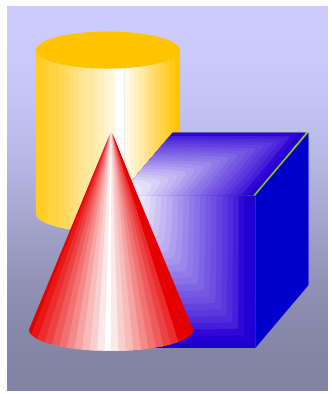
SmartImpart™

Object Orientation

Unit 1

Introduction to Object Oriented Approach

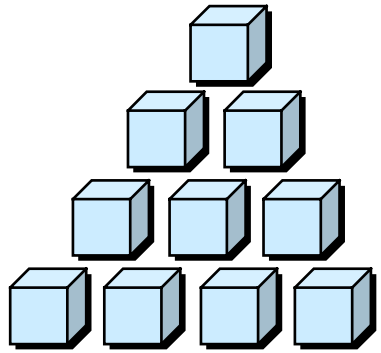


Introduction to Object Oriented Approach

- Topics
  - Procedural approach to Object-Oriented approach
  - What is an Object
  - Why choose the OO approach
  - What is a Class
  - Identifying Classes

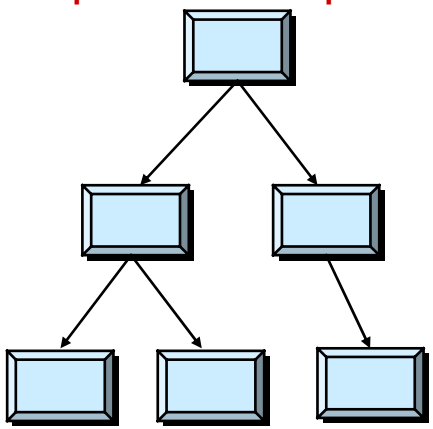
The procedural approach

- The procedural approach
  - Deals with functions as the **building blocks**
  - Easy to start with
  - Higher comfort level for a new programmer
- Simple decomposition technique for
  - Modularity
  - Reusability
  - Complexity



Functions are the building blocks

Top – down decomposition



Fit falls of Structured Programming Approach

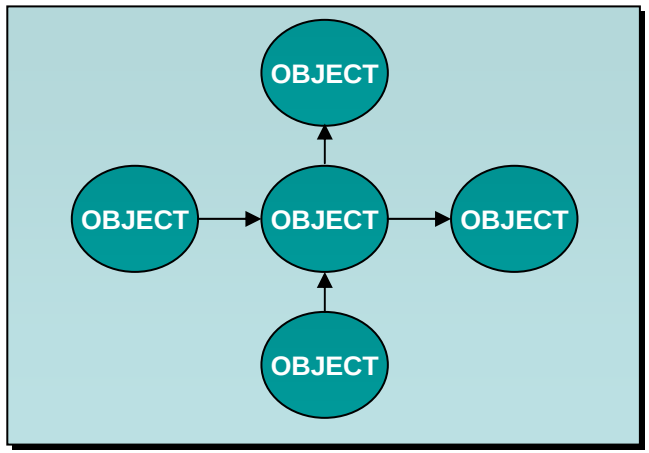
- Programs became harder to maintain.
- Existing functionality was hard to alter without adversely affecting all of the system's functionality.
- New programs were essentially built from scratch. Consequently, there was little return on the investment of previous efforts.
- Programming was not conducive to team development. Programmers had to know every aspect of how a program worked and could not isolate their efforts on one aspect of a system.
- It was hard to translate business models into programming models.
- It worked well in isolation but did not integrate well with other systems.

