苏 州 大 学 实 验 报 告

院、系	计算机学院	年级专业	19 计科图灵	姓名	张昊	学号	1927405160
课程名称	Java 程序设计					成绩	
指导教师	孔芳	同组实验者	无		实验日期	2021 年 5 月 25 日	

实 验 名 称　智能设备统一管理系统

一、需求分析

智能家居是一个融合先进自动化系统的家居平台，为用户提供对家居环境及设备的精密监控和远程控制智能设备管理系统是连接用户与智能设备的桥梁，承载着智能设备系统数据存储、逻辑处理及系统扩展的功能，在智能家居设备系统中的作用举足轻重。

本管理系统着眼于各类智能设别的统一管理和控制等功能基于 Java 的实现，淡化各智能设备与主机（上位机）的连接、通信等 IoT 领域的细节问题。

（一）功能需求

智能设备的统一管理系统应具有如下功能。

1. 实现统一的设备模型与接口：

现今市面上的设备种类繁多，做好设备的统一管理是关键问题。依托面向对象的设计思路，将实体的智能设备通过多级抽象形成系统中的逻辑设备，使得设备统一管理具有了模型上的基础。

2. 实现智能家居设备系统的数据存储：

智能家居系统需要维持长期稳定的运行状态。实际中，各种智能设备会产生大量的数据，这些数据如果使用文件系统来管理将使得程序读写文件的逻辑变得十分困难，无形中增加了系统开发的难度。而基于数据库的管理系统可以实现更可靠的数据完整性，以及并发访问性能。

3. 实现智能家居设备系统的业务逻辑处理：

管理系统承载着用户与智能设备从发出到反馈的一系列业务逻辑处理。智能管理系统不仅仅是数据的承载者，它具有一套完备的逻辑系统：从用户登录到向设备发出指令，再到向数据库请求数据，到最后的结果反馈与可视化展示，能够实现用户登录，设备通信，可视化展示，日志记录等一系列的功能。

