

# Object-Oriented Design and Modeling Using the UML

## Overview

This is the second of two chapters on object-oriented tools and techniques for system development. This chapter builds upon Chapter 10 and teaches students the important skill of object modeling during systems design. The students will learn about various unified modeling language (UML) diagrams and object-oriented design concepts.

## Chapter to Course Sequencing

If the adopter is taking a basically object-oriented approach, this chapter can be used in place of Chapters 13 and 14 in whole or in part. Adopters wanting to focus on traditional structured analysis could skip this chapter. But in almost all cases this chapter should be taught only if Chapter 10 has previously been covered.

## What's Different Here and Why?

This chapter has been extensively reworked from the sixth edition. The following changes have been made:

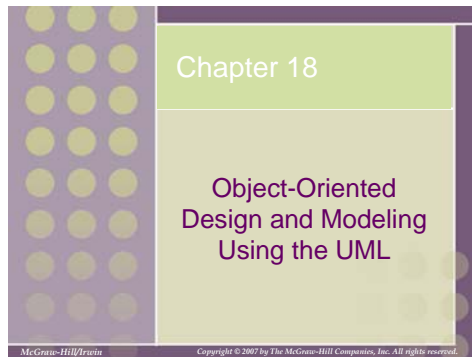
1. As with all chapters, we have streamlined the SoundStage episode into a quick narrative introduction to the concepts presented in the chapter.
2. The chapter has been revised for UML 2.0.
3. We have rearranged some of the topics in the chapter for better flow. Examples of this include object reusability and sequence diagrams.
4. The list of design object classes has been expanded to include persistence system classes as well as entity, interface, and control classes. We have added a programming IDE screen to reinforce the idea that in OO programming all code exists within a class.
5. We have refined the steps of the object design process to include modeling object class states.
6. We expanded the discussion of CRC cards as a method for identifying object class behaviors and responsibilities.

7. We expanded the discussion of sequence diagrams as a method for identifying object class behaviors and responsibilities. We added explanations and examples for frames and self-calls. We discussed how behaviors identified on a sequence diagram impact the class diagram. We added guidelines for constructing sequence diagrams.
8. We expanded the discussion of object reusability, emphasizing the twin goals of low coupling and high cohesion.
9. We expanded the discussion of design patterns, including an overview of the Gang-Of-Four patterns and detailed explanations of two of the GOF patterns.
10. We added an explanation and example of the communication diagram.

### Lesson Planning Notes for Slides

The following instructor notes, keyed to slide images from the PowerPoint repository, are intended to help instructors integrate the slides into their individual lesson plans for this chapter.

Slide 1



slide appearance after initial mouse click  
in slide show mode

This repository of slides is intended to support the named chapter. The slide repository should be used as follows:

Copy the file to a unique name for your course and unit.

Edit the file by deleting those slides you don't want to cover, editing other slides as appropriate to your course, and adding slides as desired. Print the slides to produce transparency masters or print directly to film or present the slides using a computer image projector.

Each slide includes instructor notes. To view those notes in PowerPoint, click-left on the View Menu; then click left on Notes View sub-menu. You may need to scroll down to see the instructor notes.

The instructor notes are also available in hard-copy as the Instructor Guide to Accompany Systems Analysis and Design Methods, 6/ed.

Slide 2

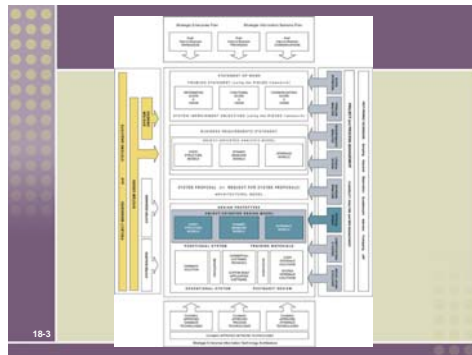
**Objectives**

- Understand entity, interface, control, persistence, and system classes.
- Understand the concepts of dependency and navigability.
- Define visibility and explain its three levels.
- Understand the concept object responsibility and how it is related to message sending between object types.
- Describe the activities involved in object-oriented design.
- Differentiate between a design use-case narrative and an analysis use-case narrative.
- Describe CRC card modeling.
- Model class interactions with sequence diagrams.
- Construct a class diagram that reflects design specifics.
- Model object states with state machine diagrams.
- Understand the role of coupling and cohesion in object reuse.
- Describe the use of design patterns and two common design patterns.
- Differentiate between design patterns, object frameworks, and components.
- Understand the use of communication diagrams, component diagrams, and deployment diagrams.

18-2

No additional notes.

Slide 3



**Teaching Notes**

This slide shows the how this chapter's content fits with the building blocks framework used throughout the textbook. The emphasis of this chapter is with the physical design phase, spanning the communication focus, knowledge focus, and process focus. It involves system designers, systems analysts, and users.

Slide 4

**Object-Oriented Design**

**Object-oriented design (OOD)** – an approach used to specify the software solution in terms of collaborating objects, their attributes, and their methods.

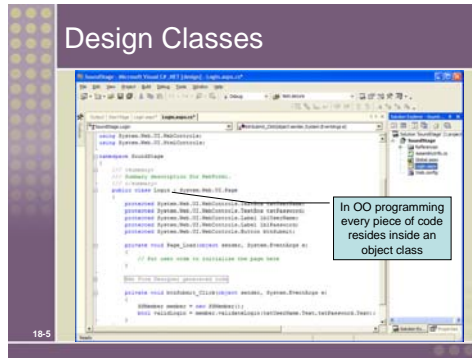
- Continuation of object-oriented analysis

18-4

**Teaching Notes**

The approach of using object-oriented techniques for designing a system is referred to as **object-oriented design**. Recall that object-oriented development approaches are best suited to projects that will implement systems using emerging object technologies to construct, manage, and assemble those objects into useful computer applications. Object-oriented design is the continuation of object-oriented analysis (Chapter 10), continuing to center the development focus around object modeling techniques. During object-oriented design, entity objects are refined while other types of objects are identified that will be introduced as the result of physical implementation decisions for the new system.

