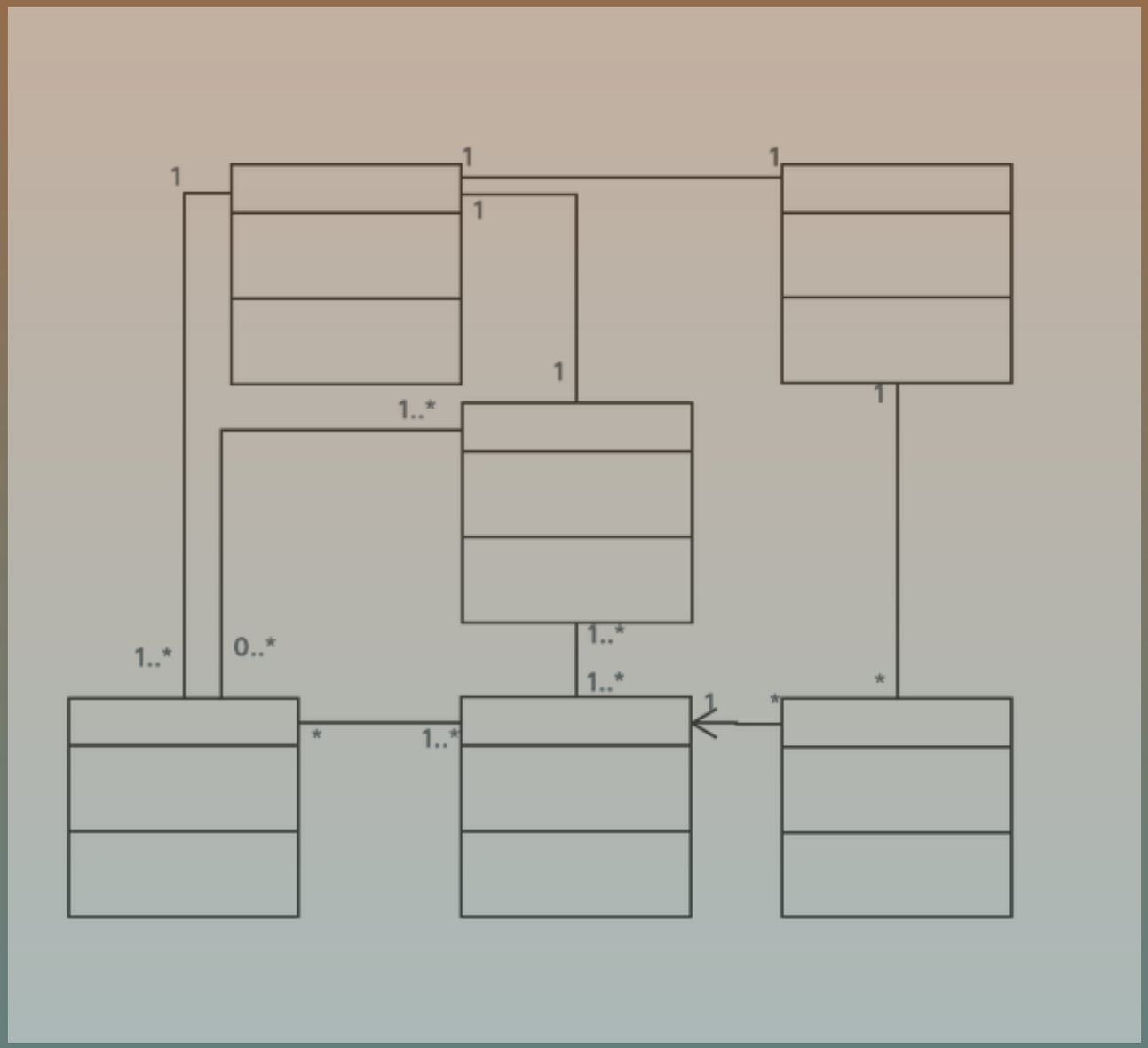
SOFTWARE ENGINEERING CONCEPTS

UML DIAGRAMS



DEFINITION OF UML

WHAT IS IT AND WHY DO WE NEED IT ?

- UML is the abbreviation of Unified Modeling Language, which provides a set of several diagrams to model and document software systems visually.

- UML diagrams are able to represent the whole software system along with the roles, actions, classes or objects.

- UML diagrams can be classified into three categories

    1. Behavior Diagrams
    2. Interaction Diagrams
    3. Structure Diagrams

THREE CATEGORIES OF UML DIAGRAMS

- **Behavior Diagrams:** A type of diagram that depicts behavioral features of a software system. Activity, state machine, use case diagrams and all the types of interaction diagrams.
- **Interaction Diagrams:** A subset of behavior diagrams which emphasize object interactions. Communication, Interaction overview, sequence and timing diagrams
- **Structure Diagrams:** A type of diagram that depicts the elements of a specification that are irrespective of time. Class, composite structure, component, deployment, object and package diagrams.