4. 提供友好的图形用户界面：

智能设别的管理系统是一个重要的人机接口。友好的图形用户界面为用户提供了从查询到管理一整套的简单易用的操作逻辑。

（二）性能需求

智能设备的统一管理系统性能具体要求如下。

1. 系统健壮性：智能设备管理系统能够保持长期稳定运行的系统，并在故障发生后能够尽可能降低故障破坏性。

2. 系统扩展性：智能设备管理系统应具备足够的扩展空间，能够灵活满足用户不同需求，并满足日后系统升级的需求。

3. 系统安全性：用户使用账号密码登录该系统，设备归属不同用户，具有用户层面的隔离性与独立性。

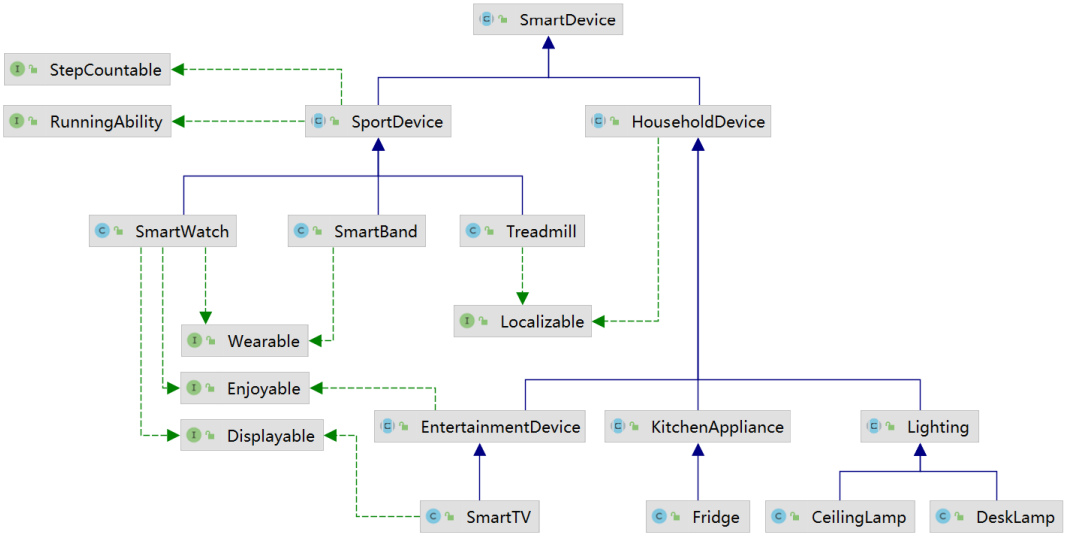
4. 系统易用性：智能家居管理系统要求操作简单、逻辑合理、简约一致。

二、详细设计方案

1. 多级设备抽象

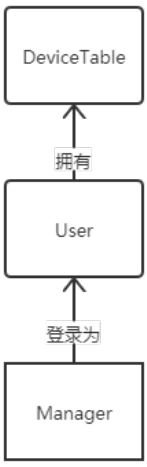
根据目前市面上的智能设备，为各种智能设备进行了多级抽象。以 SmartDevice 类（智能设备）为抽象基类，添加智能设备共有的功能：连接、断开、获取基础属性，所有具体的智能设备都继承自 SmartDevice 类。并且对智能设备进行了分类，分为运动设备（SportDevice 类）、家居设计（HouseholdDevice 类），在各子类中分别添加属于该种设备的功能。更进一步，在各自的类别下进行更细致的划分，直至具体到某一特殊的设备。如下图实现所示，整个设备的多级别抽象利用到了面向对象思想中的继承派生，将各级设备组织成树形结构，叶节点为具体的设备对应的类，非叶节点为设备所属类别对应的抽象类。在 Java 包层面，所有的抽象基类位于 manager.device.base 包内，代表具体设备类型的各类位于 manager.device 包内。

除此之外，将设备所具有的功能抽象为能力（Ability），使用接口来表示。使用以下接口来表示各种能力：计步能力（StepCountable），跑步运动能力（RunningAbility），固定位置能力（Localizable），可穿戴能力（Wearable），娱乐能力（Enjoyable），屏幕可显示能力（Displayable）。如下图虚线所示，各类根据实际功能实现各接口，以完成不同功能。在 Java 包层面，各设备能力接口位于 manager.device.ability 包内。



2. 用户账户类与统一管理类设计

在 manager.account 包内，设计了三个类，分别为用户设备表类（DeviceTable）、用户类（User）以及统一管理类（Manager），三者的关系为单向关联关系，如右图所示。统一管理类与用户类关联，即用户通过登录动作登录到管理类；用户类与用户设备表类关联，即每个用户实例持有一个设备表实例。

