**CSC 3020 – Java Programming**

**Homework 3 – [your name]**

**25 points – Due February 23, 10am**

**Late deadline is February 24, 11:59pm, but 20% off**

**a)** Save this document with your name and the homework number somewhere in the file name.

**b)** Type/paste your answers into the document.

**c)** Submit this document and your .java file(s) to the Blackboard item where you downloaded this document. Do not submit a zip file but individually attach your files.

**1) [10 points]** You've been hired by *Sort Scorchers* to write a Java console application that sorts an **array list** of random real numbers. Use a validation loop to prompt for and get from the user the number of real numbers to sort in the range 10-10,000. Create and store in the array list that many random real numbers. Randomly generate the real numbers in the range 0-1,000. Print the array list size and use formatted output to print the first ten values of the array list in two columns:

● The first column is the zero-based index of the value.

● The second column is the right-justified value.

Use the insertion sort method to sort the array list. Monitor the number of cycles and swaps during the sort, and the elapsed time in milliseconds to complete the sort. Use formatted output to print the results (cycles, swaps, total cycles and swaps, and elapsed time) in two columns:

● The first column is a left-justified label. Include units if needed.

● The second column is a right-justified value.

Print the array list size and first ten values of the array list again. Format all numbers with a comma. Format all real numbers to one decimal place. Run the program three times with the following inputs. Enter the elapsed times for each run:

|  |  |  |
| --- | --- | --- |
| Run | Array list size | Elapsed time (ms) |
| 1 | 10 | 1 |
| 2 | 5,000 | 55 |
| 3 | 10,000 | 171 |

