INT101
Programming Fundamental

2020/1

Bachelor Science in Information Technology (B.Sc.IT)

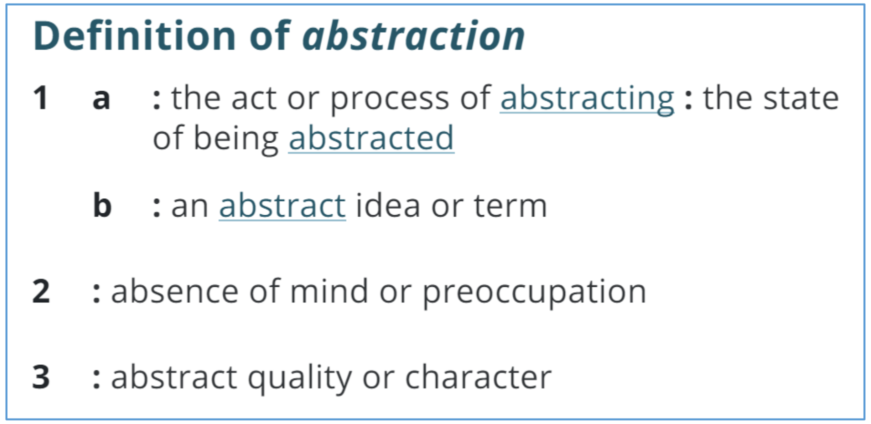
School of Information Technology (SIT)

King Mongkut’s University of Technology Thonburi (KMUTT)

Basic **Abstractions** of Software and Hardware **Architecture** for Programming

- **Abstraction**
  - a general idea or quality rather than an actual person, object, or event; an abstract quality or character (from Merriam-Webster)

- **Architecture**
  - the manner in which the components of a computer or computer system are organized and integrated (from Merriam-Webster)



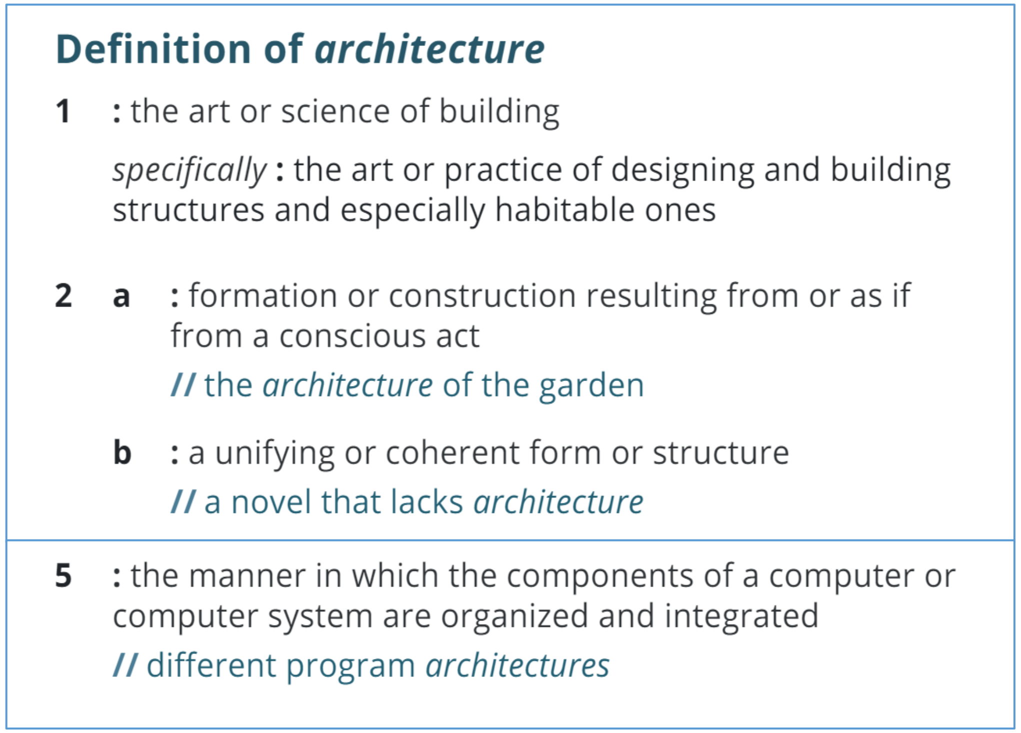
Definition of *abstraction*

1 **a** **:** the act or process of abstracting **:** the state of being abstracted

b **:** an abstract idea or term

2 **:** absence of mind or preoccupation

3 **:** abstract quality or character



Definition of *architecture*

1 **:** the art or science of building

specifically **:** the art or practice of designing and building structures and especially habitable ones

2 **a** **:** formation or construction resulting from or as if from a conscious act

// the *architecture* of the garden

b **:** a unifying or coherent form or structure

// a novel that lacks *architecture*

5 **:** the manner in which the components of a computer or computer system are organized and integrated

// different program *architectures*

Programming Languages

Machine Code

First Generation
Programming Language (1GL)

```
1000101111001000
0010101111000011
```

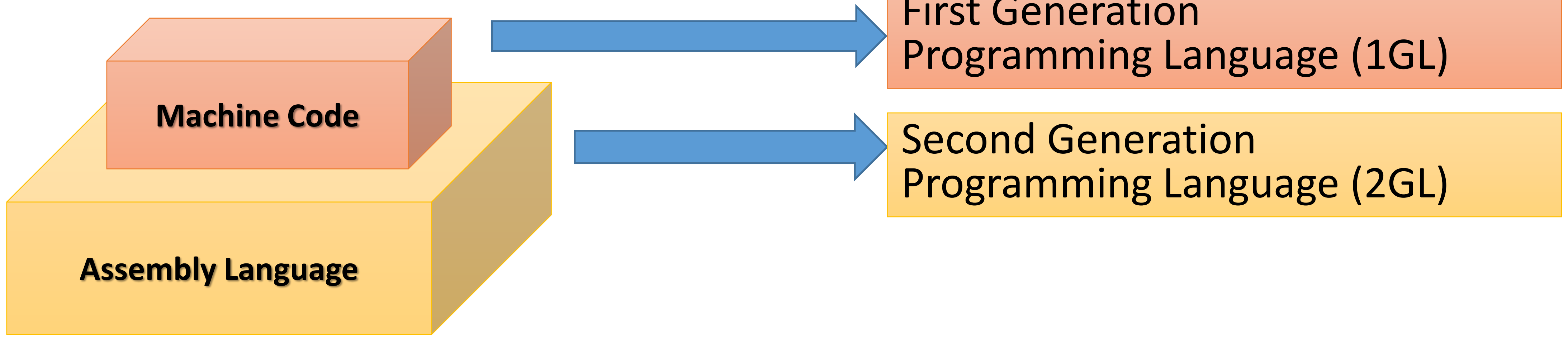
Programming Languages

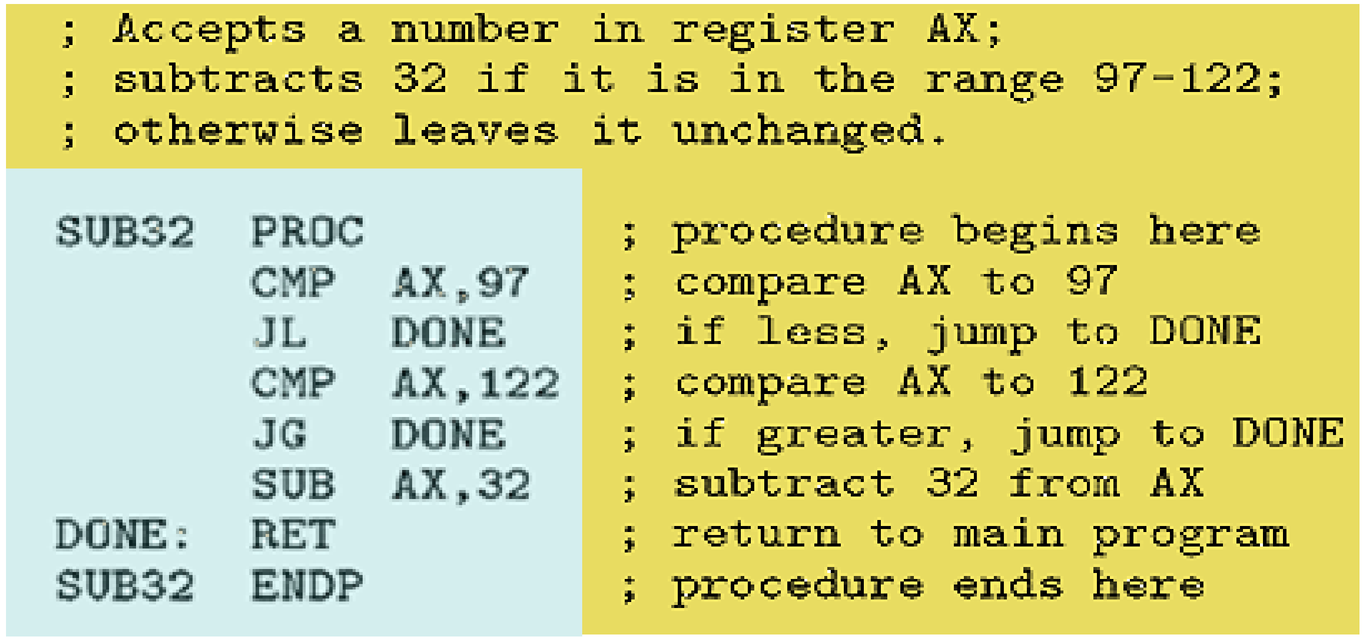
$$\begin{tikzcd}\text{Machine Code} \arrow[r] & \text{First Generation Programming Language (1GL)}\end{tikzcd}$$