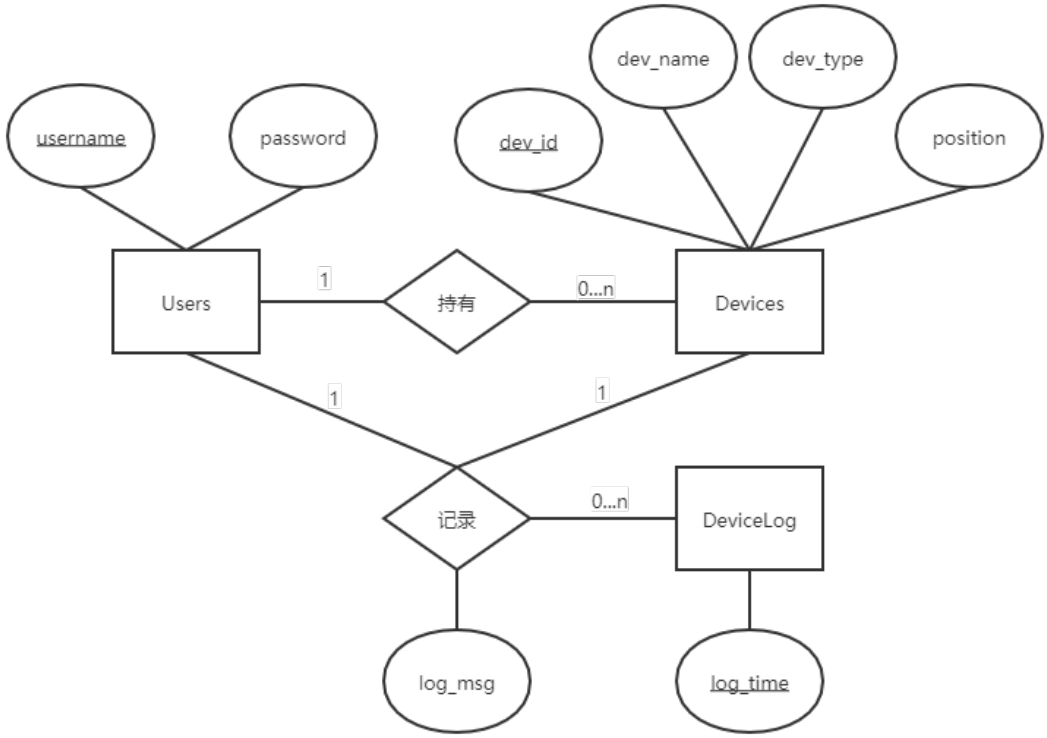


管理类的生命周期为整个图形用户界面的存在周期，负责用户、数据库的访问。统一管理类还是连接用户（包括其设备表）、数据库以及用户图形界面的桥梁，因此在类中提供各种数据访问方法，供有关类调用。

由于智能设备存在不同的使用场景和场合，本系统中基于设备的位置属性，设计了场景功能，用户可以根据使用场景不同来筛选设备，并展示给用户。

3. 基于数据库的数据存储方案

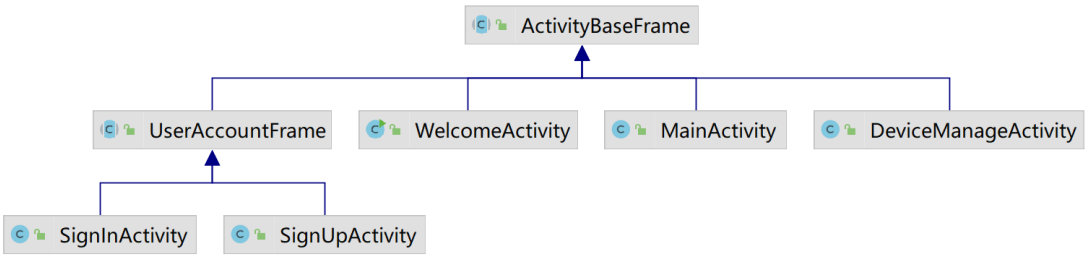
为实现系统中更可靠的数据完整性及并发访问性能，设计了一个数据库来保存用户、设备数据以及日志。根据上述逻辑关系设计数据库 E-R 模型，如下所示：



其中，Users 实体对应 User 类，Devices 对应 manager.device 包下各类，DeviceLog 实体用于表示智能设备产生的日志，通过实时的更删改查操作来实时维护与调用。

4. 图形用户界面设计

为用户设计了一套从查询到管理的简单易用的图形用户界面。将各个显示页面抽象为 Activity，并设计一个抽象基类提供一致的显示逻辑和共享的常量定义，保证全局 UI 的一致性。并且通过继承与派生，实现各页面的显示逻辑。类的继承关系如下图所示：



三、具体实现

1. 各级设备、接口实现

各级智能设备与设备能力接口的实现细节如下表所示：

类名	**SmartDevice**	所属包	manager.devices.base

父类	Object	接口实现	无
访问控制	public abstract	功能函数	public abstract void connect(); public abstract void disconnect();
说明	智能设备基类，提供所有智能设备在连接和断开时都需要使用的方法，作为抽象方法。		
接口名	**Displayable**	所属包	manager.devices.ability

访问控制	public	功能函数	void startPlayingVideo();
			void stopPlayingVideo();
			boolean isPlayingVideo();

说明	可显示设备能力接口，提供显示（以播放视频为例）所需调用的方法。		
接口名	**Enjoyable**	所属包	manager.devices.ability

访问控制	public	功能函数	void setMusic(String music);
			String getMusic();
			void startPlayingMusic();
			void stopPlayingMusic();
			boolean isPlayingMusic();

说明	娱乐能力接口，以播放音乐为例，提供娱乐能力所需调用的方法。		
接口名	**Localizable**	所属包	manager.devices.ability
访问控制	public	功能函数	String getPosition();
说明	设备固定位置能力接口，预置了常量数组 POSITIONS，内含位置常量，并提供查询位置信息的方法。后期将根据实现此接口的类的对象提供的位置信息进行场景设备的分类。		
接口名	**RunningAbility**	所属包	manager.devices.ability

访问控制	public	功能函数	void startRunning();
			double stopRunning();
			boolean isRunning();

