SAMPLE QUESTIONS FOR QUIZ 01 – COMP1098 (C# Programming)

Classes, Objects, Methods, Properties and Instance Variables

1. An object’s property can have which of the following accessors?
a) get
b) set
c) a and b
d) None
Answer: c
2. Which of the following terms refers to the data represented by an object’s instance variables?
a) Behaviors
b) Attributes
c) Methods
d) Information
Answer: b
3. Each object of a class is identical except for the values and references it holds.
Answer: True.
4. Methods always return a value or reference.
Answer: False. A method's return type can be void, indicating that there is no return value.
5. Instance variables and local variables both represent the attributes of a class.
Answer: False. Instance variables specify a class’s attributes. Local variables are used only in the body of the method in which they are declared to store data used by the method to perform its task.

4.3 Declaring a Class with a Method and Instantiating an Object of a Class

1. In UML class diagrams, the + sign
a) signifies that the section can be expanded.
b) is the public visibility symbol.
c) stands for addition.
d) None of the above
Answer: b
2. Method headers contain all of the following except:
a) access modifier
b) left brace
c) method name
d) return type
Answer: b
3. Every C# application is composed of at least one:
a) public method
b) data member
c) public class declaration
d) imported class
Answer: c
4. C# is known as an extensible language because it declares all classes ahead of time.

Answer: False. C# is known as an extensible language because you can declare new class types as needed.
5. The keyword new is used to create a new method of a class.
Answer: False. It's used to create a new *object* of a class.
4.4 Declaring a Method with a Parameter
1. Which of the following method headers does the following method call match?
    Action(“This is an example”, 15, 25.5)

a) Action()
b) Action(int x, double y, string z)
c) Action(string x , double y, int z)
d) Action(string x, int y, double z)
Answer: d
2. How would the following method header be represented in a UML class diagram?
    public void DisplayMessage(string courseName);

a) + DisplayMessage()
b) + DisplayMessage(string)
c) + DisplayMessage(courseName: string)
d) + DisplayMessage(courseName)
Answer: c
3. The comma-separated list of values specified in a method call are called ________.
a) arguments
b) references
c) objects
d) parameters
Answer: a
4. Any classes that are not explicitly placed in a namespace are implicitly placed in the global namespace.
Answer: True.

5. Methods can have only zero parameters or one parameter.
Answer: False, methods can have as many parameters as desired.
4.5 Instance Variables and Properties
1. Why would the access modifier private be used instead of public?
a) To make it more complicated for accessing methods and variables
b) To hide sensitive information
c) To help insure encapsulation
d) b and c
Answer: d
2. How can a private variable be accessed?
a) If the private variable is inside the same class as the currently executing code, then you can access it normally using the variable’s name.
b) Use the property for that variable.

c) Use a method that is in the same class as the private variable, which can access the variable.
d) All of the above
Answer: d
3. Attributes of a class are also known as:
a) constructors
b) local variables
c) fields
d) classes
Answer: c
4. Attributes are represented as variables in a class declaration.
Answer: True.

5. After defining a property, you must call the get and set accessors to manipulate it.
Answer: False. After defining a property, you can use it like a variable in your code; you may use assignment (=) operator.
6. Fields are declared outside a class declaration.
Answer: False. Fields are declared inside a class declaration but outside the bodies of the class's method declarations.
7. Declaring a class's instance variables with access modifier private indicates that only the class's methods should have access to those instance variables.
Answer: True.
8. The order in which methods are declared in a class declaration determines when those methods are called at execution time.
Answer: False. The order in which methods are declared does not determine the order in which they are called.

4.6 UML Class Diagram with a Property
1. In a UML class diagram, the word "property" is in _________.
a) braces []
b) quotes ""
c) guillemets «»
d) curly braces {}
Answer: c
2. What is the private visibility symbol in the UML class diagram?
a) +
b) –
c) *
d) ^
Answer: b
3. A class diagram helps you design a class, so it is required to show *every* implementation detail.
Answer: False. To maintain simplicity when designing the class, it is often best *not* to show every implementation detail.

4. UML represents instance variables and properties as attributes by listing the attribute name

followed by a colon and the attribute type.
Answer: True.

1. Which of the following is a reason for using the get and set accessors?
a) To follow a universal standard.
b) To allow the class to control the manner in which the data is set or returned.
c) To make a program more robust.
d) b and c
e) All of the above
Answer: d

2. Which method converts a string into an int?
a) *string_variable*.ConvertToInt
b) Convert.ToInt32
c) Type.ConvertToInt
d) All of the above
Answer: b

3. It is possible to have only one of the get or set accessors for a variable.
Answer: True.

4. By including only the get accessor, the private variable can be assigned a new value.
Answer: False. Including only the get accessor, makes the private variable read-only outside of the class.

4.8 Auto-implemented Properties
1. An auto-implemented property may have
a) only a get accessor
b) only a set accessor
c) get and set accessors
d) none of the above
Answer: c
2. You can implement validation logic in auto-implemented properties.
Answer: False. You must create user-defined properties if you want to add any logic.
4.9 Value Types vs. Reference Types

1. A variable of a reference type contains:
a) information about the type and its data
b) data of that type
c) the address of the location in memory where data is stored
d) None of the above.
Answer: c
2. Which of the following is a C# built-in *reference* type?
a) int

b) string
c) bool
d) char
Answer: b
3. What is the default value of a reference?
a) 0
b) ""
c) null
d) default
Answer: c
4. Value types normally represent single pieces of data.
Answer: True.
5. Programmers can create reference types.
Answer: True.
6. A primitive-type variable can store exactly one value at a time, whereas one object can contain many individual pieces of data.
Answer: True.
7. Primitive-type variables refer to objects in a C# application.
Answer: False. Reference-type variables refer to objects in a C# application.

4.10 Initializing Objects with Constructors

1. A ________ is called to initialize a new instance of a class.
a) constructor
b) destructor
c) creator
d) new
Answer: a

2. What keyword is associated with creating objects?
a) create
b) constructor
c) new
d) start
Answer: c

3. A default constructor has how many parameters?
a) 0
b) 1
c) 2
d) any number
Answer: a

4. Constructors are always public.
Answer: False. Constructors can have any access modifier; however, public constructors are more

common.

5. Like methods, constructors can have more than one parameter.
Answer: True.

6. The compiler provides a default constructor with a string parameter in any class that does not explicitly include a constructor.
Answer: False. The compiler provides a default constructor with *no* parameters in any class that does not explicitly include a constructor.

