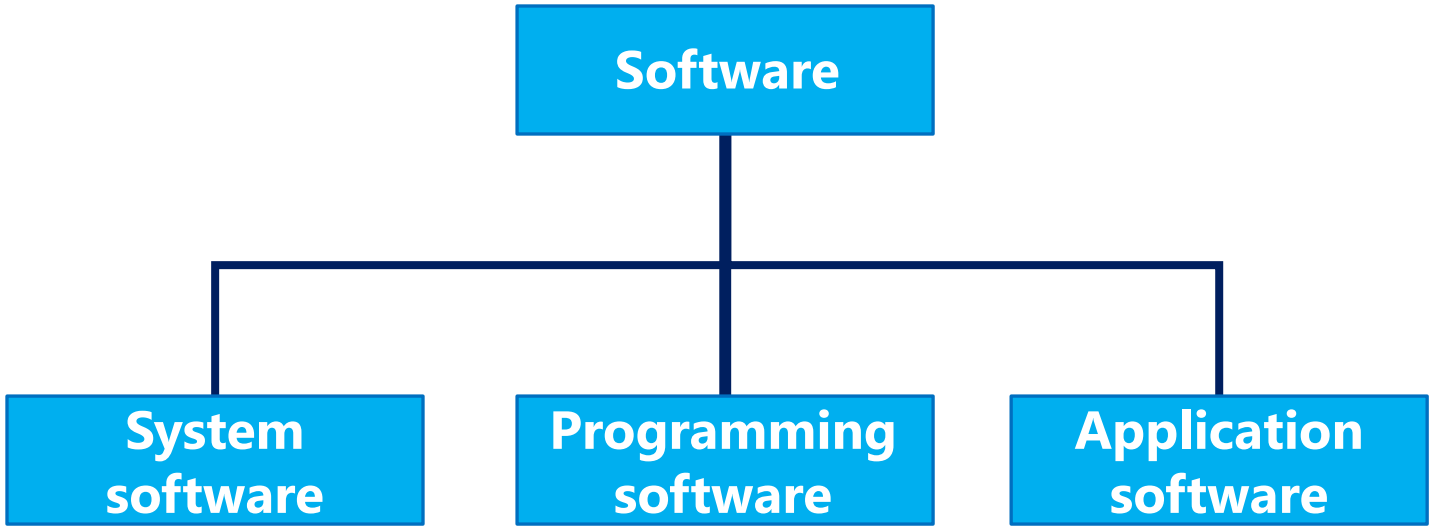
❑ What is Software ?

Software, is a collection of computer programs and related data that provide the instructions for telling a computer what to do and how to do it.

❑ Types of Software



❖ System Software

System software provides the basic functions for computer usage and helps run the computer hardware and system.

❖ Programming Software

Programming is the process of designing, writing, testing, debugging, and maintaining the source code of computer programs. This source code is written in a programming language. The purpose of programming is to create a program that exhibits a certain desired behavior.

❖ Application Software

Application software is the general designation of computer programs for performing user tasks.

Types of Application Software:

- **Mobile App:**

Application that runs on mobile platform. Example: Instagram app,

- **Desktop App:**

Application that runs stand-alone in a desktop or laptop computer.
Example: Microsoft Word, Web Browser.

- **Web App:**

Apps that run on a web browser(Mozilla, Google Chrome etc.)
Example: www.facebook.com, www.google.com

DFD

DFD- Data Flow Diagrams

Graphical representation of flow of data inside application.

Used for visualization and data processing.

DFD elements are:

- External Entity
- Process
- Data Flow
- Data Store

1) **External entity**:

Can be user or external system that performs some process or activity in project Symbolized with rectangle.

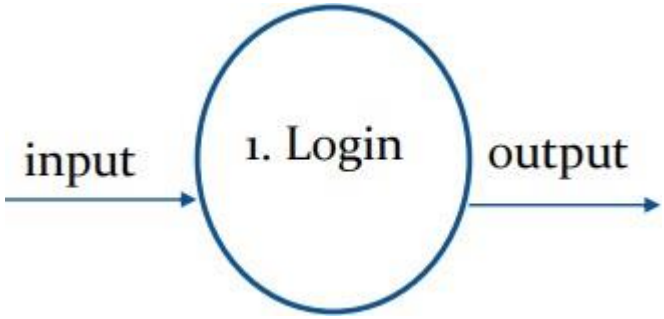
If we have entity ‘admin’ then symbol will be

2) **Process:**

Work or action taken on incoming data to produce output

Each process must have input and output

Symbolized as



TOPS Technologies

3) **Data Flow**

Can be used to show input and output of data

Should be named uniquely and don't include word 'data'

Names can be 'payment', 'order', 'complaint' etc.

