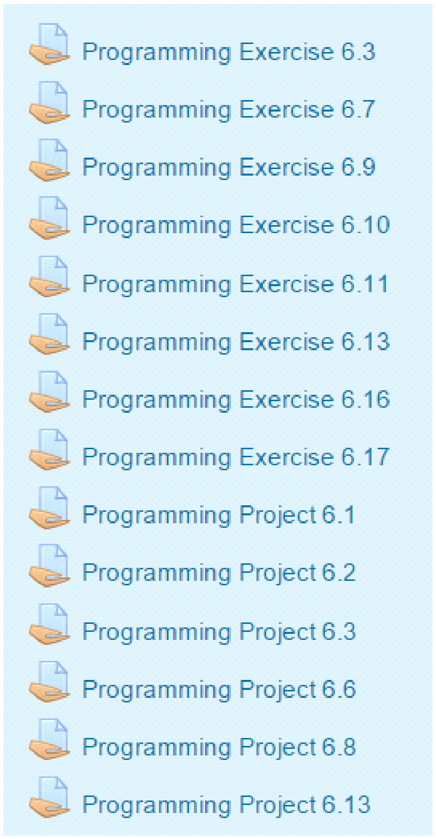
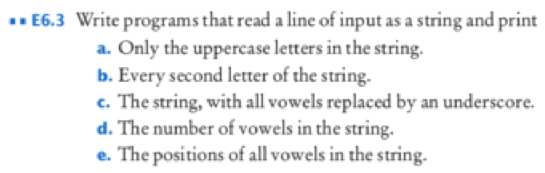
Chapter 6

Monday, November 2, 2015 8:52 AM



Programming Exercise 6.3
Programming Exercise 6.7
Programming Exercise 6.9
Programming Exercise 6.10
Programming Exercise 6.11
Programming Exercise 6.13
Programming Exercise 6.16
Programming Exercise 6.17
Programming Project 6.1
Programming Project 6.2
Programming Project 6.3
Programming Project 6.6
Programming Project 6.8
Programming Project 6.13

11/2/2015 8:52 AM - Screen Clipping



•• **E6.3** Write programs that read a line of input as a string and print

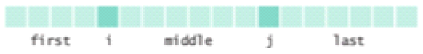
a. Only the uppercase letters in the string.
b. Every second letter of the string.
c. The string, with all vowels replaced by an underscore.
d. The number of vowels in the string.
e. The positions of all vowels in the string.

11/2/2015 8:53 AM - Screen Clipping

▪▪▪ **E6.7** Translate the following pseudocode for randomly permuting the characters in a string into a Java program.

```
Read a word.
Repeat word.length() times
    Pick a random position i in the word, but not the last position.
    Pick a random position j > i in the word.
    Swap the letters at positions j and i.
Print the word.
```

To swap the letters, construct substrings as follows:



Then replace the string with

```
first + word.charAt(j) + middle + word.charAt(i) + last
```

11/2/2015 8:54 AM - Screen Clipping

▪▪ **E6.9** Write a program that reads a word and prints the word in reverse. For example, if the user provides the input `"Harry"`, the program prints

```
yrraH
```

▪ **E6.10** Write a program that reads a word and prints the number of vowels in the word. For this exercise, assume that `a e i o u y` are vowels. For example, if the user provides the input `"Harry"`, the program prints `2 vowels`.

▪▪▪ **E6.11** Write a program that reads a word and prints all substrings, sorted by length. For example, if the user provides the input `"rum"`, the program prints

```
r
u
m
ru
um
rum
```

11/2/2015 8:54 AM - Screen Clipping

▪▪ **E6.13** Write a program that reads a number and prints all of its *binary digits:* Print the remainder `number % 2`, then replace the number with `number / 2`. Keep going until the number is `0`. For example, if the user provides the input 13, the output should be

```
1
0
1
1
```

11/2/2015 8:54 AM - Screen Clipping

▪▪ **E6.16** Write a program that reads an integer and displays, using asterisks, a filled diamond of the given side length. For example, if the side length is 4, the program should display

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

11/2/2015 8:55 AM - Screen Clipping

•• Business E6.17 *Currency conversion.* Write a program that first asks the user to type today's price for one dollar in Japanese yen, then reads U.S. dollar values and converts each to yen. Use `0` as a sentinel.

11/2/2015 8:55 AM - Screen Clipping

•• P6.1 *Mean and standard deviation.* Write a program that reads a set of floating-point data values. Choose an appropriate mechanism for prompting for the end of the data set. When all values have been read, print out the count of the values, the average, and the standard deviation. The average of a data set $\{x_1, \ldots, x_n\}$ is $\bar{x} = \sum x_i / n$, where $\sum x_i = x_1 + \ldots + x_n$ is the sum of the input values. The standard deviation is

$$s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n - 1}}$$

However, this formula is not suitable for the task. By the time the program has computed $\bar{x}$, the individual x_i are long gone. Until you know how to save these values, use the numerically less stable formula

$$s = \sqrt{\frac{\sum x_i^2 - \frac{1}{n}\left(\sum x_i\right)^2}{n - 1}}$$

You can compute this quantity by keeping track of the count, the sum, and the sum of squares as you process the input values.

Your program should use a class `DataSet`. That class should have a method

```
public void add(double value)
```

and methods `getAverage` and `getStandardDeviation`.