CATEGORIES OF UML DIAGRAMS

- Activity Diagram
- Class Diagram
- Communication Diagram
- Component Diagram
- Composite Structure Diagram
- Deployment Diagram
- Interaction Overview Diagram
- Object Diagram
- Package Diagram
- Sequence Diagram
- State Machine Diagram
- Timing Diagram
- Use Case Diagram

CLASS DIAGRAMS

THE ESSENTIAL BUILDING BLOCK

- Majority of the software systems in the industry are planned, designed and created based on object oriented programming principles.

- Class diagrams help us model our software systems by displaying all classes of the system, all the relationships between objects, operations and attributes of the objects.

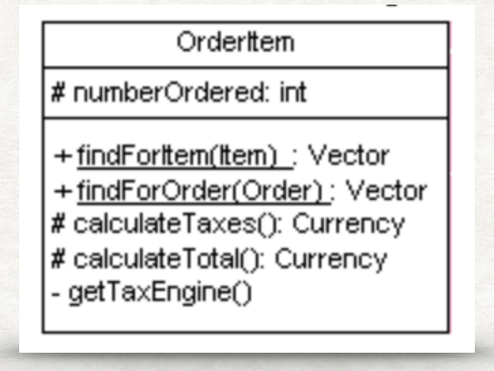
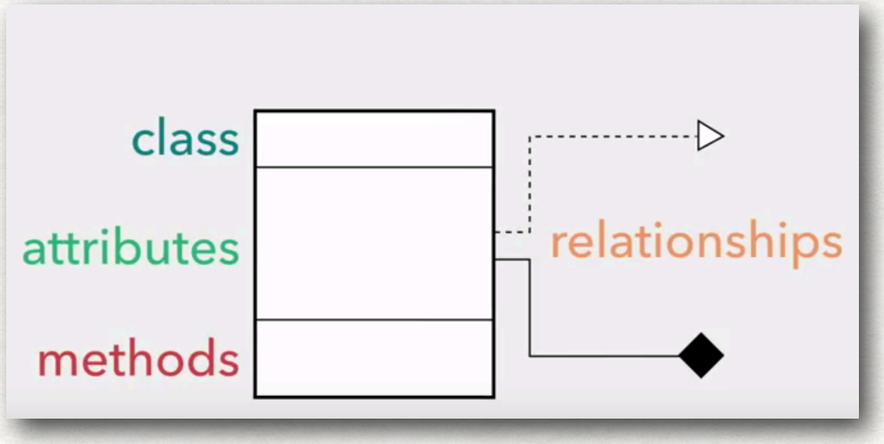


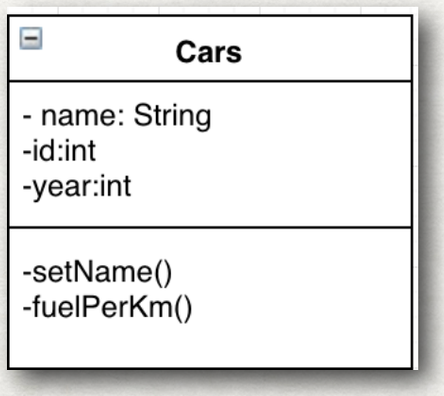
Figure 1. A simple class diagram

CLASS DIAGRAMS

INNER PARTS OF THE CLASS DIAGRAM



- **Class:** Name of the class (e.g. Cars, and brand name for the cars are the objects, instances of Cars class.)
- **Attributes:** The characteristics, defining values of a class, in other words properties, fields etc. e.g. name, id, year etc.
- **Methods:** Operations or functions of a class, specify behavioral feature of a class. e.g. setName or fulePerKm.



VISIBILITIES

AVAILABILITY OF ATTRIBUTES AND METHODS

- - (minus sign) is used for private(only available within class)
- +(plus sign) is used for public (can be accessed by any other class)
- #(hash sign) is used for protected (can be accessed by same class or subclasses)
- ~(tilda sign) is used for default (available in the same package)

RELATIONSHIPS

- **Inheritance:** Is-a relationship between super class and subclasses.
- **Association:** a relationship where all objects have their own lifecycle and there is no owner.
- **Aggregation:** is a specialized form of Association where all objects have their own lifecycle, but there is ownership and child objects can not belong to another parent object. (Has-a relationship.)
- **Composition:** It is a strong type of Aggregation. Child object does not have its lifecycle and if parent object is deleted, all child objects will also be deleted.