The UML Class diagram serves as a graphical notation for designing and visualizing object-oriented systems. This static structure diagram within the Unified Modelling Language (UML) delineates the structure of a system by displaying:

* Classes
* Attributes
* Operations (or methods)
* Relationships among objects

***Relationship between classes***

A diagram of a company

Description automatically generated

**Association**

***Definition:*** Represents a relationship between two or more classes where objects of one class are connected to objects of another.

***Representation:*** Solid line connecting the classes.

***Naming:*** Typically labeled with a verb or verb phrase to describe the relationship.

***Example:*** A Teacher class is associated with a Classroom class.

***Cardinality:*** Specifies the number of instances involved in the relationship.

***One-to-One:*** Each instance of a class is associated with one instance of another class.

***One-to-Many:*** An instance of one class is associated with multiple instances of another class.

***Many-to-Many:*** Instances of both classes are associated with multiple instances of each other.

**Inheritance (Generalization)**

***Definition:*** A relationship where one class (subclass) inherits attributes and operations from another class (superclass).

***Representation:*** Solid line with a hollow arrowhead pointing from the subclass to the superclass.

***Purpose:*** Represents an "is-a" relationship, indicating that the subclass is a type of the superclass.

***Example:*** Bird class is a general class, and Sparrow class is a specific class that inherits from Bird.

***Details:***

***Abstract Class:*** Often a superclass that cannot be instantiated on its own.

***Subclass:*** Inherits properties and behaviors from the superclass and can add new features or override existing ones.

**Realization**

***Definition:*** A relationship between an interface and the class that implements the interface.

***Representation:*** Dashed line with a solid arrowhead pointing from the implementing class to the interface.

***Purpose:*** Indicates that a class fulfills the contract defined by an interface.

***Example:*** Person class realizes the Owner interface by implementing methods such as acquireProperty() and disposeOfProperty().

***Explanation:***

***Interface:*** Defines a set of operations that a class must implement.

***Implementing Class:*** Provides concrete implementations for the operations defined in the interface.

**Dependency**

***Definition:*** Indicates that a class relies on another class for some aspect of its functionality.

***Representation:*** Dashed line with an open arrow pointing from the dependent class to the class it depends on.

***Purpose:*** Shows a "uses" relationship, where one class uses or depends on the services of another class.

***Example:*** Person class depends on Book class if it has a method like read(Book).

***Explanation:***

***Direction:*** Changes in the dependent class might require changes in the class it depends on.

***Temporary Relationship:*** Often represents a temporary link, such as a method parameter or a local variable.

**Aggregation**

***Definition:*** A special type of association that represents a whole-part relationship between the aggregate (whole) and a constituent part.

***Representation:*** Solid line with an unfilled diamond at the end connecting to the whole.

***Purpose:*** Indicates that one class is a part of another, but can exist independently.

***Example:*** Library class aggregates Book classes; books can exist outside of the library.

***Lifetimes:*** The part can exist independently of the whole.

***Example:*** A Library can have many Books, but Books can also exist outside the Library.

**Composition**

***Definition:*** A stronger form of aggregation where the part is strictly bound to the whole.

***Representation:*** Solid line with a filled diamond at the end connecting to the whole.

***Purpose:*** Indicates a whole-part relationship where parts cannot exist independently of the whole.

***Example:*** House class is composed of Room classes; rooms do not exist outside the house.

***Lifetimes:*** When the whole is destroyed, its parts are also destroyed.

***Example:*** A Car class is composed of Engine and Wheel classes; if the car is scrapped, so are its parts.

Reference : <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>