Lesson 2: Introduction to Object-Oriented Programming

1. You are developing code for a method that calculates the discount for the items sold. You name the method `CalculateDiscount`. The method defines a variable, `percentValue` of the type `double`. You need to make sure that `percentValue` is accessible only within the `CalculateDiscount` method. What access modifier should you use when defining the `percentValue` variable?
a) `private`
b) `protected`
c) `internal`
d) `public`

Answer: a
Difficulty: Medium
Section Reference: Understanding Access Modifiers
The `private` modifier restricts the access to the class in which the member was defined. The `protected` modifier restricts the access to the containing class and to any class derived directly or indirectly from the containing class. The `internal` modifier restricts the access to the code in the same assembly. The `public` modifier does not restrict access.

2. You are developing code that defines an `InitFields` method. The method takes two parameters of data type `double` and does not return any value to the calling code. Which of the following code segments would you use to define the `InitFields` method?

a)
```
public double InitFields(double l, double w)
{
    length = l;
    width = w;
    return length * width;
}
```

b)
```
public void InitFields(double l, double w)
{
    length = l;
    width = w;
}
```

c)
```
public void InitFields(double l)
{
    length = l;
    width = l;
    return;
}
```

d)
```
public double InitFields(double l, double w)
{
    length = l;
    width = w;
}
```

Answer: b
Difficulty: Medium
Section Reference: Understanding Methods
If a method does not intend to return any value, its return type is specified by the keyword `void`. As the method takes two parameters of data type `double`, the parameter list must declare two variables of type `double`.

3. You created a class named `GeoShape`. You defined a method called `Area` in the `GeoShape` class. This method calculates the area of a geometric shape. You want the derived classes of `GeoShape` to supersede this functionality to support the area calculation of additional geometric shapes. When the method `Area` is invoked on a `GeoShape` object, the area should be calculated based on the runtime type of the `GeoShape` object. Which keyword should you use with the definition of the `Area` method in the `GeoShape` class?
a) `abstract`
b) `virtual`
c) `new`
d) `overrides`

Answer: b
Difficulty: Medium
Section Reference: Understanding Polymorphism
Use the virtual keyword to define the `Area` method. When a virtual method is invoked, the runtime type of the object is checked for an overriding member. The overriding member in the most derived class is called, which might be the original member, if no derived class has overridden the member.

4. Suppose that you defined a class `Scenario` that defines functionality for running customized pivot transform on large data sets. You do not want the functionality of this class to be inherited into derived classes. What keyword should you use to define the `Scenario` class?
a) `sealed`
b) `abstract`
c) `private`
d) `internal`

Answer: a
Difficulty: Medium
Section Reference: Understanding Inheritance
Use the `sealed` keyword to define the `Scenario` class. When applied to a class, the `sealed` modifier prevents other classes from inheriting from it.

5. You need to provide printing functionality to several of your classes. Each class's algorithm for printing will likely be different. Also, not all the classes have an "is-a" relationship with each other. How should you support this functionality?
a) Add the print functionality to a base class with the `public` access modifier.
b) Have all classes inherit from an abstract base class and override the base-class method to provide their own print functionality.

c) Have all the classes inherit from a base class that provides the print functionality.
d) Create a common interface that all classes implement.

Answer: d
Difficulty: Medium
Section Reference: Understanding Interfaces
You should create a common interface that is implemented by all classes. Interfaces are used to establish contracts through which objects can interact with each other without knowing the implementation details.

6. You are writing code for a class named `Book`. You should be able to get a list of all books sorted by the author's last name. You need to write code to define this behavior of a class. Which of the following class elements should you use?
a) method
b) property
c) event
d) delegate

Answer: a
Difficulty: Medium
Section Reference: Understanding Methods
A method defines the behavior of a class. You can write a method that returns a list of all books sorted by the author's last name.

7. Suppose that you are writing code for a class named `Product`. You need to make sure that the data members of the class are initialized to their correct values as soon as you create an object of the `Product` class. The initialization code should always be executed. What should you do?
a) Create a static method in the `Product` class to initialize data members.
b) Create a constructor in the `Product` class to initialize data members.
c) Create a static property in the `Product` class to initialize data members.
d) Create an event in the `Product` class to initialize data members.

Answer: b
Difficulty: Medium
Section Reference: Understanding Classes
Constructors are special class methods that are executed when a new instance of a class is created. Constructors are used to initialize the object's data members.

8. You are creating a new class named `Sphere` derived from the `Shape` class. The `Shape` class has the following code:

```
class Shape
{
    public virtual void Area()
    {
        // additional code...
    }
```

```
}
```

The `Area` method in the `Shape` class should provide new functionality but also hide the `Shape` class implementation of the `Area` method. Which code segment should you use to accomplish this?

a)
```
class Sphere : Shape
{
    public override void Area()
    {
        // additional code ...
    }
}
```

b)
```
class Sphere : Shape
{
    public new void Area()
    {
        // additional code ...
    }
}
```

c)
```
class Sphere : Shape
{
    public virtual void Area()
    {
        // additional code ...
    }
}
```

d)
```
class Sphere : Shape
{
    public static void Area()
    {
        // additional code ...
    }
}
```

Answer: b
Difficulty: Medium
Section Reference: Understanding Polymorphism
The `new` keyword creates a new member of the same name in the derived class and hides the base class implementation. The `override` keyword is not the correct answer because it replaces a base class member in a derived class.

