Lucia Fuentes Noriega

03 Object-Oriented Programming

Test your knowledge

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

private, public, protected, internal, protected-internal and private-protected

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

const and redadonly value cannot be modified. readonly can be used on fields but not on local variables and can be assigned or changed at runtime.

Const are compiled at compile time. Static objects belong to the type of an object rather than an instance of a type

3. What does a constructor do?

creates the objects in a class

4. Why is the partial keyword useful?

helpful when working on a bigger project because you can split the files over different classes so that different people can work on

the same project simultaneously.

5. What is a tuple?

data structure that consists of multiple parts. easist way to represent data set with multiple values that may or may not be related

to each toher

6. What does the C# record keyword do?

it defines a referenece type that provides built-in functionality for encapsulating data

7. What does overloading and overriding mean?

overloading is when two or more methods in the same class have the same name but different parameters.

overriding is when the method name and parameter are the same in both the superclass and the child class.

8. What is the difference between a field and a property?

property is what you wrap fields with so that we can change the implementations without breaking the code and so that we do not

show the fields to everyone. Property defines a class public interface while fields are used in the private, internal workings

of a class.

9. How do you make a method parameter optional?

-parameter arrays

-default parameters

-use optional attribute

-method overloading

10. What is an interface and how is it different from abstract class?

Abstract class lets you create functionality that subclasses can implement and override subclasses, while interfaces only allow you to define functionality, not implement it.

11. What accessibility level are members of an interface?

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.

False

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.

False

19. True/False. In a derived class, you can override a method that was neither virtual non abstract in the base class.

True

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.

True

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

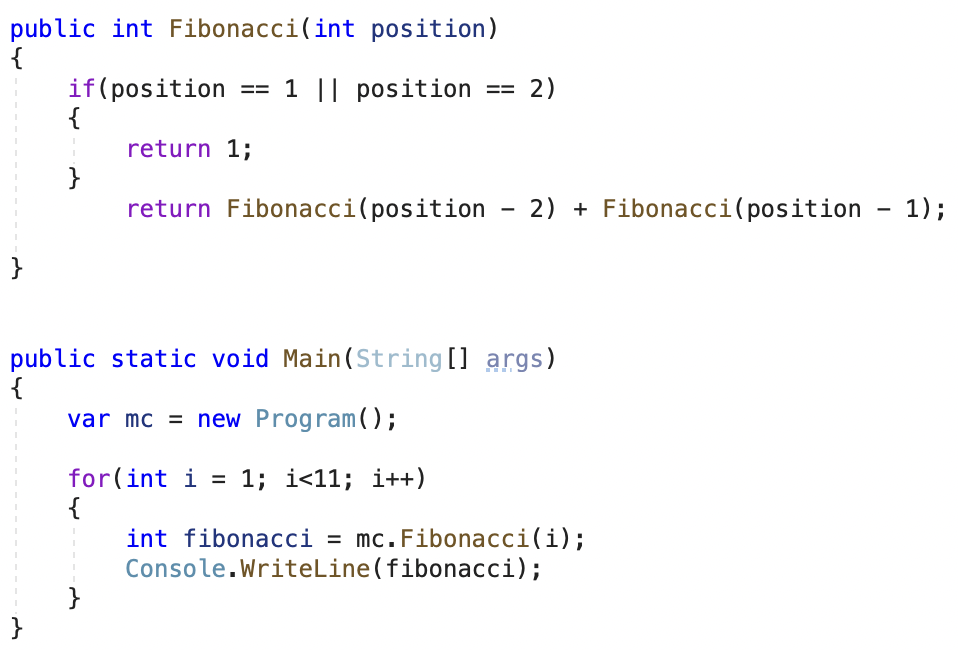
reverse the contents of it. For example, if you have 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Graphical user interface, text, application

Description automatically generated

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.



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 Colorinstance.

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