Student Learning Outcomes

1. EXPLAIN THE PRIMARY DIFFERENCE BETWEEN THE TRADITIONAL TECHNOLOGY APPROACH AND THE OBJECT-ORIENTED TECHNOLOGY APPROACH.

2. LIST AND DESCRIBE THE FIVE PRIMARY OBJECT-ORIENTED CONCEPTS.

3. EXPLAIN HOW CLASSES AND OBJECTS ARE RELATED.

4. DESCRIBE THE THREE FUNDAMENTAL PRINCIPLES OF OBJECT-ORIENTED TECHNOLOGIES.

5. LIST AND DESCRIBE TWO TYPES OF OBJECT-ORIENTED TECHNOLOGIES.
The explosion of object-oriented technologies is radically changing the way businesses view information and develop information systems. Object-oriented technologies are everywhere in the business world today. It’s difficult to find a business or IT department that isn’t using object-oriented concepts and technologies. Every single Fortune 500 company is using some type of object-oriented technology. Knowledge workers everywhere are quickly learning how to write software in object-oriented programming languages, create databases using object-oriented database management systems, and design new systems using object-oriented analysis and design techniques. The race to learn and understand object-oriented concepts began many years ago and is still going strong.

Object-oriented programming is a revolutionary concept that has changed the way system developers create computer programs. Historically, computer systems were viewed as logical procedures that took in information, processed the information, and outputted the information. Object-oriented computer systems are viewed as a set of objects that are manipulated, and they focus on objects rather than on information and procedures separately. Objects are anything you need to store and manipulate in the system. Objects can include people, animals, employees, students, courses, payroll, buildings, desks, and more. Basically, everything stored in the system is an object. It’s easy to see that the object-oriented computer system view is drastically different than the traditional computer system view.

By reading this module, you’ll be taking a giant leap toward understanding object-oriented concepts, which is a great asset for you as a knowledge worker. Even though you may not plan to be a system developer, it’s still important that you learn the fundamentals of object-oriented concepts and technologies because you’ll be involved in analyzing, designing, and using object-oriented systems. Learning about object-oriented concepts and technologies will help prepare you for your future role as a knowledge worker in the information age. This module walks you through several important object-oriented concepts including

- The traditional technology approach
- The object-oriented technology approach
- The five primary concepts of object-oriented technologies, including information, procedures, classes, objects, and messages
- The three fundamental principles of object-oriented technologies, including inheritance, encapsulation, and polymorphism
- A detailed business case example
- The three types of object-oriented technologies including object-oriented programming languages, object-oriented databases, and object-oriented technologies and client/server environments

You might already be familiar with a few of the key terms and concepts related to object-oriented technologies. If you’ve ever read or heard the words information, procedures, classes, objects, or messages, then you’ve already been introduced to object-oriented concepts. If you’re unfamiliar with these concepts there’s no need to worry, because you’ll gain a solid understanding of each one of these important concepts in the next few sections. We certainly don’t claim that you’ll be an object-oriented expert, but you’ll get a great head start on understanding the whole object-oriented environment.

It is important to note that one of the primary goals of this module is to introduce you to the object-oriented approach for developing information systems. The object-oriented (OO) approach combines information and procedures into a single view. This
statement probably seems a bit confusing but, after reading the next few sections, it will become crystal clear. Just remember the key to understanding the object-oriented approach is to recognize that combining information and procedures is quite different from the traditional technology approach where information is separated from procedures.

**Traditional Technology Approach**

Let’s take a detailed look at the traditional technology approach in order to help you understand the object-oriented approach. The *traditional technology approach* has two primary views of any system—information and procedures—and it keeps these two views separate and distinct at all times.

**INFORMATION VIEW**

You’re probably already familiar with many terms that describe information, including data, variables, or attributes. All of these terms refer to the same thing and in this text we refer to these things as *information*. You can think of information as a key characteristic stored within a system. For example, if we built a system to store student information, key characteristics could include *Student Last Name* or *Student Phone Number*. The *information view* includes all of the information stored within a system. To gain a better understanding of the information view, let’s take a look at a sample student grading system.

In Figure G.1, you see all of the different types of information required to build the student grading system, such as *Student Last Name*, *Student ID*, and *Final Exam Grade*. The information view simply focuses on the information required to develop the system. The problem with the information view is that it gives no thought to how the information is used by the system. For example, it doesn’t define the weights associated with each assignment or exam to derive the *Final Course Grade*.

**PROCEDURE VIEW**

You’re probably also familiar with many terms that describe procedures, including formulas, functions, and methods. All of these terms also refer to the same thing and in this text we refer to these things as procedures. A *procedure* manipulates or changes
information. For example, if a student moved, we would need to change his or her address in our student grading system. The way we do this is to run an Update Student Address procedure that changes the Student Address information. The procedure view contains all of the procedures within a system.

There are four primary procedures, or ways, a system can manipulate information; they include create, read, update, and delete. These four procedures are commonly referred to as CRUD (Create, Read, Update, Delete). Can you think of any other procedures you might use to manipulate information in a system? How about calculate, run, save, cut, copy, spell check, select, overwrite, or others? These are all examples of procedures that manipulate or change information.

