CICLO 2

[FORMACIÓN POR CICLOS]

Programación Básica JAVA

Semana 7

Ingeni@ Soluciones TIC

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Programa

Semana 7	Más de bases de datos relacionales	2	5	7
	Conexión a bases de datos con JDBC (I)	3	7	10
	Introducción a patrones multi-capa usando Modelo-Vista-Controlador (MVC)	2	5	7

Temas semana 7

- Bases de datos relacionales
- Ejercicio
- Conexión a una base de datos desde Java
- Ejemplos
- MVC
- Ejemplo

Conexión a base de datos

Para conectar a una base de datos se necesita:

- Driver
- URL
- Usuario
- Contraseña

```
Class.forName("com.mysql.cj.jdbc.Driver");
String url = "jdbc:mysql://localhost:3306/tienda";
Connection myConexion = DriverManager.getConnection(url, "USUARIO", "CONTRASEÑA");
```

Conexión a base de datos

Para enviar una consulta a la base de datos:

- Se crea la sentencia tipo ***String***
- Se crea un objeto tipo ***Statement***
- Si devuelve resultados se crea un objeto tipo ***ResultSet***
- Con los métodos ***get…*** se obtienen los datos de cada columna

```
String sentencia = "SELECT * FROM cliente;";
Statement consulta = myConexion.createStatement();
ResultSet resultados = consulta.executeQuery(sentencia);

while (resultados.next()) {
    cc = resultados.getInt("cc");
    nombre_1 = resultados.getString("nombre_1");
```

Conexión a base de datos

Cerrar la conexión:

- Se cierra la conexión

```
myConexion.close();
```

Conexión a base de datos

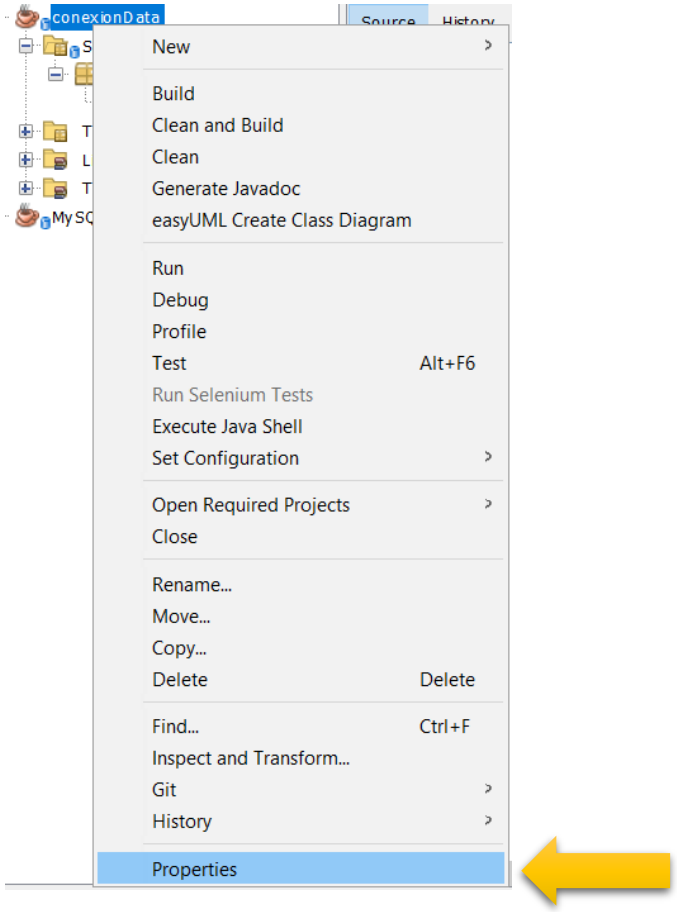
Parar el correcto manejo de bases de datos se usa la estructura Try Catch para capturar las excepciones que se presenten durante el proceso de interacción con la base de datos:

