Recurso bitácora de vuelo – Reto 4

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class ReportesView {

    private static ReportesController controller;

    public ReportesView() {
        controller = new ReportesController();
    }

    private static String repitaCaracter(Character caracter, Integer veces) {
        String respuesta = "";
        for (int i = 0; i < veces; i++) {
            respuesta += caracter;
        }
        return respuesta;
    }

    public void proyectosFinanciadosPorBanco(String banco) {
        System.out.println(
                repitaCaracter('=', 36) + " LISTADO DE PROYECTOS POR BANCO " +
repitaCaracter('=', 37));
        if (banco != null && !banco.isBlank()) {
            System.out.println(String.format("%3s %-25s %-20s %-15s %-7s %-30s", "ID",
"CONSTRUCTORA", "CIUDAD",
                    "CLASIFICACION", "ESTRATO", "LIDER"));
            System.out.println(repitaCaracter('-', 105));
            try {
```

```
                List<ProyectoBancoVo> proyectos =
controller.listarProyectosPorBanco(banco);
                for (ProyectoBancoVo proyecto : proyectos) {
                    System.out.println(proyecto);
                }
            } catch (Exception ex) {
                System.err.println("Error: " + ex);
                ex.printStackTrace();
            }
        }
    }

    public void totalAdeudadoPorProyectosSuperioresALimite(Double limiteInferior) {
        System.out.println(
                repitaCaracter('=', 1) + " TOTAL DEUDAS POR PROYECTO " +
repitaCaracter('=', 1));
        if (limiteInferior != null) {
            System.out.println(String.format("%3s %15s", "ID", "VALOR  "));
            System.out.println(repitaCaracter('-', 29));
            try {
                List<DeudasPorProyectoVo> proyectos =
controller.listarTotalAdeudadoProyectos(limiteInferior);
                for (DeudasPorProyectoVo proyecto : proyectos) {
                    System.out.println(proyecto);
                }
            } catch (Exception ex) {
                System.err.println("Error: " + ex);
                ex.printStackTrace();
            }
        }
    }

    public void lideresQueMasGastan() {
        System.out.println(
                repitaCaracter('=', 6) + " 10 LIDERES MAS COMPRADORES " +
repitaCaracter('=', 7));
        System.out.println(String.format("%-25s %15s", "LIDER", "VALOR  "));
        System.out.println(repitaCaracter('-', 41));
        try {
            List<ComprasDeLiderVo> comprasLideres =
controller.listarLideresQueMasGastan();
            for (ComprasDeLiderVo comprasLider : comprasLideres) {
                System.out.println(comprasLider);
            }
        } catch (Exception ex) {
```

```
            System.err.println("Error: " + ex);
            ex.printStackTrace();
        }
    }
}


public class ReportesController {
    private ProyectoBancoDao proyectoBancoDao;
    private ComprasDeLiderDao comprasDeLiderDao;
    private DeudasPorProyectoDao pagadoPorProyectoDao;

    public ReportesController() {
        proyectoBancoDao = new ProyectoBancoDao();
        comprasDeLiderDao = new ComprasDeLiderDao();
        pagadoPorProyectoDao = new DeudasPorProyectoDao();
    }

    public List<ProyectoBancoVo> listarProyectosPorBanco(String banco) throws
SQLException {
        return proyectoBancoDao.listar(banco);
    }

    public List<DeudasPorProyectoVo> listarTotalAdeudadoProyectos(Double limite)
throws SQLException {
        return pagadoPorProyectoDao.listar(limite);
    }

    public List<ComprasDeLiderVo> listarLideresQueMasGastan() throws SQLException {
        return comprasDeLiderDao.listar();
    }
}


public class ComprasDeLiderDao {
    public List<ComprasDeLiderVo> listar() throws SQLException {
        ArrayList<ComprasDeLiderVo> respuesta = new ArrayList<ComprasDeLiderVo>();
        Connection conn = JDBCUtilities.getConnection();
        Statement stmt = null;
        ResultSet rset = null;
        String consulta = "SELECT L.NOMBRE ||' '||L.PRIMER_APELLIDO ||'
'||L.SEGUNDO_APELLIDO AS LIDER,"
                + "       SUM(C.CANTIDAD * MC.PRECIO_UNIDAD) AS VALOR"
                + " FROM LIDER L"
                + " JOIN PROYECTO P ON (P.ID_LIDER = L.ID_LIDER)"
```

```
                + " JOIN COMPRA C ON (P.ID_PROYECTO = C.ID_PROYECTO)"
                + " JOIN MATERIALCONSTRUCCION MC ON (C.ID_MATERIALCONSTRUCCION =
MC.ID_MATERIALCONSTRUCCION)"
                + " GROUP BY L.NOMBRE ||' '||L.PRIMER_APELLIDO ||'
'||L.SEGUNDO_APELLIDO"
                + " ORDER BY VALOR DESC"
                + " LIMIT 10;";
        try {
            stmt = conn.createStatement();
            rset = stmt.executeQuery(consulta);
            while (rset.next()) {
                ComprasDeLiderVo vo = new ComprasDeLiderVo();
                vo.setLider(rset.getString("Lider"));
                vo.setValor(rset.getDouble("Valor"));

                respuesta.add(vo);
            }
        } finally {
            if (rset != null) {
                rset.close();
            }
            if (stmt != null) {
                stmt.close();
            }
            if (conn != null) {
                conn.close();
            }
        }
        return respuesta;
    }
}

public class DeudasPorProyectoDao {
    public List<DeudasPorProyectoVo> listar(Double limite) throws SQLException {
        ArrayList<DeudasPorProyectoVo> respuesta = new
ArrayList<DeudasPorProyectoVo>();
        Connection conn = JDBCUtilities.getConnection();
        PreparedStatement stmt = null;
        ResultSet rset = null;
        String consulta = "SELECT P.ID_PROYECTO AS ID, SUM(C.CANTIDAD *
MC.PRECIO_UNIDAD) AS VALOR"
                + " FROM PROYECTO P"
                + " JOIN COMPRA C ON (P.ID_PROYECTO = C.ID_PROYECTO)"
                + " JOIN MATERIALCONSTRUCCION MC ON (C.ID_MATERIALCONSTRUCCION =
MC.ID_MATERIALCONSTRUCCION)"
```

