TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE OAXACA

ASIGNATURA: TOPICOS AVANZADOS DE PROGRAMACIÓN

CATEDRÁTICO: HERNANDEZ ABREGO ANAYANSI CRISTINA

ALUMNO: GARCÍA GARCÍA JOSÉ ÁNGEL

UNIDAD: 4

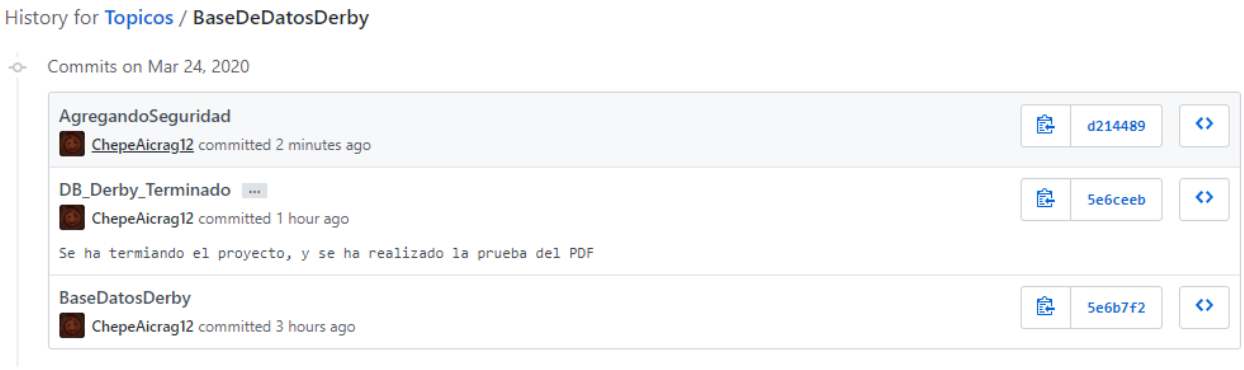
PRACTICA 4.2- Conexión a base de datos con Derby

OAXACA DE JÚAREZ, OAX, 24/MARZO/2020

Índice

BITACORA

Se utilizó como bitácora a GitHub. Se realizo en tres commits el proyecto, lo primero fue crearlo y después de ello, realizar las pruebas que vienen en el PDF y por último, agregando la seguridad.




```
/**
 * El método tiene la funcion de
 * incrementar el numero de conexiones solicitadas y ademas
 * verificar que se cree una unica instancia de Conexion;
 * si ya existe, simplemente la retorna
 * Tiene que tener la firma de static porque no requiere de una instancia ya
 * creada para ejecutarlo, debido a que primero ejecutamos el método y despues
 * su constructor.
 *
 */
public static Conexion getConexion(String url, String Usuario, String password){
```

```
/**
 * El método tiene la función de cerrar la conexión
 * para ello comprueba que la variable coneccion sea una instancia,
 * es decir, que sea !null y posterior a ello comprueba nuevamente
 * que el numero de conexiones sea 1 para cerrarla y retornar true;
 * si no es igual a 1, simplemente decrementa el valor y no cierra
 * conexion porque como mencionamos al principio, solo trabajamos con
 * una unica conexion, por lo que esta solo se cerrara cuando se tenga
 * una conexion. Si la conexion es null simplemente retorna un false y si
 * hay problemas en tiempo de ejecucion al momento de cerrar
 * la conexion las controla con el bloque try-catch y muestra el error
 *
 *
 */
public boolean CerrarConexion(){
```

CÓDIGO

```
package practica_08;
```

CLASE CONEXIÓN

```
import com.sun.istack.internal.logging.Logger;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;

/**
 *
 * @author Garcia Garcia Jose Angel
 */
public class Conexion {
    private static Connection coneccion = null; // Contenida en el pquete SQL
    private static Conexion conexion = null; // instancia a utilizar
    private static  int numConexiones = 0; // controla el numero de veces que sucedio

    /**
     * Constructor privado porque
     * Se evitará crear mas de una instancia de la clase
     * es decir, controlaremos la creacion de instancias
     * si se intenta crear mas de una, despues de la primera
     * se retornará la mismada crada al principio
     */

    private Conexion(String url, String usuario, String password){
        try {
            // Clase usada para una conexion con Derby
            Class.forName("org.apache.derby.jdbc.ClientDriver");
```

