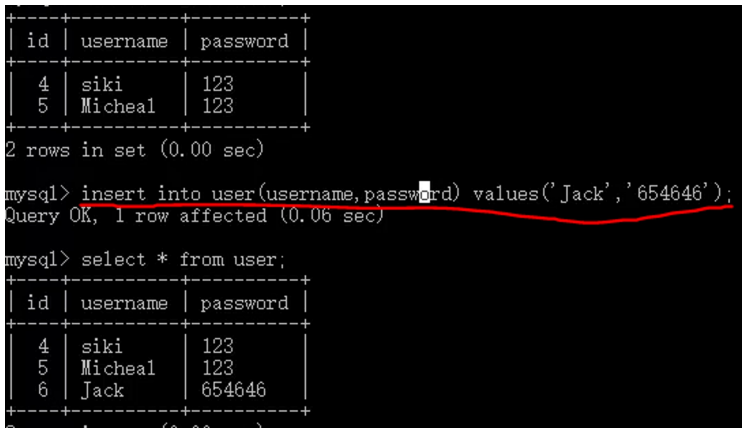
3.24

2020年3月24日　0:02

- ctrl+shift+f12：放大缩小
- shift+enter：另起一行
- ctrl+x，c，v：对一整行进行操作，ctrl+z：回退一行

- 数据库
  - 增加（返回int）：insert into 数据库其中的一张表名（字段，字段） values(值，值);
  - 删除（返回int）：delete from 数据库其中的一张表名 where 变量名 = ' ';
  - 查询（返回ResultSet）：select * from 数据库其中的一张表名　　//* 表示表中的所有字段
    select * from 数据库其中的一张表名 where 字段=xxx
  - 修改（返回int）：update 数据库其中的一张表名 set 变量名=' (要改成的内容)', 变量名=' (要改成的内容)' where 另一个没有要修改的变量名= ‘ ’ ;
  - 查看当前选择的数据库：select database();
  - 选择要进行操作的数据库：use 数据库名;
  - 显示所有的数据库：show database();
  - 只针对某些列做插入
    - 

3.25

2020年3月24日　0:02

- Git
  - 创建版本库
    - 初始化git仓库：'git init'
    - 显示git目录：'ls -ah'
    - 配置个人信息（在git中设置当前使用的用户）：名字：'git config --global user.name "xxx" '　邮箱：'git config --global user.email "1371539005@qq.com" '　每次备份都会把备份者信息存储
  - 把文件添加到版本库：（2步：commit一次可以add多个文件）
    - 把文件放在门口：打开文件所在目录，执行命令：'git add ./xxx.扩展名'(./表示当前目录，按tap可以自动补完文件名）‘git add.'添加文件夹下的所有文件
    - 把文件放进仓库：执行命令：'git commit -m "对本次提交的说明" '
  - 版本回退
    - 查看从最近到最远的提交日志：'git log'
    - 只查看日志提交说明：'git log --pretty=oneline'
    - 把文档回退到上一个版本：'git reset --hard HEAD^'
      - HEAD指向的版本就是当前的版本　HEAD^：上一个版本　HEAD^^：上上一个版本　HEAD~100：上100个版本
    - 把文档回到未来的某个版本：'git reset --hard commit_id'
      - 所有文件版本的Commit_id 的查找(查看提交历史)：'git reflog'
  - 修改
    - 查看仓库当前状态：'git status'
      查看文件修改的具体内容：'git diff xxx.扩展名'
    - 查看工作区和版本库里面最新版本的区别：'git diff HEAD -- xxx.扩展名'
    - 撤销文件在工作区的全部修改'git checkout -- xxx.扩展名'，这里有两种情况：
      - 一种是文件自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一摸一样的状态
      - 一种是文件已经添加到暂存区后，又做了修改，现在，撤销操作就回到添加暂存区后的状态
    - 撤销文件在暂存区的修改，重新放回工作区（该修改会放到工作区中去）：
      'git reset HEAD xxx.扩展名'
  - 删除文件
    - 从版本库中删除该文件：使用命令'git rm xxx.扩展名'，并且'git commit -m "remove xxx.扩展名" '
    - 删错了从版本库中恢复到最新版本：'git checkout -- xxx.扩展名'（该命令是用版本库中的版本替换工作区的版本，无论工作区是修改还是删除，都可以一键还原）

- 添加远程库：https://www.liaoxuefeng.com/wiki/896043488029600/898732864121440
  - git push origin master
- 分支：https://www.liaoxuefeng.com/wiki/896043488029600/900003767775424