Can you identify the different procedures in Figure G.1? They include the basic CRUD on most of the information and a Calculate Final Course Grade procedure that calculates the average, based on various weights, for all of a student’s grades. These procedures define how the system uses the information. Do you see a problem with the procedure view? The problem is that it gives no thought as to what information is stored in the system. For example, there is a CRUD Student Address procedure, but there is nothing that ensures there is a Student Address information field.

THE PROBLEM WITH THE TRADITIONAL TECHNOLOGY APPROACH

The primary problem with the traditional technology approach is that the separate views can lead to potential disconnects between the information and procedures. For example, it’s possible to have the correct information but not be able to do anything with it because you don’t have the corresponding procedures. Likewise, you could have the correct procedures but not be able to do anything with it because you don’t have the corresponding information. In the first instance, it’s rather like having a workbook but no spreadsheet software. In the second instance, it’s like having the spreadsheet software but no information to work with.

Object-Oriented Technology Approach

The object-oriented approach bridges the gap between information and procedures by providing a holistic view of an information system. That is, an object-oriented approach combines information and procedures into a single view.

INFORMATION AND PROCEDURE VIEWS COMBINED

Let’s take a look at the student grading system in Figure G.2. This diagram represents the same student grading system as in Figure G.1, except this diagram uses an object-oriented approach. Notice the diagram represents a holistic view of the entire system with the information in the middle and the procedures surrounding the information.

When you build an object-oriented system you think of the procedures and information as a single unit. Think about how easy it would be for a knowledge worker to look at this diagram and instantly understand what types of information are in the system and what kinds of procedures can be performed on the information. Knowledge workers who use the object-oriented approach, looking at both information and procedures combined, find it easier to perform the following three important tasks:

1. Understand the entire system
2. Determine if any key information is missing that would be required in order to perform all of the procedures
3. Determine if any key procedures, which are required to manipulate the information, are missing.

OBJECT-ORIENTED APPROACH AND THE REAL WORLD

The object-oriented approach not only makes a knowledge worker’s job easier but also improves overall systems development because the system more closely models the real world. In the real world, you actually view a given process as a combination of information and the procedures you need to act on that information. Have you ever purchased a product that requires “some assembly,” such as a mountain bike or a gas grill? Upon opening the box and spreading out the contents you immediately reach for the instruction booklet. In the instructions you’ll find a detailed set of steps concerning assembly along with a description of the various parts. Both the information and procedures are provided together, which is similar to an object-oriented approach. You won’t find the set of instructions separate from the description of the parts; they are combined because this makes logical sense.

If you only viewed the procedures in the student grading system example, would they make any sense? Probably not, and it would be difficult for you to understand the procedures without understanding the information. For example, it would be impossible for you to understand the *Calculate Final Course Grade* procedure if you didn’t know what type of information was used in this calculation.

Let’s take a look at an inventory tracking process, another great example of how the object-oriented approach models the real world. If you consider a business process such as track inventory, you can quickly identify several key characteristics required to perform this process, such as *Part Number, Part Name, Part Manufacturer, Quantity on Hand, Reorder Point, and Cost*. These are all examples of information. At the same time, you can also identify how the information needs to be manipulated to perform this process, such as *Calculate Quantity to Order, Add a New Part, Change a Part Cost*, and so on. These are all examples of procedures. In short, you are considering both
information and procedures to be an integrated part of the track inventory process. So, object-oriented concepts basically provide a real-world view in which information and procedures are combined together.

### Five Primary Concepts of Object-Oriented Technologies

There are five basic object-oriented concepts that you as a knowledge worker should understand. We introduced you to two of these in the previous section. Can you guess which they are? If you said *information* and *procedures*, you’re correct. Although we’ve already discussed these concepts, we’ll briefly review them in this section. The other three basic object-oriented concepts are *classes*, *objects*, and *messages*.

#### INFORMATION

Information comprises key characteristics stored within a system. You’re already familiar with the different types of information stored in the student grading system such as *Student Last Name* and *Final Course Grade*, so let’s take a look at information stored in a different system. Imagine you’re building a dog tracking system for a dog kennel business. What types of information do you think the system needs in order to track different types of dogs? The answer to this can be any key characteristic you can think of related to a dog including *Name*, *ID*, *Breed*, *Color*, *Weight*, *Height*, *Owner*, and so forth. These are all examples of different types of information the dog tracking system could store.

#### PROCEDURES

A *procedure* manipulates or changes information. Again, you’re already familiar with the procedures in the student grading system, including *CRUD Student Last Name* and *Calculate Final Course Grade*, so let’s try to define the different procedures required to build the dog tracking system. What types of procedures do you think the system needs? The answer to this can once again be anything you think you’ll need in order to manipulate the dog information, including *CRUD Dog Name* and *CRUD Dog ID*. It’s also important to understand that procedures are used not only for CRUD on information, but also to perform functions or operations. Can you think of any other procedures that a dog tracking system might perform besides the basic CRUD on information? How about calculating the cost for a dog to stay at the kennel or scheduling employees who work at the kennel? Both of these are examples of procedures the system might need to perform.