**//======================================================================**

**//**

**// Title: Sort Scorchers**

**// Course: CSC 3020**

**// Homework: 3-1**

**// Author: Dan Ouellette**

**// Date: 21 February 2018**

**// Description:**

**// This Java console application sorts an array list of random real**

**// numbers. It uses a validation loop to prompt for and get from the**

**// user the number of real numbers to sort in the range 10-10,000. It**

**// then creates and stores in the array list that many random real**

**// numbers. The application prints the unsorted array list size and the**

**// first ten values of the array list. It then uses the insertion sort**

**// method to sort the array list. It monitors the number of cycles and**

**// swaps during the sort, and the elapsed time in milliseconds to**

**// complete the sort. The application prints the results (cycles,**

**// swaps, total cycles and swaps, and elapsed time). It then prints the**

**// array list size and first ten values of the array list again.**

**//**

**//======================================================================**

**package wsu.HW03\_01;**

**// Import classes**

**import java.util.ArrayList;**

**import java.util.Random;**

**import java.util.Scanner;**

**//======================================================================**

**// class HW03\_01**

**//======================================================================**

**public class HW03\_01**

**{**

**//==================================================================**

**// Fields**

**//==================================================================**

**// Declare constants**

**private static final int RANDOM\_MAX = 1000;**

**private static final int PRINT\_MAX = 10;**

**private static final String COLFMTD1 = "%,1d";**

**private static final String COLFMTD2 = "%,12d";**

**private static final String COLFMTF = "%,12.1f";**

**private static final String COLFMTS1 = "%12s";**

**private static final String COLFMTS2 = "%-12s";**

**//------------------------------------------------------------------**

**// randomizeArrayList**

**//------------------------------------------------------------------**

**private static void randomizeArrayList(ArrayList<Double> al,**

**int maxSize)**

**{**

**Random rand = new Random();**

**for (int i = 0; i < maxSize; i++)**

**al.add(rand.nextDouble() \* RANDOM\_MAX);**

**}**

**//------------------------------------------------------------------**

**// printArrayList**

**//------------------------------------------------------------------**

**private static void printArrayList(String heading,**

**ArrayList<Double> al)**

**{**

**System.out.println("\n" + heading);**

**System.out.println("ArrayList size: " +**

**String.format(COLFMTD1, al.size()));**

**System.out.printf(COLFMTS1 + COLFMTS1 + "%n", "Index", "Value");**

**for (int i = 0; i < al.size(); i++)**

**if (i <= PRINT\_MAX)**