Symbolized as

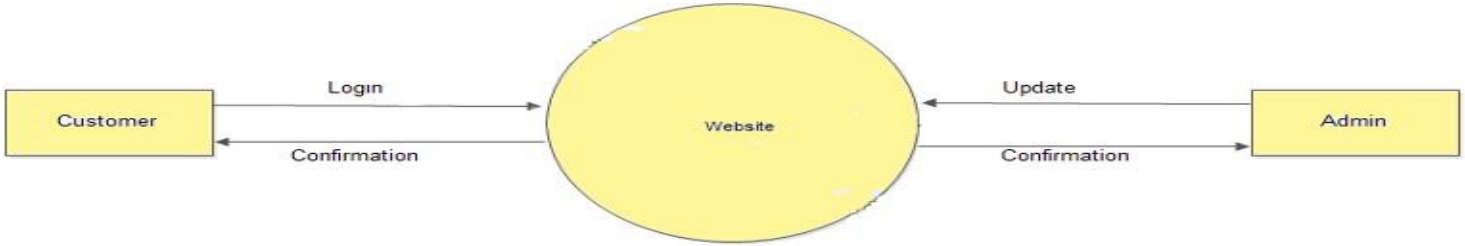
4**) Data Store**

Can be used to show database tables Only process may connect data stores

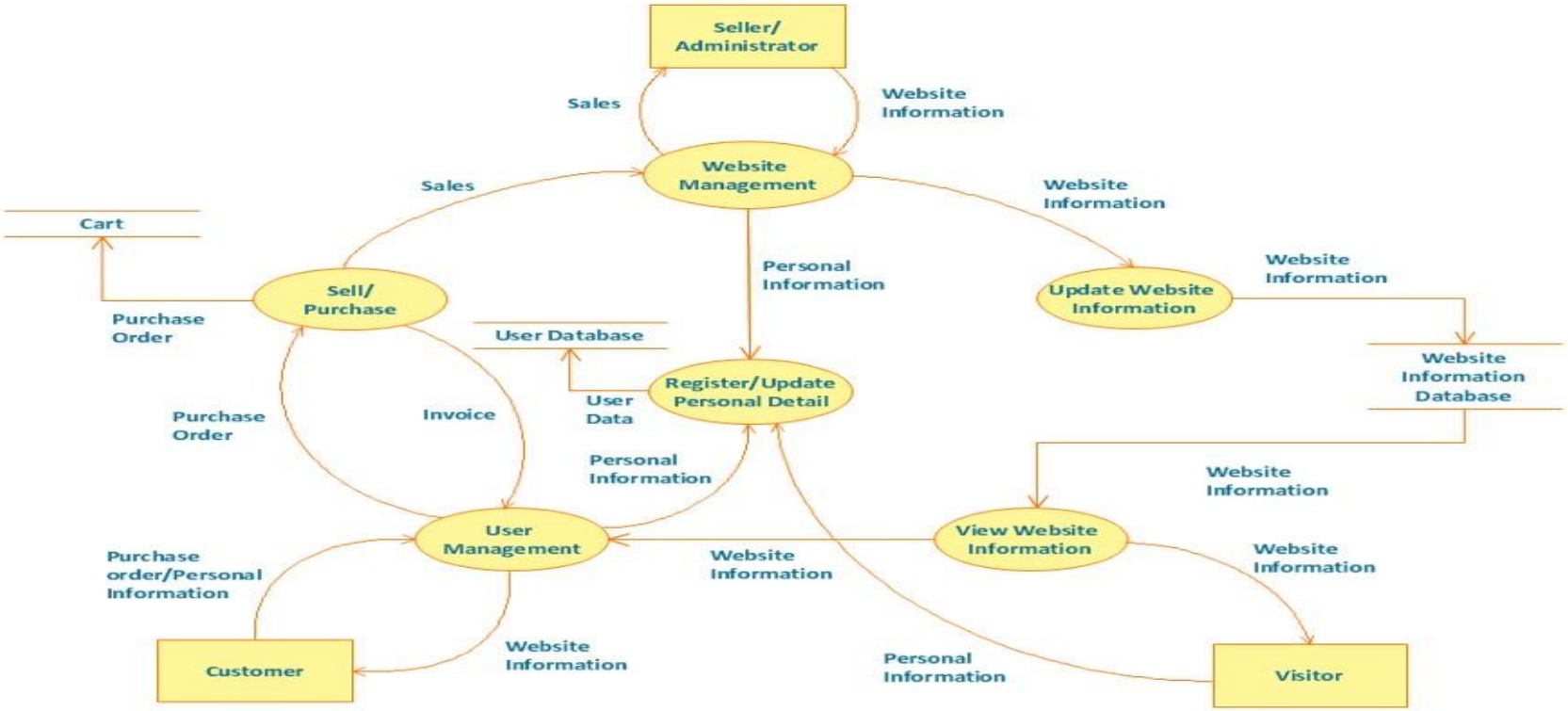
There can be two or more process sharing same data store

Symbolized as

TOPS Technologies



Level Zero DFD



Level One DFD

DFD RULES

1. Rule 1 :

Each process must have data flowing into it and coming out from it

2. Rule 2:

Each data store must have data going inside and data coming outside

3. Rule 3:

A data flow out of a process should have some relevance to one or more of the data flows into a process

4. Rule 4:

Data stored in system must go through a process.

5. Rule 5:

Two data stores can't communicate with each other unless process is involved in between .

6. Rule 6

The Process in DFD must be linked to either another process or a data store A process can't be exist by itself, unconnected to rest of the system

CONTEXT LEVEL (DFD level-0:)

It’s also context level DFD

Context level diagrams show all external entities. They do not show any data stores.

The context diagram always has only one process labelled 0

DFD Level-1(or 2)

Include all entities and data stores that are directly connected by data flow to the one process you are breaking down

show all other data stores that are shared by the processes in this breakdown

Like login process will linked to users & database in further levelling.

Flow Chart

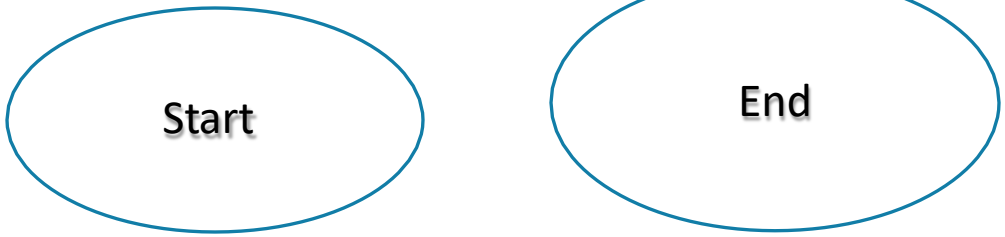
- Used to show algorithm or process . Can give step solution to the problem
- The first flow chart was made by John Von Newman in 1945
- Pictorial view of process
- Flowcharts are generally drawn in the early stages of formulating computer solutions.
- Flowcharts facilitate communication between programmers and business people.
- These flowcharts play a vital role in the programming of a problem and are quite helpful in understanding the logic of complicated and lengthy problems.
- Once the flowchart is drawn, it becomes easy to write the program in any high level language.
- Often we see how flowcharts are helpful in explaining the program to others. Hence, it is correct to say that a flowchart is a must for the better documentation of a complex program. by step