#### CLASSES

A *class* contains information and procedures and acts as a template to create objects. Yes, classes are that simple. They are simply what we call the combination of information and procedures as displayed in Figure G.3 on the next page. It sometimes helps to think of classes as similar to a definition in a dictionary. Let’s return to our dog tracking system. If you looked up the definition of the word *dog* in a dictionary, it will give you an overview of what a dog is and explain what a dog can do. A class does exactly the same thing. If you look at the class *Dog* in Figure G.3, you can quickly understand all of the information required to describe a dog and many of the procedures the dog system can perform including *Calculate Cost for Dog Stay* and *Create Dog Name*. 
MULTIPLE CLASSES

For the student grading system and the dog tracking system, we’ve identified only a single class for each. Classes become a little more difficult when you start to think of having 50 or 500 different classes in a single system.

Can you guess why a single system might have so many classes? Imagine if you put all of the information for the dog tracking system into a single class: This class would be enormous. Just think of all the different types of information it might contain. There could be owner, diet, vaccine, accident, injury, training, and kennel inventory information to name just a few. If all of this information is in a single class, the class quickly becomes unmanageable. Determining which types of information belonged to which procedures would be impossible. Breaking down the information and procedures for ease of use and understandability, or practicing information decomposition, is a great way to structure your information system. Practicing information decomposition makes IT specialists’ and knowledge workers’ jobs easier because the information is in understandable pieces. There certainly isn’t anything stopping you from putting all of the information and procedures for the entire system into a single class, but this is a bad system design and leads to multiple system problems.

OBJECTS

An object is an instance of a class. This definition probably seems a bit confusing, so an easier way to think of an object is the actual item represented by the class.

Let’s take a look at the two objects (Gus and Oreo) of the class Dog represented in Figure G.4 on the next page. Can you describe the primary difference between the Dog class diagram in Figure G.3 and the Dog object diagrams in Figure G.4? Sure, you noticed the Dog object diagram contains the information representing the actual dogs that the system tracks. Using Figure G.4, you can describe the first object, Gus. Gus is a black-and-white Border collie owned by Alex Coombe. You can also describe the second object, Oreo. Oreo is a brown St. Bernard owned by Taylor Coombe. Every dog the
system tracks is a separate object, and every dog object is an instance of (or created from) the class Dog. Gus and Oreo are both examples of objects of the class Dog.

**HOW CLASSES AND OBJECTS INTERACT**

Do you remember the original definition for a class? Classes contain information and procedures and act as templates to create objects. Essentially, the class is a blank template that defines all the different types of information the system can store about an object. Once you create an object from the class, you can fill in the template with the actual information. For example, the Dog class can give you a high-level definition of a dog object, such as Dog Name, but it can’t tell you the actual name of the dog. The class will only tell you that it can store information for the dog’s name. The object stores the actual information and can tell you the dog’s name is Gus.

**MESSAGES**

Now that you know something about objects, you may be asking if objects communicate with each other and, if so, how? These are great questions, and the fact that you’re asking them means you’re quickly becoming an object-oriented expert.

Messages are how objects communicate with each other. One object can send a message to another object asking it to perform a certain procedure. But why do objects need to communicate with each other? In general, systems are created by developing many different classes that work together to perform tasks. Let’s refer back to our student grading system. If you build this system, you might have one class representing students, a second class representing courses, and a third class representing instructors. Objects from each of the classes communicate with each other in order to enter the student’s course grades. The student object passes a request to enroll in courses to the course object. The instructor object passes a request to teach specific courses to the course object. The instructor object also passes the student’s final course grades to the course object. Combining these three classes together gives you a fully functioning student grading system.
HOW THE FIVE PRIMARY CONCEPTS INTERACT

Let’s review information, procedures, classes, objects, and messages as important object-oriented concepts and how they relate to each other.

• Information and procedures create classes
• Classes create objects
• Objects communicate with other objects via messages

Let’s take a look at the individuals who build the systems and the individuals who use the systems as another perspective on how the object-oriented concepts relate to each other.

System developers are the individuals who build the systems and are responsible for building the classes which contain information and procedures. System users are the individuals who use the system and are responsible for inputting the actual information or creating the objects. If you think about the dog tracking system, you can easily identify which individuals are responsible for the different tasks. The system developer would be responsible for building the Dog class and the associated information and procedures. The kennel owner, or system user, would be responsible for creating the objects or inputting the actual dog information.

Because you understand the basic concepts of information, procedures, classes, objects, and messages, you now understand the basics of object-oriented technologies. Next, let’s take a close look at some real-world examples of object-oriented systems.

Real-World Object-Oriented Example

Can you think of something you use every day that is a good example of an object-oriented system? Here’s a hint: A real-world object-oriented system has several different components working together to create a single system. Does that help you? We bet you can think of several different everyday items that are great examples of object-oriented systems.

How many of you thought of a car? The steering wheel, tires, and engine are different components that work together in order to accomplish the common goal of driving. How many of you thought of a computer? A computer is a great example of several different
components working together and sending messages to each other in order to accomplish a common task. (See Extended Learning Module A for how computer components work together and send messages to each other.) The keyboard, monitor, mouse, and operating system all work together to run applications. How many of you thought of a home stereo system?

A home stereo system is a perfect analogy for an object-oriented system (see Figure G.5). If you created a class to represent each stereo component, they could include some or all of the following:

- Amplifier
- CD player
- Cassette deck
- Equalizer
- Speakers
- CDs
- Cassettes

Can you name some of the information items stored within the CD class? Title, Artist, Date Recorded, and Number of Songs are a few. Can you determine how many objects would be created from the CD class? This number varies depending on how many CDs you own. Can you guess what the primary procedure is in the amplifier object? If you guessed Set Volume, then you’re correct. In fact, manipulating the volume of the music is a primary procedure of the amplifier object.