说明	跑步能力接口，提供了实现跑步能力所需调用的方法。		
接口名	**StepCountable**	所属包	manager.devices.ability
访问控制	public	功能函数	int getSteps(); void resetStepCounter();
说明	记步能力接口，提供了实现计步能力所需调用的方法。		
接口名	**Wearable**	所属包	manager.devices.ability
访问控制	public	功能函数	double sleepQuality(); int realTimeHeartRate();
说明	可穿戴设备能力接口，提供了睡眠质量和实时心率检测的方法。		
类名	**SportDevice**	所属包	manager.devices.base
父类	SmartDevice	接口实现	StepCountable, RunningAbility
访问控制	public abstract	功能函数	无新增
说明	运动健身设备基类，对计步能力和跑步能力进行了实现。		
类名	**HouseholdDevice**	所属包	manager.devices.base
父类	SmartDevice	接口实现	Localizable
访问控制	public abstract	功能函数	public void open(); public void close();

			public boolean isOpened();
说明	家居设备基类，实现固定位置能力，并添加了打开/关闭电源状态的描述。		
类名	**EntertainmentDevice**	所属包	manager.devices.base
父类	HouseholdDevice	接口实现	Enjoyable
访问控制	public abstract	功能函数	无新增
说明	娱乐设备基类，实现了娱乐能力。		
类名	**KitchenAppliance**	所属包	manager.devices.base
父类	HouseholdDevice	接口实现	无
访问控制	public abstract	功能函数	public abstract Food find(String name); public abstract void put(Food food);
说明	厨房电器基类，继承自家居设备类，提供装入食物和查找食物的抽象方法。并提供内部类 Food，描述了可以装入电器的食物的基本属性。		
类名	**Lighting**	所属包	manager.devices.base
父类	HouseholdDevice	接口实现	无
访问控制	public abstract	功能函数	public abstract Color getColor();
说明	照明电器基类，继承自家居设备类，并重写了电器电源的打开与关闭方法。提供获取灯的颜色的抽象方法。		
类名	**SmartBand**	所属包	manager.devices
父类	SportDevice	接口实现	Wearable
访问控制	public	功能函数	无新增
说明	智能手环类，为运动设备基类的特化，实现设备连接方法，并实现了可穿戴能力的各个方法。		
类名	**SmartWatch**	所属包	manager.devices
父类	SportDevice	接口实现	Wearable, Enjoyable, Displayable
访问控制	public	功能函数	无新增
说明	智能手表类，为运动设备基类的特化，实现设备连接方法，并实现了可穿戴能力、娱乐能力（音乐播放）、可显示能力（播放视频）的各个方法。		
类名	**Treadmill**	所属包	manager.devices
父类	SportDevice	接口实现	Localizable
访问控制	public	功能函数	无新增
说明	跑步机类，为运动设备基类的特化，实现设备连接方法，并实现了固定位置能力的各个方法。		
类名	**CeilingLamp**	所属包	manager.devices
父类	Lighting	接口实现	无
访问控制	public	功能函数	无新增
说明	吸顶灯类，为照明电器基类的特化，实现设备连接方法。设置默认颜色为白色，且不提供修改方法。		
类名	**DeskLamp**	所属包	manager.devices
父类	Lighting	接口实现	无
访问控制	public	功能函数	void setColor(Color color);
说明	台灯类，为照明电器基类的特化，实现设备连接方法。默认颜色为白色，提供修改照明颜色方法。		
类名	**Fridge**	所属包	manager.devices

父类	KitchenAppliance	接口实现	无
访问控制	public	功能函数	public int getFrozenTemperature(); public void setFrozenTemperature(int t);
说明	智能冰箱类，为厨房电器基类的特化，实现设备连接方法。并提供设置/获取冷冻温度的方法。		
类名	**SmartTV**	所属包	manager.devices
父类	EntertainmentDevice	接口实现	Displayable
访问控制	public	功能函数	无新增
说明	智能电视类，为娱乐设备基类的特化，实现设备连接方法，并实现了显示设备能力的各方法。		

