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 6 月 17 日	

实 验 名 称 个人自律打卡系统

一、需求分析

自律在一个人的生活中扮演着极其重要的作用，适用于多人的自律打卡系统可以使人们养成良好的习惯。本系统从个人自律打卡、记账等需求出发，基于 Java 平台设计了一套可应用于多用户的自律打卡系统。

（一）功能需求

个人自律打卡系统应具有如下功能。

1. 实现通用化、可分享的打卡活动：

面对繁多的打卡活动，需要建立起一个有组织的体系来对打卡行为进行统一的分类和管理。基于面向对象思想的继承派生的关系，将现实生活中的打卡抽象为活动，并使用对象来在计算机中表示。

2. 实现多用户、多活动的权限管理：

个人自律打卡系统不能仅仅服务于一个人，而是要面对庞大的有这一需求的客户群体。实现多用户管理与不同用户组分级管理显得尤为重要。

3. 提供友好的图形用户界面：

作为面向大众客户的打卡平台，简单易用、清晰直观的图形用户界面是重要的一环。友好的图形用户界面为用户提供了从打卡、记账到统一管理的交互逻辑。

（二）性能需求

个人自律打卡系统的性能具体要求如下。

1. 系统健壮性：个人自律打卡系统能够保持长期稳定运行，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：个人自律打卡系统应具备足够的扩展空间，能够灵活满足用户不同情境下打卡的需求，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，以不同的身份登录该系统，活动所有权归属不同用户，并可分享。

4. 系统易用性：个人自律打卡系统要求操作简单、逻辑合理、简约一致。

二、详细设计方案

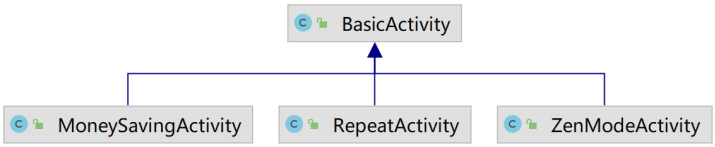
1. 用户抽象

使用者通过用户名和密码登录进入到自律打卡系统。从这一需求出发，系统要提供给用户登录和注册的入口，并且根据不同角色划分用户。因此，用户实体需要有如下的属性：用户编号、用户名、用户角色、密码（考虑到安全性，在实际存储中不存储或加密存储）。其中，用户角色设计为两个，分别为普通用户（NORMAL）和管理员（ADMIN）。两者的主要区别在于普通用户创建的活动只能自己进行打卡和管理，而管理员创建的活动除自己本身可以完全控制外，还可以授权给其他人打卡或查看。有关用户活动授权权限的详细说明，请参

阅下面关于用户打卡活动的描述。

2. 用户打卡活动抽象

在本系统中，将用户需要打卡这一动作的受事者抽象为活动（Activity）。并活动按照用户需求分为四个种类，分别是：基础活动（BasicActivity，简记为 BASIC）、可重复活动（RepeatActivity，简记为 REPEAT）、消费记录（攒钱）活动（MoneySavingActivity，简记为 MONEY）以及禅定模式活动（ZenModeActivity，简记为 Zen，用户启动该活动后在设定的时间内不能关闭）。后三者继承自基础活动，用户也可以基于基础活动扩展更丰富的活动。它们的关系如图所示：