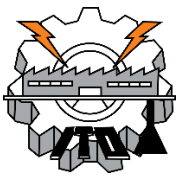
```
        // Para MySql : "com.mysql.jdbc.Driver"
        // Para Postgres : "org.postgresql.Driver"
        coneccion = DriverManager.getConnection(url,usuario, password);
    } catch (SQLException e) {
        java.util.logging.Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE,null,e);
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE, null, ex);
    }
}
/**
 * El método tiene la funcion de
 * incrementar el numero de conexiones solicitadas y ademas
 * verificar que se cree una unica instancia de Conexion;
 * si ya existe, simplemente la retorna
 * Tiene que tener la firma de static porque no requiere de una instancia ya
 * creada para ejecutarlo, debido a que primero ejecutamos el método y despues
 * su constructor.
 *
 */
 public static Conexion getConexion(String url, String Usuario, String password){
    numConexiones++;
    if(conexion == null)
        return conexion = new Conexion(url, Usuario, password);
    return conexion;
 }

 public static Connection getConeccion(){
     return coneccion;
 }
```

```
    /**
     * El método tiene la función de cerrar la conexión
     * para ello comprueba que la variable coneccion sea una instancia,
     * es decir, que sea !null y posterior a ello comprueba nuevamente
     * que el numero de conexiones sea 1 para cerrarla y retornar true;
     * si no es igual a 1, simplemente decrementa el valor y no cierra
     * conexion porque como mencionamos al principio, solo trabajamos con
     * una unica conexion, por lo que esta solo se cerrara cuando se tenga
     * una conexion. Si la conexion es null simplemente retorna un false y si
     * hay problemas en tiempo de ejecucion al momento de cerrar
     * la conexion las controla con el bloque try-catch y muestra el error
     *
     *
     */
    public boolean CerrarConexion(){
        try {
            if (coneccion != null)
              if(numConexiones == 1){
                coneccion.close();
                return true;
            }else
                numConexiones--;
            return false;
        } catch (SQLException e) {
            System.err.println(" Error al tratar de cerrar la conexion " + e);
          }
         return false;
    }
}
```

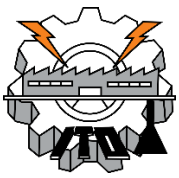
CLASE MANEJO DATOS

```
package practica_08;

import java.io.File;
import java.io.FileReader;
import java.sql.Connection;
import java.sql.Date;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author Garcia Garcia Jose Angel
 */
public class ManejoDatos {
    private Connection conexion ; // Acceso a conexion
    private Conexion crearConexion;// Crea conexion
    private final int CAMPOS_PERSONA = 4;
    private final int CAMPOS_ACTIVIDAD = 3;

    public ManejoDatos(){
        try{
        String usuario = "",contraseña = "";
        In leer = new In("contraseña.txt");
        String[] line = leer.readLine().split(" ");
        usuario = line[0];
        contraseña = line[1];
```

```
        crearConexion =
crearConexion.getConexion("jdbc:derby://localhost:1527/mediciones_personas",usuario,contraseña);
        conexion = crearConexion.getConeccion();
        }catch(Exception e){
        }
    }

    public List <Object[]> conexionConsultaPersona(String sql){
        // Regresa los registros de las personas en una lista
        List <Object []> datos = new ArrayList<Object[]>();
        DateFormat fecha = new SimpleDateFormat("yyyy-MM-dd");
        try {
            Statement ps = conexion.createStatement();
            ResultSet rs = ps.executeQuery(sql);
            while (rs.next()) {
                // Estructura del registro de persona pasado como cadena
                String[] dat = new String[CAMPOS_PERSONA];
                dat[0] = String.valueOf((Integer) rs.getInt(1));
                dat[1] = rs.getString(2);
                dat[2] = fecha.format((Date) rs.getDate(3));
                dat[3] = rs.getString(4);
                datos.add(dat);
            }
        } catch (Exception e) {
            System.err.println("Error al consultar personas " + e);
        }
        return datos;
    }

    public List <Object[]> conexionConsultaActividad(String sql){
        // Regresa los registros de actividad
```



```
        List <Object[]> datos = new ArrayList<Object[]>();
        try {
           Statement ps = conexion.createStatement();
           ResultSet rs = ps.executeQuery(sql);
            while (rs.next()) {
                // Estructura del registro activiad
                String [] dat = new String[CAMPOS_ACTIVIDAD];
                dat[0] = String.valueOf(rs.getInt(1));
                dat[1] = rs.getString(2);
                dat[2] = rs.getString(3);
                datos.add(dat);
           }
        } catch (Exception e) {
            System.err.println("Error al consultar actividades " + e);
        }
        return datos;
    }
public List <Object[]> conexionConsultaMediciones(String sql){
        // Regresa los registros de actividad
        List <Object[]> datos = new ArrayList<Object[]>();
        DateFormat fecha = new SimpleDateFormat("yyyy-MM-dd");
        try {
           Statement ps = conexion.createStatement();
           ResultSet rs = ps.executeQuery(sql);
            while (rs.next()) {
                // Estructura del registro activiad
                String [] dat = new String[CAMPOS_MEDICIONES];
                dat[0] = String.valueOf(rs.getInt(1));
                dat[1] = fecha.format((Date) rs.getDate(2));
                dat[2] = String.valueOf((Integer) rs.getInt(3));
```

