Unified Modeling Language (UML) | Object Diagrams

An **Object Diagram** can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behaviour when objects have been instantiated, we are able to study the behavior of the system at a particular instant. Object diagrams are vital to portray and understand functional requirements of a system.
In other words, “An object diagram in the Unified Modeling Language (UML), is a diagram that shows a **complete or partial view** of the structure of a modeled system **at a specific time.**”
Difference between an Object and a Class Diagram –
An object diagram is similar to a class diagram except it shows the instances of classes in the system. We depict actual classifiers and their relationships making the use of class diagrams. On the other hand, an Object Diagram represents specific instances of classes and relationships between them at a point of time.

Object Diagrams use **real world examples** to depict the nature and structure of the system at a particular **point in time**. Since we are able to use data available within objects, Object diagrams provide a **clearer view** of the relationships that exist **between objects**.

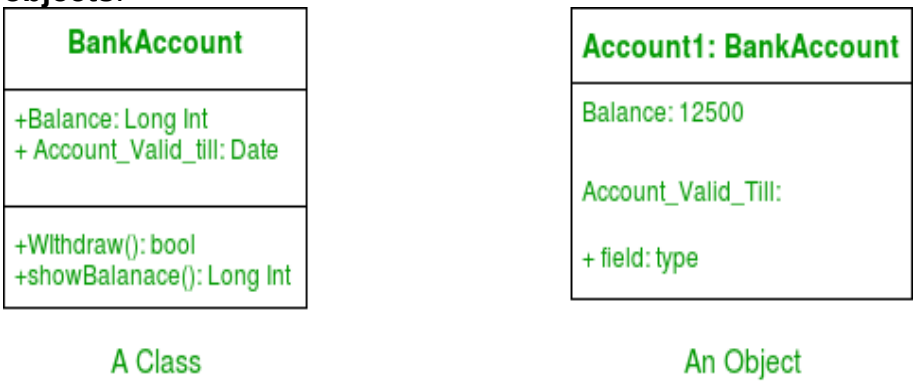

Figure – a class and its corresponding object

Notations Used in Object Diagrams –

1. **Objects or Instance specifications –** When we instantiate a classifier in a system, the object we create represents an entity which exists in the system.We can represent the changes in object over time by creating multiple instance specifications. We use a rectangle to represent an object in an Object Diagram. An object is generally linked to other objects in an object diagram.

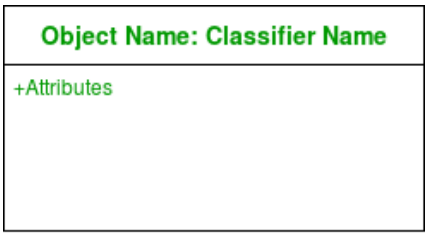

Figure – notation for an Object

For example – In the figure below, two objects of class Student are linked to an object of class College.

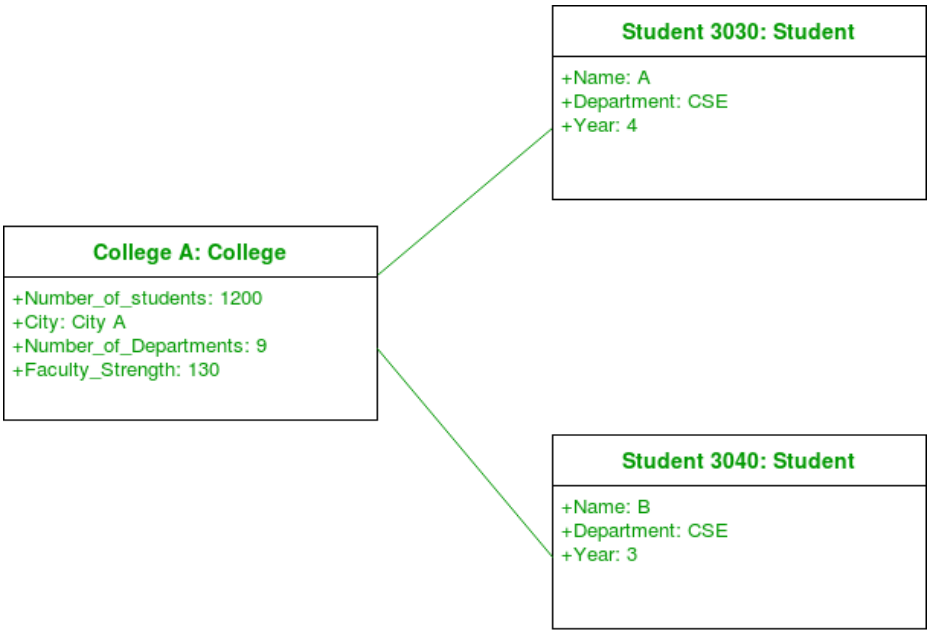

Figure – an object diagram using a link and 3 objects

2. **Links –** We use a link to represent a relationship between two objects.

Figure – notation for a link

We represent the number of participants on the link for each end of the link. We use the term association for a relationship between two classifiers. The term link is used to specify a relationship between two instance specifications or objects. We use a solid line to represent a link between two objects.

NOTATION	MEANING
0..1	Zero or one

NOTATION	MEANING
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

3. **Dependency Relationships –** We use a dependency relationship to show when one element depends on another element.

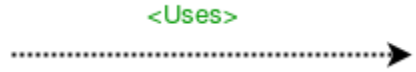

Figure – notation for dependency relationship

Class diagrams, component diagrams, deployment and object diagrams use dependency relationships. A dependency is used to depict the relationship between dependent and independent entities in the system.Any change in the definition or structure of one element may cause changes to the other. This is a unidirectional kind of relationship between two objects.
Dependency relationships are of various types specified with keywords (sometimes within angular brackets”).

