7.13* (Locating the largest element) Write the following method that returns the location of the largest element in a two-dimensional array.

```java
public static int[] locateLargest(double[][] a)
```

The return value is a one-dimensional array that contains two elements. These two elements indicate the row and column indices of the largest element in the two-dimensional array. Write a test program that prompts the user to enter a two-dimensional array and displays the location of the largest element in the array. Here is a sample run:

Enter the number of rows and columns of the array: 3 4
Enter the array:
23.5 35 2 10
  4.5 3 45 3.5
   35 44 5.5 9.6
The location of the largest element is at (1, 2)

8.11* (Algebra: linear equations) Design a class named `LinearEquation` for a 2 X 2 system of linear equations:

```
ax + by = e
ax + dy = f
```

The class contains:
* Private data fields `a`, `b`, `c`, `d`, `e`, and `f`.
* A constructor with the arguments for `a`, `b`, `c`, `d`, `e`, and `f`.
* Six get methods for `a`, `b`, `c`, `d`, `e`, and `f`.
* A method named `isSolvable()` that returns `true` if is not `0`.
* Methods `getX()` and `getY()` that return the solution for the equation.

Draw the UML diagram for the class. Implement the class. Write a test program that prompts the user to enter `a`, `b`, `c`, `d`, `e`, and `f` and displays the result. If `ad - bc` is `0`, report that “The equation has no solution.” See Exercise 3.3 for sample runs.
9.7* (Phone keypads) The international standard letter/ number mapping found on the telephone is shown below:

- 1: ABC
- 2: DEF
- 3: GHI
- 4: JKL
- 5: MNO
- 6: PQRS
- 7: TUV
- 8: WXYZ
- 9
- *: 0
- #: 

Write a method that returns a number, given an uppercase letter, as follows:

```java
public static int getNumber( char uppercaseLetter)
```

Write a test program that prompts the user to enter a phone number as a string. The input number may contain letters. The program translates a letter (upper- or lowercase) to a digit and leaves all other characters intact. Here is a sample run of the program:

Enter a string: 1-800-Flowers
1-800-3569377

Enter a string: 1800flowers
18003569377

9.15* (Finding the number of uppercase letters in a string) Write a program that passes a string to the main method and displays the number of uppercase letters in a string.

9.19* (Writing/Reading data) Write a program to create a file named Exercise9_19.txt if it does not exist. Write 100 integers created randomly into the file using text I/O. Integers are separated by spaces in the file. Read the data back from the file and display the sorted data.
10.1* (The Time class) Design a class named Time. The class contains:

* Data fields hour, minute, and second that represent a time.
* A no-arg constructor that creates a Time object for the current time. (The values of the data fields will represent the current time.)
* A constructor that constructs a Time object with a specified elapsed time since midnight, Jan 1, 1970, in milliseconds. (The values of the data fields will represent this time.)
* A constructor that constructs a Time object with the specified hour, minute, and second.
* Three get methods for the data fields hour, minute, and second, respectively. * A method named setTime( long elapsedTime) that sets a new time for the object using the elapsed time.

Draw the UML diagram for the class. Implement the class. Write a test program that creates two Time objects (using new Time() and new Time( 55550000)) and display their hour, minute, and second.

( Hint: The first two constructors will extract hour, minute, and second from the elapsed time. For example, if the elapsed time is 555550 seconds, the hour is 10, the minute is 19, and the second is 9. For the no-arg constructor, the current time can be obtained using System. currentTimeMills(), as shown in Listing 2.6, ShowCurrentTime. java.)

11.5 (The Course class) Rewrite the Course class in Listing 10.6. Use an ArrayList to replace an array to store students. You should not change the original contract of the Course class (i.e., the definition of the constructors and methods should not be changed).

13.5* (IllegalTriangleException) Exercise 11.1 defined the Triangle class with three sides. In a triangle, the sum of any two sides is greater than the other side. The Triangle class must adhere to this rule. Create the IllegalTriangleException class, and modify the constructor of the Triangle class to throw an IllegalTriangleException object if a triangle is created with sides that violate the rule, as follows:

```java
/** Construct a triangle with the specified sides */
public Triangle( double side1, double side2, double side3)
   throws IllegalTriangleException {
   // Implement it
}
```

20.5 (Summing series) Write a recursive method to compute the following series:

\[
m(i) = \sum_{n=0}^{i} \frac{(-1)^n}{2n+1}
\]

20.23* (Binary to decimal) Write a recursive method that parses a binary number as a string into a decimal integer. The method header is as follows:

```java
public static int binaryToDecimal( String binaryString)
```
12.1 (Using the FlowLayout manager) Write a program that meets the following requirements (see Figure 12.14):

* Create a frame and set its layout to FlowLayout.
* Create two panels and add them to the frame.
* Each panel contains three buttons. The panel uses FlowLayout.

![Exercise12_1](image_url)

**FIGURE 12.14** Exercise 12.1 places the first three buttons in one panel and the other three buttons in another panel.

15.1* (Displaying a 3 X 3 grid) Write a program that displays a 3 X 3 grid, as shown in Figure 15.27(a). Use red color for vertical lines and blue for horizontal.

![Exercise15_1](image_url) ![Exercise15_2](image_url) ![Exercise15_3](image_url)

**FIGURE 15.27** (a) Exercise 15.1 displays a grid. (b) Exercise 15.2 displays two objects of OvalButton. (c) Exercise 15.3 displays a checkerboard.

16.3* (Moving the ball) Write a program that moves the ball in a panel. You should define a panel class for displaying the ball and provide the methods for moving the button left, right, up, and down, as shown in Figure 16.20(a).

![Exercise16_5](image_url) ![Exercise16_8](image_url) ![Exercise16_9](image_url)

**FIGURE 16.20** (a) Exercise 16.3 displays which button is clicked on a message panel. (b) The program performs addition, subtraction, multiplication, and division on double numbers.
17.3** (Traffic lights) Write a program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green. When a radio button is selected, the light is turned on, and only one light can be on at a time (see Figure 17.35). No light is on when the program starts.

**FIGURE 17.35** The radio buttons are grouped to let you select only one color in the group to control a traffic light.