



CSA 1017

Data Structures and Algorithms 1

Assignment

Mark Said Camilleri
B.Sc. (Hons) (Computing Science)
University of Malta

Friday 4th May 2016

Contents

	Page
Statement of Completion	2
1 Arabic to Roman Numeral Converter	3
1.1 The source code for Task 1	4
2 Reverse Polish Notation Evaluator	6
2.1 The source code for Task 2	7
2.1.1 Stack Class	7
2.1.2 Question 2 Main Class	8
3 Prime Numbers	12
3.1 Prime Number Checker using divisibility check	12
3.1.1 The source code for 3.1	13
3.2 Prime Number Checker using The Sieve of Eratosthenes	15
3.2.1 The source code for 3.2	16
4 Shell Sort	19
5	20
6	21
7	22
8	23
9	24
References	24

Statement of Completion

The questions below were the ones that have been attempted:

Question 1 This question has been successfully completed.

Question 2 This question has been successfully completed.

Question 3 This question has been successfully completed.

Question 4 This question has been successfully completed.

Question 5 This question has been successfully completed.

Question 6 This question has been successfully completed.

Question 7 This question has been successfully completed.

Question 8 This question has been successfully completed.

Question 9 This question has been successfully completed.

Signature

Date

Task 1

Arabic to Roman Numeral Converter

Input	Expected Output	Actual Output
1	I	I
2	II	II
4	IV	IV
5	V	V
6	VI	IV
9	IX	IX
10	X	X
20	XX	XX
40	XL	XL
49	XLIX	XLIX
50	L	L
60	LX	LX
90	XC	XC
99	XCIX	XCIX
100	C	C
150	CL	CL
400	CD	CD
499	CDXCIX	CDXCIX
500	D	D
600	DC	DC
900	CM	CM
999	CMXCIX	CMXCIX
1000	M	M
1024	MXXIV	MXXIV
2000	The input was not a valid number between 1 and 1024 Please try again and enter a number to convert:	The input was not a valid number between 1 and 1024 Please try again and enter a number to convert:
abc	The input was not a valid number between 1 and 1024 Please try again and enter a number to convert:	The input was not a valid number between 1 and 1024 Please try again and enter a number to convert:

1.1 The source code for Task 1

```
1  /**
2   * @author Mark Said Camilleri
3   * @version 20160509
4   */
5
6
7  import java.util.InputMismatchException;
8  import java.util.Scanner;
9
10
11 public class Question1 {
12
13     public static void main(String args[]) {
14
15         //Initialize Scanner object
16         Scanner in = new Scanner(System.in);
17         in.useDelimiter("\n");
18
19         /*===== WELCOME MESSAGE TO USER
20         =====*/
21         System.out.println("
22         -----");
23         System.out.println("|      CSA 1017 - Data Structures and
24 Algorithms 1      |");
25         System.out.println("
26         |-----|");
27         System.out.println("|      Submission by Mark Said Camilleri
28         |");
29         System.out.println("|      Task 1: Arabic to Roman Numeral
30 Converter      |");
31         System.out.println("
32         |-----|");
33         System.out.print("| Please enter a number between 1 and 1024: ")
34 ;
35
36         int toConvert = 0; //value to be converted.
37         boolean isError; // temporary boolean value used for error
38         checking of the input.
39         do {
40             isError = false;
41             try {
42                 toConvert = in.nextInt();
43             } catch (InputMismatchException e) {
44                 isError = true;
45                 in.next(); //To clear the buffer
46             }
47             /*===== Makes sure input is a number is between 1 and
48 1024 =====*/
49             if (isError || toConvert < 1 || toConvert > 1024) {
50                 /*===== OUTPUT ERROR MESSAGE TO THE USER
51                 =====*/
52                 System.out.println("
53                 |-----|");
54                 System.out.println("|The input was not a valid number
```

```
        between 1 and 1024|");
44         System.out.print("|Please try again and enter a number
to convert: ");
45     }
46     } while (isError || toConvert < 1 || toConvert > 1024);
47
48     System.out.printf("| %4d = %-24s in Roman Numerals |", toConvert
, convert(toConvert));
49
50 }
51
52 /**
53  * Takes an int decimal value and outputs a string of the same value
in Roman Numerals.
54  *
55  * @param toConvert the decimal value to be converted to Roman
Numerals
56  * @return The roman numeral equivalent of the input parameter
57  */
58 private static String convert(int toConvert) {
59
60     //Defining the decimal and roman counterparts
61     final int dec[] = {1, 4, 5, 9, 10, 40, 50, 90, 100, 400, 500,
900, 1000};
62     final String rom[] = {"I", "IV", "V", "IX", "X", "XL", "L", "XC"
, "C", "CD", "D", "CM", "M"};
63
64     /* Begins by checking the input parameter against the largest
roman numeral/numeral pair.
65     * and works it's way down to the unit numeral.
66     */
67     for (int i = dec.length - 1; i >= 0; i--) {
68         //If the value is larger, then the output is concatenated
with the output of the difference.
69         if (toConvert >= dec[i])
70             return rom[i] + convert(toConvert - dec[i]);
71     }
72     return ""; //What to return at 0, the base case.
73 }
74 }
```

