Michal Bochenek  
HND Computing Software Development year 2  
EC1401916

**Technical Documentation for  
 Outcome 4:  
 Postcode Sorting**

Table of contents:  
1. Program specification  
2. Code Listing  
3. Testing Strategy

1. Program Specification:  
     
   This program is reading given data inside text file and places each entry on the array list as a single string.  
   As a next step it checks for the conditions such as:  
     
   a) size of string, we are interested in string of a size between 6 and 8 which are the only ones valid for postal codes in the UK  
   b) if it matches a given pattern of symbols with pre-conditions such as:  
     
   AN\_NAA  
   ANN\_NAA  
   AAN\_NAA  
   AANN\_NAA  
   ANA\_NAA  
   AANA\_NAA  
     
   preconditions are such as that:  
     
   - The letters Q, V and X are not used **in the first position.**   
   - The letters I,J and Z are not used in **the second position**.   
   - The only letters to appear in **the third position** are A, B, C, D, E, F, G, H, J, K, S, T, U and W.  
   - The only letters to appear in **the fourth position** are A, B, E, H, M, N, P, R, V, W, X and Y.  
    - The second half of the postcode is always consistent **numeric, alpha, alpha** format and the letters C, I, K, M, O and V are never used.   
   - Postcodes should always be in BLOCK CAPITALS as the last line of an address. Do not underline the postcode or use any punctuation. Leave a clear space of one character between the two parts of the postcode and do not join the characters in any way  
     
   ref:  
   <http://www.datadictionary.nhs.uk/version2/data_dictionary/data_field_notes/p/postcode_de.asp?shownav=0>  
     
     
   Example of a working program:

|  |
| --- |
| This is program sorting UK post codes.  These are addresses to be validated:  [EH114DEN4 1BD, Y78 8HL, KL8M 9SR, J3 9XI, EL73 2BN, A8 6HN, R42 5DE, C1 8XU, EP8S 7LP, QH5B 5HR, I4A 7IQ, JR10 5PL, C8H 7AT, O59 1EB, N93 6MJ, V3F 4NO, NI8 1NI, Y3 8EV, G8D 9YN, RY4 2OC, B2 9GW, DU1E 1MO, X5A 2QS, A1D 1VC, Y3 6QS, R35 8QE, K1 1IY, O2G 2HN, N8P 7LO, FN3X 9XN, O2P 1MN, GL4 DE, 3H114DE, !!!&&&&, A6 3MA, M85 4BE, PO11 9GN, SD9 6YF, VY49 7YG, Y60 8SQ, XF9 2PA, T9J 5QG, L7U 9PL, EV25 7IK, QH11 2YP, Y93 4GD, PG51 3XA, IY11 2FI, P50 3VA, YC1 2DG, QS4K 3HT, BG3 3XV, RQ2V 9GI, E5A 6DQ, T6S 4OA, GL43 DE77, GL4DE, GL4 5FH]  Valid 6 characters long(with whitespace) addresses:  [J3 9XI, A8 6HN, C1 8XU, Y3 8EV, B2 9GW, Y3 6QS, K1 1IY, A6 3MA]  Valid 7 characters long(with whitespace) addresses:  [Y78 8HL, R42 5DE, I4A 7IQ, C8H 7AT, O59 1EB, N93 6MJ, V3F 4NO, NI8 1NI, G8D 9YN, RY4 2OC, X5A 2QS, A1D 1VC, R35 8QE, O2G 2HN, M85 4BE, SD9 6YF, Y60 8SQ, XF9 2PA, T9J 5QG, L7U 9PL, Y93 4GD, P50 3VA, YC1 2DG, BG3 3XV, E5A 6DQ, T6S 4OA, GL4 5FH]  Valid 8 characters long(with whitespace) addresses:  [KL8M 9SR, EL73 2BN, QH5B 5HR, JR10 5PL, DU1E 1MO, FN3X 9XN, PO11 9GN, VY49 7YG, EV25 7IK, QH11 2YP, PG51 3XA, IY11 2FI, RQ2V 9GI]  Invalid 6 characters long(with whitespace) addresses:  [GL4 DE]  Invalid 7 characters long(with whitespace) addresses:  [N8P 7LO, O2P 1MN, 3H114DE, !!!&&&&]  Invalid 8 characters long(with whitespace) addresses:  [EP8S 7LP, QS4K 3HT]  Other invalid strings:  [EH114DEN4 1BD, GL43 DE77, GL4DE]  These are all valid addresses:  [J3 9XI, A8 6HN, C1 8XU, Y3 8EV, B2 9GW, Y3 6QS, K1 1IY, A6 3MA, Y78 8HL, R42 5DE, I4A 7IQ, C8H 7AT, O59 1EB, N93 6MJ, V3F 4NO, NI8 1NI, G8D 9YN, RY4 2OC, X5A 2QS, A1D 1VC, R35 8QE, O2G 2HN, M85 4BE, SD9 6YF, Y60 8SQ, XF9 2PA, T9J 5QG, L7U 9PL, Y93 4GD, P50 3VA, YC1 2DG, BG3 3XV, E5A 6DQ, T6S 4OA, GL4 5FH, KL8M 9SR, EL73 2BN, QH5B 5HR, JR10 5PL, DU1E 1MO, FN3X 9XN, PO11 9GN, VY49 7YG, EV25 7IK, QH11 2YP, PG51 3XA, IY11 2FI, RQ2V 9GI]  BUILD SUCCESSFUL (total time: 0 seconds) |

