



Laboratorio 7

Persistencia

Integrantes:

Luisa Fernanda Bermúdez Girón

Karol Daniela Ladino Ladino

Profesor:

Iván Darío Viasus Quintero

Curso:

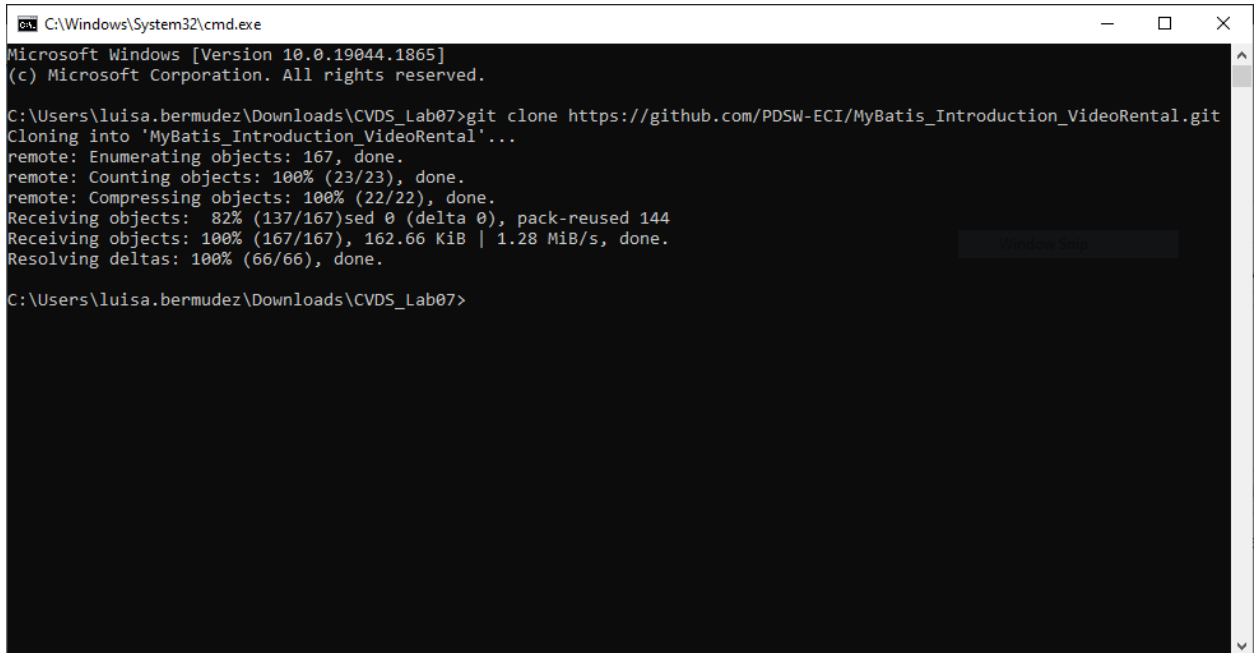
CVDS - 2

Fecha De Entrega:

06-10-2022

SECCIÓN I. - INTRODUCCIÓN A JDBC

1. Clonar el proyecto MyBatis_Introduction_VideoRental de GitHub donde se realizará la implementación completa del laboratorio.

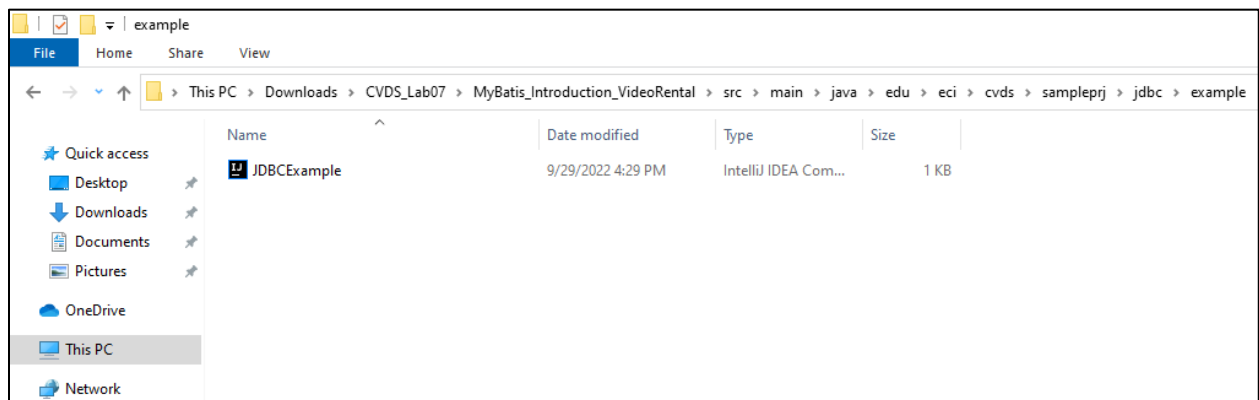


```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1865]
(c) Microsoft Corporation. All rights reserved.

C:\Users\luisa.bermudez\Downloads\CVDS_Lab07>git clone https://github.com/PDSW-ECI/MyBatis_Introduction_VideoRental.git
Cloning into 'MyBatis_Introduction_VideoRental'...
remote: Enumerating objects: 167, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (22/22), done.
Receiving objects: 82% (137/167) 0 (delta 0), pack-reused 144
Receiving objects: 100% (167/167), 162.66 KiB | 1.28 MiB/s, done.
Resolving deltas: 100% (66/66), done.

C:\Users\luisa.bermudez\Downloads\CVDS_Lab07>
```

2. Descargue el archivo JDBCExample.java y agréguelo en el paquete "edu.eci.cvds.sampleprj.jdbc.example".



3. Desde esta clase se realizará una conexión a una base de datos MySQL por medio de JDBC y sus "Prepared Statements".
Se reviso la clase que se puso en el proyecto.
4. En un motor de base de datos SQL se tiene un esquema con el siguiente modelo de base de datos (para registrar pedidos sobre productos):



5. Revise la documentación de PreparedStatement del API JDBC

The Java™ Tutorials

Using Prepared Statements

This page covers the following topics:

- Overview of Prepared Statements
- Creating a PreparedStatement Object
- Supplying Values for PreparedStatement Parameters

Overview of Prepared Statements

Sometimes it is more convenient to use a `PreparedStatement` object for sending SQL statements to the database. This special type of statement is derived from the more general class, `Statement`, that you already know.

If you want to execute a `Statement` object many times, it usually reduces execution time to use a `PreparedStatement` object instead.