7. Constructors can take arguments, but cannot return values.
Answer: True.

4.11 Floating-Point Numbers and Type decimal
1. To type a decimal literal, you must type the letter ____ at the end of a real number.
a) C or c
b) E or e
c) M or m
d) T or t
Answer: c

2. Which of the following list correctly orders which type has fewest to the most significant digits?
a) decimal, double, float
b) float, double, decimal
c) double, float, decimal
d) decimal, float, double
Answer: b

3. Let z represent a format specifier. How is the format specifier included in WriteLine?
a) { 0:z }
b) { z:0 }
c) { 0,z }
d) { z,0 }
Answer: a

4. What is the difference between a float and a double?
a) double variables store integers and float variables store floating-point numbers.
b) double variables store numbers with smaller magnitude and coarser detail.
c) double variables store numbers with larger magnitude and finer detail.
d) None of the above.
Answer: c

5. How many significant digits does a double variable have?
a) 7

b) 8
c) 14
d) 15
Answer: d

6. double represents a double-precision floating-point number, while float represents a single-precision floating-point number.
Answer: True.

7. float**,** double**,** decimal are always 100% precise.
Answer: False. All three types could only hold a limited range of numbers.

Methods:

1. _________ methods can be called without the need for an object of the class to exist.
a) special
b) independent
c) static
d) dependent
Answer: c

2. Methods that call themselves are known as ________ methods.
a) reiterative
b) self-calling
c) repeat-calling
d) recursive
Answer: d

3. Information may be _______ methods.
a) passed to
b) returned from
c) manipulated in
d) All of the above.
Answer: d

4. A method is invoked by a _______.
a) return statement
b) method header
c) method call
d) None of the above.
Answer: c

6. A return type of ____ is specified for a method that does not return a value.

a) nothing
b) null
c) void
d) None of the above.
Answer: c

7. Methods are called by writing the name of the method followed by ______ enclosed in parentheses.
a) a condition
b) arguments
c) a counter
d) None of the above.
Answer: b

8. The technique of dividing large programs into simple problems is known as divide and conquer.
Answer: True.

10. Methods can return at most one value.
Answer: True.

11. The return type of a method's return value is specified in the method call.
Answer: False. The return type for a method is specified in the method header, not in a method call.

13. Methods should be large and perform as many tasks as possible.
Answer: False. A method should perform a single well-defined task.

14. Repeating high-quality, high-performance code in a program helps ensure higher quality programs.
Answer: False. You should avoid repeating code in a program. Package the code as methods and call those methods repeatedly, but do not repeat the code itself.

7.2 Packaging Code in C#

1. Which of the following is *not* a way of packaging code?
a) methods
b) classes
c) variables
d) namespaces
Answer: c

2. Programmers write ______ to define specific tasks that may be used at many points in a program.
a) classes
b) methods
c) modules

d) None of the above.
Answer: b

3. Many prepackaged classes and methods are provided in the .NET FCL, an acronym for the **_______**.
a) Framework Class Library
b) Framework Class Listing
c) Form Class Library
d) None of the above.
Answer: a

5. Prepackaged methods and classes are available for use in the .NET Framework Class Library.
Answer: True.

7. Methods defined by the programmer may consist only of predefined .NET method calls.
Answer: False. Methods may consist of any code that the programmer wishes to execute.

7.3 static **Methods,** static **Variables, and Class** Math

1. Which of the following correctly calls the Math class method Sqrt with a value of 36?
a) Sqrt(36);
b) Math.Sqrt(36);
c) Math.Sqrt = 36;
d) None of the above.
Answer: b

2. Which of the following describes a static variable?
a) a variable with one copy shared by all class objects
b) a variable whose value may not be changed
c) all of the above
d) None of the above.
Answer: a

3. When may an application omit the string[] args parameter from the Main header?
a) when the application does not need to use strings
b) when the application does not take command-line arguments
c) when the application does not output any strings
d) All of the above
Answer: b

4. Which of the following can be an argument to a method?
a) constants
b) variables

c) expressions
d) All of the above
Answer: d

5. Any variable declared with keyword const is a constant and cannot be changed after it's declared.
Answer: True.

6. A static variable represents class-wide information.
Answer: True.

7. A program contains a copy of a static variable for each object that's instantiated.
Answer: False. A program contains only one copy of a static variable, no matter how many objects are instantiated.

8. The Math class provides only methods.
Answer: False. Math also provides data such as values for PI and the base value E for natural logarithms.

9. Main is declared static so the execution environment can invoke Main without creating an instance of the class.
Answer: True.

10. Methods are called by writing the name of the method followed by a left square bracket followed by the argument (or a comma-separated list of arguments) of the method followed by a right square bracket.
Answer: False. Methods are invoked with parentheses, not square brackets.

7.4 Declaring Methods with Multiple Parameters

1. How are various parameters separated in the method header?
a) brackets
b) braces
c) commas
d) periods
Answer: c

3. The parameter list in the method header and the arguments in the method call must agree in:
a) number
b) type
c) order
d) all of the above
Answer: d

4. When an object is concatenated with a string:

a) a compilation error occurs
b) a runtime error occurs
c) the object's ToString method is implicitly called
d) the object's class name is used
Answer: c

5. strings can be concatenated only with the method Concat.
Answer: False. strings can also be concatenated with the operators + or +=.

7. Method arguments may not be expressions.
Answer: False. Method arguments may be variables, constants or expressions.

7.5 Notes on Declaring and Using Methods

1. To call a static method, use the _________ name followed by a period, and the method with its arguments.
a) class's
b) instance variable's
c) namespace's
d) All of the above
Answer: a

2. Which keyword can programmers use to break out of a void method?
a) continue
b) break
c) return
d) next
Answer: c

3. A static method can ________.
a) call only other static methods of the same class directly
b) manipulate only static fields in the same class directly
c) be called using the class name and a dot (.)
d) All of the above
Answer: d

4. Which statement is *false*?
a) If a method does not return a value, the return-value-type in the method declaration can be omitted.
b) Placing a semicolon after the right parenthesis enclosing the parameter list of a method declaration is a syntax error.
c) Re-declaring a method parameter as a local variable in the method's body is a compilation error.
d) Forgetting to return a value from a method that should return a value is a compilation error.

Answer: a

6. To access the class's non-static members, a static method must use a reference to an object of the class.
Answer: True.

7. When a **return** statement executes, control proceeds immediately to the first statement after the method that issues the **return**.
Answer: False. When a **return** statement executes, control returns immediately to the point at which a method was invoked.

7.6 Method Call Stack and Activation Records

1. Stacks are ______________ data structures.
a) FIFO
b) Random
c) LIFO
d) None of the above.
Answer: c

2. What is pushed onto the program execution stack when a method is call?
a) the whole method
b) the class of the method being called
c) the object of method being called
d) the return address of the method being called
Answer: d

3. The local variables used in each invocation of a method during an application's execution are stored in the _________.
a) program execution stack
b) activation record
c) stack frame
d) All of the above.
Answer: d

4. Stack data structures have three fundamental methods.
False. Stacks only have two fundamental methods: push and pop.

5. Since memory in a computer is finite, an stack error may occur known as stack overflow.
Answer: True.

7.7 Argument Promotion and Casting

1. A(n) ________ conversion occurs when a type is converted to a type that can hold more data.
a) implicit
b) widening
c) explicit
d) None of the above.
Answer: b

2. An int must be explicitly cast to which of the following?
a) float
b) double
c) sbyte
d) bool
Answer: c

3. Let x be a double. How can you typecast a double into a float?
a) implicitly
b) x(float)
c) (float)x
d) x = float
Answer: c

4. Conversions that cause data loss are not allowed in C#.
Answer: False. Conversions that may cause data loss are known as narrowing conversions and they are possible in C#.

