22. Explain how instance methods differ from class methods. What differs when you want to invoke each different type? Which one requires an object in order for it to be called?

23. What does it mean to override a method? Why should the `ToString( )` method be overridden for user-defined classes?

24. Explain the role of the constructor. What is the default constructor? When do you automatically get a default constructor?

25. The following program has several syntax errors as well as style inconsistencies. Correct the syntax errors and identify the style violations.

```
public class Chair
{
private string type;
private double weight;
private double cost;
private Chair( )
 {
 }
 private Chair(weight, type, cost)
 {
 }
 public string ChairType
 {
       get
       {
              return type;
       }
       set
       {
              ChairType = value;
          }
      }
      public override ToString( )
         {
            return "Type of Chair: " + ChairType;
         }
   }
}
```

PROGRAMMING EXERCISES

1. Create a `Date` `class` with integer data members for year, month, and day. Also include a `string` data member for the name of the month. Include a method that returns the month name (as a `string`) as part of the date. Separate the day from the year with a comma in that method.

Include appropriate constructors, properties, and methods. Override the `ToString ( )` method to display the date formatted with slashes (/) separating the month, day, and year. Create a second `class` that instantiates and test the `Date` `class`.

2. Create a `class` representing a student. Include characteristics such as student number, first and last name, classification, and major. Write at least two constructors. Include properties for each of the data items. Include an instance method that returns a full name (with a space between first and last name). Create a second `class` that instantiates the first `class` with information about yourself. In the second `class`, create a `class` (`static`) method that displays name and major using the instantiated object.

3. Create a Motorway `class` that can be used as extra documentation with directions. Include data members such as name of the highway, type of street (i.e., Road, Street, Avenue, Blvd., Lane, etc.), direction (i.e., E, W, N, or S), surface (i.e., blacktop, gravel, sand, and concrete), number of lanes, toll or no toll, and the party that maintains it. Write instance methods that return the full name of the motorway, full name of the motorway and whether it is toll or not, and full name of the motorway and the number of lanes. Also include a `ToString( )` method that returns all data members with appropriate labels. Include enough constructors to make the class flexible, and experiment with using the class diagram to create the property members. In a second `class` tests the constructors, instance methods, and properties defined in the Motorway `class`.

4. Create an `Employee` `class`. Items to include as data members are employee number, first name, last name, date of hire, job description, department, and monthly salary. The `class` is often used to display an alphabetical listing of all employees. Include appropriate constructors and properties. Provide two instance methods that return the full name. The first should return first name, space last name. The second method should return the name in a format that it could be used for sorting (last name, followed by a comma, space, and then first name). Override the `ToString ( )` method to return all data members. Create a second `class` to test your `Employee` `class`.

5. Create a `Receipt` `class` that could be used by a retail store. Items to include as data members are receipt number, date of purchase, customer number, customer name and address, customer phone number, item number, description, unit price, and quantity purchased. For simplicity,

you may assume each receipt is for a single ticket item and contains a single item number. Include appropriate constructors and properties plus an additional method that calculates the total cost using the quantity and unit price. Override the `ToString ( )` method to return the information about the customer (name and phone number) and the total cost of the item purchased. Create a second `class` to test your `Receipt class`.

6. Create a `Trip class`. Include as data members destination, distance traveled, total cost of gasoline, and number of gallons consumed. Include appropriate constructors and properties. Add additional methods that calculate miles per gallon and the cost per mile. Override the `ToString ( )` method. Create a second `class` to test your `Trip class`.

7. There are a number of national and state parks available to tourists. Create a `Park class`. Include data members such as name of park, location, type of (i.e., national, state, and local) facility, fee, number of employees, number of visitors recorded for the past 12 months, and annual budget. Write separate instance methods that (1) return a string representing name of the park, the location, and type of park; (2) return a string representing the name of the park, the location, and facilities available; (3) compute cost per visitor based on annual budget and the number of visitors during the last 12 months; and (4) compute revenue from fees for the past year based on number of visitors and fee. Also include a `ToString( )` method that returns all data members with appropriate labels. Create a second `class` to test your `Park class`.

8. Write a program that includes an `Employee class` that can be used to calculate and print the take-home pay for a commissioned sales employee. All employees receive 7% of the total sales. Federal tax rate is 18%. Retirement contribution is 10%. Social Security tax rate is 6%. Write instance methods to calculate the commission income, federal and social security tax withholding amounts and the amount withheld for retirement. Use appropriate constants, design an object-oriented solution, and write constructors. Include at least one mutator and one accessor method; provide properties for the other instance variables. Create a second `class` to test your design. Allow the user to enter values for the name of the employee and the sales amount for the week in the second `class`.

9. Write a program that creates a `ProfessorRating class` consisting of professor name, professor ID, and three ratings. The three ratings are used to evaluate easiness, helpfulness, and clarity. Include appropriate

properties. Do not allow the ID to be changed after an `object` has been constructed. Provide a method in the `ProfessorRating` `class` to compute and return the overall rating average. Print all ratings and the average rating formatted with no digits to the right of the decimal from the implementation `class`. In a separate implementation `class`, allow the user to enter the values. Use a single `class` method to enter all data. Test the class by invoking the constructors to create instances of the `ProfessorRating` `class`.

10. Create a `Money` `class` that has as data members dollars and cents. Include `IncrementMoney` and `DecrementMoney` instance methods. Include constructors that enable the Money `class` to be instantiated with a single value representing the full dollar/cent amount as well as a constructor that enables you to create an instance of the `class` by sending two separate integer values representing the dollar and cent amounts. Include an instance method that returns as a string the number of dollars, quarters, nickels, dimes, and pennies represented by the object's value. Override the `ToString( )` method to return the monetary amount formatted with currency symbols. Create a second `class` to test your `Money` `class`.