```
import com.google.gson.Gson;
import com.google.gson.annotations.SerializedName;

import javax.net.ssl.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.URL;
import java.security.GeneralSecurityException;
import java.util.ArrayList;

public class Program {
    //Адрес метода отправки СМС
    private static final String SEND_SMS_URI = "https://my-sms.ru/rest/sms/send/json";
    //Адрес метода проверки статуса СМС
    private static final String CHECK_SMS_URI = "https://my-sms.ru/rest/sms/status/json";
    //Адрес метода проверки баланса пользователя
    private static final String GET_BALANCE_URI = "https://my-sms.ru/rest/sms/credit/json";

    //Имя пользователя
    private static final String USERNAME = "username";
    //Пароль
    private static final String PASSWORD = "password";

    //Зарегистрированное имя отправителя СМС
    private static final String SMS_SOURCE_NAME = "Test SMS";
    //Код ответа сервиса, соответствующий успешному принятию СМС к отправке
    //(полный список статусов отправки сообщений содержится в документации к API)
    private static final String SMS_SUCCESSFULLY_QUEUED_CODE = "2";

    public static void main(String[] args) {
        Gson gson = new Gson();

        //Изменение метода проверки сертификата для корректной работы с самоподписанным сертификатом сервиса
        setTrustManager();

        //Формирование объекта с авторизационными данными
        AuthData authData = new AuthData();
        authData.username = USERNAME;
        authData.password = PASSWORD;

        //Формирование массива сообщений
        MessageData[] messages = new MessageData[2];
        MessageData testMessage = new MessageData();
        testMessage.source = SMS_SOURCE_NAME;
        testMessage.destination = "79112223333";
        testMessage.text = "Test SMS";
        messages[0] = testMessage;
        messages[1] = testMessage;

        //Формирование и отправка запроса проверки баланса пользователя
        GetBalanceRequest getBalanceRequest = new GetBalanceRequest();
        getBalanceRequest.authData = authData;
        String result = sendRequest(GET_BALANCE_URI, gson.toJson(getBalanceRequest));
        GetBalanceResponse getBalanceResponse = gson.fromJson(result, GetBalanceResponse.class);
        System.out.printf("Your account balance: %s\n", getBalanceResponse.balance);

        //Формирование и отправка запроса отправки пакета СМС
        SendSMSRequest sendSMSRequest = new SendSMSRequest();
        sendSMSRequest.authData = authData;
        sendSMSRequest.messages = messages;
        System.out.println("\nMessage sending in progress...");
        result = sendRequest(SEND_SMS_URI, gson.toJson(sendSMSRequest));
        SendSMSResponse sendSMSResponse = gson.fromJson(result, SendSMSResponse.class);
        System.out.println("Sending statuses:");
        ArrayList<MessageIdentifier> acceptedMessagesIdentifiers = new ArrayList<>(messages.length);
        //Печать статусов отправки сообщений
        for (MessageStatus status : sendSMSResponse.messagesStatuses) {
            System.out.printf("Message ID: %s, State: %s\n",
                    status.Id, ("".equals(status.State)) ? "N/A" : status.State);
            //Сохранение идентификаторов сообщений успешно принятых к отправке
            if (SMS_SUCCESSFULLY_QUEUED_CODE.equals(status.State)) {
                MessageIdentifier messageIdentifier = new MessageIdentifier();
                messageIdentifier.Id = status.Id;
                acceptedMessagesIdentifiers.add(messageIdentifier);
            }
        }

        //Формирование и отправка запроса проверки статуса пакета СМС
        CheckSMSRequest checkSMSRequest = new CheckSMSRequest();
        checkSMSRequest.authData = authData;
        checkSMSRequest.messagesIds = acceptedMessagesIdentifiers.toArray(
                new MessageIdentifier[acceptedMessagesIdentifiers.size()]);
        System.out.println("\nMessage status checking in progress...");
        result = sendRequest(CHECK_SMS_URI, gson.toJson(checkSMSRequest));
```

```
    CheckSMSResponse checkSMSResponse = gson.fromJson(result, CheckSMSResponse.class);
    System.out.println("Message statuses:");
    //Печать статусов сообщений
    for (MessageStatus status : checkSMSResponse.messagesStatuses) {
        System.out.printf("Message ID: %s, State: %s\n",