6.26 Converting a simple type to another simple type may change the value.
Answer: True.

7.8 The Framework Class Library

1. What does the Framework Class Library hold?
a) namespaces
b) classes
c) methods
d) All of the above.
Answer: d

2. Which directive allows you to use the Framework Class Library?
a) import
b) using
c) load

d) namespace
Answer: b

3. By including using System; at the top of a source program, you can use the unqualified class name, Console, instead of the fully qualified class name, System.Console.
Answer: True.

5. It is always better to write your own code than to search and use a class from the FCL.
Answer: False. In general, classes from the FCL are optimally designed and are more robust.

7.9 Case Study: Random-Number Generation

1. A ______ is provided to a random number generator to ensure the same sequence of numbers is *not* generated during each execution.
a) date
b) parameter
c) seed
d) None of the above.
Answer: c

2. Random-number generator scaling is the process of:
a) causing each number in a range to have equal likelihood of being generated
b) modifying the range from which a number will be generated
c) calculating a sequence of numbers to be generated
d) None of the above.
Answer: b

3. Values returned by the Random class method Next are pseudo-random numbers.
Answer: True.

4. The Next method produces a random number from zero up to and including the number provided by the programmer in the method call.
Answer: False. The number produced is from zero up to and not including the number provided by the programmer.

5. A particular seed value always produces the same series of random numbers.
Answer: True.

7. The values produced by **Random** are weighted so that some values will be generated with greater frequency than others.
Answer: False. The integers in the range are produced with approximately equal likelihood.

8. Values produced by random-number generating methods are truly random. There is no way to predict the next result.
Answer: False. Actually, these numbers are pseudorandom numbers—a fixed sequence of values produced by a complex mathematical calculation.

7.10 Case Study: A Game of Chance (Introducing Enumerations)

1. A(n) ________ is a value type that contains a set of constant values.
a) enumeration
b) object
c) set
d) None of the above.
Answer: a

2. Which of the following cannot be an enum's underlying type?
a) sbyte
b) int
c) ulong
d) double
Answer: d

3. If a change is made to an underlying value in an enumeration, you must change the program in every place that the old value was used.
Answer: False. If a widely used value changes, the modification only has to be made once in the enumeration.

4. To compare a simple-type value to the underlying value of an enumeration constant, you must use a cast operator to make the two types match.
Answer: True.

7.11 Scope of Declarations

1. Identifiers declared within a class have _________.
a) block scope
b) class scope
c) local scope
d) None of the above.
Answer: b

2. If a local variable in a method has the same name as a variable in the main program, what will occur?
a) an error is generated
b) the variable in the main program is "hidden" until the method is finished executing

c) the variable in the main program will override the variable from the method
d) None of the above.
Answer: b

3. Which of the following statements describes block scope?
a) It begins at the opening **{** of the class declaration and terminates at the closing **}**
b) It limits label scope to only the method in which it is declared.
c) It begins at the identifier's declaration and ends at the terminating right brace (**}**).
d) It is valid for one statement only.
Answer: c

4. Which of these statements best defines scope?
a) Scope refers to the classes that have access to a variable.
b) Scope determines whether a variable's value can be altered.
c) Scoping allows the programmer to use a class without using its fully qualified name.
d) Scope is the portion of a program that can refer to an entity by its simple name.
Answer: d

4. Any variables declared in a for statement header have block scope within that statement.
Answer: True.

5. Methods of a class can access all members defined in that class.
Answer: True.

6. The declaration space of a class begins at the opening left brace and terminates at the closing right brace.
Answer: True.

7. A local variable or reference declared in a block can be used in blocks in which that block is nested.
Answer: False. A local variable or reference declared in a block can be used only in that block or in blocks nested *within* that block.

7.12 Method Overloading

1. Which of the following will violate the rules of overloading methods?
a) Methods with the same signatures but different return types.
b) Methods with different signatures but the same return type.
c) Methods with different number of arguments.
d) Method with different types of arguments.
Answer: a

3. Overloaded methods always have the same _________.
a) method name

b) return type
c) number of the parameters
d) order of the parameters
Answer: a

4. An overloaded method is one that
a) has a different name than another method, but the same parameters.
b) has the same name as another method, but different parameters.
c) has the same name and parameters as a method defined in another class.
d) has the same name and parameters, but a different return type than another method.
Answer: b

5. Which of the following methods are overloaded?

A. public int max (int a, int b) { … }
B. public double max (double a, double b) { … }
C. public int max (int a, int b, int c) { … }
D. public double max (double a, double b, double c) {…}

a) A and B are overloaded; C and D are overloaded
b) A and C are overloaded; B and D are overloaded
c) A, B and C are overloaded
d) All these four methods are overloaded
Answer: d

6. Suppose method1 is declared as
void method1 (int a, float b)
Which of the following methods overloads method1?
a) void method2 (int a, float b)
b) void method2 (float a, int b)
c) void method1 (float a, int b)
d) void method1 (int b, float a)
Answer: c

7. A C# class can have which of the following methods?

A.**foo(int a)**
B.**foo(int a, int b)**
C.**foo(double a)**
D.**foo(double a, double b)**
E.**foo(int b)**

a.All of the above.
b.A, B, D, E.
c.A, B, C, D.
d.A, C, D, E.

Answer: c

8. The order of a method's parameters is insignificant.
Answer: False. The order of a method's parameters helps determine the signature of the method.

9. Methods of the same name can be declared in the same class, as long as they have different sets of parameters (determined by the number, types and order of the parameters).
Answer: True.

10. Overloaded methods normally perform similar tasks, but on different types of data.
Answer: True.

7.13 Optional Parameters

1. Any parameter in a parameter list can be declared as an optional parameter.
Answer: False. The optional parameters must be at the end of the parameter list. All required parameters must appear first. Once an optional parameter is specified, all other parameters to its right in the parameter list must also be optional parameters.

2. Optional parameters enable you to specify default argument values for specific parameters.
Answer: True.

7.14 Named Parameters

1. When calling a method, you must specify the arguments in the same order that the corresponding parameters are declared in the method's parameter list.
Answer: False. Using named parameters, you can specify the argument values in any order as long as each argument is preceded by the corresponding parameter's name and colon (:). This can also be used to skip optional parameters in the parameter list.

