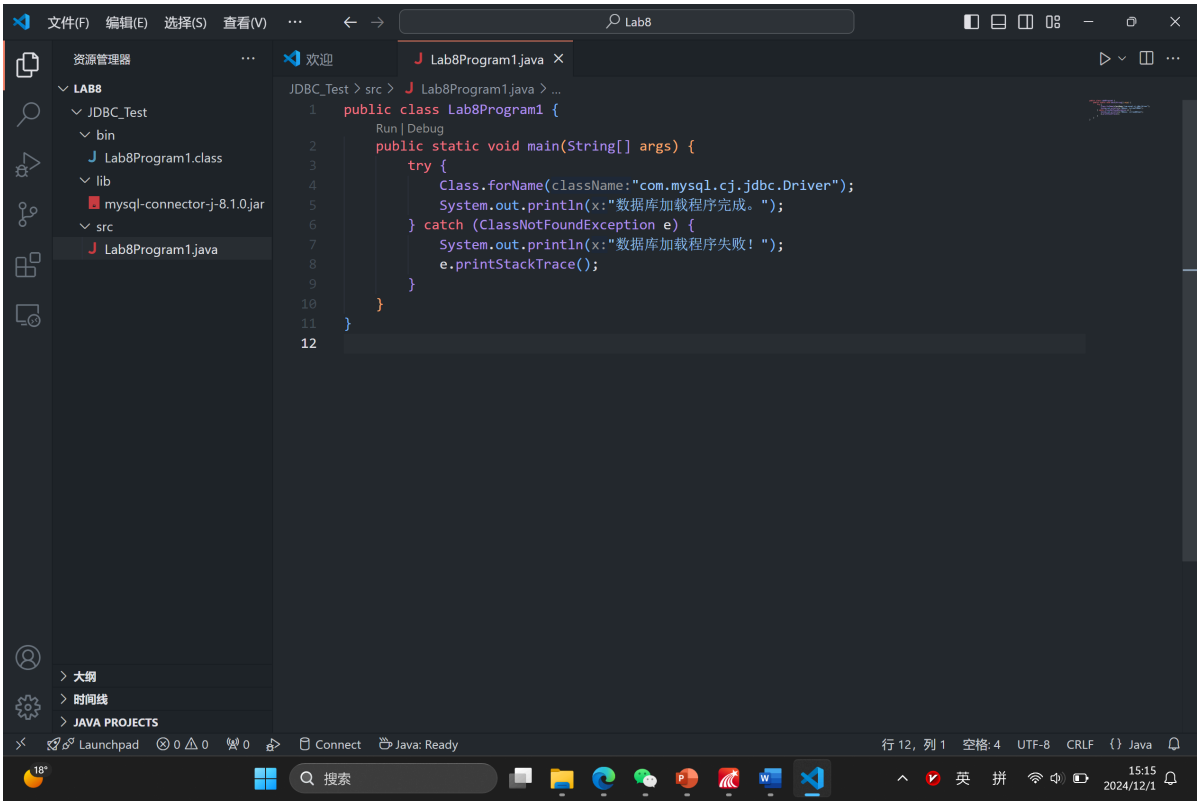
