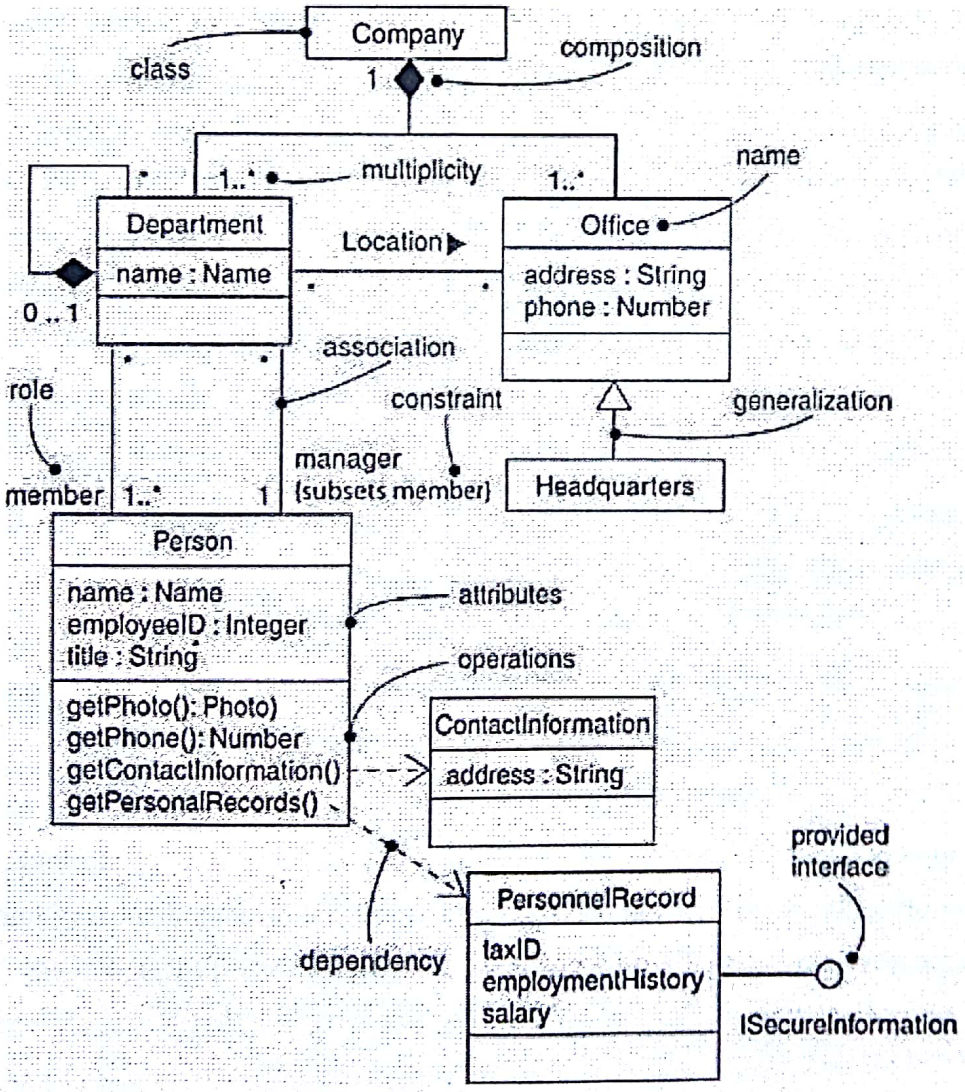
UNIT II1
CLASS & OBJECT DIAGRAMS

CLASS DIAGRAMS

Class diagrams are the most common diagram found in modeling object-oriented systems. A class diagram shows a set of classes, interfaces, and collaborations and theirrelationships.

You use class diagrams to model the static design view of a system. For the most part, this involves modeling the vocabulary of the system, modeling collaborations, or modelingschemas. Class diagrams are also the foundation for a couple of related diagrams: component diagrams and deployment diagrams.

Class diagrams are important not only for visualizing, specifying, and documenting structural models, but also for constructing executable systems through forward and reverseengineering.



GETTING STARTED

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.

www.Jntufastupdates.com

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. Whatdistinguishes 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

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 itsend 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 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 stareat a single class to understand what's going on. Rather, these semantics are carried out by a set ofclasses that work together. You use class diagrams to visualize and specify this set of classes andtheir relationships.

