Assignment 4

Due. Nov. 25 (Tuesday), 23:59.

In this assignment, write all documents in `javadoc` style.

1. Although it is often easier to think of random numbers in the context of games of chance, they have other, more practical uses in computer science and mathematics. For example, you can use random numbers to generate a rough approximation of the constant π by writing a simple program that simulates a dart board. Imagine that you have a dart board hanging on your wall. It consists of a circle painted on a square backdrop. If you throw darts at this board in a random fashion, some will fall inside the circle. If the tosses are truly random, the ratio of the number of darts that land inside the circle to the total number of darts falling anywhere inside the square should be roughly equal to the ratio between the two areas. The ratio of the areas is independent of the actual size of the dart board, as illustrated by the following formula:

$$\frac{\text{darts falling inside the circle}}{\text{darts falling inside the square}} \approx \frac{\text{area of the circle}}{\text{area of the square}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}.$$

   To simulate this process in a program, imagine that the dart board is drawn in the standard Cartesian coordinate plane you learned about in high school. You can model the process of throwing a dart randomly at the square by generating two random floating-point numbers x and y, each of which lies between -1 and 1. This (x, y) point always lies somewhere inside the square. The point (x, y) lies inside the circle if

$$\sqrt{x^2 + y^2} < 1.$$

   You can, however, simplify this condition considerably by squaring each side of the inequality, which gives rise to the following more efficient test:

$$x^2 + y^2 < 1.$$

   If you perform this simulation many times and compute the fraction of the darts that fall within the circle, the result will be somewhere in the neighborhood of $\pi/4$.

   Write a program `PiApproximation.java` that simulates throwing 10,000 darts and then uses that simulation technique described in this exercise to generate and display an approximate value of π. Don't worry if your answer is correct only in the first few digits. The strategy used in this problem is not particularly accurate, even though it occasionally proves useful as a technique for making rough approximations. In mathematics, this technique is called *Monte Carlo integration*, after the gambling center that is the capital of Monaco.

2. Using the `Student` class as a model, implement a new class called `LibraryRecord` that keeps track of the following information for a library book:

   - The title
   - The author
   - The Library of Congress catalog number
   - The publisher
   - The year of publication
   - Whether the book is circulating or noncirculating

   Your class should export the following entries:

   - A constructor that takes all six of these values and creates a new `LibraryRecord` object with them
   - A second version of the constructor that takes only the first five values and initializes the book to be circulating
   - Suitably named getter methods for each of the six fields
   - A setter method for the circulating/noncirculating flag
   - An appropriate implementation of the toString method

   Use the provided `LibraryRecordTest.java` program as a simple test of your implementation of the class.

3. Use the static methods `Integer.parseInt` and `Integer.toString`, Figure 7-2, p. 238, to write a program `HexToDecimalConverter.java` that converts hexadecimal values into their decimal equivalents. Your program should continue to read hexadecimal values until the user enters a 0. A sample run of the program might look like this:

```
This program converts hexadecimal to decimal.
Enter 0 to stop.
Enter a hexadecimal number: 10
10 hex = 16 decimal
Enter a hexadecimal number: 2A
2A hex = 42 decimal
Enter a hexadecimal number: FFFF
FFFF hex = 65535 decimal
Enter a hexadecimal number: 0
```

4. A *palindrome* is a word that reads identically backward and forward, such as *level* or *noon*. Write a predicate method `isPalindrome(str)` that returns `true` if the string `str` is a palindrome, and a private method

```
private String reverseString(String str)
```

   that reverses a string `str`. In addition, design and write a test program `Palindrome.java` that calls `isPalindrome` to demonstrate that it works. In writing the program, concentrate on how to solve the problem simply rather than how to make your solution efficient.

5. Suppose that the `ConsoleProgram` class did not include the `readInt` method to read an integer from the console and supported only the `readLine` method. Use the static methods for the character class, Figure 8-3, p. 262, and the static methods for the String class, Figure 8-4, p. 266, to write a method

   ```
   private int myReadInt(String prompt)
   ```

   that simulates the operation of `readInt` by reading in a line and then translating the characters from that line into an integer. Your implementation of `myReadInt` should allow the input to begin with a `-` character to signal a negative value. Except for that special case, however, your implementation should indicate an error if it finds any characters other than the standard decimal digits. As with the `readInt` method in `acm.io`, your implementation should note that error by printing a message and asking the user to enter a new value, as shown in the following sample run of the `Add2Integers` program from Chapter 2.

   ```
   This program adds two integers.
   Enter n1: 17
   Enter n2: 3.1416
   Illegal integer format
   Enter n2: 25
   The total is 42.
   ```

6. In statistics, a collection of data values is usually referred to as a *distribution*. A primary purpose of statistical analysis is to find ways to compress the complete set of data into summary statistics that express properties of the distribution as a whole. The most common statistical measure is the *mean*, which is simply the traditional average. The mean of a distribution is usually represented by the Greek letter μ.

   Write a method `mean(array)` that returns the mean of an array of type `double`. Test your method by incorporating it into the `GymnasticsMean.java` program provided.