Flowchart Symbol

1) **Start Or End:**

Show starting or ending of any flow chart

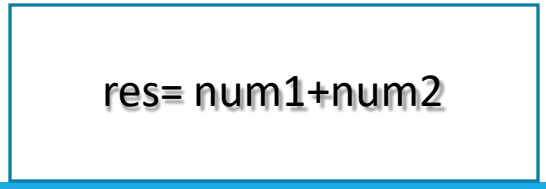
Symbolized as:



2. Process:

Defines a process like defining variables or initializing variable or performing any computation

Symbolized as

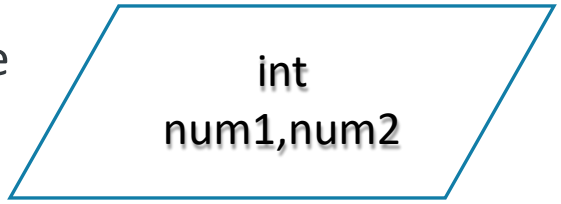


3) **Input or Output**

Used when user have to get or initialize any variable

Like get num1 and num2 from user

Symbolized as

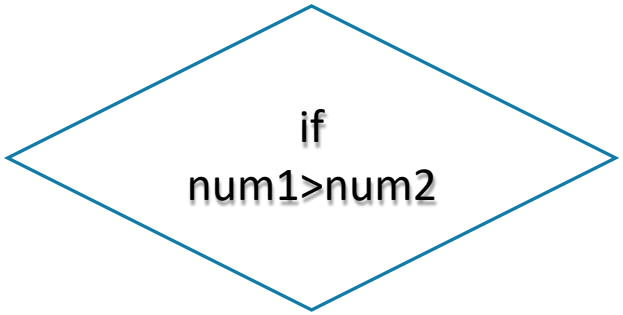


4) **Decision Making**

For checking condition this symbols can be used