1. Code Listing:

2.1 Main program:

|  |
| --- |
| package sortingpostcodes;  import java.util.ArrayList;  /\*\*  \*  \* @author ec1401916  \*/  public class SortingPostcodes  {  public static void main(String[] args)  {  //global variables  ArrayList<String> addressesread = new ArrayList<>();  ArrayList<String> valid = new ArrayList<>();  ArrayList<String> invalid = new ArrayList<>();    System.out.println("This is program sorting UK post codes.");  ReadTokens readem = new ReadTokens();  System.out.println("");    //read addresses from the txt file  readem.reading(addressesread);        //readem.validator(addressesread);    System.out.println("These are addresses to be validated: ");  System.out.println(addressesread);  System.out.println("");    //begin validation  readem.validate(addressesread, valid, invalid);      }  } |

2.2 ReadToken Class:

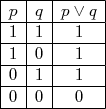
|  |
| --- |
| package sortingpostcodes;  import java.io.BufferedReader;  import java.io.File;  import java.io.FileReader;  import java.io.IOException;  import java.util.ArrayList;  import java.util.StringTokenizer;  import java.util.regex.\*;  /\*\*  \*  \* @author ec1401916  \*/  public class ReadTokens  {  //global variables  private ArrayList<String> addresses = new ArrayList<>();  private ArrayList<String> valid = new ArrayList<>();  private ArrayList<String> invalid = new ArrayList<>();    private ArrayList<String> ListOfAddresses = new ArrayList<>();      //this method reads tokens from the txt file and puts them to the arraylist  public ArrayList<String> reading(ArrayList<String> addresses)  {  try  {  File myFile = new File("postcodes1.txt");  FileReader fr = new FileReader(myFile);  BufferedReader br = new BufferedReader(fr);  String line = null;    while((line=br.readLine())!=null)  {    StringTokenizer st = new StringTokenizer(line,"");  int tokens = st.countTokens();    while(st.hasMoreTokens())  {  addresses.add(st.nextToken());  }  }      br.close();  }    catch(IOException e)  {  System.out.println("An unknown IO Error has occured");  }    return addresses;  }      public ArrayList<String> validate (ArrayList<String> addressesread, ArrayList<String> valid, ArrayList<String> invalid)  {    ArrayList<String> valid6 = new ArrayList<>();  ArrayList<String> valid7 = new ArrayList<>();  ArrayList<String> valid8 = new ArrayList<>();    ArrayList<String> invalid6 = new ArrayList<>();  ArrayList<String> invalid7 = new ArrayList<>();  ArrayList<String> invalid8 = new ArrayList<>();  //a single string length 6 with pattern instruction AN\_NAA  String strl6 ="^[^QVX[^0-9]][0-9][\\s][0-9][A-Z[^CIKMOV]]{2}$";  //a single string length 7 with patterns: ANN\_NAA, AAN\_NAA, ANA\_NAA,  String strl7 ="^[A-Z[^QVX]][0-9[A-Z[^IJZ]]][A-H[JKSTUW]0-9][\\s][0-9][A-Z[^CIKMOV]]{2}$";  //a single string length 8 with patterns: AANN\_NAA, AANA\_NAA  String strl8 ="^[A-Z[^QVX]][A-Z[^IJZ]][0-9][0-9[ABEHMNPRVWXY]][\\s][0-9][A-Z[^CIKMOV]]{2}$";    //a single string from our arraylist  String tester=null;  int strlength=0;    //do check length function here    for(int i=0; i<addressesread.size();i++)  {  tester=addressesread.get(i);    if(tester.length()==6)  {  //if strlength = 6 do this:  Pattern p6 = Pattern.compile(strl6);    //checks for given pattern if it matches the string inside matcher  Matcher m6 = p6.matcher(tester);    //if