7.15 Recursion

1. A recursive method is a method that:
a) calls another method
b) calls itself
c) has no return type
d) None of the above
Answer: b

2. Which of the following is *not* part of recursion?
a) recursive call
b) recursion step(s)
c) base case(s)
d) None of the above.
Answer: d

3. A recursion step normally includes the keyword return.
Answer: True.

4. A recursive method only knows directly how to solve the base case.
Answer: True.

5. Recursion takes a complex problem and splits it up among many methods.
Answer: False. Recursion involves one method that executes many times to solve a complex problem piece by piece.

6. Improperly defining a recursive method may result in an infinite recursion.
Answer: True.

7.16 Passing Arguments: Pass-by-Value vs. Pass-by-Reference

1. Passing an argument to a method by-value provides the method with:
a) the address of the value in memory
b) a separate copy of the value
c) the type of the value
d) None of the above.
Answer: b

2. The ________ keyword is used to pass value-type variables to methods by-reference.
a) ref
b) reference
c) RefPass
d) None of the above.
Answer: a

3. Pass-by-value is used when optimal program performance is necessary.
Answer: False. Pass-by-value reduces performance by creating unnecessary copies of data.

4. A method's changes to a variable that is passed-by-value do not affect the value of the original variable.
Answer: True.

6. A variable that has not been initialized cannot be passed to a method.
Answer: False. The out keyword allows a variable to be passed that has yet to be initialized in the program.

7. A method cannot change the actual reference of an identifier that has been passed-by-reference.
Answer: True.

Classes and Objects: A Deeper Look

10.1 Introduction

1. Object orientation uses classes to:
a) develop algorithms
b) encapsulate data and methods
c) organize large amounts of data
d) None of the above.
Answer: b

2. An instance of a user-defined type is called a(n) ________.
a) class
b) interface
c) object
d) None of the above.
Answer: c

3. Objects can hide their implementation from other objects.
Answer: True.

6. The data components of a class are called instance variables.
Answer: True.

5. Instances of a class are called class variables.
Answer: False. Instances of a class are called objects.

10.2 Time **Class Case Study**

1. The __________ of a class are also called the public services or the public interface of the class.
a) public constructors
b) public instance variables
c) public methods
d) All of the above
Answer: c

2. Instance variables or methods declared with the ________ modifier are accessible only in that class definition.
a) protected
b) static
c) private
d) None of the above.
Answer: c

3. A class's _______ initializes members of that class.
a) constructor
b) utility method
c) access modifier
d) None of the above.
Answer: a

4. Which statement is *false*?
a) The actual data representation used within the class is of no concern to the class's clients.
b) Clients generally care about *what* the class does but not *how* the class does it.
c) Clients are usually involved in a class's implementation.
d) Hiding the implementation reduces the possibility that other program parts will become dependent on class-implementation details.
Answer: c

5. Every class inherits directly or indirectly from class _________.
a) **Inheritor**
b) **Base**
c) **Super**
d) **object**
Answer:d

6. Programmers should *not* take into consideration that their code will be modified.
Answer: False. Programmers should always anticipate that their code will be modified.

7. Variables declared within class methods are called instance variables.
Answer: False. Instance variables are variables declared within a class definition, but not a method definition.

8. Member access modifiers are used to limit how much memory is available for class members.
Answer: False. Member access modifiers are used to control the accessibility of class members between separate classes.

9. The purpose of utility methods is to support the operation of a class' other methods.
Answer: True.

10. A constructor should return either **true** or **false** to determine if the object was initialized properly.
Answer: False. A constructor may *not* have a return value, attempting to do so will result in a syntax error.

10.3 Controlling Access to Members

1. Which of the following should usually be **private**?
a) methods

b) constructors
c) variables
d) All of the above
Answer: c

2. Which of the following statements is *true*?
a) Methods and instance variables can both be either **public** or **private**.
b) Information hiding is achieved by restricting access to class members via keyword **public**.
c) The **public** members of a class are not directly accessible to the client of a class.
d) None of the above is true.
Answer: a

3. The **public** methods of a class present the services that a class provides.
Answer: True.

4. Variables or methods declared with access modifier **public** are accessible wherever the program has a reference to an object of the class.
Answer: True.

5. An attempt by a method which is *not* a member of a particular class to access a **private** member of that class is a fatal runtime error.
Answer: False. Attempts to access another class's **private** members result in compilation errors.

10.4 Referring to the Current Object's Members with the this Reference

1. An object's **this** reference refers to:
a) the object itself
b) what the programmer specified
c) the entry point of the program
d) None of the above.
Answer: a

2. In a method in which a parameter has the same name as an instance variable, using the **this** reference allows you to refer to ________.
a) the parameter
b) the instance variable
c) varies depending on the situation
d) None of the above
Answer: b

3. When should a program explicitly use the this reference?
a) accessing a private variable
b) accessing a public variable

c) accessing a local variable
d) accessing a field that is shadowed by a local variable
Answer: d

4. Having a **this** reference allows:
a) A method to refer explicitly to the instance variables and other methods of the object on which the method was called.
b) A method to refer implicitly to the instance variables and other methods of the object on which the method was called.
c) An object to reference itself.
d) All of the above.
Answer: d

5. Only certain objects have a **this** reference.
Answer: False. Every object has a reference to itself.

8. **this** is often used to distinguish between a local variable and class variable that share an identical name.
Answer: True.

10.5 Time **Class Case Study: Overloaded Constructors**

1. A constructor *cannot*:
a) be overloaded.
b) initialize variables to their defaults.
c) specify return types or return values.
d) have the same name as the class.
Answer: c

2. Constructors:
a) initialize instance variables
b) when overloaded, can have identical argument lists
c) when overloaded, are selected by number and types of parameters
d) a and c
Answer: d

3. A constructor that is invoked with *no* arguments is called a ________.
a) zero-argument constructor
b) parameterless constructor
c) default constructor
d) main constructor
Answer: b

4. C# invokes the appropriate constructor by matching the number, types and order of the parameters in the constructor call to those in each constructor definition.
Answer: True.

5. An object of a class that has a reference to another object of the same class may access all the **private** data and methods of that class.
Answer: True.

7. A constructor may *not* have a return value.
Answer: True.

8. A constructor may *not* call other methods.
Answer: False. A constructor can call other class methods.

9. Attempting to overload a constructor with another constructor that has the exact same signature (name and parameters) is a compilation error.
Answer: True.

10. Only the first constructor for a class is defined without a return value. Subsequent constructors have the return type **void**.
Answer: False. Constructors can not have return types.

10.6 Default and Parameterless Constructors

1. Which statement is *false*?
a) The compiler always creates a default constructor for a class.
b) If a class has constructors, but none of the **public** constructors are parameterless, and a program attempts to call a parameterless constructor to initialize an object of the class, a compilation error occurs.
c) A constructor can be called with no arguments only if the class does not have any constructors or if the class has a **public** parameterless constructor.
d) Parameterless constructors do not have any arguments.
Answer: a

2. How many parameters does the default constructor that C# creates for you have?
a) 3
b) 1
c) 0
d) varies
Answer: c

3. The compiler will create a default constructor for a class even if you already declared a constructor(s).

Answer: False. The compiler will create a default constructor only if your class does *not* declare any constructors of its own.

