软件验证模块项目文档

1 概述

目前大部分商业软件在提供注册码时，基本上采用了以下几种机制来实现：

- **远程联网激活。**在软件每次启动时，都会联网检查软件使用情况，检查使用时间是否 到期，是否是多设备使用等。这种方式控制是非常棒的，动态性、实时性都非常好， 但劣势也非常明显，就是客户机软件要联网，如若没有联网就无法控制了。
- **本地生成注册。**这种方式是根据客户机的环境，获取客户机的信息，比如硬盘、 MAC 地址 、 CPU 等硬件信息，根据一定的算法将这些信息生成一个注册码。目前超过一半 的软件都是采用这种方式来实现的，这种方式的缺点是不能自由地控制软件的其它参 数，比如软件中可添加设备的数量。
- **配套密钥文件。**在软件发行的过程中，用软件运行到期时间、运行数量限制和已运行 时间等参数生成一个密钥文件，配套发送给用户使用。在软件启动时，直接加载这个 密钥文件进行检查。这种方式的缺点在于密钥文件的参数选择上不好把控，若只仅仅 设置运行到期时间，用户可以轻松修改电脑时间来获取更长使用时间（在不联网同步 时间的情况下）。

而本项目采用的方法是本地生成注册, 即一机一码注册方式, 根据唯一的机器码对应生成注册码。

项目可以按应用端分成两个部分:

- 客户端（提供机器码的部分）
- 注册机（根据机器码计算其对应注册码）

核心技术是通过JAVA类库获取硬盘, CPU, 主板SN码 和 MAC 地址, 分别截取其中一部分字符拼接起来得到该机器的机器码。注册码则是在该机器机器码后面加上长度为8位的日期（如20221206)作为过期时间。验证过程是将接受到的注册码分离出机器码+日期两部分, 比对机器码与本机机器码是否一致, 若一致则注册码成立。

若采用明文传输, 用户可以很简单的修改后面的日期以达到无限注册, 所以需要采取加密手段保证验证码的保密性。我们采取**RSA+BASE64**加密来实现。

而对于如何实现在打开时自动读取之前的注册状态, 不需要重复注册的功能, 我们采用生成密钥配置文件, 即在软件目录下生成 `config.properties` 文件, 在软件启动时读取其中保存的注册码, 再次进行验证操作。

2 工程目录

```
.
├── main/java/javaproject
|    ├── authen // 验证模块
|    ├── machine // 获取机器码模块
|    ├── menu  // 前端模块
├── resources // 存放资源文件, 和验证状态保存
├── pom.xml
```

3 验证模块authen

```
├──authen // 验证模块
|    ├── client // 客户端验证方法
|    ├── server  // 注册机验证方法
```

3.1 client

该类中存放客户端会用到的静态方法。

包含以下内容：

- 将传入的密文注册码用公钥解密并返回明文注册码（机器码+日期）

```
public static String decodeAthuncode(String authencode)
```

- 传入一个字符串日期"20221023", 判断是否过期, 即在当天之前

```
public static boolean isExpired(String date)
```

- 将从注册码中提取的机器码与本机机器码进行比对

```
public static boolean isValid(String deCodeMachineCode, String machineCode)
```