另外还实现了一个工厂类 SmartDeviceFactory，为实用工具类，根据传入的类型参数创建对象，在数据库中数据到 Java 对象的转换时使用。

2. 数据库实现

考虑到程序为单机运行，且规模不算大，故采用嵌入式数据库 SQLite，数据库文件保存至 data 文件夹下 data.db 文件中。并使用 JDBC 连接数据库，在程序启动（类装载时）自动初始化连接数据库。根据设计阶段对数据库的概要设计以及 E-R 关系图，创建表的 SQL 语句如下：

```
CREATE TABLE IF NOT EXISTS USERS (
       USERNAME VARCHAR(64) PRIMARY KEY NOT NULL,
       PASSMD5  VARCHAR(35) NOT NULL);
CREATE TABLE IF NOT EXISTS DEVICES (
       DEV_ID    BIGINT      PRIMARY KEY NOT NULL,
       DEV_NAME  VARCHAR(64) NOT NULL,
       DEV_TYPE  VARCHAR(32) NOT NULL,
       POSITION  VARCHAR(32) DEFAULT '穿戴设备',
       USERNAME  VARCHAR(64) NOT NULL,
       FOREIGN KEY (USERNAME) REFERENCES USERS(USERNAME));
CREATE TABLE IF NOT EXISTS DEV_LOG (
       DEV_ID    BIGINT      NOT NULL,
       LOG_TIME  TIMESTAMP   DEFAULT (DATETIME('now','localtime')),
       USERNAME  VARCHAR(64) NOT NULL,
       LOG_MSG   TEXT,
       PRIMARY KEY (DEV_ID, LOG_TIME),
       FOREIGN KEY (DEV_ID) REFERENCES DEVICES(DEV_ID),
       FOREIGN KEY (USERNAME) REFERENCES USERS(USERNAME));
```

在 manager.db 包下实现了一个实用类 Database，专用于提供 Java 程序与数据库特定数据的增删改查的访问接口。简单起见，各 SQL 语句以字符串常量的形式储存为私有的类属性。在类中提供静态方法以实现数据库的更删改查。

同时，为减少不必要的数据库连接代码重复书写，在类中设计了一个简单的回调接口 SQLQueryMaker，以及方法 doSqlQuery(SQLQueryMaker maker)。后者提供了统一的获取数据库连接的方法，使用时重写接口 SQLQueryMaker 的 make(Connection conn)方法，通过将数据库连接对象作为参数 conn 来实现具体的业务逻辑。

3. 用户与管理类实现

考虑到用户密码的安全性，在用户类以及对应数据库中不保存用户的密码，而是将密码进行加密，转换为 128 位的 MD5 值，并转换为 32 位的十六进制序列，保存至数据库。在 User 类中提供了该静态方法。

用户实例中，保存了一个设备表实例，使用 HashMap 实现，使用设备编号作为键，用于实现用户持有设备的关系。管理类中，提供了用户登录、登出、注册，添加、删除、获取设备，插入、检索日志，切换场景，清除缓存等方法，以建立起图形用户界面、数据库、用户实例、设备实例之间的联系。

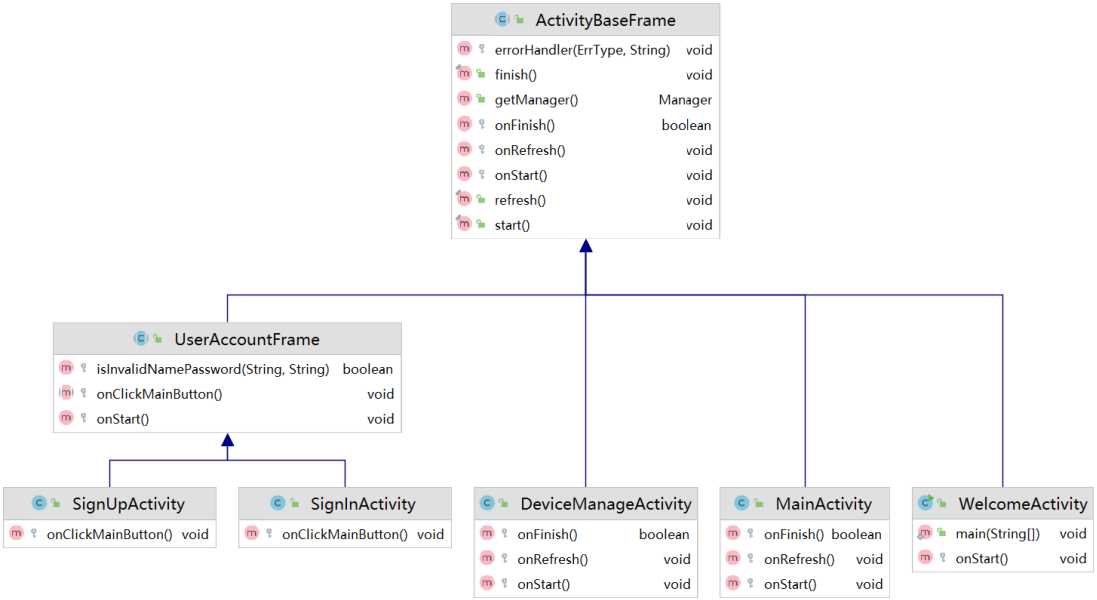
此外，为了减少对数据库的查询频率，在 User 类中添加了用户缓存（cache），并为加速查找，采用散列表来实现。当管理类调用注册方法时，首先向数据库查询密码 MD5 值是否一致，并到缓存中检索是否存在这一用户的缓存；若缓存中不存在，则检索数据库，建立用户实例，并初始化用户持有的设备。