Fit falls of Structured Programming Approach

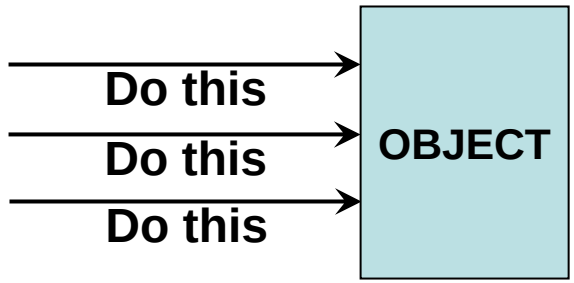
- In addition to these shortcomings, some evolutions of computing systems caused further strain on the structural program approach:
  - Nonprogrammers demanded and were given direct access to programs through the incorporation of graphical user interfaces and their desktop computers.
  - Users demanded a more-intuitive, less-structured approach to interacting with programs.
  - Computer systems evolved into a distributed model where the business logic, user interface, and backend database were loosely coupled and accessed over the Internet and intranets.

OO Approach

- An Object oriented approach views systems and programs as a collection of interacting objects.



- An object is a thing in a computer system that is capable of responding to messages



Benefits of OO Approach

- A more intuitive transition from business analysis models to software implementation models
- The ability to maintain and implement changes in the programs more efficiently and rapidly
- The ability to more effectively create software systems using a team process, allowing specialists to work on parts of the system
- The ability to reuse code components in other programs and purchase
- components written by third-party developers to increase the functionality of their programs with little effort
- Better integration with loosely coupled distributed computing systems
- Improved integration with modern operating systems
- The ability to create a more intuitive graphical user interface for the users

What is Object-Oriented Programming Language

- **Restrictive** – The programming language that supports both object classes and class inheritance.
- **Loose** - Any programming language that provides mechanisms that can be used to exploit encapsulation is (at least to some degree) object-oriented.

OO Programming Languages

- Simula – 1960
- Smalltalk – 1970
- C++ - 1983
- Objective C – 1984
- Java – 1995
- .Net Languages - 2000

Over 200 OO Languages

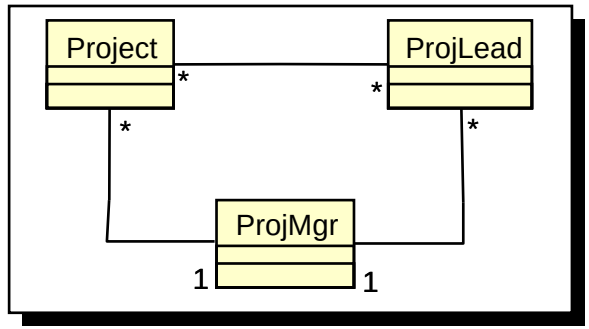
What are Objects?

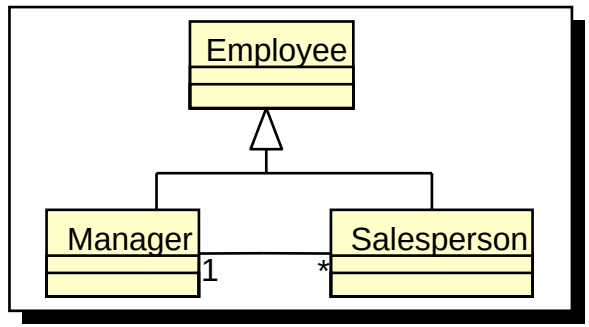
- You interact with *objects* everyday
  - A customer
  - An order
  - Your car
  - The telephone

- An object represents an entity – physical, conceptual or software
  - **Physical entity**
    - Employee, Customer, Supplier
  - **Conceptual entity**
    - Sales, Policy, TaxCalculator
  - **Software entity**
    - Linked List, Expression, Connection, etc.
- ***A programmer should make a good effort to capture the conceptual entities in addition to physical entities which are relatively straight forward to identify***

Why choose the Object Oriented approach?

- The OO approach deals with classes as the building blocks
- Allows Real World Modeling
- The idea of OOP is to try to approach programming in a more natural way by grouping all the code that belongs to a particular object—such as an account or a customer — together
- Raise the level of abstraction
- Applications can be implemented in the same terms in which they are described by users
- Easier to find nouns and construct a system centered around the nouns than actions in isolation





What is a Class?