- 得到明文注册码的机器码部分

```
public static String getMachineCode(String deCode)
```

- 得到明文注册码的日期部分

```
public static String getDate(String deCode)
```

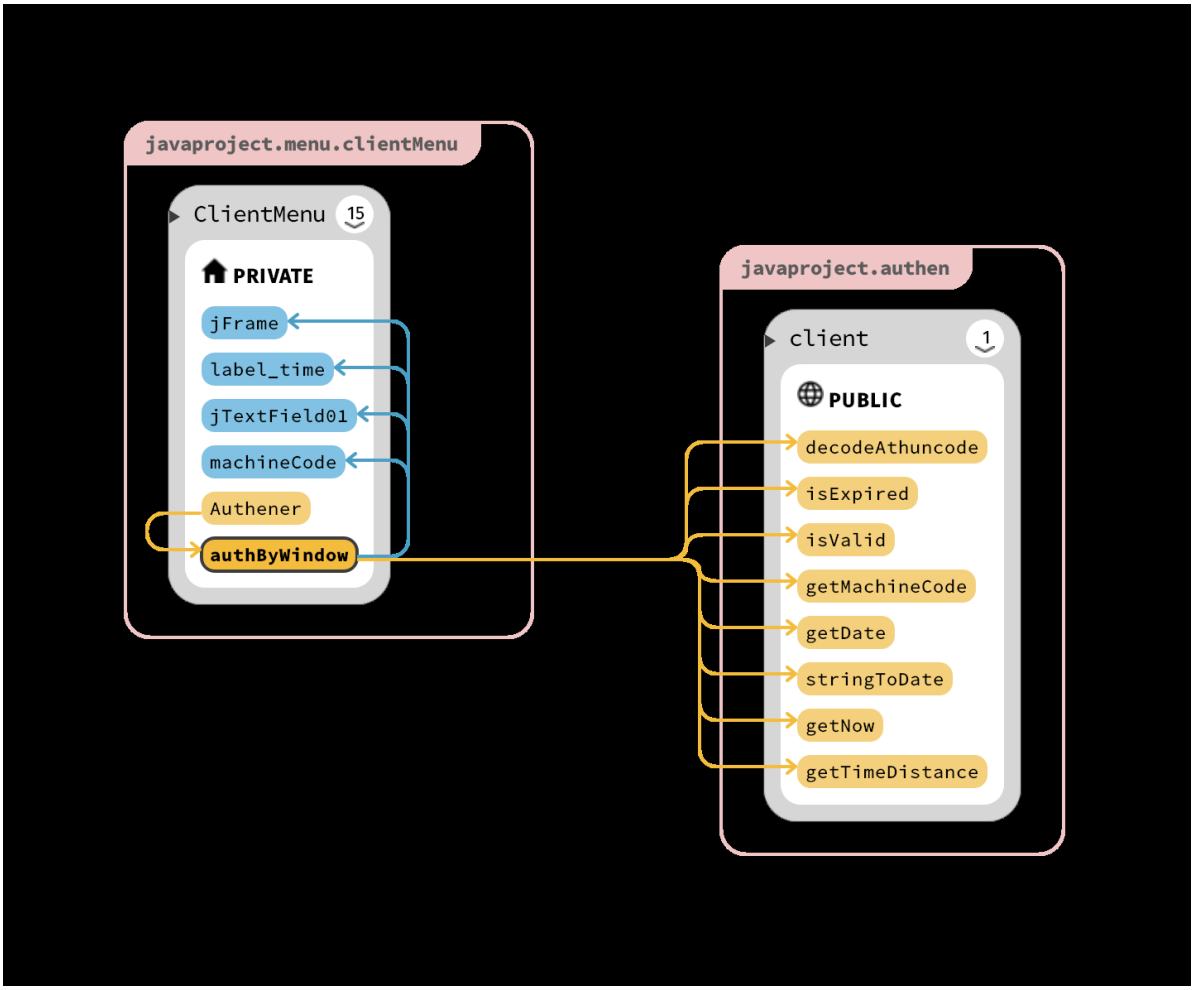
- 将字符串形式的日期转换成`Date`对象

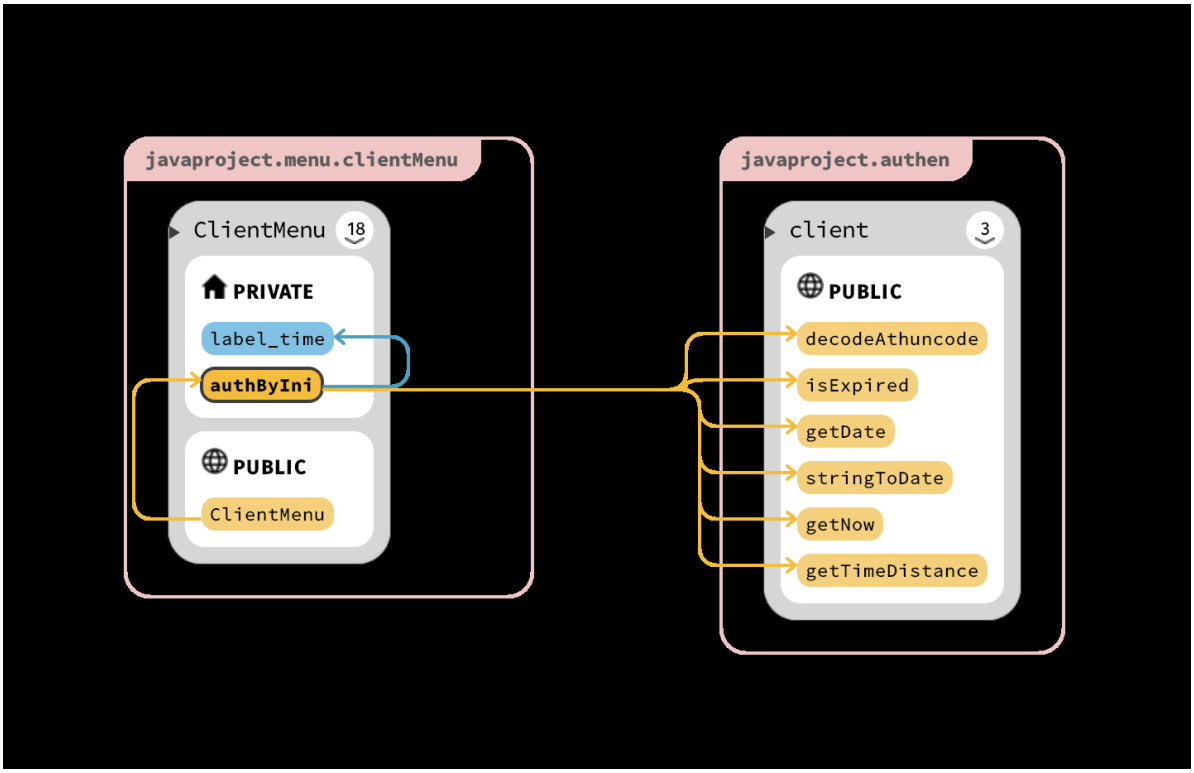
```
public static Date stringToDate(String date)
```

