Programming Language: Java
Assignment #3
Due: March 23, 11.59pm

1) (Fibonacci Series) The Fibonacci series
   0, 1, 1, 2, 3, 5, 8, 13, 21, …
   begins with the terms 0 and 1 and has the property that each succeeding term is the sum of the two preceding terms.
   a) Write a method fibonacci(n) that calculates the nth Fibonacci number. Incorporate this method into an application that enables the user to enter the value of n.
   b) Determine the largest Fibonacci number that can be displayed on your system.
   c) Modify the application you wrote in part (a) to use double instead of int to calculate and return Fibonacci numbers, and use this modified application to repeat part (b).

   Example Output
   Enter n: (n < 0 to exit): 5
   Fibonacci number is 3
   Enter n: (n < 0 to exit): 77
   Fibonacci number is 1412467027
   Enter n: (n < 0 to exit): -1

2) (Sieve of Eratosthenes) A prime number is any integer greater than 1 that is evenly divisible only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers. It operates as follows:
   a) Create a primitive type boolean array with all elements initialized to true. Array elements with prime indices will remain true. All other array elements will eventually be set to false.
   b) Starting with array index 2, determine whether a given element is true. If so, loop through the remainder of the array and set to false every element whose index is a multiple of the index for the element with value true. Then continue the process with the next element with value true. For array index 2, all elements beyond element 2 in the array that have indices which are multiples of 2 (indices 4, 6, 8, 10, etc.) will be set to false; for array index 3, all elements beyond element 3 in the array that have indices which are multiples of 3 (indices 6, 9, 12, 15, etc.) will be set to false; and so on.

   When this process completes, the array elements that are still true indicate that the index is a prime number. These indices can be displayed. Write an application that uses an array of 1000 elements to determine and display the prime numbers between 2 and 999. Ignore array elements 0 and 1.

   Example Output
   2 is prime.
   3 is prime.
   ...
   991 is prime.
   997 is prime.
   ZZZ primes found.

3) (Total Sales) Use a two-dimensional array to solve the following problem: A company has four salespeople (1 to 4) who sell five different products (1 to 5). Once a day, each salesperson passes in a slip for each type of product sold. Each slip contains the following:

   a) The salesperson number
   b) The product number
   c) The total dollar value of that product sold that day

Thus, each salesperson passes in between 0 and 5 sales slips per day. Assume that the information from all of the slips for last month is available. Write an application that will read all this information for last month's sales and summarize the total sales by salesperson and by product. All totals should be stored in the two-dimensional array sales. After processing all the information for last month, display the results in tabular format, with each column representing a particular salesperson and each row representing a particular product. Cross-total each row to get the total sales of each product for last month. Cross-total each column to get the total sales by salesperson for last month. Your tabular output should include these cross-totals to the right of the totaled rows and to the bottom of the totaled columns.

Example Output

```
Enter salesperson number (-1 to end): 1
Enter product number: 4
Enter sales amount: 1082
Enter salesperson number (-1 to end): 2
Enter product number: 3
Enter sales amount: 998
Enter salesperson number (-1 to end): 3
Enter product number: 1
Enter sales amount: 678
Enter salesperson number (-1 to end): 4
Enter product number: 1
Enter sales amount: 1554
Enter salesperson number (-1 to end): -1
 Product Salesperson 1 Salesperson 2 Salesperson 3 Salesperson 4     Total
       1          0.00          0.00        678.00       1554.00   2232.00
       2          0.00          0.00          0.00          0.00      0.00
       3          0.00        998.00          0.00          0.00    998.00
       4       1082.00          0.00          0.00          0.00   1082.00
       5          0.00          0.00          0.00          0.00      0.00
   Total       1082.00        998.00        678.00       1554.00
```

4) (Dice Rolling) Write an application to simulate the rolling of two dice. The application should use an object of class Random once to roll the first die and again to roll the second die. The sum of the two values should then be calculated. Each die can show an integer value from 1 to 6, so the sum of the values will vary from 2 to 12, with 7 being the most frequent, sum and 2 and 12 the least frequent. Figure 7.30 shows the 36 possible combinations of the two dice. Your application should roll the dice 36,000 times. Use a one-dimensional array to tally the number of times each possible sum appears. Display the results in tabular format. Determine whether the totals are reasonable (e.g., there are six ways to roll a 7, so approximately one-sixth of the rolls should be 7).

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Example Output

Sum	Frequency	Percentage
2	1007	2
3	2012	5
4	2959	8
5	3946	10
6	5020	13
7	6055	16
8	5014	13
9	4022	11
10	2993	8
11	1997	5
12	975	2

5) Write an application that displays a table of the binary, octal, and hexadecimal equivalents of the decimal numbers in the range 1 through 256. If you are not familiar with these number systems, read Appendix E first.

Example output

```
Decimal Binary      Octal   Hexadecimal
1       1           1       1
2       10          2       2
3       11          3       3
4       100         4       4

.
.
.

252     11111100    374     FC
253     11111101    375     FD
254     11111110    376     FE
255     11111111    377     FF
256     100000000   400     100
```