4. 图形用户界面实现

在 GUI 中，将各个显示页面抽象为 Activity，并设计一个抽象基类 ActivityBaseFrame，提供一致的显示逻辑和共享的常量定义，其余各显示窗口都需要继承这个类。每个窗口都持有一个统一管理类对象，该对象由主方法新建，并生存于各个显示页面（Activity）中。

ActivityBaseFrame 的构造函数接受三个参数，分别为窗口标题，窗口大小（类的静态常量提供了两种固定的窗口大小可供选择），以及统一管理类实例（用于初始化类中对象）。当打开一个窗口时需要手动调用 start 方法使窗口显示，此时会调用类中的 onStart 方法和 refresh 方法，其中 onStart 方法为预留给子类覆盖的方法，用于实现窗口的显示逻辑。当一个窗口被关闭时会自动触发 finish 方法，此时会调用 onFinish 方法，并根据方法返回值确定是否关闭窗口。当一个窗口需要刷新显示区域时需调用 refresh 方法，此时会调用 onRefresh 方法。start、finish、refresh 方法设计为 public final 方法，子类不可覆盖，protected 的 onStart、onFinish、onRefresh 方法只是简单地留空，便于子类覆盖。

ActivityBaseFrame 类被设计为抽象的，但不存在抽象方法，子类可以有选择性地实现 onStart、onFinish、onRefresh 方法以完成显示逻辑。



由于用户登录和注册窗口有相同的观感，故设计了一个类 UserAccountFrame，提供抽象方法 onClickMainButton，供子类覆盖实现点击“登录”/“注册”按钮的动作。

整个程序从 WelcomeActivity 窗口启动，在窗口中提供登录和注册按钮，以登录到系统。系统的主窗口为 MainActivity，中间显示设备列表，提供按钮以新增设备和切换场景。对各设备详细的管理窗口为 DeviceManageActivity，根据传入的设备类型显示不同的信息和管理按钮。

四、实验结果与测试

1. 项目结构与配置

（1）本项目的目录结构如下：

.idea 目录：JetBrains Intellij IDEA 集成开发环境工作目录

data 目录：存放程序运行所需数据的目录

data/img 目录：程序运行所需图片

data/data.db 文件：程序内嵌数据库文件，已包含下文中的测试用例的数据

doc 目录：项目文档（本设计报告）所在目录

lib 目录：SQLite JDBC 驱动目录，需要添加到 JRE 的 classpath 中

out/production 目录：程序编译生成的 class 文件所在目录

src 目录：源代码所在目录

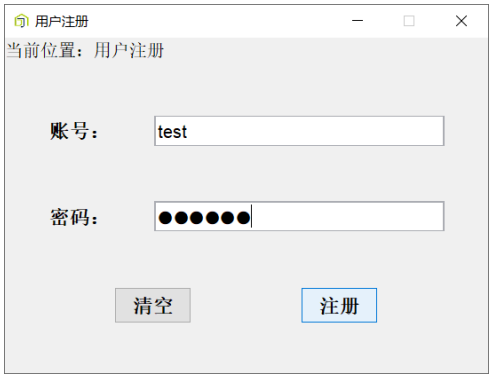
（2）测试环境：Windows 10 x64，Java SE 11，JetBrains Intellij IDEA 集成开发环境