- 得到两个日期之间差距天数, 用于实现显示“注册剩余时间"

```
public static int getTimeDistance(Date beginDate , Date endDate )
```

使用关系如下：





3.2 Server

该类存放注册机所需要的静态方法。

包含以下内容：

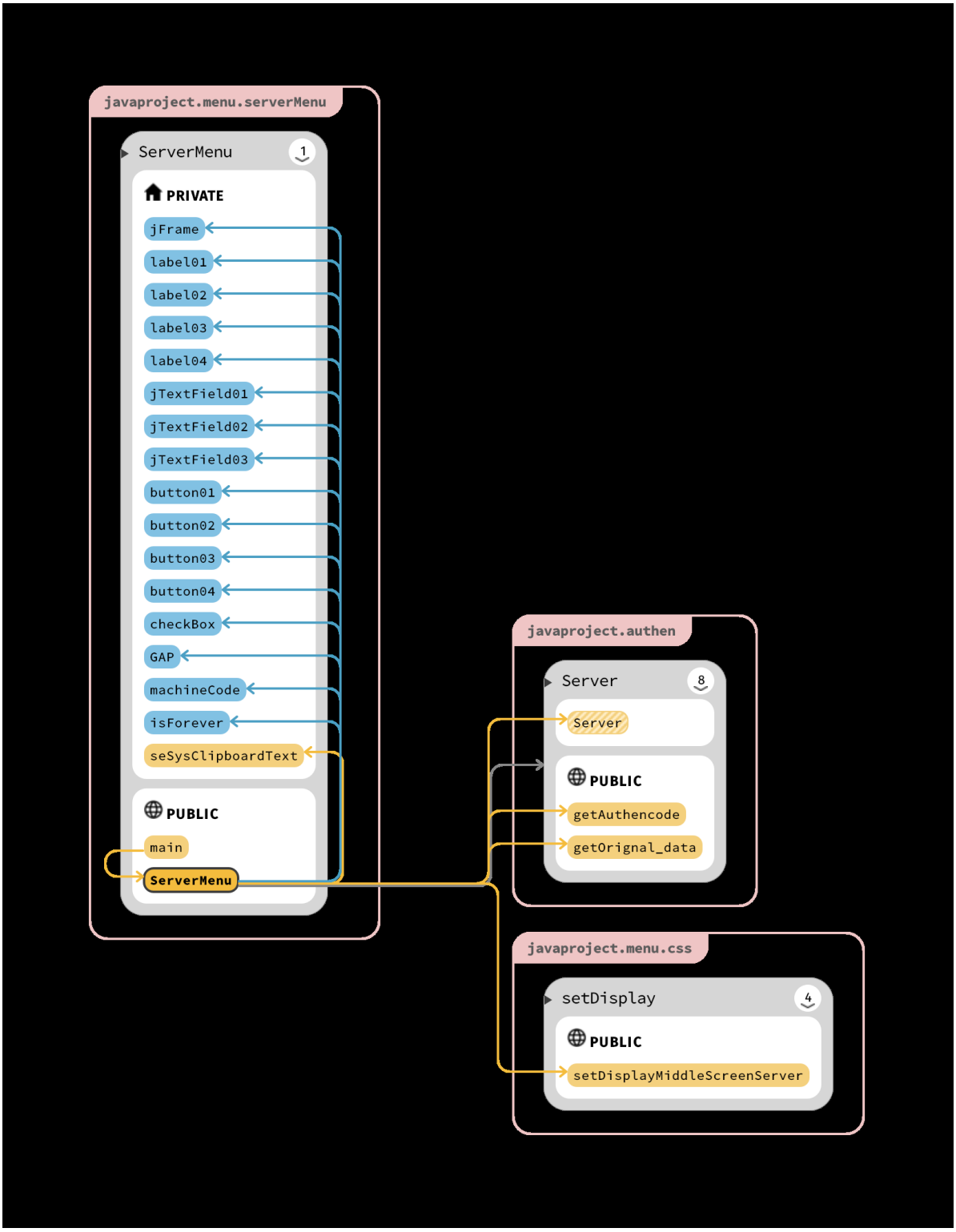
- 将明文通过私钥加密为密文

```
public String getAuthencode(String orignal_data)
```