Like if num1 is greater than num2

Can be symbolized as



TOPS Technologies

4) **Flow lines**

Lines showing flow of data and process

Showing flow of instructions also

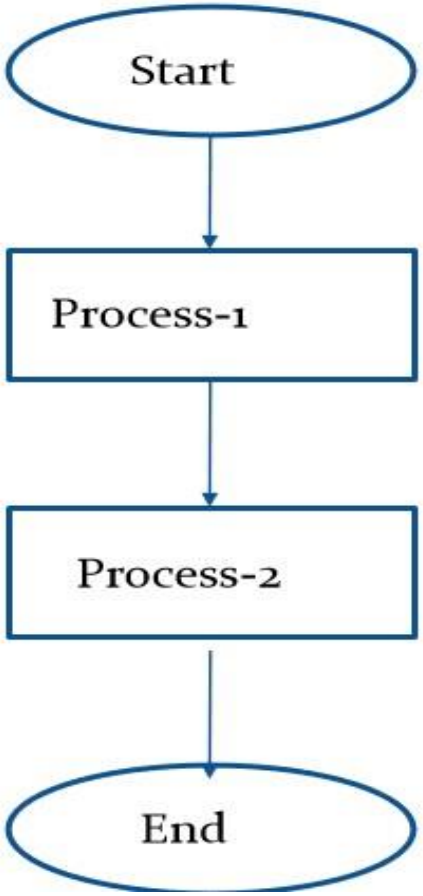
Can be symbolized as

Programs can be in three format

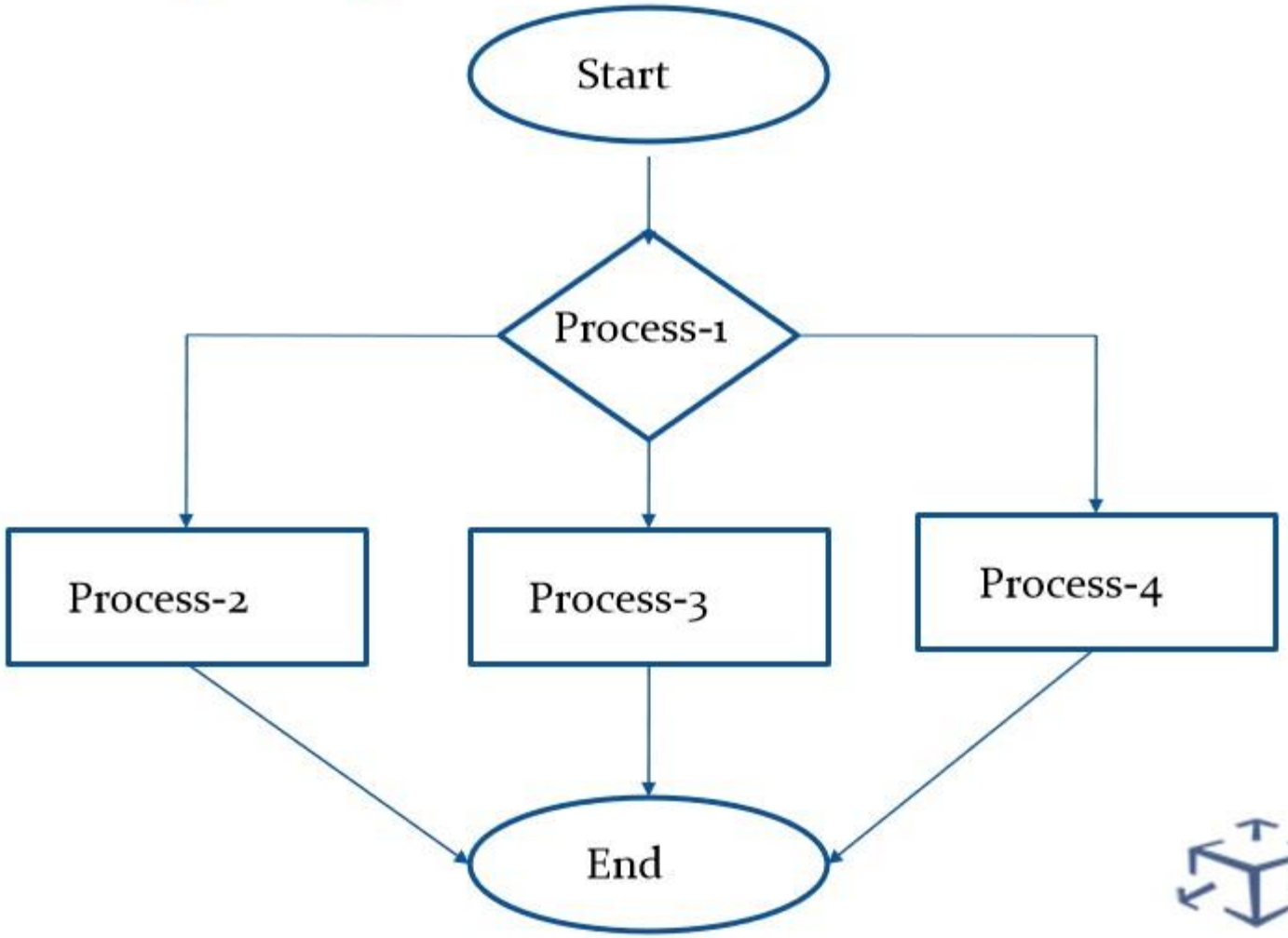
1. Linear or sequence
2. Branching
3. Looping

Following are notations can be used to show this format

Linear Program Structure

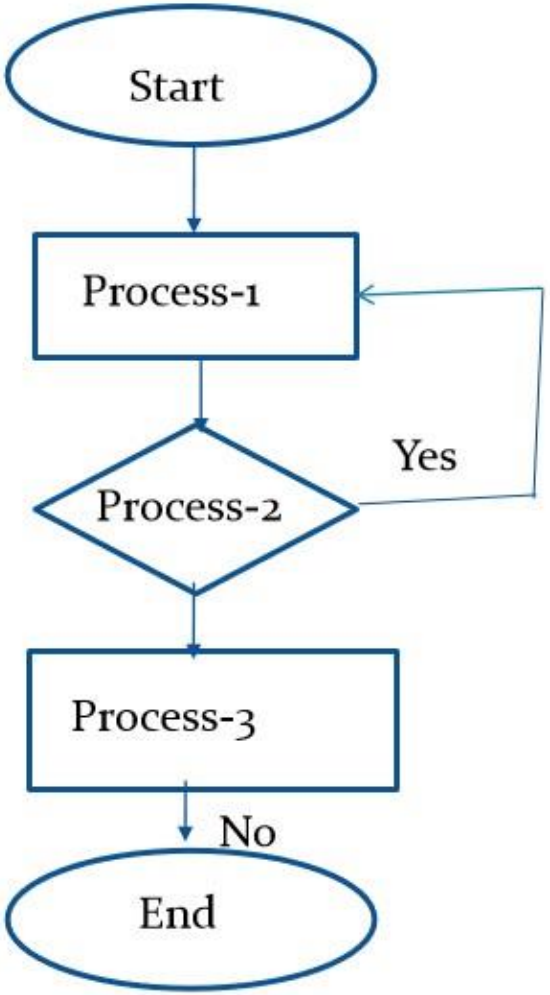


Branching Program Structure..