A **class** gives a generic notion of what objects look like

Employee
Class

Instantiate (create)

An employee
Joe

An **object** is an instance of a class

Classes are used to distinguish one type of object from another

Employee

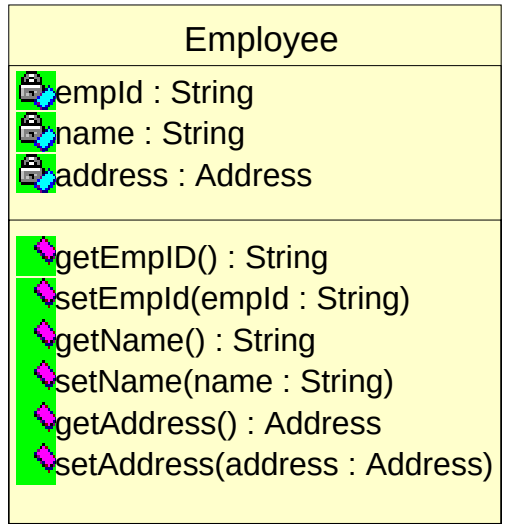
Reg. Customer

Customer

Types

Class

- User defined type
  - Encapsulates all the data and operations pertaining to an entity
  - Provided a Single representation to all the attributes defining the entity
  - Passing single representations is easier



- Data types as collections
  - A struct in C encapsulates only data. Used as a data structure to store different types of data
  - An array is used to store different elements of the same type

Identifying Classes

A trainer trains many trainees on a given technology in this course, which contains many modules – each module is comprised of different units and each unit has many topics.

- Identify the different classes from the above problem statement

Procedural approach

- Focus is on identifying **VERBS**
- Connections between functions established through Function Calls

OO approach

- Focus is on identifying **NOUNS**
- Connections between classes established through Relationships ('Is-a' and 'Has-a')

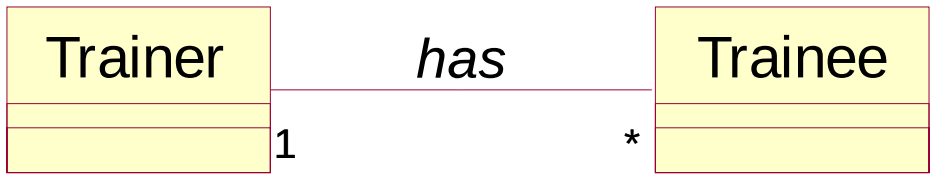
Identifying Classes

- Trainer
- Trainee
- Course
- Technology
- Module
- Unit
- Topic

- Identify the different connections (relationships) between the above classes

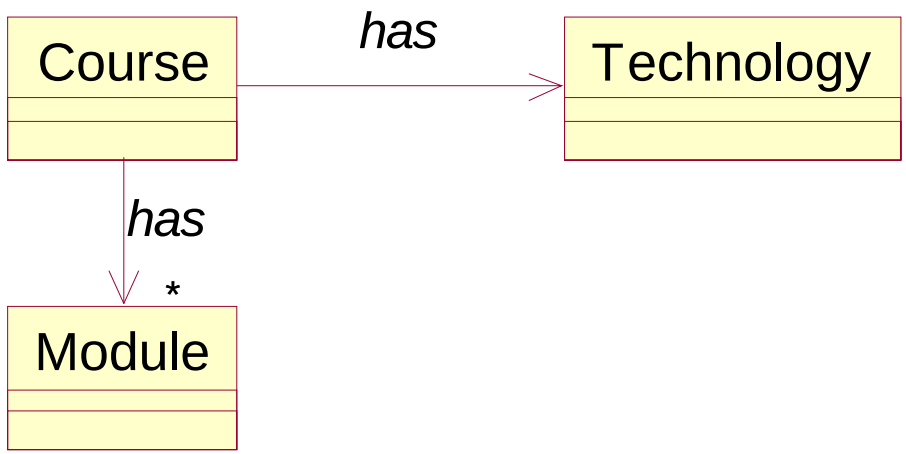
Identifying Relationships

- Trainer - Trainee
  - Trainer ‘HAS’ many Trainees
  - Every Trainee ‘HAS’ a Trainer



