03 Object-Oriented Programming

Test your knowledge

1.

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

Public, Private, Protected, Internal, Protected Internal, Private Protected

They specify the accessibility of the members in the class

2.

What is the difference between the static, const, and read only keywords when applied to a type member?

Static means it belongs to the class instead of the instance

Const means the value can not be changed after it’s declared

Read only is similar to const but the different value can be assigned in the constructor

3.

What does a constructor do?

A constructor creates a instance of object for the class

4.

Why is the partial keyword useful?

It can be used to implement the functionality of a class in multiple files and combined together at compile time

5.

What is a tuple?

It’s a data structure that contains a sequence of elements of different data types

6.

What does the C# record keyword do?

It can define a reference type that provides built-in function to encapsulate data

7.What does overloading and overriding mean?

Overloading means having the same name for two or more methods with different parameters

Overriding means writing the method inherited from a base class with different implementation

8.

What is the difference between a field and a property?

Fields should be kept private and property is used to get/set fields

9.

How do you make a method parameter optional?

Using = to assign default values when making the methods

10.

What is an interface and how is it different from

abstract class?

An interface is like a contract to get a package of methods, some differences includes: interface cannot have constructor, it can’t implement any methods, you can use multiple interface but only one inherited class

11.

What accessibility level are members of an interface?

Usually they are all public

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.

True, method hiding

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 non 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 the contents of it. 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.

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.

Designing and Building Classes using object-oriented principles

1

.

Write a program that 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

4 / 5