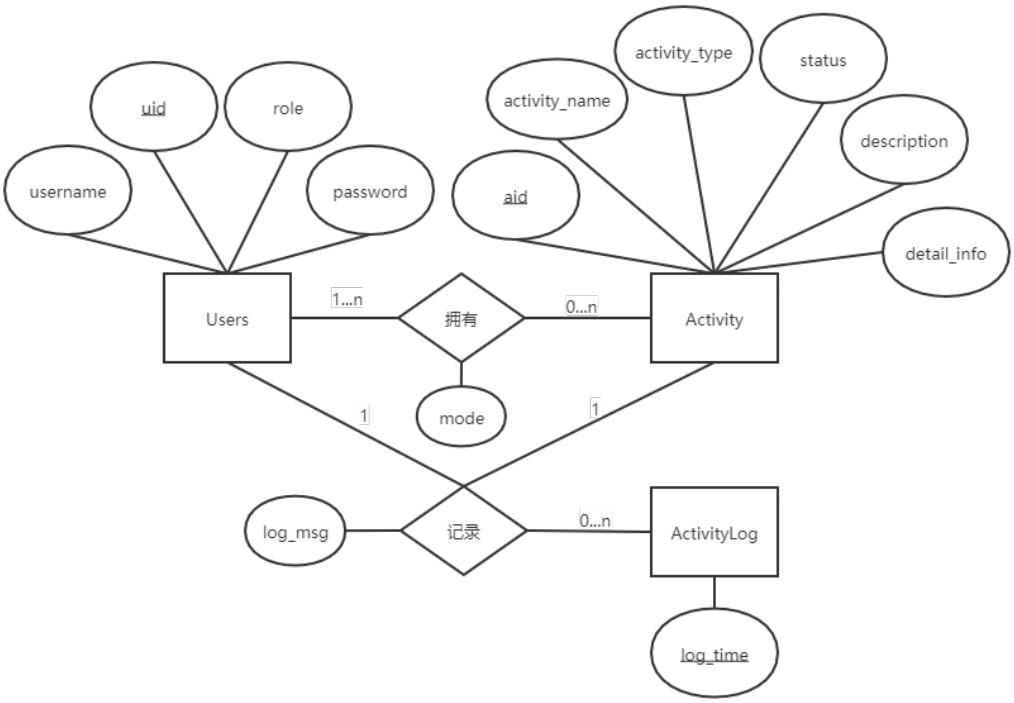


每种活动有三种状态：准备就绪（READY）、执行中（EXEC）、已经结束（FINISH）。活动新建后默认的状态为准备就绪，用户通过执行打卡动作，以及修改活动的属性，状态随之改变。FINISH 状态下不能执行打卡动作。

用户与活动的对应关系分为三种：所有者（OWN）、可查看（VISIBLE）、可打卡（EDITABLE）。所有者包含对活动的查看、打卡、编辑详情、删除的权限，可查看者只能查看活动详情以及日志，可打卡者除可查看者的权限外，还可以对活动进行打卡。共享的活动使用同一批活动数据，不进行活动的拷贝。每位用户创建的活动默认设置自身为活动的所有者，角色为管理员的用户还可以授权其他用户某一活动的权限，可以令他人查看或执行打卡。

3. 数据库设计

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、活动数据以及活动和用户之间的联系。根据上述逻辑关系设计数据库 E-R 模型，如下所示：

