```
using System;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Security;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Json;
using System.Security.Cryptography.X509Certificates;

namespace MySMSRestClient
{
    class Program
    {
        //Адрес метода отправки СМС
        private const string SEND_SMS_URI = "https://my-sms.ru/rest/sms/send/json";
        //Адрес метода проверки статуса СМС
        private const string CHECK_SMS_URI = "https://my-sms.ru/rest/sms/status/json";
        //Адрес метода проверки баланса пользователя
        private const string GET_BALANCE_URI = "https://my-sms.ru/rest/sms/credit/json";

        //Имя пользователя
        private const string USERNAME = "username";
        //Пароль
        private const string PASSWORD = "password";

        //Зарегистрированное имя отправителя СМС
        private const 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 const string SMS_SUCCESSFULLY_QUEUED_CODE = "2";

        static void Main(string[] args)
        {
            //Изменение метода проверки сертификата для корректной работы с самоподписанным сертификатом сервиса
            ServicePointManager.ServerCertificateValidationCallback = (obj, certificate, chain, errors) =>
                    ((errors & SslPolicyErrors.RemoteCertificateChainErrors) != 0) ?
                    ((certificate.Subject == certificate.Issuer) && (chain.ChainStatus.All(c =>
                            (c.Status == X509ChainStatusFlags.PartialChain ||
                            c.Status == X509ChainStatusFlags.UntrustedRoot ||
                            c.Status == X509ChainStatusFlags.NoError)))) : true;
            //Формирование объекта с авторизационными данными
            AuthData auth = new AuthData { Username = USERNAME, Password = PASSWORD };

            //Формирование массива сообщений
            MessageData[] messages = new MessageData[2];
            MessageData testMessage = new MessageData
            {
                Source = SMS_SOURCE_NAME,
                Destination = "79112223333",
                Text = "Test SMS"
            };
            messages[0] = testMessage;
            messages[1] = testMessage;

            try
            {
                //Формирование и отправка запроса проверки баланса пользователя
                HttpWebRequest getBalanceRequest = CreateRequest(GET_BALANCE_URI,
                        new GetBalanceRequest { Auth = auth });
                HttpWebResponse response = (HttpWebResponse)getBalanceRequest.GetResponse();
                GetBalanceResponse getBalanceResponse = HandleResponse<GetBalanceResponse>(response);
                Console.WriteLine("Your account balance: {0}", getBalanceResponse.Balance);

                //Формирование и отправка запроса отправки пакета СМС
                HttpWebRequest sendSMSRequest = CreateRequest(SEND_SMS_URI,
                        new SendSMSRequest { Auth = auth, Messages = messages });
                Console.WriteLine("\nMessage sending in progress...");
                response = (HttpWebResponse)sendSMSRequest.GetResponse();
                SendSMSResponse sendSMSResponse = HandleResponse<SendSMSResponse>(response);
                Console.WriteLine("Sending statuses:");
                //Печать статусов отправки сообщений
                sendSMSResponse.MessagesStatuses.ToList().ForEach(s =>
                        Console.WriteLine("Message ID: {0}, State: {1}", ("".Equals(s.Id)) ? "N/A" : s.Id, s.State));
                //Сохранение идентификаторов сообщений успешно принятых к отправке
                MessageIdentifier[] acceptedMessagesIdentifiers = sendSMSResponse.MessagesStatuses.Where(s =>
                        s.State == SMS_SUCCESSFULLY_QUEUED_CODE).Select(s =>
                                new MessageIdentifier { Id = s.Id }).ToArray();

                //Формирование и отправка запроса проверки статуса пакета СМС
                HttpWebRequest checkSMSRequest = CreateRequest(CHECK_SMS_URI,
                        new CheckSMSRequest { Auth = auth, MessagesIds = acceptedMessagesIdentifiers });
                Console.WriteLine("\nMessage status checking in progress...");
                response = (HttpWebResponse)checkSMSRequest.GetResponse();
                CheckSMSResponse checkSMSResponse = HandleResponse<CheckSMSResponse>(response);
                Console.WriteLine("Message statuses:");
                //Печать статусов сообщений
```

```
            checkSMSResponse.MessagesStatuses.ToList().ForEach(s =>
