```
package gratismeldinger.client;

import javax.microedition.lcdui.*;

public interface FMScreen {
  void back( Alert alert );

  void start( Alert alert );
}
```

```
package gratismeldinger.client;

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

public class FMSms {

  private String from;
  private String message;
  private Date when;

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

  public FMSms( byte[] bytes ) {
    ByteArrayInputStream b = new ByteArrayInputStream( bytes );
    DataInputStream bb = new DataInputStream( b );
    try {
      from = bb.readUTF();
      message = bb.readUTF();
      when = new Date( bb.readLong() );
    }
    catch ( IOException ex ) {
      ex.printStackTrace();
    }
  }

  public String getMessage() {
    return message;
  }

  public String getFrom() {
    return from;
  }

  public Date getWhen() {
    return when;
  }

  public byte[] toByte() {
    ByteArrayOutputStream b = new ByteArrayOutputStream();
    DataOutputStream bb = new DataOutputStream( b );

    try {
```



```
      bb.writeUTF( from );
      bb.writeUTF( message );
      bb.writeLong( when.getTime() );
      bb.flush();
    }
    catch ( IOException ex ) {
      System.out.println( "FMSms toByte exception: " + ex );
      ex.printStackTrace();
    }

    return b.toByteArray();

  }

}
```

```
/* TODO:
 Images in list
 About (with version nr)
 Bedre goback (alle klasser har metoden back)
 fix date/time show
 */

package gratismeldinger.client;

import java.util.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class GratisMeldinger
    extends MIDlet
    implements FMScreen {

  private static SocketHandler sh;
  private static Display disp;
  private Gauge gauge;

  public static Settings settings = new Settings();

  public final static boolean DEBUG = true;
  private static GratisMeldinger instance = null;

  private static Stack screenStack = new Stack();

  public GratisMeldinger() {
    instance = this;
    disp = Display.getDisplay( instance );
  }

  public static SocketHandler getSocketHandler() {
    return sh;
  }

  public static GratisMeldinger getInstance() {
    return instance;
  }

  public void destroyApp( boolean unc ) {
    instance = null;
    Display.getDisplay( this ).setCurrent( null );
    notifyDestroyed();
  }
```



```
  public static boolean login() {
    if ( !settings.isSet() ) {
      return false;
    }

    String login = settings.getLoginName();
    String password = settings.getPassword();
    String version = "" + '1';

    String[] send = {
       login, password, version};
    int len = sh.send( 'L', send );

    String[] s = sh.recive();
    return ( s[ 0 ].equals( "1" ) );
  }

  public void startApp() {
    newScreen( new MainMenu(), null );
    newScreen( this, null );
    changeGauge( 5 );
    settings = new Settings();
    changeGauge( 20 );
    if ( DEBUG ) {
      System.out.println( "Connecting..." );
    }
    sh = new SocketHandler();
    changeGauge( 50 );
    while ( !login() ) {
      changeGauge( 75 );
      Alert alert = new Alert( "Ugyldig bruker/passord",
                               "Manglende eller ugyldig innloggings-informasjon!\n",
                               null,
                               AlertType.WARNING );

      newScreen( settings, alert );
      int stackSize = screenStack.size();
      while ( screenStack.size() == stackSize ) {
        Thread.yield();
      }
      changeGauge( 80 );
    }
    changeGauge( 85 );
    Alert alert = new Alert( "Logget inn", "Du er nlogget inn!", null, AlertType.INFO );
    alert.setTimeout( 2000 );
    changeGauge( 95 );
    System.gc();
```

```
    changeGauge( 100 );
    goBack( alert );

  }

  public void pauseApp() {
  }

  public static void goBack( Alert alert ) {
    if ( screenStack.size() <= 1 ) {
      instance.destroyApp( false );
    }
    else {
      screenStack.pop();
      FMScreen screen = ( FMScreen ) screenStack.peek();

      screen.back( alert );
    }
  }

  public static void newScreen( FMScreen screen, Alert alert ) {