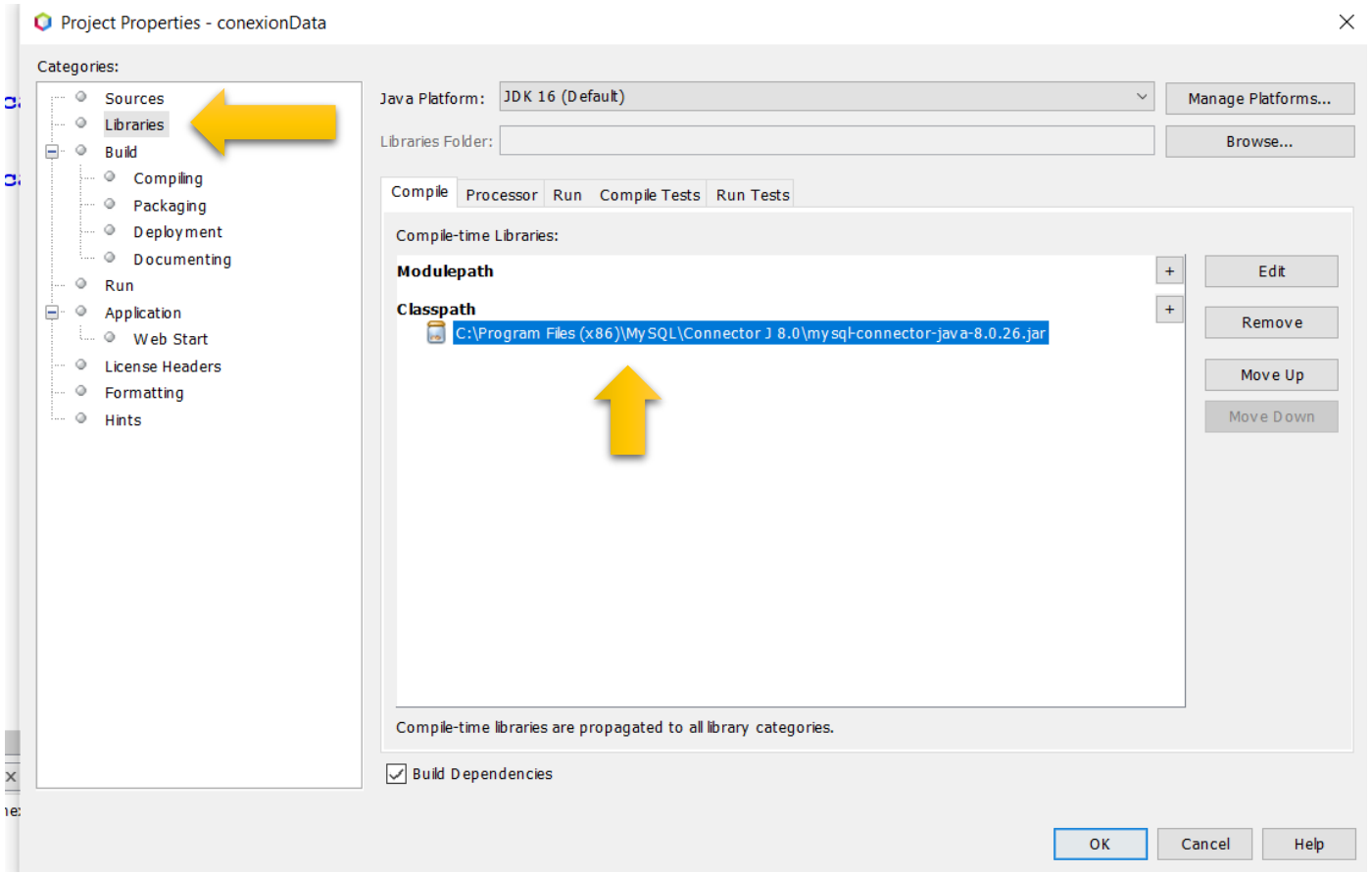
```
try{
    //INTERACCIÓN CON LA BASE DE DATOS

}catch(ClassNotFoundException e) {
    //CAPTURA DE LAS EXPCEPCIONES POR DRIVER
}catch(SQLException e ){
    //CAPTURA DE LAS EXCEPCIONES DE SQL
}
```

Conexión a base de datos

Se debe incluir dentro de las dependencias el jconector.
Se busca dentro de MySQL o se descarga de la página





Conexión a base de datos

Cree una base de datos **colegio** en Workbench y conéctese a ella desde java

.

- Normalice
- Cree el diagrama Entidad relación
- Monte en Workbench
- Llame desde Java

materia	CodMateria	horario	creditos	nombre	apellido	edad	cc
Lenguaje	2	14:00:00	3	Pedro	Perez	20	456
Inglés	4	11:00:00	3	Pedro	Perez	20	456
Lenguaje	2	14:00:00	3	Maria	Gómez	23	789
Ciencias	3	13:00:00	4	Maria	Gómez	23	789
Inglés	4	11:00:00	3	Maria	Gómez	23	789
Lenguaje	2	14:00:00	3	Luisa	Jaramillo	28	852
Ciencias	3	13:00:00	4	Luisa	Jaramillo	28	852
Inglés	4	11:00:00	3	Luisa	Jaramillo	28	852
Matemáticas	1	10:00:00	4	Lina	Giraldo	21	963
Lenguaje	2	14:00:00	3	Lina	Giraldo	21	963
Matemáticas	1	10:00:00	4	Leon	Escobar	30	741
Ciencias	3	13:00:00	4	Leon	Escobar	30	741
Inglés	4	11:00:00	3	Leon	Escobar	30	741

