REVIEW LESSON

MTA Course: Software Development Fundamentals
Lesson name: Software Development Fundamentals 2.2
Topic: Understand inheritance (One 50-minute class period)
File name: SoftDevFund_RL_2.2

Lesson Objective

2.2: Understand inheritance. *This objective may include but is not limited to:* inheriting the functionality of a base class into a derived class.

Preparation Details

Prerequisite student experiences and knowledge

Students should have experience with creating inheritance hierarchies and understand the "is-a" relationship concept associated with inheritance. This MTA Certification Exam Review lesson is written for students who have learned about object-oriented programming, including classes. 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_2.2
- SoftDevFund_SA_2.2
- SoftDevFund_SA_2.2_key

Teaching Guide

Essential Vocabulary:

abstract class—a class from which no objects can be created. An abstract class is used to define subclasses, and objects are created from the subclasses.

base class—a class from which other classes have been or can be derived by inheritance.

derived class—a class created from a base class. A derived class inherits all the features of its base class. It can include additional data elements and routines, redefine routines from the base class, and restrict access to base-class features.

inheritance—the transfer of the characteristics of a class to other classes derived from it. For example, if "vegetable" is a class, the classes "legume" and "root" can be derived from it, and each will inherit the properties of the "vegetable" class. The inherited properties of the "vegetable" class might include name, growing season, and water requirements.

interface—contains only the signatures of methods, delegates, or events. The implementation of the methods is created in the class that implements the interface.

Lesson Sequence

Activating prior knowledge/lesson staging (10 minutes)

1. Show the Activator slide in the Microsoft PowerPoint® presentation for this lesson.
    a. Ask students how the following classes might be related: Animal, Fish, Mammal, Cat, Dog, and Reptile.
    b. Have student complete the hierarchy.
    c. Remind students to check that the arrows point to the base class.

Lesson activity (30 minutes)

1. Show the PowerPoint presentation.
    a. Explain the "is-a" relationship of inheritance.
    b. Review the key terms.
    c. Explain the principle of inheritance and show an example of one class being derived from another.
    d. Explain how inheritance is not a reciprocal relationship and how methods can be called from a base class.

e. Explain how to cast a derived class to the base class.

f. Review interfaces and show an example of an interface being defined and implemented in a derived class.

g. Explain and show an example of an abstract class.

Assessment/lesson reflection (10 minutes)

1. Student Activity (SoftDevFund_SA_2.2)

Microsoft resources and Web links

Inheritance (C# Programming Guide):
http://msdn.microsoft.com/en-us/library/ms173149%28VS.80%29.aspx

Suggested best practices:

- You may choose to take some time to ask students to provide examples of programming scenarios in which they would use the concepts in this lesson: inheritance, interfaces, and abstract classes.

Additional notes to the instructor:

- None