                    Console.WriteLine("Message ID: {0}, State: {1}", ("".Equals(s.Id)) ? "N/A" : s.Id, s.State));
        }
        catch (WebException e)
        {
            Console.WriteLine("An error occurred while processing request\n" + e.Message);
        }
        Console.ReadLine();
    }

    //Метод создания POST запроса с указанными параметрами
    static HttpWebRequest CreateRequest(string uri, object requestDataObject)
    {
        HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(new Uri(uri));
        request.ContentType = "application/json; charset=utf-8";
        request.Method = "POST";
        DataContractJsonSerializer serializer = new DataContractJsonSerializer(requestDataObject.GetType());
        using (Stream stream = request.GetRequestStream())
        {
            serializer.WriteObject(stream, requestDataObject);
            stream.Flush();
        }
        return request;
    }

    //Метод обработки ответа сервиса
    //Возращает объект ответа заданного типа
    private static T HandleResponse<T>(HttpWebResponse response)
    {
        if (200 != (int)response.StatusCode)
        {
            throw new WebException(String.Format("The remote server returned an unexpected response: ({0}) {1}",
                    (int)response.StatusCode, response.StatusDescription), WebExceptionStatus.ProtocolError);
        }
        else
        {
            DataContractJsonSerializer serializer = new DataContractJsonSerializer(typeof(T));
            using (Stream stream = response.GetResponseStream())
            {
                return (T)serializer.ReadObject(stream);
            }
        }
    }
}

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

#region DataContracts Requests

[DataContract(Name = "sendSMS")]
public class SendSMSRequest
{
    [DataMember(Name = "auth")]
    public AuthData Auth { get; set; }

    [DataMember(Name = "data")]
    public MessageData[] Messages { get; set; }
}

[DataContract(Name = "checkSMS")]
public class CheckSMSRequest
{
    [DataMember(Name = "auth")]
    public AuthData Auth { get; set; }

    [DataMember(Name = "data")]
    public MessageIdentifier[] MessagesIds { get; set; }
}

[DataContract(Name = "getBalance")]
public class GetBalanceRequest
{
    [DataMember(Name = "auth")]
    public AuthData Auth { get; set; }
}

#endregion

#region DataContracts Responses

[DataContract(Name = "sendSMS")]
public class SendSMSResponse
{
    [DataMember(Name = "state", Order = 0)]
    public string RequestStatus { get; set; }
```

```
    [DataMember(Name = "data", Order = 1)]
    public MessageStatus[] MessagesStatuses { get; set; }
}

[DataContract(Name = "checkSMS")]
public class CheckSMSResponse
{
    [DataMember(Name = "state", Order = 0)]
    public string RequestStatus { get; set; }

    [DataMember(Name = "data", Order = 1)]
    public MessageStatus[] MessagesStatuses { get; set; }
}

[DataContract(Name = "getBalance")]
public class GetBalanceResponse
{
    [DataMember(Name = "state", Order = 0)]
    public string RequestStatus { get; set; }

    [DataMember(Name = "balance", Order = 1)]
    public string Balance { get; set; }
}

#endregion

#region DataContracts Members

[DataContract]
public class AuthData
{
    [DataMember(Name = "username", Order = 0)]
    public string Username { get; set; }

    [DataMember(Name = "password", Order = 1)]
    public string Password { get; set; }
}

[DataContract(Name = "message")]
public class MessageData
{
    public MessageData()
    {
        SetDefaults();
    }

    [DataMember(Name = "source", Order = 1)]
    public string Source { get; set; }

    [DataMember(Name = "destination", Order = 2)]
    public string Destination { get; set; }

    [DataMember(Name = "text", Order = 3)]
    public string Text { get; set; }

    [DataMember(Name = "date", Order = 4)]
    public string Date { get; set; }

    [DataMember(Name = "ttl", Order = 5)]
    public string TTL { get; set; }

    private void SetDefaults()
    {
        Source = "";
        Destination = "";
        Text = "";
        Date = "";
        TTL = "";
    }
}

[DataContract(Name = "message")]
public class MessageIdentifier
{
    [DataMember(Name = "id")]
    public string Id { get; set; }
}

[DataContract(Name = "message")]
public class MessageStatus
{
    [DataMember(Name = "id", Order = 0)]
    public string Id { get; set; }

    [DataMember(Name = "state", Order = 1)]
    public string State { get; set; }
```

```
    }

    #endregion
}
```