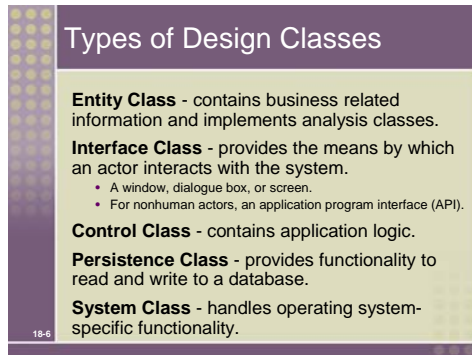
Slide 5



**Teaching Notes**

If your students have programmed in .NET or in Java or in any other OO environment they should have noticed that the code they write always goes into a class. This connection can help students understand the next slide.

Slide 6



**Conversion Notes**

The sixth edition identified only entity classes, interface classes, and control classes.

**Teaching Notes**

Why all these kinds of classes? Structuring the system this way makes the maintenance and enhancement of those classes simpler and easier.

Entity classes are what we worked with in Chapter 10 during object-oriented analysis.

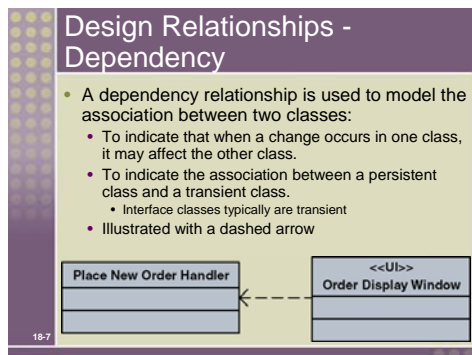
The .NET code on the previous slide is an example of an interface class.

Control classes coordinate message among the other kinds of classes to implement a use case.

The code to handle database reading/writing could be built into the entity class. But having that functionality in a separate class keeps the entity class implementation-neutral, which makes it more reusable. We will talk more about the value of reusability later in the chapter.

System classes isolate the other options from the operating system to keep them implementation-neutral and more reusable.

Slide 7



**Teaching Notes**

The Order Display Window is an interface class <<UI>>. It is dependent on the Order Processor class to respond to events initiated from the interface.

Slide 8

### Design Relationships - Navigability

- Classes with associations can navigate (send messages) to each other.
- By default the associations are bidirectional.
- Sometimes you want to limit the message sending to only one direction.
- Illustrated with an arrow pointing in the direction a message can be sent.

18-8

**Teaching Notes**

Given a User, you can find that user's current password for authentication. But given a password, you cannot find the corresponding user.

Slide 9

### Attribute and Method Visibility

**Visibility** – the level of access an external object has to an attribute or method.

- Public** attributes/methods can be accessed/invoked by any other method in any other object or class. Denoted by the + symbol
- Protected** attributes/methods can be accessed/invoked by any method in the same class or in subclasses of that class. Denoted by the # symbol
- Private** attributes/methods can be accessed/invoked by any method in the same class. Denoted by the – symbol

**Method** – the software logic that is executed in response to a message.

18-9

No additional notes.

Slide 10

### Object Responsibilities

**Object responsibility** – the obligation that an object has to provide a service when requested and thus collaborate with other objects to satisfy the request if required.

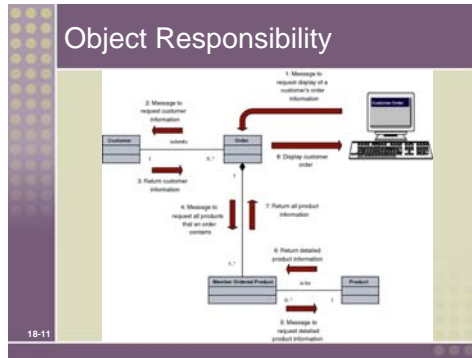
- An object responsibility is implemented by the creation of methods that may have to collaborate with other objects and methods.

18-10

**Teaching Notes**

Object responsibility is closely related to the concept of objects being able to send and/or respond to messages. For example, an ORDER object may have the responsibility to display a customer's order, but it may need to collaborate with the CUSTOMER object to get the customer data, the PRODUCT object to get the product data, and the ORDER LINE object to get specific order data about each product being ordered. Thus, CUSTOMER, PRODUCT, and ORDER LINE have an obligation to provide the requested service (provide requested data) to the ORDER object.

Slide 11



No additional notes.

Slide 12

### The Process of Object-Oriented Design

- Refining the use case model to reflect the implementation environment.
- Modeling class interactions, behaviors, and states that support the use case scenario.
- Updating the class diagram to reflect the implementation environment.

**Teaching Notes**

In performing object-oriented analysis (OOA) we identified objects and use cases based on ideal conditions and independent of any hardware or software solution. During object-oriented design (OOD) we want to refine those objects and use cases to reflect the actual environment of our proposed solution.

Slide 13

### Refining The Use Case Model

- Step 1: Transform the "Analysis" Use Cases to "Design" Use Cases
  - Implementation details
  - Controls
  - Window/web page names
  - Navigation instructions
- Step 2: Update the Use Case Model Diagram and Other Documentation to Reflect any New Use Cases

**Teaching Notes**

In this iteration of use case modeling, the use cases will be refined to include details of how the actor (or user) will actually interface with the system and how the system will respond to that stimulus to process the business event. The manner in which the user accesses the system; via a menu, window, button, bar code reader, printer, etc. should be explicitly described in detail. The contents of windows, reports, and queries should also be specified within the use case. While refining use cases is often time consuming and tedious, it is essential that they are completed.

Slide 14

Design Use Case	
<b>Member Services System</b>	
Author: K. Dittman	Date: 11/21/02
Version: 1.02	Use Case Type: <input checked="" type="checkbox"/> New
Use Case Name: Place New Order	System Requirements: <input checked="" type="checkbox"/> (1)
Use Case ID: MGS-S-020 (2)	System Analysis: <input checked="" type="checkbox"/> (1)
Priority: High	System Design: <input checked="" type="checkbox"/> (1)
Source: Requirement — MGS-RT-02	
Requirements Use Case — MGS-BU-020 (2)	
Primary Business Action: (CU) Member (Role — Active Member, Member)	
Primary System Action: (CU) Member (Role — Active Member, Member)	
Other Participating Actors: <ul style="list-style-type: none"> <li>Warehouse (Role — Distribution Center) (external resource)</li> <li>Accounts Receivable (external server)</li> </ul>	
Other Interested Stakeholders: <ul style="list-style-type: none"> <li>Marketing — interested in sales activity in order to plan new promotions.</li> <li>Procurement — interested in sales activity in order to replenish inventory.</li> <li>Management — interested in sales activity in order to evaluate company performance and customer (member) satisfaction.</li> </ul>	
Description: This use case describes the event of a user member submitting a new order for SoundStage products via the World Wide Web. The member selects the items he or she wishes to purchase. Once the member has completed shipping the member's demographic information as well as account standing will be validated. Once the products are verified as being in stock, a pending order is sent to the distribution center for it to process the shipment. For any product not in stock, a back order is created. On completion, the member will be sent an order confirmation.	
Precondition: The individual must be a registered user of the system.	
Postcondition: The member must have logged in to the system, and the member home page is being displayed.	
Trigger: The user case is initiated when the member clicks to the option to place a new order.	