The main feature of a `PreparedStatement` object is that, unlike a `Statement` object, it is given a SQL statement when it is created. The advantage to this is that in most cases, this SQL statement is sent to the DBMS right away, where it is compiled. As a result, the `PreparedStatement` object contains not just a SQL statement, but a SQL statement that has been precompiled. This means that when the `PreparedStatement` is executed, the DBMS can just run the `PreparedStatement` SQL statement without having to compile it first.

Although you can use `PreparedStatement` objects for SQL statements with no parameters, you probably use them most often for SQL statements that take parameters. The advantage of using SQL statements that take parameters is that you can use the same statement and supply it with different values each time you execute it. Examples of this are in the following sections.

However, the most important advantage of prepared statements is that they help prevent SQL injection attacks. SQL injection is a technique to maliciously exploit applications that use client-supplied data in SQL statements. Attackers trick the SQL engine into executing unintended commands by supplying specially crafted string input, thereby gaining unauthorized access to a database to view or manipulate restricted data. SQL injection techniques all exploit a single vulnerability in the application: incorrectly validated or nonvalidated string literals are concatenated into a dynamically built SQL statement and interpreted as code by the SQL engine. Prepared statements always treat client-supplied data as content of a parameter and never as a part of an SQL statement. See the section [SQL Injection in Database PL/SQL Language Reference](#), part of Oracle Database documentation, for more information.

The following method, `updateCoffeeSales`, stores the number of pounds of coffee sold in the current week in the `SALES` column for each type of coffee, and updates the total number of pounds of coffee sold in the `TOTAL` column for each type of coffee.

```

public void updateCoffeeSales(Map<String, Integer> salesForWeek) throws SQLException {
    String updateString =
        "update COFFEES set SALES = ? where COF_NAME = ?";
    String updateStatement =
        "update COFFEES set TOTAL = TOTAL + ? where COF_NAME = ?";

    try (PreparedStatement updateSales = con.prepareStatement(updateString);
        PreparedStatement updateTotal = con.prepareStatement(updateStatement)) {
        // ...
    }
}
  
```

6. En la clase JDBCExample juste los parámetros de conexión a la base de datos con los datos reales:

```

1 *JDBCExample.java X
22 import java.sql.ResultSet;
23 import java.sql.SQLException;
24 import java.util.LinkedList;
25 import java.util.List;
26 import java.util.logging.Level;
27 import java.util.logging.Logger;
28
29 /**
30  *
31  * @author hcadavid
32  */
33 public class JDBCExample {
34
35     public static void main(String args[]){
36         try {
37             String url="jdbc:mysql://desarrollo.is.escuelaing.edu.co:3306/bdprueba ";
38             String driver="com.mysql.jdbc.Driver";
39             String user="bdprueba";
40             String pwd="prueba2019";
41
42             Class.forName(driver);
43             Connection con=DriverManager.getConnection(url,user,pwd);
44             con.setAutoCommit(false);
45
46         }
47     }
48 }
  
```

7. Implemente las operaciones faltantes:

1. nombresProductosPedido

```
/**
 * Consultar los nombres de los productos asociados a un pedido
 * @param con la conexión JDBC
 * @param codigoPedido el código del pedido
 * @return
 */
public static List<String> nombresProductosPedido(Connection con, int codigoPedido){
    List<String> np=new LinkedList<>();
    String query = "Select nombre " + "From ORD_PRODUCTOS INNER JOIN ORD_DETALLE_PEDIDO detalle ON (codigo = producto_fk)" +
        "WHERE pedido_fk = ?";
    try {
        //Crear prepared statement
        prepareSt = con.prepareStatement(query);
        //asignar parámetros
        prepareSt.setInt(1, codigoPedido);
        //usar executeQuery
        ResultSet resultado = prepareSt.executeQuery();
        //Sacar resultados del ResultSet
        while(resultado.next()) {
            //llenar la lista y retornarla
            np.add(resultado.getString("nombre"));
        }
    }catch(SQLException e) {
        e.printStackTrace();
    }
    return np;
}
```

2. valorTotalPedido - El resultado final lo debe retornar la base de datos, no se deben hacer operaciones en memoria.

```
/**
 * Calcular el costo total de un pedido
 * @param con
 * @param codigoPedido código del pedido cuyo total se calculará
 * @return el costo total del pedido (suma de: cantidades*precios)
 */
public static int valorTotalPedido(Connection con, int codigoPedido){
    int total = 0;
    String query = "Select SUM(cantidad * precio) from ORD_DETALLE_PEDIDO inner join ORD_PRODUCTOS ON (codigo=producto_fk) WHERE pedido_fk = ?";
    try {
        //Crear prepared statement
        prepareSt = con.prepareStatement(query);
        //asignar parámetros
        prepareSt.setInt(1, codigoPedido);
        //usar executeQuery
        ResultSet resultado = prepareSt.executeQuery();

        //Sacar resultado del ResultSet
        while(resultado.next()) {
            total = resultado.getInt(1);
        }
    }catch(SQLException e) {
        e.printStackTrace();
    }

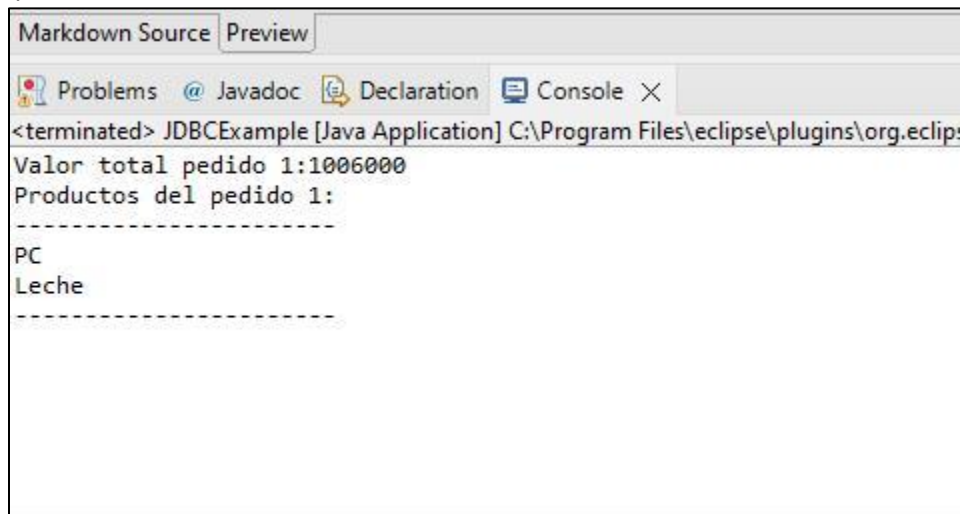
    return total;
}
```