**System.out.printf(COLFMTD2 + COLFMTF + "%n", i,**

**al.get(i));**

**}**

**//------------------------------------------------------------------**

**// insertionSortArrayList**

**//------------------------------------------------------------------**

**private static void insertionSortArrayList(ArrayList<Double> al)**

**{**

**// Declare variables**

**double value;**

**int spot;**

**long cycles = 0;**

**long swaps = 0;**

**// Loop to test each value**

**for (int i = 1; i < al.size(); i++)**

**{**

**// Loop to find spot to place value**

**value = al.get(i);**

**spot = i - 1;**

**while (spot >= 0 && al.get(spot) > value)**

**{**

**al.set(spot + 1, al.get(spot));**

**swaps = swaps + 1;**

**cycles = cycles + 1;**

**spot = spot - 1;**

**}**

**// Place value in spot**

**al.set(spot + 1, value);**

**}**

**// Print results**

**System.out.println("\nInsertion sort results");**

**System.out.printf(COLFMTS2 + COLFMTD2 + "%n",**

**"Cycles:", cycles);**

**System.out.printf(COLFMTS2 + COLFMTD2 + "%n",**

**"Swaps:", swaps);**

**System.out.printf(COLFMTS2 + COLFMTD2 + "%n",**

**"Total:", (cycles + swaps));**

**}**

**//------------------------------------------------------------------**

**// main**

**//------------------------------------------------------------------**

**public static void main(String[] args)**

**{**

**// Declare variables**

**Scanner keyboard = new Scanner(System.in);**

**int numbers;**

**ArrayList<Double> al = new ArrayList<Double>();**

**long before;**

**long after;**

**// Show application header**

**System.out.println("Welcome to Sort Scorchers");**

**System.out.println("-------------------------\n");**

**// Loop to prompt for and get number of random numbers**

**System.out.print("Enter the number of random numbers to " +**

**"generate (10-10,000): ");**

**numbers = keyboard.nextInt();**

**while (numbers < 10 || numbers > 10000)**

**{**

**System.out.print("Error: the number of random numbers to " +**

**"generate is outside the range 10-10,000.\n");**

**System.out.print("\nEnter the number of random numbers to " +**

**"generate (10-10,000): ");**

**numbers = keyboard.nextInt();**

**}**

**// Place random numbers in array list and print**

**randomizeArrayList(al, numbers);**

**printArrayList("Unsorted arraylist (first ten values)", al);**

**// Insertion sort array list**

**before = System.currentTimeMillis();**

**insertionSortArrayList(al);**

**after = System.currentTimeMillis();**

**// Print elapsed time**

**System.out.printf(COLFMTS2 + COLFMTD2 + "%n",**

**"Time (ms):", (after - before));**

**// Print sorted array list**

**printArrayList("Sorted arraylist (first ten values)", al);**

**// Close keyboard**

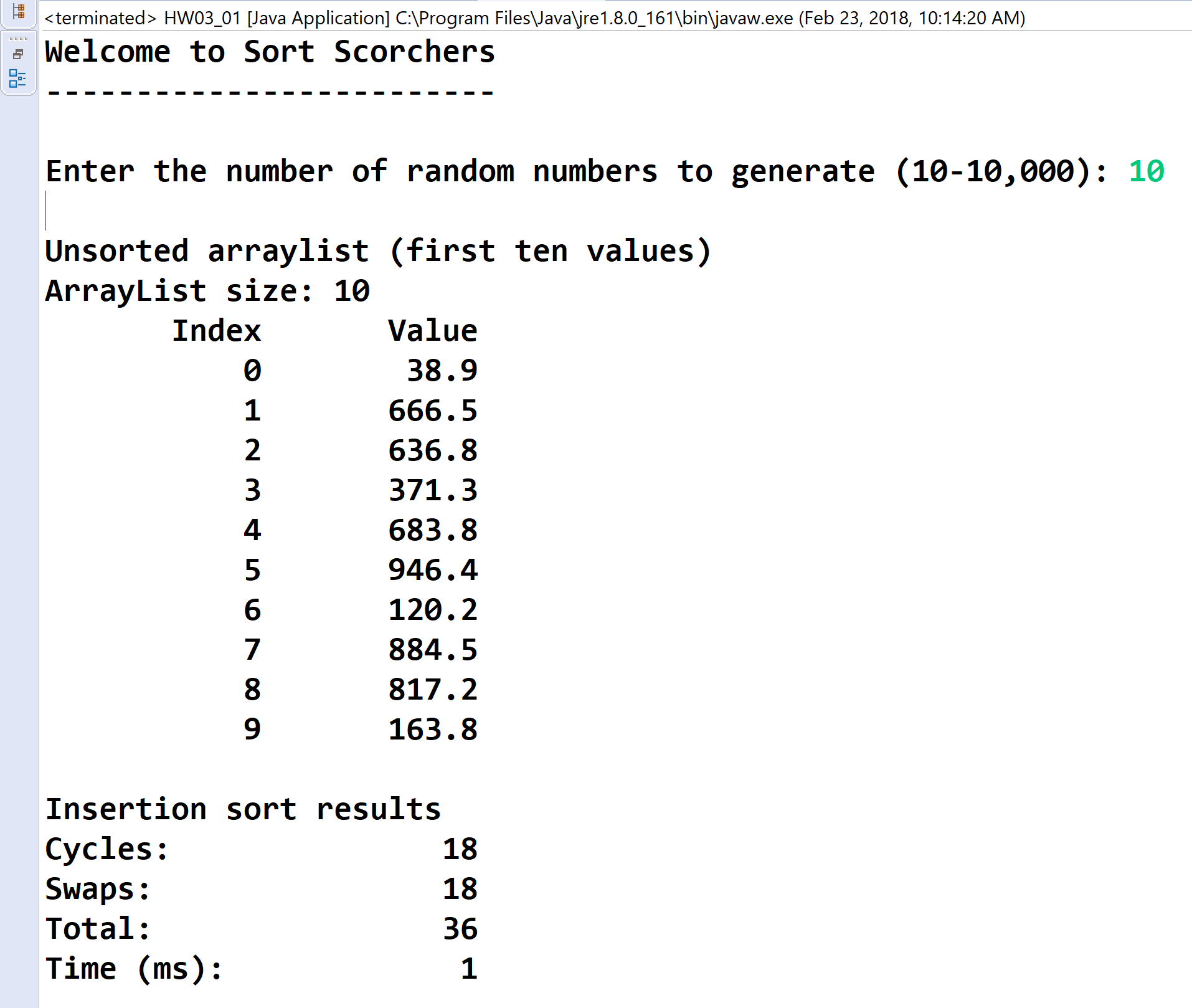
**keyboard.close();**

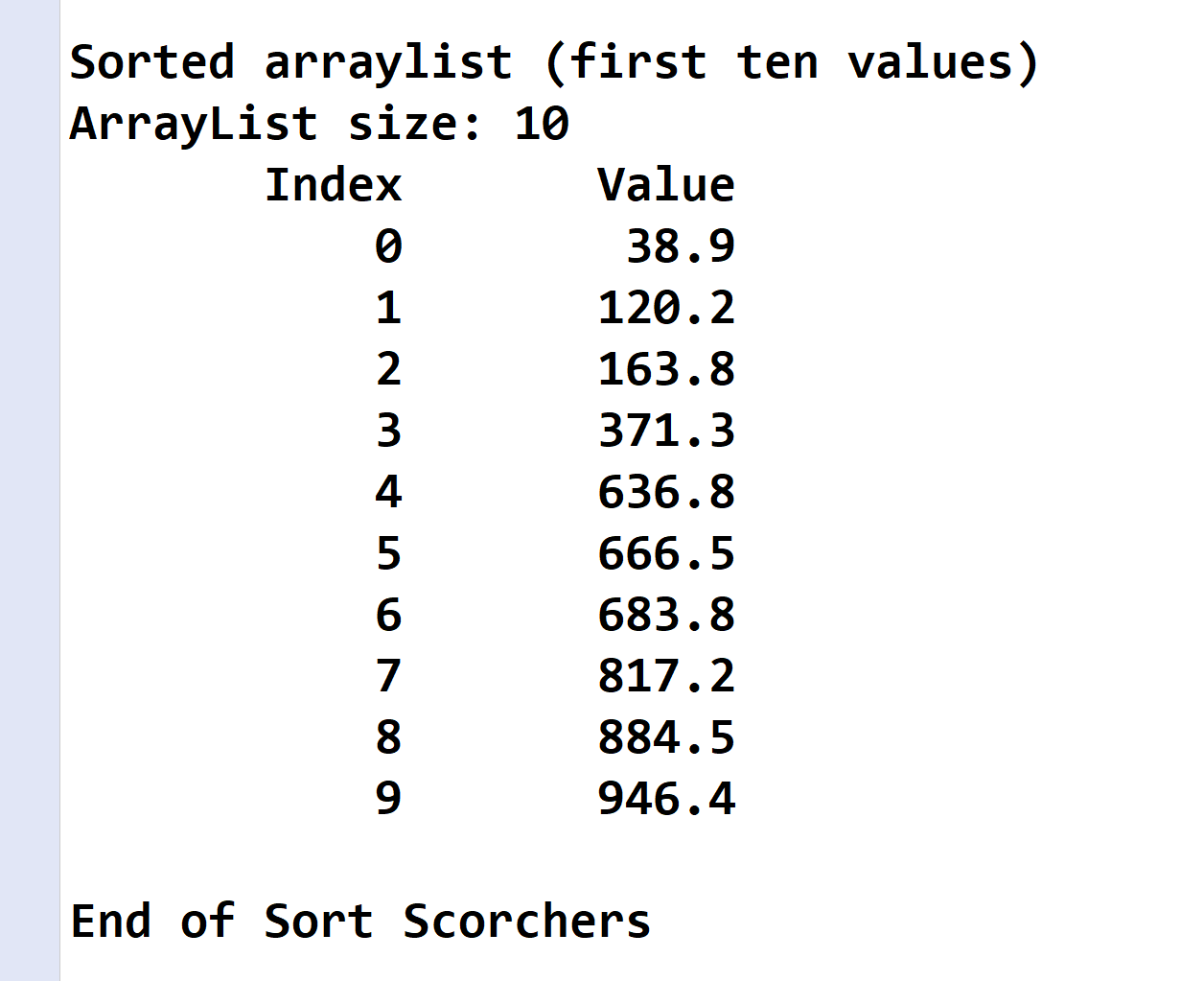
**// Show application close**

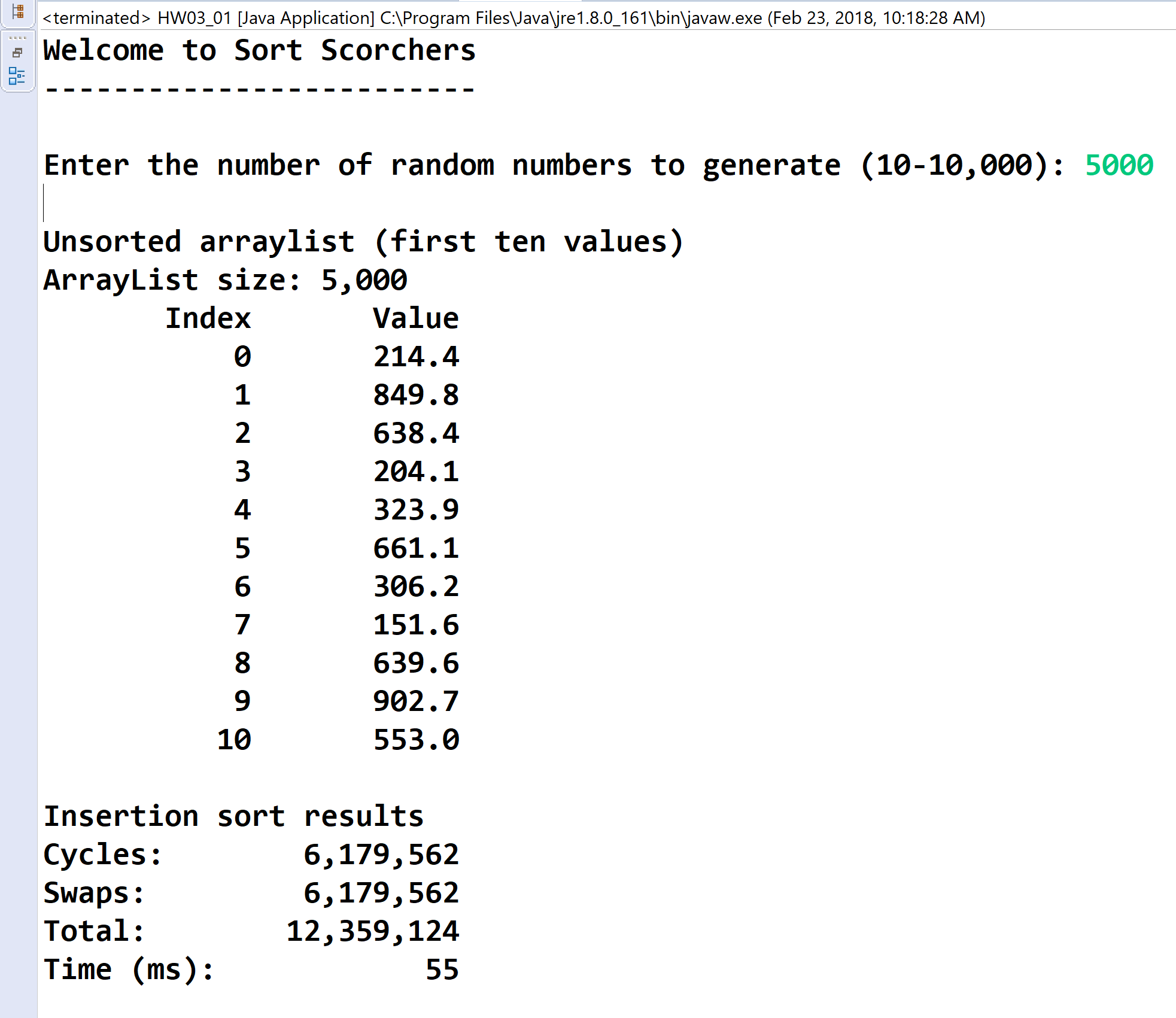
**System.out.println("\nEnd of Sort Scorchers");**

