Coding and Logic

Be able to recognise features and benefits of various language types.

Scripting

- Windows Batch files
  - Executed by cmd.exe
  - Outputs text
  - String based
- Windows Script Files
  - Executed by PowerShell.exe
  - Based on .NET framework
  - Outputs objects
  - Object orientated

Algorithm

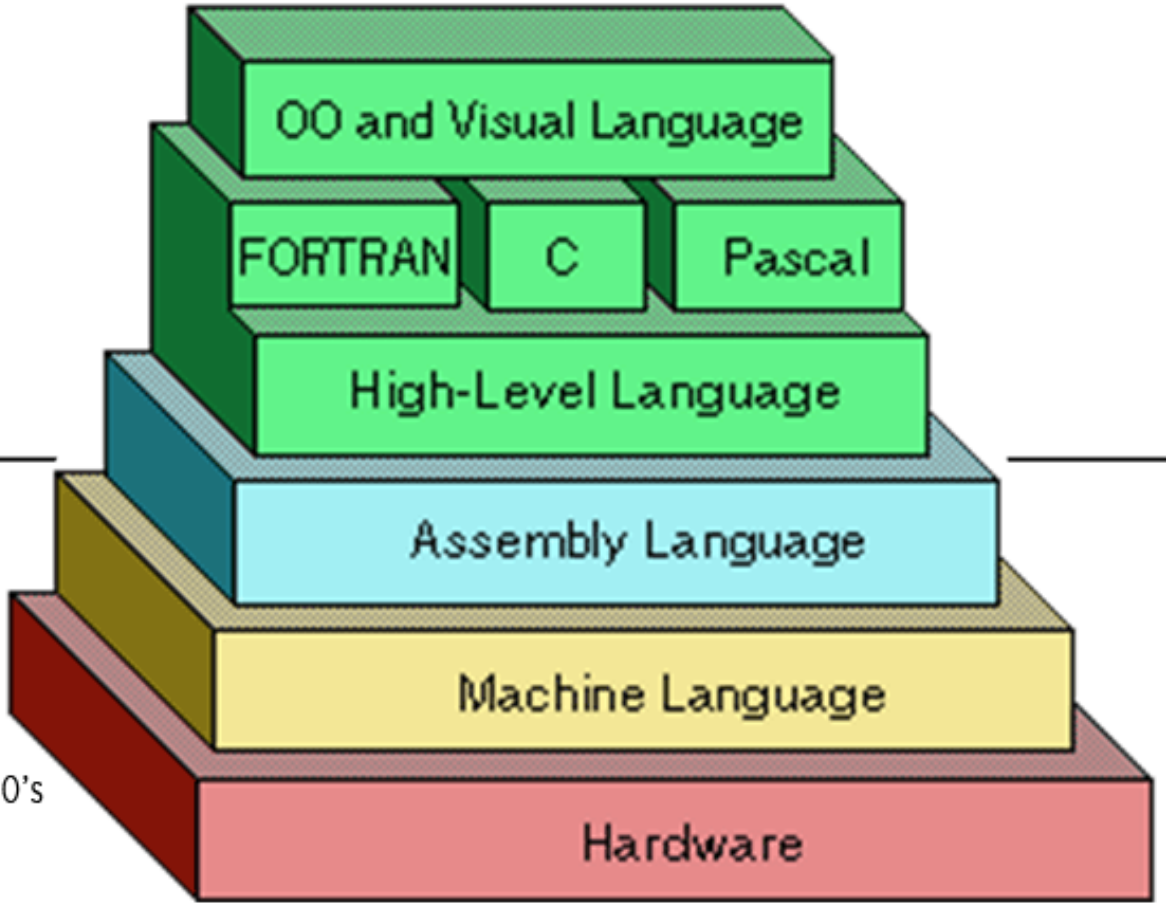
- A list of rules to follow in order to solve a problem.
- A set of guidelines that describe how to perform a task.
- a sequence of instructions telling a computer what to do
- Algorithms need to have their steps in the right order (think of a recipe)
- https://youtu.be/CvSOaYi89B4
- Therefore an algorithm is a procedure or formula for solving a problem, based on conducting a sequence of specified actions

