ITEC 120 Lab 6

Created by Dr. Ray

Reference links:
http://www.radford.edu/~aaray/template.java
http://www.radford.edu/~aaray/FunctionContainer.java

You will need to turn in a lab report for this lab.

Problem 1: Stock price contract

Stock traders from Moneybags Inc. have hired you to help them figure out how to spend all of the bailout money that they got from the government. They have a system that gathers the price of several related stocks. They need you to write some software to help them decide if they should buy the cheapest stock, the most expensive stock, and/or all of the stocks in the group. You buy all of the stocks if the average price for the stocks is between 10 and 15; you buy the cheapest stock if the most expensive stock is over 30; you buy the most expensive stock if the cheapest stock is less than 5.

They want you to write a program that will apply the previous rules on an array of stock data. You need to show them a prototype that will show them what the report will look like.

Part 1: Create the array

In order to demonstrate that your prototype to works, you need to create an array of size 10 and fill it with random values from 0 to 40 (simulating stock prices).

You can generate random numbers with the following code:

```
import java.util.*;
//other code here
Random rand = new Random();
// Get a random number between 0 and 40
int x = rand.nextInt(40);
```

Part 2: Find the min, max, and average

After you have an array with random values, find the minimum, maximum, and average value for the stocks in the array. Make sure you keep track of the indexes for the minimum and maximum stocks.

Part 3: Print out results

Using the maximum, minimum, and average values, print out which stocks (if any), that the trader should buy. You should print out the index and value of the cheapest and most expensive stocks along with the average stock value. Next you should print out what stocks (index and price) Moneybags traders should buy based on the recommendation listed earlier.

Problem 2: Bluetooth security

Communications Inc. is large company that deals with sending information to people in crowded spaces. They are thinking about using Bluetooth devices to help spread information about line length, flight delays, and similar information. However, they are concerned about how insecure Bluetooth is. They want to know how many times it takes an attacker to randomly guess an encryption key to break into the system, how many times it takes to exhaustively test each key to break into a system, and how long each method takes.

Security on Bluetooth devices works based on PINs. A PIN is usually four digits long, so it ranges from 0 to 9999. When someone wants to connect to a Bluetooth device, they have to enter the correct PIN. You will be writing a program to run tests that determine how many times it takes to enter a PIN to break into a Bluetooth device.

Part 1: Setting up the test environment

Testing a Bluetooth PIN involves creating a random number between 0 and 9999 to represent the initial PIN (the user's PIN). You will need to create an array that holds six different PINs which will be used to test both methods. Next, you need an integer array of size 9999 that represents every possible PIN for the device. You will use this array to hold the number of times each PIN has been tried. Make sure that every value in the array is set to zero before attempting part 2.

Note: the pin 0000 is represented as pin 0 for this lab and the pin 0001 is represented by the pin 1.

Part 2: Testing every PIN

The first method of testing involves testing a PIN starting at X (which is initially 0), stopping if it is the PIN you are looking for, or adding one to the PIN and repeating the process. Use this method to find out how many times it takes to find the first three PINs generated in part 1. Make sure you update the array of every PIN to reflect how many times each PIN has been checked, this will be used in part 3. Also adapt the following code to figure out the average time it takes to break a PIN number using this method:

```
int i=0;
//Preserve the time when the while loop started
Date a = new Date();
while (i < 10000000)
{
     i++;
}
Date b = new Date();
long time1 = a.getTime();
long time2 = b.getTime();
//Print out the # of milliseconds it took the code to
//execute
System.out.println("Time to execute:  " + (time2-time1));
```

Part 3: Randomly testing every PIN

Repeat the test from part 2, except do not increment by one each time. Randomly pick a number between 0 and 9999 and test to see if it is the PIN. Repeat the process until the PIN is found. Make sure you update the array of every PIN to reflect how many times each PIN has been checked, this will be used in part 3. Use the randomly testing method to find out how many times it takes to find the last three PINs generated in part 1.

Part 4: Printing out the results.

Print out how many tries it took to find the three Bluetooth PINs with the testing method in part 2, and then print out how many tries it took to find the Bluetooth PINs from part 3. Next, print out the average time for the first and second testing methods. Lastly, print out the PIN that was checked the most, and the average # of times each PIN was checked.