9. You are creating a new class named `Polygon`. You write the following code:

```
class Polygon : IComparable
{
    public double Length { get; set; }
    public double Width { get; set; }

    public double GetArea()
```

```
        {
            return Length * Width;
        }

        public int CompareTo(object obj)
        {
            // to be completed
        }
    }
```

You need to complete the definition of the `CompareTo` method to enable comparison of the `Polygon` objects. Which of the following code segments should you use?

a)
```
    public int CompareTo(object obj)
    {
        Polygon target = (Polygon)obj;
        double diff = this.GetArea() - target.GetArea();

        if (diff == 0)
            return 0;
        else if (diff > 0)
            return 1;
        else return -1;
    }
```

b)
```
    public int CompareTo(object obj)
    {
        Polygon target = (Polygon)obj;
        double diff = this.GetArea() - target.GetArea();

        if (diff == 0)
            return 1;
        else if (diff > 0)
            return -1;
        else return 0;
    }
```

c)
```
    public int CompareTo(object obj)
    {
        Polygon target = (Polygon)obj;

        if (this == target)
            return 0;
        else if (this > target)
            return 1;
        else return -1;
    }
```

d)
```
    public int CompareTo(object obj)
    {
        Polygon target = (Polygon)obj;

        if (this == target)
            return 1;
```

```
        else if (this > target)
            return -1;
        else return 0;
    }
```

Answer: a
Difficulty: Medium
Section Reference: Understanding Interfaces
The return value of the `CompareTo` method indicates the result of comparing the given parameter with the current object. According to the documentation of the `CompareTo` method,

- If the instance is equal to the parameter, `CompareTo` returns 0.
- If the parameter value is less than the instance or if the parameter is null, a positive value is returned.
- If the parameter value is greater than the instance, a negative value is returned.
- If the parameter is not of the compatible type, an `ArgumentException` is thrown.

10. You are writing code for a new method named `Process`:

```
void Draw(object o)
{

}
```

The code receives a parameter of type `object`. You need to cast this object into the type `Polygon`. At times, the value of `o` that is passed to the method might not be a valid `Polygon` value. You need to make sure that the code does not generate any `System.InvalidCastException` errors while doing the conversions. Which of the following lines of code should you use inside the `Draw` method to accomplish this goal?
a) `Polygon p = (Polygon) o;`
b) `Polygon p = o is Polygon;`
c) `Polygon p = o as Polygon;`
d) `Polygon p = (o != null) ? o as Polygon : (Polygon) o;`

Answer: c
Difficulty: Medium
Section Reference: Understanding Inheritance
The `as` operator is similar to the cast operation but, in the case of `as`, if the type conversion is not possible, null is returned rather than an exception raised. An exception may be generated with the code in the other answer choices.

11. You are writing code to handle events in your program. You define a delegate named `PolygonHandler` like this:

```
public delegate void PolygonHandler(Polygon  p);
```

You also create a variable of the `PolygonHandler` type as follows:

```
PolygonHandler handler;
```

Later in the program, you need to add a method named `CalculateArea` to the method invocation list of the `handler` variable. The signature of the `CalculateArea` method matches the signature of the `PolygonHandler` method. Any code that you write should not affect any existing event-handling code. Given this restriction, which of the following code lines should you write?
a) `handler = new PolygonHandler(CalculateArea);`
b) `handler = CalculateArea;`
c) `handler += CalculateArea;`
d) `handler -= CalculateArea;`

Answer: c
Difficulty: Medium
Section Reference: Understanding Events
You need to use the += operator rather than the simple assignment operator (=) to attach the event handler. By using the += operator, you ensure that this event handler will be added to a list of event handlers already attached with the event.

12. You are developing a C# application. You create a class of the name `Widget`. You use some third-party libraries, one of which also contains a class of the name `Widget`. You need to make sure that using the `Widget` class in your code causes no ambiguity. Which C# keyword should you use to address this requirement?
a) `namespace`
b) `override`
c) `delegate`
d) `class`

Answer: a
Difficulty: Medium
Section Reference: Understanding Namespaces
Use the `namespace` keyword. A namespace is a language element that allows you to organize code and create globally unique class names.

13. You are reviewing a C# program that contains the following class:

```
public class Rectangle
{
    public double Length {get; set;}
    public double Width { get; set; }
}
```

The program executes the following code as part of the `Main` method:

```
Rectangle r1, r2;
r1 = new Rectangle { Length = 10.0, Width = 20.0 };
r2 = r1;
r2.Length = 30;
Console.WriteLine(r1.Length);
```

What will be the output when this code is executed?

a) 10
b) 20
c) 30
d) 40

Answer: c
Difficulty: Medium
Section Reference: Understanding Values and References

The class `Rectangle` is a reference type, and the content of variable `r1` is actually a reference to a memory location that holds a `Rectangle` object. So, after the `r2 = r1;` statement,

both `r1` and `r2` point to the same memory location and in turn the same `Rectangle` object. In other words, there is only one rectangle object in memory, and both `r1` and `r2` are referring to it. When the `Length` property is modified, the change applies to both objects `r1` and `r2`.

