```
package gratismeldinger.server;

public class CheckMessage
    implements FMMessage {
  public static final char TYPE = 'C';

  public CheckMessage() {
  }

  public char getType() {
    return TYPE;
  }

  public String[] getAll() {
    return null;
  }
}
```

```
package gratismeldinger.server;

import java.sql.*;

public class FMDatabase {
  Connection con;
  Statement statement;
  ResultSet rs;
  FMDatabase() {

    try {
      Class.forName( "com.mysql.jdbc.Driver" ).newInstance();

      con = DriverManager.getConnection(
          "jdbc:mysql://skabb.com:3306/gratismeldinger", "skabb", "pukkverk" );

      statement = con.createStatement();
      System.out.println( "Connected to mysqldb!" );
    }
    catch ( Exception e ) {
      System.err.println( "Error in FMDatabase-ctor: " + e );
      e.printStackTrace();
    }

  }

  String getPassword( String user ) {
    try {
      rs = statement.executeQuery( "Select password FROM User WHERE username = '" + user + "'" );

      if ( !rs.next() ) {
        return null;
      }

      return rs.getString( 1 );
    }
    catch ( SQLException ex ) {
      System.err.println( "Exception in getPassword: " + ex );
      ex.printStackTrace();

      return null;
    }
  }

  String getPassword( int userId ) {
    try {
      rs = statement.executeQuery( "Select password FROM User WHERE id = " + userId );
```



```
      if ( !rs.next() ) {
        return null;
      }

      return rs.getString( 1 );
    }
    catch ( SQLException ex ) {
      System.err.println( "Exception in getPassword: " + ex );
      ex.printStackTrace();
      return null;
    }
  }

  String getUsername( int userId ) {
    try {
      rs = statement.executeQuery( "Select username FROM User WHERE id = " + userId );
      if ( !rs.next() ) {
        return null;
      }

      return rs.getString( 1 );
    }
    catch ( SQLException ex ) {
      System.err.println( "Exception in getPassword: " + ex );
      ex.printStackTrace();

      return null;
    }
  }

  int getUserId( String username ) {
    try {
      rs = statement.executeQuery( "Select id FROM User WHERE username = '" + username + "'" );
      if ( !rs.next() ) {
        return -1;
      }

      return rs.getInt( 1 );
    }
    catch ( SQLException ex ) {
      System.err.println( "Exception in getUserId: " + ex );
      return -1;
    }

  }

  boolean addMessage( FMSms sms ) {
```

```
    try {
      int fromUserId = getUserId( sms.getFrom() );
      int toUserId = getUserId( sms.getTo() );

      if ( fromUserId == -1 || toUserId == -1 ) {
        return false;
      }
      PreparedStatement pstmt =
          con.prepareStatement( "INSERT INTO Message (fromId , toId, message, sendtWhen, statusReport)\n
" +
                                "VALUES ( ?, ?, ?, ?, ?)" );
      pstmt.setInt( 1, fromUserId );
      pstmt.setInt( 2, toUserId );
      pstmt.setString( 3, sms.getMessage() );
      pstmt.setLong( 4, sms.getWhen().getTime() );
      pstmt.setBoolean( 5, sms.getStatusReport() );
      int res = pstmt.executeUpdate();
      return res > 0;
    }
    catch ( SQLException ex ) {
      System.err.println( "Exception in addMessage: " + ex );
      ex.printStackTrace();
      return false;
    }
  }

  FMSms[] getNewMessages( String username, boolean markAsRead ) {
    try {
      PreparedStatement pstmt =
          con.prepareStatement( "SELECT f.username, t.username, ‘message‘, ‘sendtWhen‘, ‘statusReport‘\n "
+
                                "FROM Message, User AS f, User AS t\n " +
                                "WHERE ‘toId‘ = ?\n " +
                                "AND ‘read‘ = 0\n " +
                                "AND f.id = fromId\n " +
                                "AND t.id = toId" ); //
      int toId = getUserId( username );
      pstmt.setInt( 1, toId );

      rs = pstmt.executeQuery();
      rs.last();
      FMSms ret[] = new FMSms[ rs.getRow() ];
      rs.beforeFirst();
      int i = 0;

      while ( rs.next() ) {
        ret[ i++ ] = new FMSms( rs.getString( 1 ),
```



```
                                rs.getString( 2 ),
                                rs.getString( 3 ),
                                new java.util.Date( rs.getLong( 4 ) ),
                                rs.getBoolean( 5 ) );

      }

      try {
        statement.executeUpdate( "UPDATE Message SET ‘read‘ = 1 WHERE ‘toId‘ = " + toId );
      }
      catch ( Exception ex ) {
        ex.printStackTrace();
      }
      return ret;
    }
    catch ( SQLException ex ) {