3. registrarNuevoProducto - Use su código de estudiante para evitar colisiones.

```
/**
 * Agregar un nuevo producto con los parámetros dados
 * @param con la conexión JDBC
 * @param codigo
 * @param nombre
 * @param precio
 * @throws SQLException
 */
public static void registrarNuevoProducto(Connection con, int codigo, String nombre, int precio) throws SQLException{
    String query = "insert into ORD_PRODUCTOS values(?,?,?)";

    try {
        //Crear preparedStatement
        PreparedStatement prepareSt = con.prepareStatement(query);
        //Asignar parámetros
        prepareSt.setInt(1, codigo);
        prepareSt.setString(2, nombre);
        prepareSt.setInt(3, precio);
        //usar 'execute'
        prepareSt.executeUpdate();
        con.commit();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

8. Verifique por medio de un cliente SQL, que la información retornada por el programa coincide con la que se encuentra almacenada en base de datos.



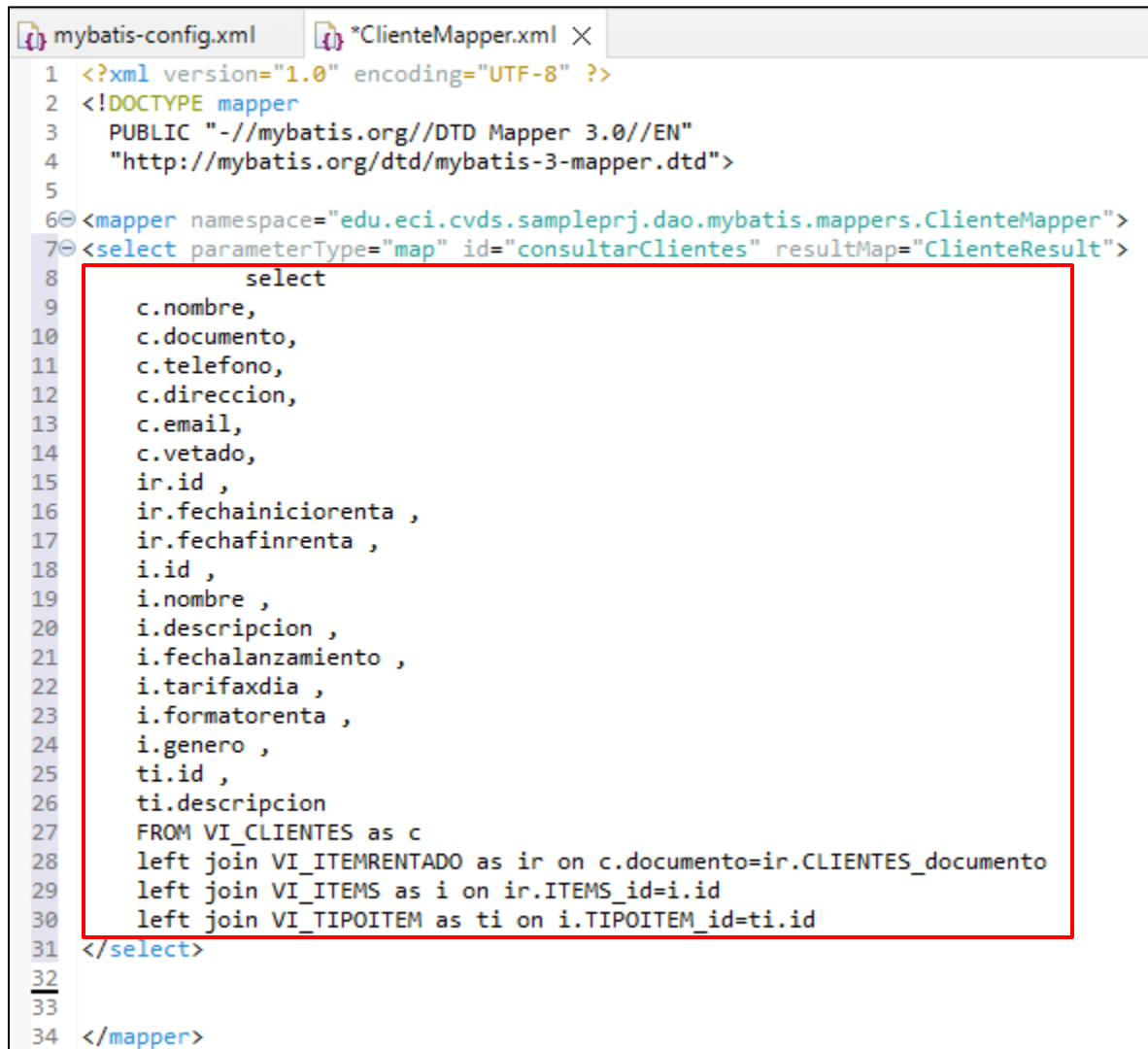
```
Markdown Source Preview
Problems @ Javadoc Declaration Console X
<terminated> JDBCExample [Java Application] C:\Program Files\eclipse\plugins\org.eclip
Valor total pedido 1:1006000
Productos del pedido 1:
-----
PC
Leche
-----
```

Parte I (Para entregar en clase)

1. Ubique los archivos de configuración para producción de MyBatis (mybatis-config.xml). Éste está en la ruta donde normalmente se ubican los archivos de configuración de aplicaciones montadas en Maven (src/main/resources). Edítelos y agregue en éste, después de la sección <settings> los siguientes 'typeAliases':

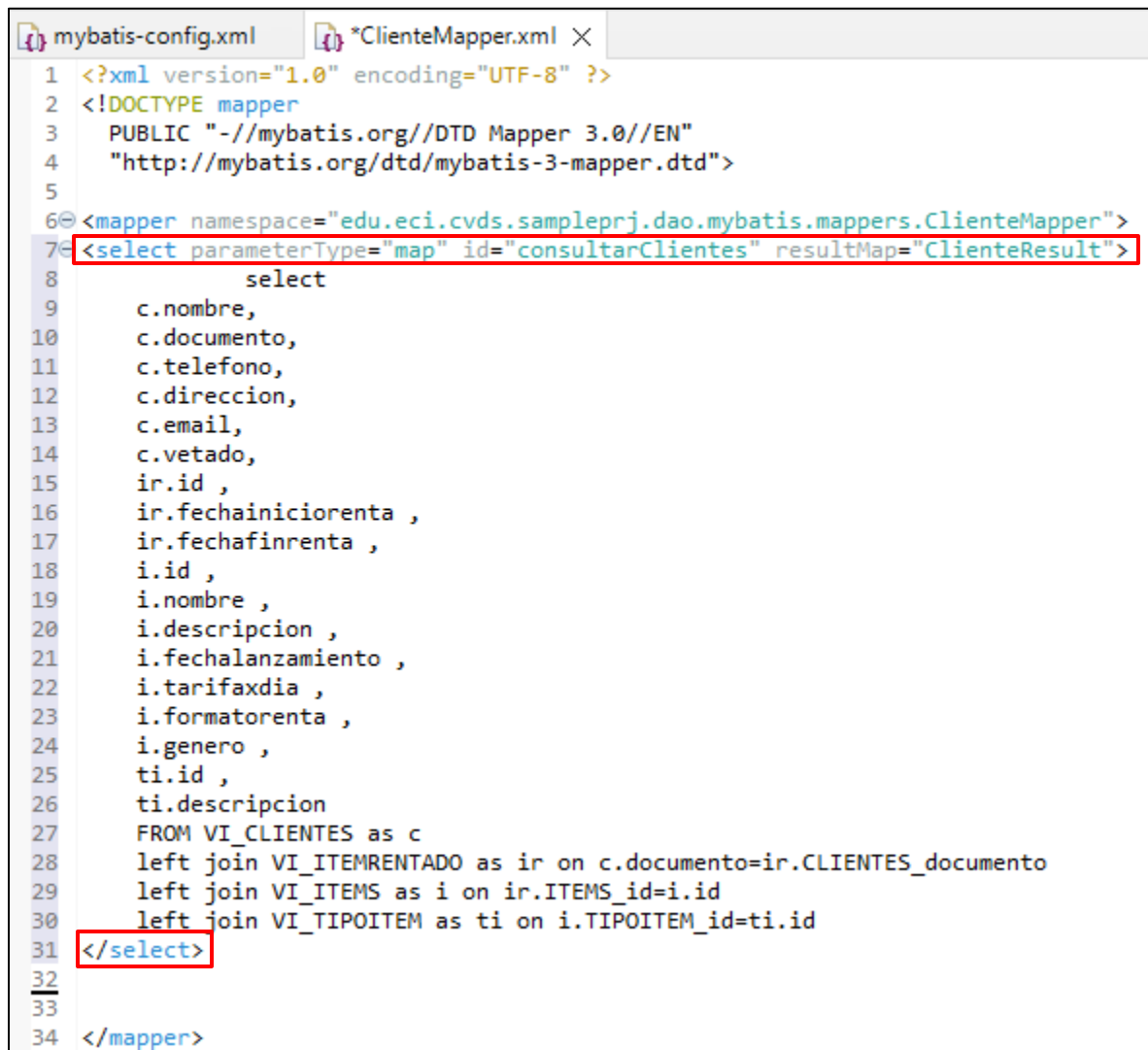
```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE configuration
3     PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-config.dtd">
5 <configuration>
6     <!--<settings>
7         <setting name="logImpl" value="LOG4J"/>
8     </settings-->
9