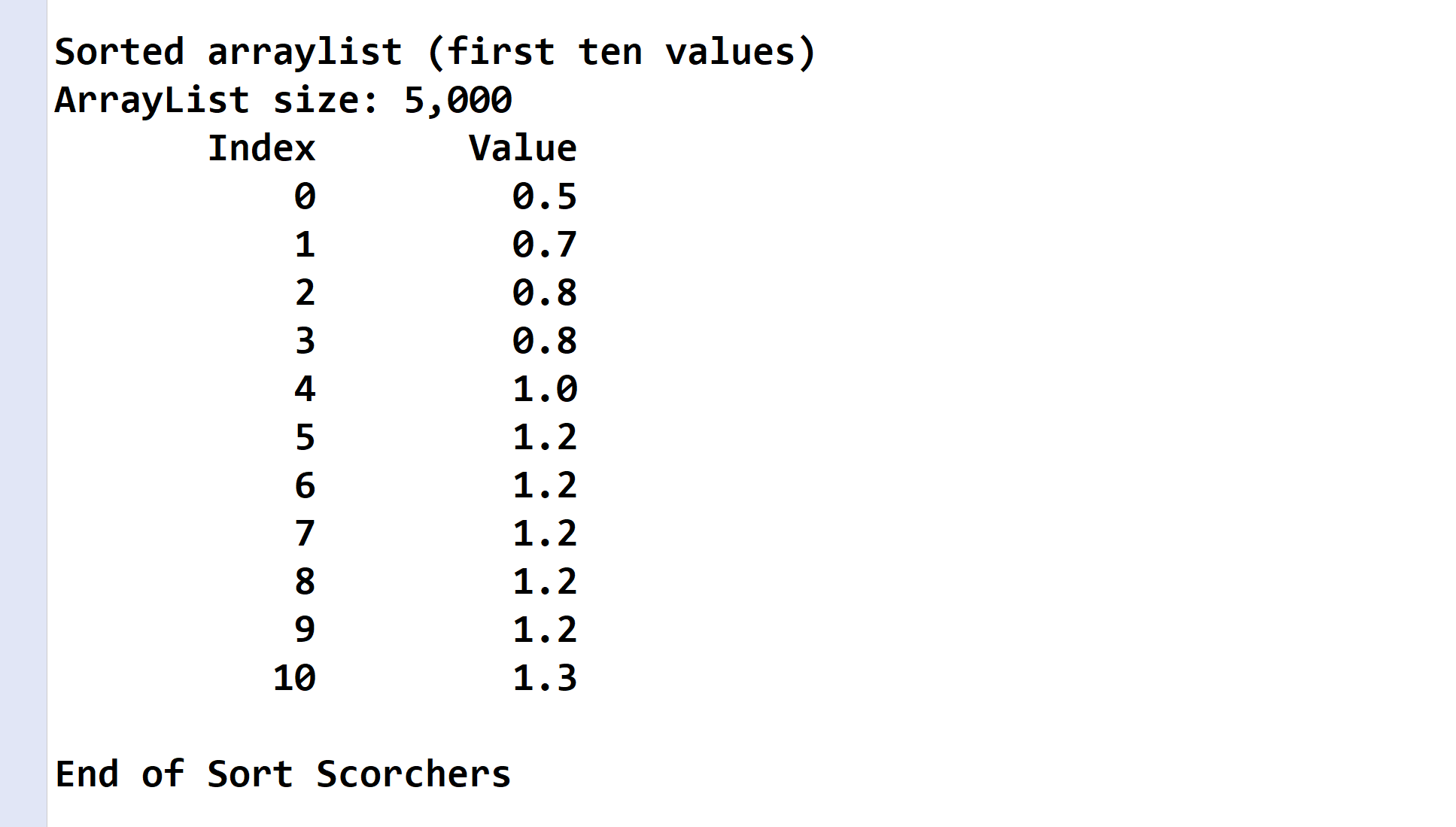
**}**

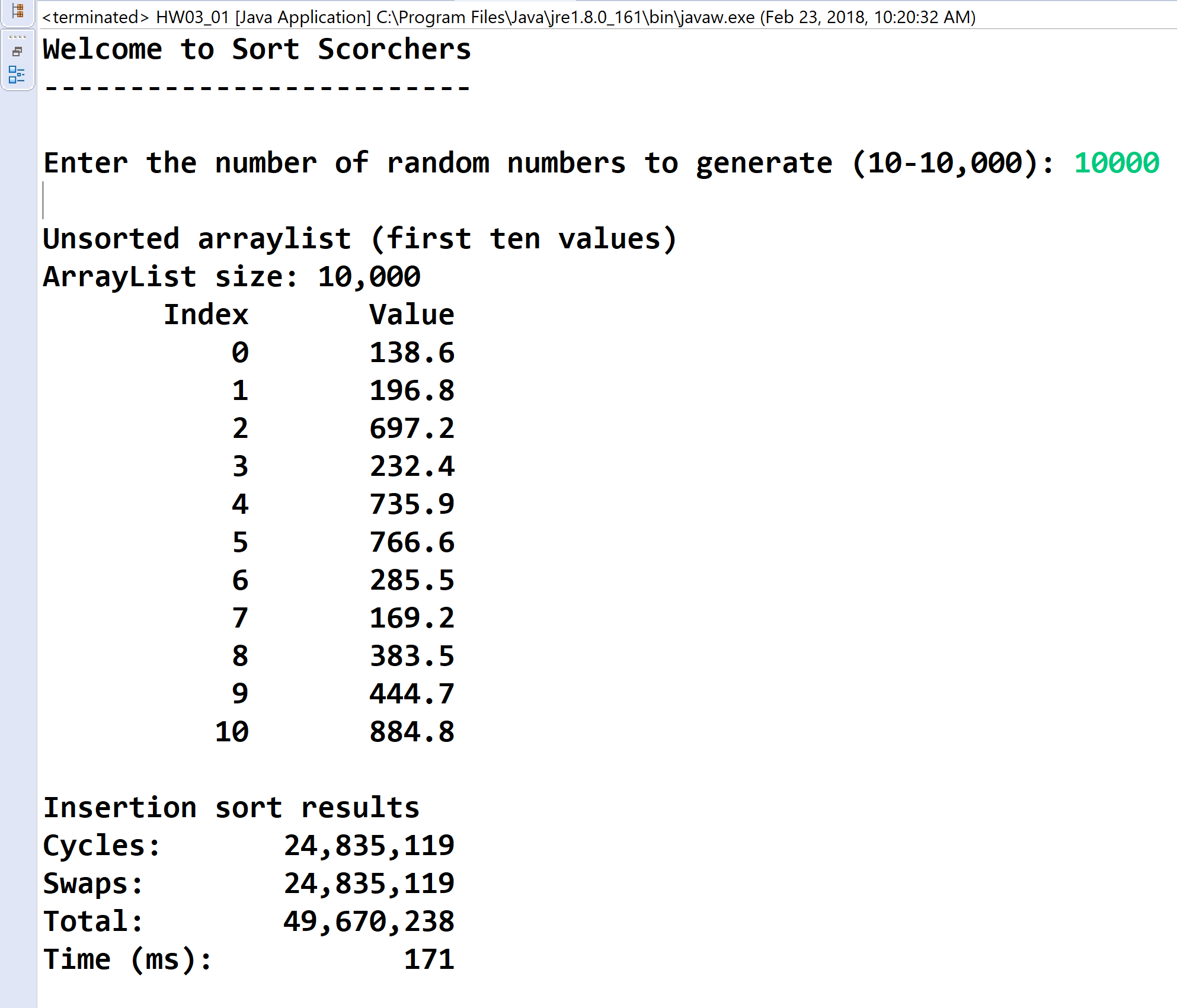
**}**

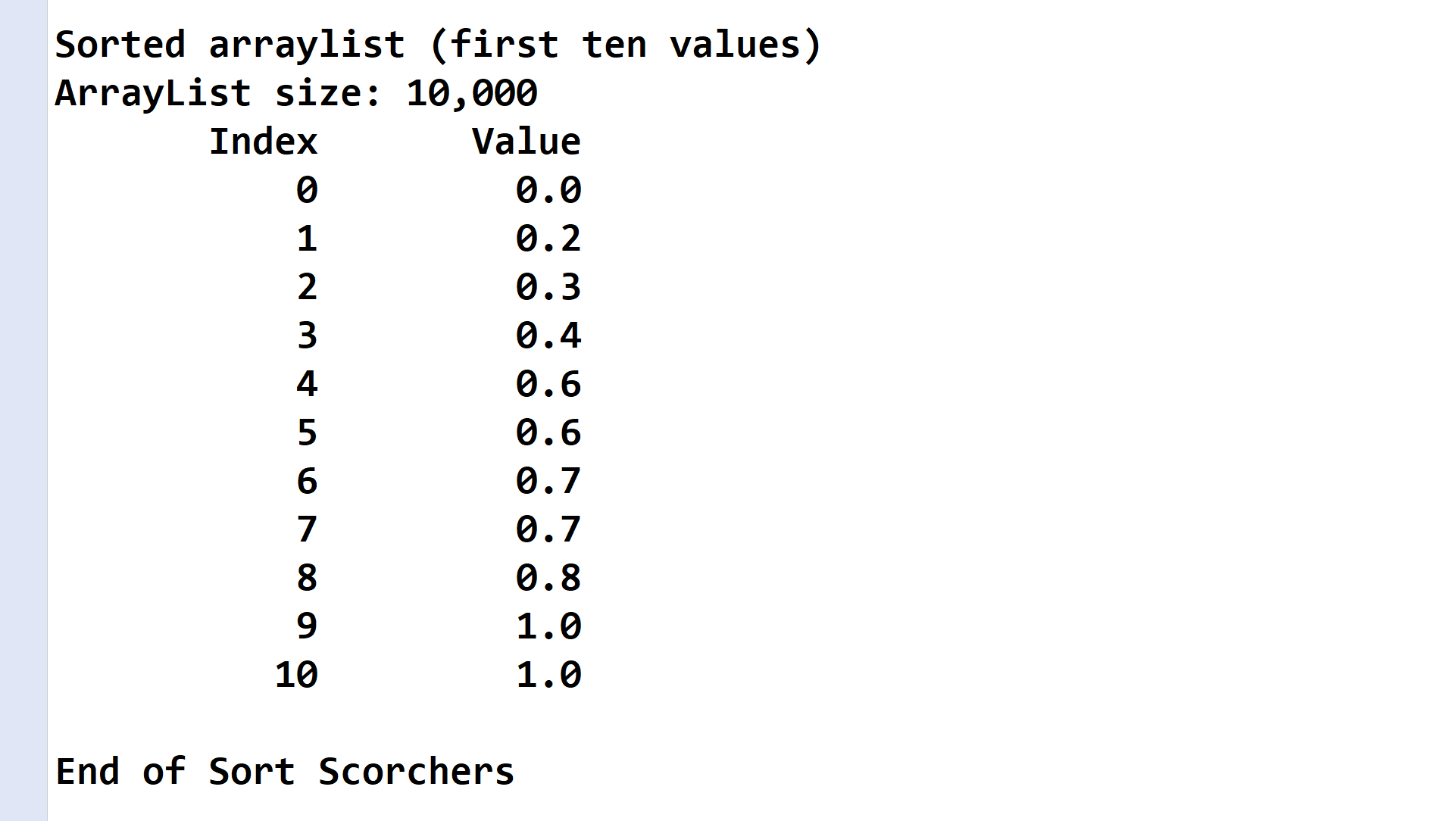




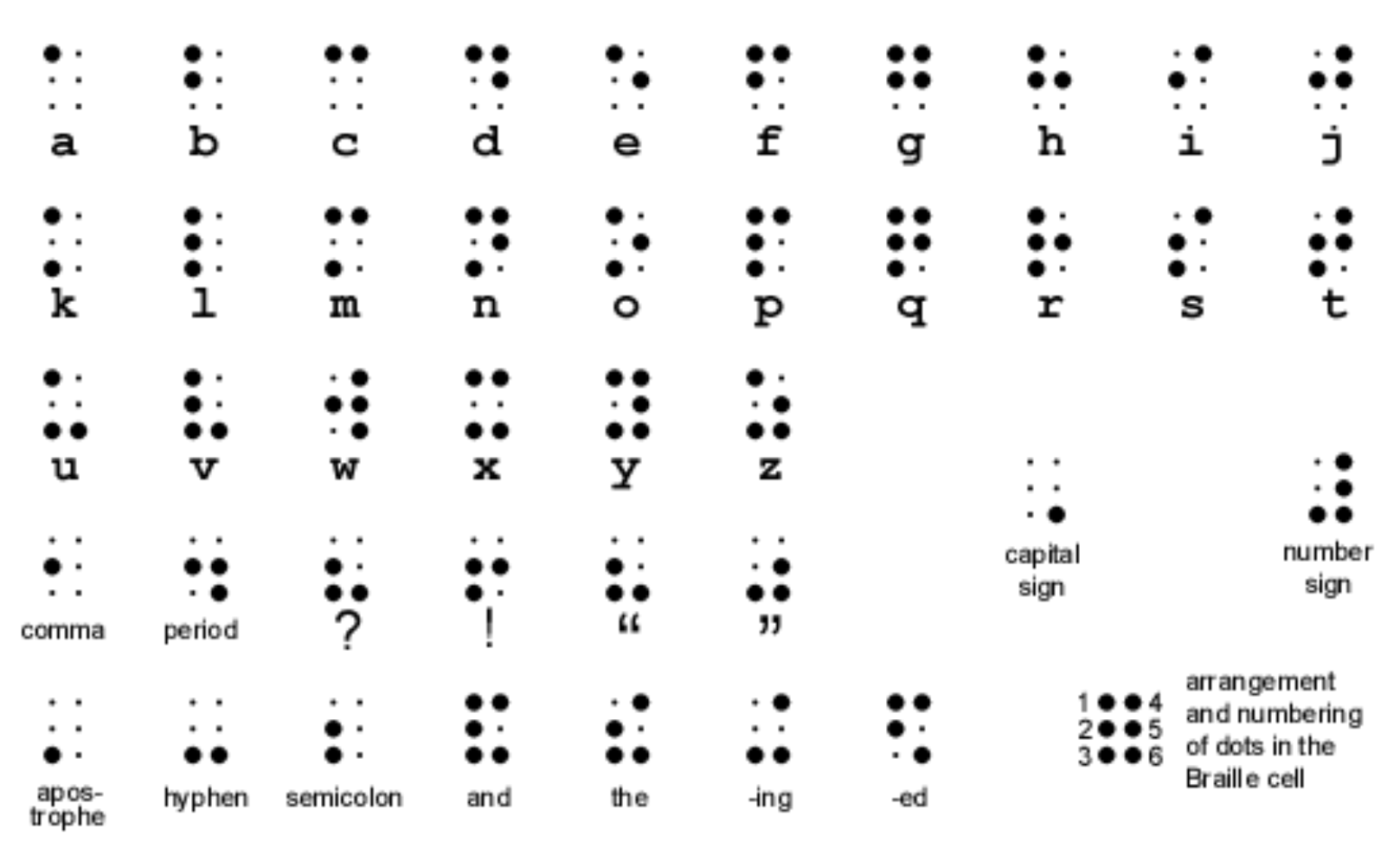








**2) [15 points]** You've been hired by *Braille Buddies* to write a Java console application that prints the equivalent Braille of the text entered by the user. Braille is a special language used by the blind to read text. It uses a 3x2 grid of dots to represent different characters. Here are the grid definitions for Grade 1 Braille:



To specify an upper case letter, the "capital sign" grid defined above precedes a lower case letter. Use the following three-dimensional array declaration to represent the Braille grid definitions. The first dimension is the letter, the second dimension is the row, and the third dimension is the column of a spot in a grid. A one (1) indicates a big spot while a zero (0) indicates a small spot:

final int[][][] BRAILLE\_GRIDS =