- 将输入的日期和机器码拼接为注册码

```
public String getOrignal_data(int year, int month, int day, String machine)
```

使用关系如下：



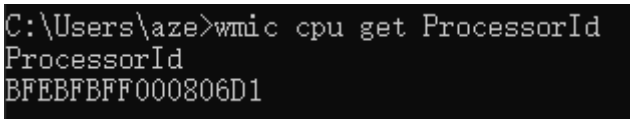
4 机器码模块

```
├── machine // 验证模块
|    ├── CPU // 获取CPU SN码
|    ├── DiskUils  // 获取硬盘 SN码
|    ├── MotherboardSN  // 获取主板 SN码
|    ├── MacTools  // 获取MAC码
|    ├── Machine  // 将获取到的码汇总截取一部分组成机器码
```

4.1 CPU

通过`Runtime.getRuntime().exec()`方法, 执行CMD命令 `wmic cpu get ProcessorId`之类并通过`Scanner`读取`process.getInpuStream`流的结果。

该CMD命令效果：



`BFEBFBFF000806D1`就是我们需要的SN码。

4.2 DiskUils

采用运行vbs脚本的方法来获取。创建一个临时的vbs文件如下：

```
Set objFSO = CreateObject("Scripting.FileSystemObject") '利用FSO对象操作
Set colDrives = objFSO.Drives '返回可用的驱动器列表
Set objDrive = colDrives.item(drive) '获取drive驱动器的信息
Wscript.Echo objDrive.SerialNumber '控制台输出SN码
```

然后通过上一步类似的方法`Runtime.getRuntime().exec()`, 来进行调用并获取结果。

4.3 MotherboardSN

一样使用vbs脚本来获取。

```
Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")
Set colItems = objWMIService.ExecQuery _
("Select * from Win32_BaseBoard")
For Each objItem in colItems
    Wscript.Echo objItem.SerialNumber
    exit for  ' do the first cpu only!
Next
```

