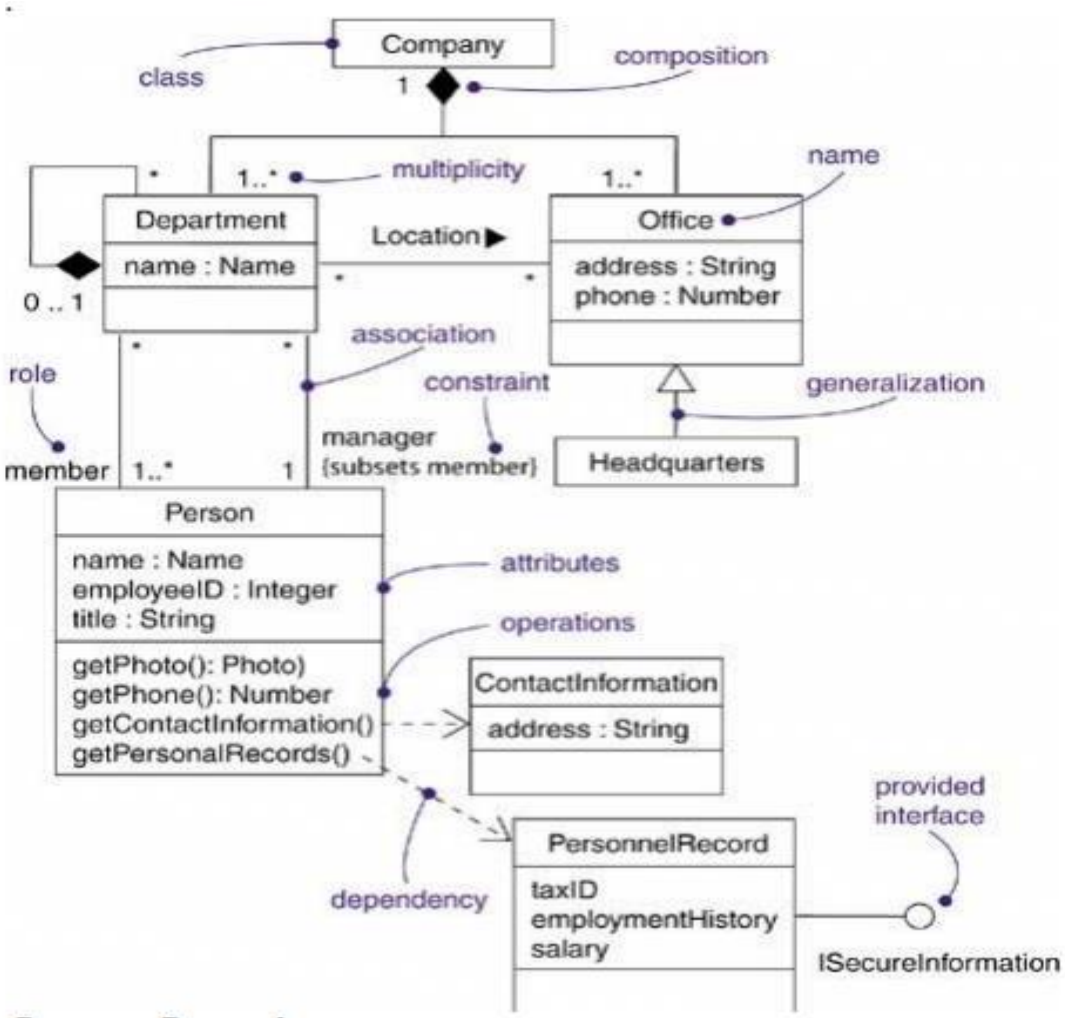
UNIT- III CLASS & OBJECT DIAGRAMS

Class Diagrams:

Building software has much the same characteristics except that, given the fluidity of software, we have the ability to define your own basic building blocks from scratch. With the UML, you use class diagrams to visualize the static aspects of these building blocks and their relationships and to specify their details for construction, as you can see in Figure



TERMS AND CONCEPTS

A class diagram is a diagram that shows a set of classes, interfaces, and collaborations and their relationships. Graphically, a class diagram is a collection of vertices and arcs.

Common Properties:

A class diagram is just a special kind of diagram and shares the same common properties as do all other diagrams name and graphical content that are a projection into a model. What distinguishes a class diagram from other kinds of diagrams is its particular content.

Contents:

Class diagrams commonly contain the following things:

• Classes

• Interfaces

• Collaborations

• Dependency, generalization, and association relationships. Like all other diagrams, class diagrams may contain notes and constraints.

Class diagrams may also contain packages or subsystems, both of which are used to group elements of your model into larger chunks. Sometimes you'll want to place instances in your class diagrams as well, especially when you want to visualize the (possibly dynamic) type of an instance.