Task 2

Reverse Polish Notation Evaluator

Input	Expected Output	Actual Output
4 5 +	9.0	9.0
12 6 -	6.0	6.0
3 2 /	1.5	1.5
7 2 *	14.0	14.0
23 85 + 92 *	9936.0	9936.0
43.5 3.2 - 4.5 * 3.24 +	184.59	184.59
34.8 62.11 * -76 /	-28.43984211	-28.439842105263157
282 -56 * 1024 - -55.6 *	934969.6	934969.6
Test	Your expression contained invalid characters. For input string "T" Your expression is invalid. Evaluation failed	Your expression contained invalid characters. For input string "T" Your expression is invalid. Evaluation failed
1 +	Stack is Empty. Your expression is invalid. Evaluation failed.	Stack is Empty. Your expression is invalid. Evaluation failed.
3 64 6 +	The stack has not been emptied. There are too many operands in your expression. Your expression is invalid. Evaluation failed.	The stack has not been emptied. There are too many operands in your expression. Your expression is invalid. Evaluation failed.
4 0 /	Infinity	Infinity

2.1 The source code for Task 2

2.1.1 Stack Class

```
1 import java.util.ArrayList;
2 import java.util.Collection;
3 import java.util.EmptyStackException;
4
5 /**
6  * Created by mark on 14/02/16.
7  *
8  * A stack implemented as an ArrayList to have it dynamically increase
9  * its size.
10 */
11 public class Stack<E> extends ArrayList {
12
13     /**
14      * Default constructor. Calls the ArrayList default constructor
15      */
16     public Stack() {
17         super();
18     }
19
20     /**
21      * Initialised a stack with the contents of the Collection in the
22      * parameter.
23      *
24      * @param c the contents to initialise the stack with.
25      */
26     public Stack(Collection<? extends E> c) {
27         super(c);
28     }
29
30     /**
31      * Pushes the data onto the stack
32      *
33      * @param data data to be pushed on the stack
34      * @throws IndexOutOfBoundsException if not successfully pushed
35      */
36     public void push(E data) throws IndexOutOfBoundsException {
37         int prevSize = this.size();
38         this.add(data);
39
40         //Condition to check if the data has been successfully added.
41         if (!(this.size() == prevSize + 1))
42             throw new IndexOutOfBoundsException("Failed to push to stack");
43     }
44
45     /**
46      * Pops the topmost item from the stack.
47      *
48      * @return the data from the top of the stack is not empty.
49      * @throws EmptyStackException if the stack is empty.
50      */
51     public E pop() throws ArrayIndexOutOfBoundsException {
52         if (this.size() == 0) throw new EmptyStackException();
53     }
54 }
```



```
50         else return (E) this.remove(this.size() - 1);
51     }
52
53     /**
54      * Returns the data at the top of the stack without popping it.
55      * @return the data if the stack is not empty. null if it is empty.
56      */
57     public E peek() {
58         if (this.size() == 0) return null;
59         else return (E) this.get(this.size() - 1);
60     }
61
62     /**
63      * Returns a string representation of the contents of the stack.
64      * @overrides toString() in class AbstractCollection<E>
65      * @return The string representation of the ArrayList if not empty.
66      * "Stack is empty" if it is empty.
67      */
68     public String toString() {
69         if (this.size() == 0) return "Stack is empty";
70         else return super.toString();
71     }
72 }
```

