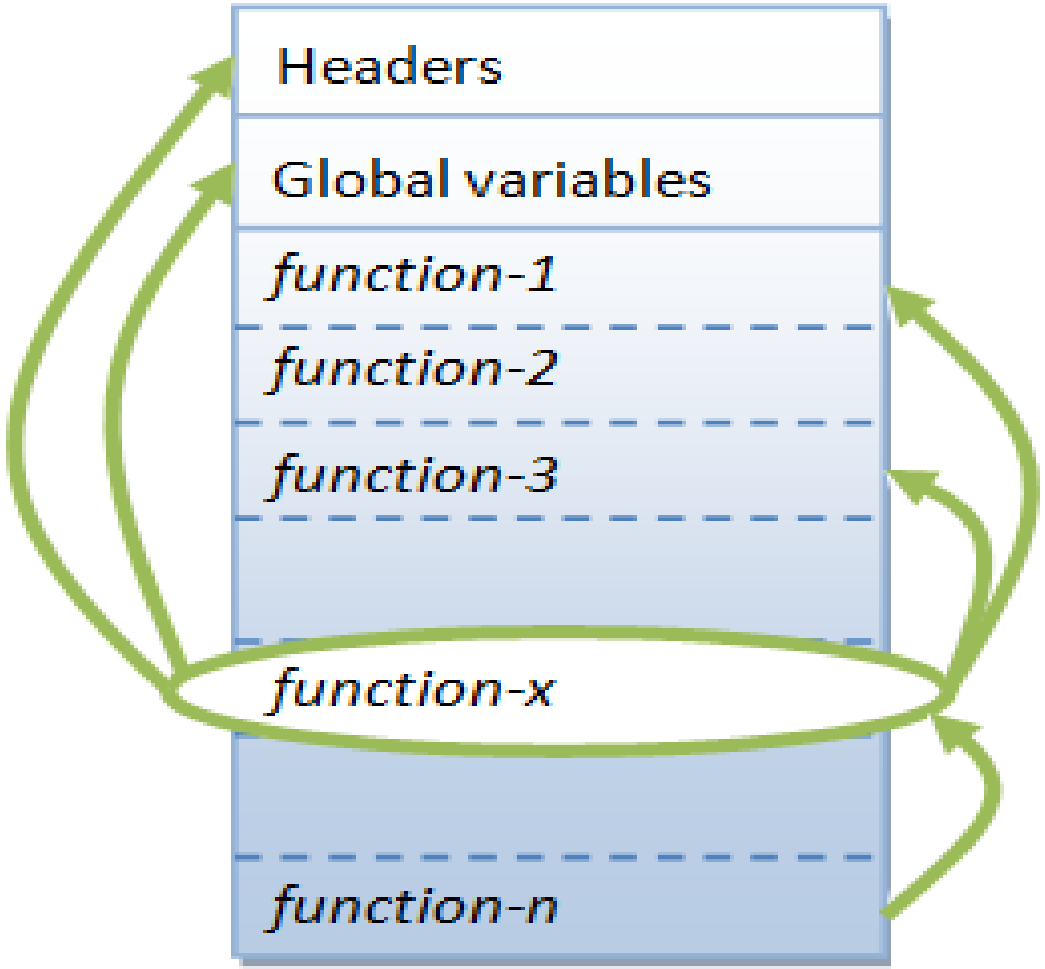


Pseudo-code conventions

Pseudo code to compute average marks scored by trainees in 3 modules



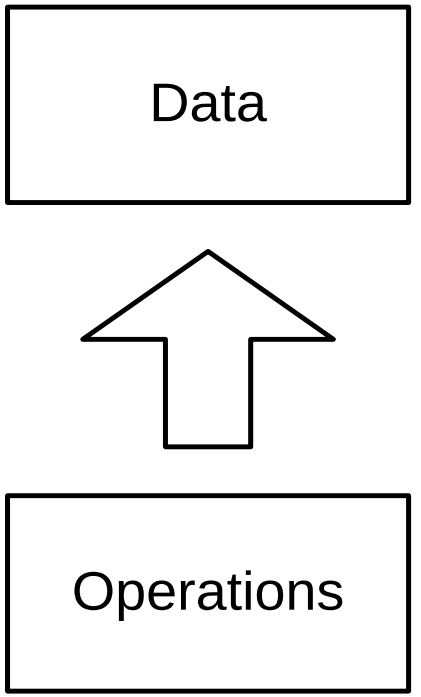
Procedural – oriented languages



A function (in C) is not well-encapsulated

