

03 Object-Oriented Programming

Test your knowledge

1. What are the six combinations of access modifier keywords and what do they do?

public	There are no restrictions on accessing public members.
private	Access is limited to within the class definition. This is the default access modifier type if none is formally specified
protected	Access is limited to within the class definition and any class that inherits from the class
internal	Access is limited exclusively to classes defined within the current project assembly
protected internal	Access is limited to the current assembly and types derived from the containing class. All members in current project and all members in derived class can access the variables.
private protected	Access is limited to the containing class or types derived from the containing class within the current assembly.

Reference:

<https://www.c-sharpcorner.com/uploadfile/puranindia/what-are-access-modifiers-in-C-Sharp/>

2. What is the difference between the static, const, and readonly keywords when applied to a type member?

static: make member of a class to belong to the type rather than to the instance of the type.

const: variable or object need to be initialized at compile time.

readonly: allow the variable or object to be initialized either at compile time or runtime.

Reference: <https://www.pluralsight.com/guides/const-vs-readonly-vs-static-csharp>

3. What does a constructor do?

Is call then the instance of the classs created, and will initial the fields in this instance.

4. Why is the partial keyword useful?

We can split a class definition while implement a large project.

5. What is a tuple?

Tuple is a C# data structure which contains a squence of smae or different data type.

6. What does the C# record keyword do?

Make a class a read only class.

7. What does overloading and overriding mean?

overloading: two or more methods in the same class have the same name but different parameter list. A runtime polymorphism.

overriding: methods that have the same signature being implemented in a super class and this super class's sub classes. A compile time polymorphism.

8. What is the difference between a field and a property?
A field should be kept private to a class.
A property will provide access to a field.
9. How do you make a method parameter optional?
Give these parameters default values.
10. What is an interface and how is it different from an abstract class?
Interface: define common behaviors and functions.
Abstract class: base class to its subclass.
11. What accessibility level are members of an interface?
Public Abstract.
12. True/False. Polymorphism allows derived classes to provide different implementations of the same method.
True.
13. True/False. The override keyword is used to indicate that a method in a derived class is providing its own implementation of a method.
True.
14. True/False. The new keyword is used to indicate that a method in a derived class is providing its own implementation of a method.
False.
15. True/False. Abstract methods can be used in a normal (non-abstract) class.
False.
16. True/False. Normal (non-abstract) methods can be used in an abstract class.
True.
17. True/False. Derived classes can override methods that were virtual in the base class.
True.
18. True/False. Derived classes can override methods that were abstract in the base class.
True.
19. True/False. In a derived class, you can override a method that was neither virtual nor abstract in the base class.
False.
20. True/False. A class that implements an interface does not have to provide an implementation for all of the members of the interface.
False.

21. True/False. A class that implements an interface is allowed to have other members that aren't defined in the interface.

True.

22. True/False. A class can have more than one base class.

False.

23. True/False. A class can implement more than one interface.

True.

Working with methods

1. Let's make a program that uses methods to accomplish a task. Let's take an array and reverse its contents. For example, if you have 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, it would become 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.

To accomplish this, you'll create three methods: one to create the array, one to reverse the array, and one to print the array at the end.

Your Main method will look something like this:

```
static void Main(string[] args) {  
    int[] numbers = GenerateNumbers();  
    Reverse(numbers);  
    PrintNumbers(numbers);  
}
```

The GenerateNumbers method should return an array of 10 numbers. (For bonus points, change the method to allow the desired length to be passed in, instead of just always being 10.)

The PrintNumbers method should use a for or foreach loop to print out each item in the array. The Reverse method will be the hardest. Give it a try and see what you can make happen. If you get stuck, here's a couple of hints:

Hint #1: To swap two values, you will need to place the value of one variable in a temporary location to make the swap:

```
// Swapping a and b.  
int a = 3;  
int b = 5;  
int temp = a;
```

```
a = b;  
b = temp;
```

Hint #2: Getting the right indices to swap can be a challenge. Use a for loop, starting at 0 and going up to the length of the array / 2. The number you use in the for loop will be the index of the first number to swap, and the other one will be the length of the array minus the index minus 1. This is to account for the fact that the array is 0-based. So basically, you'll be swapping array[index] with array[arrayLength - index - 1].