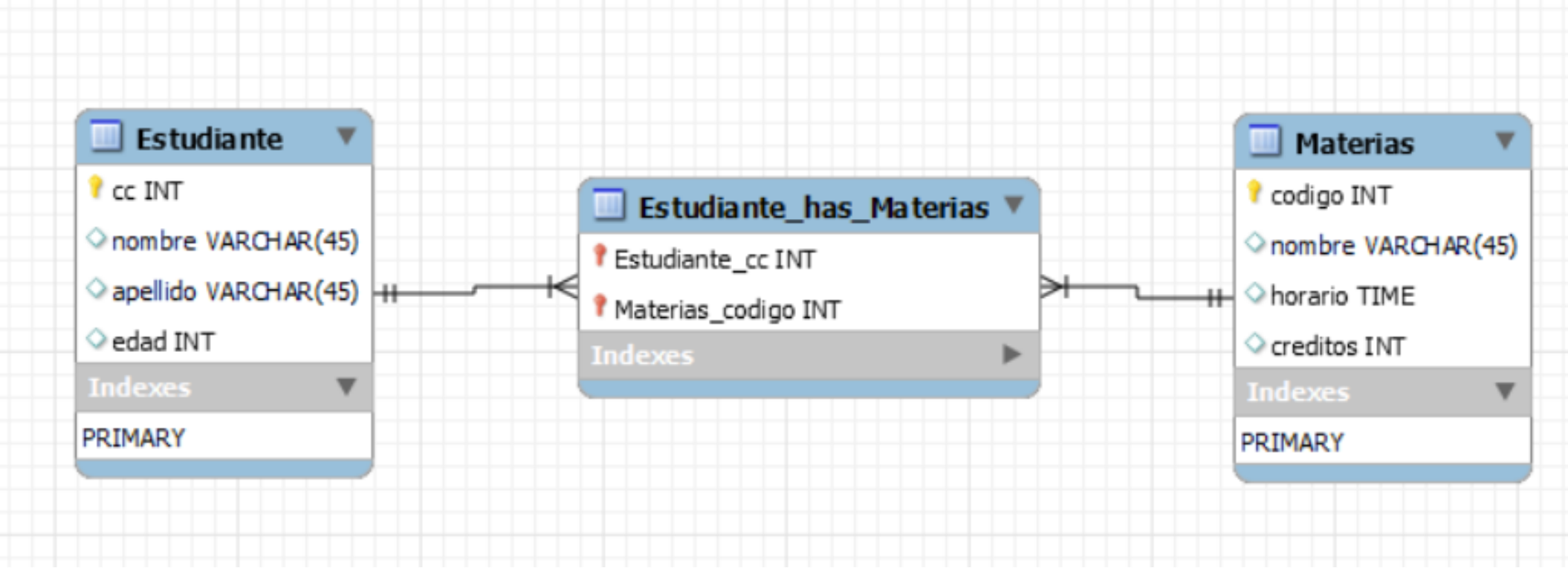
Conexión a base de datos

Estudiante			
cedula	**nombre**	**apellido**	**edad**
123	Juan	Rendón	25
456	Pedro	Perez	20
789	Maria	Gómez	23
963	Lina	Giraldo	21
852	Luisa	Jaramillo	28
741	Leon	Escobar	30