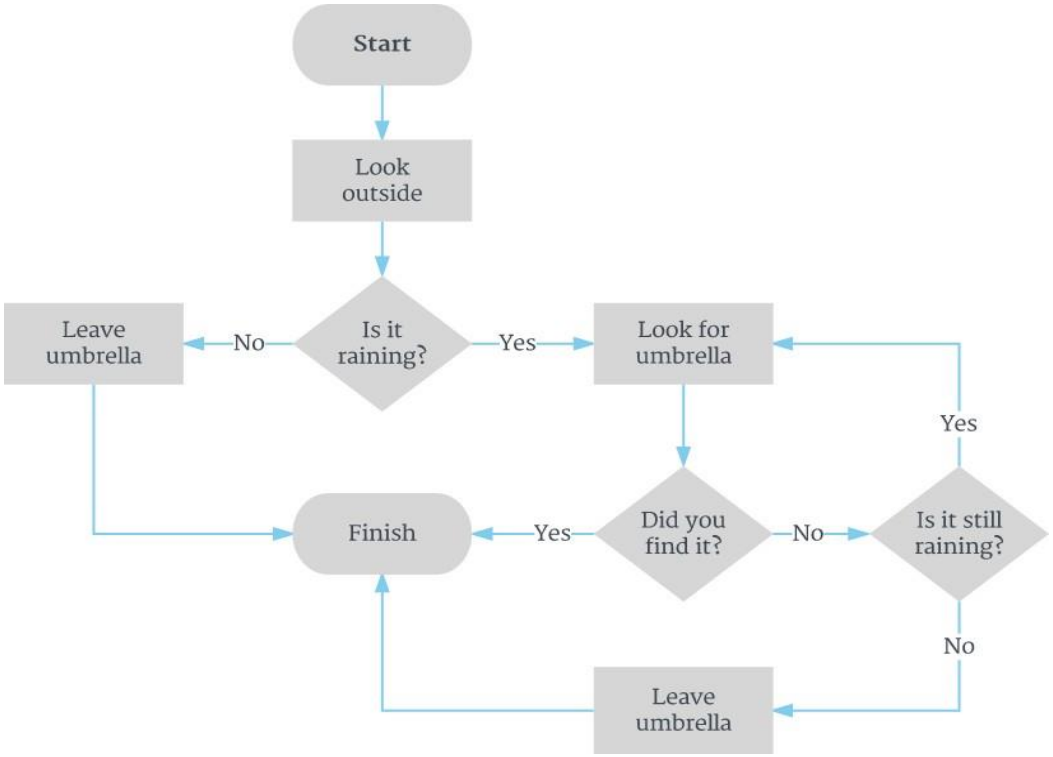
TOPS Technologies

Looping Program Structure



TOPS Technologies

Flow Chart For Taking Umbrella



CLASS DIAGRAM

The class diagrams are a neat way of visualizing the classes in our system before we actually start coding them up.

They're a static representation of the system structure.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system.

The class diagrams are widely used in the modelling of object oriented systems. They are the only UML diagrams, which can be mapped directly with object-oriented languages.

NEED OF CLASS DIAGRAM

Planning and modelling ahead of time make programming much easier.

Besides that, making changes to class diagrams is easy, whereas coding different functionality after the fact is kind of annoying.

When someone wants to build a house, they don't just grab a hammer and get to work. They need to have a blueprint — a design plan — so they can ANALYZE & modify their system.

You don't need much technical/language-specific knowledge to understand it.

CLASS DIAGRAM NOTATIONS:

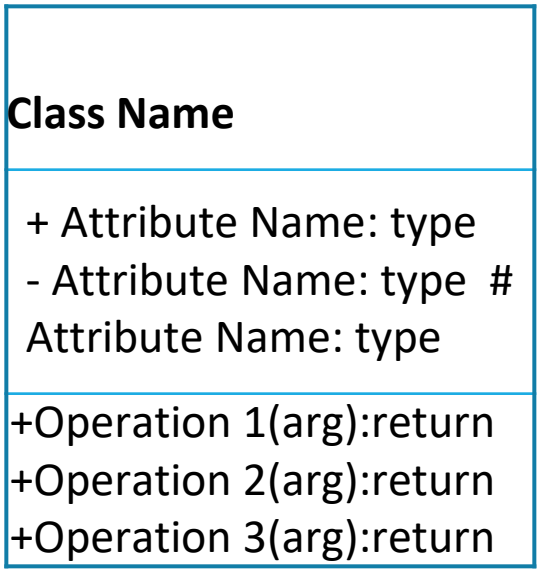
1). **Class**:

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

Class is represented with rectangles with compartments. Class name is placed in the first compartment (centred, bolded, and capitalized).

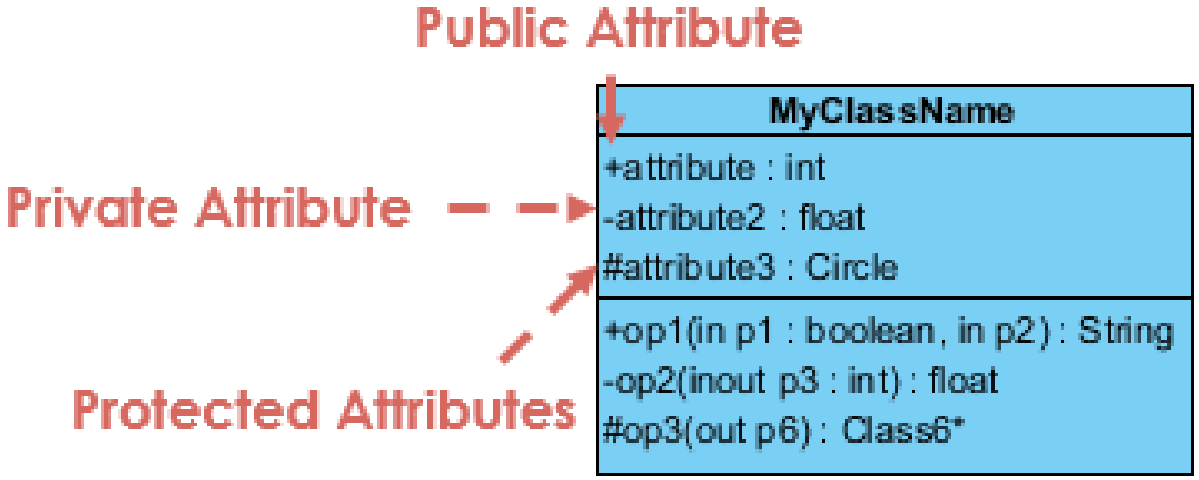
Attributes is placed in the second compartment(left-aligned, not bolded, and lowercase)

The bottom one lists the class's operations, which represents the behaviour of the class.



2). Visibility of class Members

public	+	anywhere in the program and may be called by any object within the system
private	-	the class that defines it
protected	#	(a) the class that defines it or (b) a subclass of that class
package	~	instances of other classes within the same package



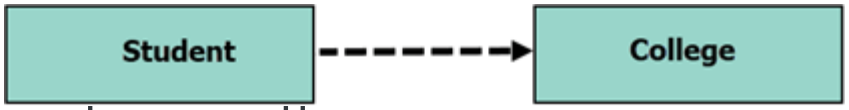
TOPS Technologies

3). Relationship:

a**). Dependency:**

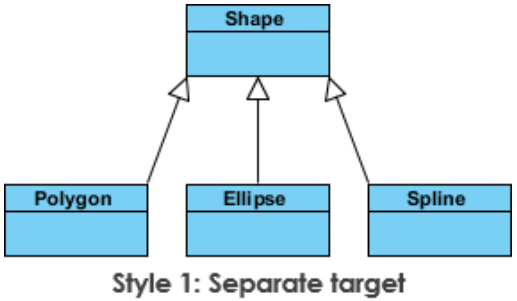
➢Relation between two or more classes in which a change in one may force changes in the other.