2. The Fibonacci sequence is a sequence of numbers where the first two numbers are 1 and 1, and every other number in the sequence after it is the sum of the two numbers before it. So the third number is $1 + 1$, which is 2. The fourth number is the 2nd number plus the 3rd, which is $1 + 2$. So the fourth number is 3. The 5th number is the 3rd number plus the 4th number: $2 + 3 = 5$. This keeps going forever.
The first few numbers of the Fibonacci sequence are: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... Because one number is defined by the numbers before it, this sets up a perfect opportunity for using recursion. Your mission, should you choose to accept it, is to create a method called `Fibonacci`, which takes in a number and returns that number of the Fibonacci sequence. So if someone calls `Fibonacci(3)`, it would return the 3rd number in the Fibonacci sequence, which is 2. If someone calls `Fibonacci(8)`, it would return 21. In your `Main` method, write code to loop through the first 10 numbers of the Fibonacci sequence and print them out.

Hint #1: Start with your base case. We know that if it is the 1st or 2nd number, the value will be 1.

Hint #2: For every other item, how is it defined in terms of the numbers before it? Can you come up with an equation or formula that calls the `Fibonacci` method again?

Designing and Building Classes using object-oriented principles

1. Write a program that demonstrates use of four basic principles of object-oriented programming /Abstraction/, /Encapsulation/, /Inheritance/ and /Polymorphism/.
2. Use /Abstraction/ to define different classes for each person type such as Student and Instructor. These classes should have behavior for that type of person.
3. Use /Encapsulation/ to keep many details private in each class.
4. Use /Inheritance/ by leveraging the implementation already created in the Person class to save code in Student and Instructor classes.
5. Use /Polymorphism/ to create virtual methods that derived classes could override to create specific behavior such as salary calculations.
6. Make sure to create appropriate /interfaces/ such as `ICourseService`, `IStudentService`, `IInstructorService`, `IDepartmentService`, `IPersonService`, `IPersonService` (should have person specific methods). `IStudentService`, `IInstructorService` should inherit from `IPersonService`.

Person

Calculate Age of the Person

Calculate the Salary of the person, Use decimal for salary

Salary cannot be negative number

Can have multiple Addresses, should have method to get addresses

Instructor

Belongs to one Department and he can be Head of the Department

Instructor will have added bonus salary based on his experience, calculate his years of experience based on Join

- Date
- Student
 - Can take multiple courses
 - Calculate student GPA based on grades for courses
 - Each course will have grade from A to F
- Course
 - Will have list of enrolled students
- Department
 - Will have one Instructor as head
 - Will have Budget for school year (start and end Date Time)
 - Will offer list of courses

7. Try creating the two classes below, and make a simple program to work with them, as described below

Create a Color class:

On a computer, colors are typically represented with a red, green, blue, and alpha (transparency) value, usually in the range of 0 to 255. Add these as instance variables.

A constructor that takes a red, green, blue, and alpha value.

A constructor that takes just red, green, and blue, while alpha defaults to 255 (opaque).

Methods to get and set the red, green, blue, and alpha values from a Color instance.

A method to get the grayscale value for the color, which is the average of the red, green and blue values.

Create a Ball class:

The Ball class should have instance variables for size and color (the Color class you just created). Let's also add an instance variable that keeps track of the number of times it has been thrown.

Create any constructors you feel would be useful.

Create a Pop method, which changes the ball's size to 0.

Create a Throw method that adds 1 to the throw count, but only if the ball hasn't been popped (has a size of 0).

A method that returns the number of times the ball has been thrown.

Write some code in your Main method to create a few balls, throw them around a few times, pop a few, and try to throw them again, and print out the number of times that the balls have been thrown. (Popped balls shouldn't have changed.)

Explore following topics

Fields

Access modifiers

Enumeration types

Constructors

Methods

Properties

Inheritance
Interfaces
Polymorphism