Common Uses :

You use class diagrams to model the static design view of a system. This view primarily supports the functional requirements of a system the services the system should provide to its end users.

When you model the static design view of a system, you'll typically use class diagrams in one of three ways.

1. To model the vocabulary of a system

Modeling the vocabulary of a system involves making a decision about which abstractions are a part of the system under consideration and which fall outside its boundaries. You use class diagrams to specify these abstractions and their responsibilities.

2. To model simple collaborations

A collaboration is a society of classes, interfaces, and other elements that work together to provide some cooperative behavior that's bigger than the sum of all the elements. For example, when you re modeling the semantics of a transaction in a distributed system, you can't just stare at a single class to understand what's going on. Rather, these semantics are carried out by a set of classes that work together. You use class diagrams to visualize and specify this set of classes and their relationships.

3. To model a logical database schema

Think of a schema as the blueprint for the conceptual design of a database. In many domains, you'll want to store persistent information in a relational database or in an object-oriented database. You can model schemas for these databases using class diagrams.

COMMON MODELING TECHNIQUES

1. Modeling Simple Collaborations

To model a collaboration,

• Identify the mechanism you'd like to model. A mechanism represents some function or behavior of the part of the system you are modeling that results from the interaction of a society of classes, interfaces, and other things.

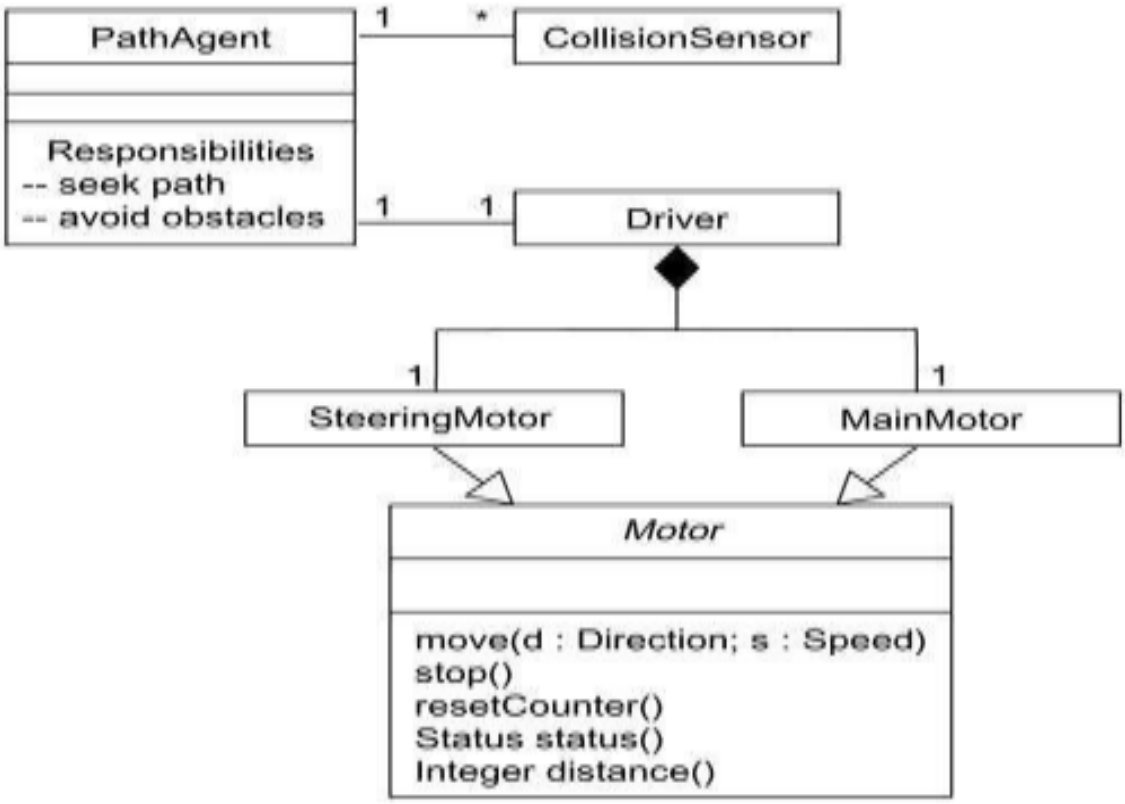
• For each mechanism, identify the classes, interfaces, and other collaborations that participate in this collaboration. Identify the relationships among these things as well.

• Use scenarios to walk through these things. Along the way, you'll discover parts of your model that were missing and parts that were just plain semantically wrong.

• Be sure to populate these elements with their contents. For classes, start with getting a good balance of responsibilities. Then, over time, turn these in to concrete attributes and operations.