OPCODE	D	W	MOD	REG	R/M		
100010	1	1	11	001	000	MOV CX,AX	; CX ← AX
001010	1	1	11	000	011	SUB AX,BX	; AX ← AX - BX

CPU Intel 8086

Programming Languages


Machine Code
Assembly Language
First Generation Programming Language (1GL)
Second Generation Programming Language (2GL)

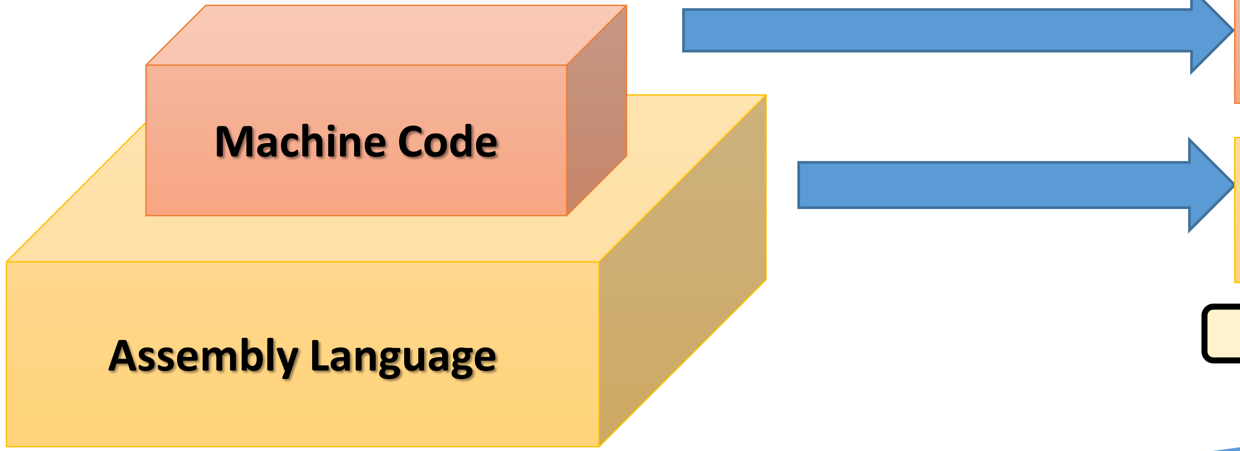


```
; Accepts a number in register AX;
; subtracts 32 if it is in the range 97-122;
; otherwise leaves it unchanged.

SUB32   PROC              ; procedure begins here
        CMP   AX,97       ; compare AX to 97
        JL    DONE        ; if less, jump to DONE
        CMP   AX,122      ; compare AX to 122
        JG    DONE        ; if greater, jump to DONE
        SUB   AX,32       ; subtract 32 from AX
DONE:   RET               ; return to main program
SUB32   ENDP              ; procedure ends here
```

http://www.cybercomputing.co.uk/Languages/Languages/Low_level/assembly.html

Programming Languages

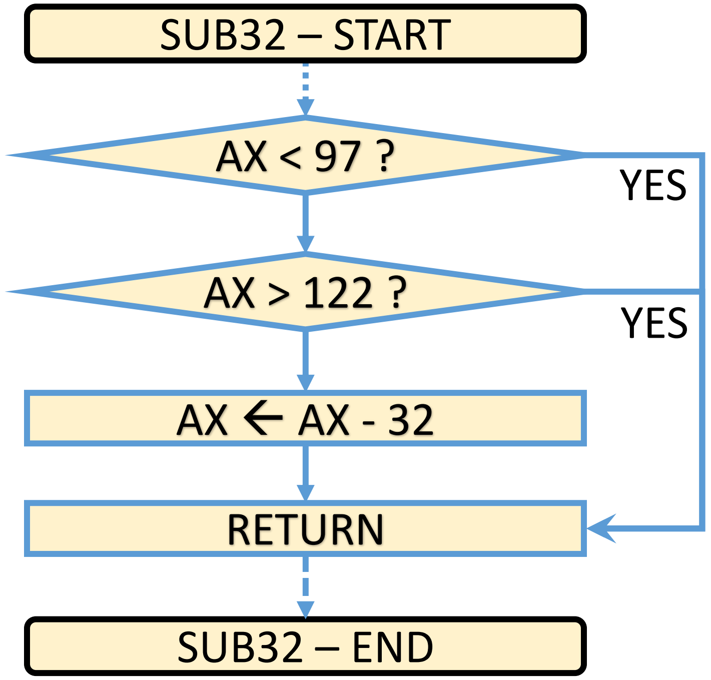


First Generation Programming Language (1GL)

Second Generation Programming Language (2GL)

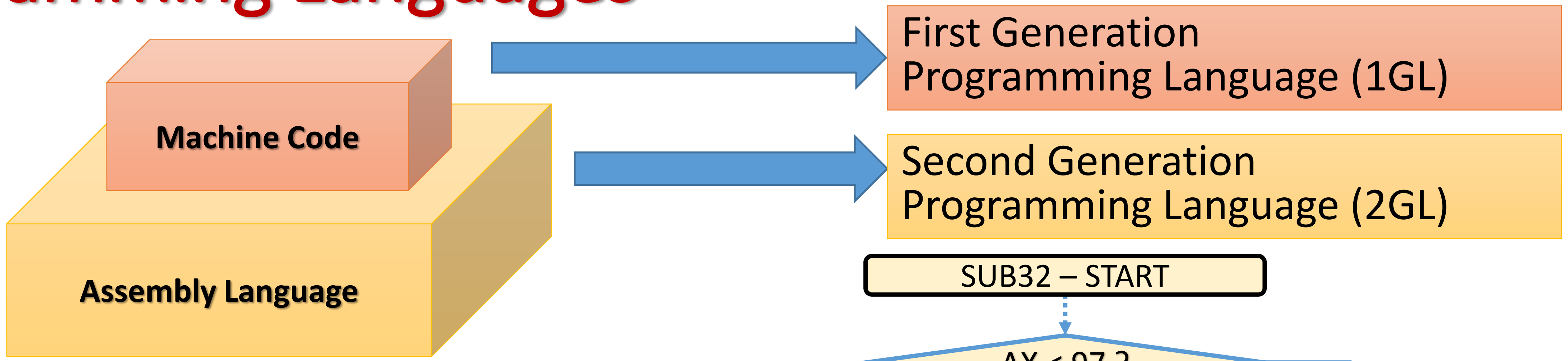
```
; Accepts a number in register AX;
; subtracts 32 if it is in the range 97-122;
; otherwise leaves it unchanged.

SUB32   PROC              ; procedure begins here
        CMP   AX,97       ; compare AX to 97
        JL    DONE        ; if less, jump to DONE
        CMP   AX,122      ; compare AX to 122
        JG    DONE        ; if greater, jump to DONE
        SUB   AX,32       ; subtract 32 from AX
DONE:   RET               ; return to main program
SUB32   ENDP              ; procedure ends here
```



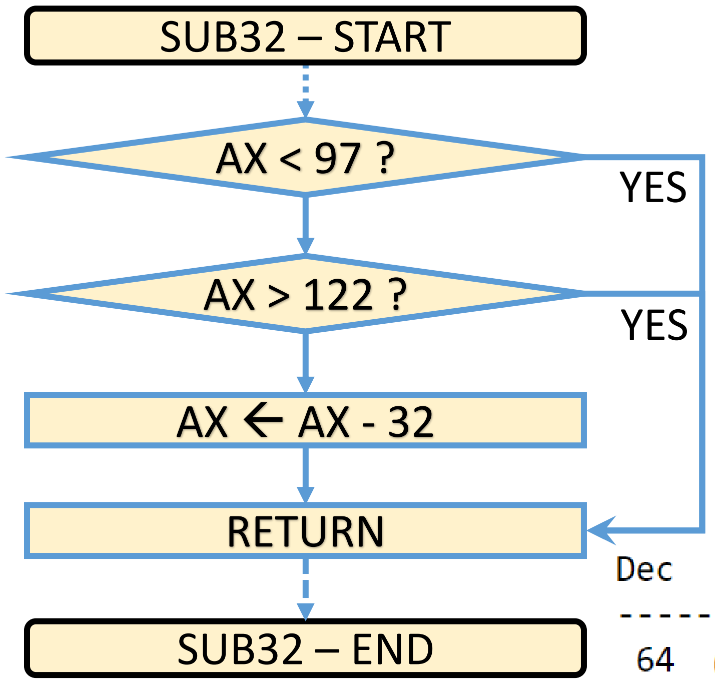
http://www.cybercomputing.co.uk/Languages/Languages/Low_level/assembly.html