然后通过上一步类似的方法`Runtime.getRuntime().exec()`, 来进行调用并获取结果。

4.4 MacTools

NetworkInterface是在JDK1.4是添加的一个类,使用 `getNetworkInterfaces` 方法即可得到当前机器上所有的网络接口。

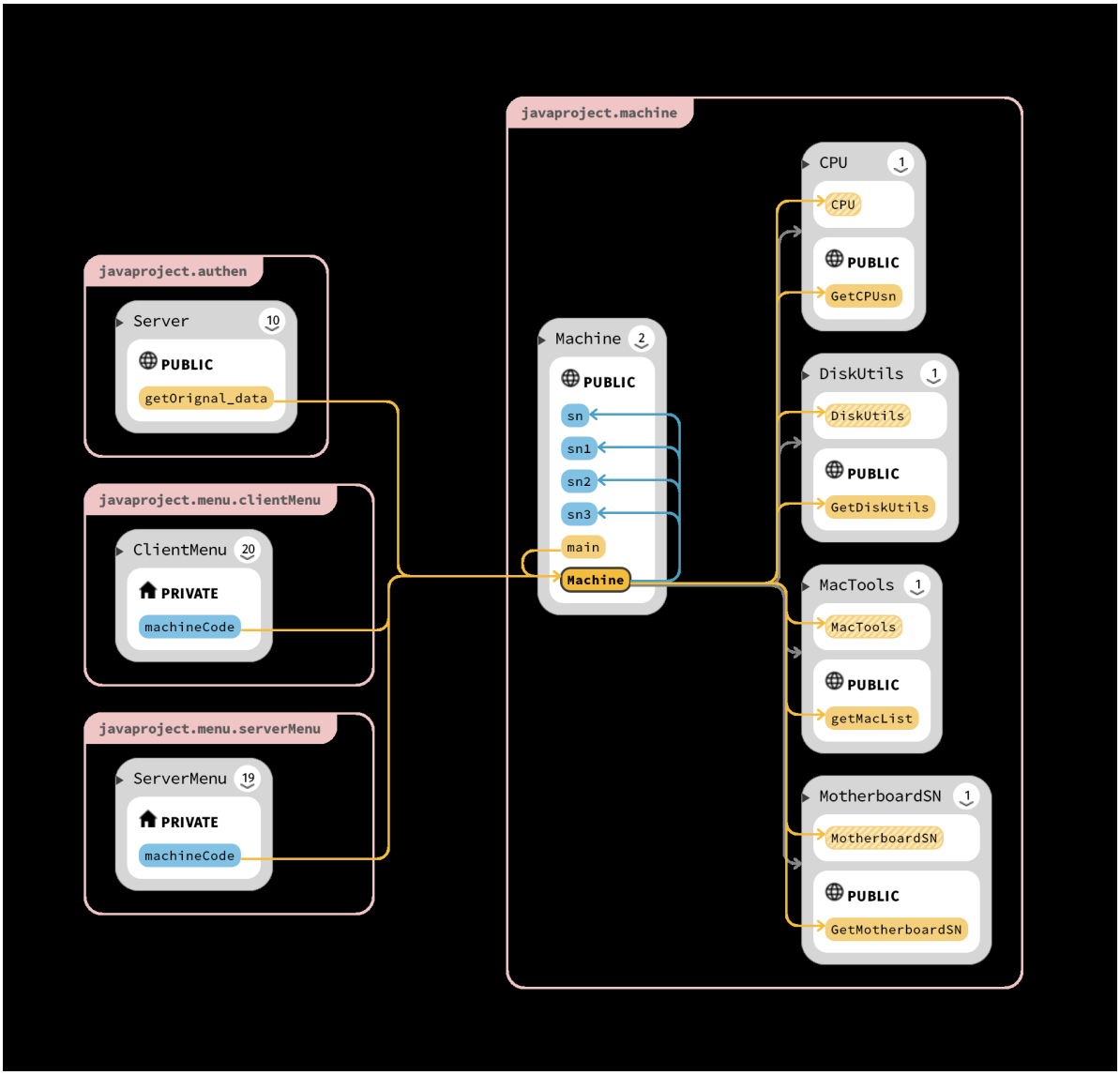
细节请查看docx目录下的MAC physical address.md文件。

4.5 Machine

- 获取机器信息并拼接成一个机器码返回