For example, Figure shows a set of classes drawn from the implementation of an autonomous robot. The figure focuses on the classes involved in the mechanism for moving the robot along a path. You'll find one abstract class (Motor) with two concrete children, SteeringMotor and MainMotor. Both of these classes inherit the five operations of their parent, Motor. The two classes are, in turn, shown as parts of another class, Driver. The class PathAgent has a one-to-one association to Driver and a one-to-many association to CollisionSensor. No attributes or operations are shown for PathAgent, although its responsibilities are given

Figure Modeling Simple Collaborations



2. Modeling a Logical Database Schema

To model a schema,

• Identify those classes in your model whose state must transcend the lifetime of their applications.

• Create a class diagram that contains these classes. You can define your own set of stereotypes and tagged values to address database-specific details.

• Expand the structural details of these classes. In general, this means specifying the details of their attributes and focusing on the associations and their multiplicities that relate these classes.

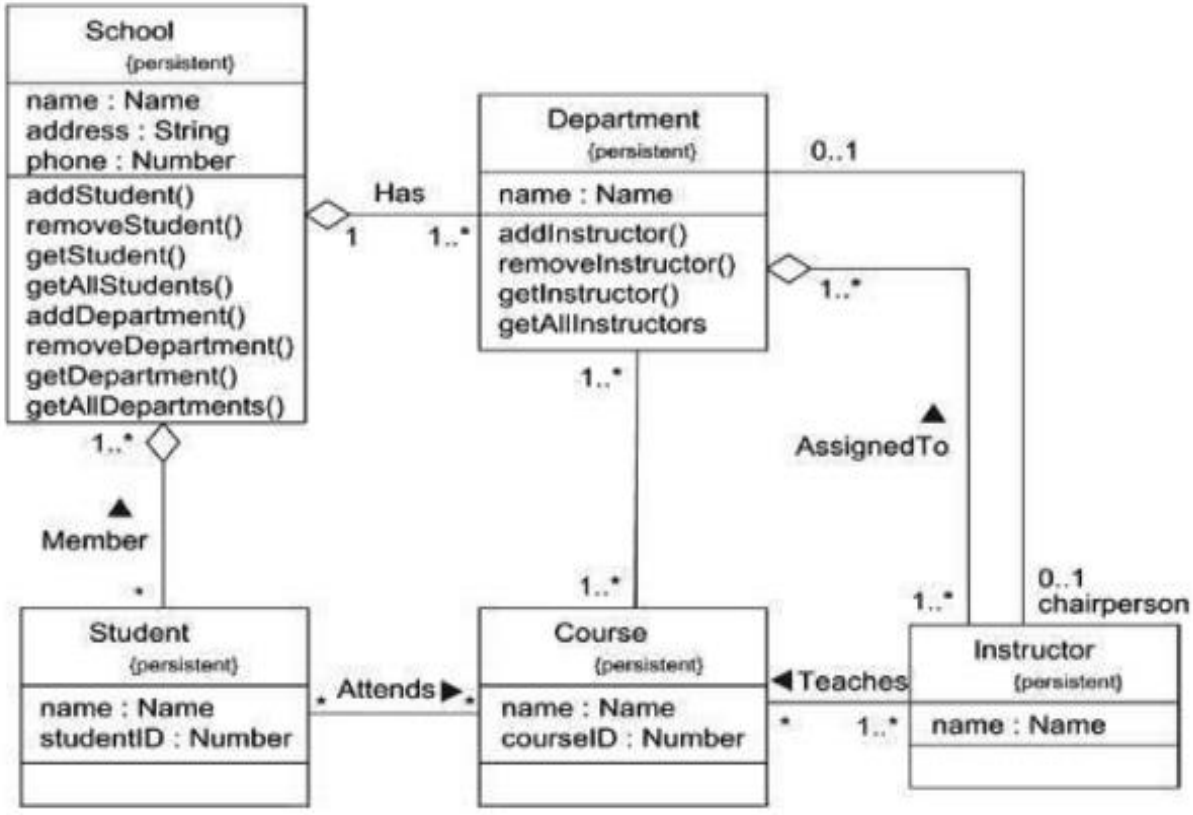
• Watch for common patterns that complicate physical database design, such as cyclic associations and one-to-one associations. Where necessary, create intermediate abstractions to simplify your logical structure.

• Consider also the behavior of these classes by expanding operations that are important for data access and data integrity. In general, to provide a better separation of concerns, business rules concerned with the manipulation of sets of these objects should be encapsulated in a layer above these persistent classes.

• Where possible, use tools to help you transform your logical design into a physical design.

The Figure shows a set of classes drawn from an information system for a school. We find the classes named Student, Course, and Instructor. There's an association between Student and Course, specifying that students attend courses. Furthermore, every student may attend any number of courses, and every course may have any number of students.