Programming Languages


Machine Code
Assembly Language
First Generation Programming Language (1GL)
Second Generation Programming Language (2GL)

```
; Accepts a number in register AX;
; subtracts 32 if it is in the range 97-122;
; otherwise leaves it unchanged.

SUB32   PROC          ; procedure begins here
        CMP   AX,97   ; compare AX to 97
        JL    DONE    ; if less, jump to DONE
        CMP   AX,122  ; compare AX to 122
        JG    DONE    ; if greater, jump to DONE
        SUB   AX,32   ; subtract 32 from AX
DONE:   RET           ; return to main program
SUB32   ENDP          ; procedure ends here
```

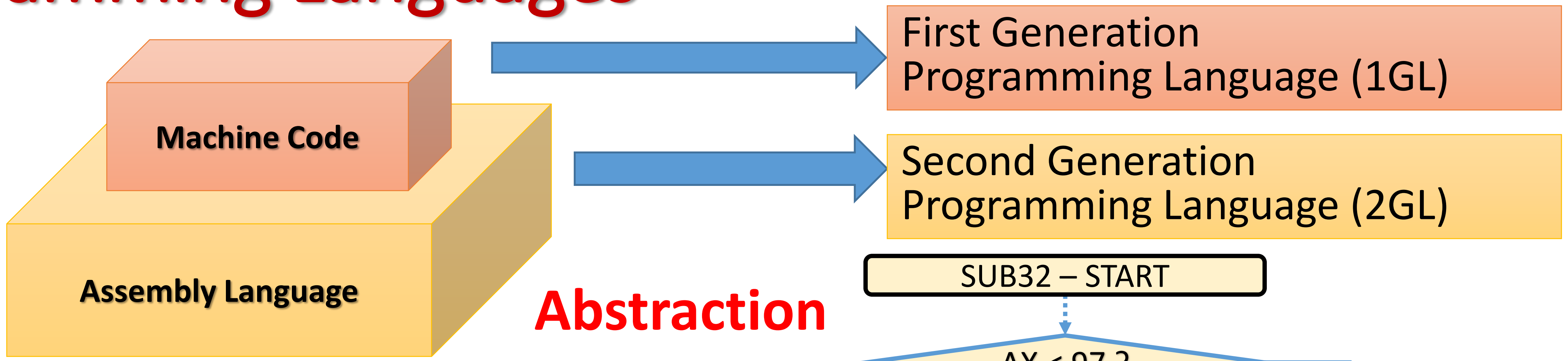


97 – 32 = 65

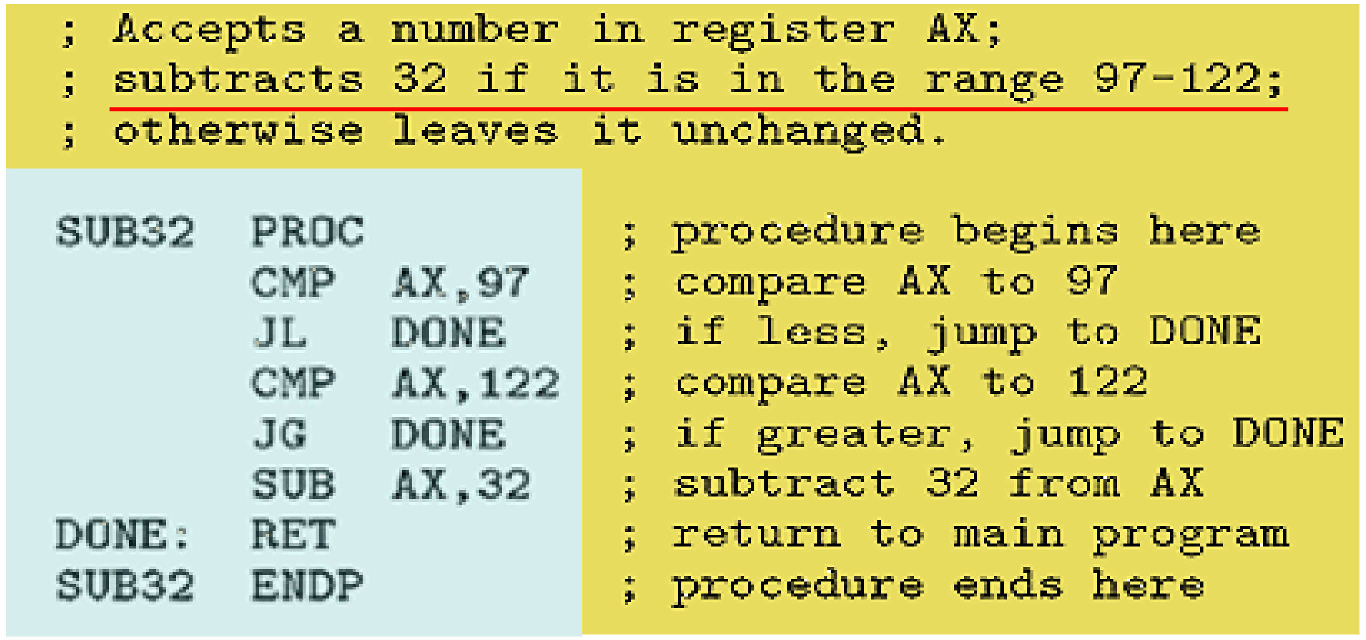
Dec	Char		Dec	Char
64	@		96	`
65	A	←	97	a
66	B		98	b
67	C		99	c
68	D		100	d
69	E		101	e
70	F		102	f

http://www.cybercomputing.co.uk/Languages/Languages/Low_level/assembly.html

Programming Languages

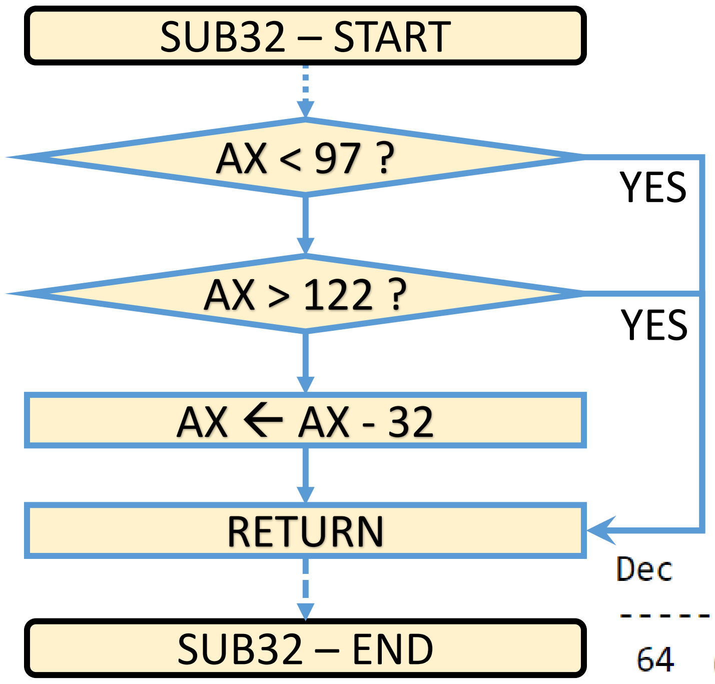

Machine Code
Assembly Language
First Generation Programming Language (1GL)
Second Generation Programming Language (2GL)

Abstraction



```
; Accepts a number in register AX;
; subtracts 32 if it is in the range 97-122;
; otherwise leaves it unchanged.

SUB32   PROC            ; procedure begins here
        CMP   AX,97     ; compare AX to 97
        JL    DONE      ; if less, jump to DONE
        CMP   AX,122    ; compare AX to 122
        JG    DONE      ; if greater, jump to DONE
        SUB   AX,32     ; subtract 32 from AX
DONE:   RET             ; return to main program
SUB32   ENDP            ; procedure ends here
```