4. Every class must have at east one constructor.
Answer: True.

10.7 Composition

1. The use of references to objects of preexisting classes as members of new objects is called ________.
a) inheritance
b) composition
c) polymorphism
d) None of the above.
Answer: b

2. Composition:
a) Is a form of software reuse.
b) Is using an object reference as a class member.
c) Is a good design practice.
d) All of the above.
Answer: d

3. Composition is sometimes referred to as a(n) ________.
a. is-a relationship
b. has-a relationship
c. have-a relationship
d. one-to-many relationship
Answer: b

4. A class cannot have references to objects of other classes as members.
Answer: False. A class can have references to objects of other classes as members. Such a capability is called composition.

5. Composition is a form of software reuse.
Answer: True.

6. Classes may be composed of only one other class.
Answer: False. Classes may be composed of as many classes as you desire.

7. User-defined classes may be composed of other user-defined classes.
Answer: True.

10.8 Garbage Collection and Destructors

1. Failure to release resources when the program is no longer using them can cause _________.
a) buffer overflows
b) resource leaks
c) exceptions
d) None of the above.
Answer: b

2. A ________ can be defined in a class to perform termination housekeeping on an object before the garbage collector reclaims the object's memory.
a) destructor
b) property
c) garbage collector
d) None of the above.
Answer: a

3. C# is like C++ in that programmers must manage memory explicitly.
Answer: False. C# performs memory management internally.

4. A destructor has the same name as the class and its constructor.
Answer: False. A destructor is the name of the class preceded by the ~ character.

5. You cannot determine when the garbage collector will execute.
Answer: True.

6. The garbage collector looks for objects with no values.
Answer: False. The garbage collector looks for objects with no "inbound" references.

10.9 static Class Members

1. Which of the following describes a **static** variable?
a) a variable with one copy shared by all class objects
b) a variable whose value may not be changed
c) all of the above
d) None of the above.
Answer: a

2. Which of the following is *false*?
a) A **static** method or property must be used to access **private static** instance variables.
b) A **static** method has no **this** reference.
c) A **static** method can be accessed even when no objects of its class have been instantiated.

d) A **static** method can call instance methods directly.
Answer: d

3. Each object of a class has its own copy of all the instance variables of the class.
Answer: True.

4. A **static** variable represents class-wide information.
Answer: True.

5. A program contains a copy of a **static** variable for each object that's instantiated.
Answer: False. A program contains only one copy of a **static** variable, no matter how many objects are instantiated.

6. Variables that are **static** have class scope.
Answer: True.

7. A **static** method may access **static** and non-**static** members.
Answer: False, **static** methods may access only **static** members.

8. Each instance of a class has a separate copy of that class's **static** variables.
Answer: False. All objects of a class share the same copy of the class's **static** variables.

10.10 readonly Instance Variables

1. Keyword _______ is used for constants whose values cannot be determined at compile time.
a) **const**
b) **readonly**
c) **static**
d) None of the above.
Answer: b

2. Instance variables declared **readonly** do not or cannot:
a) use the principle of least privilege
b) be initialized
c) be modified once the object is constructed
d) cause syntax errors if used as a left-hand value
Answer: c

3. Data members declared as **readonly** must be initialized in their declaration.
Answer: False, **readonly** members may be initialized in their class's constructor.

4. Attempting to modify a **readonly** instance variable after it's initialized is a fatal execution-time error.

Answer: False. Attempting to modify a **readonly** instance variable after it's initialized is a compilation error.

10.11 Data Abstraction and Encapsulation

1. Abstract Data Types:
a) elevate the importance of data
b) are only approximations or models of real-world concepts and behaviors
c) capture two notions, data representation and operations
d) All of the above
Answer: d.

2. The term *information hiding* refers to:
a) **public** methods
b) hiding implementation details from clients of a class
c) accessing static class members
d) the process of releasing an object for garbage collection
Answer: b

3. Stacks are commonly referred to as _______ data structures.
a) first-in, last-out
b) first-in, first-out
c) last-in, first-out
d) None of the above.
Answer: c

4. A _______ data structure can be described as a "waiting line."
a) stack
b) queue
c) list
d) None of the above.
Answer: b

6. The concept of data abstraction refers to the fact that a client does not know how a class's functionality is *implemented*, just how to *use* it.
Answer: True.

7. Clients are normally allowed to manipulate queue structures directly.
Answer: False. Clients are allowed to perform only certain operations on the data representation, guaranteeing the integrity of the structure.

8. C# is extensible because altering the base language is as simple as it's easy to extend via new types.
Answer: False. Although C# is easy to extend via new types, you cannot alter the base language itself.

9. A client of a stack class need not know how the stack is implemented. The client simply requires that when data items are placed in the stack, they will be recalled in last-in, first- out order. This concept is referred to as data abstraction, and C# classes define abstract data types (ADTs).
Answer: True.

10. Conceptually, a queue can become infinitely long. A real queue, of course, is finite. Items are returned from a queue in last-in, first-out (LIFO) order. The last item inserted in the queue is the first item removed from the queue.
Answer: False. Items are returned from a queue in first-in, first-out (FIFO) order--the first item inserted in the queue is the *first* item removed from the queue.

10.12 Class View and the Object Browser

1. ________ displays the variables and methods for all classes in a project.
a) **Class View**
b) **Object Browser**
c) **Design View**
d) None of the above.
Answer: a

2. The **Object Browser** ______________.
a) lists all the classes in the C# library
b) describes the functionality provided by a specific class
c) is separated into 3 frames
d) All of the above.
Answer: d

3. The classes, variables and methods in a program are displayed in a hierarchical structure in **Class View**.
Answer: True.

4. Developers use the **Object Browser** to learn about the functionality provided by a specific class.
Answer: True.

10.13 Object Initializers

1. Object initializers (which initialize an object and its properties) use what syntax?
a) **new** *className* **(** *property1 -> value1, …* **);**
b) **new** *className* **(** *property1 = value1, …* **);**
c) **new** *className* **{** *property1 = value1, …* **};**
d) *className* **{** *property1 = value1, …* **};**
Answer: c

--True/False Statements--

2. An object initializer calls the default constructor before executing.
Answer: True.

3. An object initializer list cannot be empty.
Answer: True.

4. You cannot validate values assigned to properties using an object initializer.
Answer: False. Any validation logic defined in the set accessors will execute when the value is assigned in the initializer list.