pattern matches the matcher it will return 1  boolean check = m6.matches();  //forward string to the valid arraylist and get next string  if(check==true)  {  valid6.add(tester);  }  else  {  invalid6.add(tester);  }    }      else if(tester.length()==7)  {  //if strlength = 7 do this:  Pattern p7 = Pattern.compile(strl7);    //checks for given pattern if it matches the string inside matcher  Matcher m7 = p7.matcher(tester);    //if pattern matches the matcher it will return 1  boolean check = m7.matches();  //forward string to the valid arraylist and get next string  if(check==true)  {  valid7.add(tester);  }  else  {  invalid7.add(tester);  }  }    else if(tester.length()==8)  {  //if strlength = 8 do this:  Pattern p8 = Pattern.compile(strl8);    //checks for given pattern if it matches the string inside matcher  Matcher m8 = p8.matcher(tester);    //if pattern matches the matcher it will return 1  boolean check = m8.matches();  //forward string to the valid arraylist and get next string  if(check==true)  {  valid8.add(tester);  }  else  {  invalid8.add(tester);  }  }  else if(tester.length()>8||tester.length()<6)  {  invalid.add(tester);  }  else  {  invalid.add(tester);  }  }      System.out.println("Valid 6 characters long(with whitespace) addresses: ");  System.out.println(valid6);  System.out.println("Valid 7 characters long(with whitespace) addresses: ");  System.out.println(valid7);  System.out.println("Valid 8 characters long(with whitespace) addresses: ");  System.out.println(valid8);  System.out.println("");    System.out.println("Invalid 6 characters long(with whitespace) addresses: ");  System.out.println(invalid6);  System.out.println("Invalid 7 characters long(with whitespace) addresses: ");  System.out.println(invalid7);  System.out.println("Invalid 8 characters long(with whitespace) addresses: ");  System.out.println(invalid8);  System.out.println("Other invalid strings: ");  System.out.println(invalid);  System.out.println("");  valid.addAll(valid6);  valid.addAll(valid7);  valid.addAll(valid8);    System.out.println("These are all valid addresses:");  System.out.println(valid);    return null;  }  } |

1. Test strategy:  
     
   To develop and later perform a post code test validation we can write a conditional logics table which will be a logical sum, resulting in TRUE condition, of all characters inside a string as an example for string length = 6 we can write:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| First | Second | Third | Fourth | Fifth | Sixth | Result |
| ^QVX[^0-9] | 0-9 | [\\s](file:///\\s) | 0-9 | A-Z[^CIKMOV] | A-Z[^CIKMOV] |  |
| Is not QVX and not a digit | Is a digit | Is a white space | Is a digit | Is between A-Z and not CIKMOV | Is between A-Z and not CIKMOV | TRUE |
| A | 9 |  | 3 | L | R | TRUE |
| Logical true/false: | | | | | | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | | | | | | |
| A | 8 |  | 6 | H | N | TRUE |
| Logical true/false: | | | | | | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | | | | | | |
| G | L | 4 |  | D | E | FALSE |
| Logical true/false: | | | | | | |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |

Using regular expression in Java we can create a pattern which will perform a simple Boolean operation resulting in the same as the above table shows, and this is what I was looking for in my program.  
  
