University of New Brunswick
CS1083 – Open Access Version

Module 2 Assignment

The purpose of this assignment is to give you practice:

- applying the linear search algorithm to a realistic problem;
- working with one-dimensional arrays;
- working with a "has a" relationship; and
- generating pseudo-random numbers.

For all assignments:

- Follow the instructions in the document "Java Coding Guidelines.pdf" available within the "Start Here" module, under the "Assignments" topic.
- For each .java file you create, include a javadoc comment (one that begins with /**) at the beginning of the file that describes that Java class. This comment block should include a line that begins with @author followed by your name and student number on the same line. (Note: inclusion of this comment is part of the instructions given in "Java Coding Guidelines.pdf")
- Include comments throughout your programs to explain any non-obvious portions of your code.
- It is recommended that you create a separate folder on your computer for each assignment. You might even wish to create separate sub-folders within an assignment folder if the assignment has multiple parts. Keeping your work organized makes it easier to find things later when you want to review what you have done.
- Few things in life are more frustrating than losing your work while working on an assignment. Get in the habit of saving frequently when working on an assignment. Also, regularly make backup copies of any files you create as part of your work for this course.
- Each module has a link to the discussion forum where you can ask questions and seek help in completing your assignments.
- Submitting your assignment involves creating a pdf file and uploading that single file to the assignment drop box on D2L. In general, that file will tend to include all Java code you wrote for the assignment, plus any output from running your programs that the assignment instructions specify you should capture. Specific submission instructions are included at the end of each assignment.

- To create the pdf file for each assignment, begin by opening a new document using the word processing program of your choice. Save the document using the name “Your Name CS1083 Module x Assignment Submission.docx” Replace x with the correct module number. “docx” may be different depending on which word processing software you choose to use.
- At the beginning of your submission document enter your name, your student number, CS1083, the assignment name (this one is “Module 2 Assignment”), and the date. It doesn’t matter if the date is when you started working on the assignment or when you submit it – either will do.
- You can add content to your submission document as you work on the various questions in the assignment. Clearly label each part of the assignment (“Part A” etc.) in your document. When your document is complete, save / export as a pdf file. Always make sure your pdf file opens properly before uploading it.
- To include Java code in your submission document, copy all the text in your .java file and then paste that text into your submission document. Use a monospaced font (e.g.: Consolas, Courier) for your code to maintain proper indentation.
- To include output from running your program in your submission document, you have two choices. You can either (a) copy and paste the text from your command prompt window, or (b) capture a screen shot of the output and include that as a picture / image in your submission document. Either approach is fine.
- To copy text from your command prompt window, try selecting the desired text and then pressing either command-c (Mac) or control-c (Windows or Linux). If you have issues, you can always use Google to see how to do this on your specific type of computer.
- To capture a screen shot of a selected portion of your command prompt window, try command-shift-4 (Mac), WindowsKey-shift-s (Windows), or shift-PrtScrn (Linux).
- Once you have pasted your output text into your submission document, make sure it is in a monospaced font (e.g.: Consolas, Courier) so you retain the alignment of output.

LotteryTicket

A popular Canadian lottery game involves buying a ticket that contains six numbers between 1 and 49, with no repeated numbers among the six. For example, a ticket might have the following six numbers:

5 9 14 23 28 47

When the draw is held, six winning numbers are randomly selected (again, between 1 and 49, with no duplicates). A ticket can win various prize amounts depending on how many numbers on the ticket match the winning numbers.

Simulate a lottery by coding three Java classes as described below.

Each **LotteryTicket** object stores an integer ticket id, plus an array of six non-repeating integers between 1 and 49. These numbers are not in sorted order.

You can download javadoc documentation for LotteryTicket in the file named:
CS1083_M2_LotteryTicket.pdf
(a download link is provided next to the link for this assignment document)

Code your LotteryTicket class to be consistent with that javadoc documentation.

Your constructor will call the chooseRandomNumbers() method to randomly generate the six numbers. Use Math.random() or the java.util.Random class for this task, whichever you prefer.

For each number to be generated for the ticket, the chooseRandomNumbers() method loops to generate numbers until the duplicateNumber() method returns false. This whole process is repeated six times until the 'numbers' array is full of values, none of which are duplicates.

You will notice that three final variables are defined in this class. Use those values (rather than integer constants) whenever possible when coding the methods. This will make your code more readable and easier to understand (not to mention simpler to change if updates are ever needed in the future).

The toString() method produces a result that is formatted differently from what you might be used to seeing with other toString() methods. Produce a String that displays ticket id and the six numbers formatted like this example :

```
1005: 16  6  3 31 10 26
```

Use the padLeft() method so you append exactly three characters for each of the six numbers. (In the above example, notice two blanks in front of 6 and 3 but only one blank in front of each two-digit number.) You don't need to add the padLeft() method to your LotteryTicket class; the class Util.java containing this method is available for download with this assignment.

