Computer Science 501
Data Structures & Algorithms
The College of Saint Rose
Fall 2015

Lab 1: Java Refresher

Due: 11:59 PM, Friday, September 11, 2015

This week's lab exercise is intended to bring you up-to-speed or back up-to-speed, as the case may be, with Java programming.

There are some questions to answer, practice programs to write, and one more substantial programming assignment to complete.

You may work alone or with a partner on this lab. Only one submission per group is needed.

Getting Set Up

To get your BlueJ environment set up for this week's lab assignment, start BlueJ and choose "New Project" from the "Project" menu. Navigate to your folder for this course and choose the name "Lab1" (no spaces) for the project.

Create a document where you will record your answers to the lecture assignment and lab questions. If you use plain text, call it "`lab1.txt`". If it's a Word document, you can call it whatever you'd like, but when you submit, be sure you convert it to a PDF document "`lab1.pdf`" before you submit it.

Bonus Review Opportunity

We have access to an automated Java review system called "Problets". You can earn up to 16 bonus points for completing problets. The link will be given in class. There are about a dozen Java problets available, and you will earn 2 points each for each problet completed, up to a total of 16 points. You may do more, but the bonus is limited to 16 points. To earn credit, provide the completion code produced when you finish each problet in your submission for this lab. The problets are to be done individually even if you work with a partner for other parts of the lab.

Lecture Assignment Questions

We will usually discuss these questions at the start of class on the lab due date, so no credit can be earned for late submissions of lecture assignment questions.

LA Question 1:

I would like to get a better sense of everyone's backgrounds coming in. Please answer each of the following. (4 points)

- Which degree program are you in, if any?
- What other computer sciences have you taken, either before or at Saint Rose?
- Briefly describe your programming experience. Which programming languages have you used and how complex were the programs you developed?
- What is your favorite restaurant in your home town and what should I order there?
- What types of computers (e.g., PC running Windows, Mac) have you used?
- If you plan to use computers other than those in Saint Rose labs for course work, what type of computer do you plan to use?

LA Question 2:

Bailey Problem 1.2, p. 26. (2 points)

LA Question 3:

a. Show the steps needed to compute the GCD of 31415 and 14142 using Euclid's algorithm. b. How much faster is this than the more naive algorithm that checks consecutive integers starting at 14142 (the smaller of the two numbers whose GCD we are computing) until the GCD is found. (5 points)

LA Question 4:

Once upon a time a farmer went to market and purchased a fox, a goose, and a bag of beans. On his way home, the farmer came to the bank of a river and rented a boat. But in crossing the river by boat, the farmer could carry only himself and a single one of his purchases - the fox, the goose, or the bag of the beans. If left together, the fox would eat the goose, or the goose would eat the beans. The farmer's challenge was to carry himself and his purchases to the far bank of the river, leaving each purchase intact. Find the shortest sequence of crossings you can (where the farmer can carry 0 or 1 of his purchases in the board on each crossing) that solve the problem. (See if you can work it out – don't just look it up!) (5 points)

LA Question 5:

Write a method that takes an array of `int` values as its parameter and returns the number of elements in the array that contain a positive number. (5 points)

LA Question 6:

Write a recursive method, `sumOdd(int n)` to find the sum of the first n odd numbers, for $n \geq 1$. I.e., compute $1 + 3 + 5 + ... + (2n - 1)$. If $n \leq 0$, simply return 0. To receive full credit for this problem, your solution must use recursion and not use any loops. (5 points)

Practice Programs

Practice Program:

Write the class described by Bailey Problem 1.8, p. 27, in a file `PhoneNumber.java`. In addition to fields and constructors as required, include an appropriate `toString` method, and write a `main` method that thoroughly tests your implementation. (7 points)

Practice Program:

Write the class described by Bailey Problem 1.12, p. 27, in a file `ComboLock.java`. Again, include a `main` method that **thoroughly** tests your implementation. Notes: (i) the first parenthetical clause tells you that your class will have two constructors: one that specifies a combination and one that uses the default, and (ii) you may safely ignore the "Aside" in the question if it is confusing. Additional information below. (10 points)

The `press` method corresponds to someone entering the next number in the combination. One of these things can happen when someone calls press:

- As soon as an incorrect `press` is made - the number passed is not the next number in the combination, the lock remains locked until the next `reset`.
- If there have already been enough `press` calls to correspond to the number of digits in the combination, nothing new happens. The lock remains locked or unlocked.
- If all previous `press`es have been correct, this `press` is compared against the next unmatched digit in the combination. If it's correct, and the number is the last one in the combination, the lock becomes unlocked.