                status.Id, ("".equals(status.State)) ? "N/A" : status.State);
    }
}

//Метод отправки POST запроса с указанными параметрами
private static String sendRequest(String uri, String data) {
    try {
        URL url = new URL(uri);
        HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
        conn.setDoOutput(true);
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/json; charset=utf-8");
        String input = data;
        OutputStream os = conn.getOutputStream();
        os.write(input.getBytes());
        os.flush();
        if (conn.getResponseCode() != HttpsURLConnection.HTTP_OK) {
            throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
        }
        BufferedReader br = new BufferedReader(new InputStreamReader(
                (conn.getInputStream())));
        String output;
        if ((output = br.readLine()) == null) {
            throw new IOException();
        }
        return output;
    } catch (IOException e) {
        System.err.println("An error occurred while processing request");
        return "";
    }
}

//DataContracts для сериализации/десериализации объектов данных переданных/полученных от сервиса

public static class SendSMSRequest {
    @SerializedName("auth")
    public AuthData authData;

    @SerializedName("data")
    public MessageData[] messages;
}

public static class CheckSMSRequest {
    @SerializedName("auth")
    public AuthData authData;

    @SerializedName("data")
    public MessageIdentifier[] messagesIds;
}

public static class GetBalanceRequest {
    @SerializedName("auth")
    public AuthData authData;
}

public static class SendSMSResponse {
    @SerializedName("state")
    public String requestStatus;

    @SerializedName("data")
    public MessageStatus[] messagesStatuses;
}

public static class CheckSMSResponse {
    @SerializedName("state")
    public String requestStatus;

    @SerializedName("data")
    public MessageStatus[] messagesStatuses;
}

public static class GetBalanceResponse {
    @SerializedName("state")
    public String requestStatus;

    @SerializedName("balance")
    public String balance;
}

public static class AuthData {
    @SerializedName("username")
```

```
        public String username;

        @SerializedName("password")
        public String password;
    }

    public static class MessageData {
        public MessageData() {
            SetDefaults();
        }

        @SerializedName("source")
        public String source;

        @SerializedName("destination")
        public String destination;

        @SerializedName("text")
        public String text;

        @SerializedName("date")
        public String date;

        @SerializedName("ttl")
        public String ttl;

        private void SetDefaults() {
            source = "";
            destination = "";
            text = "";
            date = "";
            ttl = "";
        }
    }

    public static class MessageIdentifier {
        @SerializedName("id")
        public String Id;
    }

    public static class MessageStatus {
        @SerializedName("id")
        public String Id;

        @SerializedName("state")
        public String State;
    }

    private static void setTrustManager() {
        TrustManager[] trustAllCerts = new TrustManager[]{
                new X509TrustManager() {
                    public java.security.cert.X509Certificate[] getAcceptedIssuers() {
                        return null;
                    }

                    public void checkClientTrusted(java.security.cert.X509Certificate[] certs, String authType) {
                    }

                    public void checkServerTrusted(java.security.cert.X509Certificate[] certs, String authType) {
                    }
                }
        };

        try {
            SSLContext sc = SSLContext.getInstance("SSL");
            sc.init(null, trustAllCerts, new java.security.SecureRandom());
            HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
            HttpsURLConnection.setDefaultHostnameVerifier(new HostnameVerifier() {
                public boolean verify(String hostname, SSLSession session) {
                    return true;
                }
            }
            );

        } catch (GeneralSecurityException e) {
            e.printStackTrace();
        }
    }
}
```