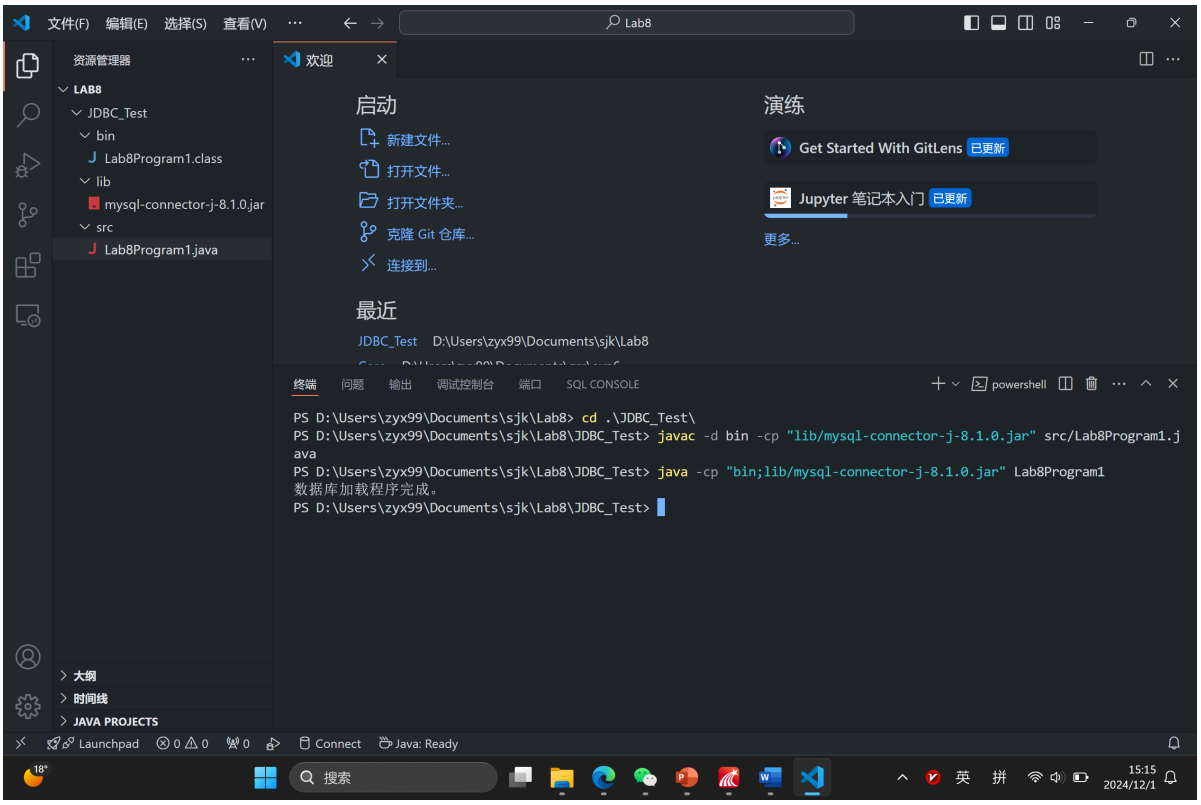
第10题：