Object-Oriented Programming: Inheritance

11.1 Introduction

1. A class inherited from two or more levels up in the hierarchy is known as a ________.
a) indirect base class
b) direct base class
c) superclass
d) None of the above
Answer: a

2. Inheritance is represented by a(n) ________ relationship.
a) "uses"
b) "is-a"
c) "has-a"
d) None of the above.
Answer: b

3. A derived class *cannot* access the _______ members of its base class.
a) private
b) static
c) protected
d) None of the above.
Answer: a

4. Which of the following statements is *false*?
a. A derived class is generally larger than its base class.
b. A base class object is a derived class object.
c. The class following the "**:**"in a class declaration is the direct base class of the class being declared.
d. C# does not support multiple inheritance.
Answer: b

5. A derived class can effect state changes in base class **private** members only through **public**, **protected, internal** methods provided in the base class and inherited into the derived class.
Answer: True

6. Inheritance is the process of building a class with object references of other classes.
Answer: False. Inheritance is the process of taking a class and extending its behaviors and capabilities. Building a class from preexisting object references is known as composition.

7. A class that inherits from another class is referred to as the derived class.
Answer: True.

8. Every object of a base class is an object of that class's derived classes.
Answer: False. Derived classes are more specific than their base classes, therefore an object of a base class cannot also be an object of derived classes.

9. A derived class is often larger and more general than its base class.
Answer: False. A derived class is often larger than its base class, meaning it represents a more specialized group of objects.

10. Multiple inheritance, widely used in C#, is the process of inheriting from more than one class.
Answer: False. Multiple inheritance is the process of inheriting from more than one class, however it is not supported in C#.

11. The process of abstraction allows you to focus on the commonalities among objects in the system.
Answer: True.

11.2 Base Classes and Derived Classes

1. Which of the following pairs demonstrates the "is-a" relationship?
a) car, engine
b) book, table of contents
c) baseball, sport
d) None of the above
Answer: c

2. Which of the following pairs demonstrates the "has-a" relationship?
a) car, vehicle
b) house, window
c) teacher, person
d) None of the above
Answer: b

3. Identify which of the following examples could be considered a base class for the Computer class?
a) machine
b) hard-drive
c) software
d) keyboard
Answer: a

4. Which of the following is *not* a base/derived class relationship?
a) Ford/Taurus
b) University/Boston University
c) Sailboat/Tugboat
d) Country/USA
Answer: c

5. An advantage of inheritance is that:
a) all methods can be inherited
b) all instance variables can be uniformly accessed by base classes and derived classes
c) Objects of a derived class can be treated like objects of their base class
d) None of the above.
Answer: c

6. A base class may have only one derived class.
Answer: False. A base class may have many derived classes.

7. Members of a base class that are private are not inherited by derived classes.
Answer: False. The private members of a base class *are* inherited by derived classes; however, derived classes must access these members through *non*-private members provided in the base class.

8. Constructors are *not* inherited.
Answer: True.

9. All classes in C# have object as either a direct or indirect base class.
Answer: True.

10. An object of one class cannot be an object of another class.
Answer: False. An object can have an “is a” relationship with its base class.

11. Derived classes provide the functionality and features inherited by base classes.
Answer: False. Base classes provide the functionality and features inherited by derived classes.

12. A base class's constructors are inherited into its derived classes.
Answer: False. Constructors are *not* inherited—they're specific to the class in which they're declared.

11.3 protected **Members**

1. When a derived-class member overrides a base-class member, the base-class member can be accessed from the derived-class by using the keyword __________.
a) base
b) top
c) super
d) None of the above
Answer: a

2. Base class methods with this level of access cannot be called from derived classes.
a) private
b) public
c) protected
d) package
Answer: a

3. The protected members of a class may be accessed in their base class or any classes derived from that base class.
Answer: True.

11.4 Relationships between Base Classes and Derived Classes

1. A method must be declared __________ for it to be overridden by derived classes.
a) overrides
b) overridable
c) virtual
d) None of the above
Answer: c

2. Which of the following classes is the root of the class hierarchy?
a) System.object
b) Point
c) ToString
d) None of the above
Answer: a

3. Every class in C#, except __________, extends an existing class.
a) Integer
b) object
c) String
d) Class
Answer: b

4. Overriding a method *differs* from overloading a method because:
a) For an overloaded constructor, the base class constructor will always be called first.
b) For an overridden constructor, the base class constructor will always be called first.
c) Overloaded methods have the same signature.
d) Overridden methods have the same signature.
Answer: d

5. To avoid duplicating code (and possibly errors), use ________, rather than ________.
a) inheritance, the “copy-and-paste” approach.
b) the “copy-and-past” approach, inheritance.
c) a class that explicitly extends object, a class that does not extend object.
d. a class that does not extend object, a class that explicitly extends object.
Answer: a

6. Consider the classes below, declared in the same file:

```
class A
{
  int a;
  public A()
  {
    a = 7;
  }
}

class B : A
{
  int b;
  public B()
  {
        b = 8;
  }
}
```

Which of the statements below is not true?
a) Both variables **a** and **b** are instance variables.
b) After the constructor for **class B** is executed, the variable **a** will have the value **7.**
c) After the constructor for **class B** is executed, the variable **b** will have the value **8**.
d) A reference to **class A** can be treated as a reference to **class B**.
Answer: d

7. Which base class members are inherited by all derived classes of that base class?
a) private instance variables and methods
b) protected instance variables and methods
c) private constructors

d) protected constructors
Answer: b

8. Which statement is *true* when a base class has protected instance variables?
a) A derived class object can assign an invalid value to the base class's instance variables, thus leaving the object in an inconsistent state.
b) Derived class methods are more likely to be written so that they depend on the base class's data implementation.
c) We may need to modify all the derived classes of the base class if the base class implementation changes.
d) All of the above.
Answer: d

9. private fields of a base class can be accessed in a derived class
a) by calling private methods declared in the base class
b) by calling public or protected methods declared in the base class
c) directly
d) All of the above
Answer: b

10. The first task of any derived-class constructor is to call its base-class constructor.
Answer: True.

11. A "copy-and-paste" approach is a simple and efficient way of providing functionality that exists in other classes.
Answer: False. A "copy-and-paste" approach is time consuming, error-prone and results in many copies of the code existing throughout a system, creating a code maintenance nightmare.

12. Using protected instance variables can cause derived-class methods to be written to depend on base-class implementation.
Answer: True.

13. The base reference may be "chained" to traverse further up in the class hierarchy.
Answer: False. "Chaining" base references is a syntax error.

11.5 Constructors in Derived Classes

1. When a derived class constructor calls its base class constructor, what happens if the base class's constructor does not assign a value to an instance variable?
a) a syntax error occurs
b) a compile-time error occurs
c) a run-time error occurs
d) the program compiles and runs correctly because the instance variables are initialized to their default values

Answer: d

2. How can a derived class call a base class constructor?
a) implicitly
b) explicitly
c) a and b
d) the derived class cannot call the base class constructor
Answer: c

3. When a program creates a derived-class object, the object constructor is the last constructor called and the first whose body finishes executing.
Answer: True.

4. If a base class constructor is overridden, the original constructor can no longer be called explicitly by the derived class.
Answer: False. An explicit call to the base class constructor can be included in a derived class constructor via the base reference.