2.1.2 Question 2 Main Class

```
1 import java.util.EmptyStackException;
2 import java.util.Scanner;
3
4 /**
5  * Created by mark on 09/02/16.
6  * This answer assumes the RPN input is correct.
7  */
8 public class Question2 {
9     public static void main(String args[]) {
10
11         //Initialize Scanner object
12         Scanner in = new Scanner(System.in).useDelimiter("\n");
13
14         /*===== WELCOME MESSAGE TO USER
15         =====*/
16         System.out.println("
17         -----");
18         System.out.println("|      CSA 1017 - Data Structures and
19         Algorithms 1      |");
20         System.out.println("
21         |-----|");
22         System.out.println("|      Submission by Mark Said Camilleri
23         |");
24         System.out.println("|      Task 2: Reverse Polish Notation
25         evaluator      |");
26         System.out.println("
27         |-----|");
28         System.out.println("|      Note: This program can only do +,-,*
29         and /      |");
30         System.out.print("| Please enter an expression to evaluate: ");
31         //prompt for user input. Assumes it is rpn.
32     }
```

```
23
24     /* Initialized a stack object (using the stack defined here).
25      * Note, no importing of the Stack class.
26      */
27     Stack<Double> nums = new Stack<>();
28
29     //Reads user input. Must be a valid RPN expression.
30     String expression = in.next();
31     System.out.println("
32 |-----|");
33
34     System.out.println("| Contents of the stack at each step:
35 |"); //some message to user.
36
37     boolean exceptionRaised = false; //used for error checking.
38     try {
39         //Iterates through the string inputted by the user.
40         for (int i = 0; i < expression.length(); i++) {
41             char cChar = expression.charAt(i);
42
43             /* If the current character is a space,
44              * nothing needs to be done.
45              */
46             if (Character.isWhitespace(cChar)) continue;
47
48             /* If it's a '+', then 2 numbers are popped, added and
49              * the answer is pushed onto the stack.
50              */
51             else if (cChar == '+') {
52                 double num1 = nums.pop();
53                 double num2 = nums.pop();
54
55                 nums.push(num2 + num1);
56             }
57             /* If it's a '-', then 2 numbers are popped,
58              * subtracted and the answer is pushed onto the stack.
59              * The conjunction is to make sure that it's not
60              * detecting a negative number. The disjunction is
61              * true if the '-' is at the end of the string or
62              * there is a space after it. These both make sure
63              * that the '-' does not belong to a negative number
64              */
65             else if (cChar == '-' && (i == expression.length() - 1
66 || Character.isWhitespace(expression.charAt(i + 1)))) {
67                 double num1 = nums.pop();
68                 double num2 = nums.pop();
69
70                 nums.push(num2 - num1);
71             }
72             /* If it's a '*', then 2 numbers are popped,
73              * multiplied and the answer is pushed onto the stack.
74              */
75             else if (cChar == '*') {
76                 double num1 = nums.pop();
77                 double num2 = nums.pop();
78
79                 nums.push(num2 * num1);
80             }
81             /* If it's a '/', then 2 numbers are popped,
```

```

78         * divided and the answer is pushed onto the stack.
79     */
80     else if (cChar == '/') {
81         double num1 = nums.pop();
82         double num2 = nums.pop();
83
84         nums.push(num2 / num1);
85     }
86     /*
87     * Otherwise, assuming it's inputted correctly, the
88     * character must be a number. In which case it is
89     * converted to a double (allowing for any real number
90     * to be inputted) and pushed onto the stack.
91     */
92     else {
93         int start = i++;
94         while (Character.isDigit(expression.charAt(i)) ||
95 expression.charAt(i) == '.')
96             i++;
97
98         nums.push(Double.parseDouble(expression.substring(
99 start, i)));
100     }
101     System.out.printf("| %-49s |\n", nums.toString());
102 }
103 if (nums.size() != 1) {
104     System.out.println("| The stack has not been emptied.
105 There are too      |");
106     System.out.println("| many operands in your expression.
107 |");
108     exceptionRaised = true;
109 }
110 } catch (IndexOutOfBoundsException ioobe) {
111     System.out.printf("| %-49s |\n", ioobe.getMessage());
112     exceptionRaised = true;
113 } catch (EmptyStackException ese) {
114     System.out.println("| Stack is Empty.
115 |");
116     exceptionRaised = true;
117 } catch (NumberFormatException nfe) {
118     System.out.println("| Your expression contained invalid
119 characters.      |");
120     System.out.printf("| %-49s |\n", nfe.getMessage());
121     exceptionRaised = true;
122 } finally {
123     if (exceptionRaised) {
124         System.out.println("| Your expression is invalid.
125 Evaluation failed.  |");
126         System.out.println("
127 ");
128         System.exit(1);
129     }
130 }
131 }
132 /*When the above iteration is complete, there should only be
133 * one item on the stack which is the answer.
134 */

```

```
127         System.out.println("
| -----|");
128         System.out.printf("| Answer of Evaluation = %-26s |\n", nums.pop
().toString());
129         System.out.println("

");
130     }
131 }
```