      ex.printStackTrace();
      return null;
    }

  }

}
/*
  rs = statement.executeQuery( "SELECT AnswerId1, AnswerId2, AnswerId3, AnswerId4 FROM Question
  WHERE Question = ’" +
         p.get( "sprgtxt" + j ).replaceAll( "\\’",
         "\\\\\\’" ) +
         "’" );
  rs.next();
  int answerid1 = rs.getInt( 1 ),
      answerid2 = rs.getInt( 2 ),
      answerid3 = rs.getInt( 3 ),
      answerid4 = rs.getInt( 4 );
  statement.executeUpdate( "UPDATE Answer SET Correct=1" +
      " WHERE Answer = ’" +
      svar.replaceAll( "\\’", "\\\\\\’" ) +
      "’ AND ( AnswerId = " + answerid1 +
      " OR AnswerId = " + answerid2 +
      " OR AnswerId = " + answerid3 +
      " OR AnswerId = " + answerid4 + ")" );
      statement.executeUpdate( "INSERT INTO Question (Question, AnswerId1, AnswerId2, AnswerId3,
  AnswerId4) VALUES (’" +
                          p.get( "sprgtxt" + j ).replaceAll( "\\’", "\\\\\\’" ) + "’ , " +
                          p.get( "svarid" + j + "_1" ) + " , " +
                          p.get( "svarid" + j + "_2" ) + " , " +
                          p.get( "svarid" + j + "_3" ) + " , " +
```

```
                    p.get( "svarid" + j + "_4" ) + ")" );
 */
```

```
package gratismeldinger.server;

public interface FMMessage {
  char getType();

  String[] getAll();

}
```

```
package gratismeldinger.server;

import java.util.*;

public class FMSms {

  private String message;
  private String to;
  private String from;
  private Date when;
  private boolean statusReport;

  public FMSms( String from, String to, String message, Date when, boolean statusReport ) {
    this.from = from;
    this.to = to;
    this.message = message;
    this.when = when;
    this.statusReport = statusReport;
  }

  public String getMessage() {
    return message;
  }

  public String getTo() {
    return to;
  }

  public String getFrom() {
    return from;
  }

  public Date getWhen() {
    return when;
  }

  public boolean getStatusReport() {
    return statusReport;
  }

}
```

```
package gratismeldinger.server;

public class LoginMessage
    implements FMMessage {
  public static final char TYPE = 'L';

  private String user, password, version;

  public LoginMessage() {
  }

  public LoginMessage( String[] s ) {
    this.user = s[ 0 ];
    this.password = s[ 1 ];
    this.version = s[ 2 ];
  }

  public char getType() {
    return TYPE;
  }

  public String[] getAll() {
    return new String[] {
        user, password, version};
  }

  public String getUser() {
    return user;
  }

  public String getPassword() {
    return password;
  }

  public void setUser( String user ) {
    this.user = user;
  }

  public void setPassword( String password ) {
    this.password = password;
  }

}
```

```
package gratismeldinger.server;

public class SMSMessage
    implements FMMessage {
  public static final char TYPE = 'M';

  private String to;
  private String message;
  private boolean report;

  public SMSMessage() {
  }

  public SMSMessage( String[] s ) {
    to = s[ 1 ];
    message = s[ 0 ];
    report = s[ 2 ].equals( "1" );
  }

  public String getTo() {
    return to;
  }

  public String getMessage() {
    return message;
  }

  public char getType() {
    return TYPE;
  }

  public String[] getAll() {
    return null;
  }

  public boolean getStatusReport() {
    return report;
  }
}
```

```
package gratismeldinger.server;

import java.net.*;

public class Server {

  private static final int portno = 25;
  private boolean running = false;
  public static FMDatabase database = new FMDatabase();

  public static void main( String[] args ) {
    for ( ; ; ) {
      try {
        Server server = new Server();
      }
      catch ( Exception ex ) {
        ex.printStackTrace();
      }
    }
  }

  public Server() throws Exception {
    ServerSocket server = new ServerSocket( portno );
    try {
      SocketHandler sh = new SocketHandler();
      running = true;
      System.out.println( "Listening for connections..." );
      while ( running ) {
        Socket s = server.accept();
        System.out.print( "New connection from: " + s.getInetAddress().getHostName() );
        sh.add( s );
        System.out.println( " - acepted..." );

      }
    }
    catch ( Exception ex ) {
      server.close();
      throw ex;
    }
  }

}
```

```
package gratismeldinger.server;

import java.io.*;
import java.net.*;
import java.util.*;

public class SocketHandler
    implements SocketHandlerInterface {

  List clients = new Vector();

  public SocketHandler() {
