BBDD

- **Manejo excepciones**
  - ***try-catch*** //Siempre usarlo cuando se instancie una clase de las de abajo o se use un método suyo
  - ***SQLException***
    - ***getMessage()*** //Devuelve un mensaje indicando por qué se ha lanzado la excepción
- **Connection**
  - ***DriverManager.getConnection("jdbc:mysql://localhost:3306/nombre_bd", "usuario", "contraseña")***
- **Statement**
  - ***miConexion.createStatement()*** //Para instanciarlo
  - ***executeUpdate(String instruccionSQL)*** //Insert, update, delete
  - ***executeQuery(String instruccionSQL)*** //Select
- **PreparedStatement**
  - ***miConexion.prepareStatement(String instruccionSQL)*** //Para instanciarlo. La instrucción llevará una ? para hacer referencia a cada parte que puede cambiar de valor. El primer índice es el 1.
  - ***executeUpdate()*** //Insert, update, delete
  - ***executeQuery()*** //Select
  - ***setInt(int indice, int valor)*** //Añadir un entero a la instruccionSQL
  - ***setString(int indice, String cadena)*** //Añadir una cadena a la instruccionSQL
- **ResultSet**
  - ***miStatement.executeQuery(instruccionSql)*** //Con Statement
  - ***miPreparedStatement.executeQuery()*** //Con PreparedStatement
  - ***next()*** //Indica si hay una línea de la tabla que coincide con la consulta
  - ***while (miResultSet.next()) {}*** //Recorrer el ResultSet
- **Código SQL**
  - ***Insert***: insert into tabla values(valor1, valor2, valor3)
  - ***Update***: update tabla set dato1 = valor1 where dato2 = valor2
  - ***Delete***: delete from tabla where dato1 = valor1
  - ***Select***: select * from tabla where dato1 = valor1

GUI

- **JFrame** //Se extiende
  - ***super(String texto)*** //Constructor con el título de la ventana
  - ***add(Component panel)*** //Añadir el panel
  - ***setBounds(int x, int y, int width, int height)*** //Darle tamaño a la ventana
  - ***setDefaultCloseOperation(EXIT_ON_CLOSE)*** //Declarar el comportamiento de la X
  - ***setVisible(boolean booleano)*** //Hacer visible/invisible la ventana
- **JPanel**
  - ***add(Component componente)*** //Añadir un label/textfield/button
  - ***setVisible(boolean booleano)*** //Hacer visible/invisible el panel
- **JLabel**
  - ***JLabel()*** //Constructor por defecto
  - ***JLabel(String texto)*** //Constructor con texto
  - ***setText(String texto)*** //Modificar el texto
  - ***getText()*** //Obtener el texto
- **JTextField**
  - ***JTextField(int columnas)*** //Constructor con nº de columnas definidas
  - ***setText(String texto)*** //Modificar el texto
  - ***getText()*** //Obtener el texto
- **JButton**
  - ***JButton(String texto)*** //Constructor por defecto

- ***JButton(String texto)*** //Constructor con texto
- ***setText(String texto)*** //Modificar el texto
- ***getText()*** //Obtener el texto
- ***addActionListener(this)*** //Añadir un ActionListener, el this indica que la respuesta al evento está en la propia clase
- ***setEnabled(boolean booleano)*** //Activar/desactivar botón

- **ActionListener** //Interfaz que se implementa
  - ***ActionEvent***
    - ***getSource()*** //Obtener la dirección de memoria del objeto desencadenante del evento
- Cuando hay dos ventanas, instanciarlas en el main y la contraria a la clase como atributo. Añadir un setter del atributo y llamarlo en el main.

COLECCIONES

- **ArrayList<E>** //Array dinámico indexado
  - ***add(E elemento)*** //Añadir un elemento, si ya existe no se añade
  - ***add(int pos, E elemento)*** //Añadir un elemento en un posición concreta, si ya existe no se añade
  - ***clear()*** //Elimina todos los elementos del ArrayList
  - ***contains(E elemento)*** //Devuelve verdadero si el elemento existe
  - ***get(int pos)*** //Devuelve un elemento por índice
  - ***indexOf(E elemento)*** //Devuelve la posición en la que se encuentra la primera ocurrencia del elemento
  - ***isEmpty()*** //Devuelve verdadero si el ArrayList está vacío
  - ***lastIndexOf(E elemento)*** //Devuelve la posición en la que se encuentra la última ocurrencia del elemento
  - ***remove(int pos)*** //Eliminar un elemento por índice, si no existe no se elimina
  - ***remove(E elemento)*** //Eliminar la primera ocurrencia de un elemento, si no existe no se elimina
  - ***set(int pos, E elemento)*** //Reemplaza el elemento en la posición indicada por el nuevo elemento
  - ***size()*** //Devuelve el tamaño del ArrayList
- **HashSet<E>** //Lista de valores que no se pueden repetir
  - ***add(E elemento)*** //Añadir un elemento, si ya existe no se añade. Admite nulos
  - ***clear()*** //Elimina todos los elementos del HashSet
  - ***contains(E elemento)*** //Devuelve verdadero si el elemento existe
  - ***isEmpty()*** //Devuelve verdadero si el HashSet está vacío
  - ***remove(E elemento)*** //Eliminar un elemento, si no existe no se elimina
  - ***size()*** //Devuelve el tamaño del HashSet
- **HashMap<K, V>** //Lista de pares clave-valor y la clave no se puede repetir
  - ***containsKey(K clave)*** //Devuelve verdadero si la clave existe
  - ***get(K clave)*** //Devuelve el valor correspondiente a la clave
  - ***isEmpty()*** //Devuelve verdadero si el HashMap está vacío
  - ***keySet()*** //Devuelve un Set con las claves del HashMap
  - ***put(K clave, V valor)*** //Añadir un par clave-valor, si ya existe la clave no se añade el par
  - ***remove(K clave)*** //Eliminar un elemento por clave, si no existe la clave no se elimina
  - ***size()*** //Devuelve el tamaño del HashMap
- Para recorrer cada colección se usa el foreach, para el ArrayList también el for normal.
- Para los HashMap se usan las claves y se recorre el keySet().