**Teaching Notes**

Notice that the course of events now describes the windows which will be displayed to the user, and the contents (i.e., field names) of the windows. Descriptions of, error messages, special action buttons, possible cursor movements, and other window characteristics should be included in each design use case step.

Slide 15

Design Use Case (continued)					
Typical Course of Events:	<table border="1"> <thead> <tr> <th>Actor Action</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td> <b>Step 1:</b> The member clicks on the place new order icon (or link).  <b>Step 2:</b> The member scrolls through the available items by using the scroll bar buttons, the Page Up and Page Down keys, or the navigational controls specified in step 5. The member selects the ones he or she wishes to purchase by clicking the checkboxes and entering the quantity to be ordered.  <b>Step 3:</b> The member verifies demographic information (shipping and billing addresses). If no changes are necessary, the member clicks the [Continue] button.  <b>Step 4:</b> The member verifies the order. If no changes are necessary, the member clicks the [Continue] button.  <b>Step 5:</b> The member responds by clicking the appropriate check box for the desired payment option.  <b>Step 11:</b> The member verifies the order. If no changes are necessary, the member clicks the [Continue] button.                 </td> <td> <b>Step 6:</b> The system responds by displaying window #27—Catalog Display: a list of SoundStage products.*                      * If the product list is greater than 50, which is the maximum number to be displayed on one page, the system calculates the number of pages required to display the products. The system then provides the member with the necessary navigational buttons, such as [Prev], [Next], [First], [Last], and [1] through [14], and so on.  <b>Step 7:</b> Once the member has completed making selections, the system verifies the member's demographic information (shipping and billing addresses) and displays it in window #23—Member Profile Display. The system also prompts the member to make any required changes.  <b>Step 8:</b> For each product ordered, the system verifies the product availability and determines an expected ship date, determines the price to be charged to the member, and determines the total of the total order. If an item is not immediately available, it indicates that the product is back ordered or that it has not been released for shipping (the proceeds) if an item is no longer available, that is indicated also. The system then displays a summary of the order in window #25—Order Summary Display. The system also prompts the member to make any required changes.                 </td> </tr> </tbody> </table>	Actor Action	System Response	<b>Step 1:</b> The member clicks on the place new order icon (or link). <b>Step 2:</b> The member scrolls through the available items by using the scroll bar buttons, the Page Up and Page Down keys, or the navigational controls specified in step 5. The member selects the ones he or she wishes to purchase by clicking the checkboxes and entering the quantity to be ordered. <b>Step 3:</b> The member verifies demographic information (shipping and billing addresses). If no changes are necessary, the member clicks the [Continue] button. <b>Step 4:</b> The member verifies the order. If no changes are necessary, the member clicks the [Continue] button. <b>Step 5:</b> The member responds by clicking the appropriate check box for the desired payment option. <b>Step 11:</b> The member verifies the order. If no changes are necessary, the member clicks the [Continue] button.	<b>Step 6:</b> The system responds by displaying window #27—Catalog Display: a list of SoundStage products.* * If the product list is greater than 50, which is the maximum number to be displayed on one page, the system calculates the number of pages required to display the products. The system then provides the member with the necessary navigational buttons, such as [Prev], [Next], [First], [Last], and [1] through [14], and so on. <b>Step 7:</b> Once the member has completed making selections, the system verifies the member's demographic information (shipping and billing addresses) and displays it in window #23—Member Profile Display. The system also prompts the member to make any required changes. <b>Step 8:</b> For each product ordered, the system verifies the product availability and determines an expected ship date, determines the price to be charged to the member, and determines the total of the total order. If an item is not immediately available, it indicates that the product is back ordered or that it has not been released for shipping (the proceeds) if an item is no longer available, that is indicated also. The system then displays a summary of the order in window #25—Order Summary Display. The system also prompts the member to make any required changes.
Actor Action	System Response				
<b>Step 1:</b> The member clicks on the place new order icon (or link). <b>Step 2:</b> The member scrolls through the available items by using the scroll bar buttons, the Page Up and Page Down keys, or the navigational controls specified in step 5. The member selects the ones he or she wishes to purchase by clicking the checkboxes and entering the quantity to be ordered. <b>Step 3:</b> The member verifies demographic information (shipping and billing addresses). If no changes are necessary, the member clicks the [Continue] button. <b>Step 4:</b> The member verifies the order. If no changes are necessary, the member clicks the [Continue] button. <b>Step 5:</b> The member responds by clicking the appropriate check box for the desired payment option. <b>Step 11:</b> The member verifies the order. If no changes are necessary, the member clicks the [Continue] button.	<b>Step 6:</b> The system responds by displaying window #27—Catalog Display: a list of SoundStage products.* * If the product list is greater than 50, which is the maximum number to be displayed on one page, the system calculates the number of pages required to display the products. The system then provides the member with the necessary navigational buttons, such as [Prev], [Next], [First], [Last], and [1] through [14], and so on. <b>Step 7:</b> Once the member has completed making selections, the system verifies the member's demographic information (shipping and billing addresses) and displays it in window #23—Member Profile Display. The system also prompts the member to make any required changes. <b>Step 8:</b> For each product ordered, the system verifies the product availability and determines an expected ship date, determines the price to be charged to the member, and determines the total of the total order. If an item is not immediately available, it indicates that the product is back ordered or that it has not been released for shipping (the proceeds) if an item is no longer available, that is indicated also. The system then displays a summary of the order in window #25—Order Summary Display. The system also prompts the member to make any required changes.				

**Teaching Notes**

The design use case step includes references to extension and abstract use cases. Recall that extension use cases extend the functionality of the original use case by extracting complex or hard to understand logic into its own use case. Abstract use cases are those that contain steps that are used by more than one design use case.