Task 3

Prime Numbers

3.1 Prime Number Checker using divisibility check

Input	Expected Output	Actual Output
1	1 is not a prime number	1 is not a prime number
2	2 is a prime number	2 is a prime number
3	3 is a prime number	3 is a prime number
4	4 is not a prime number	4 is not a prime number
-2	-2 is not a prime number	-2 is not a prime number
a 677	677 is a prime number	677 is a prime number
34939	34939 is a prime number	34939 is a prime number
188737204	188737204 is not a prime number	188737204 is not a prime number
381165334	381165334 is not a prime number	381165334 is not a prime number
947396057	947396057 is a prime number	947396057 is a prime number
9223372036854775807	9223372036854775807 is not a prime number	9223372036854775807 is not a prime number
-9223372036854775808	-9223372036854775808 is not a prime number	-9223372036854775808 is not a prime number
9223372036854775808	Your input was not accepted. Please restart the program and try again with a valid input.	Your input was not accepted. Please restart the program and try again with a valid input.
Test	Your input was not accepted. Please restart the program and try again with a valid input.	Your input was not accepted. Please restart the program and try again with a valid input.

3.1.1 The source code for 3.1

```

1 import java.util.InputMismatchException;
2 import java.util.Scanner;
3
4 /**
5  * @author Mark Said Camilleri
6  * @version 20160511
7  */
8
9 public class Question3a {
10
11     public static void main(String args[]) {
12
13         //Initialize Scanner object
14         Scanner in = new Scanner(System.in).useDelimiter("\n");
15
16         /*===== WELCOME MESSAGE TO USER
17 =====*/
18         System.out.println("
19 -----");
20         System.out.println("|      CSA 1017 - Data Structures and
21 Algorithms 1      |");
22         System.out.println("
23 -----");
24         System.out.println("|      Submission by Mark Said Camilleri
25 |");
26         System.out.println("|      Task 3.1: Prime Number Checker
27 |");
28         System.out.println("
29 -----");
30         System.out.print("| Please enter an integer to check: ");
31
32         try {
33             //The number to check whether or not it's prime
34             long prime = in.nextLong();
35
36             System.out.println("
37 -----");
38             if (isPrime(prime))
39                 System.out.printf("| %31d is a prime number |\n", prime)
40 ;
41             else
42                 System.out.printf("| %27d is not a prime number |\n",
43 prime);
44
45             System.out.println("
46 -----");
47
48         } catch (InputMismatchException ime) {
49             System.out.println("| Your input was not accepted. Please
50 restart the      |");
51             System.out.println("| program and try again with a valid
52 input.          |");
53             System.out.println("
54 -----");
55
56             System.exit(1);
57         }
58     }
59 }

```

```
41     }
42 }
43
44 /**
45  * This method checks the input parameter to see if it is a prime
46  * number or not.
47  *
48  * @param prime The number to check if it is prime
49  * @return true if the number is prime, false if it isn't
50  */
51 private static boolean isPrime(long prime) {
52     /*Firstly, if the number is 1, 0 or negative then it's not
53     * prime
54     */
55     if (prime <= 1) return false;
56     //If the number is 2, 2 is prime.
57     else if (prime == 2) return true;
58     //If the number is even and not 2, then it's not prime.
59     else if (prime % 2 == 0) return false;
60     /* Otherwise, divide this number by all odd numbers till the
61     * square root of the number. If one is divisible then it's
62     * not prime.
63     */
64     else {
65         double root = Math.sqrt(prime);
66         for (int i = 3; i <= root; i += 2) {
67             if (prime % i == 0) return false;
68         }
69         return true;
70     }
71 }
72 }
```

3.2 Prime Number Checker using The Sieve of Eratosthenes

For this an `int` variable had to be used, due to it being the length of an array. As a result the range of numbers that can be checked was smaller.

Input	Expected Output	Actual Output
1	1 is not a prime number	1 is not a prime number
2	2 is a prime number	2 is a prime number
3	3 is a prime number	3 is a prime number
4	4 is not a prime number	4 is not a prime number
-2	Sieve of Eratosthenes only works on +ve integers. Please restart the program and try again with a valid input	Sieve of Eratosthenes only works on +ve integers. Please restart the program and try again with a valid input
677	677 is a prime number	677 is a prime number
34939	34939 is a prime number	34939 is a prime number
188737204	188737204 is not a prime number	188737204 is not a prime number
381165334	381165334 is not a prime number	381165334 is not a prime number
947396057	947396057 is a prime number	947396057 is a prime number
2147483646	2147483646 is not a prime number	Requested array size exceeds VM limit Please restart the program and try again with a smaller input ¹
-2147483646	Sieve of Eratosthenes only works on +ve integers. Please restart the program and try again with a valid input	Sieve of Eratosthenes only works on +ve integers. Please restart the program and try again with a valid input
9223372036854775807	Your input was not accepted. Please restart the program and try again with a valid input.	Your input was not accepted. Please restart the program and try again with a valid input.
Test	Your input was not accepted. Please restart the program and try again with a valid input.	Your input was not accepted. Please restart the program and try again with a valid input.