```
                + " WHERE C.PAGADO = 'No'"
                + " GROUP BY P.ID_PROYECTO"
                + " HAVING SUM(C.CANTIDAD * MC.PRECIO_UNIDAD) > ?"
                + " ORDER BY VALOR DESC";
        try {
            stmt = conn.prepareStatement(consulta);
            stmt.setDouble(1, limite);
            rset = stmt.executeQuery();
            while (rset.next()) {
                DeudasPorProyectoVo vo = new DeudasPorProyectoVo();
                vo.setId(rset.getInt("ID"));
                vo.setValor(rset.getDouble("VALOR"));

                respuesta.add(vo);
            }
        } finally {
            if (rset != null) {
                rset.close();
            }
            if (stmt != null) {
                stmt.close();
            }
            if (conn != null) {
                conn.close();
            }
        }
        return respuesta;
    }
}


public class ProyectoBancoDao {
    public List<ProyectoBancoVo> listar(String banco) throws SQLException {
        ArrayList<ProyectoBancoVo> respuesta = new ArrayList<ProyectoBancoVo>();
        Connection conn = JDBCUtilities.getConnection();
        PreparedStatement stmt = null;
        ResultSet rset = null;
        String consulta = "SELECT P.ID_PROYECTO AS ID, P.CONSTRUCTORA, P.CIUDAD,
P.CLASIFICACION,"
                + "       T.ESTRATO, L.NOMBRE||' '||L.PRIMER_APELLIDO||'
'||L.SEGUNDO_APELLIDO AS LIDER"
                + " FROM PROYECTO P"
                + " JOIN TIPO T ON (P.ID_TIPO = T.ID_TIPO)"
                + " JOIN LIDER L ON (P.ID_LIDER = L.ID_LIDER)"
                + " WHERE P.BANCO_VINCULADO = ?"
```

```
                + " ORDER BY FECHA_INICIO DESC, CIUDAD, CONSTRUCTORA";
        try {
            stmt = conn.prepareStatement(consulta);
            stmt.setString(1, banco);
            rset = stmt.executeQuery();
            while (rset.next()) {
                ProyectoBancoVo vo = new ProyectoBancoVo();
                vo.setId(rset.getInt("id"));
                vo.setConstructora(rset.getString("constructora"));
                vo.setCiudad(rset.getString("ciudad"));
                vo.setClasificacion(rset.getString("clasificacion"));
                vo.setEstrato(rset.getInt("estrato"));
                vo.setLider(rset.getString("lider"));

                respuesta.add(vo);
            }
        } finally {
            if (rset != null) {
                rset.close();
            }
            if (stmt != null) {
                stmt.close();
            }
            if (conn != null) {
                conn.close();
            }
        }
        return respuesta;
    }
}


public class ComprasDeLiderVo {
    private String lider;
    private Double valor;

    public String getLider() {
        return lider;
    }

    public void setLider(String lider) {
        this.lider = lider;
    }

    public Double getValor() {
```

```
        return valor;
    }

    public void setValor(Double valor) {
        this.valor = valor;
    }

    @Override
    public String toString() {
        return String.format("%-25s %,15.1f", lider, valor);
    }
}


public class DeudasPorProyectoVo {
    private Integer id;
    private Double valor;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public Double getValor() {
        return valor;
    }

    public void setValor(Double valor) {
        this.valor = valor;
    }

    @Override
    public String toString() {
        return String.format("%3d %,15.1f", id, valor);
    }
}


public class ProyectoBancoVo {
    private Integer id;
    private String constructora;
    private String ciudad;
```

```
    private String clasificacion;
    private Integer estrato;
    private String lider;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getConstructora() {
        return constructora;
    }

    public void setConstructora(String constructora) {
        this.constructora = constructora;
    }

    public String getCiudad() {
        return ciudad;
    }

    public void setCiudad(String ciudad) {
        this.ciudad = ciudad;
    }

    public String getClasificacion() {
        return clasificacion;
    }

    public void setClasificacion(String clasificacion) {
        this.clasificacion = clasificacion;
    }

    public Integer getEstrato() {
        return estrato;
    }

    public void setEstrato(Integer estrato) {
        this.estrato = estrato;
    }

    public String getLider() {
```

```
        return lider;
    }

    public void setLider(String lider) {
        this.lider = lider;
    }

    @Override
    public String toString() {
        return String.format("%3d %-25s %-20s %-15s %7d %-30s",
                id, constructora, ciudad, clasificacion, estrato, lider);
    }
}


public class JDBCUtilities {
    private static final String UBICACION_BD = "ProyectosConstruccion.db";

    public static Connection getConnection() throws SQLException {
        String url = "jdbc:sqlite:" + UBICACION_BD;
        return DriverManager.getConnection(url);
    }
}
```