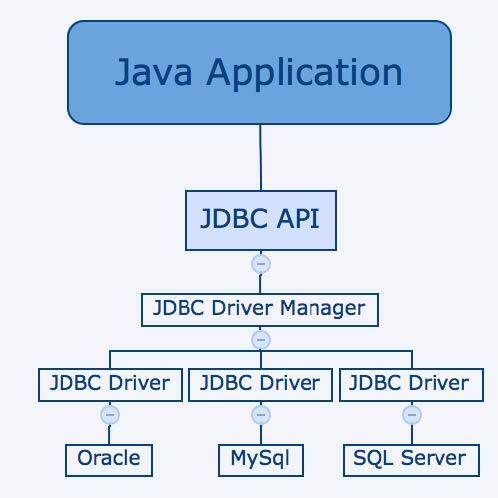
**JDBC的体系结构**

JDBC（Java DataBase Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。 Java 具有坚固、安全、易于使用、易于理解和可从网络上自动下载等特性，是编写数据库应用程序的杰出语言。所需要的只是 Java应用程序与各种不同数据库之间进行对话的方法。 JDBC可以在各种平台上使用Java，如Windows，Mac OS和各种版本的UNIX。 JDBC API支持用于数据库访问的两层和三层处理模型，但通常，JDBC体系结构由两层组成：

* JDBC API：这提供了应用程序到JDBC管理器连接。
* JDBC驱动程序API：这支持JDBC管理器到驱动程序连接。

JDBC API使用驱动程序管理器和特定于数据库的驱动程序来提供与异构数据库的透明连接，如下图所示： 

**JDBC的核心组件**

JDBC的核心组件包括：

* DriverManager： 此类管理数据库驱动程序列表。使用通信子协议将来自java应用程序的连接请求与适当的数据库驱动程序匹配。
* Driver:此接口处理与数据库服务器的通信，我们很少会直接与Driver对象进行交互。而是使用DriverManager对象来管理这种类型的对象。
* Connection：该界面具有用于联系数据库的所有方法。连接对象表示通信上下文，即，与数据库的所有通信仅通过连接对象。
* Statement：使用从此接口创建的对象将SQL语句提交到数据库。除了执行存储过程之外，一些派生接口还接受参数。
* ResultSet：在使用Statement对象执行SQL查询后，这些对象保存从数据库检索的数据。它作为一个迭代器，允许我们遍历其数据。
* SQLException：此类处理数据库应用程序中发生的任何错误

**使用步骤**

构建JDBC应用程序涉及以下六个步骤：

* 导入包：需要包含包含数据库编程所需的JDBC类的包。大多数情况下，使用import java.sql.\*就足够了。
* 注册JDBC驱动程序：要求您初始化驱动程序，以便您可以打开与数据库的通信通道。
* 打开连接：需要使用DriverManager.getConnection（）方法创建一个Connection对象，该对象表示与数据库的物理连接。
* 执行查询：需要使用类型为Statement的对象来构建和提交SQL语句到数据库。
* 从结果集中提取数据：需要使用相应的ResultSet.getXXX（）方法从结果集中检索数据。
* 释放资源：需要明确地关闭所有数据库资源，而不依赖于JVM的垃圾收集。

建立JDBC连接所涉及的编程可简单概括为以下四个步骤

* 导入JDBC包：将Java语言的*import*语句添加到Java代码中导入所需的类。
* 注册JDBC驱动程序：此步骤将使JVM将所需的驱动程序实现加载到内存中，以便它可以满足您的JDBC请求。
* 数据库URL配置：这是为了创建一个格式正确的地址，指向要连接到的数据库。
* 创建连接对象：最后，调用DriverManager对象的getConnection（）方法来建立实际的数据库连接。