  }

  public void add( Socket s ) {
    ClientHandler client = null;
    try {
      client = new ClientHandler( s, this );
      clients.add( client );
      client.start();
    }
    catch ( Exception ex ) {
      clients.remove( client );

    }

  }

  public void remove( ClientHandler c ) {
    try {
      System.out.println("Disconnecting: " + c.socket.getInetAddress().getHostName());
      c.stop();
    }
    catch ( Exception ex ) {
      ex.printStackTrace();
    }
    try {
      clients.remove( c );
    }
    catch ( Exception ex ) {
      ex.printStackTrace();
    }
  }
}

class ClientHandler
    extends Thread {
```



```
  Socket socket;
  private SocketHandlerInterface sh;
  private PrintStream ps;
  private BufferedInputStream bis;
  private String user = null;

  static final boolean DEBUG = true;

  public ClientHandler( Socket socket, SocketHandlerInterface sh ) {
    this.socket = socket;
    this.sh = sh;

    try {
      ps = new PrintStream( socket.getOutputStream() );
      bis = new BufferedInputStream( socket.getInputStream() );
    }
    catch ( IOException ex ) {
      ex.printStackTrace();
    }

  }

  private void handleLogin( LoginMessage m ) throws IOException {
    if ( DEBUG ) {
      System.out.println( "LoginMessage: " + m.getUser() + " - " + m.getPassword() );
    }

    if ( Server.database.getPassword( m.getUser() ).equalsIgnoreCase( m.getPassword() ) ) {
      sendString( "1" );
      System.out.println( "Password ok!" );
      user = m.getUser();
    }
    else {
      sendString( "0" );
      System.out.println( "Wrong user/password" );
    }

  }

  public void handleNewSms( SMSMessage m ) throws IOException {
    if ( user != null ) { //logged in
      if ( DEBUG ) {
        System.out.println( "SMSMessage: " + m.getTo() + " - " + m.getMessage() );
      }
      FMSms sms = new FMSms( user, m.getTo(), m.getMessage(),Calendar.getInstance().getTime(),
m.getStatusReport() );
```

```
    sendString( Server.database.addMessage( sms ) ? "1" : "0" );
  }

}

public void run() {
  try {
    for ( ; ; ) {
      FMMessage p = getNextMessage();
      if ( p == null ) {
        sh.remove(this);
        return;
      }

      if ( p.getType() == LoginMessage.TYPE ) {
        handleLogin( ( LoginMessage ) p );
      }

      else if ( p.getType() == SMSMessage.TYPE ) {
        handleNewSms( ( SMSMessage ) p );
      }
      else if ( p.getType() == CheckMessage.TYPE ) {
        FMSms[] sms = Server.database.getNewMessages( user, true );
        if ( sms == null || sms.length == 0 ) {
          sendString( "0" );
        }
        else {
          Vector v = new Vector();
          String send = "" + sms.length;
          String arr[] = new String[ sms.length * 3 + 1 ];
          arr[ 0 ] = "" + Integer.toString( sms.length, 32 );
          for ( int i = 0; i < sms.length; ++i ) {
            arr[ i * 3 + 1 ] = sms[ i ].getFrom();
            arr[ i * 3 + 2 ] = Long.toString( sms[ i ].getWhen().getTime(), 10 ); //TODO: make more efficent
            arr[ i * 3 + 3 ] = sms[ i ].getMessage();
          }
          sendString( arr );
        }
      }
    }
  }
  catch ( IOException ex ) {
    ex.printStackTrace();
    sh.remove( this );
  }
}
```



```
void sendString( String s ) throws IOException {
  final char ENDOFMESSAGE = '\0';
  ps.write( ( s + ENDOFMESSAGE ).getBytes() );
  ps.flush();
}

void sendString( String[] ss ) throws IOException {
  final char ENDOFMESSAGE = '\0';
  final char DELIMITER = '\1';
  String s = "";

  for ( int i = 0; i < ss.length; ++i ) {
    s += ss[ i ] + DELIMITER;

  }
  s = s.substring( 0, s.length() - 1 ); //remove last char

  ps.write( ( s + ENDOFMESSAGE ).getBytes() );
  ps.flush();
}

FMMessage getNextMessage() throws IOException {
  final char ENDOFMESSAGE = '\0';
  final char DELIMITER = '\1';

  FMMessage ret = null;

  int c = bis.read();
  String s = new String();

  while ( c != ENDOFMESSAGE ) {
    if ( c == -1 ) {
      return null;
    }
    s += ( char ) c;
    c = bis.read();
  }

  char type = s.charAt( 0 );
  String[] ss = s.substring( 1 ).split( "\\" + DELIMITER );

  if ( type == LoginMessage.TYPE ) {
    ret = new LoginMessage( ss );
  }
  else if ( type == SMSMessage.TYPE ) {
    ret = new SMSMessage( ss );
```

```
    }
    else if ( type == CheckMessage.TYPE ) {
      ret = new CheckMessage();
    }
    else {
      System.err.println( "Unknown type: " + type );
    }
    return ret;
  }
}

interface SocketHandlerInterface {
  void remove( ClientHandler c );
}
```