高版本的JDBC不需要显式加载；不过为了测试，仍然保留这一步。
加载的代码：

```
/*
 \* 初始化文件，只需在项目的一开始运行一次即可。
 \* 事实上现代JDBC很多也可以省略这一步。
 \* 不过在这里可以用来测试项目构建是否正确。
 \* 使用一些现在IDE可以避免使用命令行，不过我用vscode习惯了。
*/

public class Lab8Program1 {
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("数据库加载程序完成。");
        } catch (ClassNotFoundException e) {
            System.out.println("数据库加载程序失败！");
            e.printStackTrace();
        }
    }
}
```



使用命令行运行（需要注意文件的位置组织。javac命令执行后会出现.class文件。）：



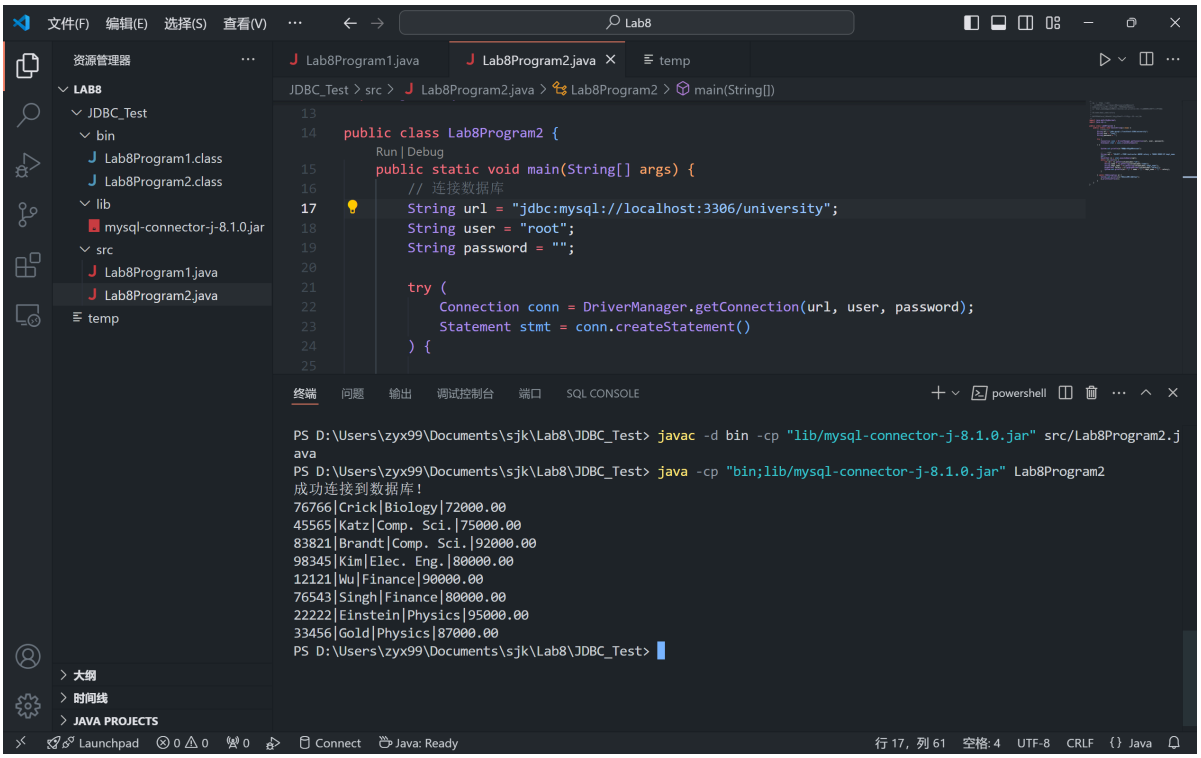
测试成功。

实现输出要求表格的代码：

```
/*
 * 10. (简答题, 7.8分)
 * 加载JDBC驱动，使用Java连接university数据库，
 * 查询instructor表中salary大于70000的教师信息，
 * 按照dept_name的升序排列后用System.out.println()输出，各列数据之间用|分隔。
 *
 * ID|name|dept_name|salary
 *
 * 将编写的Java类代码贴在下方输入框中，并给出输出结果。
 */
import java.math.BigDecimal;
import java.sql.*;

public class Lab8Program2 {
    public static void main(String[] args) {
        // 连接数据库
        String url = "jdbc:mysql://localhost:3306/university";
        String user = "root";
        String password = "";

        try (
            Connection conn = DriverManager.getConnection(url, user, password);
            Statement stmt = conn.createStatement()
        ) {

            System.out.println("成功连接到数据库！");

            // 执行查询
            String sql = "SELECT * FROM instructor WHERE salary > 70000 ORDER BY
dept_name ASC";
```

```
            ResultSet rs = stmt.executeQuery(sql);
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                String dept_name = rs.getString("dept_name");
                BigDecimal salary = rs.getBigDecimal("salary");
                System.out.println(id + "|" + name + "|" + dept_name + "|" +
salary);
            }

        } catch (SQLException e) {
            System.out.println("数据库操作失败！");
            e.printStackTrace();
        }
    }
}
```