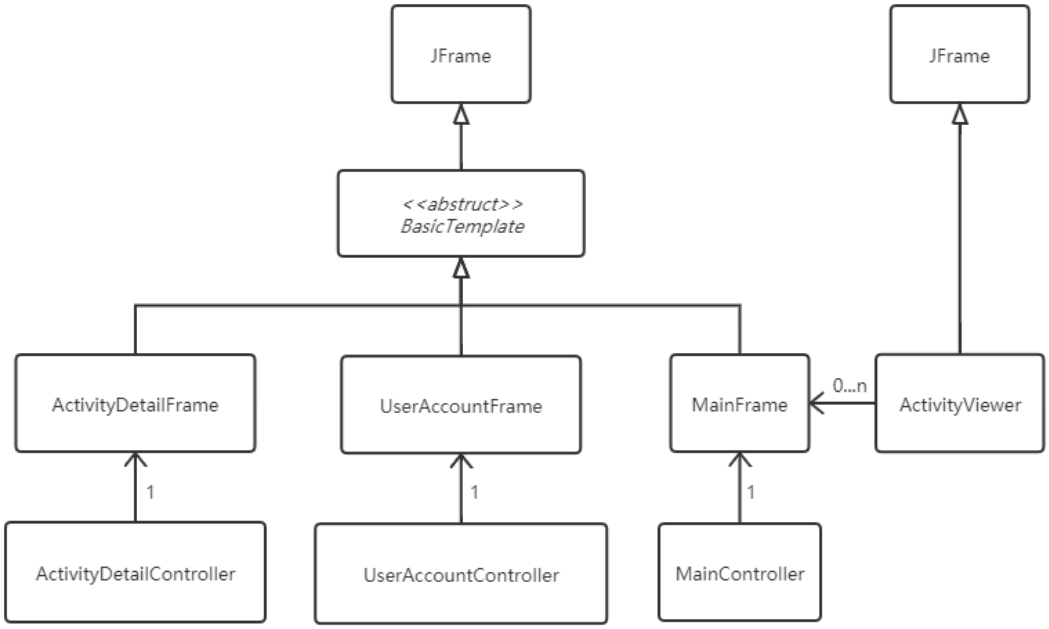


其中，Users 实体对应用户类，Activity 实体对应上述活动类及其子类。ActivityLog 实体用于表示用户打卡等动作产生的日志，通过实时的更删改查操作来实时维护与调用。

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。采用 MVC 设计模式，M——模型（Model）为之前所述的用户、活动实体；V——视图（View）为窗体类；C——控制器（Controller）为每个窗口对应的负责来联系窗体和模型的实例，依赖于各窗体类存在。

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，实现各窗口的显示逻辑。图形用户界面中各类的关系如下：



三、具体实现

1. 用户类的实现

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。

用户类对象的生存周期为从用户登录后直至程序结束，由主窗口的控制器（MainController）管理。

2. 活动类的实现

各打卡活动的实现细节如下表所示：

类名	**BasicActivity**	所属包	punch.ui.activity
父类	Object	访问控制	public
功能函数	public void punch() throws IllegalStateException; 其余为属性的 Getter/Setter		
说明	基础活动类，定义了执行打卡的方法，以及用户活动拥有模式常量、活动状态常量。 打卡前为 READY 状态，打卡后为 FINISH 状态。		

类名	**RepeatActivity**	所属包	punch.ui.activity
父类	BasicActivity	访问控制	public
功能函数	public void punch() throws IllegalStateException; 以及打卡次数属性的 Getter/Setter		
说明	可重复打卡活动类，添加打卡次数属性（默认为 1 次），重写了执行打卡的方法。打卡前为 READY 状态，打卡后剩余次数不为 0 为 EXEC 状态，剩余次数归零后状态为 FINISH，修改次数不为零后状态变为 EXEC。		
类名	**MoneySavingActivity**	所属包	punch.ui.activity
父类	BasicActivity	访问控制	public

功能函数	public void punch() throws IllegalStateException;
	public void punch(int money) throws IllegalStateException;
	以及攒钱目标和已攒金额属性的 Getter/Setter		

说明	消费记录（攒钱）活动类，添加攒钱目标和已攒金额属性，重写了执行打卡的方法，并新增一种打卡方法：记录金额。打卡前为 READY 状态，打卡后目标和已攒之差不为 0 为 EXEC 状态，为 0 后状态为 FINISH，修改目标后若差不为 0 状态变为 EXEC。		
类名	**ZenModeActivity**	所属包	punch.ui.activity
父类	BasicActivity	访问控制	public
功能函数	public void punch() throws IllegalStateException; 以及持续时间属性的 Getter/Setter		
说明	禅定模式打卡活动类。禅定模式是指用户启动该活动后在设定的时间内不能关闭的一种模式。添加了持续时间属性，并重写了执行打卡的方法。打卡前为 READY 状态，首次打卡记录时间，状态转变为 EXEC，再次打卡判断两次打卡时间是否达到设定的持续时间，若达到则结束打卡，状态转变为 FINISH，否则拒绝打卡。可以通过为处于 FINISH 状态的活动重新设定持续时间使之转变为 READY 状态。		

另外还实现了一个工厂类 ActivityFactory，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。