11/2/2015 8:56 AM - Screen Clipping

•• P6.2 The *Fibonacci numbers* are defined by the sequence

$$f_1 = 1$$
$$f_2 = 1$$
$$f_n = f_{n-1} + f_{n-2}$$

Reformulate that as

```
fold1 = 1;
fold2 = 1;
fnew = fold1 + fold2;
```

After that, discard `fold2`, which is no longer needed, and set `fold2` to `fold1` and `fold1` to `fnew`. Repeat an appropriate number of times.

Implement a program that prompts the user for an integer *n* and prints the *n*th Fibonacci number, using the above algorithm.

Fibonacci numbers describe the growth of a rabbit population.

11/2/2015 8:57 AM - Screen Clipping

••• P6.3 *Factoring of integers.* Write a program that asks the user for an integer and then prints out all its factors. For example, when the user enters 150, the program should print

```
2
3
5
5
```

Use a class `FactorGenerator` with a constructor `FactorGenerator(int numberToFactor)` and methods `nextFactor` and `hasMoreFactors`. Supply a class `FactorPrinter` whose `main` method reads a user input, constructs a `FactorGenerator` object, and prints the factors.

11/2/2015 8:57 AM - Screen Clipping

•• **P6.6** *The Drunkard's Walk.* A drunkard in a grid of streets randomly picks one of four directions and stumbles to the next intersection, then again randomly picks one of four directions, and so on. You might think that on average the drunkard doesn't move very far because the choices cancel each other out, but that is not the case.

11/2/2015 8:58 AM - Screen Clipping

Represent locations as integer pairs (x, y). Implement the drunkard's walk over 100 intersections, starting at (0, 0), and print the ending location.

11/2/2015 8:58 AM - Screen Clipping

•• **P6.8** *The Buffon Needle Experiment.* The following experiment was devised by Comte Georges-Louis Leclerc de Buffon (1707–1788), a French naturalist. A needle of length 1 inch is dropped onto paper that is ruled with lines 2 inches apart. If the needle drops onto a line, we count it as a *hit*. (See Figure 10.) Buffon discovered that the quotient *tries/hits* approximates π.

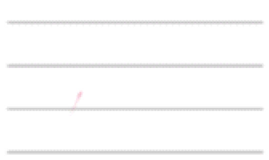


Figure 10
The Buffon Needle Experiment

For the Buffon needle experiment, you must generate two random numbers: one to describe the starting position and one to describe the angle of the needle with the x-axis. Then you need to test whether the needle touches a grid line.

Generate the *lower* point of the needle. Its x-coordinate is irrelevant, and you may assume its y-coordinate y_{low} to be any random number between 0 and 2. The angle α between the needle and the x-axis can be any value between 0 degrees and 180 degrees (π radians). The upper end of the needle has y-coordinate

$$y_{\text{high}} = y_{\text{low}} + \sin\alpha$$

The needle is a hit if y_{high} is at least 2, as shown in Figure 11. Stop after 10,000 tries and print the quotient *tries/hits*. (This program is not suitable for computing the value of π. You need π in the computation of the angle.)

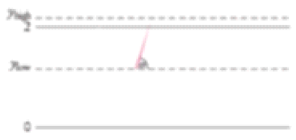


Figure 11
A Hit in the Buffon Needle Experiment

11/2/2015 8:59 AM - Screen Clipping

•• Science P6.13 *Projectile flight*. Suppose a cannonball is propelled straight into the air with a starting velocity v_0. Any calculus book will state that the position of the ball after t seconds is $s(t) = -\frac{1}{2}gt^2 + v_0t$, where $g = 9.81\ \mathrm{m/s^2}$ is the gravitational force of the earth. No calculus textbook ever mentions why someone would want to carry out such an obviously dangerous experiment, so we will do it in the safety of the computer.

In fact, we will confirm the theorem from calculus by a simulation. In our simulation, we will consider how the ball moves in very short time intervals Δt. In a short time interval the velocity v is nearly constant, and we can compute the distance the ball moves as $\Delta s = v\Delta t$. In our program, we will simply set

```
const double DELTA_T = 0.01;
```

11/2/2015 8:59 AM - Screen Clipping

and update the position by

```
s = s + v * DELTA_T;
```

The velocity changes constantly—in fact, it is reduced by the gravitational force of the earth. In a short time interval, $\Delta v = -g\Delta t$, we must keep the velocity updated as

```
v = v - g * DELTA_T;
```

In the next iteration the new velocity is used to update the distance.

Now run the simulation until the cannonball falls back to the earth. Get the initial velocity as an input (100 m/s is a good value). Update the position and velocity 100 times per second, but print out the position only every full second. Also printout the values from the exact formula $s(t) = -\frac{1}{2}gt^2 + v_0t$ for comparison.

Note: You may wonder whether there is a benefit to this simulation when an exact formula is available. Well, the formula from the calculus book is *not* exact. Actually, the gravitational force diminishes the farther the cannonball is away from the surface of the earth. This complicates the algebra sufficiently that it is not possible to give an exact formula for the actual motion, but the computer simulation can simply be extended to apply a variable gravitational force. For cannonballs, the calculus-book formula is actually good enough, but computers are necessary to compute accurate trajectories for higher-flying objects such as ballistic missiles.

11/2/2015 8:59 AM - Screen Clipping