```
public String getMachineCode()
```

使用关系如下：



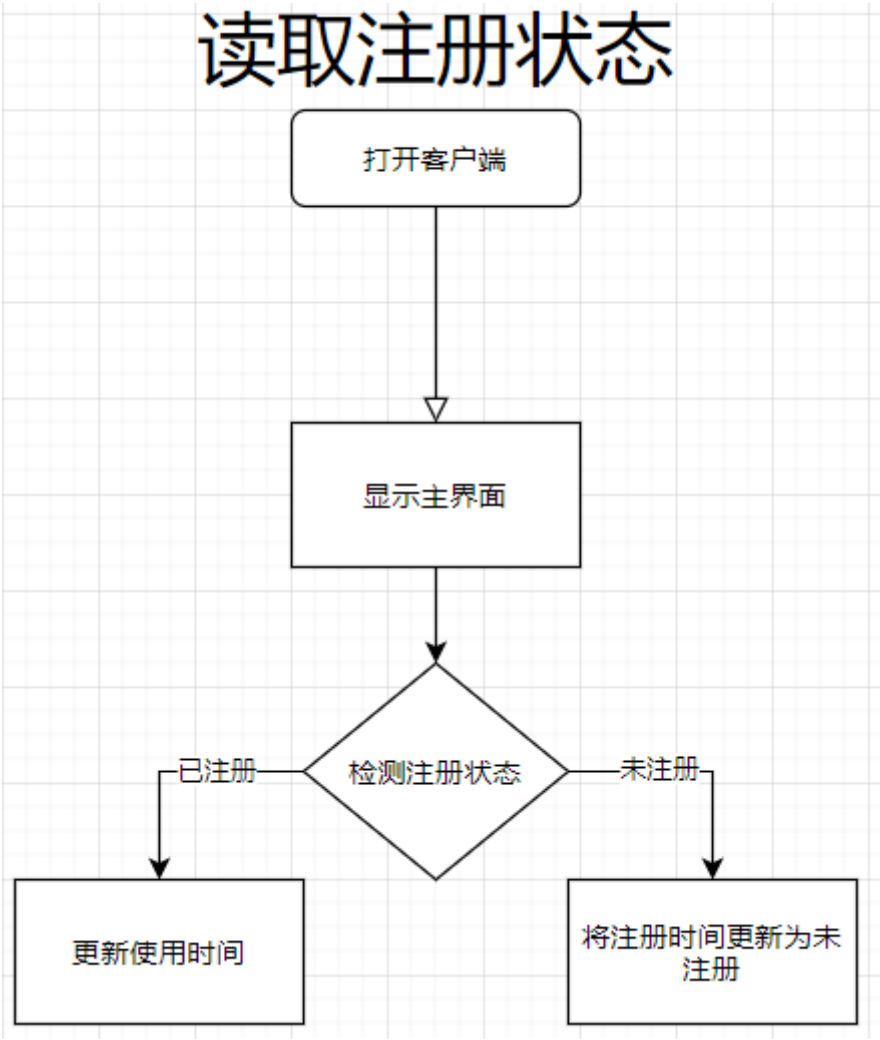
5 menu

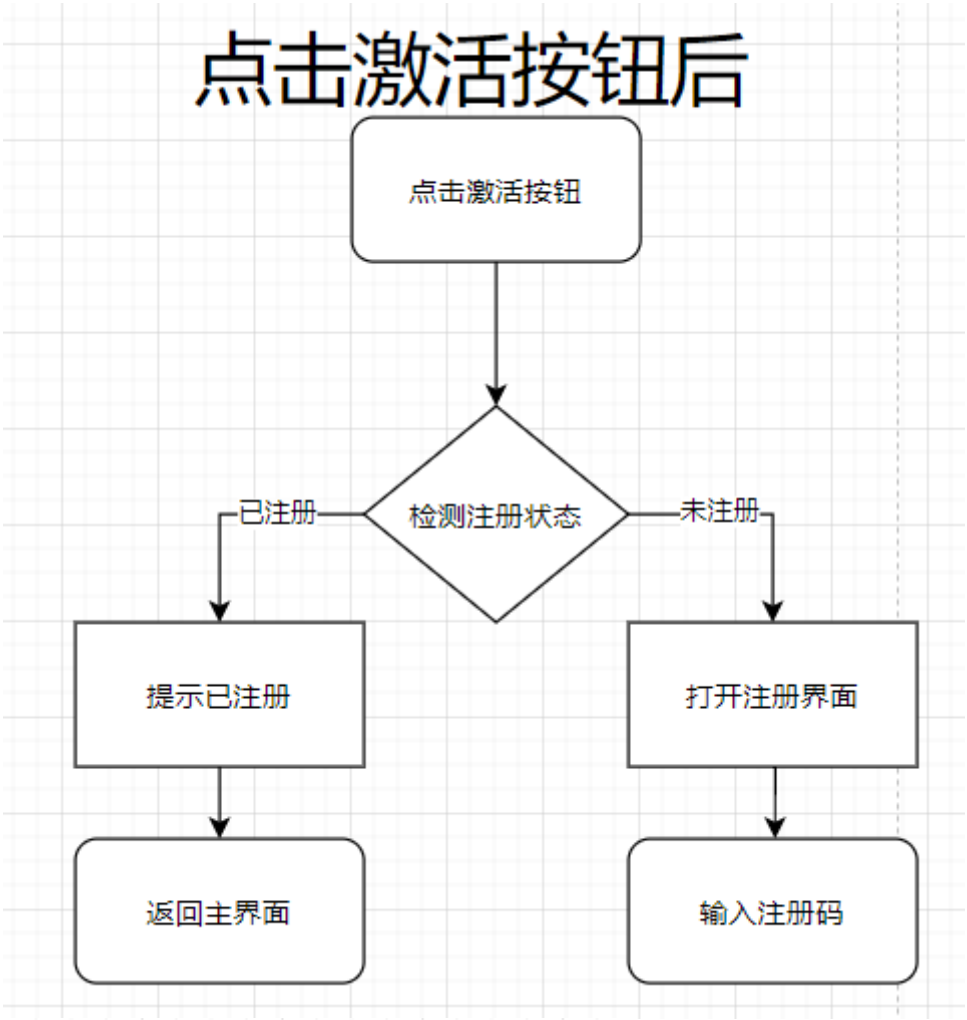
```
├── menu // 前端模块
|    ├── clientMenu // 客户端功能实现
|    ├── css  // 设定窗口大小
|    ├── serverMenu  // 注册机功能实现
```