**Class.forName();** 注册驱动程序最常见的方法是使用Java的Class.forName()方法，将驱动程序的类文件动态加载到内存中，并将其自动注册。

1. try {
2. Class.forName("com.mysql.cj.jdbc.Driver");
3. }catch(ClassNotFoundException ex) {
4. System.out.println("Error: unable to load driver class!");
5. System.exit(1);
6. }

**数据库URL配置** 加载驱动程序后，可以使用DriverManager.getConnection（）方法建立连接。为了方便参考，特列出三个重载的DriverManager.getConnection()方法 -

* getConnection（String url）
* getConnection（String url，Properties prop）
* getConnection（String url，String user，String password）

以下是常用的DBMS连接方式：

| **RDBMS** | **JDBC驱动程序名称** | **网址格式** |
| --- | --- | --- |
| MYSQL8 | com.mysql.cj.jdbc.Driver | jdbc:mysql://hostname:3306/databaseName?serverTimezone=UTC |
| MySQL | com.mysql.jdbc.Driver | jdbc:mysql://hostname:3306/databaseName |
| ORACLE | oracle.jdbc.driver.OracleDriver | jdbc:oracle:thin:@hostname:port Number：databaseName |
| DB2 | COM.ibm.db2.jdbc.net.DB2Driver | jdbc:db2:hostname:port Number / databaseName |
| SYBASE | com.sybase.jdbc.SybDriver | jdbc:sybase:Tds:hostname:port Number / databaseName |

注意MySQL8与先前的版本所使用的驱动程序是不同的。

**创建数据库连接对象**

1. String URL = "jdbc:mysql://localhost:3306/dbname?serverTimezone=UTC";
2. String USER = "username";
3. String PASS = "password"
4. Connection conn = DriverManager.getConnection(URL, USER, PASS);

完整的连接地址: jdbc:mysql://127.0.0.1:3306/dbname?useUnicode=true&characterEncoding=UTF8&useSSL=false&serverTimezone=UTC"

**关闭数据库连接** 为确保连接关闭，您可以在代码中提供一个“finally”块。一个finally块总是执行，不管是否发生异常。 要关闭上面打开的连接，你应该调用close（）方法如下: conn.close();

**JDBC执行SQL语句** 一旦获得了连接，我们可以与数据库进行交互。JDBC Statement和PreparedStatement接口定义了能够发送SQL命令并从数据库接收数据的方法和属性。

| **接口** | **推荐使用** |
| --- | --- |
| Statement | 用于对数据库进行通用访问。在运行时使用静态SQL语句时很有用。Statement接口不能接受参数。不过你可以通过字符串运算，将参数值拼接成一条SQL语句。 |
| PreparedStatement | 当您计划多次使用SQL语句时使用。PreparedStatement接口在运行时接受输入参数。 |

**创建Statement对象**

在使用Statement对象执行SQL语句之前，需要使用Connection对象的createStatement()方法创建Statement的一个实例，如下例所示：

1. Statement stmt = null;
2. try {
3. stmt = conn.createStatement( );
4. . . .
5. }
6. catch (SQLException e) {
7. . . .
8. }
9. finally {
10. . . .
11. }

**执行Statement对象** 创建Statement对象后，您可以使用它来执行一个SQL语句，其中有三个执行方法之一。

* boolean execute(String SQL)：如果可以检索到ResultSet对象，则返回一个布尔值true; 否则返回false。使用此方法执行SQL DDL语句或需要使用真正的动态SQL时。
* int executeUpdate(String SQL)：返回受SQL语句执行影响的行数。使用此方法执行预期会影响多个行的SQL语句，例如INSERT，UPDATE或DELETE语句。
* ResultSet executeQuery(String SQL)：返回一个ResultSet对象。当您希望获得结果集时，请使用此方法，就像使用SELECT语句一样。

**关闭Statement对象** 就像关闭一个Connection对象以保存数据库资源一样，由于同样的原因，还应该关闭Statement对象。 调用close()方法即可关闭Statement对象。如果先关闭Connection对象，它也会关闭Statement对象。但是，应始终显式关闭Statement对象，以确保正确清理。