3. 数据库的关系定义与数据增删改查

考虑到程序规模不算大，采用嵌入式数据库 SQLite，数据库文件保存至 data/database 目录下 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
(
    UID       BIGINT PRIMARY KEY NOT NULL,
    USERNAME  VARCHAR(64)        NOT NULL UNIQUE,
    PASSMD5   VARCHAR(35)        NOT NULL,
    ROLE      VARCHAR(10)                DEFAULT 'NORMAL'
        CHECK ( ROLE IN ('NORMAL', 'ADMIN') )
);
CREATE TABLE IF NOT EXISTS ACTIVITY
(
    AID         BIGINT PRIMARY KEY NOT NULL,
```

```
    ACTIVITY_NAME VARCHAR(64)       NOT NULL,
    ACTIVITY_TYPE VARCHAR(10)       NOT NULL
        CHECK (ACTIVITY_TYPE IN ('BASIC','REPEAT','ZEN','MONEY')),
    STATUS        VARCHAR(10)       NOT NULL DEFAULT 'READY'
        CHECK ( STATUS IN ('READY', 'EXEC', 'FINISH') ),
    DESCRIPTION   TEXT              NOT NULL,
    DETAIL_INFO   TEXT                       DEFAULT ''
);
CREATE TABLE IF NOT EXISTS ACTIVITY_OWN
(
    UID    BIGINT     NOT NULL,
    AID    BIGINT     NOT NULL,
    A_MODE VARCHAR(10) NOT NULL DEFAULT 'OWN'
        CHECK ( A_MODE IN ('OWN', 'VISIBLE', 'EDITABLE') ),
    PRIMARY KEY (UID, AID),