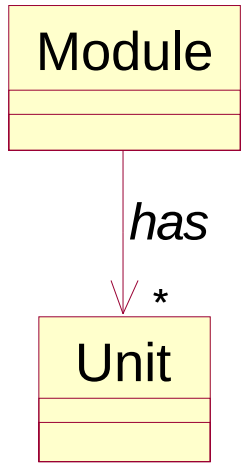
Identifying Relationships

- Course – Technology
- Course - Module
  - Course ‘HAS’ an associated Technology
  - Course ‘HAS’ many Modules

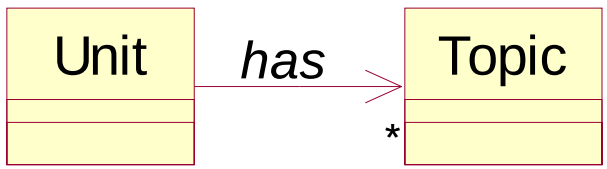


Identifying Relationships

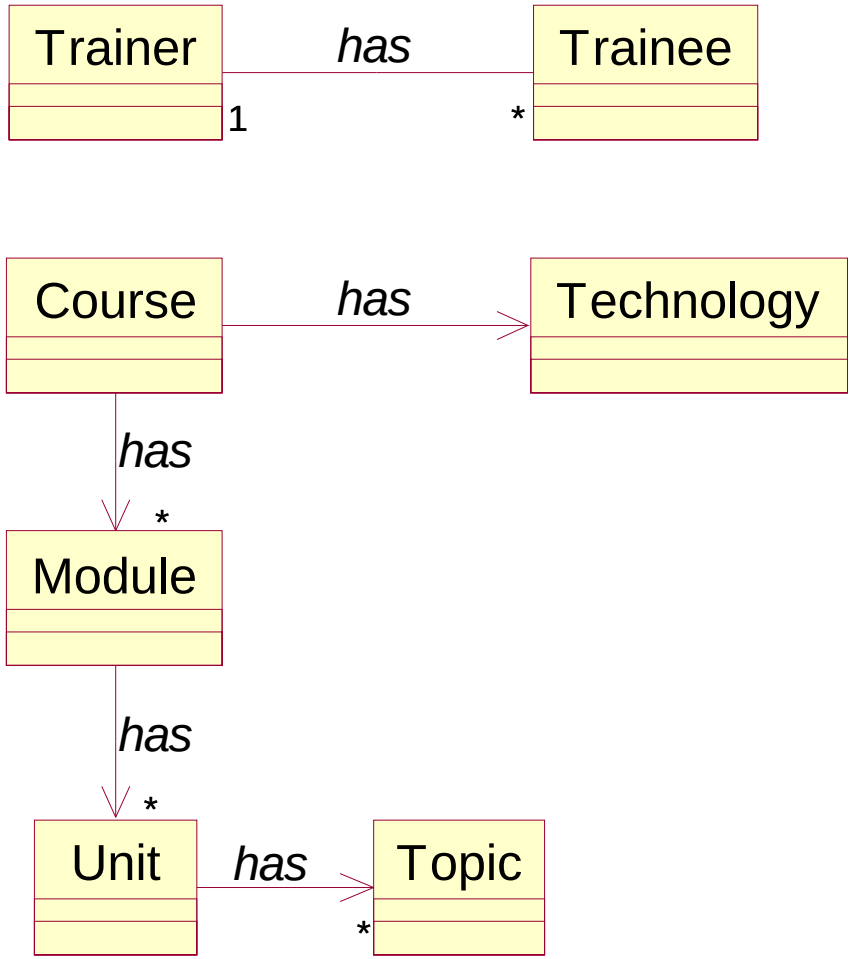
- Module – Unit
  - Module ‘HAS’ many Units



