Lab Exercises 2

Dr. Sarvar Abdullaev
`s.abdullaev@inha.uz`

October 12, 2020

You must attempt all exercises given in this lab. After completing it, you must push it to your corresponding GitHub Classroom repository. Note, you do not have to upload compiled Java bytecode or screenshot of your program's output.

1 Multiples

Write a method `isMultiple` that determines, for a pair of integers, whether the second integer is a multiple of the first. The method should take two integer arguments and return true if the second is a multiple of the first and false otherwise. *[Hint: Use the remainder operator.]* Incorporate this method into an application that inputs a series of pairs of integers (one pair at a time) and determines whether the second value in each pair is a multiple of the first.

2 Perfect Numbers

An integer number is said to be a perfect number if its factors, including 1 (but not the number itself), sum to the number. For example, 6 is a perfect number, because $6 = 1 + 2 + 3$. Write a method `isPerfect` that determines whether parameter number is a perfect number. Use this method in an application that displays all the perfect numbers between 1 and 1000. Display the factors of each perfect number to confirm that the number is indeed perfect. Challenge the computing power of your computer by testing numbers much larger than 1000. Display the results

3 Reverse Digits

Write a method that takes an integer value and returns the number with its digits reversed. For example, given the number 7631, the method should return 1367. Incorporate the method into an application that reads a value from the user and displays the result.

4 Duplicate Elimination

Use a one-dimensional array to solve the following problem: Write an application that inputs five numbers, each between 10 and 100, inclusive. As each number is read, display it only if it's not a duplicate of a number already read. Provide for the "worst case," in which all five numbers are different.

5 Airline Reservations System

A small airline has just purchased a computer for its new automated reservations system. You've been asked to develop the new system. You're to write an application to assign seats on each flight of the airline's only plane (capacity: 10 seats).

Your application should display the following alternatives: `Please type 1 for First Class` and `Please type 2 for Economy`. If the user types 1, your application should assign a seat in the first-class section (seats 1–5). If the user types 2, your application should assign a seat in the economy section (seats 6–10). Your application should then display a boarding pass indicating the person's seat number and whether it's in the first-class or economy section of the plane.

Use a one-dimensional array of primitive type boolean to represent the seating chart of the plane. Initialize all the elements of the array to false to indicate that all the seats are empty. As each seat is assigned, set the corresponding element of the array to true to indicate that the seat is no longer available.

Your application should never assign a seat that has already been assigned. When the economy section is full, your application should ask the person if it's acceptable to be placed in the first-class section (and vice versa). If yes, make the appropriate seat assignment. If no, display the message "`Next flight leaves in 3 hours`".

6 Sieve of Eratosthenes

A prime number is any integer greater than 1 that's evenly divisible only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers. It operates as follows:

1. 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.

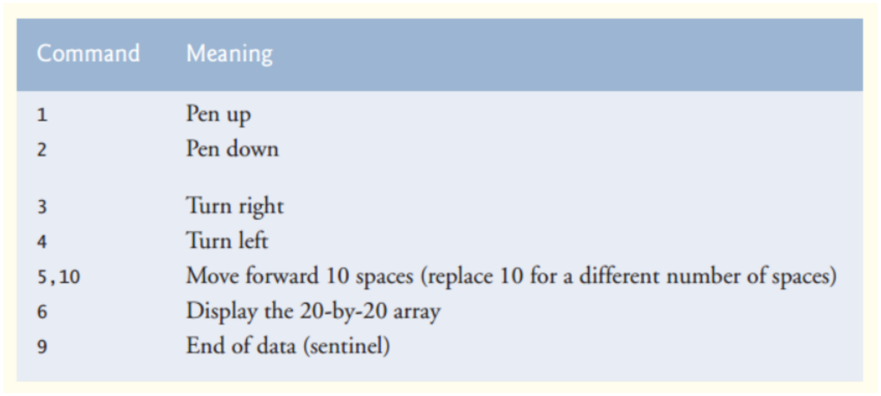
2. Starting with array index 2, determine whether a given element is true. If so, loop through the remainder of the array and set to false every element whose index is a multiple of the index for the element with value true. Then continue the process with the next element with value true. For array index 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 1,000 elements to determine and display the prime numbers between 2 and 999. Ignore array elements 0 and 1.

7 Turtle Graphics

The Logo language made the concept of turtle graphics famous. Imagine a mechanical turtle that walks around the room under the control of a Java application. The turtle holds a pen in one of two positions, up or down. While the pen is down, the turtle traces out shapes as it moves, and while the pen is up, the turtle moves about freely without writing anything. In this problem, you'll simulate the operation of the turtle and create a computerized sketchpad.

Use a 20-by-20 array floor that's initialized to zeros. Read commands from an array that contains them. Keep track of the current position of the turtle at all times and whether the pen is currently up or down. Assume that the turtle always starts at position (0, 0) of the floor with its pen up. The set of turtle commands your application must process are shown in Figure 1



Command	Meaning
1	Pen up
2	Pen down
3	Turn right
4	Turn left
5,10	Move forward 10 spaces (replace 10 for a different number of spaces)
6	Display the 20-by-20 array
9	End of data (sentinel)

Figure 1: Turtle Graphics Commands

Suppose that the turtle is somewhere near the center of the floor. The following "program" would draw and display a 12-by-12 square, leaving the pen in the up position:

```
2
5,12
3
5,12
3
5,12
3
5,12
1
6
9
```

As the turtle moves with the pen down, set the appropriate elements of array floor to 1s. When the 6 command (display the array) is given, wherever there's a 1 in the array, display an asterisk or any character you choose. Wherever there's a 0, display a blank.

Write an application to implement the turtle graphics capabilities discussed here. Write several turtle graphics programs to draw interesting shapes. Add other commands to increase the power of your turtle graphics language.

7.1 Bonus Challenge - Optional

Try this challenge if you have a solid understanding of Algorithms. Can you write an A* algorithm which will return a list of instructions for the turtle to draw an arbitrary pattern on a 20-by-20 grid? Test your algorithm on simple 2D shapes such as circle, square, triangle, etc.