Figure Modeling a Schema

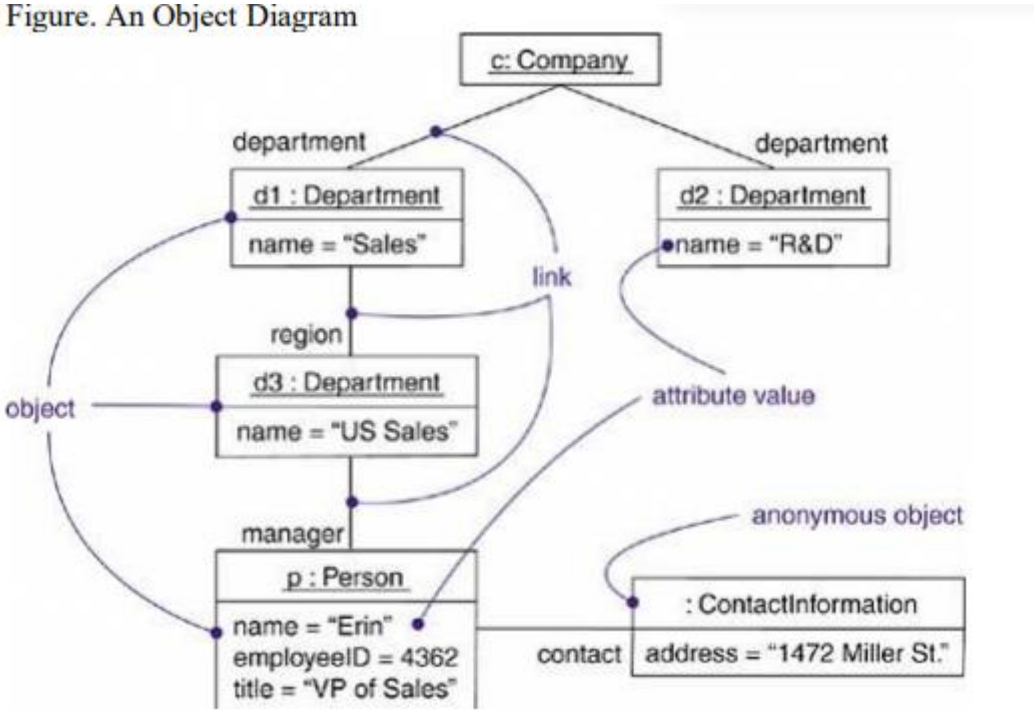


Object Diagrams:

Object diagrams model the instances of things contained in class diagrams. An object diagram shows a set of objects and their relationships at a point in time.

An object diagram covers a set of instances of the things found in a class diagram. An object diagram, therefore, expresses the static part of an interaction, consisting of the objects that collaborate but without any of the messages passed among them. In both cases, an object diagram freezes a moment in time, as in Figure .

Figure. An Object Diagram



TERMS AND CONCEPTS

An object diagram is a diagram that shows a set of objects and their relationships at a point in time. Graphically, an object diagram is a collection of vertices and arcs.

Common Properties

An object diagram is a special kind of diagram and shares the same common properties as all other diagrams that is, a name and graphical contents that are a projection into a model. What distinguishes an object diagram from all other kinds of diagrams is its particular content.

Contents

Object diagrams commonly contain

•Objects

•Links

Like all other diagrams, object diagrams may contain notes and constraints.

Sometimes you'll want to place classes in your object diagrams as well, especially when you want to visualize the classes behind each instance.

Common Uses

You use object diagrams to model the static design view or static process view of a system just as you do with class diagrams, but from the perspective of real or prototypical instances. This view primarily supports the functional requirements of a systemthat is, the services the system should provide to its end users. Object diagrams let you model static data

structures.

When you model the static design view or static process view of a system, you typically use object diagrams in one way:

To model object structures

Modeling object structures involves taking a snapshot of the objects in a system at a given moment in time. An object diagram represents one static frame in the dynamic storyboard represented by an interaction diagram. You use object diagrams to visualize, specify, construct, and document the existence of certain instances in your system, together with their relationships to one another.

COMMON MODELING TECHNIQUES

Modeling Object Structures

To model an object structure,

•Identify the mechanism you'd like to model. A mechanism represents some function or behavior of the part of the system you are modeling that results from the interaction of a society of classes, interfaces, and other things.

• Create a collaboration to describe a mechanism.

• For each mechanism, identify the classes, interfaces, and other elements that participate in this collaboration; identify the relationships among these things as well.

•Consider one scenario that walks through this mechanism. Freeze that scenario at a moment in time, and render each object that participates in the mechanism.

•Expose the state and attribute values of each such object, as necessary, to understand the scenario.

•Similarly, expose the links among these objects, representing instances of associations among them.

Figure Modeling Object Structures