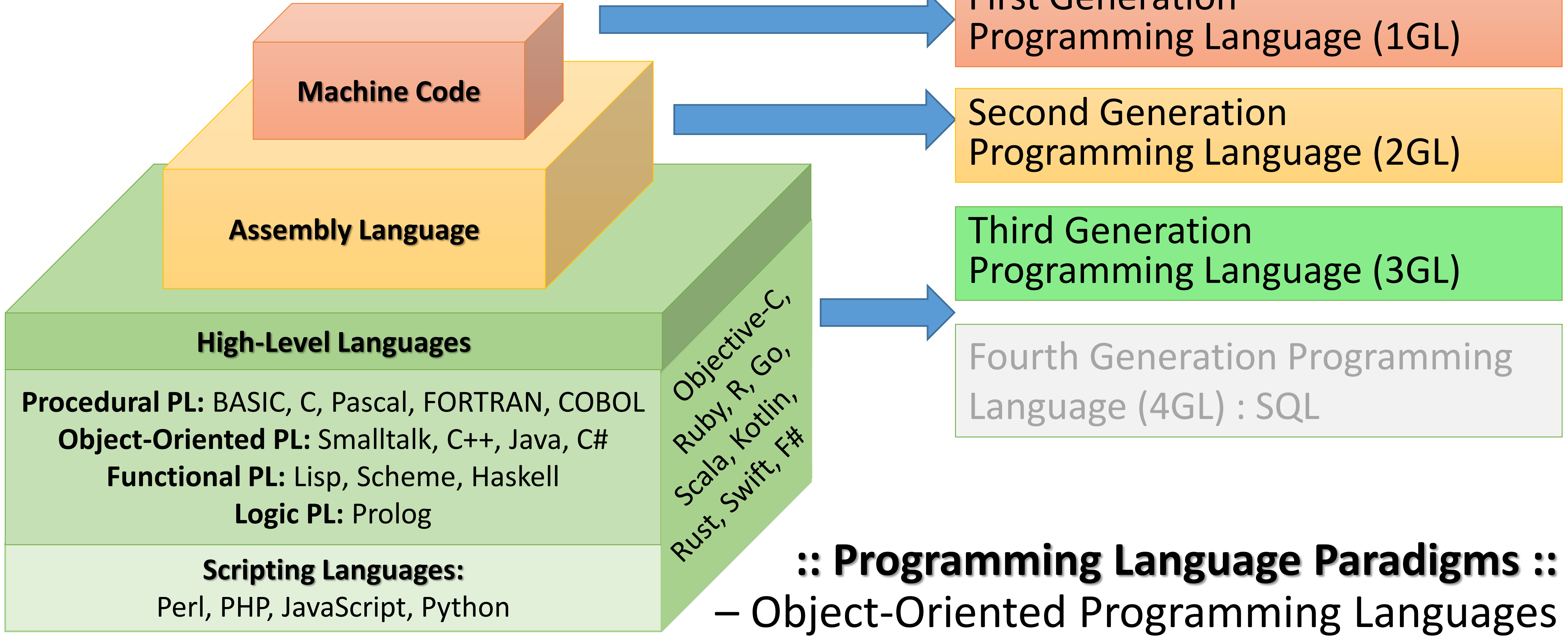
UPPER(AX)

97 – 32 = 65

```
Dec  Char        Dec  Char
---------        ----------
 64  @            96  `
 65  A  <-----    97  a
 66  B            98  b
 67  C            99  c
 68  D           100  d
 69  E          8101  e
 70  F           102  f
```

http://www.cybercomputing.co.uk/Languages/Languages/Low_level/assembly.html

Programming Languages


Machine Code
Assembly Language
High-Level Languages
Procedural PL: BASIC, C, Pascal, FORTRAN, COBOL
Object-Oriented PL: Smalltalk, C++, Java, C#
Functional PL: Lisp, Scheme, Haskell
Logic PL: Prolog
Scripting Languages:
Perl, PHP, JavaScript, Python
Objective-C, Ruby, R, Go, Scala, Kotlin, Rust, Swift, F#
First Generation Programming Language (1GL)
Second Generation Programming Language (2GL)
Third Generation Programming Language (3GL)
Fourth Generation Programming Language (4GL) : SQL

:: Programming Language Paradigms ::

– Object-Oriented Programming Languages

– Functional Programming Languages

– Logic Programming Languages

https://en.wikipedia.org/wiki/Programming_paradigm

Imperative -> Structured -> Procedural -> Object-Oriented

Imperative Programming

- Statements
  - read, compute, write
- Branching: IF THEN
- Jumping: GOTO

Structured Programming

- Statement blocks
  - Loop - while, for
  - If-then-else
- avoid GOTO

Procedural Programming

- Subprogram
- Subroutine
- Procedure
- Function

```
  I = 1
  N = 10
  FAC = 1
START:
  IF I <= N THEN
    FAC = FAC * I
    I = I + 1
    GOTO START
  END
  PRINT FAC
```

if, goto/label

```
I = 1
N = 10
FAC = 0
WHILE I <= N DO
  FAC = FAC * I
  I = I + 1
PRINT FAC
```

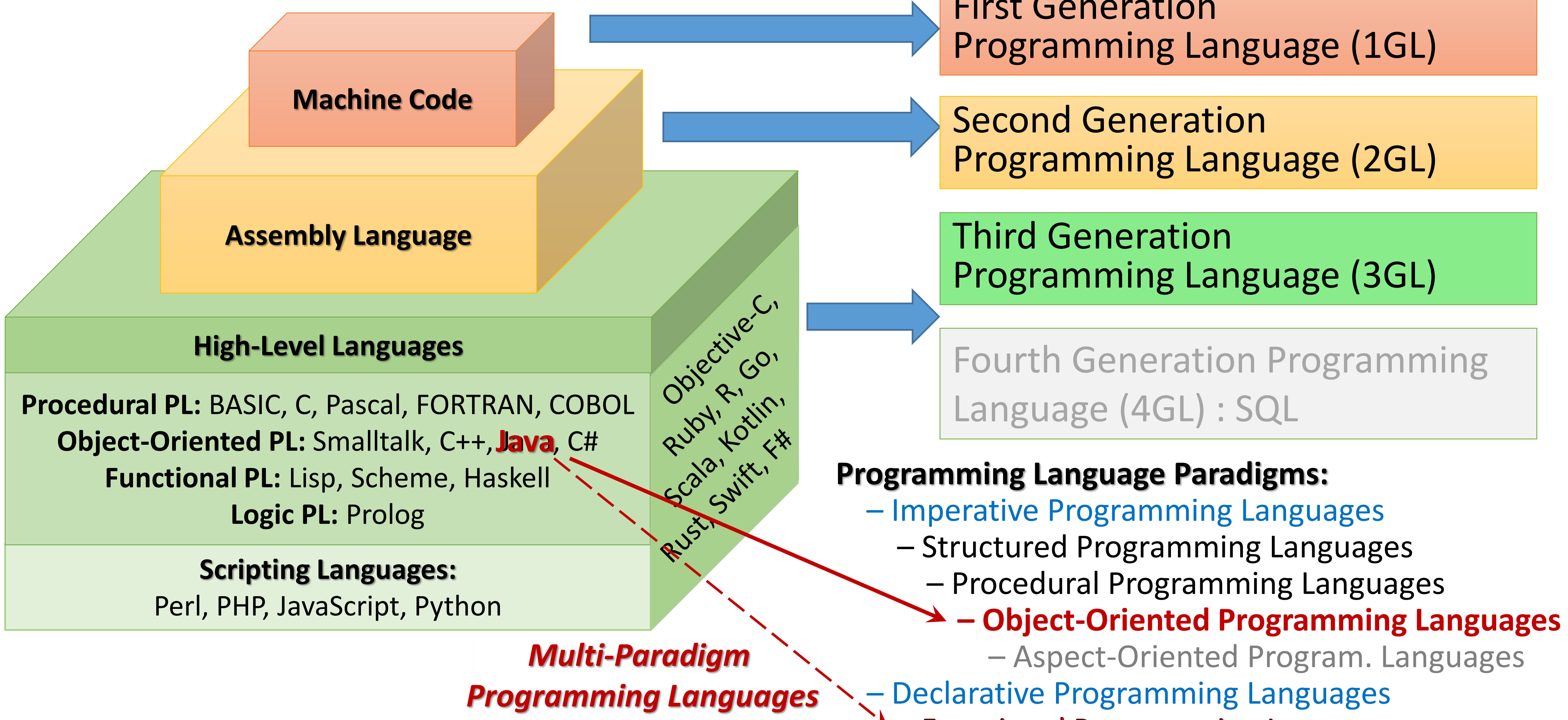
block: if, while, for

```
FUNCTION FACTORIAL(N)
  I = 1
  FAC = 1
  WHILE I <= N DO
    FAC = FAC * I
    I = I + 1
  RETURN FAC

RESULT = FACTORIAL(10)
PRINT RESULT
```

procedure / reusable

Programming Languages



Machine Code
Assembly Language
High-Level Languages
Procedural PL: BASIC, C, Pascal, FORTRAN, COBOL
Object-Oriented PL: Smalltalk, C++, Java, C#
Functional PL: Lisp, Scheme, Haskell
Logic PL: Prolog
Scripting Languages:
Perl, PHP, JavaScript, Python
Objective-C, Ruby, R, Go, Scala, Kotlin, Rust, Swift, F#
First Generation Programming Language (1GL)
Second Generation Programming Language (2GL)
Third Generation Programming Language (3GL)
Fourth Generation Programming Language (4GL) : SQL
Multi-Paradigm Programming Languages

Programming Language Paradigms:

- Imperative Programming Languages
  - Structured Programming Languages
    - Procedural Programming Languages
    - **Object-Oriented Programming Languages**
      - Aspect-Oriented Program. Languages
- Declarative Programming Languages
  - Functional Programming Languages
  - Logic Programming Languages

https://en.wikipedia.org/wiki/Programming_paradigm

Object-Oriented Programming Concept

An Object-Oriented Program

- A program is
  - **a collection of objects**
    - **sending messages** to one another
      - to perform some tasks