    screenStack.push( screen );
    screen.start( alert );
  }

  public void back( Alert alert ) {
    this.destroyApp( false );
  }

  public void start( Alert alert ) {
    Form form = new Form( "Logger inn..." );
    gauge = new Gauge( "0 %", false, 100, 0 );
    form.append( gauge );
    disp.setCurrent( form );
  }

  private void changeGauge( int percent ) {
    gauge.setLabel( percent + " %" );
    gauge.setValue( percent );
  }
}
```

```
package gratismeldinger.client;

import java.util.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;

public class Inbox
    implements FMScreen, CommandListener, RecordListener {

  List list;
  RecordStore res;
  StaticArrayList alist;

  int lastSelected;
  private static final Command backCommand = new Command( "Tilbake", Command.BACK, 3 );
  private static final Command openCommand = new Command( "Åpne", Command.OK, 1 );
  private static final Command answerCommand = new Command( "Svar", Command.SCREEN, 2 );
  private static final Command deleteCommand = new Command( "Slett", Command.SCREEN, 2 );
  private Gauge gauge;
  public Inbox() {

 }

  void makeNewList() {
    list = new List( "Innboks", List.IMPLICIT );
    alist = new StaticArrayList( 20 );
    list.addCommand( backCommand );
    list.addCommand( openCommand );
    list.addCommand( deleteCommand );
    list.setCommandListener( this );
  }

  void loadList() {
    makeNewList();
    changeGauge(25);
    try {
      RecordEnumeration enum = res.enumerateRecords( null, null, false );
      changeGauge(35);
      while ( enum.hasNextElement() ) {
        changeGauge(gauge.getValue()+1);
        int id = enum.nextRecordId();
        alist.insert( id );
        FMSms sms = new FMSms( res.getRecord( id ) );
        changeGauge(gauge.getValue()+1);
        list.append( sms.getFrom(), null );
      }
      changeGauge(60);
```



```
    }
    catch ( Exception ex ) {
      System.out.println( "Error in loadList: " + ex );
      ex.printStackTrace();
    }
  }

  void addMessage( FMSms sms ) {
    byte[] bytes = sms.toByte();
    try {
      res.addRecord( bytes, 0, bytes.length );
    }
    catch ( RecordStoreException ex ) {
      System.out.println( "Error addMessage: " + ex );
    }

  }

  void removeMessage( int id ) {
    try {
      res.deleteRecord( alist.get( id ) );
    }
    catch ( Exception ex ) {
      ex.printStackTrace();
    }
  }

  void getNewMessages() {
    GratisMeldinger.getSocketHandler().send( 'C', null );
    changeGauge(65);
    String[] ss = GratisMeldinger.getSocketHandler().recive();
    changeGauge(75);
    int num = Integer.parseInt( ss[ 0 ], 36 );
    for ( int i = 1; i <= num * 3; i += 3 ) {
      addMessage( new FMSms( ss[ i ], ss[ i + 2 ], new Date( Long.parseLong( ss[ i + 1 ], 10 ) ) ) );
    }
    changeGauge(100);

  }

  public void commandAction( Command command, Displayable displayable ) {
    if ( command == List.SELECT_COMMAND || ( command == openCommand && displayable == list ) )
  {
      try {
        lastSelected = list.getSelectedIndex();
        FMSms sms = new FMSms( res.getRecord( alist.get( lastSelected ) ) );
```

```
      GratisMeldinger.newScreen( new ViewMessage( sms, lastSelected ), null );
    }
    catch ( RecordStoreException ex ) {
      ex.printStackTrace();
    }
  }
  else if ( command == backCommand ) {
    GratisMeldinger.goBack( null );
  }
  else if ( command == deleteCommand ) {
    removeMessage( list.getSelectedIndex() );
    //TODO: beep
  }

}

public void recordAdded( RecordStore recordStore, int int1 ) {
  try {
    FMSms sms = new FMSms( recordStore.getRecord( int1 ) );
    alist.insertAt( 0, int1 );
    list.insert( 0, sms.getFrom(), null );
  }
  catch ( Exception ex ) {
    ex.printStackTrace();
  }

}

public void recordChanged( RecordStore recordStore, int int1 ) {
}

public void recordDeleted( RecordStore recordStore, int int1 ) {
  int val = alist.getRev( int1 );
  list.delete( val );
  alist.delete( val );
}

public void back( Alert alert ) {
