CSA 1017
Data Structures and Algorithms 1
Assignment

Mark Said Camilleri

B.Sc. (Hons) (Computing Science)

University of Malta

Friday 4th May 2016

Contents

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

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:

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         System.out.println("
    --------------------------------------------------------");
21         System.out.println("|    CSA 1017 - Data Structures and
    Algorithms 1    |");
22         System.out.println("
    |------------------------------------------------------|");
23         System.out.println("|         Submission by Mark Said Camilleri
         |");
24         System.out.println("|     Task 1: Arabic to Roman Numeral
    Converter     |");
25         System.out.println("
    |------------------------------------------------------|");
26         System.out.print("| Please enter a number between 1 and 1024: ")
    ;
27
28
29         int toConvert = 0; //value to be converted.
30         boolean isError; // temporary boolean value used for error
    checking of the input.
31         do {
32             isError = false;
33             try {
34                 toConvert = in.nextInt();
35             } catch (InputMismatchException e) {
36                 isError = true;
37                 in.next(); //To clear the buffer
38             }
39             /*=========== Makes sure input is a number is between 1 and
    1024 =============*/
40             if (isError || toConvert < 1 || toConvert > 1024) {
41                 /*=================== OUTPUT ERROR MESSAGE TO THE USER
    =================*/
42                 System.out.println("
    |------------------------------------------------------|");
43                 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 ve 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 paramerer 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

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

The source code for Task 2

Task 2.1   Stack Class

```java
import java.util.ArrayList;
import java.util.Collection;
import java.util.EmptyStackException;

/**
 * Created by mark on 14/02/16.
 *
 * A stack implemented as an ArrayList to have it dynamically increase
   its size.
 */
public class Stack<E> extends ArrayList {

    /**
     * Default constructor. Calls the ArrayList default constructor
     */
    public Stack() {
        super();
    }

    /**
     * Initialised a stack with the contents of the Collection in the
    parameter.
     *
     * @param c the contents to initialise the stack with.
     */
    public Stack(Collection<? extends E> c) {
        super(c);
    }

    /**
     * Pushes the data onto the stack
     *
     * @param data data to be pushed on the stack
     * @throws IndexOutOfBoundsException if not sucessfully pushed
     */
    public void push(E data) throws IndexOutOfBoundsException {
        int prevSize = this.size();
        this.add(data);

        //Condition to check if the data has been successfully added.
        if (!(this.size() == prevSize + 1))
            throw new IndexOutOfBoundsException("Failed to push to stack
    ");
    }

    /**
     * Pops the topmost item from the stack.
     * @return the data from the top of the stack is not empty.
     * @throws EmptyStackException if the stack is empty.
     */
    public E pop() throws ArrayIndexOutOfBoundsException {
        if (this.size() == 0) throw new EmptyStackException();
        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.
      "Stack is empty" if it is empty.
66         */
67        public String toString() {
68            if (this.size() == 0) return "Stack is empty";
69            else return super.toString();
70        }
71 }
```

Task 2.2   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          System.out.println("
      --------------------------------------------------");
16          System.out.println("|    CSA 1017 - Data Structures and
      Algorithms 1    |");
17          System.out.println("
      |--------------------------------------------------|");
18          System.out.println("|        Submission by Mark Said Camilleri
          |");
19          System.out.println("|     Task 2: Reverse Polish Notation
      evaluator      |");
20          System.out.println("
      |--------------------------------------------------|");
21          System.out.println("|     Note: This program can only do +,-,*
      and /      |");
22          System.out.print("| Please enter an expression to evaluate: ");
      //prompt for user input. Assumes correctness.
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           System.out.println("| Contents of the stack at each step:
             |");//some message to user.
33
34           boolean exceptionRaised = false; //used for error checking.
35           try {
36               //Iterates through the string inputted by the user.
37               for (int i = 0; i < expression.length(); i++) {
38                   char cChar = expression.charAt(i);
39
40                   /* If the current character is a space,
41                    * nothing needs to be done.
42                    */
43                   if (Character.isWhitespace(cChar)) continue;
44
45                   /* If it's a '+', then 2 numbers are popped, added and
46                    * the answer is pushed onto the stack.
47                    */
48                   else if (cChar == '+') {
49                       double num1 = nums.pop();
50                       double num2 = nums.pop();
51
52                       nums.push(num2 + num1);
53                   }
54                   /* If it's a '-', then 2 numbers are popped,
55                    * subtracted and the answer is pushed onto the stack.
56                    * The conjunction is to make sure that it's not
57                    * detecting a negative number. The disjunction is
58                    * true if the '-' is at the end of the string or
59                    * there is a space after it. These both make sure
60                    * that the '-' does not belong to a negative number
61                    */
62                   else if (cChar == '-' && (i == expression.length() - 1
    || Character.isWhitespace(expression.charAt(i + 1)))) {
63                       double num1 = nums.pop();
64                       double num2 = nums.pop();
65
66                       nums.push(num2 - num1);
67                   }
68                   /* If it's a '*', then 2 numbers are popped,
69                    * multiplied and the answer is pushed onto the stack.
70                    */
71                   else if (cChar == '*') {
72                       double num1 = nums.pop();
73                       double num2 = nums.pop();
74
75                       nums.push(num2 * num1);
76                   }
77                   /* 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 ot 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)) ||
       expression.charAt(i) == '.')
95                            i++;
96
97                        nums.push(Double.parseDouble(expression.substring(
       start, i)));
98                    }
99                    System.out.printf("| %-49s |\n", nums.toString());
100             }
101             if (nums.size() != 1) {
102                 System.out.println("| The stack has not been emptied.
       There are too    |");
103                 System.out.println("| many operands in your expression.
                     |");
104                 exceptionRaised = true;
105             }
106
107         } catch (IndexOutOfBoundsException ioobe) {
108             System.out.printf("| %-49s |\n", ioobe.getMessage());
109             exceptionRaised = true;
110         } catch (EmptyStackException ese) {
111             System.out.println("| Stack is Empty.
                 |");
112             exceptionRaised = true;
113         } catch (NumberFormatException nfe) {
114             System.out.println("| Your expression contained invalid
       characters.        |");
115             System.out.printf("| %-49s |\n", nfe.getMessage());
116             exceptionRaised = true;
117         } finally {
118             if (exceptionRaised) {
119                 System.out.println("| Your expression is invalid.
       Evaluation failed.    |");
120                 System.out.println("

       ");
121                 System.exit(1);
122             }
123         }
124         /*When the above iteration is complete, there should only be
125          * one item on the stack whcih is the answer.
126          */
127         System.out.println("
       |--------------------------------------------------|");
```

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

     ");
130        }
131 }
```

Task 3

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
677	677 is a prime number	677 is a prime number
34939	34939 is a prime number	34939 is a prime number
381165334	381165334 is not a prime number	381165334 is not a prime number
188737204	188737204 is not a prime number	188737204 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.