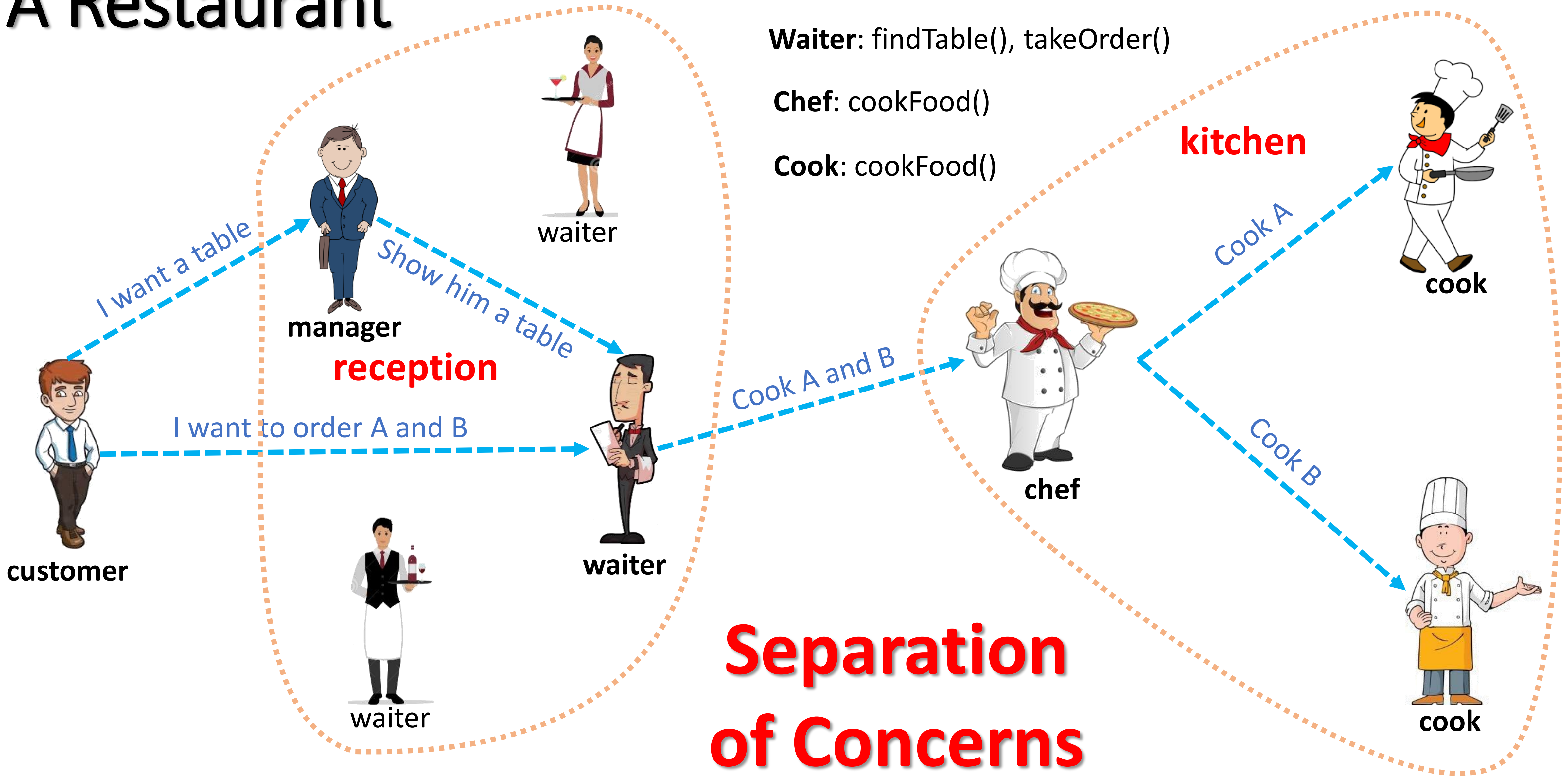
A Restaurant

Manager: findTable()

Waiter: findTable(), takeOrder()

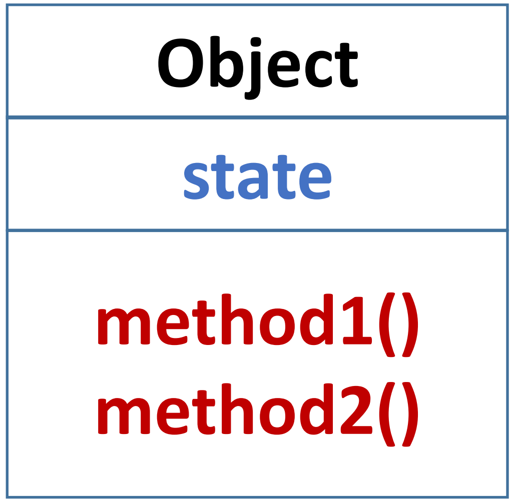
Chef: cookFood()

Cook: cookFood()



Separation of Concerns

An Object



- Object is an entity that
  - Have a **state** (data/information),
  - Can behave according to the **messages** received.
    - All possible messages that an object can receive are pre-defined: **methods.**
    - Upon receiving a message, object may change its state.

Car (Object – Instance)

odometer
speed

- unlockDoor()
- lockDoor()
- openDoor()
- closeDoor()

doorLockStatus
doorOpenStatus

- startEngine()
- stopEngine()
- changeGear()
- accelerate()

engineStatus
gearStatus

- turnWheel()
- break()

wheelPosition

- turnOnAirConditioner()
- turnOffAirConditioner()
- setAirConditionerTemperature()
- setAirConditionerFanSpeed()

airConOnOffStatus
airConTemperature
airConFanSpeed

- turnOnRadio()
- turnOffRadio()
- setRadioVolume()
- setRadioChannel()

radioOnOffStatus
radioVolumeLevel
radioChannel

Car (Object –> Collection of Objects)

speed

odometer

- **Door System**
  - unlock()
  - lock()
  - open()
  - close()

lockStatus

openStatus

- **Engine**
  - start()
  - stop ()
  - changeGear()
  - accelerate()

engineStatus

gearStatus

- **Wheel System**
  - turnWheel()
  - break()

wheelPosition

- **Air Conditioner**
  - turnOn()
  - turnOff()
  - setTemperature()
  - setFanSpeed()

onOffStatus

temperature

fanSpeed

- **Radio**
  - turnOn()
  - turnOff()
  - setVolume()
  - setChannel()