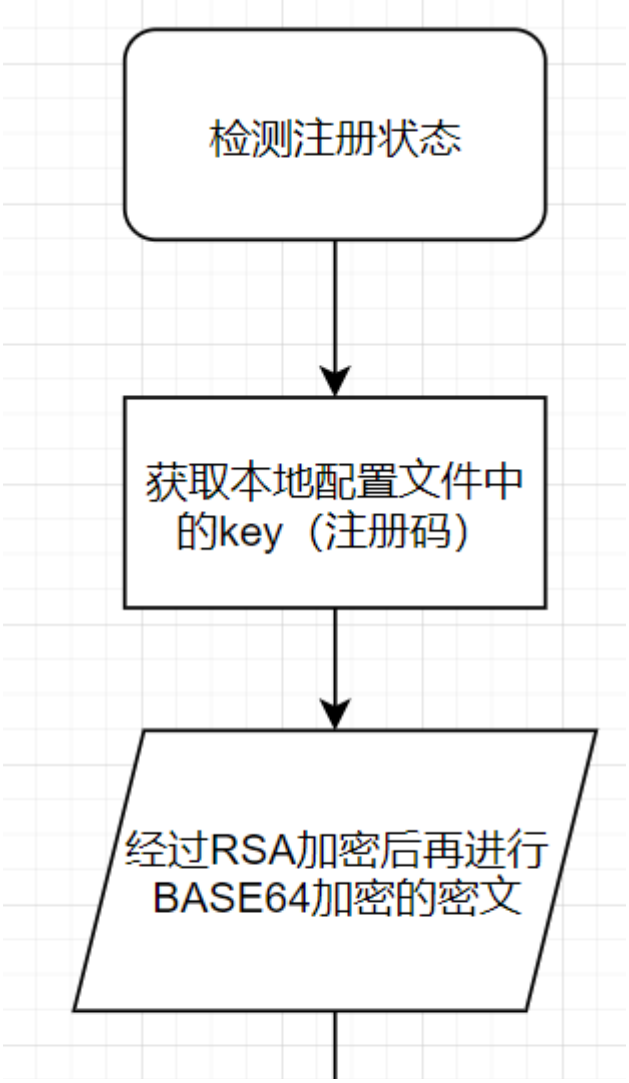
5.1 clientMenu

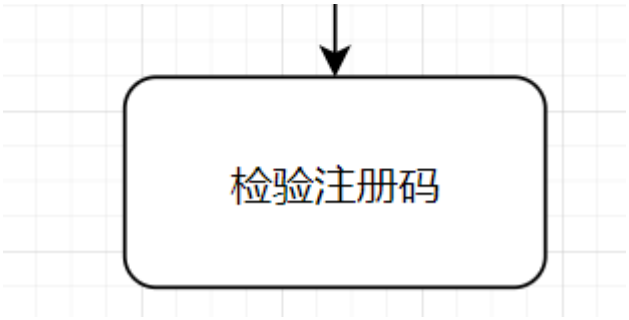
使用JFrame类库实现GUI界面, 采用Group对齐方式排版, 同时使用FlatLaf类库来美化界面。

