```java
package ballmerpeak.turtlenet.shared;

import ballmerpeak.turtlenet.shared.Tokenizer;
import java.security.*;
import java.io.Serializable;
import java.util.Arrays;

public class Message implements Serializable {
    //You shouldn't use this, rather use MessageFactory.newMessage(command, data)
    //GWT cannot use the factory, it shouldn't construct messages but pass their
    //    data as arguments to whatever needs it. Maybe have an async factory?
    public Message (String cmd, String _content, long timeCreated, String RSAsig) {
        command   = cmd;
        content   = _content;
        signature = RSAsig;
        timestamp = timeCreated;
    }

    public Message () {
        command = "NULL";
        content = "";
        signature = "";
        timestamp = -1;
    }

    /* "POST\520adfc4\Hello, World!\123" -> new Message("POST", "Hello, World!", "520adfc4", 123) */
    public static Message parse (String msg) {
        String[] tokens = new String[4];
        Tokenizer tokenizer = new Tokenizer(msg, '\\');
        tokens[0] = tokenizer.nextToken(); //command
        tokens[1] = tokenizer.nextToken(); //signature
        tokens[2] = msg.substring(msg.indexOf("\\", msg.indexOf("\\",0)+1)+1, msg.lastIndexOf("\\")); //message content
        tokens[3] = msg.substring(msg.lastIndexOf("\\")+1); //timestamp
        long ts = Long.parseLong(tokens[3]);

        return new Message(tokens[0], tokens[2], ts, tokens[1]);
    }

    public String toString () {
        return command + "\\" + signature + "\\" + content + "\\" + timestamp;
    }

    /* universal */
    public String getCmd () {
        return command;
    }

    public String getSig () {
        return signature;
    }

    public String getContent () {
        return content;
    }

    public long getTimestamp () {
        return timestamp;
    }

    /* type specific */
    public String POSTgetText() {
        Tokenizer tokenizer = new Tokenizer(content, ':');
        String[] colonPairs = new String[tokenizer.countTokens()];
        for (int i = 0; tokenizer.hasMoreTokens(); i++)
            colonPairs[i] = tokenizer.nextToken();
        return colonPairs[colonPairs.length-1];
    }

    public String POSTgetWall() {
        Tokenizer tokenizer = new Tokenizer(content, ':');
        String[] colonPairs = new String[tokenizer.countTokens()];
        for (int i = 0; tokenizer.hasMoreTokens(); i++)
            colonPairs[i] = tokenizer.nextToken();
        return colonPairs[0];
    }

    public String[] POSTgetVisibleTo() {
        Tokenizer tokenizer = new Tokenizer(content, ':');
        String[] colonPairs = new String[tokenizer.countTokens()];
        for (int i = 0; tokenizer.hasMoreTokens(); i++)
            colonPairs[i] = tokenizer.nextToken();
        return Arrays.copyOfRange(colonPairs, 1, colonPairs.length-1);
    }

    public String CLAIMgetName() {
        return content;
    }

    //content in form "field1:value1;field2:value2;"
    public String[][] PDATAgetValues() {
        //Split into colon pairs, semicolon delimiter
        Tokenizer tokenizer = new Tokenizer(content, ';');
        String[] colonPairs = new String[tokenizer.countTokens()];
```

```
94            for (int i = 0; tokenizer.hasMoreTokens(); i++)
95                colonPairs[i] = tokenizer.nextToken();
96
97            //split into field/value pairs, colon delimiter
98            String[][] values = new String[colonPairs.length][2];
99            for (int i = 0; i < colonPairs.length; i++) {
100               values[i][0] = Message.beforeColon(colonPairs[i]);
101               values[i][1] = Message.afterColon(colonPairs[i]);
102           }
103
104           return values;
105       }
106
107       /* establish a chat and the people in it, without any messages */
108       // returns an array of strings and now of keys because of GWT,
109       //   Crypto.decodeKey should be used to turn each string into a key
110       public String[] CHATgetKeys() {
111           Tokenizer st = new Tokenizer(content, ':');
112           String[] keys = new String[st.countTokens()];
113           for (int i = 0; i < keys.length; i++)
114               keys[i] = st.nextToken();
115           return keys;
116       }
117
118       /* PCHAT adds messages to a conversation */
119       /* returns <conversation ID, messageText> */
120       public String PCHATgetText() {
121           Tokenizer st = new Tokenizer(content, ':');
122           String convoID = st.nextToken();
123           String text    = st.nextToken();
124           return text;
125       }
126
127       public String PCHATgetConversationID() {
128           Tokenizer st = new Tokenizer(content, ':');
129           String convoID = st.nextToken();
130           String text    = st.nextToken();
131           return convoID;
132       }
133
134       public String CMNTgetText() {
135           Tokenizer st = new Tokenizer(content, ':');
136           String itemID  = st.nextToken();
137           String text    = st.nextToken();
138           return text;
139       }
140
141       public String CMNTgetItemID() {
142           Tokenizer st = new Tokenizer(content, ':');
143           String itemID  = st.nextToken();
144           String text    = st.nextToken();
145           return itemID;
146       }
147
148       public String LIKEgetItemID() {
149           return content;
150       }
151
152       public String UNLIKEgetItemID() {
153           return content;
154       }
155
156       public String EVNTgetName() {
157           Tokenizer st = new Tokenizer(content, ':');
158           long start  = Long.parseLong(st.nextToken());
159           long end    = Long.parseLong(st.nextToken());
160           String name = st.nextToken();
161           return name;
162       }
163
164       public long EVNTgetStart() {
165           Tokenizer st = new Tokenizer(content, ':');
166           long start  = Long.parseLong(st.nextToken());
167           long end    = Long.parseLong(st.nextToken());
168           String name = st.nextToken();
169           return start;
170       }
171
172       public long EVNTgetEnd() {
173           Tokenizer st = new Tokenizer(content, ':');
174           long start  = Long.parseLong(st.nextToken());
175           long end    = Long.parseLong(st.nextToken());
176           String name = st.nextToken();
177           return end;
178       }
179
180       /* time of revocation, not timestamp of message */
181       /* there cannot be a REVOKEgetKey due to GWT */
182       public long REVOKEgetTime() {
183           try {
184               return Long.parseLong(content);
185           } catch (Exception e) {
186               //Invalid timestamp
```

```
187                 return -1;
188             }
189         }
190 
191         public String ADDCATgetName() {
192             return Message.afterColon(content);
193         }
194 
195         public boolean ADDCATgetValue() {
196             return Message.beforeColon(content).equals("true");
197         }
198 
199         public String UPDATECATgetName() {
200             return Message.afterColon(content);
201         }
202 
203         public boolean UPDATECATgetValue() {
204             return Message.beforeColon(content).equals("true");
205         }
206 
207         public String ADDTOCATgetName() {
208             return Message.afterColon(content);
209         }
210 
211         public String ADDTOCATgetKey() {
212             return Message.beforeColon(content);
213         }
214 
215         public String REMFROMCATgetCategory() {
216             return Message.afterColon(content);
217         }
218 
219         public String REMFROMCATgetKey() {
220             return Message.beforeColon(content);
221         }
222 
223         public String ADDKEYgetKey() {
224             return content;
225         }
226 
227         public static String beforeColon(String s) {
228             return s.substring(0, s.indexOf(':'));
229         }
230 
231         public static String afterColon(String s) {
232             return s.substring(s.indexOf(':')+1);
233         }
234 
235         public String command;
236         public String content;
237         public String signature;
238         public long timestamp;
239     }
```