{

{{ 1, 0}, { 0, 0}, { 0, 0}}, // 'a' - 0

{{ 1, 0}, { 1, 0}, { 0, 0}}, // 'b' - 1

{{ 1, 1}, { 0, 0}, { 0, 0}}, // 'c' - 2

{{ 1, 1}, { 0, 1}, { 0, 0}}, // 'd' - 3

{{ 1, 0}, { 0, 1}, { 0, 0}}, // 'e' - 4

{{ 1, 1}, { 1, 0}, { 0, 0}}, // 'f' - 5

{{ 1, 1}, { 1, 1}, { 0, 0}}, // 'g' - 6

{{ 1, 0}, { 1, 1}, { 0, 0}}, // 'h' - 7

{{ 0, 1}, { 1, 0}, { 0, 0}}, // 'i' - 8

{{ 0, 1}, { 1, 1}, { 0, 0}}, // 'j' - 9

{{ 1, 0}, { 0, 0}, { 1, 0}}, // 'k' - 10

{{ 1, 0}, { 1, 0}, { 1, 0}}, // 'l' - 11

{{ 1, 1}, { 0, 0}, { 1, 0}}, // 'm' - 12

{{ 1, 1}, { 0, 1}, { 1, 0}}, // 'n' - 13

{{ 1, 0}, { 0, 1}, { 1, 0}}, // 'o' - 14

{{ 1, 1}, { 1, 0}, { 1, 0}}, // 'p' - 15

{{ 1, 1}, { 1, 1}, { 1, 0}}, // 'q' - 16

{{ 1, 0}, { 1, 1}, { 1, 0}}, // 'r' - 17

{{ 0, 1}, { 1, 0}, { 1, 0}}, // 's' - 18

{{ 0, 1}, { 1, 1}, { 1, 0}}, // 't' - 19

{{ 1, 0}, { 0, 0}, { 1, 1}}, // 'u' - 20

{{ 1, 0}, { 1, 0}, { 1, 1}}, // 'v' - 21

{{ 0, 1}, { 1, 1}, { 0, 1}}, // 'w' - 22

{{ 1, 1}, { 0, 0}, { 1, 1}}, // 'x' - 23

{{ 1, 1}, { 0, 1}, { 1, 1}}, // 'y' - 24

{{ 1, 0}, { 0, 1}, { 1, 1}}, // 'z' - 25

{{ 0, 0}, { 0, 0}, { 0, 1}} // upper case follows - 26

};

Use a validation loop to prompt for and get from the user a string that contains only upper and lower case letters and spaces. If an invalid string is entered, print an error message. If a valid string is entered, convert and display it in Braille. Print the following four lines for each string:

|  |  |
| --- | --- |
| Line | Output |
| Character | ● For an lower case letter, print the letter.  ● For an upper case letter, print "UP" and the corresponding lower case letter.  ● For the space character, print a space. |
| Rows 1-3 | ● For lower case letters, print its grid.  ● For upper case letters, prints two grids: the "capital sign" grid and the corresponding lower case letter grid.  ● For the space character, print spaces. |

