```
1  package Assign_3;
2
3  import java.io.Serializable;
4
5  public class ConCharacterSequence implements CharacterSequence, Serializable {
6
7    private char[] seq;
8
9    public ConCharacterSequence () {
10
11     seq = new char[0];
12
13   }
14
15   public ConCharacterSequence (char c) {
16
17     seq = new char[1];
18
19     seq[0] = c;
20
21   }
22
23   public ConCharacterSequence (CharSequence cs) {
24
25     seq = new char[cs.length()];
26
27     for (int i=0; i<cs.length(); i++) {
28
29       seq[i] = cs.charAt(i);
30     }
31   }
32
33   public ConCharacterSequence (char[] c) {
34
35     seq = new char[c.length];
36
37     for (int i=0; i<c.length; i++) {
38
39       seq[i] = c[i];
40     }
41
42   }
43
44
45   public boolean equals ( CharSequence cs ) {
46
47     boolean result = true;
48
49     if (cs.length() == 0) {
50
51       result = false;
52
53     }
54
55     for (int i=0; i<cs.length(); i++) {
56
57       if (seq[i] != cs.charAt(i)) {
58
59         result = false;
60
61       }
```

```
62      }
63
64      return result;
65
66    }
67
68    public int compareTo ( CharSequence cs ) {// compare the length of two string
69
70      int result;
71
72      result = seq.length - cs.length();
73
74      return result;
75
76    }
77
78    public ConCharacterSequence toLowerCase ( ) {// to lower case
79
80      char[] result;
81
82      result = new char[seq.length];
83
84      for (int i=0; i<seq.length; i++) {
85
86        if (Character.isUpperCase(seq[i])) {
87
88           result[i] = Character.toLowerCase(seq[i]);
89
90        }
91        else {
92
93           result[i] = seq[i];
94
95        }
96      }
97
98
99      return new ConCharacterSequence(result);
100
101   }
102
103   public ConCharacterSequence toUpperCase() {// to upper case
104
105     char[] result;
106
107     result = new char[seq.length];
108
109     for (int i=0; i<seq.length; i++) {
110
111       if (Character.isLowerCase(seq[i])) {
112
113         result[i] = Character.toUpperCase(seq[i]);
114       }
115       else {
116
117          result[i] = seq[i];
118
119       }
120     }
121
122     return new ConCharacterSequence(result);
```

```
123
124
125
126   }
127
128  public ConCharacterSequence trim ( ) {
129
130    char[] result;
131    int pos=0;
132    int start=0;
133    int end=seq.length-1;
134
135      while (Character.isWhitespace(seq[start])) {
136
137        start=start+1;
138
139
140      }
141
142      while (Character.isWhitespace(seq[end])) {
143
144        end=end-1;
145
146      }
147
148
149    result = new char[end-start+1];
150
151    for (int i=start; i<=end; i++) {
152
153      result[pos] = seq[i];
154      pos = pos + 1;
155
156    }
157
158    return new ConCharacterSequence(result);
159
160  }
161
162  public ConCharacterSequence trimAll ( ) {// this method is remove all the space
163
164    char[] result;
165    int count=0;
166    int pos=0;
167
168    for (int i=0; i<seq.length; i++) {
169
170      if (Character.isWhitespace(seq[i])) {
171
172        count++;
173
174      }
175
176    }
177
178    result = new char[seq.length-count];
179
180    for (int i=0; i<seq.length; i++) {
181
182      if (! Character.isWhitespace(seq[i])) {
183
```

```
184         result[pos] = seq[i];
185         pos = pos + 1;
186 
187       }
188 
189     }
190 
191     return new ConCharacterSequence(result);
192 
193   }
194 
195   public ConCharacterSequence replace ( char oldChar, char newChar ) {//this method is used to
    replace char to char
196 
197     for (int i=0; i<seq.length; i++) {
198 
199       if (seq[i] == oldChar) {
200 
201         seq[i] = newChar;
202 
203       }
204 
205     }
206 
207     return new ConCharacterSequence(seq);
208 
209   }
210 
211   public ConCharacterSequence concat ( CharSequence tail ) {//this method is used to concat strings
212 
213     char [] result;
214     int pos=0;
215     result = new char[seq.length+tail.length()];
216 
217     if (tail.length() == 0) {
218 
219       return new ConCharacterSequence(seq);
220 
221     }
222     else {
223 
224       for (int i=0; i<seq.length; i++) {
225 
226         result[i] = seq[i];
227 
228 
229       }
230 
231       for (int i=seq.length; i<seq.length+tail.length(); i++) {
232 
233         result[i] = tail.charAt(pos);
234         pos = pos + 1;
235 
236       }
237 
238       return new ConCharacterSequence(result);
239     }
240   }
241 
242   public boolean isPalindrome ( ) {// this method is used to check if the string is palindrome in the
    lowercase and removed space
```

```
243
244     int pos=0;
245     char[] reverse = new char[seq.length];
246     boolean result = true;
247
248     if (seq.length == 0) {
249
250       result = false;
251
252     }
253
254     for (int i=seq.length-1; i>=0; i--) {
255
256
257       reverse[pos] = seq[i];
258       pos = pos + 1;
259
260     }
261
262     if (! seq.equals(reverse)) {
263
264       result = false;
265
266     }
267
268     return result;
269
270 }
271
272 public char charAt( int index ) {// to check the index's length
273
274   if (index < 0 || index >= seq.length) {
275
276      throw new IndexOutOfBoundsException();
277   }
278   else {
279
280      return seq[index];
281
282   }
283
284 }
285
286
287 public int length ( ) {
288
289   return seq.length;
290
291
292 }
293
294 public ConCharacterSequence subSequence ( int start, int end ) {
295
296   char[] result;
297   int pos=0;
298
299   result = new char[end-start+1];
300
301   if (start < 0 || start >= seq.length || end < 0 || end >= seq.length) {
302
303     throw new IndexOutOfBoundsException();
```

```
304
305    }
306    else {
307
308      for (int i=start; i<=end; i++) {
309
310        result[pos] = seq[i];
311        pos = pos + 1;
312
313      }
314
315    }
316
317    return new ConCharacterSequence(result);
318
319  }
320
321  public String toString() {
322
323     String result = "";
324
325     result = new String(seq);
326
327     return result;
328
329  }
330
331 }//Xinan Wang No.5535802
```