Slide 16

Design Use Case (continued)		
	<table border="1"> <tbody> <tr> <td> <b>Step 9:</b> The system checks the status of the member's account. If satisfactory, the system prompts the member to select the desired payment option (to be billed later or pay immediately with a credit card).  <b>Step 10:</b> The system then displays a final summary of the order in window #25—Order Summary Display. The system also prompts the member to make any required changes.  <b>Step 13:</b> The system records the order information (including back orders, if necessary).  <b>Step 13:</b> Invoke abstract use case MGS-AUC020 (2): Determine Appropriate Distribution Center and Release Order to Ship.  <b>Step 14:</b> Once the order is processed, the system generates an order confirmation and displays it in window #24—Order Confirmation Display. The system also sends the confirmation by e-mail. Invoke abstract use case MGS-AUC040 (2): Send Electronic Member Confirmation.                 </td> </tr> </tbody> </table>	<b>Step 9:</b> The system checks the status of the member's account. If satisfactory, the system prompts the member to select the desired payment option (to be billed later or pay immediately with a credit card). <b>Step 10:</b> The system then displays a final summary of the order in window #25—Order Summary Display. The system also prompts the member to make any required changes. <b>Step 13:</b> The system records the order information (including back orders, if necessary). <b>Step 13:</b> Invoke abstract use case MGS-AUC020 (2): Determine Appropriate Distribution Center and Release Order to Ship. <b>Step 14:</b> Once the order is processed, the system generates an order confirmation and displays it in window #24—Order Confirmation Display. The system also sends the confirmation by e-mail. Invoke abstract use case MGS-AUC040 (2): Send Electronic Member Confirmation.
<b>Step 9:</b> The system checks the status of the member's account. If satisfactory, the system prompts the member to select the desired payment option (to be billed later or pay immediately with a credit card). <b>Step 10:</b> The system then displays a final summary of the order in window #25—Order Summary Display. The system also prompts the member to make any required changes. <b>Step 13:</b> The system records the order information (including back orders, if necessary). <b>Step 13:</b> Invoke abstract use case MGS-AUC020 (2): Determine Appropriate Distribution Center and Release Order to Ship. <b>Step 14:</b> Once the order is processed, the system generates an order confirmation and displays it in window #24—Order Confirmation Display. The system also sends the confirmation by e-mail. Invoke abstract use case MGS-AUC040 (2): Send Electronic Member Confirmation.		

No additional notes

Slide 17

### Design Use Case (concluded)

**Alternate Cases:**

**Alt-Step 3a:** If the member clicks on the item name, the system displays a pop-up window, W1—Product Detail Display, which contains all the product details, including a graphic of the cover. The member clicks the [Close] button to close the pop-up window.

**Alt-Step 3b:** If member wants to perform keyword search, invoke abstract use case MS3—AUC303/07 Search Product Catalog by Keyword.

**Alt-Step 3c:** If member wants to change demographic information, invoke abstract use case MS3—AUC307/07 Change Member Profile.

**Alt-Step 3d:** If the order requires changes the member can delete any item no longer wanted by increasing the check box to item and/or changing the order quantity. Once the member has completed the order changes, he or she clicks the [Update Order] button. The system reprocesses the order. **Go to step 4.**

**Alt-Step 3e:** If the member clicks the [Go Home Shopping] button, **go to Step 3.** If the member clicks the [Update Member Profile] button, invoke abstract use case MS3—AUC307/07 Change Member Profile and then **go to step 4.**

**Alt-Step 3f:** If the member's account is not in good standing, display to the member using window MS3—Member Account Status Change the account status, the reason the order is being held, and what actions are necessary to resolve the problem. In addition an e-mail is sent to the member with the same information. Invoke abstract use case MS3—AUC304/07 Send Electronic Member Correspondence. The system prompts the member to hold the order for later processing or cancel the order. If the member wishes to hold the order by clicking the [Save Order] button, the system records the order information, places it on hold status, and then displays the SoundStage main page, window MS3—Member Home Page. If the member chooses to cancel the order by clicking the [Cancel Order] button, the system erases the included information, and then displays the SoundStage main page, window MS3—Member Home Page. Terminate the use case.

**Alt-Step 3g:** If the member selects the option to pay by credit card, invoke abstract use case MS3—AUC370/07 Pay by Credit Card.

**Alt-Step 3h:** If the member cancels the credit card, the system prompts the member to hold the order for later processing or cancel the order. If the member wishes to hold the order by clicking the [Save Order] button, the system records the order information, places it on hold status, and then displays the SoundStage main page, window MS3—Member Home Page. If the member chooses to cancel the order by clicking the [Cancel Order] button, the system erases the included information, and then displays the SoundStage main page, window MS3—Member Home Page. Terminate the use case.

18-17

No additional notes

Slide 18

### Modeling Class Interactions, Behaviors, and States

- Step 1: Identify and Classify Use-Case Design Classes
- Step 2: Identify Class Attributes
- Step 3: Identify Class Behaviors and Responsibilities
- Step 4: Model Object States
- Step 5: Model Detailed Object Interactions

18-18

**Teaching Notes**

In this activity we want to identify and categorize the design objects required by the functionality that was specified in each use case, and identify the object interactions, their responsibilities, and their behaviors.

Slide 19

### Step 1: Identify and Classify Use-Case Design Classes

Interface, Control, and Entity Classes of Place New Order Use Case		
Interface Classes	Controller Classes	Entity Classes
W00-Member Home Page W02-Member Profile Display W03-Display Order Summary W04-Display Order Confirmation W09-Member Account Status Display W11-Catalog Display W15-Product Detail Display	Place New Order Handler	Billing Address Shipping Address Email Address Active Member Member Order Member Ordered Product Product Title Audio Title Game Title Video Title Transaction

18-19

**Teaching Notes**

The interface object column contains a list of objects mentioned in the use case that the users directly interface with, such as screens, windows, card readers, and printers. The only way an actor or user can interface with a system is via an interface object. Therefore, there should be at least one interface object per actor or user.

The controller object column contains a list of objects that encapsulate application logic or business rules. As a reminder, a use case should reveal one controller object per unique user or actor.

The entity object column contains a list of objects that correspond to the business domain objects whose attributes were referenced in the use case.



Slide 20

### Step 2: Identify Class Attributes

- Many attributes already identified during object-oriented analysis.
- Revised use cases may mention additional attributes.
- Must update class diagram to include new attributes.

