REVIEW LESSON

MTA Course: Software Development Fundamentals
Lesson name: Software Development Fundamentals 2.4
Topic: Understand encapsulation (One 50-minute class period)
File name: SoftDevFund_RL_2.4

Lesson Objective:

2.4: Understand encapsulation. *This objective may include but is not limited to:* creating classes that hide their implementation details while still allowing access to the required functionality through the interface; access modifiers

Preparation Details

Prerequisite student experiences and knowledge

Students should have had experience with using modifiers such as `public` and `private` to restrict access to class members and unified modeling language (UML) diagramming as a means to present an overview of a class or inheritance hierarchy. This MTA Certification Exam Review lesson is written for students who have learned about object-oriented programming. Students who do not have the prerequisite knowledge and experiences cited in the objective will find additional learning opportunities using resources such as those listed in the Microsoft® resources and Web links at the end of this review lesson.

Instructor preparation activities

None

Resources, software, and additional files needed for this lesson:

- SoftDevFund_PPT_2.4

Teaching Guide

Essential Vocabulary:

encapsulation—In object-oriented programming, the packaging of attributes (properties) and functionality (methods or behaviors) to create an object that is essentially a "black box"—one whose internal structure remains private and whose services can be accessed by other objects only through messages passed via a clearly defined interface (not to be confused with the `interface` keyword). Encapsulation ensures that the object providing service can prevent other objects from manipulating its data or procedures directly, and it enables the object requesting service to ignore the details of how that service is provided. Also known as *information hiding.*

`internal`—an access modifier for classes and class members. Internal members are accessible only within files in the same assembly.

`private`—a member access modifier. Private access is the least permissive access level. Private members are accessible only within the body of the class in which they are declared.

`protected`—a member access modifier. A protected member is accessible from within the class in which it is declared, and from within any class derived from the class that declared this member.

`public`—an access modifier for classes and class members. Public access is the most permissive access level. There are no restrictions on accessing public members.

Lesson Sequence

Activating prior knowledge/lesson staging (5 minutes)

1. Show the Activator slide in the Microsoft PowerPoint® presentation for this lesson.
    a. *Ask:* How is encapsulation like a black box?
    b. Encapsulation *hides* the implementation of class members. In the case of the `drawCircle()` method, we know that we input a number and we get a circle drawn on the screen, but we don't know how the method did it.

Lesson activity (35 minutes)

1. Show the PowerPoint presentation.
    a. Review the definition of encapsulation and discuss the access modifiers.
    b. Explain the `Student` class example of encapsulation.

c. Demonstrate how to encapsulate class data using methods called accessors (getters) and mutators (setters) or properties.

d. Describe how access to class members and classes themselves can be controlled using access modifiers.

e. Go through each of the access modifiers and explain the examples. Give extra situations in which access should be modified.

Assessment/lesson reflection (10 minutes)

1. Show the Lesson Review slide in the PowerPoint presentation.
2. Allow students some time to provide examples of situations in which the following keywords should be used: `public`, `private`, `protected`, `internal`
3. Students should be given some time to write and share their answers with a peer. Call on students to share their answers with the class.

Microsoft resources and Web links

Access Modifiers (C# Programming Guide)
http://msdn.microsoft.com/en-us/library/wxh6fsc7%28VS.71%29.aspx

Suggested best practices:

- None

Additional notes to the instructor:

- The extended lesson reflection replaces a lesson worksheet.