10    <properties resource="config.properties">
11    </properties>
12    <typeAliases>
13        <typeAlias type="edu.eci.cvds.samples.entities.Cliente" alias="Cliente"/>
14        <typeAlias type="edu.eci.cvds.samples.entities.Item" alias="Item"/>
15        <typeAlias type="edu.eci.cvds.samples.entities.ItemRentado" alias="ItemRentado"/>
16        <typeAlias type="edu.eci.cvds.samples.entities.TipoItem" alias="TipoItem"/>
17    </typeAliases>
18
19    <environments default="development">
20        <environment id="development">
21            <transactionManager type="JDBC" />
22            <dataSource type="POOLED">
23                <property name="driver" value="${driver}" />
24                <!--property name="url" value="jdbc:mysql://localhost:3306/video_rental"/!-->
25                <property name="url" value="${url}" />
26                <property name="username" value="${username}" />
27                <property name="password" value="${password}" />
28
29                <!--property name="url" value="jdbc:mysql://localhost:3306/video_rental"/!-->
30            </dataSource>
31        </environment>
32    </environments>
33
34    <mappers>
35        <mapper resource="mappers/ClienteMapper.xml"></mapper>
36        <mapper resource="mappers/ItemMapper.xml"></mapper>
37        <mapper resource="mappers/TipoItemMapper.xml"></mapper>
38    </mappers>
39
40
41 </configuration>
```

2. Lo primero que va a hacer es configurar un 'mapper' que permita que el framework reconstruya todos los objetos Cliente con sus detalles (ItemsRentados). Para hacer más eficiente la reconstrucción, la misma se realizará a partir de una sola sentencia SQL que relaciona los Clientes, sus Items Rentados, Los Items asociados a la renta, y el tipo de item. Ejecute esta sentencia en un cliente SQL (en las estaciones Linux está instalado EMMA), y revise qué nombre se le está asignando a cada columna del resultado:



```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper
3     PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6 <mapper namespace="edu.eci.cvds.sampleprj.dao.mybatis.mappers.ClienteMapper">
7 <select parameterType="map" id="consultarClientes" resultMap="ClienteResult">
8     select
9         c.nombre,
10        c.documento,
11        c.telefono,
12        c.direccion,
13        c.email,
14        c.vetado,
15        ir.id ,
16        ir.fechainiciorenta ,
17        ir.fechafinrenta ,
18        i.id ,
19        i.nombre ,
20        i.descripcion ,
21        i.fechalanzamiento ,
22        i.tarifaxdia ,
23        i.formatorenta ,
24        i.genero ,
25        ti.id ,
26        ti.descripcion
27    FROM VI_CLIENTES as c
28    left join VI_ITEMRENTADO as ir on c.documento=ir.CLIENTES_documento
29    left join VI_ITEMS as i on ir.ITEMS_id=i.id
30    left join VI_TIPOITEM as ti on i.TIPOITEM_id=ti.id
31 </select>
32
33
34 </mapper>
```

3. Abra el archivo XML en el cual se definirán los parámetros para que MyBatis genere el 'mapper' de Cliente (ClienteMapper.xml). Ahora, mapee un elemento de tipo <select> al método 'consultarClientes':



```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper
3     PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6 <mapper namespace="edu.eci.cvds.sampleprj.dao.mybatis.mappers.ClienteMapper">
7     <select parameterType="map" id="consultarClientes" resultMap="ClienteResult">
8         select
9             c.nombre,
10            c.documento,
11            c.telefono,
12            c.direccion,
13            c.email,
14            c.vetado,
15            ir.id ,
16            ir.fechainiciorenta ,
17            ir.fechafinrenta ,
18            i.id ,
19            i.nombre ,
20            i.descripcion ,
21            i.fechalanzamiento ,
22            i.tarifaxdia ,
23            i.formatorenta ,
24            i.genero ,
25            ti.id ,
26            ti.descripcion
27        FROM VI_CLIENTES as c
28        left join VI_ITEMRENTADO as ir on c.documento=ir.CLIENTES_documento
29        left join VI_ITEMS as i on ir.ITEMS_id=i.id
30        left join VI_TIPOITEM as ti on i.TIPOITEM_id=ti.id
31    </select>
32
33
34 </mapper>
```


4. Note que el mapeo hecho anteriormente, se indica que los detalles de a qué atributo corresponde cada columna del resultado de la consulta están en un 'resultMap' llamado "ClienteResult". En el XML del mapeo agregue un elemento de tipo <resultMap>, en el cual se defina, para una entidad(clase) en particular, a qué columnas estarán asociadas cada una de sus propiedades (recuerde que propiedad != atributo). La siguiente es un ejemplo del uso de la sintaxis de <resultMap> para la clase Maestro, la cual tiene una relación 'uno a muchos' con la clase DetalleUno y una relación 'uno a uno' con la clase DetalleDos, y donde -a la vez-, DetalleUno tiene una relación 'uno-a-uno-' con DetalleDos:

```
<resultMap type='Maestro' id='MaestroResult'>
  <id property='propiedad1' column='COLUMNA1'/>
  <result property='propiedad2' column='COLUMNA2'/>
  <result property='propiedad3' column='COLUMNA3'/>
  <association property="propiedad5" javaType="DetalleDos"></association>
  <collection property='propiedad4' ofType='DetalleUno'></collection>
</resultMap>

<resultMap type='DetalleUno' id='DetalleUnoResult'>
  <id property='propiedadx' column='COLUMNAX'/>
  <result property='propiedadY' column='COLUMNAY'/>
  <result property='propiedadZ' column='CUMNNAZ'/>
  <association property="propiedadw" javaType="DetalleDos"></association>
</resultMap>

<resultMap type='DetalleDos' id='DetalleDosResult'>
  <id property='propiedadR' column='COLUMNAR'/>
  <result property='propiedades' column='COLUMNAS'/>
  <result property='propiedadT' column='COLUMNAT'/>
</resultMap>
```

Como observa, Para cada propiedad de la clase se agregará un elemento de tipo <result>, el cual, en la propiedad 'property' indicará el nombre de la propiedad, y en la columna 'column' indicará el nombre de la columna de su tabla correspondiente (en la que se hará persistente). En caso de que la columna sea una llave primaria, en lugar de 'result' se usará un elemento de tipo 'id'. Cuando la clase tiene una relación de composición con otra, se agrega un elemento de tipo <association>. Finalmente, observe que si la clase tiene un atributo de tipo colección (List, Set, etc), se agregará un elemento de tipo <collection>, indicando (en la propiedad 'ofType') de qué tipo son los elementos de la colección. En cuanto al indentificador del 'resultMap', como convención se suele utilizar el nombre del tipo de dato concatenado con 'Result' como sufijo.

Teniendo en cuenta lo anterior, haga cuatro 'resultMap': uno para la clase Cliente, otro para la clase ItemRentado, otro para la clase Item, y otro para la clase Tipoltem.

```

mybatis-config.xml  *ClienteMapper.xml  TipotItem.java  ItemRentado.java
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE mapper
3    PUBLIC "-//mybatis.org/DTD Mapper 3.0//EN"
4    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6  <mapper namespace="edu.eci.cvds.sampleprj.dao.mybatis.mappers.ClienteMapper">
7    <resultMap type='Cliente' id='ClienteResult'>
8      <id property='documento' column='documento' />
9      <result property='nombre' column='nombre' />
10     <result property='telefono' column='telefono' />
11     <result property='direccion' column='direccion' />
12     <result property='vetado' column='vetado' />
13     <collection property='rentados' ofType='ItemRentado'></collection>
14   </resultMap>
15
16   <resultMap type='ItemRentado' id='ItemRentadoResult'>
17     <id property='id' column='id' />
18     <result property='fechainiciorenta' column='fechainiciorenta' />
19     <result property='fechafinrenta' column='fechafinrenta' />
20     <association property="item" javaType="Item"></association>
21   </resultMap>
22
23   <resultMap type='Item' id='ItemResult'>
24     <id property='id' column='id' />
25     <result property='nombre' column='nombre' />
26     <result property='descripcion' column='descripcion' />
27     <result property='fechaLanzamiento' column='fechaLanzamiento' />
28     <result property='tarifaxDia' column='tarifaxDia' />
29     <result property='formatoRenta' column='formatoRenta' />
30     <result property='genero' column='genero' />
31     <association property="tipo" javaType="TipoItem"></association>
32   </resultMap>
33
34   <resultMap type='TipoItem' id='TipoItemResult'>
35     <id property='id' column='id' />
36     <result property='descripcion' column='descripcion' />
37   </resultMap>

```

- Una vez haya hecho lo anterior, es necesario que en el elemento `<collection>` del maestro se agregue una propiedad que indique cual es el 'resultMap' a través del cual se podrá 'mapear' los elementos contenidos en dicha colección. Para el ejemplo anterior, como la colección contiene elementos de tipo 'Detalle', se agregará el elemento **resultMap** con el identificador del 'resultMap' de Detalle:

```
<collection property='propiedad3' ofType='Detalle' resultMap='DetalleResult'></collection>
```

Teniendo en cuenta lo anterior, haga los ajustes correspondientes en la configuración para el caso del modelo de Alquiler de películas.

```
<resultMap type='Cliente' id='ClienteResult'>
  <id property='documento' column='documento' />
  <result property='nombre' column='nombre' />
  <result property='telefono' column='telefono' />
  <result property='direccion' column='direccion' />
  <result property='vetado' column='vetado' />
  <collection property='rentados' ofType='ItemRentado' resultMap='ItemRentadoResult'></collection>
</resultMap>
```

6. Si intenta utilizar el 'mapper' tal como está hasta ahora, se puede presentar un problema: ¿qué pasa si las tablas a las que se les hace JOIN tienen nombres de columnas iguales?... Con esto MyBatis no tendría manera de saber a qué atributos corresponde cada una de las columnas. Para resolver esto, si usted hace un query que haga JOIN entre dos o más tablas, siempre ponga un 'alias' con un prefijo el query. Por ejemplo, si se tiene

```
select ma.propiedad1, det.propiedad1 ....
```

Se debería cambiar a:

```
select ma.propiedad1, det.propiedad1 as detalle_propiedad1
```

```
<select parameterType="map" id="consultarClientes" resultMap="ClienteResult">
  select
    c.nombre,
    c.documento,
    c.telefono,
    c.direccion,
    c.email,
    c.vetado,

    ir.id as itemren_id ,
    ir.fechainiciorenta as itemren_fechainiciorenta ,
    ir.fechafinrenta as itemren_fechafinrenta,

    i.id as item_id ,
    i.nombre as item_nombre ,
    i.descripcion as item_descripcion,
    i.fechalanzamiento as item_fechalanzamiento,
    i.tarifaxdia as item_tarifaxdia ,
    i.formatorenta as item_formatorenta ,
    i.genero as item_genero ,

    ti.id as tipoit_id ,
    ti.descripcion as tipoit_descripcion
  FROM VI_CLIENTES as c
  left join VI_ITEMRENTADO as ir on c.documento=ir.CLIENTES_documento
  left join VI_ITEMS as i on ir.ITEMS_id=i.id
  left join VI_TIPOITEM as ti on i.TIPOITEM_id=ti.id
</select>
```

Y posteriormente, en la 'colección' o en la 'asociación' correspondiente en el 'resultMap', indicar que las propiedades asociadas a ésta serán aquellas que tengan un determinado prefijo:

```
<resultMap type='Maestro' id='MaestroResult'>
  <id property='propiedad1' column='COLUMNA1' />
  <result property='propiedad2' column='COLUMNA2' />
  <result property='propiedad3' column='COLUMNA3' />
  <collection property='propiedad4' ofType='Detalle' columnPrefix='detalle_'></collection>
</resultMap>
```

```
<resultMap type='Cliente' id='ClienteResult'>
  <id property='documento' column='documento' />
  <result property='nombre' column='nombre' />
  <result property='telefono' column='telefono' />
  <result property='direccion' column='direccion' />
  <result property='vetado' column='vetado' />
  <collection property='rentados' ofType='ItemRentado' resultMap='ItemRentadoResult' columnPrefix='itemren_'></collection>
</resultMap>
```

```
<resultMap type='ItemRentado' id='ItemRentadoResult'>
  <id property='id' column='id' />
  <result property='fechainiciorenta' column='fechainiciorenta' />
  <result property='fechafinrenta' column='fechafinrenta' />
  <association property="item" javaType="Item" columnPrefix='item_'></association>
</resultMap>
```

```
<resultMap type='Item' id='ItemResult'>
  <id property='id' column='id' />
  <result property='nombre' column='nombre' />
  <result property='descripcion' column='descripcion' />
  <result property='fechalanzamiento' column='fechaLanzamiento' />
  <result property='tarifaxdia' column='tarifaxDia' />
  <result property='formatorenta' column='formatoRenta' />
  <result property='genero' column='genero' />
  <association property="tipo" javaType="TipoItem" columnPrefix='tipoit_'></association>
</resultMap>
```

7. Use el programa de prueba suministrado (MyBatisExample) para probar cómo a través del 'mapper' generado por MyBatis, se puede consultar un Cliente.

```
...
SqlSessionFactory sessionfact = getSqlSessionFactory();
SqlSession sqlss = sessionfact.openSession();
ClientMapper cm=sqlss.getMapper(ClientMapper.class);
System.out.println(cm.consultarClientes());
...
```

```

/**
 * Programa principal de ejemplo de uso de MyBATIS
 * @param args
 * @throws SQLException
 */
public static void main(String args[]) throws SQLException {
    SqlSessionFactory sessionfact = getSqlSessionFactory();

    SqlSession sqlss = sessionfact.openSession();

    ClienteMapper cm=sqlss.getMapper(ClienteMapper.class);
    System.out.println(cm.consultarClientes());

    sqlss.commit();

    sqlss.close();
}

```

Parte II (para el Miércoles)

1. Configure en el XML correspondiente, la operación consultarCliente(int id) del 'mapper' ClienteMapper.

En este caso, a diferencia del método anterior (cargar todos), el método asociado al 'mapper' tiene parámetros que se deben usar en la sentencia SQL. Es decir, el parámetro 'id' de *public Cliente consultarCliente(int id);* se debe usar en el WHERE de su correspondiente sentencia SQL. Para hacer esto tenga en cuenta:

- Agregue la anotación @Param a dicho parámetro, asociando a ésta el nombre con el que se referirá en la sentencia SQL:

```
public Cliente consultarCliente(@Param("idcli") int id);
```

```
public interface ClienteMapper {

    public Cliente consultarCliente(@Param("idcli")int id);
}
```

- Al XML (<select>, <insert>, etc) asociado al método del mapper, agregue la propiedad *parameterType="map"*.
- Una vez hecho esto, podrá hacer referencia dentro de la sentencia SQL a este parámetro a través de: `#{idcli}`

```

<select parameterType="map" id="consultarCliente" resultMap="ClienteResult">
    select
        c.nombre,
        c.documento,
        c.telefono,
        c.direccion,
        c.email,
        c.vetado,

        ir.id as itemren_id ,
        ir.fechainiciorenta as itemren_fechainiciorenta ,
        ir.fechafinrenta as itemren_fechafinrenta,

        i.id as item_id ,
        i.nombre as item_nombre ,
        i.descripcion as item_descripcion,
        i.fechalanzamiento as item_fechalanzamiento,
        i.tarifaxdia as item_tarifaxdia ,
        i.formatorenta as item_formatorenta ,
        i.genero as item_genero ,

        ti.id as tipoit_id ,
        ti.descripcion as tipoit_descripcion
    FROM VI_CLIENTES as c
    left join VI_ITEMRENTADO as ir on c.documento=ir.CLIENTES_documento
    left join VI_ITEMS as i on ir.ITEMS_id=i.id
    left join VI_TIPOITEM as ti on i.TIPOITEM_id=ti.id
    where c.documento = #{idcli}
</select>

```

2. Verifique el funcionamiento haciendo una consulta a través del 'mapper' desde MyBatisExample.
3. Configure en el XML correspondiente, la operación agregarItemRentadoACliente. Verifique el funcionamiento haciendo una consulta a través del 'mapper' desde MyBatisExample.

```

public void agregarItemRentadoACliente(@Param("idcli")int id,
    @Param("idit")int idit,
    @Param("fechainicio")Date fechainicio,
    @Param("fechafin")Date fechafin);

```

```

<insert parameterType="map" id="agregarItemRentadoACliente">
    insert into
        VI_ITEMRENTADO (CLIENTES_documento, ITEMS_id, fechainiciorenta, fechafinrenta)
    VALUES (#{idcli}, #{idit}, #{fechainicio}, #{fechafin});
</insert>

```

4. Configure en el XML correspondiente (en este caso ItemMapper.xml) la operación 'insertarItem(Item it). Para este tenga en cuenta:
 - o Al igual que en los dos casos anteriores, el query estará basado en los parámetros ingresados (en este caso, un objeto Item). En este caso, al hacer uso de la anotación @Param, la consulta

SQL se podrá componer con los atributos de dicho objeto. Por ejemplo, si al parámetro se le da como nombre ("item"): **insertarItem(@Param("item")Item it)**, en el query se podría usar **#{item.id}**, **#{item.nombre}**, **#{item.descripcion}**, etc. Verifique el funcionamiento haciendo una consulta a través del 'mapper' desde MyBatisExample.