The Procedural Paradigm

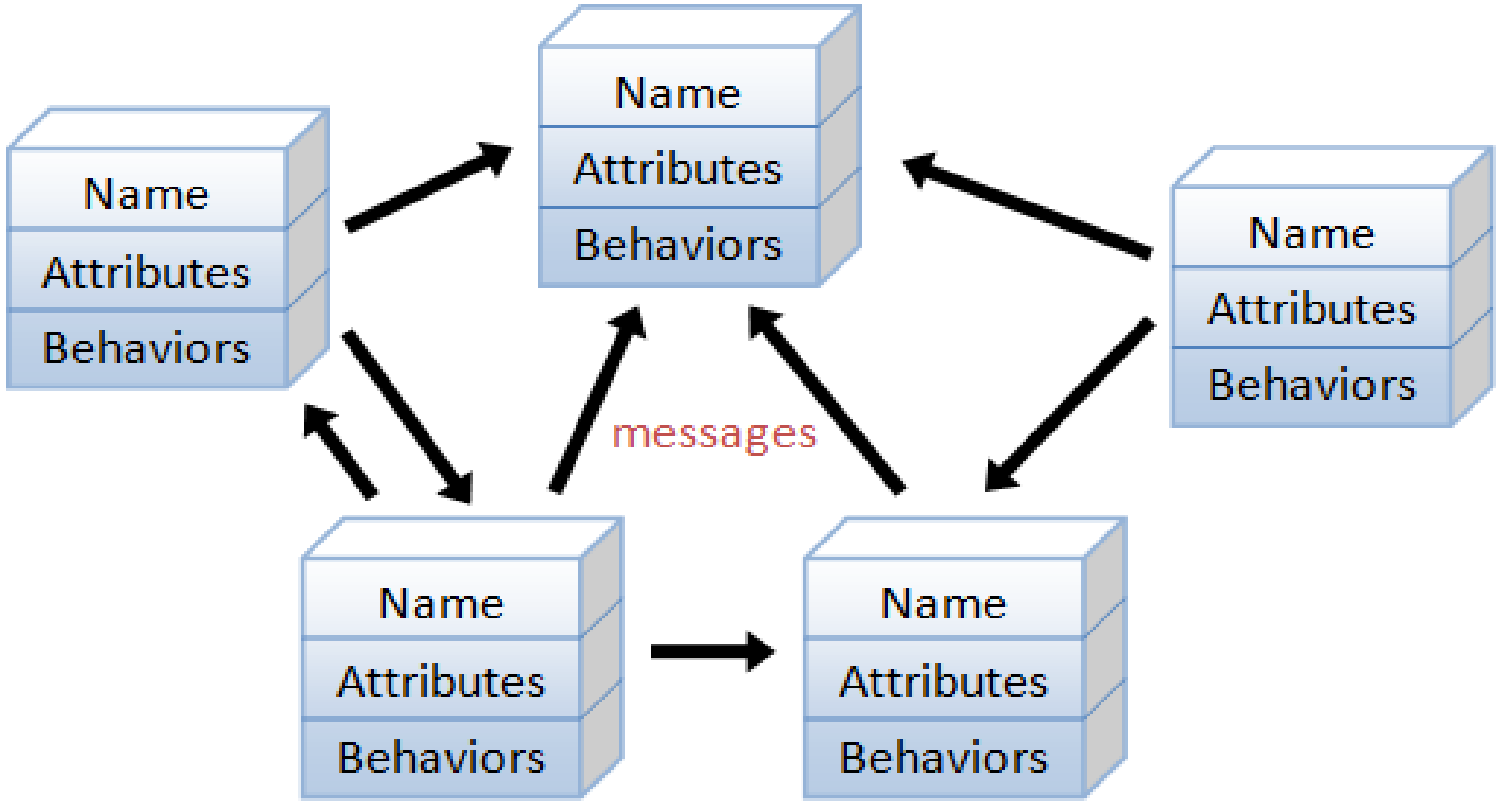
- Operations are separate from data but manipulate it.
- Example: A stack data type
  - Data store (array, linked list)
  - Separate operations for pushing, popping, etc.