（3）主类：manager.ui.WelcomeActivity

（4）测试流程

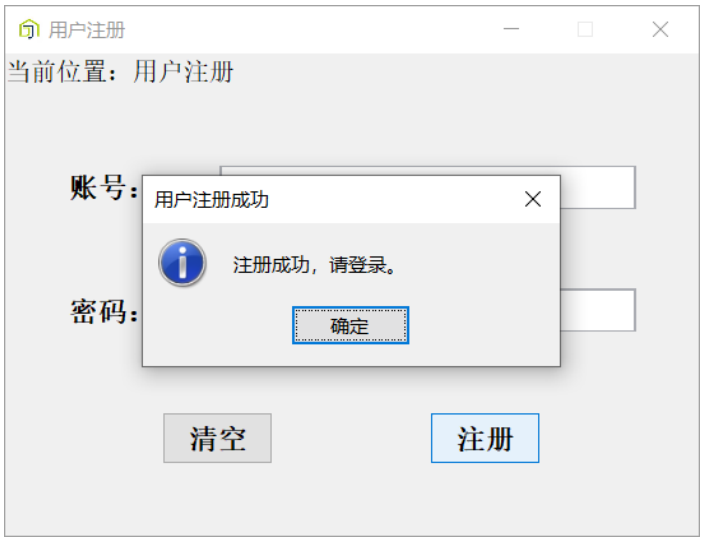
运行程序，如图：

首先点击下方“注册”按钮，弹出注册窗口，输入用户名 test 和密码 123456，点击注册：

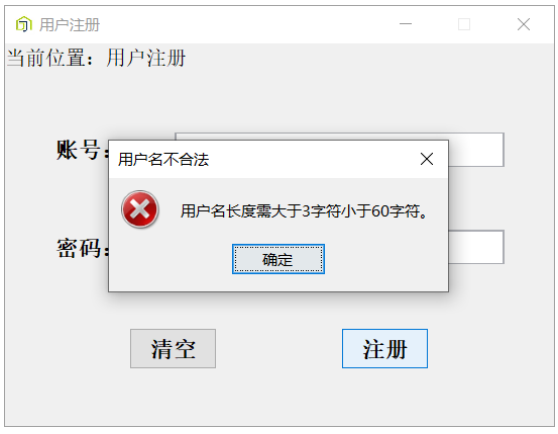


【可选】点击“清空”按钮可以清空两个输入框内的内容。（登录窗口逻辑相同）

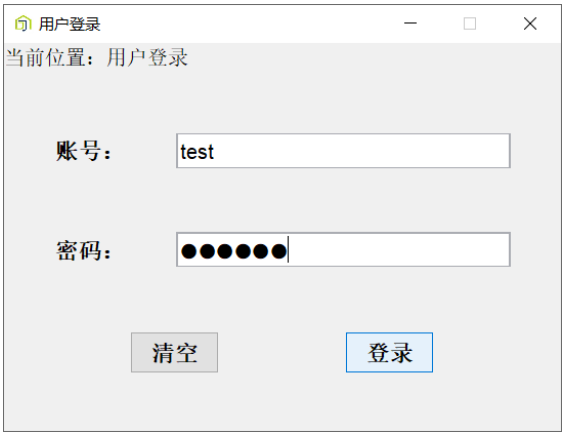
若注册成功会弹窗提示：



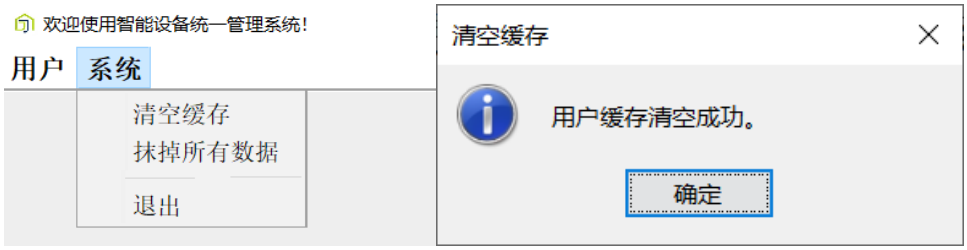
若用户名小于 3 位或大于 20 位，或（且）密码小于 6 位或大于 20 位，则会提示错误：（以用户名过短为例，登录窗口逻辑相同）