18-20

No additional notes

Slide 21

### Step 3: Identify Class Behaviors and Responsibilities

- Analyze use cases to identify required system behaviors
  - Search for verb phrases
  - Some will reflect manual actions, others automated
- Associate behaviors and responsibilities with classes
- Model classes that have complex behavior
- Examine class diagram for additional behaviors
- Verify classifications

18-21

No additional notes

Slide 22

### Condensed Behavior List

Condensed Behavior List for <i>Place New Order</i> Use Case	
Behaviors	Class Type
Process new member order	Control
Retrieve product catalog information	Entity
Display W11-Catalog Display window	Interface
Retrieve member demographic information	Entity
Display W02-Member Profile Display window	Interface
Validate quantity amount	Entity
Verify the product availability	Entity
Determine an expected ship date	Entity
Determine cost of the total order	Entity
Display W03-Order Summary Display window	Interface
Prompt user	Interface
Check Status of member account	Entity
...	

18-22

**Teaching Notes**

Behaviors can be detected by identifying verbs in a use case narrative.

This is a partial list. See the text for the full list of behaviors

Some of the behaviors assigned to entity classes would eventually be implemented with persistence classes. However, the entity class would maintain the responsibility of calling the persistence class. Thus the entity class would retain the behavior.

Slide 23

### Tools for Identifying Behaviors and Responsibilities

**Class Responsibility Collaboration (CRC) Card** - a card that lists all behaviors and responsibilities assigned to a class.

- Often built interactively in a group setting that walks through a use case

**Sequence diagram** - a UML diagram that models the logic of a use case by depicting the interaction of messages between objects in time sequence.

18-23

**Teaching Notes**

Different methodologies may emphasize one tool over the other or use other tools

Slide 24

### CRC Card Listing Behaviors and Collaborators of a Class

Object Name: Member Order	
Sub Object:	
Super Object: Transaction	
Behaviors and Responsibilities	Collaborators
Report order information Calculate subtotal cost Calculate total order cost Update order status Create Ordered Product Delete Ordered Product	Member Ordered Product

18-24

**Teaching Notes**

CRC stands for Class Responsibility Collaboration.

The CRC card contains all use-case behaviors and responsibilities that have been associated with an object.

A CRC card for the object type MEMBER ORDER is depicted in the figure above. Notice that the CRC card contains all use case behaviors and responsibilities that have been associated with the object type MEMBER ORDER.

Also in the figure above, the MEMBER ORDER object needs collaboration from the MEMBER ORDERED PRODUCT object to retrieve information about each of the products being ordered. Remember if an object needs another object's attribute to accomplish a behavior, the collaborating object needs to have a behavior or method to provide that attribute.

Slide 25

### Sequence Diagram

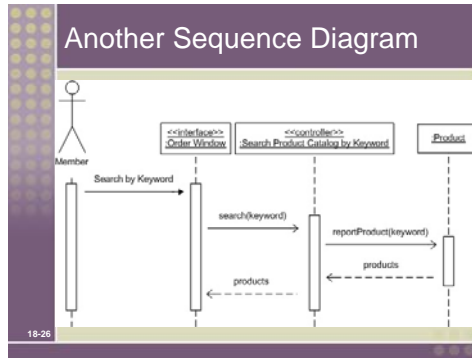
- Actor
- Interface class
- Controller class
- Entity classes
- Messages
- Activation bars
- Return messages
- Self-call
- Frame

18-25

**Teaching Notes**

In class we often take a simple use case narrative, scan for verbs to identify behaviors, and interactively build a sequence diagram.

Slide 26



No additional notes.

Slide 27

### Guidelines for Constructing Sequence Diagrams

- Identify the scope of the sequence diagram, whether entire use-case scenario or one step.
- Draw actor and interface class if scope includes that.
- List use-case steps down the left-hand side.
- Draw boxes for controller class and each entity class that must collaborate in the sequence (based on attributes or behaviors previously assigned).
- Add persistence and system classes if scope includes that.
- Draw messages and point each to class that will fulfill the responsibility.
- Add activation bars to indicate object instance lifetimes.
- Add return messages as needed for clarity.
- Add frames for loops, optional steps, alternate steps, etc.

No additional notes.

Slide 28

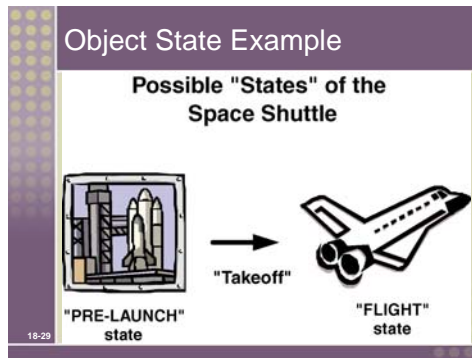
### Step 4: Model Object States

- **Object state** – a condition of the object at one point in its lifetime.
- **State transition event** – occurrence that triggers a change in an object’s state through updating of one or more of its attribute values.
- **State machine diagram** – a UML diagram that depicts:
  - the combination of states that an object can assume during its lifetime,
  - the events that trigger transitions between states,
  - the rules governing the objects in transition.

**Teaching Notes**

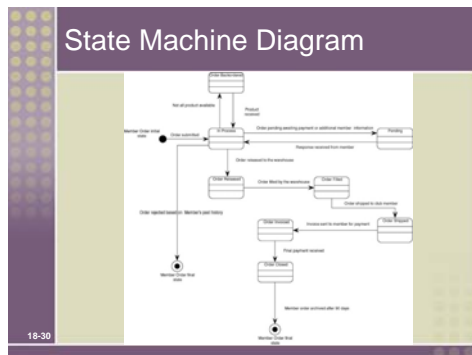
The concept of state is depicted on the next slide.

Slide 29



No additional notes.

Slide 30

**Teaching Notes**

State machine diagrams are not required for all objects, just those that have clearly identifiable states and complex behavior.

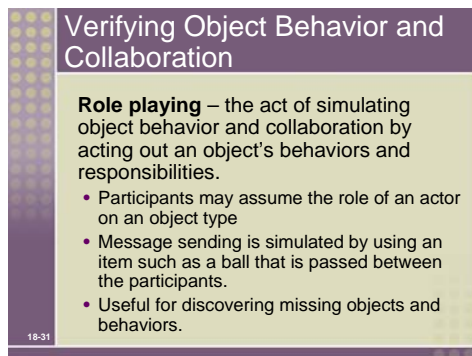
The solid circle represents the object's initial state.