In order for the system to work, objects are created from each class, and each of the objects work with specific information and procedures. The information for the CD player, for example, could include the Manufacturer Name, Model Number, or Play Speed. The procedures for the CD player could include Play, Fast Forward, Rewind, Skip, or Stop. If you called the Play procedure for the CD player object, do you think you would be able to hear music? No, you would not be able to hear anything. The CD player can play the CD, but it has to send a message to the amplifier object, which sends a message to the speaker object to play the music.

A home stereo system is a true example of a real-world object-oriented system. Each component must work together in order for the system to function. Thus, each component in a home stereo system really is an object. As an object, each component works with only certain information and performs certain procedures. If one component needs another procedure performed, it must send a message to another object (or component) that can perform that procedure.
A business can gain huge advantages by using object-oriented technologies. The three fundamental principles of object-oriented technologies are

1. Inheritance
2. Encapsulation
3. Polymorphism

**INHERITANCE**

One of the most powerful features of object-oriented technologies is inheritance. Inheritance is the ability to define superclass and subclass relationships among classes. Generalization (parent) and specialization (child) relationships are another way of thinking of superclass and subclass relationships. Take a minute and review Figure G.6 and try to determine which class is the superclass and which classes are the subclasses? The Car class is the superclass and the Bronco and Porsche classes are the subclasses. Another way to state this relationship is that the Bronco and Porsche subclasses inherit all of the information and procedures from the Car superclass. For example, the CRUD procedures are not defined in the Bronco and Porsche subclasses because they are inherited from the Car superclass.

Defining inheritance is simply a matter of defining generic and specific information and procedures. Generic information and procedures apply to all classes. Specific information and procedures apply only to a particular subclass. For example, notice the Car superclass in Figure G.6 contains generic information and procedures that are shared by both the Bronco and Porsche subclasses. Both the Bronco and Porsche subclasses have a Model, Year, Price, and Color and both can honk horn, break, and drive.

The subclasses contain specific information and procedures that are unique to each particular subclass. The Bronco subclass contains information for 4-Wheel Drive and the Porsche subclass contains a procedure to Drive over 140 mph. These unique features are stored in the subclass because they are not generic enough to store in the Car superclass. If you stored the procedure for Drive over 140 mph in the Car superclass, then every subclass of Car inherits this procedure. This is a rather big mistake since there are many types of cars that can’t drive over 140 mph. Hence, we must store this unique feature at the subclass level.

Inheritance, or the ability to define superclass and subclass relationships, is a unique feature of object-oriented technologies. Can you think of the benefits that inheritance brings to an IT department or system?

**BUSINESS BENEFITS OF INHERITANCE: REUSE**

The true business benefit gained from using inheritance is the ability to easily expand and maintain a system. Let’s look at an example of how defining a superclass allows a system to be easily expanded. If you decided to add a piece of information called Mileage to the Car class, it’s only developed in one place, the Car class, and it’s automatically inherited by every subclass. The Mileage information is reused by all of the subclasses.

If you’re using the traditional approach to develop this system, the Mileage information must be developed in each of the Car, Bronco, and Porsche classes because the traditional approach does not support the concept of inheritance or reuse. Imagine if you
had 50 subclasses of the Car class. Obviously, if you used the traditional approach it
would take a great deal of time and energy to build Mileage information into all 50 sub-
classes. Using the object-oriented approach you only build it once and all 50 subclasses
automatically inherit this valuable piece of information.

Another example of how inheritance makes it easy to expand a system would be if you
decided to add a new class, Volkswagen Beetle, to the system. If you placed this class as
a subclass of Car, it automatically inherits all of the information and procedures already
developed in the Car class. All you need to develop is any unique information or proce-
dures that the Volkswagen Beetle class requires. If you added a Volkswagen Beetle class
using the traditional approach, you must build every single piece of information and
every procedure including Model, Year, Price, Color, CRUD, Honk horn, drive and
break. Inheritance is obviously a valuable principle for businesses in terms of saving
money, effort, and time when developing and maintaining information systems.

Figure G.6
Inheritance
CAN YOU FIND THE ERRORS IN THIS CLASS DIAGRAM?

Take a look at the Car class diagram and determine what errors exist. If you’d like a hint, think about information, procedures, and inheritance.
ENCAPSULATION

Encapsulation means information hiding. This concept has a simple definition and provides tremendous benefits when you’re building an information system. Let’s take a look at how object-oriented technologies encapsulate information.

Objects are sometimes referred to as black boxes, whereby the information inside an object is hidden and all that can be viewed is the object. Imagine you’re in a park and you see a dog. You can instantly determine the dog is an object of the class Dog. However, you don’t know all of the information about the dog. You can’t tell the dog’s name, weight, height, or other characteristics just by looking at it. Hence, the actual information stored in the object is hidden, or encapsulated, but this doesn’t prevent you from seeing the dog and being able to communicate with the dog.