➢It will always create a weaker relationsh



➢Dependency indicates that one class depends on another.

b). **Generalization(Inheritance):**



Style 1: Separate target

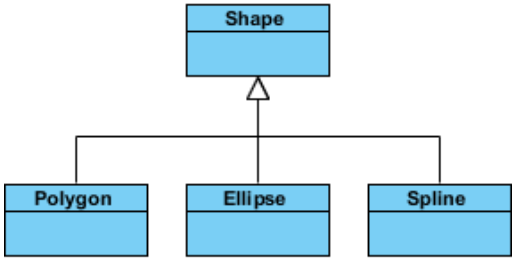
➢It refers to a relationship between two classes class is a specialized version of another

➢Represents is-a relationship. Example" Honda

is a type of car.

So the class Honda would have a generalization

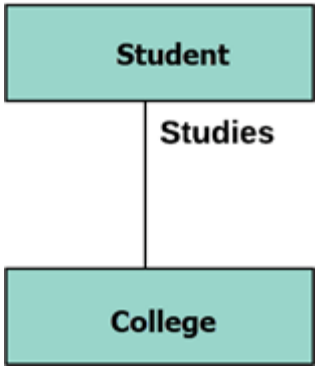
relationship with the class car.



c). Association: ——————— **or** ———————→

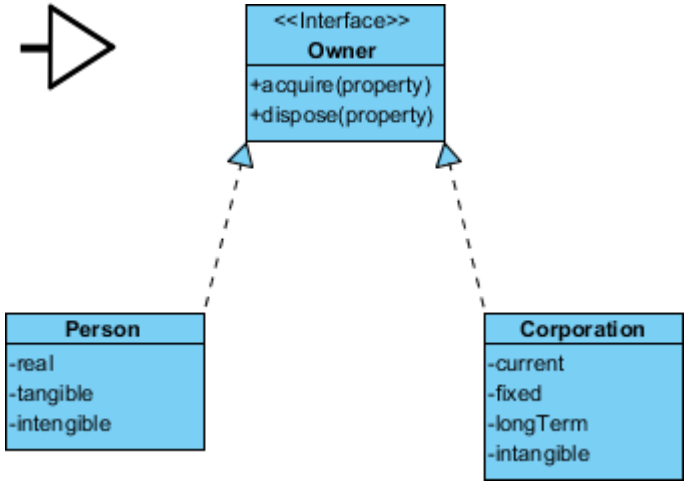
➢This type of relationship represents static relationships between classes A and B.

➢For example; an employee works for an organization.



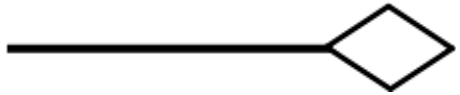
d). **Realization/Implementation** - - - - - - ▷

➢Relationship between two class elements, in which one class element implements/executes the behaviour that the other model element specifies.



TOPS Technologies

e) Aggregation

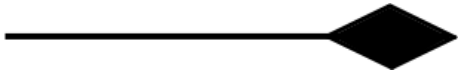


Aggregation is a special type of association that models a whole- part relationship between aggregate and its parts.

Describes has-a relationship.

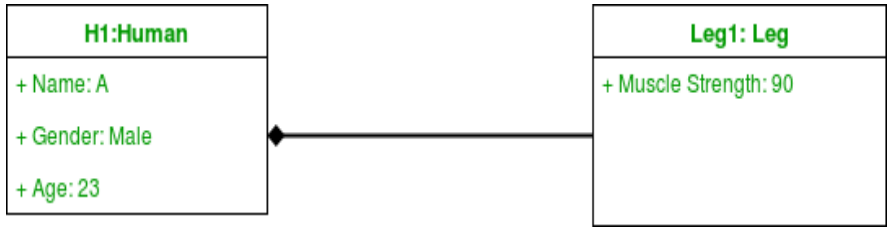
Child can exist without the parent.

f) Composition:



Composition is a type of association where the child cannot exist independent of the other.