    FOREIGN KEY (AID) REFERENCES ACTIVITY (AID),
    FOREIGN KEY (UID) REFERENCES USERS (UID)
);
CREATE TABLE IF NOT EXISTS ACTIVITY_LOG
(
    AID     BIGINT NOT NULL,
    UID     BIGINT NOT NULL,
    LOG_TIME TIMESTAMP DEFAULT (DATETIME('now', 'localtime')),
    LOG_MSG  TEXT   NOT NULL,
    PRIMARY KEY (AID, UID, LOG_TIME),
    FOREIGN KEY (UID) REFERENCES USERS (UID),
    FOREIGN KEY (AID) REFERENCES ACTIVITY (AID)
);
```

在 punch.db 包下实现了一个单例类 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。在类中提供实例方法以实现数据库的更删改查。

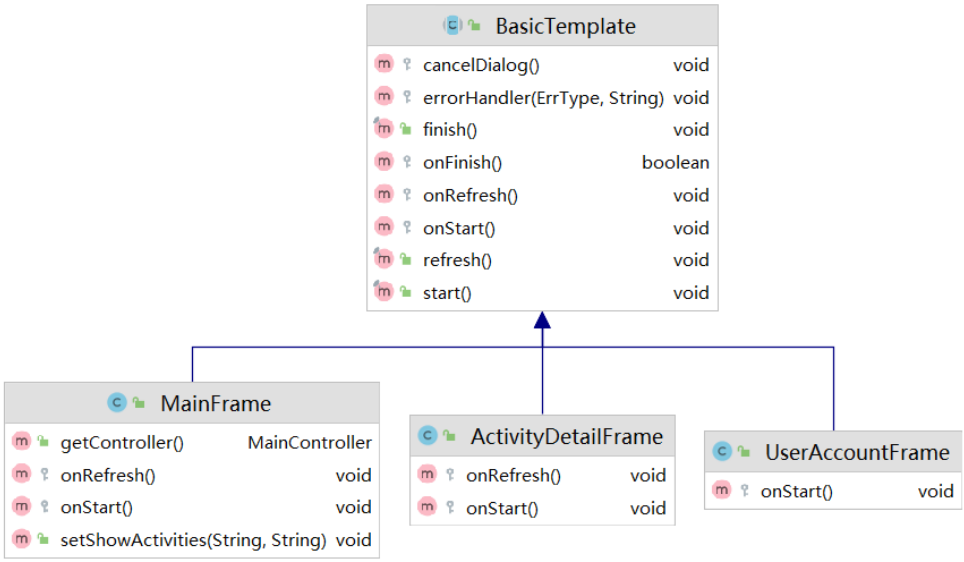
单例类的实现上采用内部静态类持有单一实例，Database 类提供 getInstance 方法获取该实例，并拒绝 Database 类被实例化。此外，为减少对数据库的频繁查询，在 Database 类中使用 HashMap 实现并了用户表和活动表的缓存。

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。

4. 图形用户界面的实现

在图形用户界面中采用 MVC 模式，各个显示窗口定义在 punch.ui.frame 包内，并设计一个抽象基类 BasicTemplate，为 GUI 的通用模板，提供一致的显示逻辑和共享的常量定义，其余显示窗口都需要继承这个类。每个窗口类持有相应的控制器类对象，由创建窗口时构造并作为参数传入，生存于各个显示窗口中。控制器对象负责联系窗体、模型与数据库，定义

在 punch.ui.controller 包内。关于 punch.ui.frame 包中各类的继承关系与非私有方法如图所示：