Let’s take a look at an example of encapsulation. If you order AT&T digital cable, you receive a remote control that is used as the interface to the digital cable box. An interface is any device that calls procedures and can include such things as a keyboard, mouse, and touch screen. Recently, AT&T changed its entire digital cable system including the menu colors, item locations, and cable features. Suddenly, when a user turned on the television everything on the digital cable menu looked completely different. However, the remote control, or system interface, continued to work exactly the same as it did before the system changed. The same buttons on the remote control were used to turn the TV on and off, to adjust the volume up and down, and to select different menu items. Since the remote control didn’t change, the customers could use their system exactly the same way as they did before. The only change the users had to deal with was getting used to the new look and feel of the improved digital cable system.

BUSINESS BENEFITS OF ENCAPSULATION: QUALITY

Can you explain the business benefits AT&T received from its use of encapsulation? AT&T used encapsulation to hide the digital cable system changes from its users. The system changed significantly but, because the user’s remote control, or interface, continued to work exactly the same, the users were unaffected. The users continued to receive the same high-quality service from AT&T. In other words, the system changes were hidden from the system users. The users operated the digital cable system exactly the same way as they did before the changes, through their remote control. This is a great example of the huge benefits a company can receive by using encapsulation. Just imagine the millions of dollars AT&T would be required to spend if it had to retrain all of its customers every time it wanted to make a system change.

POLYMORPHISM

Polymorphism is an impressive word and, like so many technical words, though it looks and sounds intimidating, it has a very simple meaning. Polymorphism simply means “to have many forms.” A great example of polymorphism is the word bark. What do you think of when you hear the word bark? We bet many of you will think of a dog’s bark, but the word could also refer to tree bark. This is true polymorphism, the ability to use the same word to mean different things.

Let’s take a look at Figure G.7 on the next page for a great example of polymorphism. Notice the Rectangle, Square, and Circle classes all contain procedures called CalculateArea. However, the CalculateArea procedure is different for each class. The formula to calculate the area of a rectangle (Length * Width) is completely different from the formula to calculate the area of a square (Side * Side). This is a great example of polymorphism because each class has an identically named procedure that performs different calculations.
BUSINESS BENEFITS OF POLYMORPHISM: PRODUCTIVITY

If you build a procedure using a traditional technology, you must ensure that the procedure has a unique name. Can you imagine how difficult this would be if there were 1,000 procedures in the system? Using an object-oriented approach removes the problem of defining complex naming structures and increases productivity.

Putting It All Together: A Business Example

Let’s assume you’re starting your own business, Ice Blue Snowboards. Your business manufactures and sells snowboards, bindings, boots, and apparel. In order to prepare for the launch of the business, you researched similar businesses to discover any problems they’ve encountered so you can avoid making the same mistakes. The following is a list of common competitor problems:

- Eighteen months to get a new product to market
- Inventory control
- Scalability and expandability
Let’s take a detailed look at each problem and discuss how using an object-oriented approach will help you minimize or eliminate these problems.

**EIGHTEEN MONTHS TO GET A NEW PRODUCT TO MARKET**

In order for your business to be competitive you must be able to get your new products on the market as soon as they’re finished. Having a fast time-to-market is critical for the success of your new business. Eighteen months is simply too long for you to wait to get a new product on the market. Using an object-oriented approach will help you reduce this critical time-to-market factor. The typical high-level processes for developing new products include the following:

- Generating and accepting the idea
- Manufacturing the product
- Updating all current systems to support the new product
- Implementing the new product

Each time a new product is developed, it takes almost 18 months for your competitors to get it on the market. I bet you’re asking how this process could take 18 months. The reason this process takes so long is primarily because several applications must be changed every time a new product is introduced. A typical business will have a different system for every business function including inventory, sales, orders, marketing, billing, and so forth. Every system must be updated with the new product information before you can begin selling the product to your customers. Can you think of how using an object-oriented approach could help you decrease the time and effort required to update every system?

The following is an example of how you could define an object-oriented snowboard selling system. First, a class called Snowboard would be created. The Snowboard class would be responsible for tracking all snowboard information including Pricing, Model, Features, Discounts, and so on. The Snowboard class would also be responsible for establishing all of the procedures associated with marketing and selling snowboards including advertising and promoting in magazines and competitor pricing analysis. Second, an interface would be designed to perform all of these procedures. This would encapsulate the information and procedures in the Snowboard superclass and allow you the flexibility to change things without affecting the knowledge workers. Third, you would practice inheritance. A subclass would be created for each particular type of snowboard. Defining subclasses, or using inheritance, saves you a great deal of time and energy because each subclass inherits all of the information and procedures from the superclass. The only work required to create a subclass is defining the unique information or procedures associated with the subclass. After creating each class once you can reuse the classes across all of your business applications. Finally, you would create objects that communicate with all of the other objects across every system in your business.
ONE HOUR TO GET A NEW PRODUCT TO MARKET

Defining a new snowboard product would now require no more than one hour. The following are the steps required to define a new product:

- Create a single subclass for the new product
- Assign a superclass/subclass relationship between the new class and the Snowboard class
- Define any unique pieces of information for the new product subclass
- Define any unique procedures for the new product subclass
- Reuse the new product subclass by copying it directly into every business application
- Create objects from the new subclass

By understanding and using object-oriented technologies, you’ve decreased your product introduction time from 18 months to 1 hour. The key to reducing the time-to-market for your products is having business systems that use the same generic interfaces for dealing with products. Every time you introduce a new product, as long as it conforms to the common generic interface, all of the business applications are able to add it without any modification or interruption to the knowledge workers.