How to read the patterns shown below:  
  
**A** – alphabetic character  
**N** – single digit from 0 to 9  
**\\s** – single white space character  
**^**…**$** – start and the end of a string  
[…]{2} – condition valid for the next two characters  
  
**^** – logical negation

  
  
  
  
**[**…**]** – logical alternative if used inside another group of characters […[…]]

  
  
**A-Z0-9** – logical conjunction  
  
\begin{array}{|c|c|c|}  \hline 
p &  q & p  \wedge  q  \\ \hline 
1&1&1 \\ \hline 
1&0&0 \\ \hline 
0&1&0 \\ \hline 
0&0&0 \\ \hline
\end{array}  
  
  
Single string length of 6 with pattern instance of: AN\_NAA

**^[^QVX[^0-9]] [0-9] [\\s] [0-9] [A-Z[^CIKMOV]]{2}$**

Single string length of 7 with patterns: ANN\_NAA, AAN\_NAA, ANA\_NAA,

**^[A-Z[^QVX]] [0-9[A-Z[^IJZ]]] [A-H[JKSTUW]0-9] [\\s] [0-9] [A-Z[^CIKMOV]]{2}$**

Single string length of 8 with patterns: AANN\_NAA, AANA\_NAA

**^[A-Z[^QVX]] [A-Z[^IJZ]] [0-9] [0-9[ABEHMNPRVWXY]] [\\s] [0-9] [A-Z[^CIKMOV]]{2}$**

Test results (from program output):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test name - String type:** | **Expected** | **Output** | **Comments:** | **Date** |
| 6 character long | AN\_NAA  with all preconditions being true | [J3 9XI, A8 6HN, C1 8XU, Y3 8EV, B2 9GW, Y3 6QS, K1 1IY, A6 3MA] | Program found these postal codes as presented, there were no exceptions or unexpected results during run - PASS | 12/01/2016 |
| Invalid 6 character long: | AN\_NAA  with some/all preconditions being false | [GL4 DE] | PASS | 12/01/2016 |
| 7 character long | ANN\_NAA, AAN\_NAA, ANA\_NAA  with all preconditions being true | [Y78 8HL, R42 5DE, I4A 7IQ, C8H 7AT, O59 1EB, N93 6MJ, V3F 4NO, NI8 1NI, G8D 9YN, RY4 2OC, X5A 2QS, A1D 1VC, R35 8QE, O2G 2HN, M85 4BE, SD9 6YF, Y60 8SQ, XF9 2PA, T9J 5QG, L7U 9PL, Y93 4GD, P50 3VA, YC1 2DG, BG3 3XV, E5A 6DQ, T6S 4OA, GL4 5FH] | PASS | 12/01/2016 |
| Invalid 7 character long: | ANN\_NAA, AAN\_NAA, ANA\_NAA  with some/all preconditions being false | [N8P 7LO, O2P 1MN, 3H114DE, !!!&&&&] | PASS | 12/01/2016 |
| 8 character long | AANN\_NAA, AANA\_NAA  with all preconditions being true | [KL8M 9SR, EL73 2BN, QH5B 5HR, JR10 5PL, DU1E 1MO, FN3X 9XN, PO11 9GN, VY49 7YG, EV25 7IK, QH11 2YP, PG51 3XA, IY11 2FI, RQ2V 9GI] | PASS | 12/01/2016 |
| Invalid 8 character long: | AANN\_NAA, AANA\_NAA  with some/all preconditions being false | [EP8S 7LP, QS4K 3HT] | PASS | 12/01/2016 |
| Other invalid | String length<6 and >8 | [EH114DEN4 1BD, GL43 DE77, GL4DE] | PASS | 12/01/2016 |
| Text file missing | File not found | This is program sorting UK post codes.  An unknown IO Error has occured or file not found!  BUILD SUCCESSFUL (total time: 0 seconds) | PASS | 12/01/2016 |