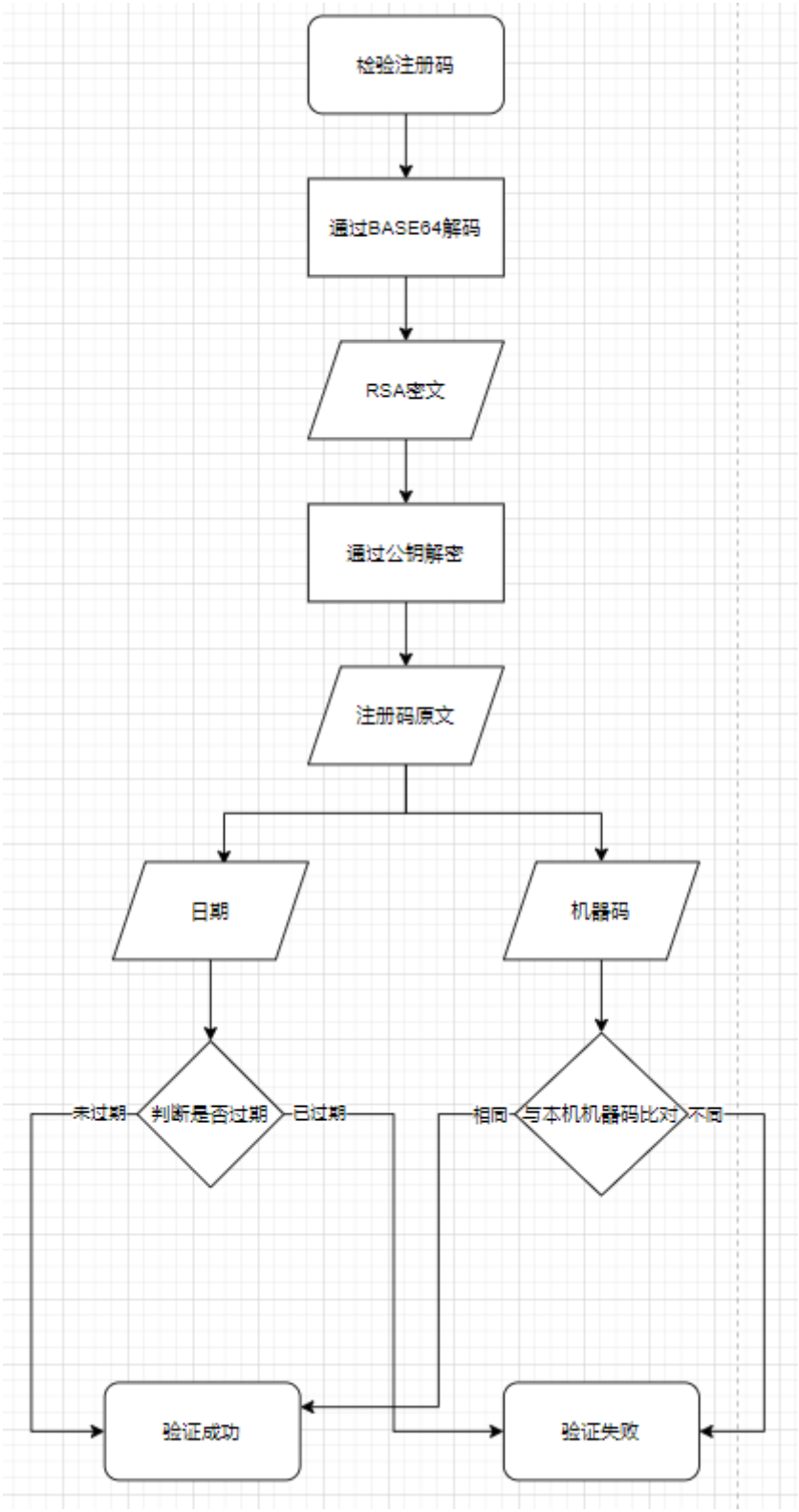
运行流程如图：

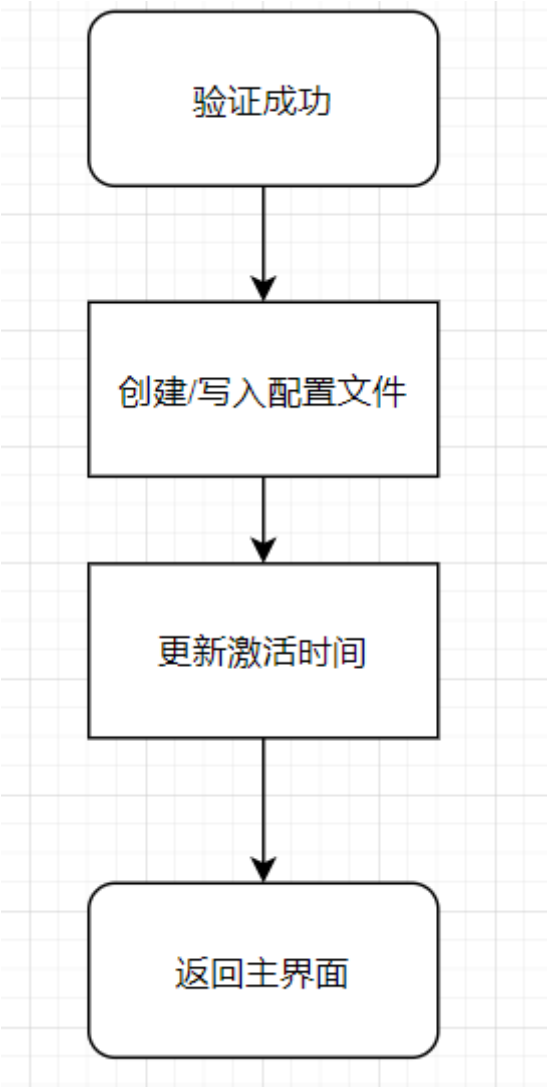

读取注册状态
打开客户端
显示主界面
检测注册状态
已注册
未注册
更新使用时间
将注册时间更新为未注册


点击激活按钮后
点击激活按钮
检测注册状态
已注册
未注册
提示已注册
打开注册界面
返回主界面
输入注册码


检测注册状态
获取本地配置文件中的key（注册码）
经过RSA加密后再进行BASE64加密的密文


检验注册码


检验注册码
通过BASE64解码
RSA密文
通过公钥解密
注册码原文
日期
机器码
判断是否过期
未过期
已过期
与本机机器码比对
相同
不同
验证成功
验证失败



5.2 serverMenu

根据机器码和设定的到期日期来计算出注册码