Figure G.9
Snowboard Business Case
Class Diagram
Figure G.9 is a great example of a snowboard business case class diagram. This figure assumes your business offers an electronic catalog on the Internet listing all of the snowboarding products you sell. By creating a catalog object whose primary procedure is to update the electronic catalog, you wouldn’t be required to change anything when adding a new product to the catalog. The catalog object already understands how products are added and listed in the catalog. The new product object simply sends a message to the catalog object to execute the update catalog procedure. The new product would be added to the catalog without any need for system modification.

INVENTORY CONTROL

Controlling inventory is one of the largest problems facing businesses today. Businesses need to be able to have sufficient inventory on hand to meet current production needs while minimizing the associated expenses including storing the inventory, transporting the inventory, and maintaining the inventory. This problem is a part of supply chain management and it hits all businesses small and large. Managing the supply chain is fundamental to the success of any business. If you neglect to manage your supply chain, then you’ll find your business experiencing high storage costs, the inability to manufacture products due to low inventories, and lost inventory.

Can you guess how supply chain management can be implemented efficiently using object-oriented technologies? You can define \textit{Shipping, Distribution,} and \textit{Vehicle} classes. Creating shipping, distribution, and vehicles objects allow you to control your inventory. Shipping objects know their origin, their destinations, and their primary goal of arriving at the destination on time with the lowest expense. Distribution objects designate which modes of transportation will be used to ship the inventory. Vehicle objects move the inventory. Using these three objects will help your business track and maintain correct inventory levels.

SCALABILITY AND EXPANDABILITY

\textit{Scalability} refers to how well your system can adapt to increased demands. Increased demands can include such things as additional users, information, and processing speeds. When a system grows, the size of the database that stores the information grows. As the database grows, the system tends to perform procedures slower. \textit{Expandability} refers to how easy it is to add features and functions to a system. If you design a system without thinking about expandability, you’ll run into major problems when your business starts to grow.

Take a look at Figure G.10 on page 19. Do you notice a problem with the class diagram? Can you explain why the \textit{ExpertSnowModel} class was unable to be placed in a superclass/subclass relationship with the \textit{Snowboard} class? As we discussed, it’s important to take advantage of inheritance as it saves time and effort when developing a system. The reason the \textit{ExpertSnowModel} class can’t inherit the functionality from the \textit{Snowboard} class is because of the procedure called \textit{Deep Powder Specific}. \textit{Deep Powder Specific} is a procedure that doesn’t apply to all types of snowboards. Snowboards are designed for powder, ice, and all mountain terrain. For this reason, the \textit{ExpertSnowModel} class can’t be designated as a \textit{Snowboard} subclass. If you did place the \textit{ExpertSnowModel} class as a subclass of Snowboard, then an \textit{ExpertSnowModel} is now able to perform the \textit{Deep Powder Specific} procedure. This is a rather large system error. Defining superclasses that are too specific and not being able to take advantage of inheritance is a common problem with object-oriented systems.
Did you notice anything else wrong with the class diagram? The *ExpertSnowModel* class doesn’t contain any information. This indicates a disconnect between the information and procedures. The *ExpertSnowModel* class contains the Create, Read, Update, and Delete procedures, but what are these procedures going to manipulate if there isn’t any information? This is a typical error if you are using the traditional technology approach because information and procedures are viewed separately. Using the object-oriented approach, you notice this disconnect right away.

Now that you are armed with the powerful knowledge associated with object-oriented technologies you’ll be able to build all of the systems needed to start your Ice Blue Snowboards business. You’ll even be able to avoid many of the common pitfalls your competitors have faced.
Types of Object-Oriented Technologies

Many technologies in use today support object-oriented concepts and techniques. Many more of these technologies are being developed every day as the business world continues to rush toward the use of objects. As a knowledge worker, it’s important that you have some general knowledge of the different types of object-oriented technologies available and used throughout IT departments.

OBJECT-ORIENTED PROGRAMMING LANGUAGES

In general, programs are what make computers work. A program is a set of instructions that, when executed, cause a computer to behave in a specific manner. A program is almost like a recipe. It contains ingredients and directions, or information and procedures that tell the computer how to perform different tasks. Can you think of any examples of programs? Microsoft Word and Excel are both examples of programs.

A programming language is the tool developers use to write a program. For example, English, French, and Italian are all different types of languages you can use to write a paper. Java and C++ are two different types of languages you can use to write a program. An object-oriented programming language is a programming language used to develop object-oriented systems. Programming languages that are not object oriented can’t handle classes, objects, messages, inheritance, or encapsulation. Currently, there are close to 100 different object-oriented programming languages available. The three most popular languages today are Java, C++, and Smalltalk.

OBJECT-ORIENTED DATABASE SYSTEMS

Relational databases (the most popular, which you learned about in Chapter 3) organize information into fields, records, and files (or relations). Object-oriented databases work with traditional database information and also complex data types such as diagrams, schematic drawings, video, and sound and text documents.

The relational database model, although it may allow you to store and view such data types, does not include good mechanisms for allowing you to manipulate and change information within those data types. For example, you can include a CAD drawing of a part as a field in a relational database, but it’s literally impossible to work with any specific information in the drawing (such as cuts, specific components, and the ordering of assembly) without having that information also stored in other fields.