- Unit – Topic
  - Unit ‘HAS’ many Topics



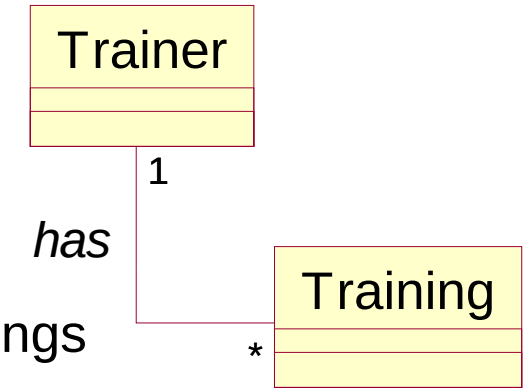
The OO Model



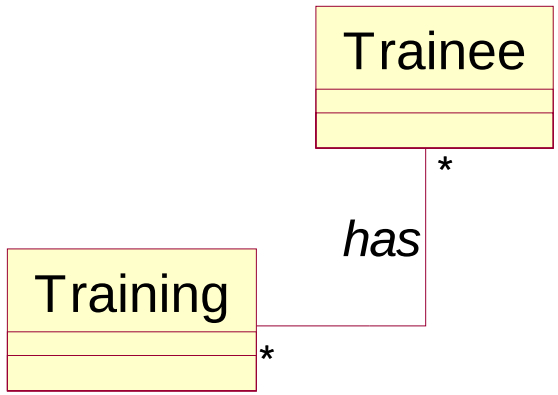
- How do you relate the Trainer & Trainee to the Course?

Conceptual Entity

- Trainer – Training
  - A Trainer (HAS) conducts many Trainings
  - A Training HAS a Trainer

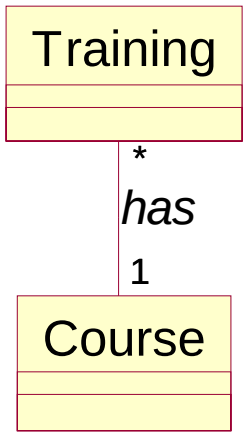


- Trainee – Training
  - A Trainee (HAS) attends many Trainings
  - A Training HAS a many Trainees