The object transitions through a life cycle of different states represented by rounded-corner rectangles

Each arrow represents an event that triggers a change from one state to another

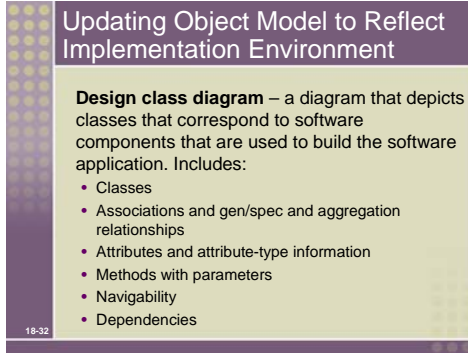
The solid circle inside the hollow circle represents the object's final state.

Slide 31



No additional notes.

Slide 32



**Updating Object Model to Reflect Implementation Environment**

**Design class diagram** – a diagram that depicts classes that correspond to software components that are used to build the software application. Includes:

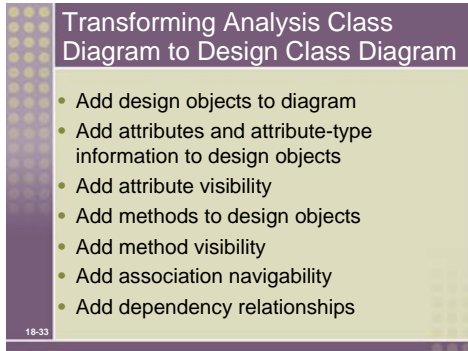
- Classes
- Associations and gen/spec and aggregation relationships
- Attributes and attribute-type information
- Methods with parameters
- Navigability
- Dependencies

18-32

**Teaching Notes**

The Design class diagram is the design equivalent of the class diagram prepared in OOA (Chapter 10).

Slide 33



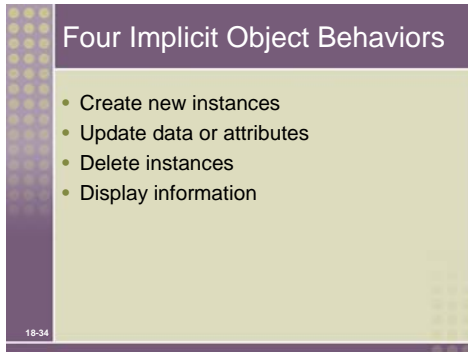
**Transforming Analysis Class Diagram to Design Class Diagram**

- Add design objects to diagram
- Add attributes and attribute-type information to design objects
- Add attribute visibility
- Add methods to design objects
- Add method visibility
- Add association navigability
- Add dependency relationships

18-33

No additional notes.

Slide 34



**Four Implicit Object Behaviors**

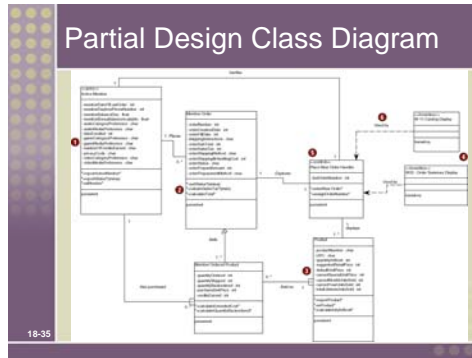
- Create new instances
- Update data or attributes
- Delete instances
- Display information

18-34

**Teaching Notes**

When identifying behaviors, don't forget these implied behaviors for every object.

Slide 35



**Teaching Notes**

The figure above is a partial view of our object class diagram which correlates to the objects used in the "Place New Member Order" use case. Notice we have given each behavior or method a name. Normally these names reflect the programming language used to develop the system.

Slide 36

### Object Reusability

**Coupling** - the degree to which one class is connected to or relies upon other classes.

**Cohesion** - the degree to which the attributes and behaviors of a single class are related to each other.

- The two overarching goals of object-oriented design are low coupling and high cohesion.
- Allows for object reuse.

No additional notes.

Slide 37

### Object Reusability The OO Success Story

Comparison of an OO Language and a 3GL Language			
Programming Language	Project Duration (calendar months)	Level of Effort (person months)	Software Size (lines of code)
PL/1	19	152	265,000
Smalltalk	3.5	10.4	22,000

**Teaching Notes**

PL/1 is a traditional 3GL language. Smalltalk is an OO language.

Slide 38

### Design Patterns

**Design pattern** - a common solution to a give problem in a given context, which supports reuse of proven approaches and techniques.

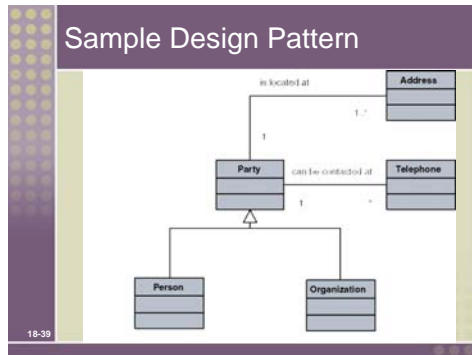
- Advantages
  - Allow us to design with the experiences of those who came before rather than having to "reinvent the wheel."
  - Provide designers a short-hand notation for discussing design issues.

18-38

**Teaching Notes**

You can think of design patterns as an FAQ, a "rule of thumb" or helpful advice for design issues.

Slide 39



**Teaching Notes**

A exercise to establish the value of design patterns is to brainstorm the kinds of information systems that could use this pattern. The list could include utility companies, IT consultants, lumber yards, and e-commerce sites.

Slide 40

### Gang-of-Four Patterns

Creational	Structural	Behavioral
Abstract factor	Adapter	Chain of responsibility
Builder	Bridge	Command
Factory method	Composite	Flyweight
Prototype	Decorator	Interpreter
Singleton	Façade	Iterator
	Proxy	Mediator
		Memento
		Observer
		State
		Strategy
		Template method
		Visitor

18-40

**Teaching Notes**

There are many other patterns beside the GOF patterns (as the Martin Fowler pattern shown on the previous slide).

Slide 41

### Strategy Pattern

Pattern:	Strategy
Category:	Behavioral
Problem:	How to design for varying and changing policy algorithms?
Solution:	Define each algorithm in a separate class with a common interface.