14. You are reviewing a C# program. The program contains the following class:

```
public struct Rectangle
{
    public double Length {get; set;}
    public double Width { get; set; }
}
```

The program executes the following code as part of the `Main` method:

```
Rectangle r1, r2;
r1 = new Rectangle { Length = 10.0, Width = 20.0 };
r2 = r1;
r2.Length = 30;
Console.WriteLine(r1.Length);
```

What will be the output when this code is executed?

a) 10
b) 20
c) 30
d) 40

Answer: a
Difficulty: Medium
Section Reference: Understanding Values and References
The `struct` is a value rather than a reference type, so both `r1` and `r2` maintain their own copies of data. So, after the `r2 = r1;` statement, both `r1` and `r2` point to different memory locations. When the `Length` property for `r2` object is modified, the change doesn't affect the object `r1`.

15. You are developing a C# application. You need to decide whether to declare a class member as static. Which of the following statements is true about static members of a class?
a) You can use the `this` keyword reference with a static method or property.
b) Only one copy of a static field is shared by all instances of a class.
c) Static members of a class can be used only after an instance of a class is created.
d) The `static` keyword is used to declare members that do not belong to individual objects but to a class itself.

Answer: d
Difficulty: Medium
Section Reference: Understanding Static Members
The `static` keyword is used to declare members that do not belong to individual objects but to a class itself. A static member cannot be referenced through an instance object. Instead, a static member is referenced through the class name. It is not possible to use the `this` keyword reference with a static method or property because the `this` keyword can be used only to access instance objects.

16. Suppose that you are a new C# developer and are reviewing object-oriented programming fundamentals. Which of the following statements is not true?
a) A class is a concrete instance of an object.
b) A class defines the template for an object.
c) A class is a definition of a new data type.
d) A constructor is used to initialize the data members of the object.

Answer: a
Difficulty: Medium
Section Reference: Understanding Static Members
A class is not a concrete instance of an object. Instead, an object is a concrete instance of a class. The facts in the other answer choices are all correct.

17. You are C# developer who is developing a Windows application. You develop a new class that must be accessible to all the code packaged in the same assembly. Even the classes that are in the same assembly but do not directly or indirectly inherit from this class must be able to access the code. Any code outside the assembly should not be able to access the new class. Which access modifier should you use to declare the new class?
a) `public`
b) `protected`
c) `private`
d) `internal`

Answer: d
Difficulty: Medium
Section Reference: Understanding Access Modifiers
For the `private` access modifier, access is restricted only to the containing class. For the `public` access modifier, access is not restricted. For the `protected` access modifier, access is restricted

only to the derived classes. For the `internal` access modifier, access is restricted only to the code in the same assembly.

18. You are C# developer who is developing a Windows application. You need to provide a common definition of a base class that can be shared by multiple derived classes. Which keyword should you use to declare the new class?
a) `virtual`
b) `sealed`
c) `interface`
d) `abstract`

Answer: d
Difficulty: Medium
Section Reference: Understanding Inheritance
The `abstract` classes provide a common definition of a base class that can be shared by multiple derived classes. The `sealed` classes, on the other hand, provide complete functionality but cannot be used as base classes. The virtual and interface keywords cannot be applied to a class.

19. You are C# developer who is developing a Windows application. You write the following code:

```
Object o;
```

Later in the code, you need to assign the value in the variable `o` to an object of `Rectangle` type. You expect that at runtime the value in the variable `o` is compatible with the `Rectangle` class. However, you need to make sure that no exceptions are raised when the value is assigned. Which of the following code should you use?
a) `Rectangle r = (Rectangle) o;`
b) `Rectangle r = o;`
c) `Rectangle r = o as Rectangle;`
d) `Rectangle r = o is Rectangle;`

Answer: c
Difficulty: Medium
Section Reference: Understanding Inheritance
In case of simple cast operation, the runtime checks whether the value of the variable `o` is compatible with the `Rectangle` class. If, at execution time, the value of `o` is not compatible with the `Rectangle` class, the runtime throws a `System.InvalidCastException`. The `as` operator is similar to the cast operation but, in the case of `as`, if the type conversion is not possible, null is returned rather than an exception raised.

20. You are C# developer who is developing a Windows application. You need to provide derived classes the ability to share common functionality with base classes but still define their own unique behavior. Which object-oriented programming concept should you use to accomplish this functionality?
a) encapsulation

b) abstraction
c) polymorphism
d) inheritance

Answer: c
Difficulty: Medium
Section Reference: Understanding Polymorphism
Polymorphism is the ability of derived classes to share common functionality with base classes but still define their own unique behavior. Inheritance is a feature of object-oriented programming that allows you to develop a class once, and then reuse that code over and over as the basis of new classes.