  Display.getDisplay( GratisMeldinger.getInstance() ).setCurrent( list );
  list.setSelectedIndex( lastSelected, true );
}

public void start( Alert alert ) {
  StringItem s = new StringItem("Innboks", "Henter innboks...");
  Form form = new Form("Vent..");
  form.append(s);
  gauge = new Gauge( "5 %", false, 100, 0 );
```



```
  form.append( gauge );
  form.setCommandListener(this);
  Display.getDisplay( GratisMeldinger.getInstance() ).setCurrent( form );
  try {
    res = RecordStore.openRecordStore( "Inbox", true );
    changeGauge(10);
    res.addRecordListener( this );
  }
  catch ( Exception ex ) {
    System.err.println( ex + " in Inbox CTOR" );
    ex.printStackTrace();
  }
  changeGauge(15);
  loadList();
  getNewMessages();
  Display.getDisplay( GratisMeldinger.getInstance() ).setCurrent( list );
}

private void changeGauge( int percent ) {
  gauge.setLabel( percent + " %" );
  gauge.setValue( percent );
}



class ViewMessage
    implements FMScreen, CommandListener {
  int num;
  FMSms sms;
  Form f;

  ViewMessage( FMSms sms, int num ) {
   this.sms = sms;
    Calendar cal = Calendar.getInstance();
    cal.setTime( sms.getWhen() );
    String when = "" + cal.get( Calendar.DAY_OF_MONTH ) + "." + cal.get( Calendar.MONTH ) + "." +
       cal.get( Calendar.YEAR ) + "  " + cal.get( Calendar.HOUR ) + ":" + cal.get( Calendar.MINUTE );
    StringItem b = new StringItem( sms.getFrom(), sms.getMessage() ); // \n i neg. av medling?
    f = new Form(when);
    f.append( b );
    f.addCommand( backCommand );
    f.addCommand( answerCommand );
    f.addCommand( deleteCommand );
    f.setCommandListener( this );
    this.num = num;
  }

  public void back( Alert alert ) {
```

```
      start( alert );
    }

    public void start( Alert alert ) {
      if ( alert == null ) {
        Display.getDisplay( GratisMeldinger.getInstance() ).setCurrent( f );
      }
      else {
        Display.getDisplay( GratisMeldinger.getInstance() ).setCurrent( alert, f );
      }
    }

    public void commandAction( Command command, Displayable displayable ) {
      if ( command == backCommand ) {
        GratisMeldinger.goBack( null );
      }
      else if ( command == deleteCommand ) {
        removeMessage( num );
        Alert alert = new Alert( "Melding slettet", "Melding Slettet!", null, AlertType.CONFIRMATION );
        alert.setTimeout( 2000 );
        GratisMeldinger.goBack( alert );
      }
      else if ( command == answerCommand ) {
        GratisMeldinger.newScreen( new NewMessage( "", sms.getFrom() ), null );
      }
    }

  }
}
```

```
package gratismeldinger.client;

import javax.microedition.lcdui.*;

public class MainMenu
    implements FMScreen, CommandListener {

  private static final Command sendCommand = new Command( "Send", Command.SCREEN, 1 );
  private static final Command okCommand = new Command( "OK", Command.OK, 1 );
  private static final Command cancelCommand = new Command( "Tilbake", Command.CANCEL, 2 );
  private static final Command breakCommand = new Command( "Linjeskift", Command.SCREEN, 2 );

  private static final String[] menuNames = {
      "Ny melding",
      "Innboks",
      "Utboks",
      "Instillinger",
      "Avslutt",
  };
  private List selectionList = new List( "GratisMelding", List.IMPLICIT, menuNames, null );

  public MainMenu() {
    selectionList.setCommandListener( this );
  }

  public void back( Alert alert ) {
    start( alert );
  }

  public void start( Alert alert ) {
    if ( alert == null ) {
      Display.getDisplay( GratisMeldinger.getInstance() ).setCurrent( selectionList );
    }
    else {
      Display.getDisplay( GratisMeldinger.getInstance() ).setCurrent( alert, selectionList );
    }
  }

  public void commandAction( Command command, Displayable displayable ) {
    if ( selectionList.getSelectedIndex() == menuNames.length - 1 ) {
      StringItem s = new StringItem(null, "Avslutter...");
      Form form = new Form("Avlsutter");
      form.append(s);
      Display.getDisplay( GratisMeldinger.getInstance() ).setCurrent( form );
      GratisMeldinger.goBack( null );