5. When a base class method is overridden in a derived class, it is common to have the derived class version call the base class version and do some additional work.
Answer: True.

11.6 Software Engineering with Inheritance

1. Which of the following statements are *true*?
   A. We can use inheritance to customize existing software.
   B. A base class specifies commonality.
   C. A base class can be modified without modifying derived classes
   D. A derived class can be modified without modifying its derived class.

a) All of the above
b) None of the above
c) A, B and C
d) A, B and D
Answer: a

2. Inheritance preserves the integrity of a base class.
Answer: True.

3. A key to improving the software development process is encouraging software reuse.
Answer: True.

4. To enhance performance and reduce errors, it's a good idea to make derived classes larger than they need to be.

Answer: False. It is best to make classes contain only what they need to and to inherit from the class "closest" to what you need to prevent resources from being wasted.

5. A base class is designed by factoring out similarities among a set of classes.
Answer: True.

6. When creating derived classes, you must use discretion in choosing the proper base class. Ideally, the base class will not contain superfluous capabilities or information.
Answer: True.

11.7 Class object

1. The default implementation of method ToString of object returns a string representing ________.
a) the object's type
b) the object class name
c) namespace_name.object_class_name
d) None of the above
Answer: c

2. The default Equals implementation determines:
a) whether two references refer to the same object in memory.
b) whether two references have the same type.
c) whether two objects have the same instance variables.
d) whether two objects have the same instance variable values.
Answer: a

3. Method **Equals** will accurately compare *any* two objects.
Answer: False. When objects of a particular class must be compared for equality, the class should override method **Equals** to define the proper way to compare the contents of the two objects.

4. Every object in C# has at least seven methods: Equals, Finalize, GetHashCode, GetType, MemberwiseClone, ReferenceEquals, and ToString.
Answer: True.

Object-Oriented Programming: Polymorphism

12.1 Introduction

1. Polymorphism enables you to:
a) program in the general.
b) program in the specific.
c) absorb attributes and behavior from previous classes.
d) hide information from the user.
Answer: a

2.Which of the following is *false* about interfaces?
a) An interface describes a set of methods that can be called on an object, providing a default implementation for the methods.
b) An interface describes a set of methods that can be called on an object, without providing concrete implementation for the methods.
c) Interfaces are useful when attempting to assign common functionality to possibly unrelated classes.
d) Once a class implements an interface, all objects of that class have an is-a relationship with the interface type.
Answer: a

3. Polymorphism specifically enables the creation of programs that handle:
a) classes that are containers for other classes
b) large amounts of data with efficiency
c) a wide variety of classes in a general manner
d) None of the above
Answer: c

4. Polymorphism allows classes to be added with little or no modifications to the generic portion of a program.
Answer: True.

5. When used correctly, polymorphism will never require changes to be made to any part of the program.
Answer: False. Parts of a program that need modification are those that require direct knowledge of the particular class that's added to the hierarchy.

6. Unfortunately, polymorphic programs make it difficult to add new capabilities to a system.
Answer: False. Polymorphic programs make it easy to add new capabilities to a system.

7. Polymorphism enables objects of different classes that are related by a class hierarchy to be processed generically.
Answer: True.

8. The major drawback to polymorphically designed programs is that they do not take into account the future addition or deletion of classes.

Answer: False. Polymorphically designed programs are extremely extensible and easily handle new classes that extend the same hierarchy of classes processed by the program.

12.2 Polymorphism Examples

1.For which of the following would polymorphism *not* provide a clean solution?
a) A billing program where there's a variety of clients who are billed with different fee structures.
b) A maintenance log program where a variety of machine data is collected and maintenance schedules are produced for each machine based on the data collected.
c) A program to compute a 5% savings account interest for a variety of clients.
d) An IRS program that maintains information on a variety of taxpayers and determines who to audit based on criteria for classes of taxpayers.
Answer: c

2. Polymorphism allows for specifics to be dealt with during:
a. execution
b. compilation
c. programming
d. debugging
Answer: a

3. Polymorphism involves:
a) the same message sent to a variety of objects
b) a specific message sent to each specific object
c) different messages sent to one object
d) None of the above.
Answer: a

4. Polymorphism allows the addition of classes providing they were at least considered during program development.
Answer: False. Polymorphism can be used to include classes that were not even envisioned during program development.

5. When a request is made to use a derived-class-object method through a base-class reference, C# polymorphically chooses the correct overridden method in the appropriate derived class that is associated with the object.
Answer: True.

6. Polymorphism allows a programmer to command a wide variety of objects even if the programmer does not know the objects' types.
Answer: True.

7. The use of polymorphism helps promote software extensibility.
Answer: True.

12.3 Demonstrating Polymorphic Behavior

1. Which statement *best* describes the relationship between base class and derived class types?
a. A derived class reference *cannot* be assigned to a base class variable and a base class reference *cannot* be assigned to a derived class variable.
b. A derived class reference *can* be assigned to a base class variable and a base class reference *can* be assigned to a derived class variable.
c. A base class reference *can* be assigned to a derived class variable, but a derived class reference *cannot* be assigned to a base class variable.
d. A derived class reference *can* be assigned to a base class variable, but a base class reference *cannot* be assigned to a derived class variable.
Answer: d

2. If an application needs to perform a derived-class-specific operation on a derived class object reference by a base class variable, the application must first cast the base class reference to a derived class reference through a technique known as _________.
a) downcasting
b) upcasting
c) decreasecasting
d) increasecasting
Answer: a

3. Casting base class references to derived class references is known as downcasting.
Answer: True

4. If a derived class reference is assigned to a base class variable, the variable must be cast back to the derived class before any derived class methods can be called with it.
Answer: True

12.4 Abstract Classes and Methods

1. Abstract classes may *not* be:
a) inherited
b) accessed by derived-classes
c) instantiated
d) None of the above.
Answer: c

2. The abstract methods and properties do *not* provide an implementation.
Answer: True.

3. Concrete classes provide implementations for *every* method and property they define.
Answer: True.

4. The programmer may define implementations for abstract methods to act as a default.
Answer: False. Attempting to provide an implementation for an abstract method is a syntax error.

5. All methods in an abstract class are inherently abstract.
Answer: False. An abstract class may contain instance data and non-abstract methods.

6. An abstract base classes can be used to declare references.
Answer: True.

7. The abstract keyword has the same effect as the virtual keyword.
Answer: False. An abstract method has no implementation and must be overridden while a virtual method provides an implementation and the option of being overridden.

8. An abstract class may be derived from another abstract class.
Answer: True.

9. Objects of **abstract** base classes can be instantiated.
Answer: False. Only objects of concrete classes can be instantiated.