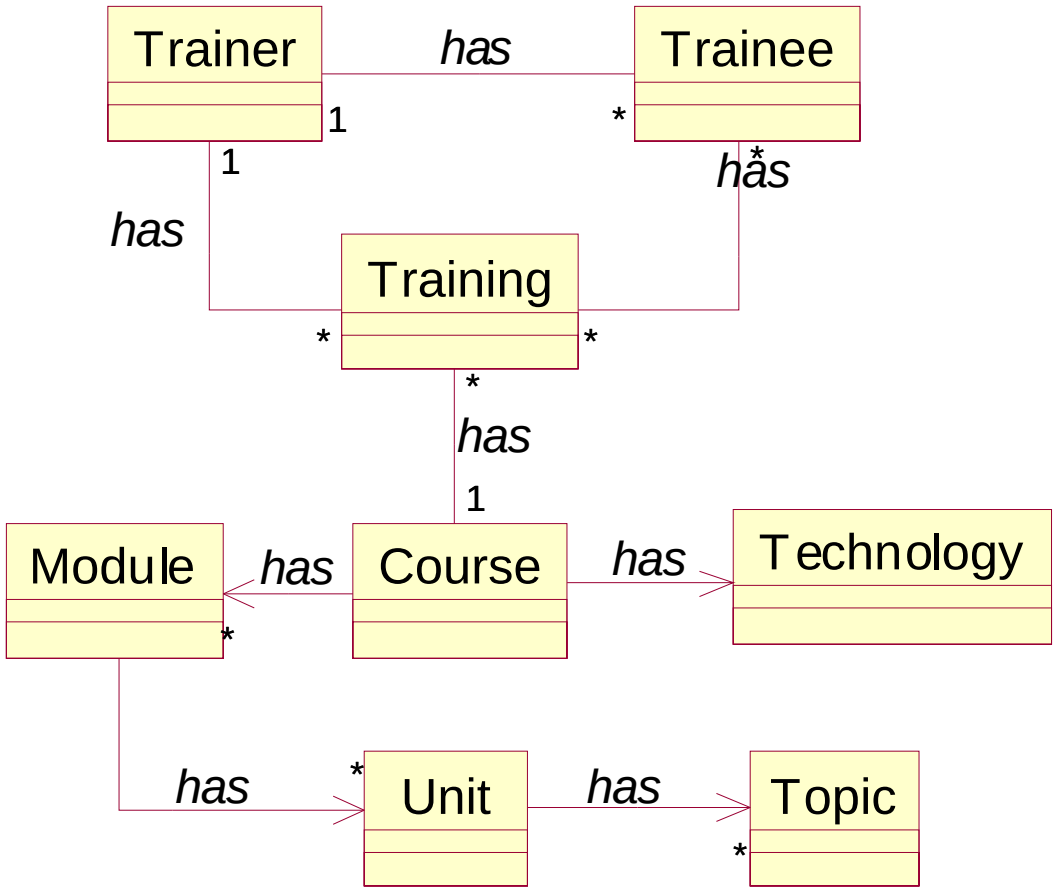


Conceptual Entity

- Training - Course
  - The Training (HAS) an association with a Course (conducted for a Course)
  - A Course HAS many Trainings



Solution



- Easier to model real-world problems through the OO approach than through the procedural approach

Exercise

A company sells different items to customers who have placed orders. An order can be placed for several items. However, a company gives special discounts to its registered customers.

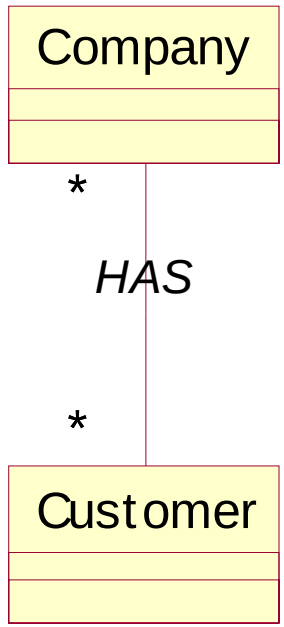
- Identify the different classes from the above problem statement
- Identify the different connections (relationships) between the above classes

Identifying Classes

- Company
- Item
- Order
- Customer
- RegCustomer

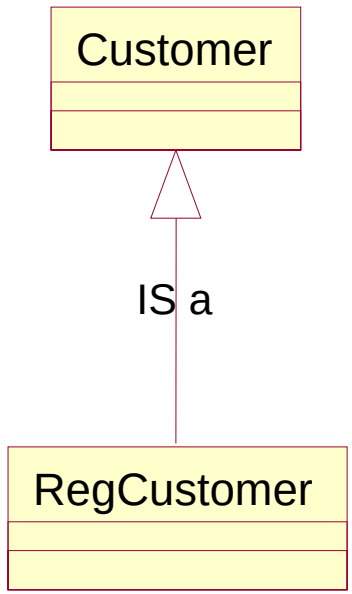
Identifying Relationships

- Company - Customer
  - Company ‘HAS’ many Customers
  - Customer ‘HAS’ many Companies