其中，BasicTemplate 的构造函数接受两个参数，分别为窗口标题，窗口大小（该类的静态常量提供了多种固定的窗口大小可供选择）。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。

BasicTemplate 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。

启动类（主类）PunchInStartup 首先启动 UserAccountFrame 窗口并提供相应控制器对象，以打开用户登录/注册窗口，以供用户登录到系统。系统的主窗口为 MainActivity，顶部为四个按钮，分别是“新增活动”、“筛选活动”、“可见范围”和“用户日志”。主窗口中间显示活动列表，采用 GridLayout 的 2 列布局。

主窗口中每个活动的显示是通过自定义组件 ActivityViewer 来实现的，该组件定义在包 punch.ui.component 中，使用 BoxLayout 提供了显示活动名和基本信息的功能。

对各设备详细的管理窗口为 ActivityDetailFrame，根据传入的设备类型显示不同的按钮。

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/database 目录：内嵌数据库文件目录，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

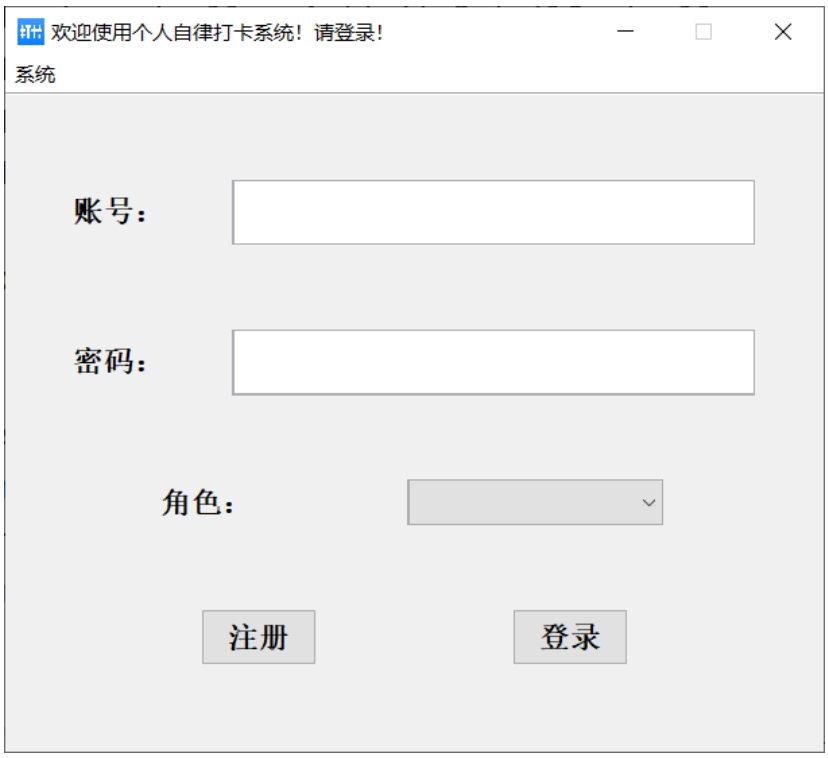
src 目录：源代码所在目录

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

（3）主类：punch.PunchInStartup

（4）测试流程

运行程序，如图：

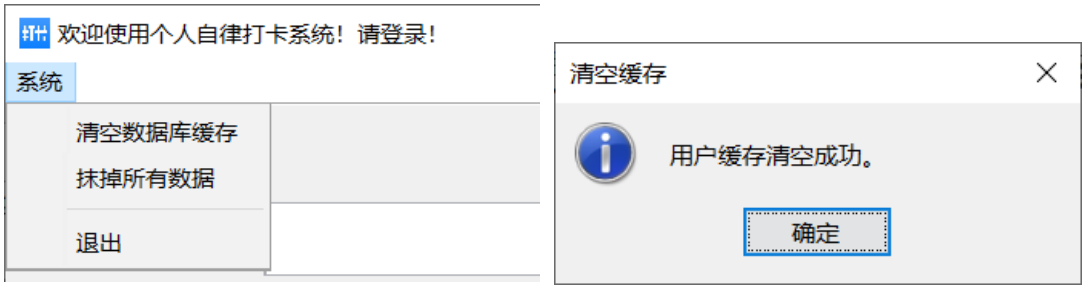


输入用户名 test1 和密码 123456，选择角色为 NORMAL，点击“注册”按钮以注册平台用户（如已经注册则点击“登录”按钮）：

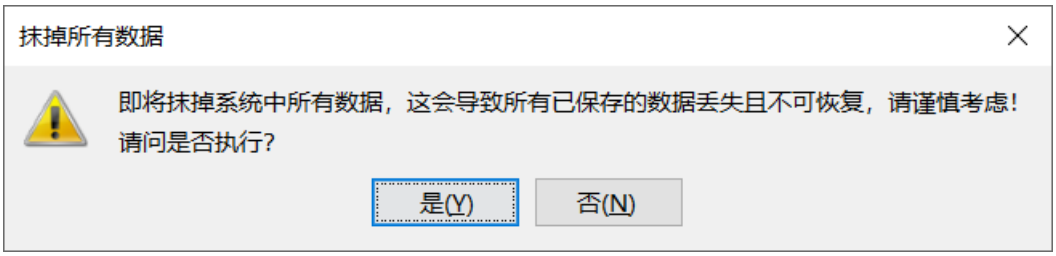
若注册成功会弹窗提示：

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）

【可选】用户可以点击菜单栏“系统”—“清空缓存”以清除系统中的用户缓存数据；

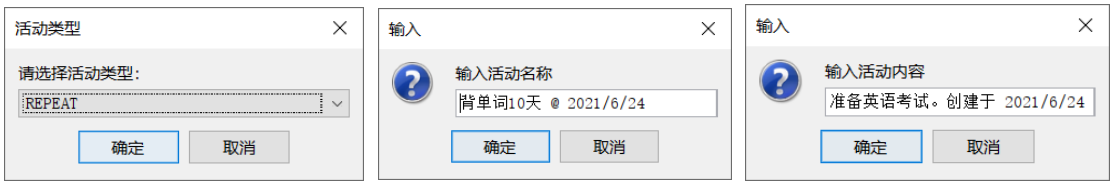