High Level Language

- Easy for Programmers to understand
- Contains Engilish Words

Low Level Langugae

- The computer's own Language
- Binary numbers, in 1's and 0's



justcode.me

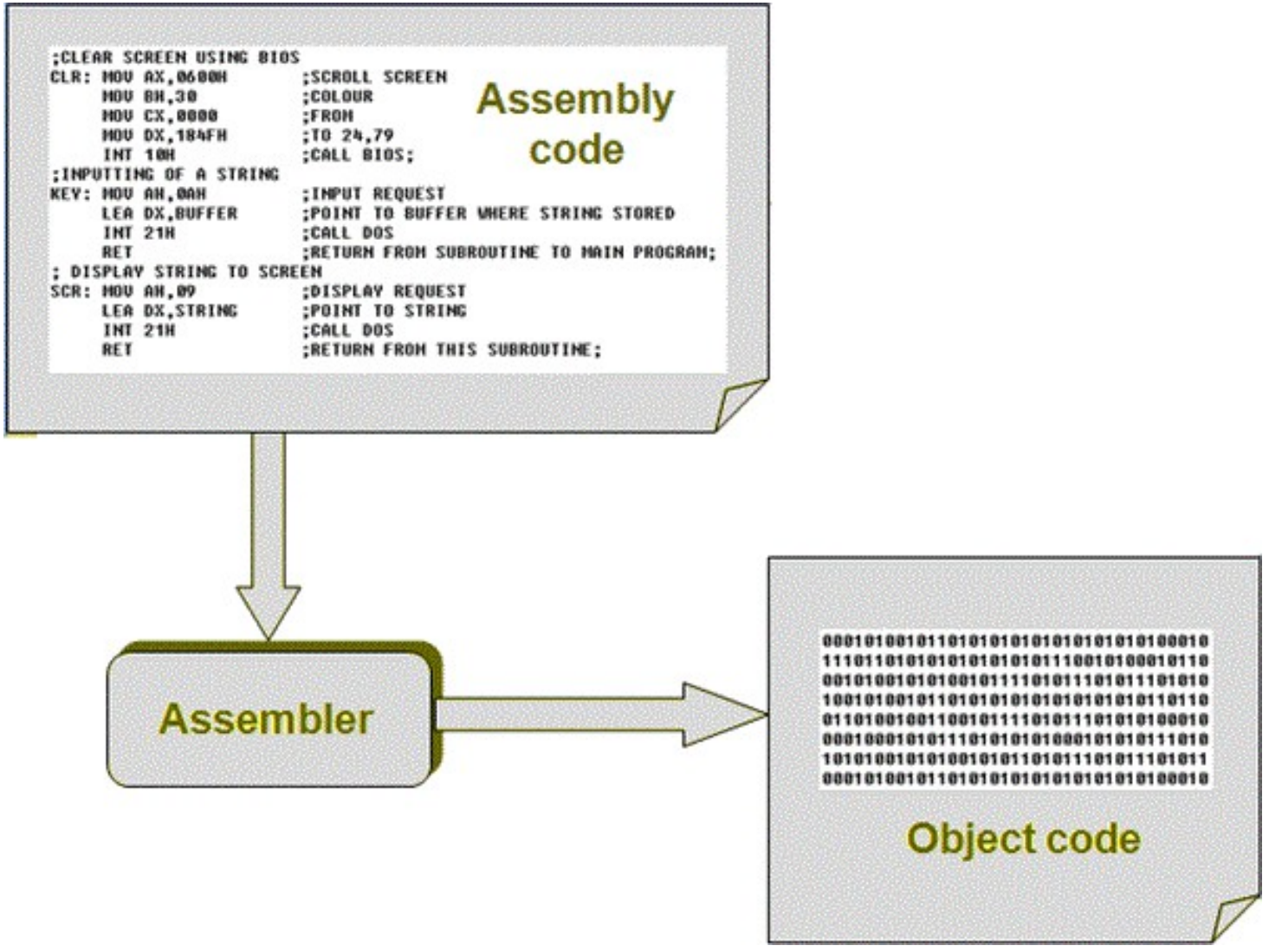
Low Level Programming Languages

- Assembly Language
  - Simple commands called mnemonics
  - ADD, SUB, DIV, JMP, MOV, HALT, GO
  - Processor dependant (Therefore Hardware Specific)
- Difficult for humans to read
- Quick execution however
- Drivers written in Assembly Language
- Converted into Machine Code (or Object Code) using an “Assembler”