Procedural programming

- Emphasis is on doing things
- Large programs are divided into smaller programs known as functions
- Most of the functions share global data
- Data move openly around the system from function to function
- Functions transform data from one form to another
- Employs top-down approach in program design

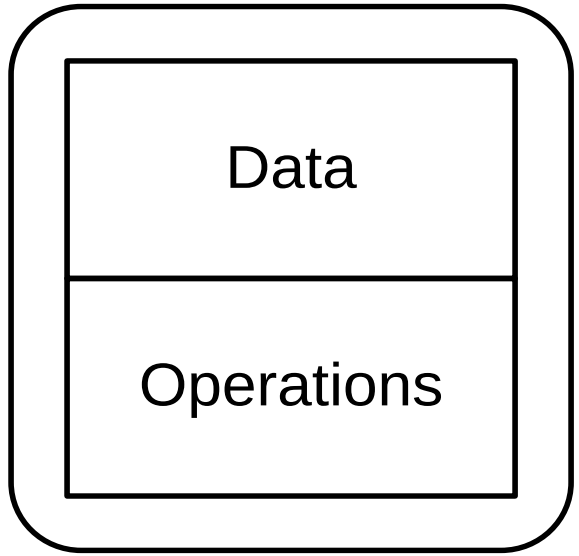
Object oriented language



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

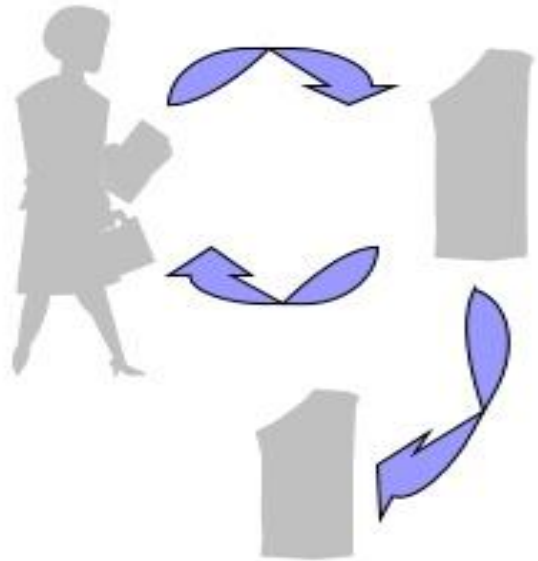
The Object-Oriented Paradigm

- Data is packaged (encapsulated) with the operations that manipulate it.
- Example: a stack object
  - Data store within a stack object
  - Operations within the stack object