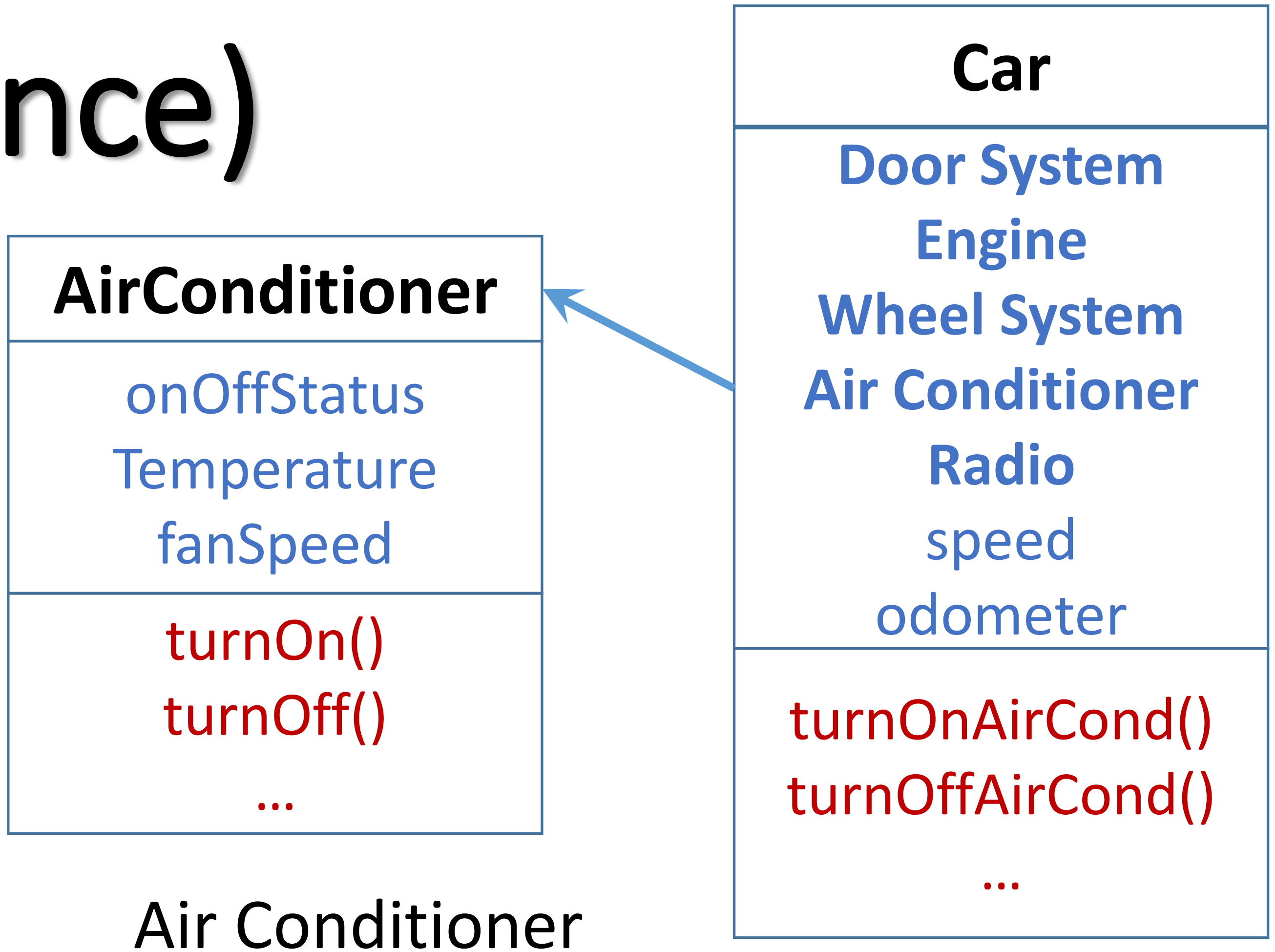
onOffStatus

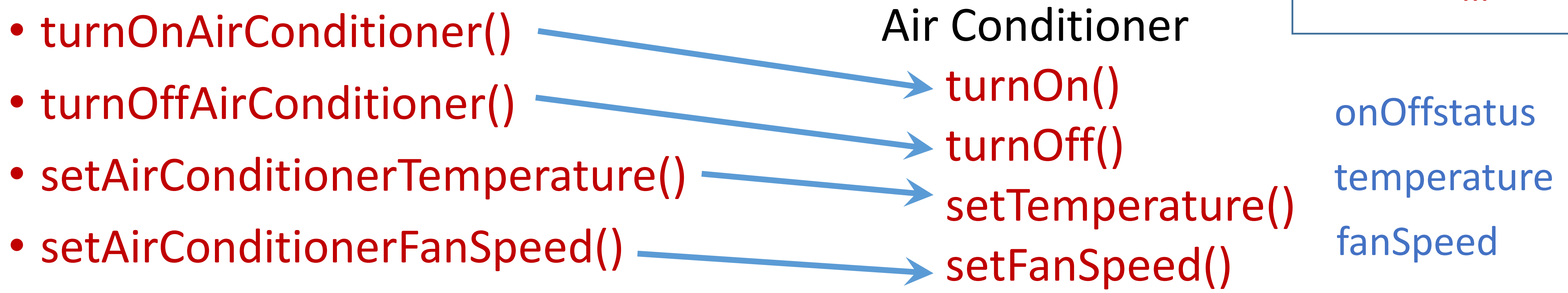
volumeLevel

channel

Car (Object – Instance)

- Door System
- Engine
- Wheel System
- Air Conditioner
- Radio



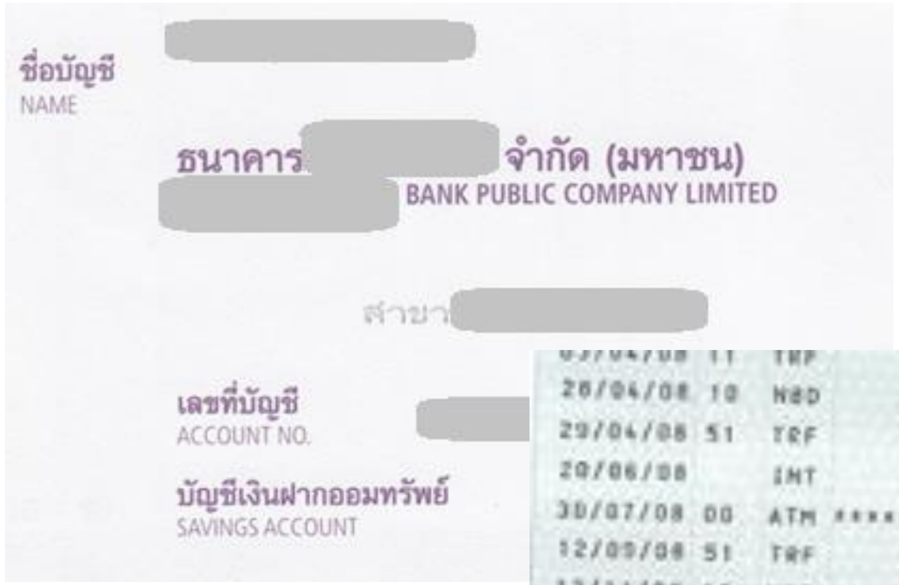


Vending Machine

- Methods

- State

Bank Account



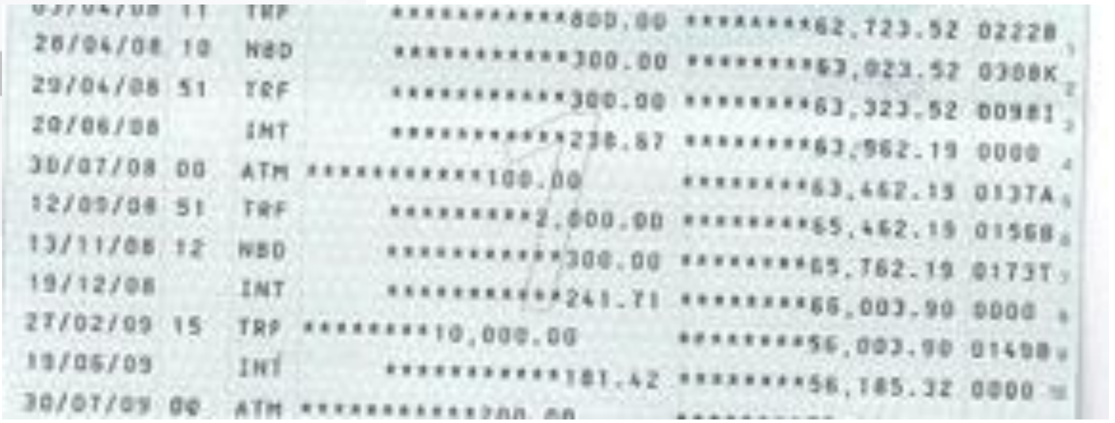
ชื่อบัญชี
NAME [redacted]

ธนาคาร[redacted] จำกัด (มหาชน)
[redacted] BANK PUBLIC COMPANY LIMITED

สาขา[redacted]

เลขที่บัญชี
ACCOUNT NO. [redacted]

บัญชีเงินฝากออมทรัพย์
SAVINGS ACCOUNT



03/04/08	11	TRF		*************800.00	*********62,723.52	0222B
28/04/08	10	WBD		*************300.00	*********63,023.52	0308K
29/04/08	51	TRF		*************300.00	*********63,323.52	00981
20/06/08		INT		*************238.67	*********63,562.19	0000
30/07/08	00	ATM	*************100.00		*********63,462.19	0137A
12/09/08	51	TRF		**********2,000.00	*********65,462.19	0156B
13/11/08	12	WBD		*************300.00	*********65,762.19	0173T
19/12/08		INT		*************241.71	*********66,003.90	0000
27/02/09	15	TRF	*********10,000.00		*********56,003.90	01498
19/06/09		INT		*************181.42	*********56,185.32	0000
30/07/09	00	ATM	*************200.00			

- an account number
- an account owner
- a balance
- a transaction history

- deposit – an amount to deposit, the date of deposit
- withdraw – an amount to withdraw, the date of withdraw
- transfer – an amount to transfer, an account to transfer to, the date of transfer
- inquiry – (the date of inquiry)

- adding an interest – the date of adding the interest, (interest rate)

- open a new account – an account owner, the date of account opening
- close the account – the date of account closing

Elevator (Lift)

- Methods
- State

Air Conditioner

- turn on
- turn off
- increase/decrease temperature
- increase/decrease fan speed
- set air direction – the air direction
- set on/off timer – the time interval to turn on/off

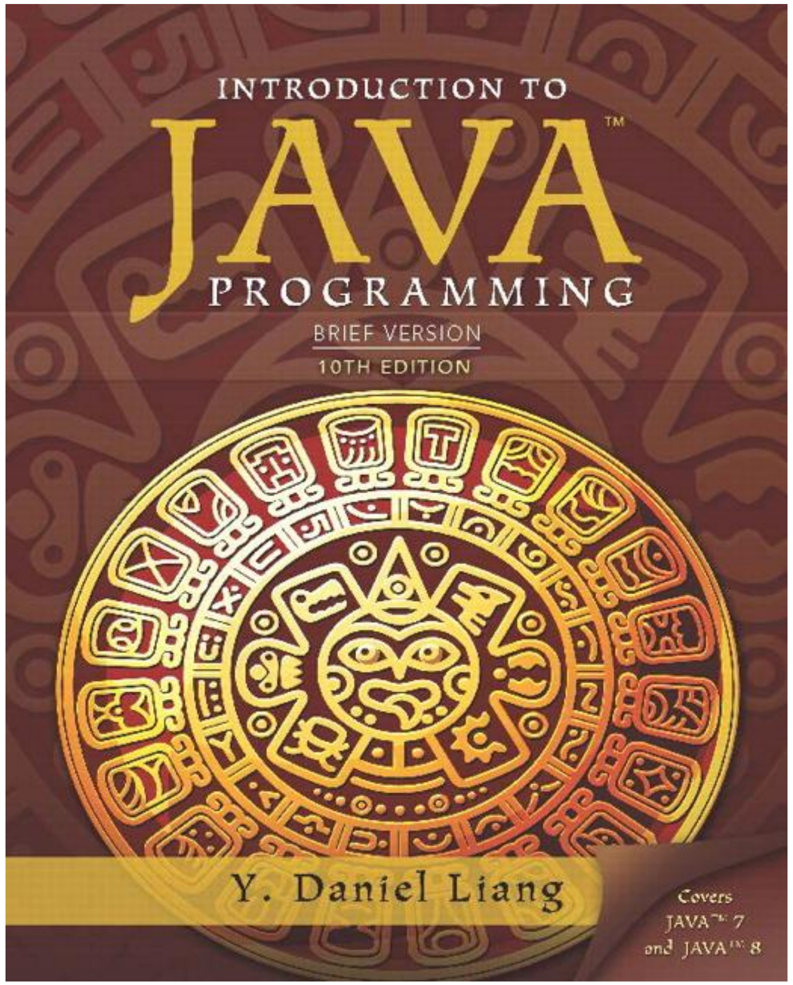
Television

- turn on
- turn off
- set on/off timer – the time interval to change to
- change the channel – the channel to change to
- go back to the previous channel
- increase/decrease volume
- change the TV mode (TV, Cable TV, HDMI, VGA, …)
- change contrast/brightness
- …