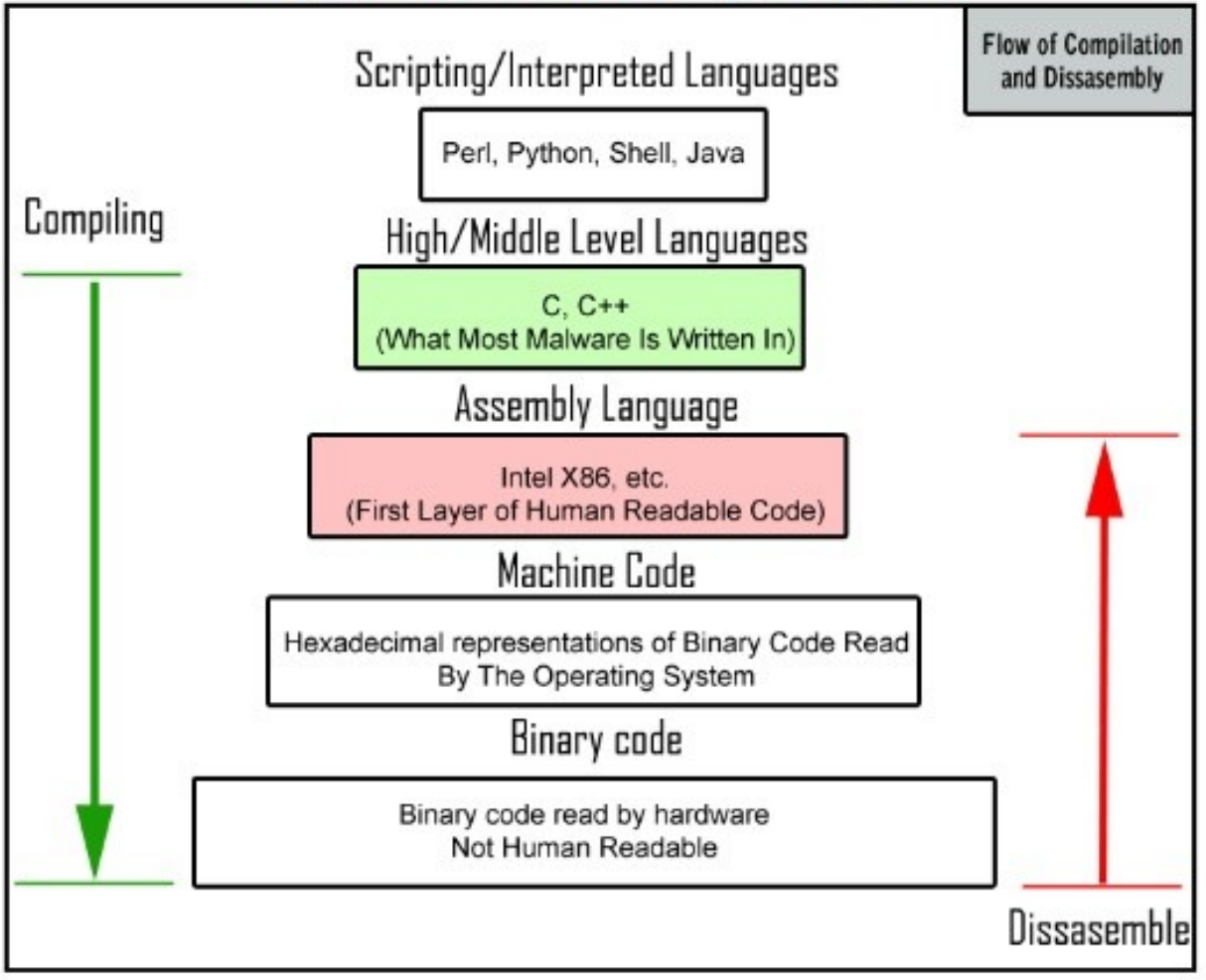
Assembly Code

An assembly language program that adds 10 data bytes. Data is stored in memory location starting from 4460H. The result is 8 bits only and in stored in 4480H.

Label	Instruction	Comments
	LXI H, 4460H ;	HL register pair points at memory location 4460H
	MVI C, 0A H ;	Sets up a decrement counter
	MVI A, 00H ;	Clears Accumulator
jump1:	MOV B, M ;	Moves content of memory location to B
	ADD B ;	Adds A and B and stores result at A
	INX H ;	HL pair points at M+1 memory location
	DCR C ;	Decreases C counter by one
	JNZ jump1: ;	Until C is not control jumps to jump1:
	STA 4480H ;	Stores result at 4480H (when C=0)
	HLT ;	Terminates the program


;CLEAR SCREEN USING BIOS
CLR: MOV AX,0600H ;SCROLL SCREEN
MOV BH,30 ;COLOUR
MOV CX,0000 ;FROM
MOV DX,184FH ;TO 24,79
INT 10H ;CALL BIOS;
;INPUTTING OF A STRING
KEY: MOV AH,0AH ;INPUT REQUEST
LEA DX,BUFFER ;POINT TO BUFFER WHERE STRING STORED
INT 21H ;CALL DOS
RET ;RETURN FROM SUBROUTINE TO MAIN PROGRAM;
; DISPLAY STRING TO SCREEN
SCR: MOV AH,09 ;DISPLAY REQUEST
LEA DX,STRING ;POINT TO STRING
INT 21H ;CALL DOS
RET ;RETURN FROM THIS SUBROUTINE;
Assembly code
Assembler
Object code

High Level Languages



Procedural Languages

- Commands executed in sequence
- Uses variables (Names for memory allocation)
- Uses Loops to control flow
- Top to Bottom Approach
- Large programs divided into functions (Modular)
- Complied to Object Code for a specific platform

Script Languages