结果：



和直接使用SQL得到的结果相同。
注：这里使用BigDecimal是因为salary的类型是decimal(8,2)，使用int可行但不合适。

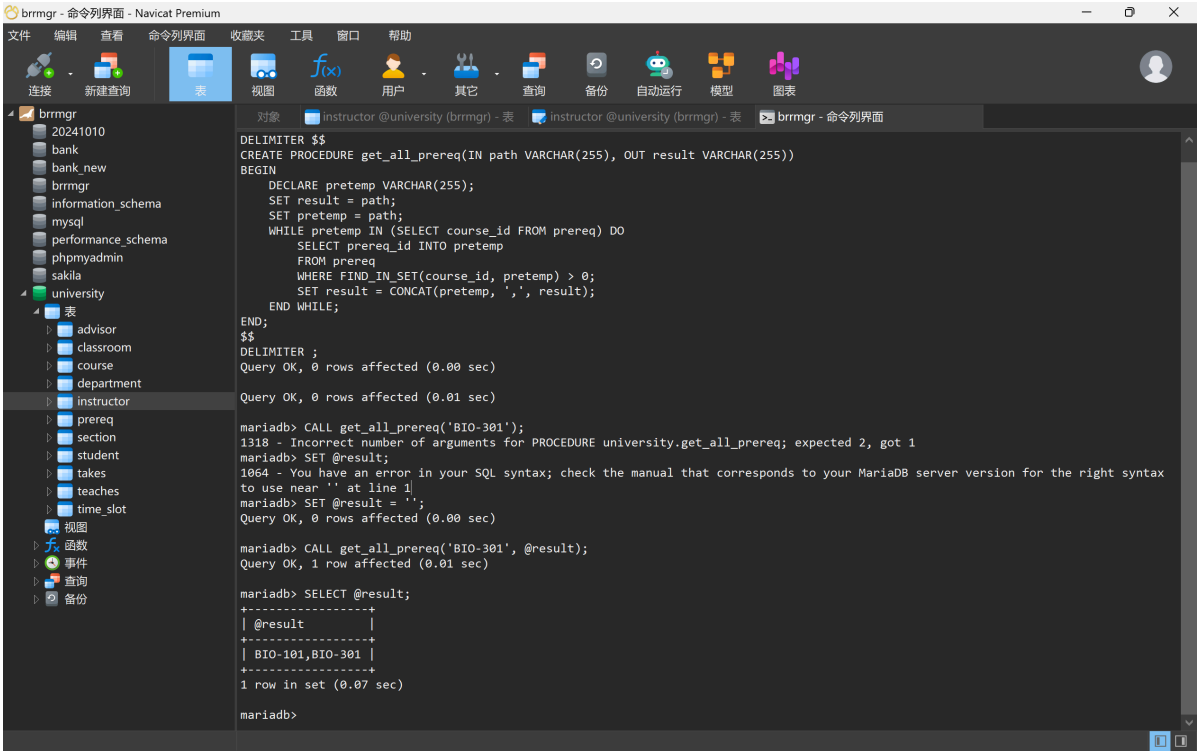
第11题：

（1）我在这里使用的是上一次作业中查询先修课程的存储函数，经过微改得到存储过程。

```
DROP PROCEDURE IF EXISTS get_all_prereq;
DELIMITER $$
CREATE PROCEDURE get_all_prereq(IN path VARCHAR(255), OUT result VARCHAR(255))
BEGIN
  DECLARE pretemp VARCHAR(255);
  SET result = path;
  SET pretemp = path;
  WHILE pretemp IN (SELECT course_id FROM prereq) DO
    SELECT prereq_id INTO pretemp
```

```
    FROM prereq
    WHERE FIND_IN_SET(course_id, pretemp) > 0;
    SET result = CONCAT(pretemp, ',', result);
  END WHILE;
END;
$$
DELIMITER ;
```