2. 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-orienteddatabase. You can model schemas for these databases using class diagrams.COMMON MODELING TECHNIQUES

Modeling Simple Collaborations

To model a collaboration,

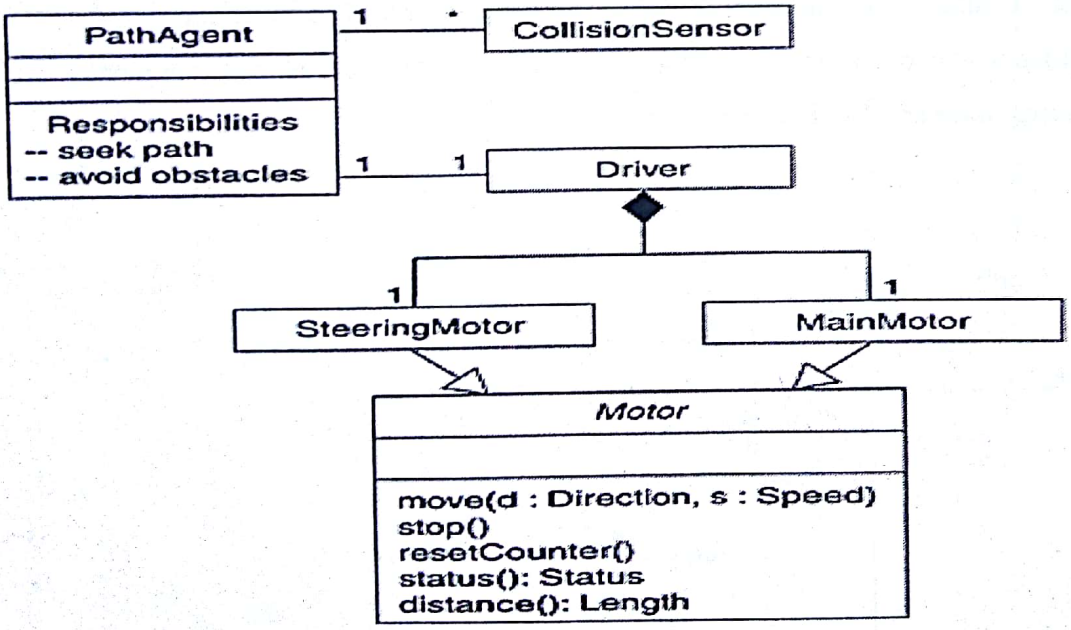
- Identify the mechanism you'd like to model. A mechanism represents some function or behavior of the part of the system you are modeling that results from the interaction of a

www.Jntufastupdates.com

society of classes, interfaces, and other things.

- For each mechanism, identify the classes, interfaces, and other collaborations thatparticipate 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 yourmodel 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 agood balance of responsibilities. Then, over time, turn these into concrete attributes andoperations.

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 apath. 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 twoclasses are, in turn, shown as parts of another class, Driver. The class PathAgent has a one-to-oneassociation to Driver and a one-to-many association to CollisionSensor. No attributes oroperations are shown for PathAgent, although its responsibilities are given.



MODELING A LOGICAL DATABASE SCHEMA

To model a schema,

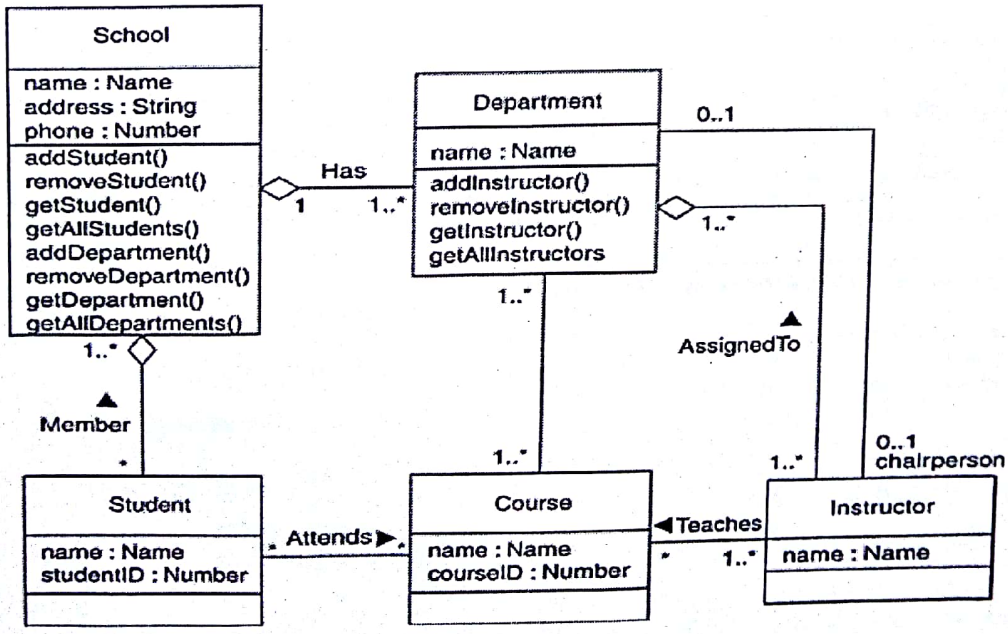
- Identify those classes in your model whose state must transcend the lifetime of their applications.
- Create a class diagram that contains these classes. You can define your own set ofstereotypes