Procedural vs. Object-Oriented

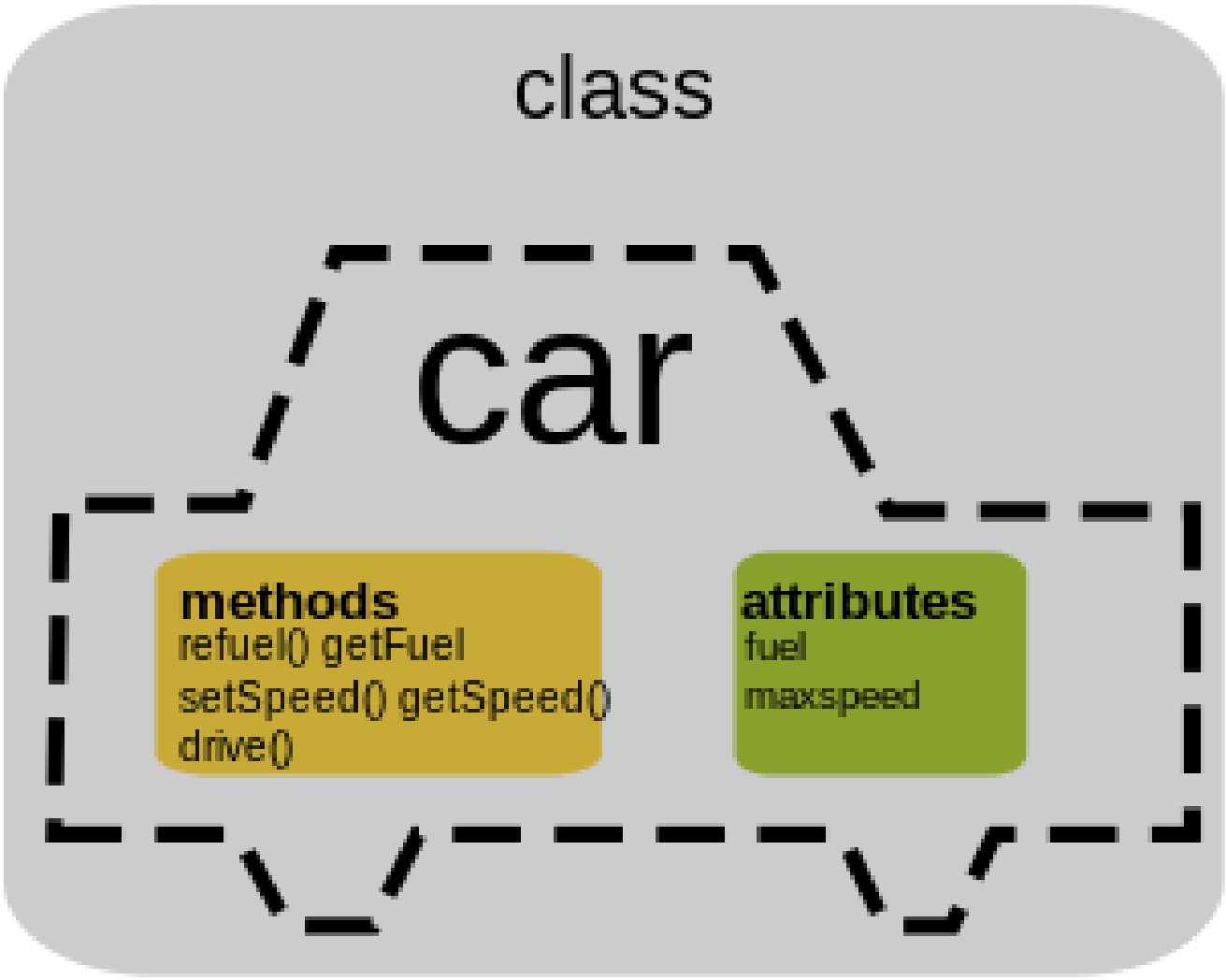
- Procedural



Withdraw, deposit, transfer

- Object Oriented

Customer, money, account


class
car
methods
refuel() getFuel
setSpeed() getSpeed()
drive()
attributes
fuel
maxspeed

Features of Object oriented programming

- Oop treats data as a critical element
- It ties data more closely to the functions that operate on it
- Data and functions are build around the objects
- Data structures are designed such that they characterize the objects
- Data is hidden and can not be accessed by external functions
- New data and functions can be easily added whenever necessary

Problems with the OO Paradigm

- It’s a lot more complicated than the procedural paradigm.
  - Harder to learn
  - Harder to use properly

Objects

An **object** is an entity that has properties and exhibits behavior.

- Things are *encapsulated* when they are packaged together in a container.
- Objects encapsulate properties and behavior.
- Ideally, objects model things in the world.
  - Example: Traffic simulation

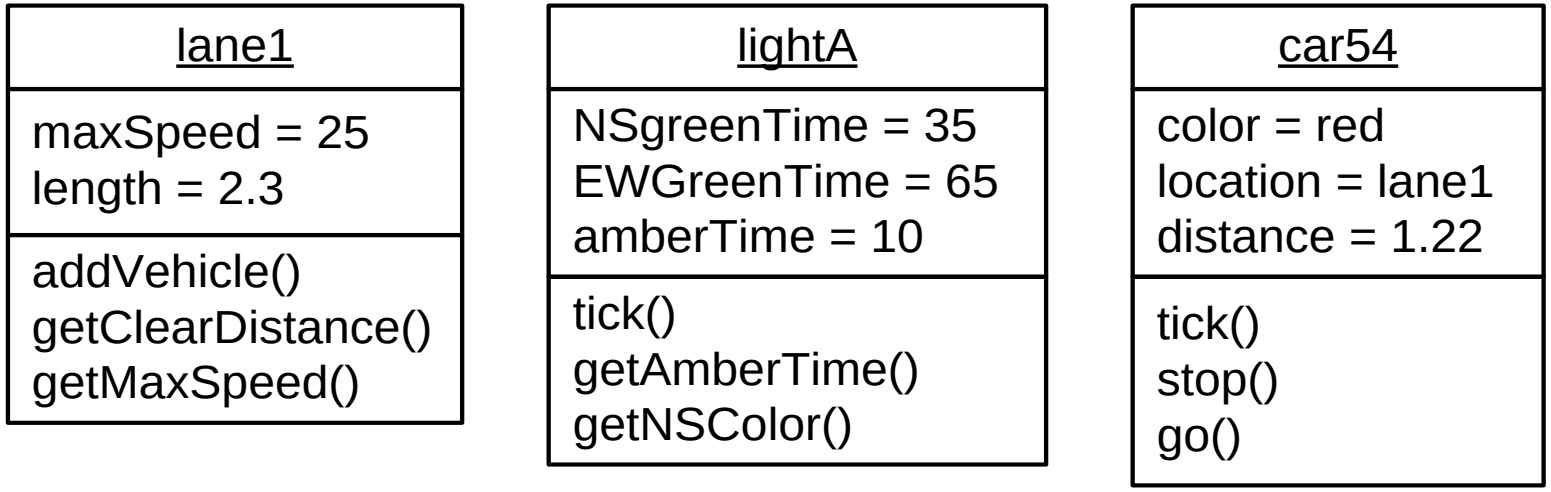
Attributes and Operations

An **attribute** is a named data value held by an object or a class.