- Interpreted, not compiled
- Errors discovered at run time
- Plain text files
- Perl, Php, Python, PowerShell, VBScript, Javascript

Script Languages

- Perl - #!/usr/bin/perl
- Python - #!/usr/local/bin/python
- PHP - <?php
- VBScript - <script type="text/vbscript">
- Javascript - <script language = "Javascript">

Event Driven Programming

- An Approach, not a language
- Improves responsiveness, flexibility
- Where Program written to Responds to Events
- Can be good or bad events
- e.g. Mouse Click, Mouse move, Key Down, Key Up, Key Press etc
- Program flow depends on events

Object Orientated Programming

- All objects have attributes and behaviour
- Objects have **both** data and behaviour
- Procedural Programming Functions have inputs and outputs
- Main difference
  - OO - attributes and behaviours are contained within a single object
  - Procedural - attributes and behaviours are separated.

Object Orientated Programming

- Encourages Modularisation and code reuse
- Uses Classes to define Objects
- 3 Main Characteristics
  - Encapsulation
  - Inheritance
  - Polymorphism

Classes

- Think of a Television
  - We do not have to open the case in order to use it.
  - We have some controls to use it (an interface).
  - We can still understand what a television is, even when it’s connected to an external device (DVD,PC etc).
  - It is complete when we purchased it
  - Any external requirements are well documented.
  - The TV will not crash!