www.Jntufastupdates.com

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 theseclasses.
- Watch for common patterns that complicate physical database design, such as cyclic associations and one-to-one associations. Where necessary, create intermediateabstractions to simplify your logical structure.
- Consider also the behavior of these classes by expanding operations that are important fordata 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 beencapsulated in a layer above these persistent classes.
- Where possible, use tools to help you transform your logical design into a physicaldesign.

The Figure shows a set of classes drawn from an information system for a school. This figure expands upon an earlier class diagram, and you'll see the details of these classes revealed to alevel sufficient to construct a physical database. Starting at the bottom-left of this diagram, you will find the classes named Student, Course, and Instructor. There's an association between Student and Course, specifying that students attend courses. Furthermore, every student mayattend any number of courses, and every course may have any number of students.



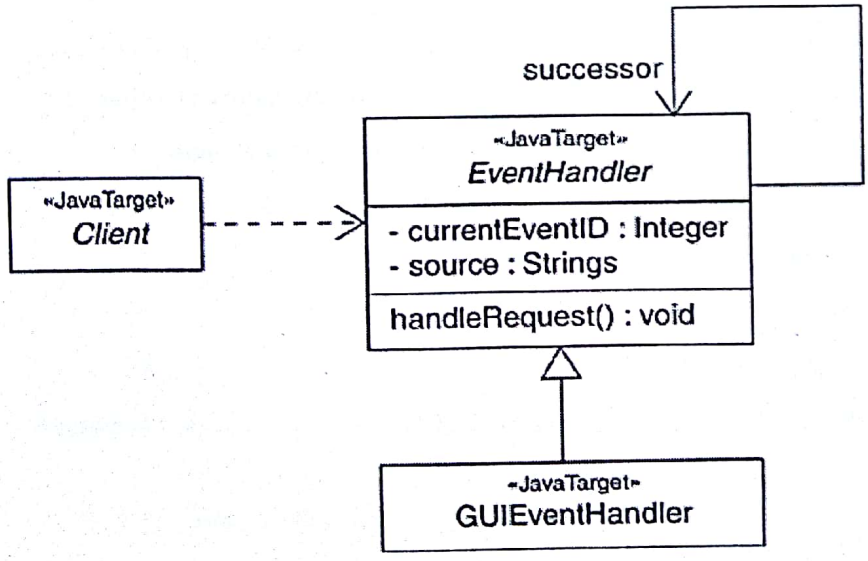
To forward engineer a class diagram,

- Identify the rules for mapping to your implementation language or languages of choice.This is something you'll want to do for your project or your organization as a whole.

www.Jntufastupdates.com

- Depending on the semantics of the languages you choose, you may want to constrainyour use of certain UML features. For example, the UML permits you to model multiple inheritance, but Smalltalk permits only single inheritance. You can choose to prohibit developers from modeling with multiple inheritance (which makes your models language-dependent), or you can develop idioms that transform these richer features into the implementation language (which makes the mapping more complex).
- Use tagged values to guide implementation choices in your target language. You can dothis at the level of individual classes if you need precise control. You can also do so at ahigher level, such as with collaborations or packages.
- Use tools to generate code.

Figure illustrates a simple class diagram specifying an instantiation of the chain of responsibility pattern. This particular instantiation involves three classes: Client, EventHandler,and GUIEventHandler. The classes Client and EventHandler are abstract, whereas GUIEventHandler is concrete. EventHandler has the usual operation expected of this pattern(handleRequest), although two private attributes have been added for this instantiation.