Another feature of object-oriented databases is that you are not restricted to two-dimensional tables. This gives you greater flexibility in storing and defining procedures that work with complex data types. In fact, most of today’s multimedia applications rely on the use of objects and object-oriented databases.

Most other database models also restrict you to working with specific data types: alphabetic, numeric, decimal, currency, date, and so on. In an object-oriented database environment, however, you can create and work with data types that may be unique to a specific business process. For example, if you had an object that included an address you could easily design this field to include a street address and a unit number. You could then define a procedure that requires the entry of both pieces of information. This is an example of a unique data type that requires not only a street address but also a unit number.
Client/server is the emerging blueprint for organizational networks, and most organizations are choosing to develop client/server networks through object-oriented technologies. A client/server network is a network in which one or more computers are servers and provide services to the other computers, which are called clients. Spreading objects across a client/server network makes logical sense: client workstations contain objects with local procedures for working with local information, and servers contain objects with global procedures for working with global information.

In Figure G.11, the server handles the entire data management function, the client handles the entire presentation function, and both share in processing the logic or business rules. So the server object contains procedures for retrieving and storing information (data management) and for processing some of the logic or business rules. Likewise,
RESEARCHING POET

One of the most popular object-oriented databases is Poet. You can find information about Poet at www.poet.com. Visit that site and do some research. As you do, answer the following questions:

1. How can you learn more about Poet? Web seminar? Face-to-face seminar? Other means?

2. Who are some customers of Poet?

3. Who are some business partners of Poet?

4. To what extent do Poet solutions focus on “e” or electronic business initiatives?

5. What is a catalog management solution?

THE FUTURE OF OBJECT-ORIENTED TECHNOLOGIES

The number of object-oriented development tools increases daily and with this increase the importance of understanding object-oriented concepts becomes even more critical. In the future, object-oriented tools will perform tasks and provide functionality that we haven’t even thought of yet. We hope this module has provided you with a solid understanding of object-oriented technologies and concepts to prepare you for your job as a knowledge worker.
1. **Explain the primary difference between the traditional technology approach and the object-oriented technology approach.** The primary difference between the traditional technology approach and the object-oriented technology approach is the way information and procedures are viewed and developed. The *traditional technology approach* has two primary views of any system—information and procedures—and it keeps these two views separate and distinct at all times. The primary problem with this approach is that the separate views allow for information disconnects. For example, you might have all of the required information but not have the correct procedures to manipulate the information. The *object-oriented (OO) approach* combines information and procedures into a single view.

2. **List and describe the five primary object-oriented concepts.** The five primary object-oriented concepts include:
   1. Information is any key characteristic stored within a system.
   2. A *procedure* manipulates or changes information.
   3. A *class* contains information and procedures and acts as a template to create objects (instances of a class).
   4. An *object* is an instance of a class.
   5. *Messages* are how objects communicate with each other. One object can send a message to another object asking it to perform a certain procedure.

   In general, the combination of information and procedures creates a class. A class creates objects, and objects communicate with other objects via messages. A system usually contains many classes, and many objects can be created from a single class.

3. **Explain how classes and objects are related.** Classes contain information and procedures and are used as a template to create objects. The class is basically a blank template, which defines all of the different types of information the system can store about an object. An object can only be created from a class. Once you create an object from a class, you can fill in the template with the actual object information.

   One way to describe the interaction between classes and objects is to think of a class as a high-level definition and to think of an object as a real item. For example, the Dog class can give you a high-level definition of a dog object, such as *Dog Name*, but it can’t tell you the actual name of the dog. The class will only tell you that it can store information for the dog’s name. The actual name of the dog, for example Gus, is stored in the dog object.

4. **Describe the three fundamental principles of object-oriented technologies.** The three fundamental principles of object-oriented technologies include the following:
   - **Inheritance** is the ability to define superclass and subclass relationships among classes. The subclass inherits all of the information and procedures from its superclass. Other names for superclass/subclass relationships include parent/child relationships and generalization/specification relationships.
   - **Encapsulation** means information hiding. A remote control is used to control AT&T’s digital cable. The remote control hides the system information from the knowledge workers.
   - **Polymorphism** simply means to have many forms. Basically, polymorphism provides you with the ability to use the same word and have it mean different things. The word *Bark* is an example of polymorphism. Bark can refer to a dog’s bark or tree bark. The same word can be used to mean different things.

5. **List and describe two types of object-oriented technologies.** One type of object-oriented technology is an object-oriented programming language. An *object-oriented programming language* is a programming language used to develop object-oriented systems. For example, English, French, and Italian are all different types of languages you can use to write a paper. Java and C++ are two different types of languages you can use to write a program. An object-oriented programming language must be used in order to develop an object-oriented system.
A second type of object-oriented technology is an object-oriented database system. **Object-oriented databases** work with traditional database information and also complex data types such as diagrams, schematic drawings, video, sound, and text documents.