    }
    else if ( selectionList.getSelectedIndex() == 0 ) {
```



```
      GratisMeldinger.newScreen( new NewMessage( "", "" ), null );
    }
    else if ( selectionList.getSelectedIndex() == 1 ) {

      GratisMeldinger.newScreen( new Inbox(), null );
    }
    else if ( selectionList.getSelectedIndex() == 3 ) {
      GratisMeldinger.newScreen( GratisMeldinger.settings, null );
    }
  }

}
```

```
package gratismeldinger.client;

import javax.microedition.lcdui.*;

public class NewMessage
    implements FMScreen, CommandListener {

  private static final Command sendCommand = new Command( "Send", Command.OK, 1 );
  private static final Command okCommand = new Command( "Send", Command.OK, 1 );
  private static final Command backCommand = new Command( "Tilbake", Command.BACK, 2 );
  private static final Command clearCommand = new Command( "T skjerm", Command.SCREEN, 2 );

  TextBox text;

  Form form;
  TextField numField;
  ChoiceGroup cg;

  Display disp;
  String sNum;

  public NewMessage( String txt, String num ) {
    text = new TextBox( "GratisMelding", "", 500, TextField.ANY );
    text.addCommand( sendCommand );
    text.addCommand( backCommand );
    text.setCommandListener( this );
    System.err.println( "Text done" );
    sNum = num;
    disp = Display.getDisplay( GratisMeldinger.getInstance() );

  }

  public void commandAction( Command command, Displayable displayable ) {
    if ( command == sendCommand ) {
      GratisMeldinger.newScreen( new SendMessage(), null );
    }
    else if ( command == backCommand ) {
      GratisMeldinger.goBack( null );
    }
    else if ( command == clearCommand ) {
      text.setString( "" );
    }
  }

  public void back( Alert alert ) {
    start( alert );
  }
```



```
  public void start( Alert alert ) {
    if ( alert == null ) {
      disp.setCurrent( text );
    }
    else {
      disp.setCurrent( alert, text );
    }
  }

class SendMessage
    implements FMScreen, CommandListener {
  SendMessage() {
    if ( sNum == null ) {
      sNum = "";
    }
    form = new Form( "Mottaker" );
    numField = new TextField( "Send til:", sNum, 11, TextField.PHONENUMBER );
    String ss[] = {
        "Leveringsrapport"};
    cg = new ChoiceGroup( "Annet: ", ChoiceGroup.MULTIPLE, ss, null );

    form.append( numField );
    form.append( cg );
    form.setCommandListener( this );
    form.addCommand( okCommand );
    form.addCommand( backCommand );

  }

  public void commandAction( Command command, Displayable displayable ) {
    if ( command == okCommand ) {
      SocketHandler sh = GratisMeldinger.getSocketHandler();
      if ( numField.getString().startsWith( "+47" ) ) {
        numField.setString( numField.getString().substring( 3 ) );

      }
      String[] ss = {
          text.getString(),
          numField.getString(),
          cg.isSelected( 0 ) ? "1" : "0"
      };
      sh.send( 'M', ss );
      ss = sh.recive();
      Alert alert;
      if ( ss[ 0 ].equals( "1" ) ) {
        alert = new Alert( "Melding sendt", "Melding sendt!", null, AlertType.CONFIRMATION );
```

```
        }
        else {
          alert = new Alert( "Melding ikke sendt!", ss[ 1 ], null, AlertType.ERROR );
        }
        alert.setTimeout( 2000 );
        GratisMeldinger.goBack( alert );
      }
    }

    public void back( Alert alert ) {
      start( alert );
    }

    public void start( Alert alert ) {
      if ( alert == null ) {
        disp.setCurrent( form );
      }
      else {
        disp.setCurrent( alert, form );
      }
    }
  }
}
```

```
package gratismeldinger.client;

import java.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;

/*1. record: loginname
  2. record: password
 */
public class Settings
    implements RecordListener,
    CommandListener,
    FMScreen {

  private static final Command backCommand = new Command( "Tilbake", Command.BACK, 2 );
  private static final Command saveCommand = new Command( "Lagre", Command.OK, 2 );

  private Form form;
  private TextField userField;
  private TextField passwordField;

