```java
package com.sanfoundry.setandstring;
import java.util.Scanner;

public class PlayfairCipherEncryption
{
    private String KeyWord        = new String();
    private String Key            = new String();
    private char   matrix_arr[][] = new char[5][5];

/*
 * method removes all duplicates from string k and creates new string adjustedKey
 */
    public void setKey(String k)
        {
            String adjustedKey = new String();
            boolean sameChar = false;
            adjustedKey = adjustedKey + k.charAt(0);
            for (int i = 1; i < k.length(); i++)
            {
                for (int j = 0; j < adjustedKey.length(); j++)
                {
                    if (k.charAt(i) == adjustedKey.charAt(j))
                    {
                        sameChar = true;
                    }
                }
                if (sameChar == false)
                    adjustedKey = adjustedKey + k.charAt(i);
                sameChar = false;
            }
            KeyWord = adjustedKey;
        }

/*
 * method generates key by appending alphabet to key w/ no repeated letters
 */
        public void KeyGen()
        {
            boolean append = true;
            char current;
            Key = KeyWord;
            for (int i = 0; i < 26; i++)
            {
                current = (char) (i + 97);
                //exclude letter j
                if (current == 'j')
                    continue;
                for (int j = 0; j < KeyWord.length(); j++)
                {
                    if (current == KeyWord.charAt(j))
                    {
                        append = false;
                        break;
                    }
                }
                if (append)
                    Key = Key + current;
                append = true;
            }
```

```
60              System.out.println(Key);
61              createCipherGrid();
62          }
63  /*
64   * creates 5x5 matrix grid with key
65   */
66          private void createCipherGrid()
67          {
68              int counter = 0;
69              for (int i = 0; i < 5; i++)
70              {
71                  for (int j = 0; j < 5; j++)
72                  {
73                      matrix_arr[i][j] = Key.charAt(counter);
74                      System.out.print(matrix_arr[i][j] + " ");
75                      counter++;
76                  }
77                  System.out.println();
78              }
79          }
80  /*
81   * replace letters i with j and appends x to separate repeated letters
82   */
83          private String format(String old_text)
84          {
85              int i = 0;
86              int len = 0;
87              String text = new String();
88              len = old_text.length();
89              for (int tmp = 0; tmp < len; tmp++)
90              {
91                  if (old_text.charAt(tmp) == 'j')
92                  {
93                      text = text + 'i';
94                  }
95                  else
96                      text = text + old_text.charAt(tmp);
97              }
98              len = text.length();
99              for (i = 0; i < len; i = i + 2)
100             {
101                 //separates repeated letters
102                 if (text.charAt(i + 1) == text.charAt(i))
103                 {
104                     text = text.substring(0, i + 1) +
105                             'x' + text.substring(i + 1);
106                 }
107             }
108             return text;
109         }
110 /*
111  * appends x if string length is not even and puts pairs of letters into array x
112  */
113         private String[] Divid2Pairs(String new_string)
114         {
115             String Original = format(new_string);
116             int size = Original.length();
117         if (size % 2 != 0)
118             {
```

```
119             //appending x increases size
120                 size++;
121                 Original = Original + 'x';
122             }
123             String letterPairs[] = new String[size / 2];
124             int counter = 0;
125             for (int i = 0; i < size / 2; i++)
126             {
127                 letterPairs[i] = Original.substring(counter, counter + 2);
128                 counter = counter + 2;
129             }
130             return letterPairs;
131         }
132 /*
133  *gets position of each letter from 5x5 matrix
134  */
135         public int[] GetDiminsions(char letter)
136         {
137             int[] dimensions = new int[2];
138             if (letter == 'j')
139                 letter = 'i';
140             for (int i = 0; i < 5; i++)
141             {
142                 for (int j = 0; j < 5; j++)
143                 {
144                     if (matrix_arr[i][j] == letter)
145                     {
146                         dimensions[0] = i;
147                         dimensions[1] = j;
148                         break;
149                 }
150                 }
151             }
152             return dimensions;
153         }
154 /*
155  *alters array to encode message
156  */
157         public String encryptMessage(String userInput)
158         {
159             String src_arr[] = Divid2Pairs(userInput);
160             String Code = new String();
161             char firstLetter;
162             char secondLetter;
163             int Dimensions1st[] = new int[2];
164             int Dimensions2nd[] = new int[2];
165             for (int i = 0; i < src_arr.length; i++)
166             {
167                 firstLetter = src_arr[i].charAt(0);
168                 secondLetter = src_arr[i].charAt(1);
169                 Dimensions1st = GetDiminsions(firstLetter);
170                 Dimensions2nd = GetDiminsions(secondLetter);
171 
172                 if (Dimensions1st[0] == Dimensions2nd[0])
173                 {
174                     if (Dimensions1st[1] < 4)
175                     Dimensions1st[1]++;
176                     else
177                         Dimensions1st[1] = 0;
```

```
178                     if (Dimensions2nd[1] < 4)
179                         Dimensions2nd[1]++;
180                     else
181                         Dimensions2nd[1] = 0;
182                 }
183                 else if (Dimensions1st[1] == Dimensions2nd[1])
184                 {
185                     if (Dimensions1st[0] < 4)
186                         Dimensions1st[0]++;
187                     else
188                         Dimensions1st[0] = 0;
189                     if (Dimensions2nd[0] < 4)
190                         Dimensions2nd[0]++;
191                     else
192                         Dimensions2nd[0] = 0;
193                 }
194                 else
195                 {
196                     int temp = Dimensions1st[1];
197                     Dimensions1st[1] = Dimensions2nd[1];
198                     Dimensions2nd[1] = temp;
199                 }
200                 Code = Code + matrix_arr[Dimensions1st[0]][Dimensions1st[1]]
201                        + matrix_arr[Dimensions2nd[0]][Dimensions2nd[1]];
202             }
203             return Code;
204         }
205 /*
206  * takes user input to be encoder, starts and calls methods
207  */
208     public static void main(String[] args)
209         {
210             PlayfairCipherEncryption message = new
211                   PlayfairCipherEncryption();
212             Scanner sc = new Scanner(System.in);
213             System.out.println("Enter a keyword:");
214             String keyword = sc.next();
215             message.setKey(keyword);
216             message.KeyGen();
217             System.out.println("Enter word to encrypt: " +
218                                "(Make sure length of message is even)");
219             String userInput = sc.next();
220             //if (userInput.length() % 2 == 0)
221             //{
222                 System.out.println("Encryption: " +
223                                    message.encryptMessage(userInput));
224             //}
225             //else
226             //{
227                // System.out.println("Message length should be even");
228             //}
229             sc.close();
230         }
231 }
232
```