19-41

**Teaching Notes**

A subtype class can be created for any kind of promotion. Always called with a calcDiscount behavior. Can have different internal calculations. Entire Member Order instance is passed as parameter. All other classes interact with Promotion super-type.

Slide 42

### Adapter Pattern

Pattern:	Adapter
Category:	Structural
Problem:	How to provide a stable interface to similar classes with different interfaces?
Solution:	Add a class that acts as an adapter to convert the interface of a class into another interface that the client classes expect.

19-42

**Teaching Notes**

Sales Tax Adapter class provides an unchanging interface to Member Order class (and others). Brand X Adapter translates interfaces of Sales Tax Adapter to interface of Brand X Sales Tax Calculator. If ever change from Brand X, have only to write a new adapter subtype.

Slide 43

### Frameworks and Components

**Object framework** – a set of related, interacting objects that provide a well-defined set of services for accomplishing a task.

**Component** – a group of objects packaged together into one unit. An example of a component is a dynamic link library (DLL) or executable file.

19-43

**Teaching Notes**

By using frameworks and components developers can concentrate on developing the logic that is new or unique to the application, thus reducing the overall time required to build the entire system.



Slide 44

### Additional UML Design and Implementation Diagrams

**Communication diagram** - models the interaction of objects via messages, focusing on the structural organization of objects in a network format.

**Component diagram** - depicts the organization of programming code divided into components and how the components interact.

**Deployment diagram** - depicts the configuration of software components within the physical architecture of the systems hardware "nodes."

19-44

**Teaching Notes**

Each of these diagrams will be illustrated on the following slides.

Slide 45

### Communication Diagram

1. Class  
2. Messages  
3. Self-calls  
4. Numbering scheme - messages should be numbered with a nested scheme.

19-45

**Teaching Notes**

A communication diagram is similar to a sequence diagram. But while a sequence diagram focuses on the timing or sequence of messages, a communication diagram focuses on the structural organization of objects in a network format. Sequence diagrams are generally better when you want to emphasize the sequence of calls. Communication diagrams are better when you want to emphasize the links. And they are easier to draw on whiteboards in when brainstorming alternative solutions.

Slide 46

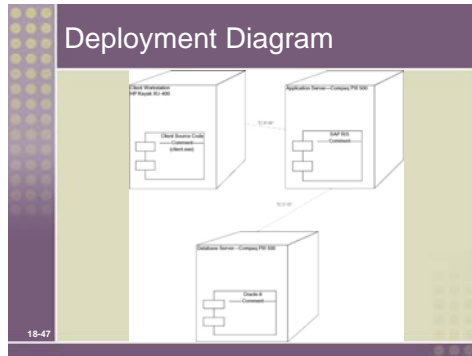
### Component Diagram

19-46

**Teaching Notes**

Component diagrams are implementation type diagrams and are used to graphically depict the physical architecture of the software of the system. They can be used to show how programming code is divided into modules (or *components*) and depict the dependencies between those components.

Slide 47

**Teaching Notes**

Deployment diagrams are also implementation type diagrams that describe the physical architecture of the hardware and software in the system. They depict the software components, processors, and devices that make up the system's architecture.

## Answers to End of Chapter Questions and Exercises

### Review Questions

1. Entity objects: they contain information about the business itself. They are also related to items in real life. In entity objects, there are attributes portraying the different instances of the entity. Their behaviors are also encapsulated.

Interface objects: they describe how users and the system communicate with each other. Only through interface objects can they interact. The two main responsibilities of the interface objects are:

- a. They convert the input from the users into information so that the system can understand and use the information to perform business events.
- b. They also take business-related data, translate the data, and present them to the users.

Control objects: they are used to make a business event performed by the system more robust and simplified. These objects contain application logic or business rules to facilitate the control.

2. It is necessary to have the structure of the object-based system divided into three kinds of objects to have their responsibilities and behaviors work together will make the maintenance, enhancement, and abstraction of the objects easier and simpler. Such an approach is also effective for the client/server model. The application logic (control objects) and the presentation logic (interface objects) will be put in the client side while the repository (entity objects) will be put in the server side.
3. Navigability is a kind of association between two classes. Even though most associations are bidirectional, some are not. Sometimes, it is necessary to limit the message of a class to be sent to only one direction. The other class will not have the ability to send a message back. For example, if there is a STUDENT object, it will only make sure for the STUDENT object to send a message to the GRADE object to request its grade. It is not common to use GRADE to find the STUDENT back. Thus, sometimes, it may be necessary to specify the navigability of an association between two classes.
4. Visibility in object-oriented design is the way attributes and methods are accessed by other objects. There are three levels of visibility: public, protected, and private.

If an attribute or method is public, any other methods or objects can invoke them.

If an attribute or method is protected, only the methods in the same class or the subclass where the attribute or the method belongs can be used to invoke them.

If an attribute or method is private, only the methods of the same class can invoke them.

5. The key reason for object reusability is to reduce system development time and costs. In order to do so, we have to design the objects using a good generalization/ specialization hierarchy. The main objective is to design an object in a very general way so that it can be used in other application as well.
6. Design patterns can be used to identify and document a set of common solutions used to solve a particular kind of problem. These solutions are grouped into a library maintained by the developers so that they can use the solutions to solve similar problems, instead of coming up with new solutions every single time.

In addition to that, developers will use object frameworks to achieve reusability. Object frameworks contain many related and collaborating objects to provide a set of services during development so that the development time can be shorter.

Lastly, components are used by developers. Components are objects being put together into one unit so that developers can share the programming codes with others easily.

7.
  - a. Improve the use-case model to show the actual implementation environment
  - b. Form interactions and behaviors of the object to support the use-case scenario
  - c. Update the object model to show the actual implementation environment
8. The goal is to refine the use-cases so that they will reflect the details of how the users will interact with the interface of the system and how the system reacts. Every single step of how the users interface with the system is included and described in detail.

This is essential because user manuals and testing will need to utilize use cases during system implementation. Programmers may need to make use of the use cases as well when they start building the systems.

9. Since interface objects are the interaction between the users and the system, upon the identification process, we should look for terms such as screens, windows, printers, or menu, in the use cases.

For control objects, we should look for items that may contain application logic or business rules. A rule of thumb is that a use case should show one control object for each unique system user.

For entity objects, we should look for attributes representing a business domain, such as customer address, product number, or product name.

