C# Programming: From Problem Analysis to Program Design, 4th edition

ISBN 978-1-285-09626-1

Chapter 4

1. e. classes

2. a. public

3. a. private

4. e. Employee

5. e. answer = gradStudent.CalculateAvg(87.2, 90);

6. a. ToString( )

7. d. data behavior

8. c. int CalculateAvg(double val1, double val2)

9. d. new

10. a. private member data

11. b. public Employee( )

12. d. accessor

13. e. Name

14. a. Employee employee1 = new Employee( );

15. b. public void SetGpa(double gpaValue)

16. e. methods show the return type on the diagram

17. c. public

18. e. private…public

19. d. void ReduceAccount( )

20. b. override

21. a. Lines 12, 15/16 and 22 are heading.

b. Line 5

c. Line 12

d. Lines 15 and 16

e. Line 18

22. When you write your own classes you create instance methods. Instance methods do not use the static keyword in their heading. To call an instance method, an object must be instantiated and associated with the method. Instance methods are associated with objects. Class methods are associated with the entire class and do not require an object be instantiated. Class methods use the static keyword in their heading and are called with the class name.

23. To override a method you are providing a new definition for a method. The ToString( ) method is often overridden to show what should be displayed if an object of that class type is displayed. If it is not overridden, the namespace and class name are displayed when the object is printed using Write( ) or WriteLine( ).

24. The constructor is used to create an instance of the class. It is a method used to create an object. The default constructor does not have any arguments. If you do not write any constructors for the class you automatically get a default constructor. Write even one constructor and you lose the default.

25. The data members and methods should be indented or tabbed inside the curly braces. Constructors should have public access. Constructor taking three arguments also needs the data type in the heading. Properties are not named following convention of using same name as data member (i.e. Type as opposed to ChairType). ToString( ) method needs string return type. In the ToString( ) method, access the data member (type), not the property to avoid de-referencing it.

Below is the working solution with the above modifications.

public class Chair

{

private string type;

private double weight;

private double cost;

public Chair( )

{

}

public Chair(double aWeight, string aType, double aCost)

{

weight = aWeight;

type = aType;

cost = aCost

}

public string Type

{

get

{

return type;

}

set

{

type = value;

}

}

public override string ToString( )

{

return “Type of Chair: “ + type;

}

}