The linear search algorithm comes into play twice in the LotteryTicket class, each time with a little twist:

- In the duplicateNumber() method, the linear search algorithm is used to return either true or false (rather than an index like the examples discussed in the instructional video).
- In the countWinningNumbers() method, linear search is used to determine whether or not to add 1 to a counter.

Reminder: Don't try to code this entire class before testing it. Instead, use a simple "throwaway" test class with a main method that creates a single lottery ticket. Code the LotteryTicket constructor and then get the constructor to work with your simple test class. Then add a method to LotteryTicket and do the simplest test you can think of. Keep using this strategy until you've built all the parts of LotteryTicket. This kind of incremental development will save you lots of time and will reduce your frustration. You should use this strategy throughout the course – and actually, throughout your entire career. There is no need to include this kind of simple test class with your eventual assignment submission.

LotteryDraw

A **LotteryDraw** object is used to store an array of LotteryTicket objects.

You can download javadoc documentation for LotteryDraw in the file named:
CS1083_M2_LotteryDraw.pdf
(a download link is provided next to the link for this assignment document)

Code your LotteryDraw class to be consistent with that javadoc documentation.

The size of the 'tickets' array is set when constructing a LotteryDraw object. This array is initially empty. Then the addTicket() method makes it possible to add lottery tickets one at a time to the draw. This means that at any given time the array might be empty, partially filled, or completely filled. The ticketQty instance variable keeps track of how many tickets have been added to the array. Once the array is full, the addTicket() method will not add any more tickets.

The winning numbers for a draw are six randomly selected numbers between 1 and 49, with no duplicates.

Does that sound familiar? It should – that's the exact same criteria used to generate six numbers when constructing a LotteryTicket object. So in your LotteryDraw class, use LotteryTicket as a simple way to generate a set of winning numbers … by having your constructor simply create a LotteryTicket object called winningNumbers. (This means the winning numbers are selected before tickets are added to the draw – we can pretend / assume the winning numbers are kept secret until winners are announced at some later time.)

The prizeAmount array is defined with the following Java code:

```
private static final double[] prizeAmount
   = {0.0, 0.0, 10.0, 100.0, 1000.0, 100000.0, 1000000.0};
```

This array is used by the getPrizeAmount() method to determine the prize won by a ticket as follows:

Quantity of matches with the winning numbers	Prize amount
0 of 6	$0.00
1 of 6	$0.00
2 of 6	$10.00
3 of 6	$100.00
4 of 6	$1,000.00
5 of 6	$100,000.00
All 6 numbers match	$1,000,000.00

Hint: The getPrizeAmount() method should use the quantity of matches as an index into the prizeAmount array. Before doing so, check to make sure that quantity is a valid index for that array.

(Continued on the next page…)

LotteryDrawTest

The **LotteryDrawTest** class is used to try out your LotteryTicket and LotteryDraw classes.

Create a LotteryDraw object that can hold up to 20 tickets.

Loop to add 10 LotteryTicket objects to the draw, with ticket ids from 1001 to 1010.

Then use the various methods provided by LotteryDraw and LotteryTicket to help produce a report that looks exactly like the following (except the randomly generated numbers and prize results will be different every time you run your program):

```
Winning Numbers: 31 8 15 45 9 23

         Tickets           #Matched    Prize
======================   ========  =========
1001:  8 46  5 36 33 16       1       $0.00
1002:  6 10 28 17  9 43       1       $0.00
1003: 29 11 26 30 24  4       0       $0.00
1004:  3 31 23 45 44 49       3     $100.00
1005: 16  6  3 31 10 26       1       $0.00
1006: 14 43 39 30 37 48       0       $0.00
1007: 15 33 25 18 46 32       1       $0.00
1008: 11 17 25 45 43 31       2      $10.00
1009: 27 39 25 21 16  3       0       $0.00
1010: 45 36 33 42  2 48       1       $0.00
```

Use getCurrencyInstance() and format() from the NumberFormat class to format the prizes as currency amounts.

Use Util.padLeft() to help vertically align the values as shown above.
(a download link for Util.java is provided next to the link for this assignment document)

(Submission instructions are on the next page…)

Submission Instructions

Include the following in your submission document:

- Your name, student number, CS1083, Module 2 Assignment, and the date.
- Complete code for your LotteryTicket, LotteryDraw, and LotteryDrawTest classes.
  (You don't need to include the Util class.)
- Two samples of output from running LotteryDrawTest. Include an example where a ticket wins at least $100.00. You might have to run the program a number of times before you get such a winner.

D2L DROPBOX SUBMISSION INSTRUCTIONS

Remember to save / export your submission document as a pdf file, and then upload that one pdf file to D2L as follows:

1. In the top-navigation bar on the course screen, select 'Assessments' and then 'Assignments'.
2. Select the assignment title and follow the instructions to upload your submission document.