1. Statement stmt = null;
2. try {
3. stmt = conn.createStatement( );
4. . . .
5. }
6. catch (SQLException e) {
7. . . .
8. }
9. finally {
10. stmt.close();
11. }

**PreparedStatement** PreparedStatement的接口扩展了Statement接口，其优点是可以动态地提供参数。如果语句被多次执行，其执行效率比Statement高。

1. PreparedStatement pstmt = null;
2. try {
3. String SQL = "Update Employees SET age = ? WHERE id = ?";
4. pstmt = conn.prepareStatement(SQL);
5. . . .
6. }
7. catch (SQLException e) {
8. . . .
9. }
10. finally {
11. pstmt.close();
12. }

JDBC中的所有参数都用？符号代替，这被称为参数标记(又叫占位符)。在执行SQL语句之前，必须为每个参数提供值。 用setXXX(参数序号,参数值)方法将值绑定到对应参数，其中XXX代表要绑定到输入参数的值的Java数据类型。如果忘记提供值，将收到一个SQLException。参数的序号从1开始。 记得调用close()方法，显示关闭PreparedStatement对象。

**ResultSet** ResultSet对象维护指向结果集中当前行的游标。有多种类型的“游标”，如果没有指定任何ResultSet类型，则取缺省值TYPE\_FORWARD\_ONLY。

| **类型** | **类型描述** |
| --- | --- |
| ResultSet.TYPE\_SCROLL\_INSENSITIVE | 光标可以向前和向后滚动，结果集对创建结果集后发生的数据库的其他更改不敏感。 |
| ResultSet.TYPE\_SCROLL\_SENSITIVE | 光标可以向前和向后滚动，结果集对创建结果集之后发生的其他数据库所做的更改敏感。 |
| ResultSet.TYPE\_FORWARD\_ONLY | 光标只能在结果集中向前移动。 |

**ResultSet的遍历** 用ResultSet的next()方法取得游标当前行的值，并用getXXX(列名)方法获列这一行某个列的值，XXX是与列匹配的数据类型，next()会推进游标到下一行，如果next()返回false表示数据集已取完(或者是空集)。 示例：

1. resultSet = statement.executeQuery("select \* from student;");
2. while (resultSet.next()) {
3. System.out.println(resultSet.getString("sno"));
4. System.out.println(resultSet.getString("sname"));
5. System.out.println(resultSet.getInt("sage"));
6. }

**编程要求**

正确使用JDBC，查询金融应用场景数据库finance的client表(客户表)中邮箱不为空的客户信息，列出客户姓名，邮箱和电话.一个展示结果的示例如下(字体颜色是平台自动加的，不是编程要求)：

姓名 邮箱 电话

夏雅惠 [57433144004@qq.com](mailto:57433144004@qq.com) 18962433158

钟庭玮 [59434199077@163.com](mailto:59434199077@163.com) 18110434192

注：标题以及字段值之间用制表符隔开。第1列和第2列间用一个制表符，第2列和第3列间，标题用4个制表符，字段值用两个制表符隔开。

JDBC驱动程序(mysql-connector-java-8.0.23.jar)由平台提供，直接使用即可，不用特别设置。

请在右侧代码文件编辑器适当的位置补充代码，实现上述编程要求。

附client表的结构: 表1 client(客户表)

| **字段名称** | **数据类型** | **约束** | **说明** |
| --- | --- | --- | --- |
| c\_id | INTEGER | PRIMARY KEY | 客户编号 |
| c\_name | VARCHAR(100) | NOT NULL | 客户名称 |
| c\_mail | CHAR(30) | UNIQUE | 客户邮箱 |
| c\_id\_card | CHAR(20) | UNIQUE NOT NULL | 客户身份证 |
| c\_phone | CHAR(20) | UNIQUE NOT NULL | 客户手机号 |
| c\_password | CHAR(20) | NOT NULL | 客户登录密码 |