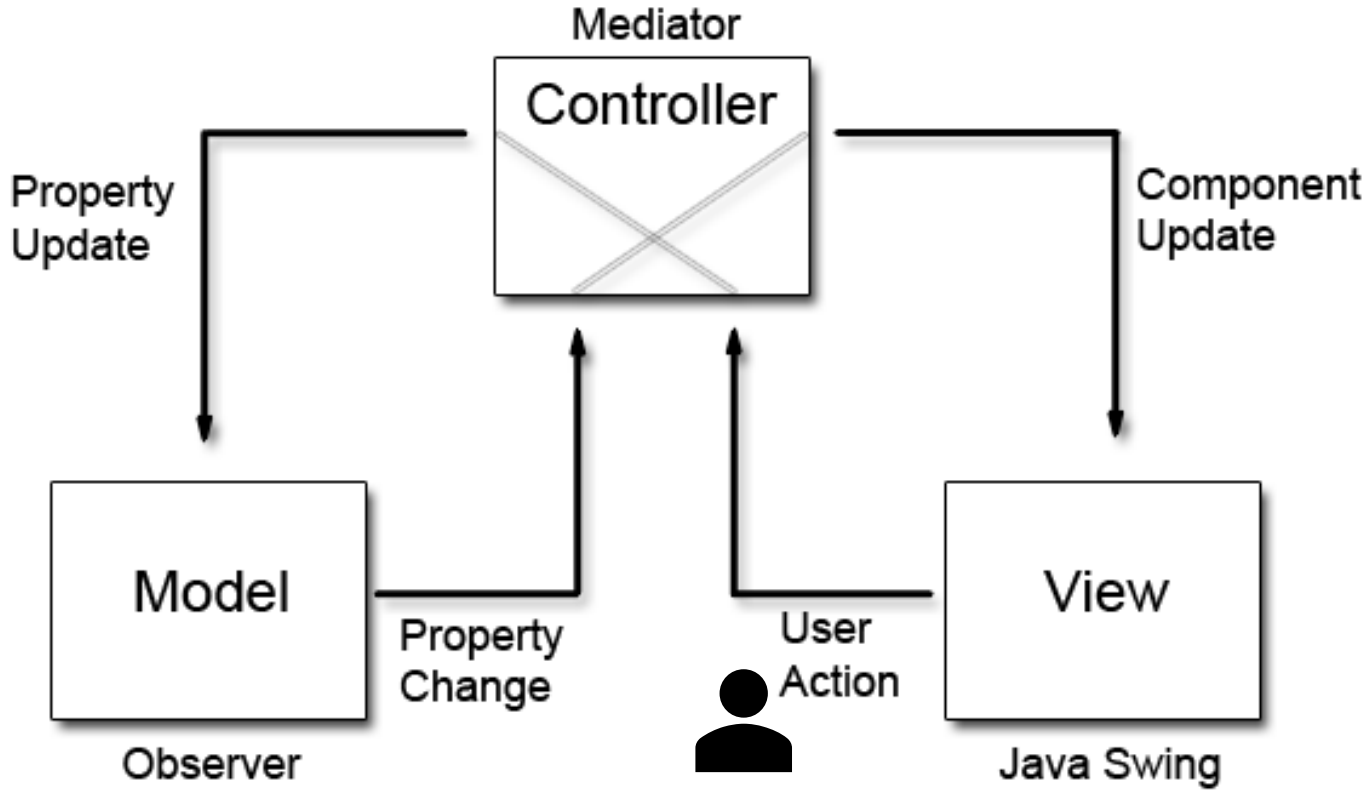
Relacion	
cedulaEst	codMat
123	1
123	2
123	3
456	2
456	4
789	2
789	3
789	4
963	1
963	2
852	3
852	2
852	4
741	3
741	4
741	1

Materias			
codigoC	nombre	horario	creditos
1	Matemáticas	10:00:00	4
2	Lenguaje	14:00:00	3
3	Ciencias	13	4
4	Inglés	11	3

Conexión a base de datos



Modelo – Vista – Controlador



Modelo – Vista – Controlador

Modelo:

El modelo define qué datos debe contener la aplicación. Si el estado de estos datos cambia, el modelo generalmente notificará a la vista (para que la pantalla pueda cambiar según sea necesario) y, a veces, el controlador (si se necesita una lógica diferente para controlar la vista actualizada).

Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio).

Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

Modelo – Vista – Controlador

Vista:

La vista define cómo se deben mostrar los datos de la aplicación.

Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario), por tanto requiere de dicho 'modelo' la información que debe representar como salida.

Modelo – Vista – Controlador

Controlador:

El controlador contiene una lógica que actualiza el modelo y / o vista en respuesta a las entradas de los usuarios de la aplicación.

El Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'

Modelo – Vista – Controlador

Ejemplo:

Cambiar una label al presionar un botón.

Modelo – Vista – Controlador

Ejercicio:

Modelo – Vista – Controlador

Ejercicio 1: Modelo

En el modelo se colocan los atributos y los métodos a ser implementados.

Atributos:

numero1, numero2, resultado

Tener en cuenta los métodos getter y setter.

Modelo – Vista – Controlador

Ejercicio 1: vista

Presentación para el usuario, interfaz de usuario

Modelo – Vista – Controlador

```java
package controlador;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

// Otros archivos
import modelo.modelo;
import vista.*;

public class controlador implements ActionListener{
    private vista vista1;
    private modelo modelo1;

    controlador(){}

    controlador (vista vista1, modelo modelo1){
        this.vista1 = vista1;
        this.modelo1 = modelo1;
        this.vista1.btnCalcularSuma.addActionListener(this);
```

Modelo – Vista – Controlador

```
public void iniciar(){
        this.vista1.setVisible(true);
    }

@Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==vista1.btnCalcularSuma){
            double num1 = Double.parseDouble(vista1.txtSumaNum1.getText());
            double num2 = Double.parseDouble(vista1.txtSumaNum2.getText());

            modelo1.setNumero1(num1);
            modelo1.setNumero2(num2);
            modelo1.sumar();

            String res = String.valueOf(modelo1.getResultado());

            vista1.txtSumaRes.setText(res);
        }
    }
```