12.5 Case Study: Payroll System Using Polymorphism

1. If a method needs to be called polymorphically, what type of reference should be used to point to the object that the method is being called with?
a) a reference of the base class that defines the behavior of the object
b) a reference of the same type as the object
c) an object reference to the actual object
d) None of the above.
Answer: a

2.The convention of the UML is to denote the name of an abstract class in:
a) bold
b) italics
c) a diamond
d) there is no convention of the UML to denote abstract classes—they are listed just as any other class
Answer: b

3. If the base class contains only **abstract** method declarations, the base class is used for:
a) implementation inheritance.
b) interface inheritance.
c) Both.
d) Neither.
Answer: b

4. Which declaration declares **abstract** method **method1** in **abstract** class **Class1** (**method1** returns an int and takes no arguments)?
a) **public int method1();**
b) **public int abstract method1();**
c) **public abstract int method1();**
d) **public int nonfinal method1();**
Answer: c

5.Which of the following statements about **abstract** base classes is *true*?
a) **abstract** base classes *may* contain data.
b) **abstract** base classes *may not* contain implementations of methods.
c) **abstract** base classes *must* declare all methods as **abstract**.
d) **abstract** base classes must declare all data members not given values as **abstract**.
Answer: a

6. Consider the **abstract** class below:

```
public abstract class Foo
{
   private int a;
   public int b;

   public Foo( int aVal, int bVal )
   {
      a = aVal;
      b = bVal;
   } // end Foo constructor

  public abstract int calculate();
} // end class Foo
```

Any concrete subclass that extends class **Foo**:
a) Must implement a method called **calculate**.
b) Will not be able to access the instance variable **a**.
c) Will not be able to instantiate an object of class **Foo**.
d) All of the above.
Answer: d

7. Every object in C# knows its own class and can access this information through method ______.
a) **GetType()**
b) **GetInformation()**
c) **ObjectClass()**
d) **ObjectInformation()**
Answer: a

8. Assigning a derived class reference to a base class variable is safe:
a. because the derived class *has an* object of its base class.
b. because the derived class *is an* object of its base class.
c. only when the base class is **abstract**.
d. only when the base class is concrete.
Answer: b

9. An **abstract** class cannot have instance data and non-**abstract** methods.
Answer: False. An **abstract** class can have instance data and non-**abstract** methods.

10. Attempting to instantiate an object of an **abstract** class is a logic error.
Answer: False. This is a compilation error.

11. Operator is returns true if two matching types are being compared.
Answer: True.

12.6 sealed Methods and Classes

1. The keyword sealed is applied to methods and classes to:
a) prevent overriding and inheritance
b) guarantee an implementation exists
c) specify a class is concrete
d) None of the above.
Answer: a

2. ________ code is the process by which the compiler replaces method calls with the code of a method to improve performance.
a) Debugging
b) Inlining
c) Compiling
d) None of the above.
Answer: b

3. Classes and methods are declared sealed for all of the following reasons, *except*:
a) sealed methods allow inlining the code.
b) sealed methods and classes prevent further inheritance.
c) sealed methods are **static**.
d) sealed methods can improve performance.
Answer: c

4. All of the following methods are implicitly sealed except:
a) a method in an **abstract** class.
b) a **private** method.
c) a method declared in a sealed class.
d) **static** method.
Answer: a

5. Declaring a method sealed me***Answer:***
a) it will prepare the object for garbage collection
b) it cannot be accessed from outside its class
c) it cannot be overloaded
d) it cannot be overridden
Answer: d

6. All static and private methods are implicitly sealed.
Answer: True.

7. Sealing methods allows the compiler to optimize the program by "inlining code."
Answer: True.

8. All methods in a sealed class must be explicitly declared sealed.
Answer: False. All methods in a sealed class are implicitly sealed.

12.7 Case Study: Creating and Using Interfaces

1. A(n) ________ is best used for providing services that bring together disparate objects.
a) abstract class
b) concrete class
c) interface
d) None of the above.
Answer: c

2. The conventional name for an interface to be called Car is:
a) InterfaceCar
b) ICar
c) CarI
d) None of the above.
Answer: b

3. The purpose of an interface is to:
a) provide similar objects with the same functionality, even though each will implement the functionality differently
b) provide different types of objects with the same functionality, even though each will implement the functionality differently
c) provide default implementations of methods and properties
d) None of the above.
Answer: b

4. Which of the following characteristics can be used to create an interface for a file, a cat and a house?
a) door
b) tail
c) age
d) None of the above.
Answer: c

5. Which of the following does *not* complete the sentence correctly?
An interface ____.
a) forces classes that implement it to declare *all* the interface methods.
b) is used in place of an **abstract** class when there is *no* default implementation to inherit.
c) cannot be instantiated.
d) can be instantiated.
Answer: d

6. Interfaces can have ____ methods.
a) 0
b) 1
c) 2

d) any number of
Answer: d

7. Which is used to specify that a class will be implementing an interface?
a) **using**
b) **:**
c) **implements**
d) **extends**.
Answer: b

8. A class that implements an interface but does *not* declare all of the interface's methods must be declared:
a) **public**
b) **interface**
c) **abstract**
d) **final**
Answer: c

9. Constants declared in an interface are implicitly _______.
a. **private**.
b. **static**.
c. **abstract**.
d. All of the above.
Answer: b

10. If a class leaves one method in an interface undeclared, the class is implicitly declared by C# as an **abstract** class.
Answer: False. If a class leaves one method in an interface undeclared, the class becomes an **abstract** class and must be declared **abstract** in the first line of its class declaration.

11. An interface is typically used in place of an **abstract** class when there is no default implementation to inherit.
Answer: True.

12. An interface is used when there's a default implementation to inherit.
Answer: False. An interface is used when there's *no* default implementation to inherit.

13. Abstract classes can provide data *and* services for objects.
Answer: True.

14. Declaring an interface protected allows for extra security precautions.
Answer: False. Declaring an interface private or protected is an error.

15. A class that implements an interface must define all methods and properties of that interface.
Answer: True.

16. An interface can not provide properties with get and set accessors.
Answer: False. An interface can provide properties with get and set accessors.

17. A class that implements an interface may not act as a base class for other classes.
Answer: False. A class that implements an interface may be inherited by derived classes.

18. An interface reference may invoke *only* the methods that the interface declares.
Answer: True.

12.8 Operator Overloading

1. Operator overloading is the process of:
a) enabling C#'s operators to work with class objects
b) using operators to create new classes
c) using operators to specify which versions of overloaded methods to use
d) None of the above.
Answer: a

2. Overloaded operator methods must be declared public and ________.
a) sealed
b) protected
c) static
d) None of the above.
Answer: c

3. Keyword _______ is used to indicate a method overloads a specific operator.
a) implement
b) operator
c) overload
d) op
Answer: b

4. Methods that overload binary operators must take two arguments.
Answer: True.

5. Operators should be overloaded to perform actions similar to those that they normally perform on objects of built-in types.
Answer: True.

6. At least one argument of an operator overload method must be a reference to an object of the class in which the operator is overloaded.
Answer: True.