An **operation** is a named object or class behavior.

A **feature** is an attribute or operation.

Attribute and Operation Examples



Object Characteristics

- Every object has its own **attributes** and **operations**, though these may be the same as some other objects.
- Every object has a **unique identity**.
  - Two objects with the same attributes holding the same values and behaviors are distinct.

Messages

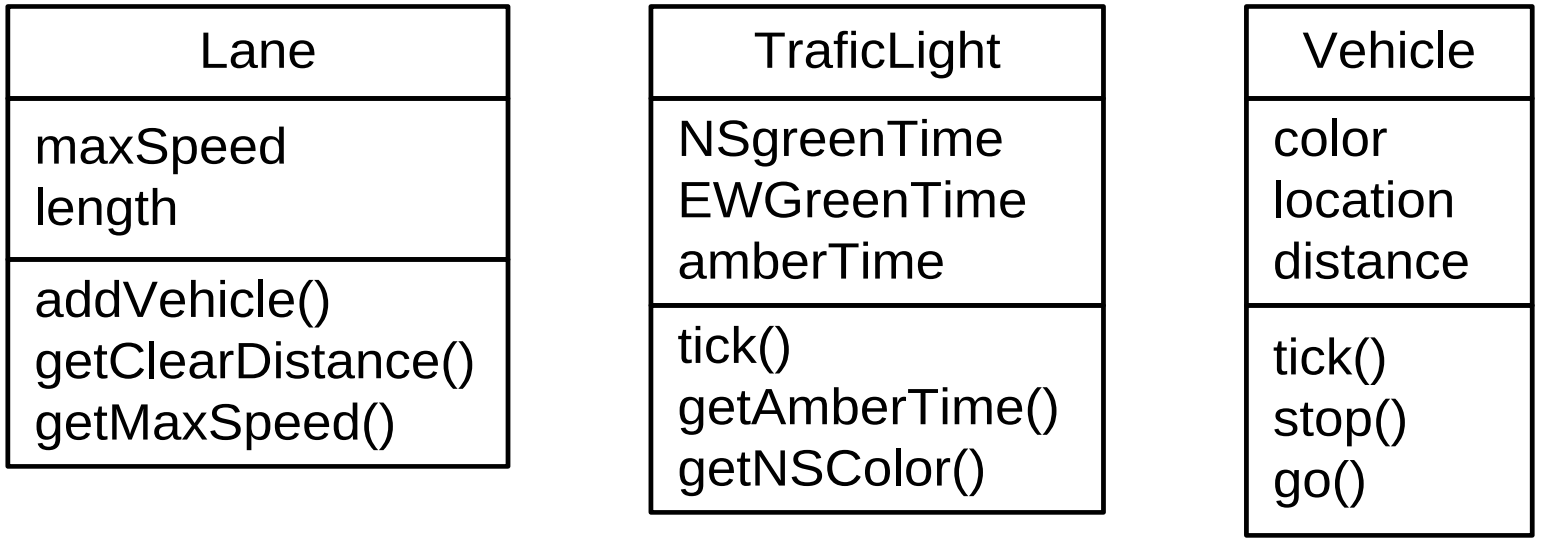
- Objects communicate by sending one another messages.
- Messages are abstractions of function calls, signals, exceptions, etc.
- Ideally, objects send messages that model real-world communications between things.
  - Example: Traffic simulation