演示：

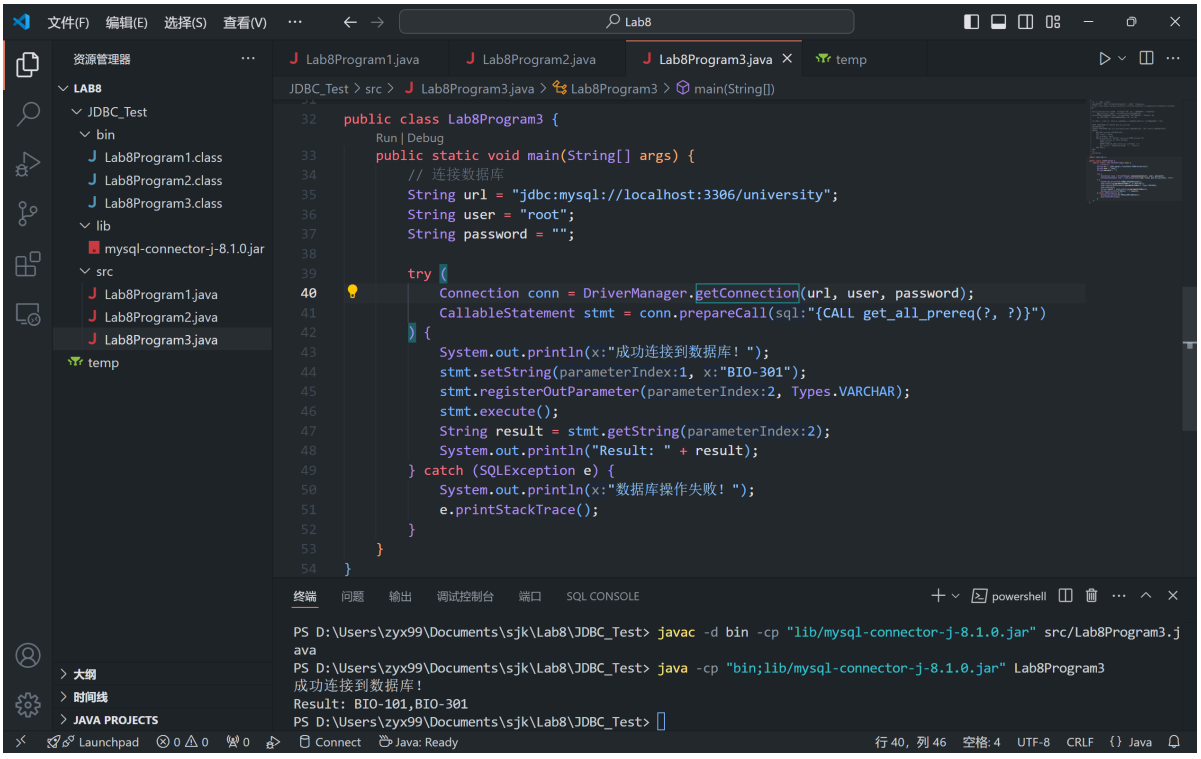


（2）代码实现：

```
/*
 * 11. (简答题, 7.8分)
 * 查阅资料学习JDBC CallableStatements执行存储过程的方法:
 * https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-usagenotes-
statements-callable.html
 *
 * （1）在Univerisity中编写一个带输入参数和输出参数的存储过程demoSp，
 *       功能自定，给出存储过程定义并说明自定义的功能。
 * （2）编写Java代码调用存储过程，指定输入参数，获取输出参数后输出。
 *       给出调用存储过程的关键代码并给出输出结果。
 */
import java.sql.*;

public class Lab8Program3 {
    public static void main(String[] args) {
        // 连接数据库
        String url = "jdbc:mysql://localhost:3306/university";
        String user = "root";
        String password = "";

        try (
            Connection conn = DriverManager.getConnection(url, user, password);
```

```
            CallableStatement stmt = conn.prepareCall("{CALL get_all_prereq(?,
?)}")
        ) {
            System.out.println("成功连接到数据库！");
            stmt.setString(1, "BIO-301");
            stmt.registerOutParameter(2, Types.VARCHAR);
            stmt.execute();
            String result = stmt.getString(2);
            System.out.println("Result: " + result);
        } catch (SQLException e) {
            System.out.println("数据库操作失败！");
            e.printStackTrace();
        }
    }
}
```

结果：



与数据库查询时相同。