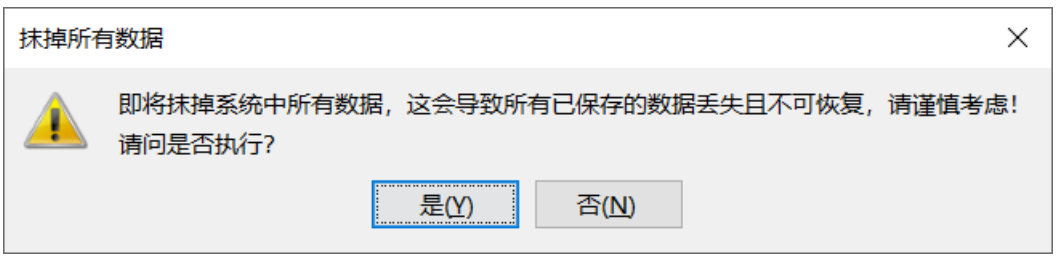
注册成功后点击“登录”按钮，如上逻辑登录系统：



【可选】用户在欢迎窗口可以点击菜单栏“系统”—“清空缓存”以清除系统中的用户缓存数据；

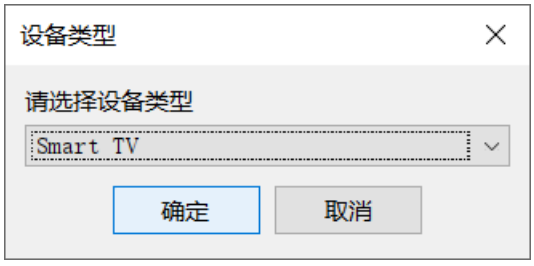


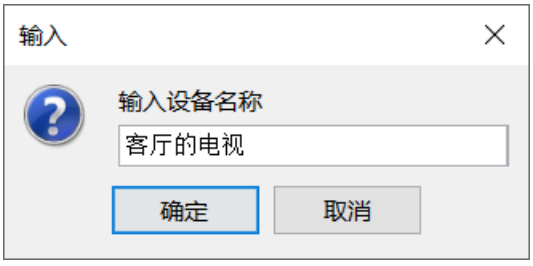
【可选】点击菜单栏“系统”—“抹掉所有数据”以完全清空数控中数据，会弹出用户确认。

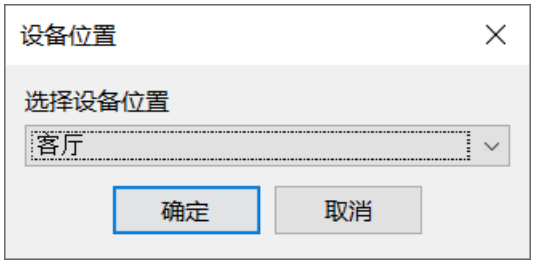


进入系统后，会打开主窗口：

以添加一个智能电视为例。点击下方“添加设备”按钮，或点击菜单栏“设备”—“添加按钮”，打开设备添加向导窗口。 选择智能电视“Smart TV”，输入设备名称，选择设备位置（所属场景），添加成功后会给出提示信息：







关闭后会在“控制中心”看到新增的设备。

以同样的方法添加多种设备，最终效果如图：

点击“选择场景”按钮，或点击菜单栏“设备”—“选择场景”，在弹出的对话框中选择场景，或“设备总览”，或“穿戴设备”，主界面将显示相应的场景的设备：

（以客厅为例）

点击相应设备的“管理”按钮，打开对应设备的管理窗口：

智能手环

智能电视

智能手表

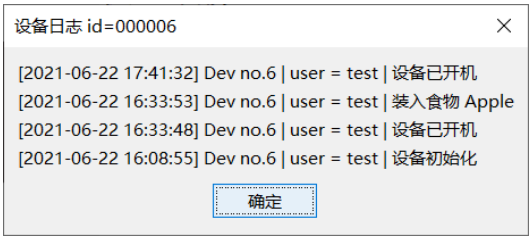
跑步机

台灯

冰箱

吸顶灯

其中，点击“设备日志”可以查看近 30 条本设备有关日志，点击菜单栏“用户”—“日志”可以查看该用户所拥有的所有设备的日志（这里以设备日志为例）：



在管理各智能设备中，如果出现了不合法的操作，会弹出错误提示，例如冰箱设置了超过范围的温度：