¹This error depends on the amount of memory allocated for the program

3.2.1 The source code for 3.2

```

1 import java.util.Arrays;
2 import java.util.InputMismatchException;
3 import java.util.Scanner;
4
5 /**
6  * @author Mark Said Camilleri
7  * @version 20160511
8  */
9 public class Question3b {
10
11     public static void main(String args[]) {
12
13         //Initialize Scanner object
14         Scanner in = new Scanner(System.in).useDelimiter("\n");
15
16         /*===== WELCOME MESSAGE TO USER
17         =====*/
18         System.out.println("
19         -----");
20         System.out.println("|      CSA 1017 - Data Structures and
21 Algorithms 1      |");
22         System.out.println("
23         -----");
24         System.out.println("|      Submission by Mark Said Camilleri
25         |");
26         System.out.println("|      Task 3.2: Sieve of Eratosthenes
27         |");
28         System.out.println("
29         -----");
30         System.out.print("| Please enter a positive integer to check: ")
31 ;
32         try {
33             /*
34              * Value to check if it's prime. Note that this is now an int
35              * since the Sieve of Eratosthenes requires an array of this
36              * amount of elements.
37              */
38             int prime = in.nextInt();
39
40             System.out.println("
41             -----");
42             if (sieve(prime))
43                 System.out.printf("| %31d is a prime number |\n", prime)
44 ;
45             else
46                 System.out.printf("| %27d is not a prime number |\n",
47 prime);
48             System.out.println("
49
50 ");
51         } catch (InputMismatchException ime) {
52             System.out.println("| Your input was not accepted. Please
53 restart the      |");
54             System.out.println("| program and try again with a valid
55 input.          |");
56         }
57     }
58 }

```

```

41         System.out.println("
42     ");
43         System.exit(1);
44     } catch (IllegalArgumentException iae) {
45         System.out.printf("| %-49s |\n", iae.getMessage());
46         System.out.println("| Please restart the program and try
again with a |");
47         System.out.println("| valid input.
|");
48         System.out.println("
49     ");
50         System.exit(1);
51     } catch (OutOfMemoryError oome) {
52         System.out.printf("| %-49s |\n", oome.getMessage());
53         System.out.println("| Please restart the program and try
again with a |");
54         System.out.println("| smaller input.
|");
55         System.out.println("
56     ");
57         System.exit(1);
58     }
59 }
60
61 /**
62  * Runs the Sieve of Eratosthenes algorithm to check if the input
63  * is a prime number or not.
64  *
65  * @param prime The number to check if it is prime or not.
66  * @return true if the input is prime, false if it isn't.
67  * @throws IllegalArgumentException if the input is <= 0
68  */
69 private static boolean sieve(int prime) throws
IllegalArgumentException, OutOfMemoryError {
70     //Checks whether the input is valid or not
71     if (prime <= 0)
72         //Throws exception if input is not valid.
73         throw new IllegalArgumentException("Sieve of Eratosthenes
only works on +ve integers.");
74     else {
75         //Each ith element is true if i-1 is prime. False otherwise.
76         boolean[] nos = new boolean[prime];
77
78         //We begin by assuming all the values are prime.
79         Arrays.fill(nos, true);
80
81         //Then, 1 is crossed out since it is a square.
82         nos[0] = false;
83
84         /*
85          * After which each value is checked. i is assumed to be prime
86          * unless marked as not prime. Multiples of i are not prime.
87          * Therefore all the multiples of i are marked as not prime.
88          * This keeps going on until the inputted value is set to not
89          * prime or reached and still set to prime.

```

```
87         */
88         for (int i = 2; i <= prime; i++) {
89             //Checks if i is marked as prime.
90             if (nos[i - 1]) {
91                 //If it is, all multiples of i are marked as not
prime
92                 for (int j = i + i; j <= prime; j += i) {
93                     nos[j - 1] = false;
94                 }
95             }
96         /*
97          * If the inputted value is marked as prime, we can stop
98          * there since we found out whether it's prime or not.
99          * Otherwise, this program keeps going on till the end
100          * (i = prime).
101          */
102         if (!nos[prime - 1]) break;
103     }
104
105     return nos[prime - 1];
106 }
107 }
108 }
```

Task 4

Shell Sort

Task 5

Task 6

Task 7

Task 8

Task 9