```
public interface ItemMapper {

    public List<Item> consultarItems();

    public Item consultarItem(@Param("idit")int id);

    public void insertarItem(@Param("item")Item it);

}
```

```
<insert parameterType="map" id="insertarItem">
    insert into
    VI_ITEMS (id, nombre, descripcion, fechaLanzamiento, tarifaxDia, formatoRenta, genero, TIPOITEM_id)
    VALUES (#{item.id}, #{item.nombre}, #{item.descripcion}, #{item.fechaLanzamiento}, #{item.tarifaxDia}, #{item.formatoRenta},
    #{item.genero}, #{item.tipo.id});
</insert>
```

5. Configure en el XML correspondiente (de nuevo en ItemMapper.xml) las operaciones 'consultarItem(int it)' y 'consultarItems()' de ItemMapper. En este caso, tenga adicionalmente en cuenta:
 - Para poder configurar dichas operaciones, se necesita el 'resultMap' definido en ClientMapper. Para evitar tener CODIGO REPETIDO, mueva el resultMap *ItemResult* de ClienteMapper.xml a ItemMapper.xml. Luego, como dentro de ClienteMapper el resultMap *ItemRentadoResult* requiere del resultMap antes movido, haga referencia al mismo usando como referencia absoluta en 'namespace' de ItemMapper.xml:

```
<resultMap type='ItemRentado' id="ItemRentadoResult">
<association
...resultMap='edu.eci.cvds.sampleprj.dao.mybatis.mappers.ItemMapper.ItemResult'
></association>
</resultMap>
```

ClienteMapper

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <TYPE mapper
3 <BLIC "-//mybatis.org/DTD Mapper 3.0//EN"
4 <http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6 <@pper namespace="edu.eci.cvds.sampleprj.dao.mybatis.mappers.ClienteMapper">
7 <@resultMap type="Cliente" id="ClienteResult">
8   <id property="documento" column="documento"/>
9   <result property="nombre" column="nombre"/>
10  <result property="telefono" column="telefono"/>
11  <result property="direccion" column="direccion"/>
12  <result property="vetado" column="vetado"/>
13  <collection property="rentados" ofType="ItemRentado" resultMap="ItemRentadoResult" columnPrefix="itemren_"></collection>
14 </resultMap>
15
16 <@resultMap type="ItemRentado" id="ItemRentadoResult">
17   <id property="id" column="id"/>
18   <result property="fechainiciorenta" column="fechainiciorenta"/>
19   <result property="fechafinrenta" column="fechafinrenta"/>
20   <association property="item" javaType="Item" resultMap="edu.eci.cvds.sampleprj.dao.mybatis.mappers.ItemMapper.ItemResult" columnPrefix="item_"></association>
21 </resultMap>
22
23
```

```
24 <select parameterType="map" id="consultarClientes" resultMap="ClienteResult">
25   select
26     c.nombre,
27     c.documento,
28     c.telefono,
29     c.direccion,
30     c.email,
31     c.vetado,
32
33     ir.id as itemren_id ,
34     ir.fechainiciorenta as itemren_fechainiciorenta ,
35     ir.fechafinrenta as itemren_fechafinrenta,
36
37     i.id as item_id ,
38     i.nombre as item_nombre ,
39     i.descripcion as item_descripcion,
40     i.fechalanzamiento as item_fechalanzamiento,
41     i.tarifaxdia as item_tarifaxdia ,
42     i.formatorenta as item_formatorenta ,
43     i.genero as item_genero ,
44
45     ti.id as tipoit_id ,
46     ti.descripcion as tipoit_descripcion
47   FROM VI_CLIENTES as c
48   left join VI_ITEMRENTADO as ir on c.documento=ir.CLIENTES_documento
49   left join VI_ITEMS as i on ir.ITEMS_id=i.id
50   left join VI_TIPOITEM as ti on i.TIPOITEM_id=ti.id
51 </select>
52
```



```

52
53  <select parameterType="map" id="consultarCliente" resultMap="ClienteResult">
54      select
55          c.nombre,
56          c.documento,
57          c.telefono,
58          c.direccion,
59          c.email,
60          c.vetado,
61
62          ir.id as itemren_id ,
63          ir.fechainiciorenta as itemren_fechainiciorenta ,
64          ir.fechafinrenta as itemren_fechafinrenta,
65
66          i.id as item_id ,
67          i.nombre as item_nombre ,
68          i.descripcion as item_descripcion,
69          i.fechalanzamiento as item_fechalanzamiento,
70          i.tarifaxdia as item_tarifaxdia ,
71          i.formatorenta as item_formatorenta ,
72          i.genero as item_genero ,
73
74          ti.id as tipoit_id ,
75          ti.descripcion as tipoit_descripcion
76      FROM VI_CLIENTES as c
77      left join VI_ITEMRENTADO as ir on c.documento=ir.CLIENTES_documento
78      left join VI_ITEMS as i on ir.ITEMS_id=i.id
79      left join VI_TIPOITEM as ti on i.TIPOITEM_id=ti.id
80      where c.documento = #{idcli}
81  </select>
82

```

```

82
83  <insert parameterType="map" id="agregarItemRentadoACliente">
84      insert into
85          VI_ITEMRENTADO (CLIENTES_documento, ITEMS_id, fechainiciorenta, fechafinrenta)
86          VALUES (#{idcli}, #{idit}, #{fechainicio}, #{fechafin});
87  </insert>
88
89  </mapper>

```

ItemMapper

```

ItemMapper.xml
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE mapper
3      PUBLIC "-//mybatis.org/DTD Mapper 3.0//EN"
4      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6  <mapper namespace="edu.eci.cvds.sampleprj.dao.mybatis.mappers.ItemMapper">
7      <resultMap type="Item" id="ItemResult">
8          <id property="id" column="id"/>
9          <result property="nombre" column="nombre"/>
10         <result property="descripcion" column="descripcion"/>
11         <result property="fechalanzamiento" column="fechalanzamiento"/>
12         <result property="tarifaxdia" column="tarifaxDia"/>
13         <result property="formatorenta" column="formatoRenta"/>
14         <result property="genero" column="genero"/>
15         <association property="tipo" javaType="TipoItem" resultMap="edu.eci.cvds.sampleprj.dao.mybatis.mappers.TipoItemMapper.TipoItemResult" columnPrefix="tipoit">
16         </association>
17     </resultMap>