Classes

- Provide a well-defined interface - such as the TV remote control.
- Represent a clear concept - such as the concept of a television.
- Be complete and well-documented - the television should have a plug and should have a manual that documents all features.
- The code should be robust - it should not crash, like the television.

Exercise

- Think about a dog.
  - What makes a dog a dog?
  - What does a dog do?

Classes

- States - (or data) are the values that the object has.
- Methods - (or behaviour) are the ways in which the object can interact with its data, the actions.
- A Class defines the concept of something
- An instance of a class is called an object
- Objects can be real (a PC) or conceptual (a Database)
- Objects have own identity and are independent from each other

Exercise

- Revisit the dog exercise and decide what is are states (or data) and what are methods(or behaviour)

Encapsulation

- A term to describe the hiding of the mechanics of the object
  - Cannot see the actual implementation
  - we don't need to understand how the object works.
  - Need to understand is the interface that is provided for us.
- Encapsulation is "data-hiding" allowing only certain parts of an object to be visible and other parts will be hidden

Encapsulation Advantages

- The user need only understand the interface.
- The user need not understand how the implementation works or was created.
- The programmer can change the implementation, but does not need to notify the user.
- If the programmer does not change the interface in any way, the user will be unaware of any changes.

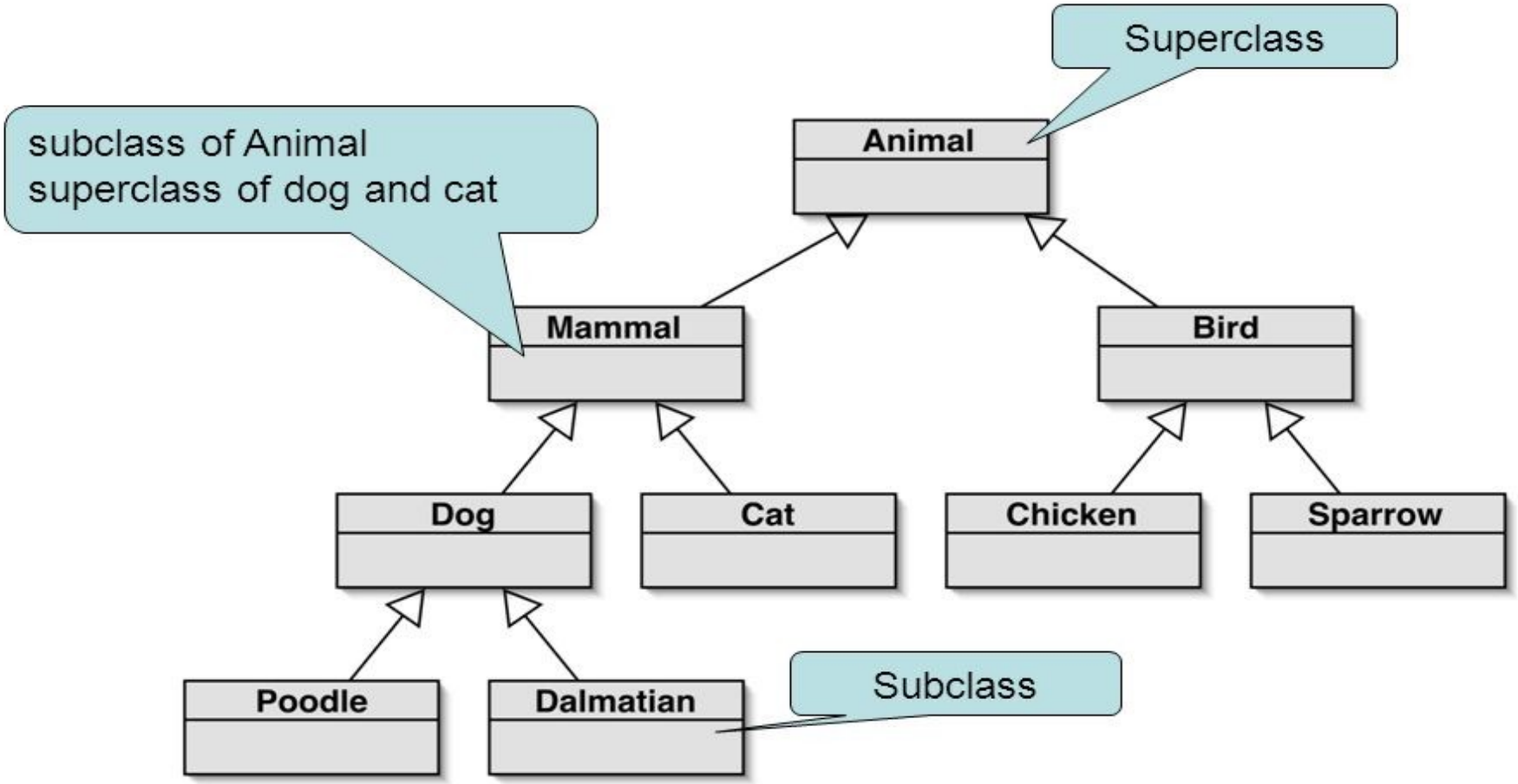
Encapsulation

- Public, *Protected*, Private
- Public methods – defines the interface
- Private methods – defines the implementation
- Protected – Used in inheritance. Derived Classes can access protected items.
- In Summary:
  - Hides the behaviour of an object from its implementation
  - Separates what an object looks like from how it implements behaviour

Inheritance

- When thinking of a **type** of object, if we have several descriptions with some commonality between these descriptions, we can group the descriptions and their commonality using inheritance to provide a compact representation of these descriptions.
- Object-oriented programming groups the commonalities
  - Allows class creation to describe their differences from other classes.
