5.1 For each of the following pairs, which represents a class and which represents an object of that class?

a. Superhero, Superman
b. Justin, Person
c. Rover, Pet
d. Magazine, Time
e. Christmas, Holiday

5.25 Write a method called `multiConcat` that takes a String and an integer as parameters. Return a String that consists of the string parameter concatenated with itself count times, where count is the integer parameter. For example, if the parameter values are `"hi"` and 4, the return value is`"hihihihi"`. Return the original string if the integer parameter is less than 2.

5.26 Overload the `multiConcat` method from Exercise 5.25 such that if the integer parameter is not provided, the method returns the string concatenated with itself. For example, if the parameter is `"test"`, the return value is `"testtest"`.

5.28 Explain why a static method cannot refer to an instance variable.

5.32 Draw a UML class diagram for the Transactions program.

7.1 Which of the following are valid declarations? Which instantiate an array object? Explain your answers.

```
int primes = {2, 3, 4, 5, 7, 11};
float elapsedTimes[] = {11.47, 12.04, 11.72, 13.88};
int[] scores = int[30];
int[] primes = new {2, 3, 5, 7, 11};
int[] scores = new int[30];
char grades[] = {'a', 'b', 'c', 'd', 'f'};
char[] grades = new char[];
```

7.3 Describe what problem occurs in the following code. What modifications should be made to it to eliminate the problem?

```
int[] numbers = {3, 2, 3, 6, 9, 10, 12, 32, 3, 12, 6};
for (int count = 1; count <= numbers.length; count++)
     System.out.println (numbers[count]);
```

7.4 Write an array declaration and any necessary supporting classes to represent the following statements:

a. students' names for a class of 25 students
b. students' test grades for a class of 40 students
c. credit-card transactions that contain a transaction number, a merchant name, and a charge
d. students' names for a class and homework grades for each student
e. for each employee of the L&L International Corporation: the employee number, hire date, and the amount of the last five raises

Programming Problems:

5.1 Revise the `Coin` class such that its state is represented internally using a `boolean` variable. Test the new versions of the class as part of the`CountFlips` and `FlipRace` programs.

5.3 Repeat programming project 5.1, representing the state of the coin using an enumerated type.

5.4 Design and implement a class called `Sphere` that contains instance data that represents the sphere's diameter. Define the `Sphere` constructor to accept and initialize the diameter, and include getter and setter methods for the diameter. Include methods that calculate and return the volume and surface area of the sphere (see programming project 3.5 for the formulas). Include a `toString` method that returns a one-line description of the sphere. Create a driver class called `MultiSphere`, whose main method instantiates and updates several `Sphere` objects.

5.6 Design and implement a class called `Box` that contains instance data that represents the height, width, and depth of the box. Also include a `boolean`variable called `full` as instance data that represents if the box is full or not. Define the `Box` constructor to accept and initialize the height, width, and depth of the box. Each newly created `Box` is empty (the constructor should initialize full to false). Include getter and setter methods for all instance data. Include a `toString` method that returns a one-line description of the box. Create a driver class called `BoxTest`, whose `main` method instantiates and updates several `Box` objects.

5.10 Using the `Die` class defined in this chapter, design and implement a class called `PairOfDice`, composed of two `Die` objects. Include methods to set and get the individual die values, a method to roll the dice, and a method that returns the current sum of the two die values. Rewrite the `SnakeEyes`program using a `PairOfDice` object.

5.11 Using the `PairOfDice` class from programming project 5.10, design and implement a class to play a game called `Pig`. In this game, the user competes against the computer. On each turn, the current player rolls a pair of dice and accumulates points. The goal is to reach 100 points before your opponent does. If, on any turn, the player rolls a 1, all points accumulated for that round are forfeited and control of the dice moves to the other player. If the player rolls two 1's in one turn, the player loses all points accumulated thus far in the game and loses control of the dice. The player may voluntarily turn over the dice after each roll. Therefore the player must decide to either roll again (be a pig) and risk losing points or relinquish control of the dice, possibly allowing the other player to win. Implement the computer player such that it always relinquishes the dice after accumulating 20 or more points in any given round.

7.3 Design and implement an application that creates a histogram that allows you to visually inspect the frequency distribution of a set of values. The program should read in an arbitrary number of integers that are in the range `1` to `100` inclusive; then produce a chart similar to the one below that indicates how many input values fell in the range `1` to `10`, `11` to `20`, and so on. Print one asterisk for each value entered.

```
1   - 10   | *****
11  - 20   | **
21  - 30   | ******************
31  - 40   |
41  - 50   | ***
51  - 60   | ********
61  - 70   | **
71  - 80   | *****
81  - 90   | *******
91  - 100  | *********
```

7.10 Define a class called `Quiz` that manages a set of up to 25 `Question` objects. Define the `add` method of the `Quiz` class to add a question to a quiz. Define the `giveQuiz` method of the `Quiz` class to present each question in turn to the user, accept an answer for each one, and keep track of the results. Define a class called `QuizTime` with a main method that populates a quiz, presents it, and prints the final results.