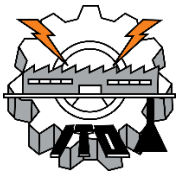


```
                dat[3] =  String.valueOf((Integer) rs.getInt(4));
                dat[4] = String.valueOf((Integer) rs.getInt(5));
                dat[5] = String.valueOf((Integer) rs.getInt(6));
                dat[6] = String.valueOf((Integer) rs.getInt(7));
                dat[7] = String.valueOf((Integer) rs.getInt(8));
                datos.add(dat);
            }
        } catch (Exception e) {
            System.err.println("Error al consultar mediciones " + e);
        }
        return datos;
    }
}
```

CLASE PRUEBA ACCESO DATOS

```
package practica_08;
import java.util.ArrayList;
/**
 *
 * @author Garcia Garcia Jose Angel
 */
public class PruebaAccesoDatos {
    public static void main(String[] args) {
        ManejoDatos baseDatos = new ManejoDatos();
        System.out.println("----------CONSULTA DE ACTIVIADES----------");
        consulta_actividades(baseDatos);
        System.out.println("\n\n----------CONSULTA DE PERSONAS----------");
        consulta_personas(baseDatos);
        System.out.println("\n\n----------CONSULTA DE MEDICIONES----------");
        consulta_mediciones(baseDatos);
    }
```

```
    public static void consulta_personas(ManejoDatos baseDatos){
        String consulta = "SELECT * FROM chepe.PERSONA";
        ArrayList<Object[]> actividad = (ArrayList<Object[]>)
baseDatos.conexionConsultaActividad(consulta);
        for (int i = 0; i < actividad.size(); i++) {
            Object [] reg = actividad.get(i);
            System.out.println("");
            for (int j = 0; j < reg.length; j++) {
                if(j > 0)
                System.out.printf("%-30s",reg[j]);
                else
                System.out.printf("%-10s",reg[j]);
            }
        }
    }
    public static void consulta_actividades(ManejoDatos baseDatos){
        String consulta = "SELECT * FROM chepe.TIPOACTIVIDAD";
        ArrayList<Object[]> actividad = (ArrayList<Object[]>)
baseDatos.conexionConsultaActividad(consulta);
        for (int i = 0; i < actividad.size(); i++) {
            Object [] reg = actividad.get(i);
            System.out.println("");
            for (int j = 0; j < reg.length; j++) {
                System.out.printf("%-25s",reg[j]);
            }
        }
    }
    public static void consulta_mediciones(ManejoDatos baseDatos){
        String consulta = "SELECT * FROM chepe.MEDICIONES";
```