- Canine **is a type of** animal
- Dog **is a type of** Canine
- A Jack Russell **is a type of** dog
- My pet dog Jazz **is a type of** Lancashire Heeler

Inheritance



Inheritance

- Super Class sometimes called a Base Class
- Sub Class sometimes called a derived class

Inheritance

- Objects inherit a behaviour and can add further specialised behaviour
- Objects can Inherit a behaviour and replace it
- Cuts down on the amount of code that needs to be written and debugged
- Promotes Code reuse

Polymorphism

- When a class inherits from another class it inherits both the states and methods of that base class
- Polymorphism means "multiple forms"
- There are two forms of polymorphism, over-riding and over-loading.

Polymorphism - OverRiding

- A derived class inherits its methods from the base class. It may be necessary to redefine an inherited method to provide specific behaviour for a derived class - and so alter the implementation.
- Over-riding is the term used to describe the situation where the same method name is called on two different objects and each object responds differently.
- Over-riding allows different kinds of objects that share a common behaviour to be used in code that only requires that common behaviour.

Polymorphism - Overloading

- The same method name can be used, but the number of parameters or the types of parameters can differ, allowing the correct method to be chosen by the compiler.
- Consider a simple Add function:
  - Adding two integers – add(int a, int b)
  - Adding two floats – add(float a, float b)
  - Adding three integers - add(int a, int b, int c)

OO Terms

- OBJECT-ORIENTED ANALYSIS: Examines the requirements of a system or a problem from the perspective of the classes and objects found in the vocabulary of the problem domain
- OBJECT-ORIENTED DESIGN: Defines systems as made of objects and classes, specifying their relationships (like inheritance) and interactions.
- OBJECT-ORIENTED PROGRAMMING: Implementing a programs with a collection of objects using classes.

Advantages and Disadvantages

- Advantages
  - Code reuse
  - Code uniqueness using inheritance & encapsulation
  - Code Maintainability
  - Code Independence using encapsulation
  - Code highly organised and modular
- Disadvantages
  - Programs are usually larger as inheritance is resolved at run time
  - Programs can be slower, partly due to the modular implementation