### Key Terms and Concepts

- Class, 6
- CRUD, 4
- Encapsulation, 14
- Expandability, 18
- Information decomposition, 7
- Information view, 3
- Inheritance, 11
- Interface, 14
- Message, 8
- Object, 7
- Object-oriented approach, 4
- Object-oriented database, 20
- Object-oriented programming language, 20
- Polymorphism, 14
- Procedure, 3
- Procedure view, 4
- Program, 20
- Programming language, 20
- Scalability, 18
- Traditional technology approach, 3

### Short-Answer Questions

1. What is the combination of information and procedures?
2. What is an example of a real-world object-oriented system?
3. What are classes?
4. How are messages used?
5. What is the information view?
6. What is an example of information?
7. What is information decomposition?
8. Why would you use information decomposition?
9. What is an instance of a class?
10. What is a procedure?
11. How do objects and classes relate?
12. How do superclasses and subclasses relate?
13. What is polymorphism?
14. What is an object?
15. Why is scalability important when building a system?
16. What is a benefit of using the object-oriented approach?
17. What is an interface?
18. What is another term for a superclass/subclass relationship?
19. Why is expandability important when building a system?
20. What is another term used to describe information hiding?

### Short-Question Answers

For each of the following answers, provide an appropriate question:

1. Procedure view.
2. Created from a class.
3. Student Last Name.
4. CRUD.
5. Actual item stored in a system.
6. Contains multiple classes.
8. Instance of a class.
10. Key characteristic of a system.
11. Scalability.
13. Separates information and procedures.
Assignments and Exercises

1. CLASSES IN THE CLASSROOM  Take a look around your classroom and on a piece of paper list 20 different objects located in the classroom. Objects might include desks, chairs, lights, students, and so forth. If you were going to build a classroom inventory tracking system, how many classes would you need to define in order to track all of the objects? What would be the name of each class? What information and procedures would be stored in each class? On a separate sheet of paper, draw a class diagram displaying all of the classes along with the different types of information and procedures they contain. Be sure to try to take advantage of inheritance. Take both sheets of paper and match each object to an appropriate class. If all of the objects match to a class, you created a successful system. Chances are every student’s class diagram is probably going to be different because there is no right or wrong answer to this exercise, so be creative and have fun defining your classes. Be sure to look at some of the other student’s diagrams to see how they defined their classes.

2. TREES-R-US  You’ve been hired to build an inventory tracking system for the Trees-R-Us landscaping company. Trees-R-Us is excited about the use of inheritance in object-oriented systems and wants to see how you’re going to use it in the system. Trees-R-Us has already defined the Tree, Grass, Flowers, Fence, Equipment, and Plant classes. Your job is to define all of the information and procedures for each class and the inheritance structure of the classes, or the superclass and subclass relationships. Please provide a class diagram that displays all of the classes for the Tree-R-Us inventory tracking system and be sure to include inheritance. Again, there is no right or wrong answer for this exercise, so be creative and look at some of the other students’ class diagrams to see how they defined their classes.

3. OBJECT-ORIENTED CONCEPTS AND A REAL-WORLD SYSTEM  Create a list of a computer and all of its parts. Be sure to include the monitor, keyboard, mouse, hard drive, disk drive, memory, CD-ROM drive, software, and printer. Write a brief explanation answering each of the following:
   - What types of information are associated with the computer?
   - What types of procedures are associated with the computer?
   - What parts of the computer, if any, represent the classes?
   - What parts of the computer, if any, represent the objects?
   - How many classes are there?
   - How many objects are there?
   - How are messages used?
   - How do all of the components work together to create a complete system?

4. UNDERSTANDING OBJECT-ORIENTED CONCEPTS AND TERMINOLOGY  Create a brief presentation explaining the primary differences between each of the following:
   - Traditional technology approach and object-oriented technology approach
   - Information and procedures
   - Classes and objects
   - Messages and interfaces
   - Encapsulation and inheritance
   - Generalization and specialization

Feel free to use any of the figures located in this module. You can find them on the Web site that supports this text at www.mhhe.com/haag.

5. EXPLAINING OBJECT-ORIENTED TECHNOLOGIES TO YOUR MANAGER  Assume you’re working for a large oil and gas company. Your current manager has very little experience with object-oriented technologies and has asked you to write a paper describing, in generic terms, each
of the following object-oriented concepts. Be sure to include explanations on how using these object-oriented concepts will contribute to building and implementing successful information systems.

- Encapsulation
- Polymorphism
- Inheritance

6. CLASSES AT THE VIDEO STORE Consider your local video rental store. What would be three important classes? How many different objects do you think there are for each class? On a separate sheet of paper, draw a class diagram displaying all of the classes along with the different types of information and procedures they contain. Do you think video rental stores in general use object-oriented systems? Why or why not?

7. RESEARCHING JAVA Java is perhaps the most popular of all the object-oriented programming languages. It was developed by Sun Microsystems in the early to mid-1990s. Connect to http://java.sun.com (Sun’s Web site for Java) and click on the Case Studies link. Find a case study that interests you about a company that has used Java to implement an object-oriented system. Prepare a short report for your class detailing that case study. Now, do some more research on the Web. Although Java is currently the dominant object-oriented programming language, there are some new ones on the horizon that will compete against it. Find one such new and emerging object-oriented programming language. What is it? How is it designed to compete with Java? Who provides it?

PHOTO CREDITS

Fig. G.5a, p. 10, Photo courtesy of Bose Corporation
Fig. G.5b, p. 10, © The Image Bank
Fig. G.5c, p. 10, © Nance Truworthy

Fig. G.8a, p. 16, Snowboard © David Madison Sports Images
Fig. G.8c, p. 16, Snowboard © David Madison Sports Images