Identifying Relationships

- Customer - RegCustomer
  - RegCustomer ‘IS’ a Customer

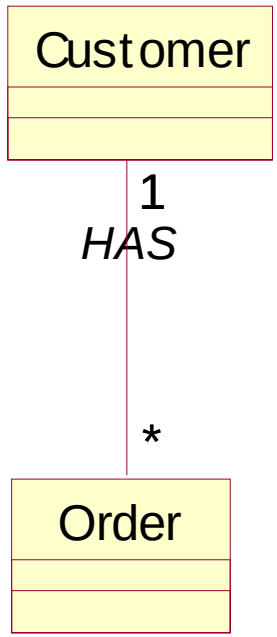


Identifying Relationships

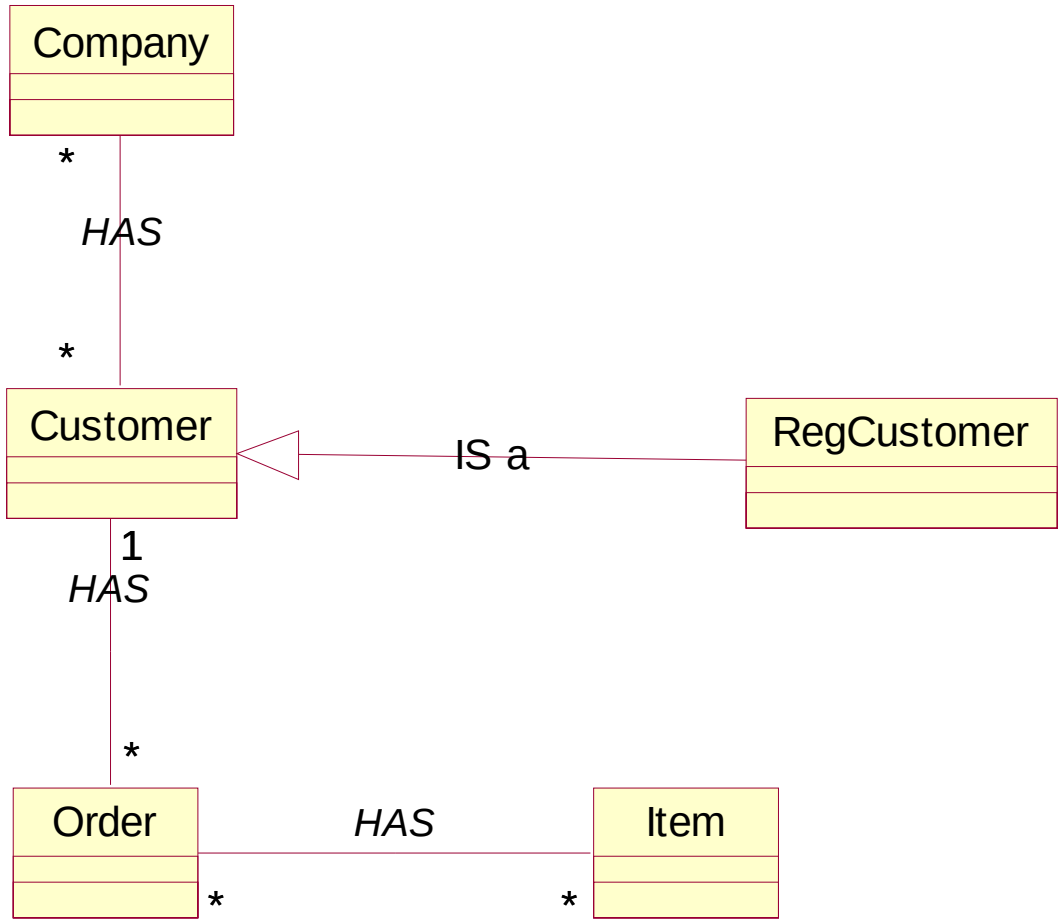
- Item - Order
  - Order HAS many Items
  - Item HAS many Orders

Identifying Relationships

- Customer - Order
  - Customer HAS many Orders
  - Order HAS one Customer



The OO Model



Question Time

Please try to limit the questions to the topics discussed during the session.

Participants can clarify other doubts during the breaks.

Thank you.