LESSON 2.5

10754 Microsoft .NET Fundamentals

Understand and Use Different Data Types in the .NET Framework

LESSON 2.5

10754 Microsoft .NET Fundamentals

Lesson Overview

Understand and use different data types in the .NET Framework.

In this lesson, you will:

- Review data types, including value and reference data types.
- Explore collections in the Microsoft® .NET Framework.

Guiding Questions

1. What's the difference between a value data type and a reference data type?
2. What classes does the .NET Framework provide for storing and retrieving collections of data?

Anticipatory Set

What two pieces of information do you need to specify in declaring a variable? In what order must they be specified?

Give a few examples of variable declarations.

Data Types

- Data types are classifications that determine how data is stored and what operations can be performed on the data.
- Data types apply to all values that can be stored in computer memory or participate in the evaluation of an expression.
  - Every variable, literal, constant, enumeration, property, procedure parameter, procedure argument, and procedure return value has a data type.
- A variable is defined by specifying an identifier (i.e., the name of the variable) and its data type.

Data Type Categories

- Data types can be either built-in or user-defined.
  - Programming languages provide built-in support for many commonly used data types. Examples include `int`/`Integer`, `double`/`Double`, `char`/`Char`, and `bool`/`Boolean`.
  - Data types also can refer to user-defined data types, such as classes and interfaces.
- More important, data types are either *value types* or *reference types*.

Value Types vs. Reference Types

- Value types directly store a value.
  - Value types are like a pocket holding coins—the money itself is stored in the pocket.
  - Examples: all numeric types (`byte`, `int`, `double`, `decimal`, etc.) as well as `bool`, `char`, and all structures.
- Reference types do not store a value; they contain a pointer (or "reference") to a memory container where the actual value is stored.
  - Reference types are like a check—it is not actual money but a reference to the bank account where the money is stored.
  - Examples: `String`, all arrays, and all `Class` types (such as `Button` and `TextBox`).

Value Types vs. Reference Types (continued)

- The difference between value types and reference types may seem subtle, but it can have important implications.
- Important consequence: copying variable contents.
  - Assigning one value type variable to another copies the contained value. In our analogy, this makes copies of the actual coins found in the pocket.
  - Assigning one reference type variable to another copies only the reference. If you copy a reference type variable and then change the value of the original variable you may change the value referenced in the new variable.

Arrays

- Generally, use square brackets to specify the index of an individual member of the array.

```
rosterNames[5]
```

  - The number is an integer that starts with 0 for the first entry in the array.
- Limitation:
  - The size of an array cannot be changed after the array is created.

Collections in the .NET Framework

- The .NET Framework provides specialized classes for storing and retrieving "collections" of related or similar data.
  - Example: the roster for a class of students. To store their names, you could make many `String` variables; however, a collection would allow you to keep all the names in one object.
- Most of the collection classes in the .NET Framework implement the same interfaces (from the `System.Collections` namespace).
  - Working with the different variations is usually similar.
  - Example: Many collections implement an `Add` method for adding data to the collection, and a `Remove` method for removing an object.

Lesson Review

Complete the Student Activity.