3.26

2020年3月26日 8:26

- JDBC编程
  - 一：注册驱动（告诉Java程序即将要连接的数据库）
    - 方式一
      - Driver driver = new com.mysql.jdbc.Driver();
        DriverManager.registerDriver(driver);
      - 要导包
        - import java.sql.Driver;
          import java.sql.DriverManager;
    - 方式二：类加载方式注册驱动（常用）
      - try{Class.forName（"com.mysql.jdbc.Driver"）
        }catch（ClassNotFoundException e）{
        e.printStackTrace（）；
        }
  - 二：获取链接（表示JVM的进程和数据库之间的通道打开了，使用完后一定要关闭）
    - /*
      url:统一资源定位符
      url包括以下几个部分：
      协议 IP PORT（服务器上软件的端口） 资源名
      jdbc:mysql://localhost:3306/text
      jdbc:mysql:// 协议
      localhost IP地址
      3306 MySQL数据库端口号
      text 具体数据库实例名
      说明：localhost和127.0.0.1都是本机地址
      */
      String url = "jdbc:mysql://localhost:3306/text";
      String user = "root";
      String password = "myasdw902";
      Connection con = DriverManager.getConnection(url,user,password);
    - 要导包
      - import java.sql.Connection;
  - 三：获取数据库操作对象（专门执行sql语句的对象）
    - Statement stmt = con.createStatement();//创建一个Statement对象来将SQL语句发送到数据库
  - 四：执行SQL语句（DQL,DML...）
    - String sql = "insert into text_detail(text_name,text_password) value('henhao','945')";
    - //sql语句，不需要分号
      //executeUpdate(String a)专门执行DML语句，返回int（影响数据库中的记录条数）
      int count = stmt.executeUpdate(sql);//此处的sql语句为 增 删 改 语句

```
System.out.println(count == 1 ? "保存成功":"保存失败");
```

- 五：处理查询结果集（只有第四步执行的是select语句才有第五步）
- 六：释放资源（使用完后一定要关闭资源）
  - //为保证资源一定释放。在finally语句块中关闭资源
    //并且要遵循先打开后关闭，要分别对其try...catch（SQLException e）{e.printStackTrace()}...，避免第一个close();出错后面的close没有被执行

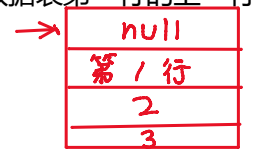
3.27

2020年3月27日 19:39

- JDBC：实际开发中不建议把数据库的信息写死在程序中
  - 获取连接时使用资源绑定器绑定属性配置文件
    - Java程序
      - 要导包： import java.util.*;//导入该包的全部类
      - /*仅仅需要把myresource.properties文件放在src下，如果是放在package下，则程序的filename应该是 “package名/文件名(不要后缀)“
        ResourceBundle bundle = ResourceBundle.getBundle("resources/db");
        String driver = bundle.getString("driver");
        String url = bundle.getString("url");
        String user = bundle.getString("user");
        String password = bundle.getString("password");
    - 配置文件： (xxx.properties)

      ```
      //没有分号
      driver = com.mysql.jdbc.Driver
      url = jdbc:mysql://localhost:3306/text
      user = root
      password = myasdw902
      ```

  - 处理查询结果集
    - resultSet rs = stmt.executeQuery(sql)；//关闭时先关闭它，executeQuery(sql)专门执行查询语句
    - rs.next()：使光标在结果集向下移一行，如果那一行有数据则返回true ， 无效返回false
      //光标最开始指向数据表第一行的上一行，该行没有数据

    - rs.getString(1/“查询结果集的字段名”)：不管数据库中的数据类型是什么。都以String的形式取出。
      1代表要取得数据得列数，JDBC中的下标从1开始，不是从0开始。
      建议写程序时用“查询结果集的字段名” ，可以增强程序的健壮性
      selest重命名：String sql = "select text_name as t,text_password from text_detail";后面应该为re.getString("t") 而不再是 re.getString("text_name")
    - rs.getInt(“查询结果集的字段名”)：以int的形式取出
      rs.getDouble(“查询结果集的字段名”)：double的形式取出

3.28

2020年3月28日 15:18

- 导入jar包：
  - 新建一个empty project -> 在project里面建立jdbc模块 -> 右键模块打开Open Module Setting -> Libraries ->"+"添加Java库

- 把变量拼到一个字符串里面
  - String sql = "select * from text_detail where text_name = '"+loginName+"' and text_password = '"+loginPassword+"' ";
    // ‘"+变量名+"’
- 事务：保证多条SQL语句一起成功或失败
  - con.setAutoCommit(false);//开启事务
    con.commit（）；//提交事务，中间的SQL语句会一起提交（事务结束）
    con.roollback（）；//上面执行过程若出现异常，回滚事务（事务结束），保证数据的安全性，要放在try catch（SQLException）里面

