## DataPacket.java

```
1 package com.nxkundu.server.bo;
 3 import java.io.Serializable;
4 import java.util.Base64;
 5 import java.util.Date;
6 import java.util.UUID;
 8 import com.google.gson.Gson;
10 /**
11 *
12 * @author nxkundu
13 *
14 * @email nxk161830@utdallas.edu
15 * @name Nirmallya Kundu
16 *
17 * DataPacket
18 * This DataPacket object is sent and received
19 * between the server and the client
20 *
21 * The ACTION_TYPE defines the type of Action that needs to taken
22 * when the server or the client receives
23 *
24 * Type of ACTION_TYPE are:
25 * ACTION_TYPE_MESSAGE, ACTION_TYPE_SIGNUP, ACTION_TYPE_LOGIN, ACTION_TYPE_LOGIN_SUCCESS,
26 * ACTION_TYPE_LOGIN_FAILED, ACTION_TYPE_SIGNUP_FAILED, ACTION_TYPE_LOGOUT,
27 * ACTION_TYPE_ONLINE, ACTION_TYPE_ACK
28 *
29 * The MESSAGE_TYPE defines the type of Message received
30 * and what the specific action the server or the client
31 * will take upon receiving
32 *
33 * Type of MESSAGE_TYPE are:
34 * MESSAGE_TYPE_MESSAGE, MESSAGE_TYPE_BROADCAST_MESSAGE, MESSAGE_TYPE_IMAGE_MESSAGE
35 *
36 * Each DataPacket has a unique Id UUID id
37 *
```

```
38 * Methods:
39 *
40 * 1> clone() - Creates a clone of the DataPacket object
41 * This is used in case of Broadcast Message
42 * as we need to make a clone of the same DataPacket object
43 * and send it to all the clients
44 *
45 * 2> toJSON() - Converts the DataPacket object to the
46 * JSON object making it suitable to send and receive
47 * over the network
48 *
49 * 3> getByteImage() - Decodes the Image
50 *
51 * 4> setByteImage() - Encodes the Image
52 *
53 *
54 */
55 public class DataPacket implements Serializable, Cloneable{
56
      /**
57
58
59
       */
60
      private static final long serialVersionUID = 1L;
61
62
      public static final String ACTION_TYPE_MESSAGE = "MESSAGE";
63
      public static final String ACTION_TYPE_SIGNUP = "SIGNUP";
64
      public static final String ACTION_TYPE_LOGIN = "LOGIN";
65
      public static final String ACTION_TYPE_LOGIN_SUCCESS";
66
      public static final String ACTION_TYPE_LOGIN_FAILED = "LOGIN_FAILED";
67
      public static final String ACTION_TYPE_SIGNUP_FAILED = "SIGNUP_FAILED";
68
      public static final String ACTION_TYPE_LOGOUT = "LOGOUT";
      public static final String ACTION_TYPE_ONLINE = "ONLINE";
69
70
      public static final String ACTION_TYPE_ACK = "ACK";
71
72
      public static final String MESSAGE_TYPE_MESSAGE = "SINGLE_MESSAGE";
73
      public static final String MESSAGE_TYPE_BROADCAST_MESSAGE = "BROADCAST_MESSAGE";
74
      public static final String MESSAGE_TYPE_IMAGE_MESSAGE = "IMAGE_MESSAGE";
```

```
75
76
      private UUID id;
77
78
      private String action;
79
80
      private String messageType;
81
82
      private Client fromClient;
83
      private Client toClient;
84
85
      private String message;
86
87
      private String stringImage;
88
89
      private long timestamp;
90
91
      private boolean isACK;
92
93
      private int timesResentDataPacket;
94
      95
96
97
      public DataPacket(Client fromClient, String action) {
98
         super();
         this.action = action;
99
         this.setId(UUID.randomUUID());
100
         this.setACK(false);
101
         this.setTimestamp(new Date().getTime());
102
103
         this.fromClient = fromClient;
         this.setTimesResentDataPacket(0);
104
     }
105
106
      107
108
109
      /**
      * Creates a clone of the DataPacket object
110
      * This is used in case of Broadcast Message
111
```

```
* as we need to make a clone of the same DataPacket object
112
113
        * and send it to all the clients
114
        */
115
       @Override
       public Object clone() throws CloneNotSupportedException {
116
           return super.clone();
117
118
       }
119
120
        * Converts the DataPacket object to the
121
        * JSON object making it suitable to send and receive
122
123
        * over the network
124
        * @return
125
        */
       public String toJSON() {
126
127
128
           Gson gson = new Gson();
           String strJSON = gson.toJson(this);
129
130
           return strJSON;
       }
131
132
133
       * Decodes the Image
134
135
        * @return
136
        */
       public byte[] getByteImage() {
137
138
139
           return Base64.getDecoder().decode(stringImage);
140
       }
141
142
        * Encodes the Image
143
144
        * @param byteImage
145
146
       public void setByteImage(byte[] byteImage) {
147
           this.stringImage = Base64.getEncoder().encodeToString(byteImage);
148
```

## DataPacket.java

```
}
149
150
151
      152
153
154
      @Override
      public String toString() {
155
         return "DataPacket [id=" + id + ", action=" + action + ", messageType=" + messageType + ", fromClient="
156
157
               + fromClient + ", toClient=" + toClient + ", message=" + message + ", stringImage=" + stringImage
               + ", timestamp=" + timestamp + ", isACK=" + isACK + "]";
158
159
      }
160
161
      162
163
      public String getStringImage() {
164
165
         return stringImage;
166
      }
167
168
      public void setStringImage(String stringImage) {
169
         this.stringImage = stringImage;
      }
170
171
172
      public boolean isACK() {
173
         return isACK;
      }
174
175
176
      public void setACK(boolean isACK) {
177
         this.isACK = isACK;
      }
178
179
      public String getAction() {
180
181
         return action;
182
183
184
      public void setAction(String action) {
185
         this.action = action;
```

```
186
       }
187
188
       public String getMessageType() {
189
           return messageType;
190
       }
191
       public void setMessageType(String messageType) {
192
           this.messageType = messageType;
193
194
       }
195
       public Client getFromClient() {
196
           return fromClient;
197
198
       }
199
       public void setFromClient(Client fromClient) {
200
           this.fromClient = fromClient;
201
202
       }
203
204
       public Client getToClient() {
           return toClient;
205
206
       }
207
       public void setToClient(Client toClient) {
208
209
           this.toClient = toClient;
       }
210
211
       public String getMessage() {
212
213
           return message;
214
       }
215
       public void setMessage(String message) {
216
217
           this.message = message;
218
       }
219
       public UUID getId() {
220
           return id;
221
       }
222
```

```
223
224
       public void setId(UUID id) {
225
           this.id = id;
       }
226
227
228
       public long getTimestamp() {
229
           return timestamp;
230
       }
231
       public void setTimestamp(long timestamp) {
232
           this.timestamp = timestamp;
233
234
       }
235
236
       public int getTimesResentDataPacket() {
237
           return timesResentDataPacket;
238
       }
239
240
       public void setTimesResentDataPacket(int timesResentDataPacket) {
241
           this.timesResentDataPacket = timesResentDataPacket;
242
       }
243
       public void incrementTimesResentDataPacket() {
244
           this.timesResentDataPacket += 1;
245
246
       }
247
248 }
249
```