REVIEW LESSON

MTA Course: Software Development Fundamentals
Lesson name: Software Development Fundamentals 2.1
Topic: Understand the fundamentals of classes (One 50-minute class period)
File name: SoftDevFund_RL_2.1

Lesson Objective:

2.1: Understand the fundamentals of classes. *This objective may include but is not limited to:* properties, methods, events, and constructors; how to create a class; how to use classes in code.

Preparation Details

Prerequisite student experiences and knowledge

Students should have had experience with the fundamentals of object-oriented programming, and specifically, creating classes. 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 bottom of this review lesson.

Instructor preparation activities

None

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

- SoftDevFund_PPT_2.1
- SoftDevFund_SA_2.1
- SoftDevFund_SA_2.1_key

Teaching Guide

Essential Vocabulary:

class—in object-oriented programming, a generalized category that describes a group of more specific items, called *objects.* A class is a descriptive tool used in a program to define a set of properties or a set of behaviors/methods (actions available to other parts of the program) that characterize any member (object) of the class. Program classes are comparable in concept to the categories that people use to organize information about their world (such as animal, vegetable, and mineral) that define the types of entities they include and the ways those entities behave.

constructor—a method that allows the programmer to set default values, limit instantiation, and write code that is flexible and easy to read.

method—in object-oriented programming, a process or behavior performed by an object when it receives a message.

object—in object-oriented programming, a variable comprising both routines and data that is treated as a discrete entity which is referred to as an instance of the class.

object-oriented programming—a programming paradigm in which a program is viewed as a collection of discrete objects that are self-contained collections of data structures and routines that interact with other objects.

property—members of a class that provide a flexible mechanism to read, write, or compute the values of private fields. Properties are viewed or accessed by "accessor" methods within the class and are changed or modified by "modifier" methods within the class.

Lesson Sequence

Activating prior knowledge/lesson staging (10 minutes)

1. Show the Activator slide in the PowerPoint presentation for this lesson.
    - **a.** Look around the room. What objects do you see that are related to others (desk, chair, table); (pencil, pen); (student, teacher)?
    - **b.** What general name would you give to each of these groups? (Furniture, writing utensils, person.)
    - **c.** Relate this Activator to object-oriented programming. All these objects interact with one another in the context of a classroom. If we were to model a classroom as a computer program, we would have multiple desks and students. We can represent objects with similar characteristics with a class by defining properties and behaviors.

Lesson activity (20 minutes)

1. Show the PowerPoint® presentation.
   a. Define and describe the relationship between objects and classes.
   b. Go through the implementation of a class and pause to concentrate on the key components:
      - Property
      - Method
      - Constructor
   c. Review how to use classes in code by looking at various examples.

Assessment/lesson reflection (20 minutes)

1. Student Activity (SoftDevFund_SA_2.1)

Microsoft resources and Web links

Objects and Classes (C# Programming Guide)
http://msdn.microsoft.com/en-us/library/ms173109%28VS.80%29.aspx

Suggested best practices:

- Make the connection between objects in real life and objects in programming.
- Make sure students understand the difference between an object reference and the object itself.