10. The goal of constructing object robustness diagrams is to decide how objects interact with each other to perform a given business event. The components of the diagrams include the system user, the interface objects, controller objects, entity objects, and arrows showing the interaction.
11. We should look for all the verb phrases in the use cases. It is because verb phrases will indicate a behavior which needs to be completed in a use case scenario.
12. An object state is the trait and condition of an object at a particular point of time. The conditions of any given object can change. The object state is used to depict the different changes. For a change to happen, a state transition event will need to take place. State transition event is, therefore, the trigger of the change of an object.
13.
  - a. Identify the state of the object in the very beginning and at the end
  - b. Identify any other states that an object can have besides its initial and end state
  - c. Identify events that trigger the change of state
  - d. Identify when the change of state happens
14. Sequence diagram: it depicts the detailed interaction between the different objects in a time sequence.

Collaboration diagram: it shows how messages flow between different objects in message sequence.

15.
  - Classes
  - Associations, general or specific, and aggregation relationships
  - Attributes
  - Methods with parameters
  - Navigability
  - Dependencies

## Problems and Exercises

1. Object reusability is the main reason behind using object-oriented methods for systems development. Object reusability can dramatically reduce system development time, which in turn dramatically reduces costs.
- 2.

Programming Language	Project Duration (calendar months)	Level of Effort (person-months)	Software size (Lines of Code)
PL/1	30.0	240.0	41,800
OO Language	5.5	16.4	3,500

3.
  - True; 1) to indicate that a change occurring in one class may affect the other class, and 2) to show the association between a transient class and a persistent class.
  - True
  - False; no such term officially exists
  - False; interface classes are usually transient
  - True
4.
  - a. Display screen, keypad control panel, digital memory card reader, USB port, parallel port, paper input tray, paper output tray
  - b. Display screen, keypad, octane selection buttons, card reader, receipt printer, microphone/speaker, gasoline nozzle, ADC reader (Exxon Mobil as of this date)
  - c. ADC reader, electric eye, camera
5. Yes, use cases are refined by adding detailed descriptions of the following user-system interactions:
  - How actors (users) will actually interface with the system and how the system will actually respond in order to process the business event
  - The specific methods and interfaces by which the user accesses the system

The overall purpose is to transition the use cases from analysis-based use cases to design-based use cases

6.
  - Window controls, e.g., icons, buttons and links, are explicitly stated in system design use cases.
  - The term for a set of collaborating objects which are related, have an interface and which can act as a single unit is component.
  - To be able to reuse objects, they need to be designed correctly by defining

them within an appropriate generalization/specialization hierarchy so they are general enough for easy use in other applications.

- During the design phase, use cases and objects are refined to mirror the actual environment of the solution, rather than an environment based upon a logical ideal.

7. 1D, 2A, 3M, 4K, 5L, 6H, 7J, 8C, 9E, 10F, 11I, 12B, 13G
8. Your use case should be consistent with the principles and guidelines you learned in previous chapters. In addition, it should include detailed descriptions of the system-user interactions
9. Your matrix should show at least one interface object per actor or user, and at least one control object per unique user or actor. The entity objects should map to the business domain objects that you included in the design use case.
10. The purpose of the object robustness diagram is to create a model that can be used to help determine how objects need to interact in order to carry out the business event. The object robustness diagram represents messages and communications through the use of arrow, and uses symbols to represent actors, interface objects, control objects and entity objects.  
  
In your object robustness diagram, interface objects must be used to show actor-system interactions; control objects are used to coordinate messages between entity objects and interface objects.
11. Your table shows a behavior for each verb phrase in your design use case. Each automated behavior should be associated with an object type (control, entity or interface).
12. The purpose of the CRC card is to document all the use case behaviors, responsibilities and collaborations that are associated with an object type. The CRC card should show the object name, and include any sub or super objects. Behaviors and responsibilities may or may not have an associated collaborator; however, collaborators must be associated with a specific behavior.
13. There is no right or wrong answer to these questions per se. But your answers should indicate that you understand the essential differences between object-oriented analysis and design, and other approaches.

## Projects and Research

1. The intent of this question is to give the student detailed exposure to one of, if not the leading thinkers in the field of design and enterprise architecture, and to their theoretical underpinnings. Responses to these questions are open-ended, but should indicate at least a basic understanding of the rudimentary concepts espoused in Fowler's works.
2. As with the preceding question, this question is intended to broaden the student's base of knowledge vis-a-vis object-oriented design and reusability. The response should be a well-prepared and in-depth analysis, and one which indicates that the student has explored the writings of more than author on this subject.
3. This question is intended to provide practical experience and feedback to the student in a critical application of object-oriented design. Responses may vary in terms of subject matter, but should be consistent with the pattern illustrated in the textbook.
4. Same as with preceding question.
5. This question is intended to expose the student to extended object-oriented design structures. Responses should indicate a basic understanding of what component and deployment diagram are, and how they are used to describe physical architecture.
6. The intent of this question is to expose the student to object-oriented programming basics in order to gain a better understanding of how the object-oriented approach is used throughout the entire systems development life cycle. Responses are somewhat open-ended, but should be able to effectively link the impact of object-oriented analysis and design upon object oriented programming, and to provide a description of the basic constructs and processes used in object-oriented programming. .NET languages are (arguably) the most popular object-oriented programming languages at this time.

## Minicases

1. a. A striking difference between traditional UML and web-based UML is that web-based UML separates class diagrams and differs notations based on where the action is taking place. I.e. scripts that run on the server are shown on the server-side class diagram, or with server notations as opposed to client-side applications and scripting.



- b. Web-based, since the diagrams are for a website.
2.
    - a. Clearly, the work done for the mini-cases on this theme will differ dramatically. However, you should expect students to address issues such as source data automation, data handling, productivity and ease of use issues, and a generally sound analysis.
    - b. Refer to the chapter contents for referencing UML grading. Insist on clarity and completeness. The burden of acceptance is when the student can hand the diagrams over to someone, and that person can create the system without any other guidance.
    - c. Grade on correctness, completeness and professionalism.
  3. These answers will vary based on the project chosen. Note: check the source code for academic integrity violations, as many students will “borrow” code from peers.
  4. Note to professor: Insist that students test each other’s work with the intention of finding ALL of the flaws and making the prototype the best it can be. You will need to be clear that the class will not be graded on a curve – that if all prototypes are excellent, they can all get an “A.” (Otherwise, students will ‘forget’ to test a portion of the other team’s work!

### **Team and Individual Exercises**

There are no answers to this section.