To access the correct grid in array BRAILLE\_GRIDS, convert a lower case letter to value between 0 and 25 (the first dimension) using the fact that each character is assigned a unique integer. Here is a run with a sample string:

Enter a string (letters and spaces only) to convert to Braille (q to exit): Hello World

Character: UP h e l l o UP w o r l d

Row 1: 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 1 1 0 1 0 1 0 1 1

Row 2: 0 0 1 1 0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 1 0 0 1

Row 3: 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 0 1 0 0 0

Enter a string (letters and spaces only) to convert to Braille (q to exit):

Continue to prompt the user for strings until they enter “q”. Use these strings for the last three inputs:

|  |  |
| --- | --- |
| Input | Text |
| 1 | The quick Fox |
| 2 | Jumps over |
| 3 | the Lazy Dog |

**//======================================================================**

**//**

**// Title: Braille Buddies**

**// Course: CSC 3020**

**// Homework: 3-2**

**// Author: Dan Ouellette**

**// Date: 21 February 2018**

**// Description:**

**// This Java console application prints the equivalent Braille of the**

**// text entered by the user. Braille is a special language used by the**

**// blind to read text. It uses a 3x2 grid of dots to represent**

**// different characters. To specify an upper case letter, the "capital**

**// sign" grid precedes a lower case letter. The application uses a**

**// three-dimensional array to represent the Braille grid definitions.**

**// The first dimension is the letter, the second dimension is the row,**

**// and the third dimension is the column of a spot in a grid. It uses**

**// Use a validation loop to prompt for and get from the user a string**

**// that contains only upper and lower case letters and spaces. If an**

**// invalid string is entered, it prints an error message. If a valid**

**// string is entered, it converts and displays it in Braille.**

**//**

**//======================================================================**

**package wsu.HW03\_02;**

**// Import classes**

**import java.util.Scanner;**

**//======================================================================**

**//class HW03\_02**

**//======================================================================**

**public class HW03\_02**

**{**

**//------------------------------------------------------------------**

**// validString**

**//------------------------------------------------------------------**

**private static Boolean validString(String s)**

**{**

**// Declare variables**

**int i;**

**// Loop to check string**

**i = 0;**

**while (i < s.length() &&**

**(Character.isLetter(s.charAt(i)) || s.charAt(i) == ' '))**

**i = i + 1;**

**// Test if invalid character found**

**if (i < s.length())**

**return false;**

**else**

**return true;**

**}**

**//------------------------------------------------------------------**

**// main**

**//------------------------------------------------------------------**

**public static void main (String[] args)**

**{**

**// Declare constants**

**final int GRID\_ROWS = 3;**

**final int INDEX\_OFFSET = 97;**

**final String COLFMTD = "%2d";**

**final String COLFMTS1 = "%-10s";**

**final String COLFMTS2 = "%6s";**

**final String COLFMTS3 = "%2s";**

**final int[][][] BRAILLE\_GRIDS =**

**{**

**{{ 1, 0}, { 0, 0}, { 0, 0}}, // 'a' - 0**

**{{ 1, 0}, { 1, 0}, { 0, 0}}, // 'b' - 1**

**{{ 1, 1}, { 0, 0}, { 0, 0}}, // 'c' - 2**

**{{ 1, 1}, { 0, 1}, { 0, 0}}, // 'd' - 3**

**{{ 1, 0}, { 0, 1}, { 0, 0}}, // 'e' - 4**

**{{ 1, 1}, { 1, 0}, { 0, 0}}, // 'f' - 5**