Abstraction,Binding, Realization, Substitution and Usage are the types of dependency relationships used in UML.
For example – In the figure below, an object of Player class is dependent (or uses) an object of Bat class.

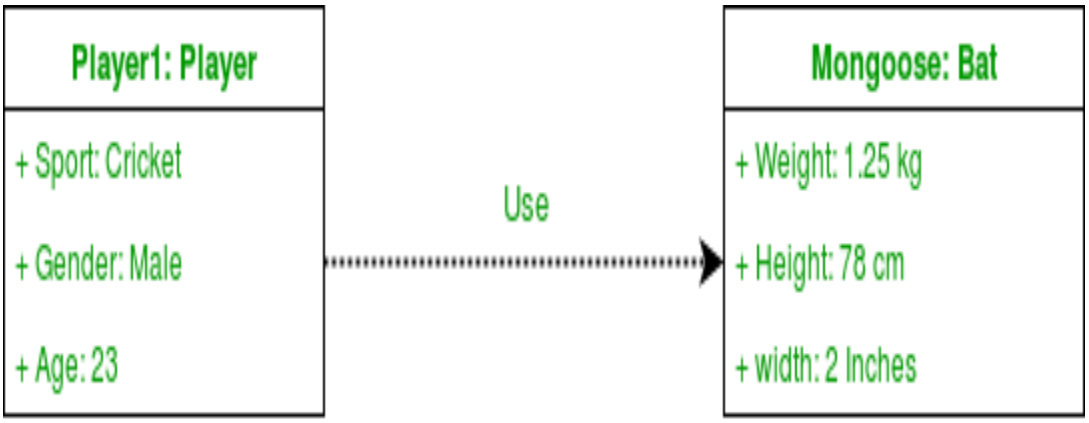

Figure – an object diagram using a dependency relationship

4. **Association –** Association is a reference relationship between two objects (or classes).

Figure – notation for association

Whenever an object uses another it is called an association.We use association when one object references members of the other object. Association can be uni-directional or bi-directional. We use an arrow to represent association.
For example – The object of Order class is associated with an object of Customer class.

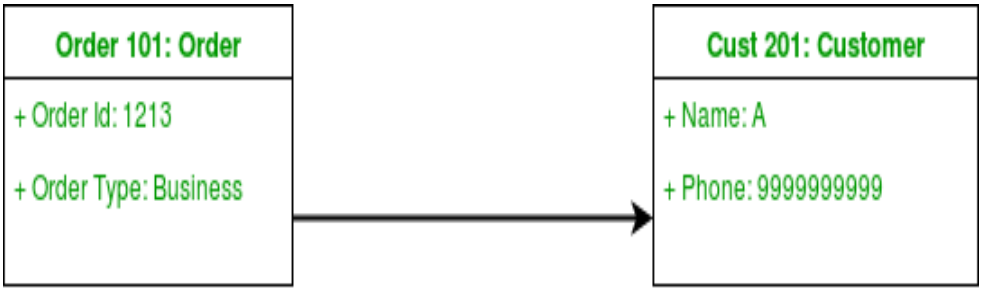

Figure – an object diagram using association

5. **Aggregation –** Aggregation represents a “has a” relationship.

Figure – notation for aggregation

Aggregation is a specific form of association.on relationship; aggregation is more specific than ordinary association. It is an association that represents a part-whole or part-of relationship. It is a kind of parent -child relationship however it isn’t inheritance. Aggregation occurs when the lifecycle of the contained objects does not strongly depend on the lifecycle of container objects.

Figure – an object diagram using aggregation
For example – A library has an aggregation relationship with books. Library has books or books are a part of library. The existence of books is independent of the existence of the library.While implementing, there isn't a lot of difference between aggregation and association. We use a hollow diamond on the containing object with a line which joins it to the contained object.

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

Figure – notation for composition

Composition is also a special type of association. It is also a kind of parent child relationship but it is not inheritance. Consider the example of a boy Gurkaran: Gurkaran is composed of legs and arms. Here Gurkaran has a composition relationship with his legs and arms. Here legs and arms cant exist without the existence of their parent object. So whenever independent existence of the child is not possible we use a composition relationship. We use a filled diamond on the containing object with a line which joins it to the contained object.

Figure – an object diagram using composition
For Example – In the figure below, consider the object Bank1. Here an account cannot exist without the existence of a bank.

Figure – a bank is composed of accounts

Difference between Association and Dependency –

Association and dependency are often confused in their usage. A source of confusion was the use of transient links in UML 1. Meta-models are now handled differently in UML 2 and the issue has been resolved.

There are a large number of dependencies in a system. We only represent the ones which are essential to convey for understanding the system. We need to understand that every association implies a dependency itself. We , however, prefer not to draw it separately. An association implies a dependency similar to a way in which generalization does.

How to draw an Object Diagram?

1. Draw all the necessary class diagrams for the system.
2. Identify the crucial points in time where a system snapshot is needed.
3. Identify the objects which cover crucial functionality of the system.
4. Identify the relationship between objects drawn.

Uses of an Object Diagram –

- Model the static design(similar to class diagrams) or structure of a system using prototypical instances and real data.
- Helps us to understand the functionalities that the system should deliver to the users.
- Understand relationships between objects.
- Visualise, document, construct and design a static frame showing instances of objects and their relationships in the dynamic story of life of a system.
- Verify the class diagrams for completeness and accuracy by using Object Diagrams as specific test cases.
- Discover facts and dependencies between specific instances and depicting specific examples of classifiers.