Book

- ISBN
- Title
- Authors
- Publisher
- Price

- get/set ISBN()
- get/set Title()
- add/remove Author()
- get Authors()
- get/set Publisher()
- get/set Price()



Dice (rolling dice)

- Methods
- State

Time Stamp

(day-month-year , hour : minute)
(to set/read time for an appointment)

- Methods
- State

Vending Machine

for customer

receiveMoney()   moneyInserted

returnMoney()   **totalMoney**

giveProduct()   **totalProducts**

---

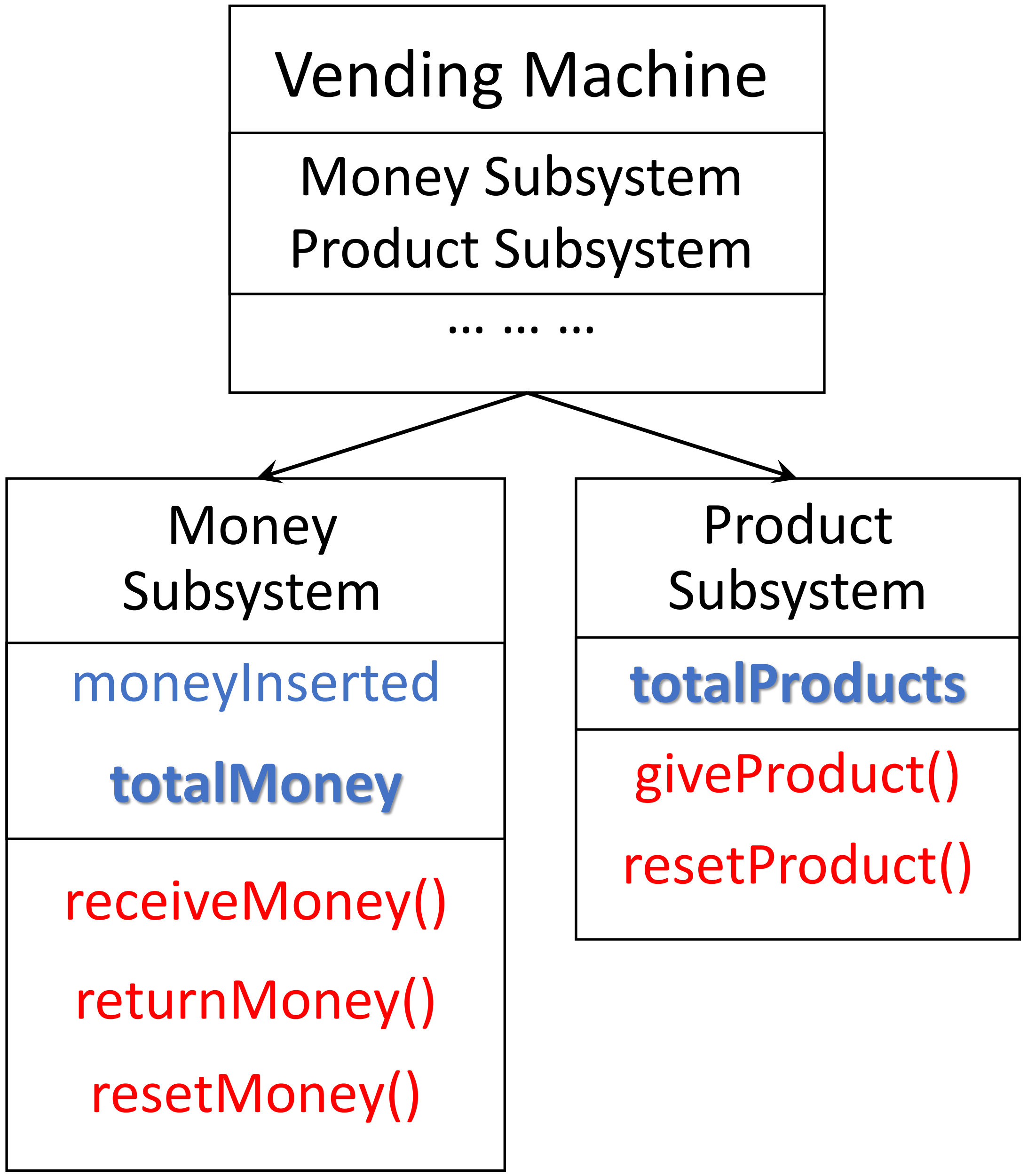
resetMoney()

resetProduct()

for service maintenance

Other Issues:

- On/Off Switch
  - OnOffStatus
- Temperature Control
  - currentTemperature
  - targetTemperature



Elevator (Lift)

outside	up()
	down()
		requestedDirectionsFromFloors

inside	openDoor()
	closeDoor()
		doorStatus

	gotoFloor()	**floorsToGo**

	alarmOn()
	alarmOff()
		alarmStatus

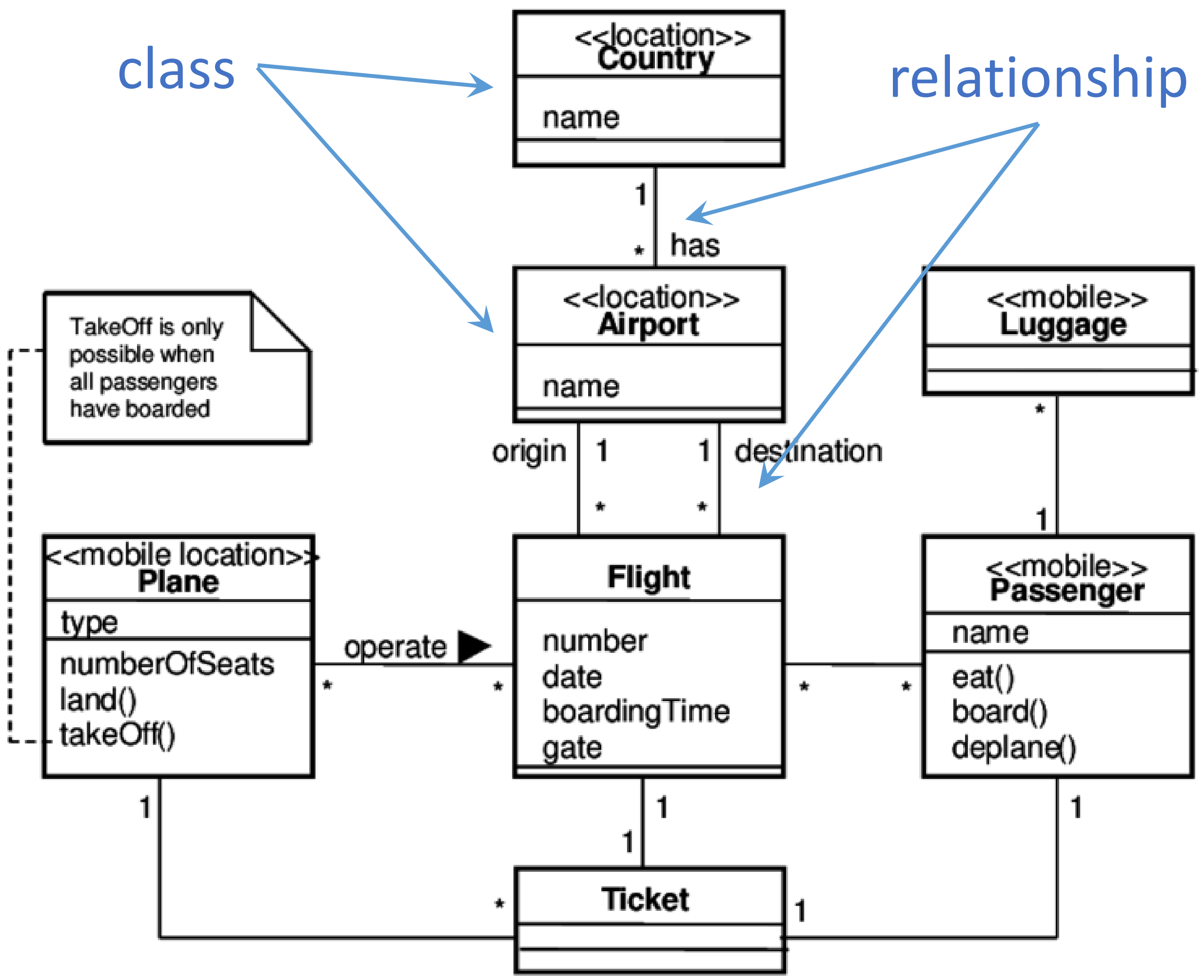
administration	turnOn()
	turnOff()
		onOffStatus

