CHEAT SHEET

This cheat sheet is designed as a way for you to quickly study the key points of this chapter.

Value types

- Store the values directly.
- Are an alias for `System` types such as `int` for `System.Int32`.
- Are passed as a copy to methods.
- Framework includes standard data types most commonly required.
- Legal values are based on number of bits used to store the type.

Data structures

- Data structures involved structs, enumerations, and classes.
- Structs are lightweight data structures.
- Structs can contain member variables and methods.
- Structs are passed by value unlike reference types, which are passed by reference.

Enumerations

- Enumerations contain a list of named constants.
- They make code more readable.
- They use an underlying value for the named constant.
- Underlying values of type `int` start at 0 and increment by one unless otherwise indicated in the declaration.

Reference types

- Reference types are also commonly referred to as *classes*.
- Classes contain member variables to store characteristics.
- Classes contain member functions to provide functionality.
- Class files encompass data and functionality in one package.

Modifiers

- Modifiers are used to determine access for classes and class members.
- See Table 3-3 for a complete list of modifiers.
- Modifiers are listed first in declarations.

Fields

- Fields contain the data for classes.
- Fields are also known as *member variables*.
- They describe characteristics of the class.
- They should be marked private to avoid unwanted modification.

Constructors

- Use to initialize classes.
- Do not include a return type.
- Use the same name as the class.
- May contain no parameters (default constructor).
- If no constructor is defined, compiler generates a default constructor.

Methods

- Provide functionality for a class
- Can be used with modifiers
- Can return values or not (return type void)
- Can accept arguments through parameters in the signature
- Can use optional and named parameters

Overloaded methods

- Same method name with multiple instances for different functionality
- Defined by the signature (name, types, and kinds of parameters)

Abstract methods

- Do not define an implementation
- Can be declared in abstract classes only
- End with a semicolon

Overridden methods

- Hide the implementation of a method of the same name in the base class
- Provide a means to change method behavior in derived class
- Used for virtual and abstract methods in base class

Extension methods

- Can be applied to your own types or even existing types in .NET
- Extend existing classes by adding methods without recompiling

Optional parameters

- Enable you to choose which parameters are required in a method.
- Defined as optional by including a default value.
- The default value is used if none is passed by caller.
- Must exist after required parameters.
- If multiple optional parameters exist and a value is specified for one, all preceding optional parameters must also be supplied values.

Named parameters

- Allow for giving parameters in a method a name
- Increase code readability
- Enable you to pass arguments to a method in an order other than in the method signature

Encapsulation

- Also known as data hiding.
- Involves making member variable private.
- Data exposed through properties.
- Functionality and data are all enclosed as part of the class.
- Creates a "black box" concept.

Properties

- Present the public interface to your class
- Enforce encapsulation
- May be read/write, read-only, or write-only
- Can be used to perform data validation on incoming and outgoing data values

Indexed properties

- Allow array-like access to groups of items
- Must be access using an index in the same manner as arrays

Generic types

- Design classes without specifying the types at definition stage
- Design methods without specifying the types for parameters at definition stage
- Use a placeholder at definition stage that will be replaced by type during instantiation
- Enable type-safe coding
- Increases performance due to reduction in conversions, boxing/unboxing

REVIEW OF KEY TERMS

abstract method Indicates that the thing modified has a missing or incomplete implementation. The abstract modifier can be used with classes, methods, properties, indexers, and events. Use the abstract modifier in a class declaration to indicate that a class is intended to be only a base class of other classes.

accessor methods Methods used to access hidden member variables.

class files File that contain a C# class. Classes encapsulate data and functionality into one unit of code.

classes Coding components that enable you to create custom types that group together characteristics, methods, and events.

constructors Class methods executed when an object of a given type is created.

data structures Components in code that are used to store data within the program.

encapsulation The hiding of details around the implementation of an object so there are no external dependencies on the particular implementation.

enumerations A distinct type consisting of a set of named constants.

event publisher The object in code that will raise the event for the listener or subscriber.

event subscriber The object that listens for an event to be raised

fields Variables that store characteristic data for a class.

heap An area of memory used by the .NET compiler to store reference type variables

instance fields The same as fields but are known as instance fields because they relate to an instance of an object. In other words, their values are not shared among objects of the same class.

memory address An addressable location in computer memory that is used to store and retrieve values stored there.

methods Provide the functionality for a class.

modifiers Modify declarations of types and type members.

overloaded methods Methods with identical names for procedures that operate on different data types.

override To extend or modify the abstract or virtual implementation of an inherited method, property, indexer, or event.

properties Members that provide a flexible mechanism to read, write, or compute the values of private fields.

reference types Class files or other objects represented as references to the actual data (memory addresses).

signature In this case, a method signature. It is the unique identifying components of the method such as return type, name, and parameters.

stack An area of memory used by the .NET compiler to store value types during program execution.

EXAM TIPS AND TRICKS

The Review of Key Terms and the Cheat Sheet for this chapter can be printed off to help you study. You can find these files in the ZIP file for this chapter at `www.wrox.com/remtitle.cgi?isbn=1118612094` on the Download Code tab.