Child will never exist independent of a parent



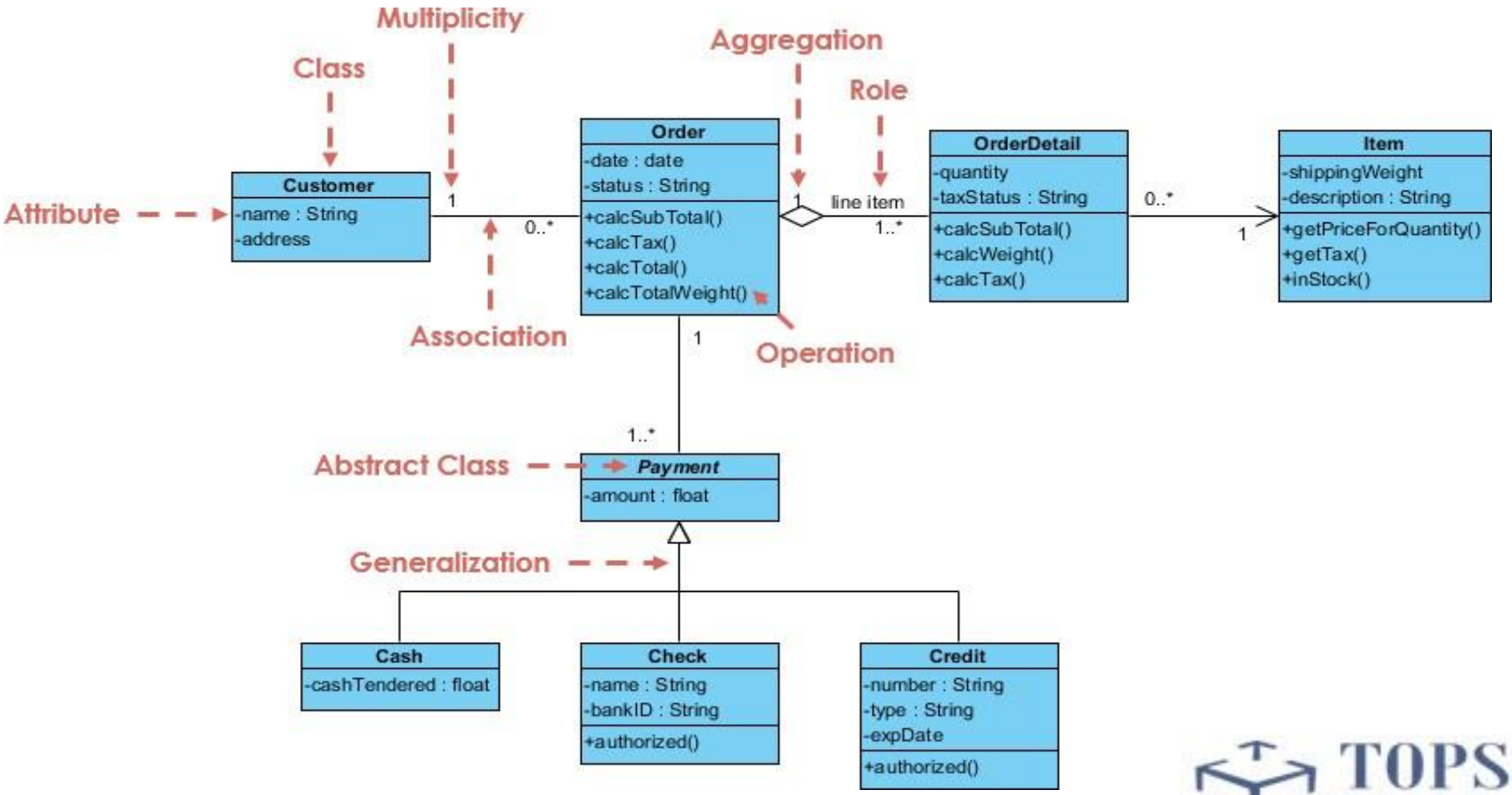
TOPS Technologies

Multiplicity:

- Represent cardinality between associated entities
- It specifies how many instances of attributes are created when a class is initialized. If a multiplicity is not specified, by default one is considered as a default multiplicity.
- Let's say that that there are 100 students in one college. The college can have multiple students.

NOTATION	MEANING
0..1	Zero or one
1	One only
0..*	Zero or more
*	Zero or more
1..*	One or more
7	Seven only
0..2	Zero or two
4..7	Four to seven

Class Diagram for Payment System



TOPS Technologies

Model Design With UML

Unified Modelling Language (UML) is a general purpose modelling language.

The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language, it is rather a visual language. We use UML diagrams to portray the behaviour and structure of a system.

Model

Model is a simplification of reality,. A model may provide

- Blueprint of a system
- Organization of the system
- Dynamic of the system

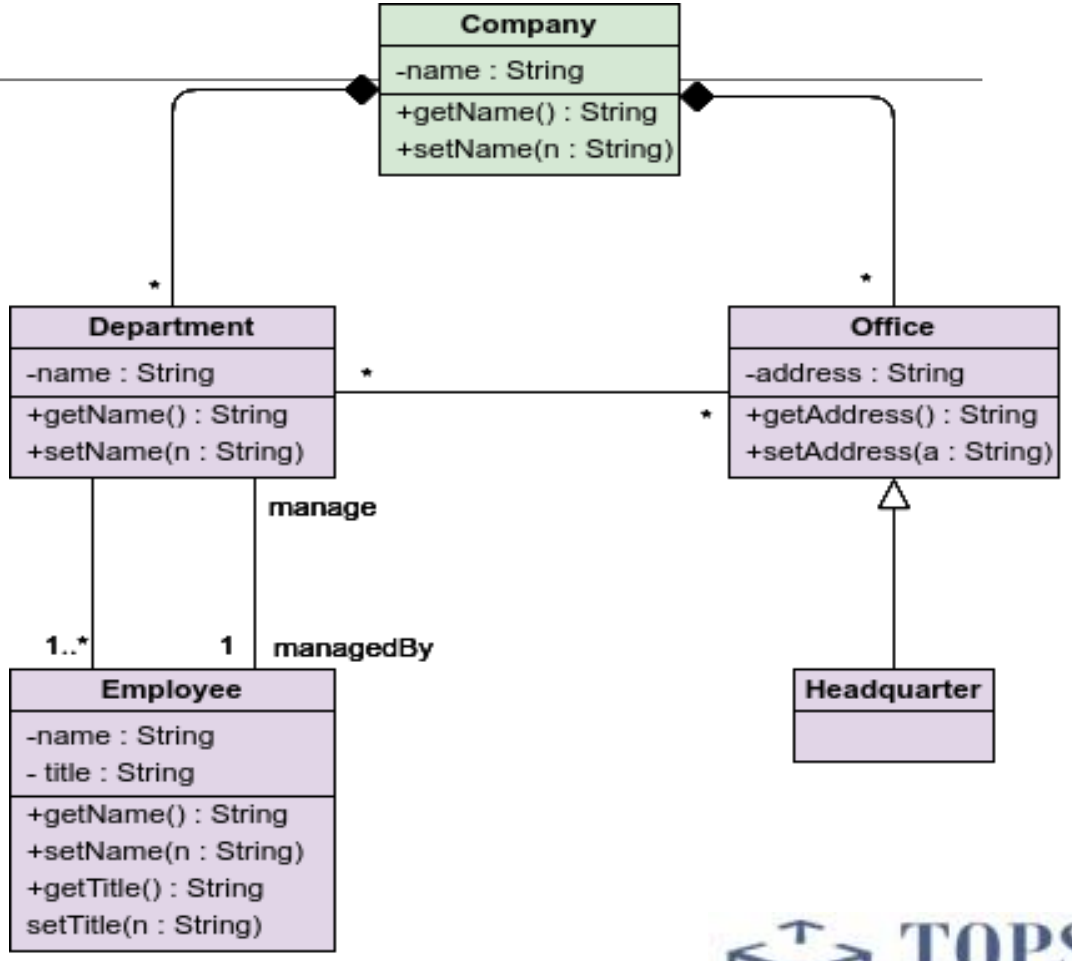
Three kinds of models describe a system from different viewpoints:

1.Class Model
2.State Model
3.Interaction Model

A complete description of a system requires models from all 3 viewpoints

Class Model

- The class model describes the static structure of the objects in a. system and their relationships.
- The class model contains class diagrams
- A class diagram is a graph whose nodes are classes and whose arcs are relationships among classes.



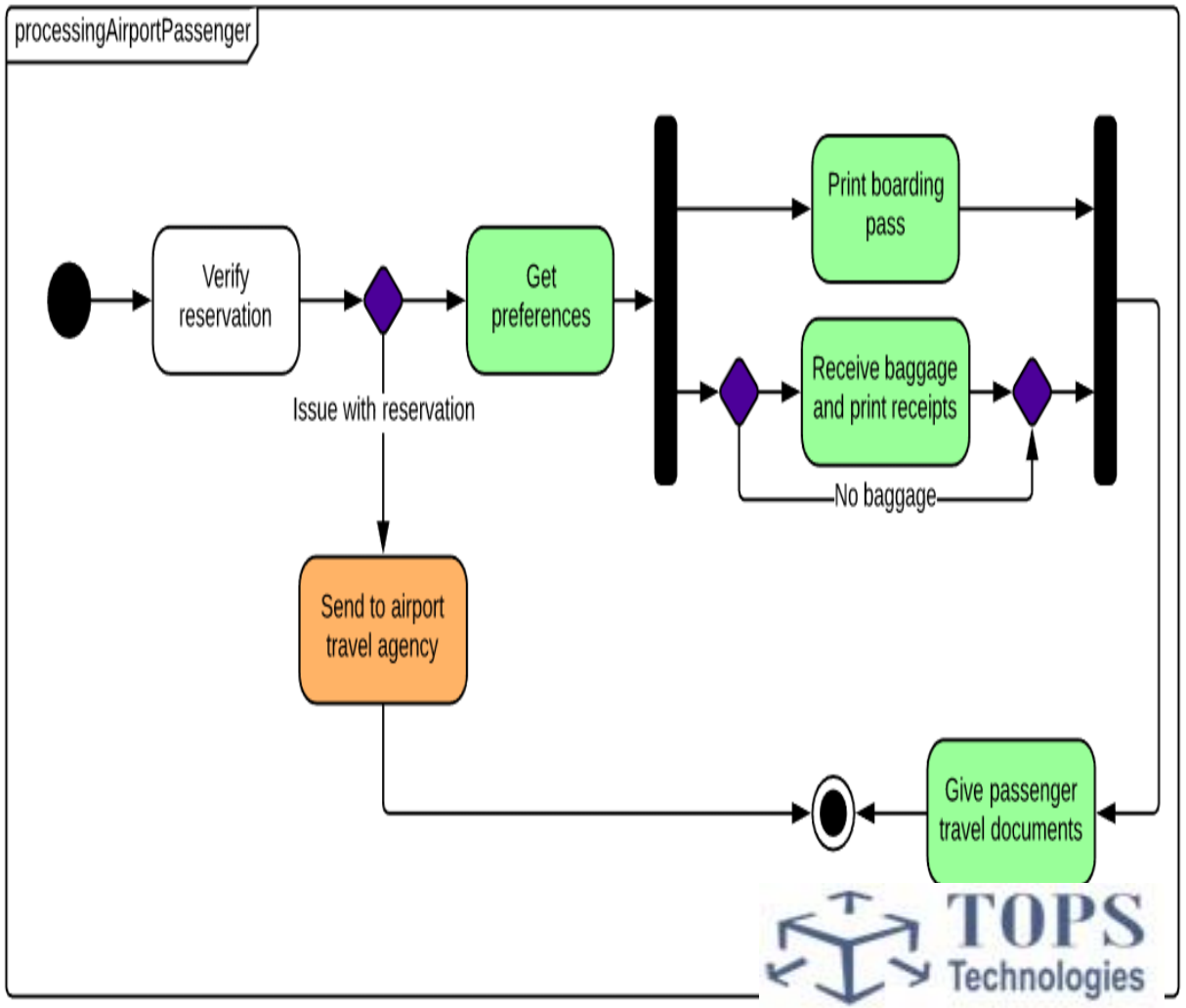
State Model

The state model describes the aspects of an object that change over time.

State diagrams express the state model.

The state diagram is a graph whose nodes are states and whose arcs are transitions between states caused by events.

State diagram for an airport check-in is shown in figure.



Interaction Model

The interaction model describes how the objects in a system cooperate to achieve broader results.

The interaction model starts with use case that are then elaborated with sequence and activity diagrams.

A use case focuses on the functionality of a system i.e, what a system does for users.