**{{ 1, 1}, { 1, 1}, { 0, 0}}, // 'g' - 6**

**{{ 1, 0}, { 1, 1}, { 0, 0}}, // 'h' - 7**

**{{ 0, 1}, { 1, 0}, { 0, 0}}, // 'i' - 8**

**{{ 0, 1}, { 1, 1}, { 0, 0}}, // 'j' - 9**

**{{ 1, 0}, { 0, 0}, { 1, 0}}, // 'k' - 10**

**{{ 1, 0}, { 1, 0}, { 1, 0}}, // 'l' - 11**

**{{ 1, 1}, { 0, 0}, { 1, 0}}, // 'm' - 12**

**{{ 1, 1}, { 0, 1}, { 1, 0}}, // 'n' - 13**

**{{ 1, 0}, { 0, 1}, { 1, 0}}, // 'o' - 14**

**{{ 1, 1}, { 1, 0}, { 1, 0}}, // 'p' - 15**

**{{ 1, 1}, { 1, 1}, { 1, 0}}, // 'q' - 16**

**{{ 1, 0}, { 1, 1}, { 1, 0}}, // 'r' - 17**

**{{ 0, 1}, { 1, 0}, { 1, 0}}, // 's' - 18**

**{{ 0, 1}, { 1, 1}, { 1, 0}}, // 't' - 19**

**{{ 1, 0}, { 0, 0}, { 1, 1}}, // 'u' - 20**

**{{ 1, 0}, { 1, 0}, { 1, 1}}, // 'v' - 21**

**{{ 0, 1}, { 1, 1}, { 0, 1}}, // 'w' - 22**

**{{ 1, 1}, { 0, 0}, { 1, 1}}, // 'x' - 23**

**{{ 1, 1}, { 0, 1}, { 1, 1}}, // 'y' - 24**

**{{ 1, 0}, { 0, 1}, { 1, 1}}, // 'z' - 25**

**{{ 0, 0}, { 0, 0}, { 0, 1}} // upper case follows - 26**

**};**

**// Declare variables**

**Scanner keyboard = new Scanner(System.in);**

**String s;**

**// Show application header**

**System.out.println("Braille Buddies");**

**System.out.println("---------------\n");**

**// Loop to prompt for and get strings**

**System.out.print("Enter a string (letters and spaces only) " +**

**"to convert to Braille (q to exit): ");**

**s = keyboard.nextLine();**

**while (!s.equalsIgnoreCase("q"))**

**{**

**// Test if valid string**

**if (!validString(s))**

**System.out.println("Error: the string contains " +**

**"characters other than letters and spaces.");**

**else**

**{**

**// Loop to print character row**

**System.out.println();**

**System.out.printf(COLFMTS1, "Character:");**

**for (int i = 0; i < s.length(); i++)**

**{**

**char c = s.charAt(i);**

**if (Character.isUpperCase(c))**

**{**

**System.out.printf(COLFMTS2, "UP");**

**c = Character.toLowerCase(c);**

**}**

**System.out.printf(COLFMTS2, c);**

**}**

**System.out.println();**

**// Loop to convert to Braille rows**

**for (int row = 0; row < GRID\_ROWS; row++)**

**{**

**// Loop to convert Braille row**

**System.out.printf(COLFMTS1, "Row " + (row + 1) + ":");**

**for (int k = 0; k < s.length(); k++)**

**{**

**char c = s.charAt(k);**

**if (Character.isUpperCase(c))**

**{**

**System.out.printf(COLFMTS3, " ");**

**System.out.printf(COLFMTD,**

**BRAILLE\_GRIDS[26][row][0]);**

**System.out.printf(COLFMTD,**

**BRAILLE\_GRIDS[26][row][1]);**

**c = Character.toLowerCase(c);**

**}**

**// Test if character is space**

**if (Character.isWhitespace(c))**

**System.out.printf(COLFMTS2, " ");**

**else**

**{**

**int plane = (int) c - INDEX\_OFFSET;**

**System.out.printf(COLFMTS3, " ");**

**System.out.printf(COLFMTD,**

**BRAILLE\_GRIDS[plane][row][0]);**

**System.out.printf(COLFMTD,**

**BRAILLE\_GRIDS[plane][row][1]);**

**}**

**}**

**System.out.println();**

**}**

**}**

**// Prompt for and get next string**

**System.out.print("\nEnter a string (letters and spaces " +**

**"only) to convert to Braille (q to exit): ");**

**s = keyboard.nextLine();**

**}**

**// Close keyboard**

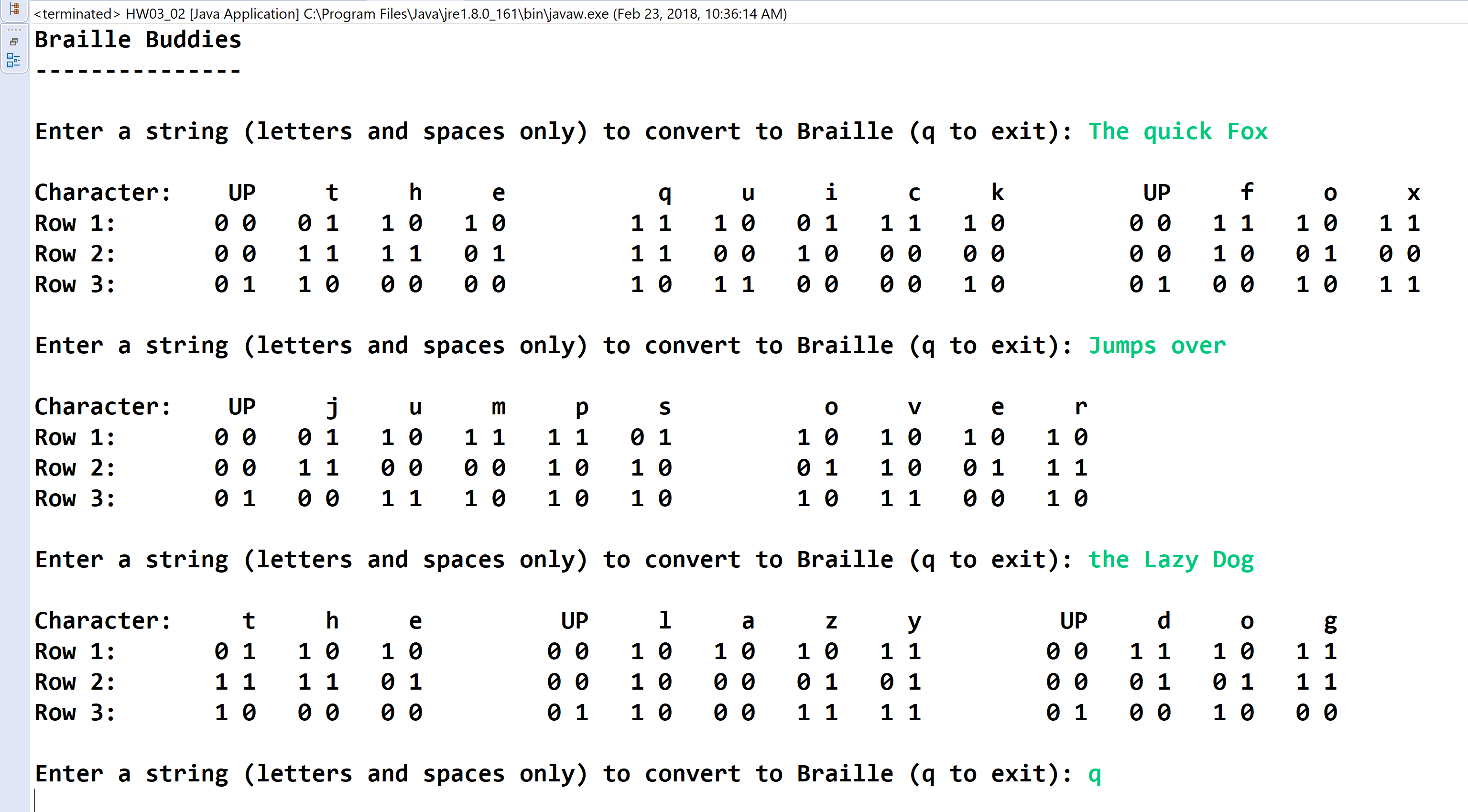
**keyboard.close();**

**// Show application close**

**System.out.println("\nEnd of Braille Buddies");**

**}**

**}**





\* **Copying-and-pasting application code to a Word document**

1) From the program editor window, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

\*\* **Copying-and-pasting application output to a Word document**

1) From the Eclipse main screen, maximize the Console window.

2) From the Console window, press **ALT-PrintScreen**.

3) From within the Word document, press **CTRL-V**.