Classes

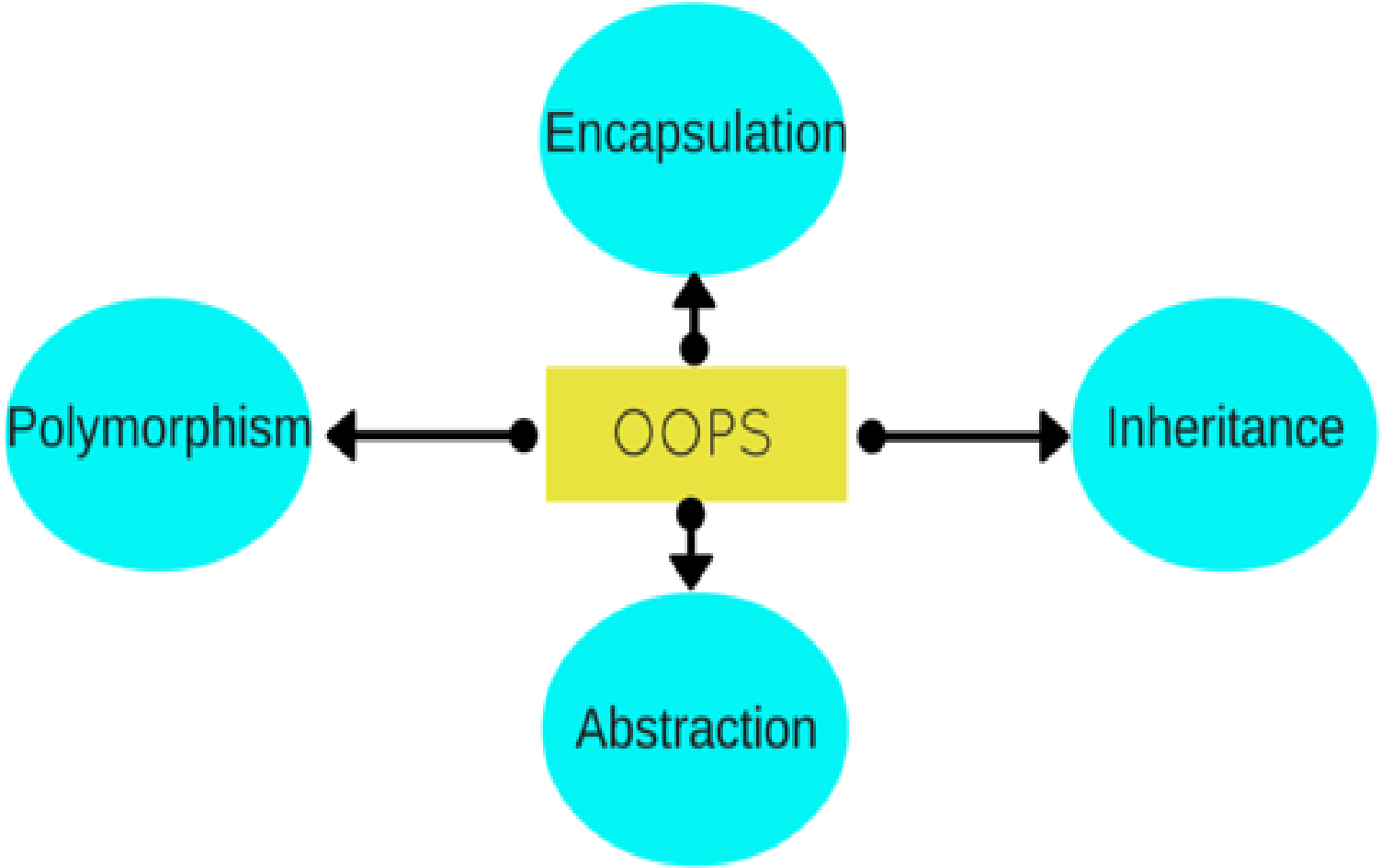
An **class** is an abstraction of a set of objects with common attributes and operations.

- Classes have attributes and operations.
- **A class is not a set of objects—it is more like a blueprint or a specification**.
- Objects are said to be *instances* of classes.