  ByteArrayOutputStream bout = new ByteArrayOutputStream();
  DataOutputStream dout = new DataOutputStream( bout );
  RecordStore res;

  private String user;
  private String password;

  public Settings() {
    try {
      res = RecordStore.openRecordStore( "Settings", false );
      user = new String( res.getRecord( 1 ) );
      password = new String( res.getRecord( 2 ) );
      if ( user != null && user.equals( "null" ) ) {
        user = null;
      }
      if ( password != null && password.equals( "null" ) ) {
        password = null;

      }
    }
    catch ( RecordStoreException ex ) {
      try {
        res = RecordStore.openRecordStore( "Settings", true );
        res.addRecord( null, 0, 0 );
        res.addRecord( null, 0, 0 );
      }
```



```
      catch ( RecordStoreException ex1 ) {
      }
    }
    catch ( NullPointerException ex ) {
      ex.printStackTrace();
      System.err.println( "NPE at SettingsCtor" );
    }
  }

  public void start( Alert alert ) {
    if ( form == null ) { //not init
      form = new Form( "Instillinger" );
      userField = new TextField( "Telefonnummer: ", user, 8, TextField.NUMERIC );
      passwordField = new TextField( "Passord", password, 20, TextField.PASSWORD );
      form.append( userField );
      form.append( passwordField );
      form.addCommand( backCommand );
      form.addCommand( saveCommand );
      form.setCommandListener( this );
    }
    Display disp = Display.getDisplay( GratisMeldinger.getInstance() );
    if ( alert != null ) {
      disp.setCurrent( alert, form );
    }
    else {
      disp.setCurrent( form );

    }
  }

  /*Returns the user's loginname (his telephonenumber */
  String getLoginName() {
    return user;
  }

  /*Returns the user's password */
  String getPassword() {
    return password;
  }

  /*Checks if settings has been set.*/
  boolean isSet() {
    return! ( user == null || user.length() == 0 || password == null || password.length() == 0 );
  }

  public void recordAdded( RecordStore recordStore, int recordId ) {
    //Do the same as recordChanged
```

```
  recordChanged( recordStore, recordId );
 }

 public void recordChanged( RecordStore recordStore, int recordId ) {
  String what;
  try {
   if ( recordId == 1 ) {
    user = new String( recordStore.getRecord( recordId ) );
   }
   else if ( recordId == 2 ) {
    password = new String( recordStore.getRecord( recordId ) );
   }
  }
  catch ( RecordStoreException ex1 ) {
  }

 }

 public void recordDeleted( RecordStore recordStore, int recordId ) {
  if ( recordId == 1 ) {
   user = null;
  }
  else if ( recordId == 2 ) {
   password = null;
  }
 }

 public void commandAction( Command command, Displayable displayable ) {
  if ( command == backCommand ) {
   GratisMeldinger.goBack( null );
  }
  else if ( command == saveCommand ) {
   try {
    byte[] b = userField.getString().getBytes();
    res.setRecord( 1, b, 0, b.length );
    b = passwordField.getString().getBytes();
    res.setRecord( 2, b, 0, b.length );
    GratisMeldinger.goBack( new Alert( "Lagret", "Instillingene er nlagret", null,
AlertType.CONFIRMATION ) );
   }
   catch ( RecordStoreException ex ) {
   }

  }
 }

 public void back( Alert alert ) {
```



```
   Display disp = Display.getDisplay( GratisMeldinger.getInstance() );
   if ( alert != null ) {
    disp.setCurrent( alert, form );
   }
   else {
    disp.setCurrent( form );

   }
  }
 }
```

```
package gratismeldinger.client;

import java.io.*;
import javax.microedition.io.*;

public class SocketHandler {
  OutputStream os;
  InputStream is;
  StreamConnection con;

  int totalSendt = 0;
  int totalRecived = 0;

  public SocketHandler() {
    try {
      con = ( StreamConnection ) Connector.open( "socket://reistadbakk.com:25" );
      os = con.openOutputStream();
      is = con.openInputStream();
    }
    catch ( Exception ex ) {
      System.err.println( "Exception at SocketHandler: " + ex );
      //show alert and exit
      // System.exit( 1 );
    }
  }

  /*Sends a message to the server.