All of these classes specify a mapping to Java, as noted in their stereotype. Forward engineeringthe classes in this diagram to Java is straightforward, using a tool. Forward engineering the class EventHandler yields the following code.

```
public abstract class EventHandler {
```


www.Jntufastupdates.com

```
EventHandler successor; private
Integer currentEventID;private
String source;
EventHandler() {}
public void handleRequest() {}
}
```

REVERSE ENGINEERING:

Reverse engineering is the process of transforming code into a model through a mapping from a specific implementation language.

To reverse engineer a class diagram,

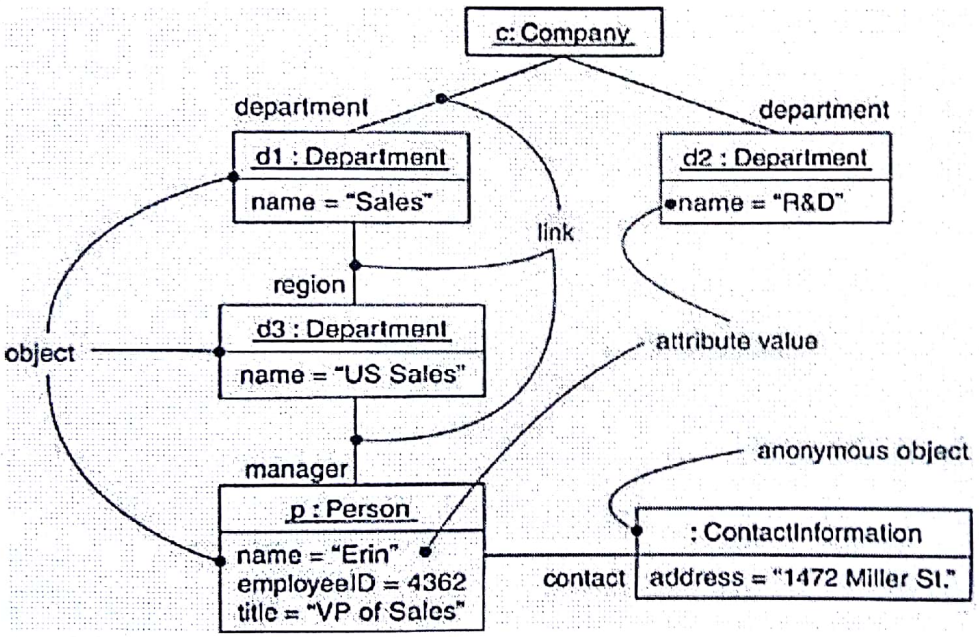
- Identify the rules for mapping from your implementation language or languages of choice. This is something you'll want to do for your project or your organization as awhole.
- Using a tool, point to the code you'd like to reverse engineer. Use your tool to generate anew model or modify an existing one that was previously forward engineered. It isunreasonable to expect to reverse engineer a single concise model from a large body ofcode. You need to select portion of the code and build the model from the bottom.
- Using your tool, create a class diagram by querying the model. For example, you mightstart with one or more classes, then expand the diagram by following specific relationships or other neighboring classes. Expose or hide details of the contents of thisclass diagram as necessary to communicate your intent.
- Manually add design information to the model to express the intent of the design that is missing or hidden in the code.

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.

You use object diagrams to model the static design view or static process view of a system. This involves modeling a snapshot of the system at a moment in time and rendering a set of objects, their state, and their relationships.

Object diagrams are not only important for visualizing, specifying, and documenting structural models, but also for constructing the static aspects of systems through forward and reverse engineering.



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.

COMMON MODELING TECHNIQUES

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 asociety of classes, interfaces, and other things.
- Create a collaboration to describe a mechanism.
- For each mechanism, identify the classes, interfaces, and other elements that participatein this collaboration; identify the relationships among these things as well.
- Consider one scenario that walks through this mechanism. Freeze that scenario at amoment 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.

For example, Figure shows a set of objects drawn from the implementation of an autonomous robot. This figure focuses on some of the objects involved in the mechanism used by the robot to calculate a model of the world in which it moves. There are many more objects involved in a running system, but this diagram focuses on only those abstractions that are directly involved in creating this world view.