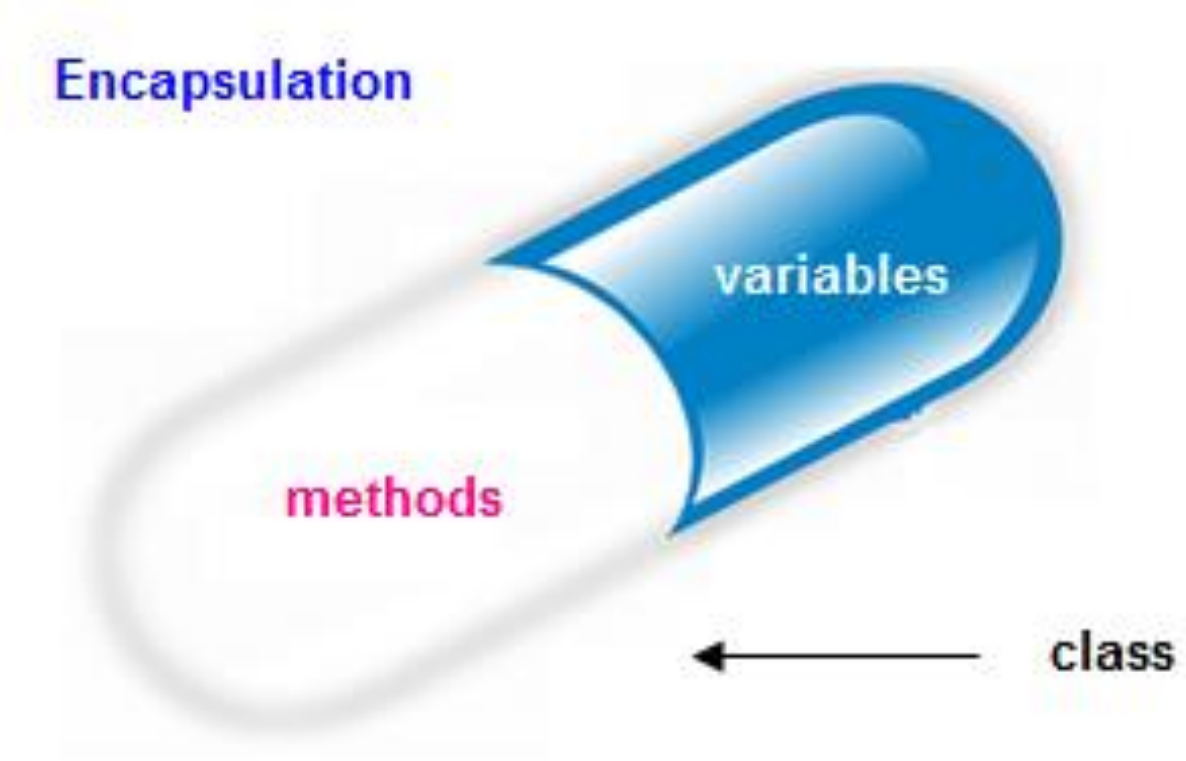
Examples of Classes



Object oriented programming



Encapsulation?



Encapsulation

- Hide the data from outside world
- In c++, encapsulation achieved using the access specifiers (private, public and protected)
- By encapsulation we can preventing direct access from outside, and thus have complete control, protection and integrity of the data.

Encapsulation Advantages 1

- *Problem and solution similarity*—Software objects are like real-world things.
- *Abstraction*—Data structures and algorithm details are ignored in design.
- *Information hiding*—Implementations can be truly hidden.

  (Shielding the internal details of a program unit's processing from other program units)

Encapsulation Advantages 2

- *Coupling*—Operations are not related through shared data structures.
  (The degree of connection between pairs of modules)
- *Cohesion*—It's easier to keep related operations and data together.
  (The degree to which a module's parts are related to one another)

Encapsulation Advantages 3

- *Reusability*—It's easier to reuse objects than collections of operations and various data structures.
- *Scalability*—Objects can be of arbitrary size.
- *New features*—The object-oriented paradigm offers new and powerful ways of doing things, as we will see.

Abstraction?

Data Abstraction

- Data abstraction refers to hiding the internal implementations and show only the necessary details to the outside world.
- In C++ data abstraction is implemented using interfaces and abstract classes.