   @Param type The messagetype.
   @Param arguments for the type
   @Returns Number of bytes sent
   */
  public int send( char type, String[] args ) {
    final char ENDOFMESSAGE = '\0';
    final char DELIMITER = '\1';

    String send = "";
    System.out.println( "Sending.." );

    send += type;
    if ( args != null && args.length != 0 ) {
      send += args[ 0 ];
      System.out.println( "0:" + args[ 0 ] );
      for ( int i = 1; i < args.length; ++i ) {
        System.out.println( i + ":" + args[ i ] );
        send += DELIMITER;
        send += args[ i ];
      }
```



```
    }

    send += ENDOFMESSAGE;

    byte bytes[] = send.getBytes();

    try {
      os.write( bytes );
    }
    catch ( IOException ex ) {
      return -1;
    }
    totalSendt += bytes.length;

    return bytes.length;
  }

  public String[] recive() {
    final char ENDOFMESSAGE = '\0';
    final char DELIMITER = '\1';

    java.util.Vector v = new java.util.Vector();
    System.out.println( "Reciving: " );

    try {
      int c = is.read();
      ++totalRecived;
      String s = new String();

      while ( c != ENDOFMESSAGE ) {
        if ( c == DELIMITER ) {
          v.addElement( s );
          System.out.print( s + " - " );
          s = "";
        }
        else {
          s += ( char ) c;
        }
        c = is.read();
        ++totalRecived;
      }
      if ( s.length() != 0 ) {
        System.out.print( s + " - " );
        v.addElement( s );
      }
      v.trimToSize();
    }
```

```
    catch ( IOException ex ) {
      System.err.println( "IOException at recive: " + ex );
      return null;
    }
    System.out.println();
    String ss[] = new String[ v.size() ];
    v.copyInto( ss );
    return ss;
  }

}
```

```
package gratismeldinger.client;

public class StaticArrayList {
  int[] arr;
  int cursor = 0;

  public StaticArrayList( int capacity ) {
    arr = new int[ capacity ];
    for ( int i = 0; i < capacity; ++i ) {
      arr[ i ] = -1;
    }
  }

  void insert( int val ) {
    for ( int i = arr.length - 1; i > cursor; --i ) {
      arr[ i ] = arr[ i - 1 ];

    }
    arr[ cursor++ ] = val;

  }

  void insertAt( int loc, int val ) {
    cursor = loc;
    insert( val );
  }

  void set( int loc, int val ) {
    arr[ loc ] = val;
  }

  void delete( int loc ) {
    for ( int i = loc; i < arr.length - 1; ++i ) {
      arr[ i ] = arr[ i + 1 ];

    }
    arr[ arr.length - 1 ] = -1;
  }

  int get( int loc ) {
    return arr[ loc ];
  }

  int getRev( int val ) {
    for (int i = 0; i < arr.length; ++i)
      if (arr[i] == val)
        return i;
```



```
    return -1;
  }

}
```