3.29

2020年3月29日 9:42

- 行级锁（悲观锁）： for update
  - Select * from 表格名 where 字段='xxx' for update

    锁住以后别的线程或别的线程开启的事务都无法对锁定的哪一行数据进行修改
- 解决SQL注入

```
String sql = "select * from text_detail where text_name = ? and
text_password = ? for update";
    try {
        con = Util.getConnection();
        pstmt = con.prepareStatement(sql);
        pstmt.setString(1,userName);//给占位符？传值，下表从0开始
        pstmt.setString(2,userPassword);
        rs = pstmt.executeQuery();
    }catch...
```

- Java的修饰符类型

  public 的类、类属变量及方法，包内及包外的任何类均可以访问；

  protected 的类、类属变量及方法，包内的任何类，及包外的那些继承了此类的子类才能访问；

  private 的类、类属变量及方法，包内包外的任何类均不能访问；

  firendly 如果一个类、类属变量及方法不以这三种修饰符来修饰，它就是friendly类型的，那么包内的任何类都可以访问它，而包外的任何类都不能访问它(包括包外继承了此类的子类)，因此，这种类、类属变量及方法对包内的其他类是友好的，开放的，而对包外的其他类是关闭的。

3.31

2020年3月31日 15:34

- Father user = new Son（）；需要调用子类特有的方法时再向下造型
- 组件component
  - 窗口
  - Dialog 弹窗
  - Penal 面板：界面上分割的区域
  - 文本框
    - ITextFiled
      - get/setText()，用于获取/设置 JTextField 中的文本。
  - 列表框
  - Button 按钮 JButton

    它们控制文本、图片和方向：
    get/setText(): 获取/设置标签的文本。
    get/seticon(): 获取/设置标签的图片。
    get/setHorizontalAlignment(): 获取/设置文本的水平位置。
    get/setVerticalAlignment(): 获取/设置文本的垂直位置。
    get/setDisplayedMnemonic(): 获取/设置访问键（下划线字符），与 Alt 按钮组合时，造成按钮单击。

  - 监听事件
  - 鼠标事件
  - 键盘事件
  - Label 标签

- 快捷键
  - 进入代码：ctrl+鼠标点击

4.2

2020年4月2日 20:15

- JMenu与JMenuItem区别
  - JMenu：下面可以继续添加JMenu或JMenuItem
  - JMenuItem：下面不可以继续添加东西
- 设置JFrame最大化：
  - this.setExtendedState（Jframe.MAXIMIZED_BOTH）；
- 设置JFrame居中显示：
  - this.setLocationRelativeTo（null）；
- 登录失败弹出窗口：
  - JOptionPane.showMessageDialog(null, "用户名或密码错误！");
- 登录成功后进入主界面：
  - dispose（）；//销毁登录界面
    new MainFrm（）.setVisible（true）；//创建主界面，MainFrm是主界面的变量名
- 在主界面上创建内部窗体：JInternalFrame
  - 页面设计
    - 可最关闭：closeable为true
    - 可最小化：iconifiable为true

4.3

2020年4月3日 15:13

- 窗口中添加文本域：
  - JTextArea
- 点击打开内部窗体
  - JavaText javaText = new JavaText;
    javaText.setVisible（true）;
    table.add（javaText）;
- 实现模糊查询
  StringBuffer sb = new StringBuffer("select * from 表名");
  if(StringUtil.isNotEmpty(bookType.getBookTypeName())){
  sb.append("and bookTypeName like '%+bookType.getBookTypeName()+%'");
  }
  PreparedStatement pstmt = con.prepareStatement(sb.toString().replaceFirst("and","where");
  return pstmt.executeQuery();
- 弹出

4.7

2020年4月7日 13:24

- 正则表达式
  - \d：匹配0-9的任意一个数字（写正则表达式时要写为//d，第一个反斜杠表示这是一个反义字符）
  - 特殊符号 . 来匹配任意一个字符： “a.c”
    - “abc” ✔ “adc” ✔ “ac” ✘ “abbc” ✘
  - \w：匹配一个字母、数字、或下划线 “java\w”
    - “javac” ✔ “java8” ✔ “java_” ✔ “java#” ✘ “java ” ✘
  - \W：匹配一个非字母、数字、或下划线 “java\w”
    - “javac” ✘ “java8” ✘ “java_” ✘ “java#” ✔ “java ” ✔
  - {n}：匹配n个字符： “\d{6}”
    - “123456” ✔ “888888” ✔ “12345 “ ✘ “1234567” ✘
  - {n，m}：匹配n到m个字符： “\d{3，5}”
    - “123” ✔ “12345” ✔
  - {n，}：可以匹配至少n个字符：” \d{3,} “

4.8

2020年4月8日 19:34

- 模糊查询
  - %： 表示任意0个或多个字符。可匹配任意类型和长度的字符，有些情况下若是中文，请使用两个百分号（%%）表示。

    比如 SELECT * FROM [user] WHERE u_name LIKE '%三%'

    将会把u_name为“张三”，“张猫三”、“三脚猫”，“唐三藏”等等有“三”的记录全找出来。
  -