Inheritance

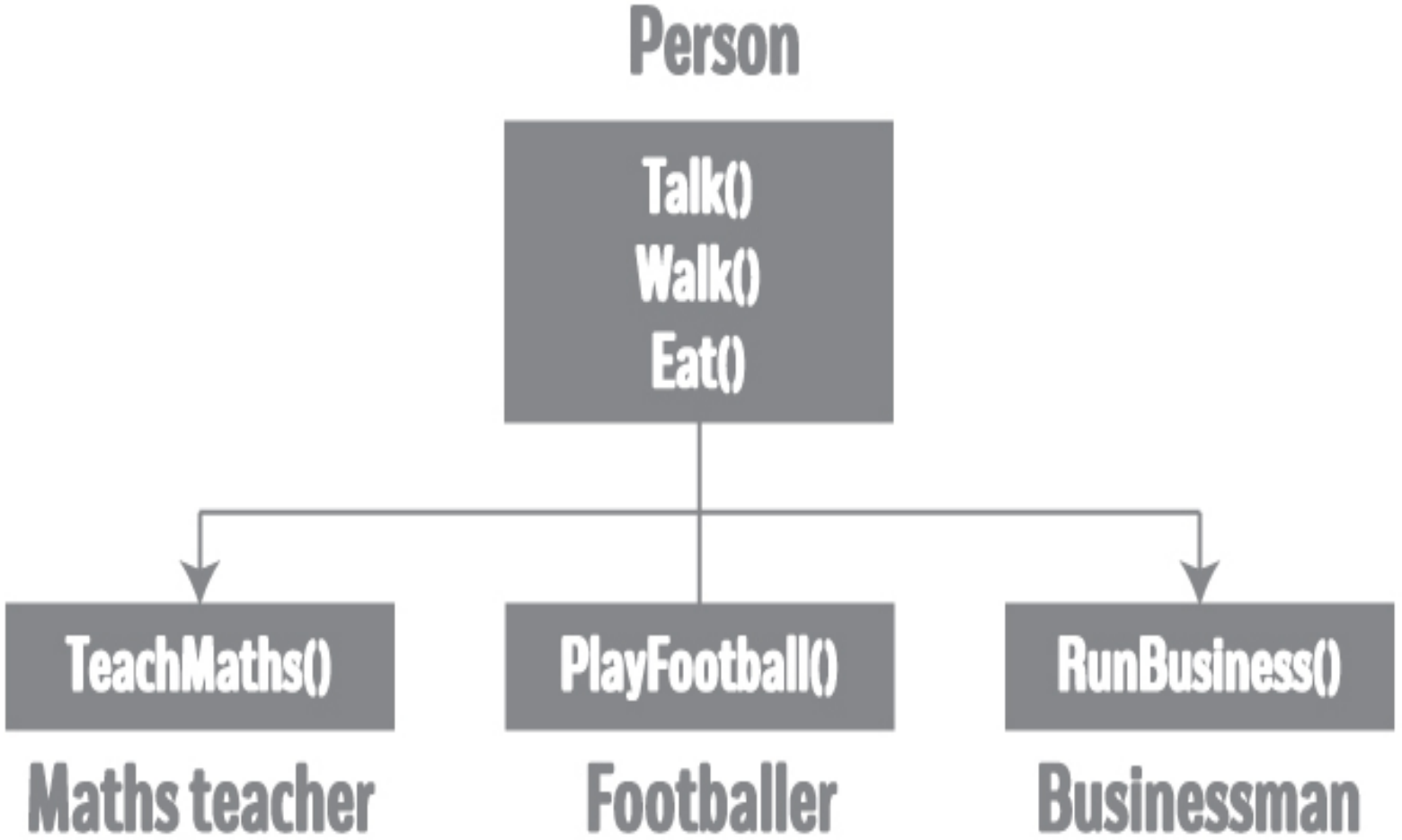
Why inheritance should be used?

- Suppose, in your game, you want three characters - a **maths teacher**, a **footballer** and a **businessman**.
- Since, all of the characters are persons, they can walk and talk. However, they also have some special skills. A maths teacher can **teach maths**, a footballer can **play football** and a businessman can **run a business**.
- You can individually create three classes who can walk, talk and perform their special skill

Why inheritance should be used?

- In each of the classes, you would be copying the same code for walk and talk for each character.
- If you want to add a new feature - eat, you need to implement the same code for each character. This can easily become error prone (when copying) and duplicate codes.

Solution?


Person
Talk()
Walk()
Eat()
TeachMaths()
PlayFootball()
RunBusiness()
Maths teacher
Footballer
Businessman

Polymorphism

- The process of representing one Form in multiple forms is known as **Polymorphism**.
- Original form or original method always resides in base class and multiple forms represents overridden method which resides in derived classes.
- Typically, polymorphism occurs when there is a hierarchy of classes and they are related by inheritance

Example of polymorphism

In Shopping malls behave like Customer

In Bus behave like Passenger

In School behave like Student

At Home behave like Son

Polymorphism in programming

- In programming languages, *polymorphism* means that some code or operations or objects behave differently in different contexts
- For Example, the +(plus) operator in C++
  - 4 + 5 ← Integer addition
  - 3.14 + 2.0 ← Floating point addition
  - S1 + “bar” ← String concatenation

Advantages of OOP

- ***Code Reuse and Recycling***: Objects created for Object Oriented Programs can easily be reused in other programs.
- ***Encapsulation (part 1)***: Once an Object is created, knowledge of its implementation is not necessary for its use. In older programs, coders needed understand the details of a piece of code before using it
- ***Encapsulation (part 2)***: Objects have the ability to hide certain parts of themselves from programmers. This prevents programmers from tampering with values they shouldn't.
- ***Design Benefits***: Large programs are very difficult to write. Object Oriented Programs force designers to go through an extensive planning phase, which makes for better designs with less flaws.
- ***Software Maintenance:*** Programs are not disposable. Legacy code must be dealt with on a daily basis, either to be improved upon (for a new version of an exist piece of software) or made to work with newer computers and software.

Procedure oriented programming VS OOP

	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.
Overloading	In POP, Overloading is not possible.	In OOP, overloading is possible in the form of Function Overloading and Operator Overloading.
Examples	Example of POP are : C, VB, FORTRAN, Pascal.	Example of OOP are : C++, JAVA, VB.NET, C#.NET.