Using JDBC to connect to a relational Database

To compile the Java program you need the JDBC library for the database that want to connect such as MySQL or PostgreSQL etc.

You can download PostgreSQL JDBC Driver from:

- https://jdbc.postgresql.org/download/

Import necessary SQL classes in your class:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

MySql sample code:

```
public static Connection connect(String host, int port, String dbName, String user, String passwd)
{
    Connection dbConnection = null;
    try
    {
        String dbString = null;
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        dbString = "jdbc:mysql://" + host + "/" + dbName;
        dbConnection = DriverManager.getConnection(dbString, user, passwd);
    }
    catch (Exception e)
    {
        System.err.println("Failed to connect with the DB");
        e.printStackTrace();
    }
    return dbConnection;
}
```

Postgresql replacement in the above example:

```
Class.forName("org.postgresql.Driver").newInstance
jdbc:postgresql://host:port/database
```

```
private static void executeStatement(String q, Connection con)
{
    try
    {
        Statement stmt = con.createStatement();
        stmt.execute(q);
        stmt.close();
    }
    catch (SQLException e)
    {
        System.out.println("Statement: \n" + q);
        e.printStackTrace();
        throw new RuntimeException("Query " + q);
    }
}
```

```
public static ResultSet executeQuery(String query, Connection con)
{
    try
    {
        Statement stmt = con.createStatement();
        ResultSet results = stmt.executeQuery(query);
        return results;
    }
    catch (Exception e)
    {
        e.printStackTrace();
        return null;
    }
}
```

```
public static void main(String[] args) {
    Connection con = connect("ares.science.unitn.it", 3306, "db", "ppuser", "ppPasswd");
    String command;

    command = "create table Students(sid varchar(25), name varchar(50), dateOfBirth varchar(25))";
    executeStatement(command, con);

    command = "insert into Students values (\"s1\", \"John\", \"01/01/1999\")";
    executeStatement(command, con);
    command = "insert into Students values (\"s2\", \"Mary\", \"02/02/2000\")";
    executeStatement(command, con);

    command = "select * from Students";
    ResultSet results = executeQuery(command, con);

    try
    {
        while (results.next())
        {
            String cid = results.getString(1);
            String stName = results.getString(2);
            //String age = results.getInt(1);
            System.out.println("cid=" + cid + " stName=" + stName);
        }
        Statement stmt = results.getStatement();
        results.close();
        stmt.close();
    }
    catch (Exception e)
    {
        e.printStackTrace();
        throw new RuntimeException();
    }


    command = "drop table Students";
    executeStatement(command, con);

    try
    {
        con.close();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

SQL Injection Attack

```
String query = "SELECT userName, balance FROM accounts"
    + "WHERE userID=" + request.getParameter("userID") +
  "and password='" + request.getParameter("Password") + "'";

try
{
    Statement statement = connection.createStatement();
    ResultSet rs = statement.executeQuery(query);
    while (rs.next())
    {
        page.addTableRow(rs.getString("userName"),
                         rs.getFloat("balance"));
    }
}
catch (SQLException e)
    {}
```

Executed SQL Under Normal Condition:

```
SELECT userName, balance
FROM accounts
WHERE userID=512 and password='thisisyoda'
```

A possible SQL injection attack:

```
userID = 1' or '1' = '1
password = 1' or '1' = '1
```

Executed Under Attack Condition:

```
SELECT userName, balance FROM
accounts WHERE userID='1' OR '1'='1' and password='1' OR '1'='1'
```

So it is equal to `SELECT userName, balance FROM accounts` without any where clause and reveal all the username and passwords

Develop Safe Code using PreparedStatement instead of Statement

```
String query = "SELECT userName, balance "+
               "FROM accounts WHERE userID = ?
                and password = ?";

try {
  PreparedStatement statement = connection.prepareStatement(query);
  statement.setInt(1, request.getParameter("userID"));
  ResultSet rs = statement.executeQuery();
  while (rs.next())
  {
    page.addTableRow(rs.getString("userName"),
                     rs.getFloat("balance"));
  }
} catch (SQLException e)
       { ... }
```

It is also possible to change the DB using SQL injection attack