```

```

17
18Ⓣ <select parameterType="map" id="consultarItems" resultMap="ItemResult">
19     select
20         i.id,
21         i.nombre,
22         i.descripcion,
23         i.fechalanzamiento,
24         i.tarifaxdia,
25         i.formatorenta,
26         i.genero,
27
28         ti.id as tipoit_id ,
29         ti.descripcion as tipoit_descripcion
30
31     FROM VI_ITEMS as i
32     left join VI_TIPOITEM as ti on i.TIPOITEM_id=ti.id
33 </select>
34

```

```

34
35Ⓣ <select parameterType="map" id="consultarItem" resultMap="ItemResult">
36     select
37         i.id,
38         i.nombre,
39         i.descripcion,
40         i.fechalanzamiento,
41         i.tarifaxdia,
42         i.formatorenta,
43         i.genero,
44
45         ti.id as tipoit_id ,
46         ti.descripcion as tipoit_descripcion
47
48     FROM VI_ITEMS as i
49     left join VI_TIPOITEM as ti on i.TIPOITEM_id=ti.id
50     where i.id = #{idit}
51 </select>
52

```

```

52
53Ⓣ <insert parameterType="map" id="insertarItem">
54     insert into
55     VI_ITEMS (id, nombre, descripcion, fechaLanzamiento, tarifaxDia, formatoRenta, genero, TIPOITEM_id)
56     VALUES (#{item.id}, #{item.nombre}, #{item.descripcion}, #{item.fechaLanzamiento}, #{item.tarifaxDia}, #{item.formatoRenta},
57     #{item.genero}, #{item.tipo.id});
58 </insert>
59
60 </mapper>

```

TipoltemMapper

```
TipoltemMapper.xml X
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper
3     PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6 <mapper namespace="edu.eci.cvds.sampleprj.dao.mybatis.mappers.TipoItemMapper">
7     <resultMap type='TipoItem' id='TipoItemResult'>
8         <id property='id' column='id' />
9         <result property='descripcion' column='descripcion' />
10    </resultMap>
11
12 </mapper>
13
```

Resultado consultas:

```
ClienteMapper cm=sqlss.getMapper(ClienteMapper.class);
System.out.println("-----");
System.out.println("-----Clientes-----");
System.out.println("-----");
System.out.println(cm.consultarClientes());
```

```
-----Clientes-----
[Cliente(nombre=xzdf, documento=7899871234564, rentados=
[], Cliente(nombre=holliiiiiiiiii, documento=8452451654645645684, rentados=
[ItemRentado{id=2159091, item=null, fechainiciorenta=2022-03-26, fechainfrenta=2022-04-07}, ItemRentado{id=2159093, item=null, fechainiciorenta=1928-03-28, fechainfrenta=1928-04-01}, ItemRentado{id=2159094, item=null, fechainiciorenta=2022-03-26, fechainfrenta=2022-04-07}], Cliente(nombre=alejandro Toro, documento=1874919427, rentados=
[ItemRentado{id=2158710, item=null, fechainiciorenta=2020-11-18, fechainfrenta=2020-11-20}, ItemRentado{id=2158918, item=null, fechainiciorenta=2021-04-08, fechainfrenta=2021-04-13}], Cliente(nombre=qr, documento=1874919426, rentados=
[ItemRentado{id=2158892, item=null, fechainiciorenta=2021-03-28, fechainfrenta=2021-04-18}, ItemRentado{id=2158893, item=null, fechainiciorenta=2021-03-28, fechainfrenta=2021-04-18}, ItemRentado{id=2158914, item=null, fechainiciorenta=2021-04-06, fechainfrenta=2021-04-11}, ItemRentado{id=2159097, item=null, fechainiciorenta=2022-03-26, fechainfrenta=2022-03-28}], Cliente(nombre=Pepe, documento=1874919425, rentados=
[ItemRentado{id=2158672, item=null, fechainiciorenta=2020-10-15, fechainfrenta=2020-10-17}, ItemRentado{id=2158674, item=null, fechainiciorenta=2020-10-15, fechainfrenta=2020-10-23}], Cliente(nombre=pruebalabi, documento=999999999, rentados=
[ItemRentado{id=2158917, item=null, fechainiciorenta=2021-04-08, fechainfrenta=2021-04-11}], Cliente(nombre=carlosvalentina, documento=2166131, rentados=
[ItemRentado{id=2132981, item=null, fechainiciorenta=2020-12-24, fechainfrenta=2020-12-25}, ItemRentado{id=2158538, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2158539, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2158540, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2158541, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2158542, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2158543, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2159124, item=null, fechainiciorenta=2022-10-15, fechainfrenta=2022-10-15}, ItemRentado{id=2159126, item=null, fechainiciorenta=2022-10-15, fechainfrenta=2022-10-15}]]]
```

```
System.out.println("-----");
System.out.println("-----Informacion Cliente-----");
System.out.println("-----");
cm.agregarItemRentadoACliente(2154421, 93, new Date(), new Date());
System.out.println(cm.consultarCliente(2154421));
```

```
-----Informacion Cliente-----
[Cliente(nombre=james, documento=2154421, rentados=
[ItemRentado{id=2132981, item=null, fechainiciorenta=2020-12-24, fechainfrenta=2020-12-25}, ItemRentado{id=2158538, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2158539, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2158540, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2158541, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2158542, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2158543, item=null, fechainiciorenta=2020-10-10, fechainfrenta=2020-10-20}, ItemRentado{id=2159124, item=null, fechainiciorenta=2022-10-15, fechainfrenta=2022-10-15}, ItemRentado{id=2159126, item=null, fechainiciorenta=2022-10-15, fechainfrenta=2022-10-15}]]]
```

```
ItemMapper im=sqlss.getMapper(ItemMapper.class);
System.out.println("-----");
System.out.println("-----Items-----");
System.out.println("-----");
System.out.println(im.consultarItems());
```

```
Item nuevoItem = new Item(new TipoItem(2, "ficción", 52, "star wars", "película", new Date(), 3000, "dvd", "ficción"));
im.insertarItem(nuevoItem);
System.out.println("-----");
System.out.println("Item agregado");
System.out.println(im.consultarItem(52));
```

```
-----
Item agregado
Item{tipo=TipoItem{id=2, descripcion=Accion}, id=52, nombre=star wars, tarifaxDia=3000}
```

url repositorio: <https://github.com/LuisaGiron/ivn>