internal status	currentFloor movingDirection	currentWeight maxWeight

Elevator
Engine Controller Alarm Subsystem Door Subsystem Weight Subsystem
… …

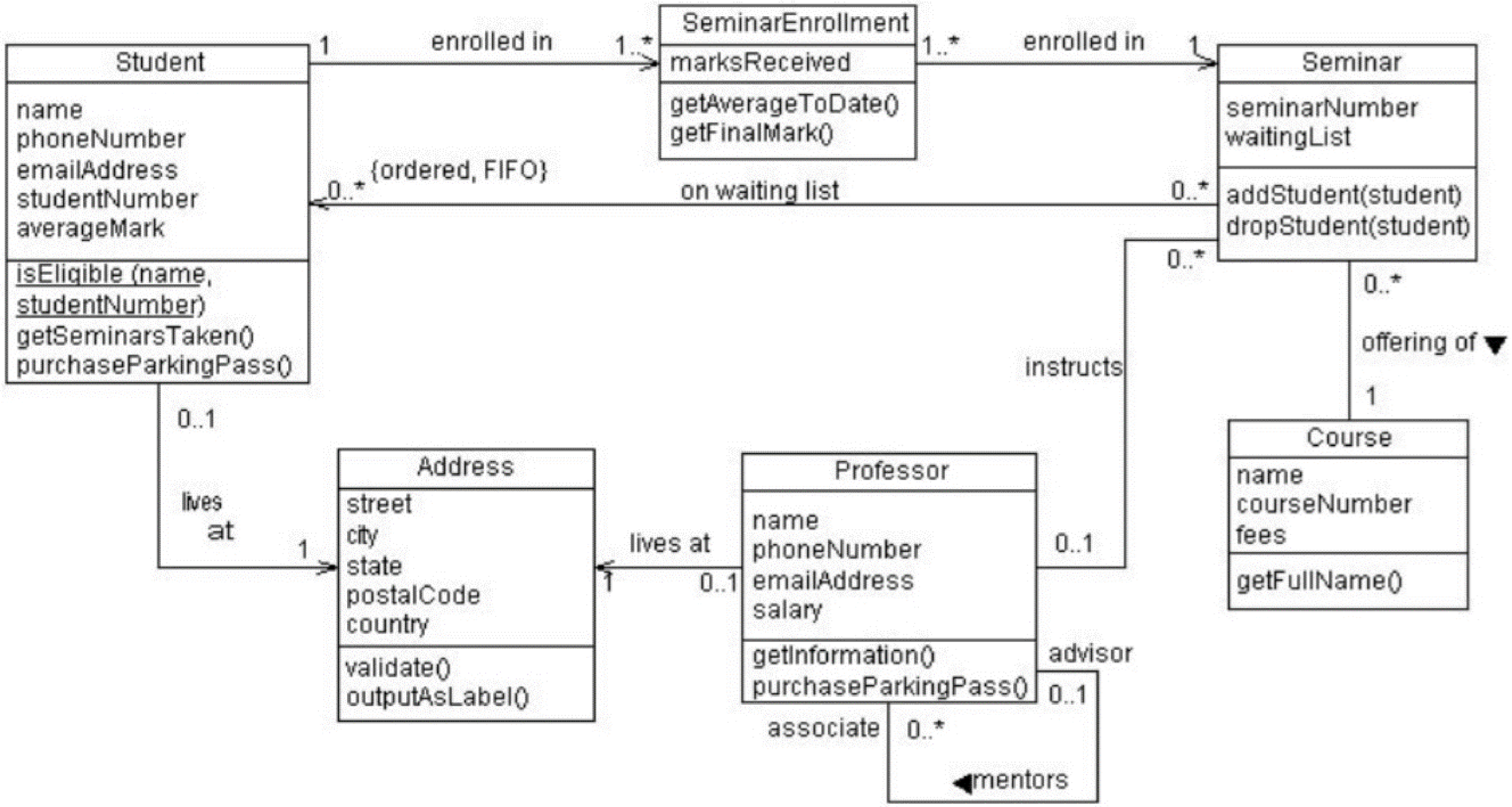
Unified Modeling Language (UML): Class Diagram

- The static structure of a system showing **class structures** in the system and **the relationships among these classes**
- A class structure consists of
  - state (attributes / instance variables)
  - behaviors (operations / methods)



https://www.researchgate.net/figure/Class-diagram-of-the-airport-example_fig1_226891682

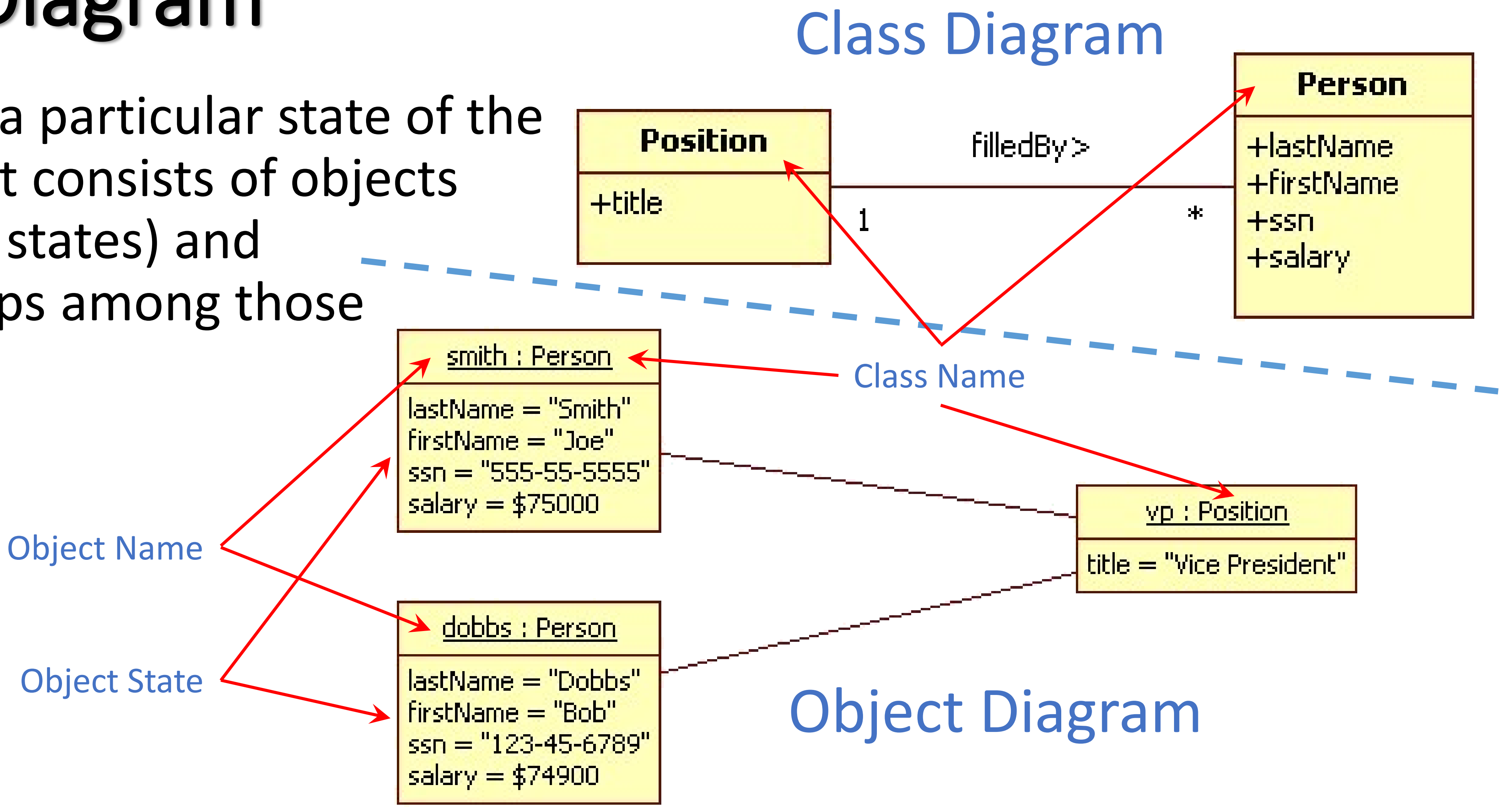
Class Diagram Example



https://www.slideshare.net/Nadia_Nazeer/class-diagrams-64840010

Unified Modeling Language (UML): Object Diagram

- Represent a particular state of the system that consists of objects (with their states) and relationships among those objects



http://www.cs.sjsu.edu/faculty/pearce/oom/ooa/domain/domains.htm