Practice Program:

Write a program `MatrixStats.java` that reads in a table of data from a file and the computes and prints various statistics about that data. Details about the program are below. (15 points)

Notes and requirements for the `MatrixStats.java` program:

- Your program should take a file name as its only command-line parameter.
- That file should contain a table of integer data. The first line of the file has two numbers that specify the number of rows and columns of data follow. The remainder of the file contains the actual data. For example, a valid file might contain:

```
3 4
9 1 -2 2
0 2 3 23
-17 10 0 4
```

- Your program should catch and report all exceptions related to reading the file, and should catch any errors (such as not enough numbers are provided, or the file contains non-integer values), and report them. The program should not crash in any circumstance, instead should always report meaningful errors.
- After successful reading of the file into memory, the program should then report the following to standard output (*i.e.*, `System.out`) in this order:
  - the minimum, maximum, sum and average of all numbers in the table, including the row and column numbers of the minimum and maximum
  - the minimum, maximum, sum and average of each row in the table, including the column number within the row of the minimum and maximum entries
  - the minimum and maximum sum of any row, including the row number where each occurs
  - the minimum, maximum, sum and average of each column in the table, including the row number within the column of the minimum and maximum entries
  - the minimum and maximum sum of any column, including the column number where each occurs

  Note that none of these should be computed or reported while the file is being read. They should all be computed from the in-memory representation after the input file has been closed.

For the input file shown above, my program produces the following output:

```
Minimum element at [2][0] is -17
Maximum element at [1][3] is 23
Overall sum: 35, average: 2.92
Row 0:
  Minimum element at [0][2] is -2
  Maximum element at [0][0] is 9
  Row sum: 10, average: 2.50
Row 1:
  Minimum element at [1][0] is 0
  Maximum element at [1][3] is 23
  Row sum: 28, average: 7.00
Row 2:
  Minimum element at [2][0] is -17
```

```
  Maximum element at [2][1] is 10
  Row sum: -3, average: -0.75
Minimum row sum in row 2 is -3
Maximum row sum in row 1 is 28
Column 0:
  Minimum element at [2][0] is -17
  Maximum element at [0][0] is 9
  Column sum: -8, average: -2.67
Column 1:
  Minimum element at [0][1] is 1
  Maximum element at [2][1] is 10
  Column sum: 13, average: 4.33
Column 2:
  Minimum element at [0][2] is -2
  Maximum element at [1][2] is 3
  Column sum: 1, average: 0.33
Column 3:
  Minimum element at [0][3] is 2
  Maximum element at [1][3] is 23
  Column sum: 29, average: 9.67
Minimum column sum in column 0 is -8
Maximum column sum in column 3 is 29
```

Programming Assignment

Write the program for the lab exercise at the end of Chapter 1 of Bailey.

- Write your entire program in a single Java source file `Date.java`. The `main` method required by step 4 of the procedure might normally be placed in a separate class, but keep it in the `Date` class here for simplicity.
- Avoid large `switch` statements by using arrays from which you can look up information such as the day-of-week adjustment table and the number of days in each month.
- Don't forget to answer the thought questions as well. Please put the answers to these in your `lab1.txt` or `lab1.pdf` file.

Submitting

Before 11:59 PM, Friday, September 11, 2015, submit your lab for grading. There are two things you need to do to complete the submission: (i) Copy your file with the answers to the lecture assignment and lab questions into your project directory. Be sure to use the correct file name. If you prepared your answers in Word, export to a PDF file and submit that. (ii) Upload a copy

of your lab (a `.7z` or `.zip` file containing your project directory) using Submission Box at `http://sb.teresco.org` under assignment "Lab1".

Grading

This assignment is worth 100 points, which are distributed as follows:

Feature	Value	Score
Problets bonus points	up to 16	
Lecture assignment questions	26	
`PhoneNumber.java` instance variables	1	
`PhoneNumber.java` constructors	2	
`PhoneNumber.java` `isTollFree` method	1	
`PhoneNumber.java` `toString` method	1	
`PhoneNumber.java` complete `main` method	2	
`ComboLock.java` instance variables	2	
`ComboLock.java` general constructor	1	
`ComboLock.java` default constructor	1	
`ComboLock.java` `press`/`reset`/`lock`/`isLocked`	4	
`ComboLock.java` complete `main` method	3	
`MatrixStats.java` command line parameter	1	
`MatrixStats.java` read from file	2	
`MatrixStats.java` error checking	2	
`MatrixStats.java` compute and print all stats	10	
Basic `Date` class	5	
`Date` constructor to generate an arbirtrary date	3	
`Date` constructor to generate valid random dates	5	
Method in `Date` to compute day of week	5	
`main` method to play Conway's game	8	
`Date.java` comments	5	
`Date.java` naming conventions	3	
`Date.java` formatting	2	
Thought questions	5	
Total	100	