【**可选**】点击菜单栏“系统”—“抹掉所有数据”以完全清空数控中数据，会弹出用户确认。



进入主窗口如图：

以添加一个可重复活动为例。点击上方“新增活动”按钮，打开活动新增向导窗口。 选择活动类型“REPEAT”，输入活动名称和活动具体内容，添加成功后可以在主窗口看到该活动。



用同样的方法添加各类活动，最终显示效果如下：

点击每个活动下的“更多”按钮可以进入活动详情，可打卡或做详细配置：

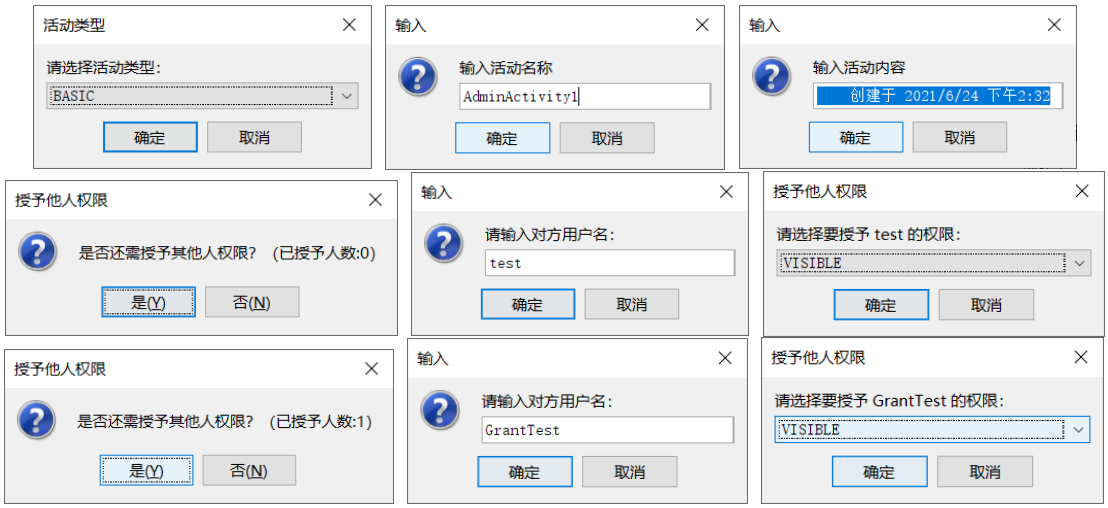
点击“用户日志”可以查看当前用户近 30 条活动记录，点击“活动日志”可以查看当前活动近 30 条活动记录（这里以“活动日志”为例）：

点击主窗口的“筛选活动”按钮，选择某一类别即可看到相应类别下的活动（以基础活动“BASIC”为例）：

下面测试管理员活动授权功能。

首先使用第一步的方法新建一个管理员账号（用户名 AdminTest，密码 654321，角色“ADMIN”）和一个用户账号（用户名 GrantTest，密码 112233，角色“NORMAL”）。先以

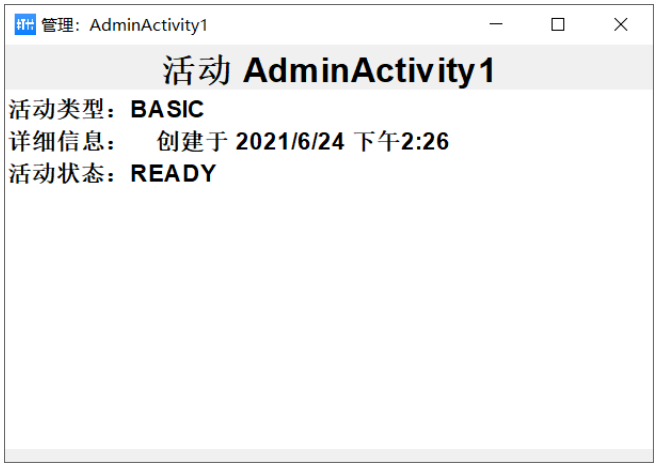
管理员账号登录，创建基础活动 AdminActivity1，授予普通用户 test 和 GrantTest 可查看（VISIBLE）权限：



以同样的方法创建可重复打卡活动 AdminActivity2，授予普通用户 test 和 GrantTest 可打卡（EDITABLE）权限。（图略）AdminTest 的主窗口如图所示：

登录到 GrantTest 用户，点击“可见范围”，选择“VISIBLE”，可见 AdminActivity1：

点击“更多”，可以看到详细信息，但不能打卡/修改：



登录到 test 用户，首页可见 AdminActivity1 和 AdminActivity2：

点击“可见范围”，选择“EDITABLE”，可见 AdminActivity2，点击“更多”进入详情页后可以打卡和查看日志：

“可见范围”和“筛选活动”可以联合使用，以对活动列表进行过滤。

“可见范围”可选有“ALL”（全部）、“OWN”（自己创建的）、“VISIBLE”（可查看）、“EDITABLE”（可打卡）；“筛选活动”可选为全部活动以及四种活动类别。

以 test 用户，可见范围“OWN”（自己创建的），筛选活动“ZEN”（禅定模式活动）为例，主窗口显示如下：