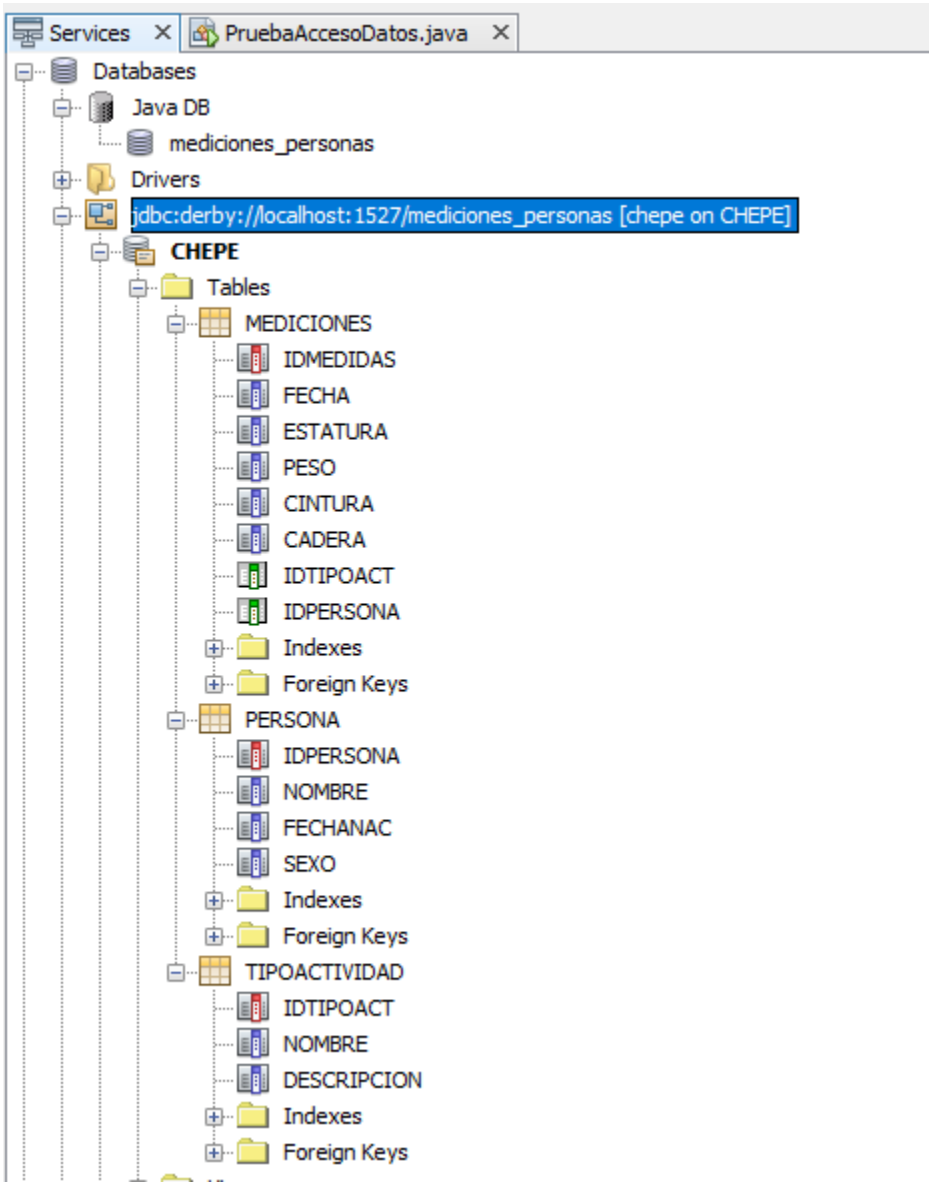
```
        ArrayList<Object[]> actividad = (ArrayList<Object[]>)
baseDatos.conexionConsultaMediciones(consulta);
        for (int i = 0; i < actividad.size(); i++) {
            Object [] reg = actividad.get(i);
            System.out.println("");
            for (int j = 0; j < reg.length; j++) {
                System.out.printf("%-20s",reg[j]);
            }
        }
    }
}
```

La clase In que aparece es una clase externa, que la puede encontrar en el siguiente enlace

https://introcs.cs.princeton.edu/java/stdlib/In.java.html

PRUEBAS

Prueba de que se haya creado la Base de Datos correctamente.



Y también la prueba de que se ejecuta de forma correcta.

```
Tue Mar 24 17:20:30 CST 2020 : Se ha instalado el gestor de seguridad con la política de seguridad de servidor básica.
Tue Mar 24 17:20:37 CST 2020 : Servidor de red Apache Derby: Se ha iniciado 10.11.1.2 - (1629631) y está listo para aceptar las conexiones en el puerto 1527
```

Procederemos a ver los datos que contiene cada tabla de nuestra base de datos.

Tabla Persona

SELECT * FROM CHEPE.PERSO... ×

Page Size: 20 Total Rows: 5 Page: 1 of 1 Matching R

#	IDPERSONA	NOMBRE	FECHANAC	SEXO
1	1	Miguel Angel Garcia	2020-01-01	H
2	2	Jose Angel Garcia	2020-01-01	H
3	3	Maria De Los Angeles Garcia	2020-01-01	M
4	4	Antonio Garcia	2019-06-13	H
5	5	Reyna Garcia	2019-08-13	M

Tabla Actividades

SELECT * FROM CHEPE.TIPOA... ×

Page Size: 20 Total Rows: 4 Page: 1 of 1

#	IDTIPOACT	NOMBRE	DESCRIPCION
1	1	SEDENTARIO	No hacer practicamente nada de ejercicio
2	2	LIGERAMENTE ACTIVAS	Hacer ejercicio suave de 1 a 3 veces por semana
3	3	MODERADAMENTE ACTIVA	Hacer deporte 3 a 5 veces por semana
4	4	MUY ACTIVAS	Hacer deporte 6 a 7 días por semana

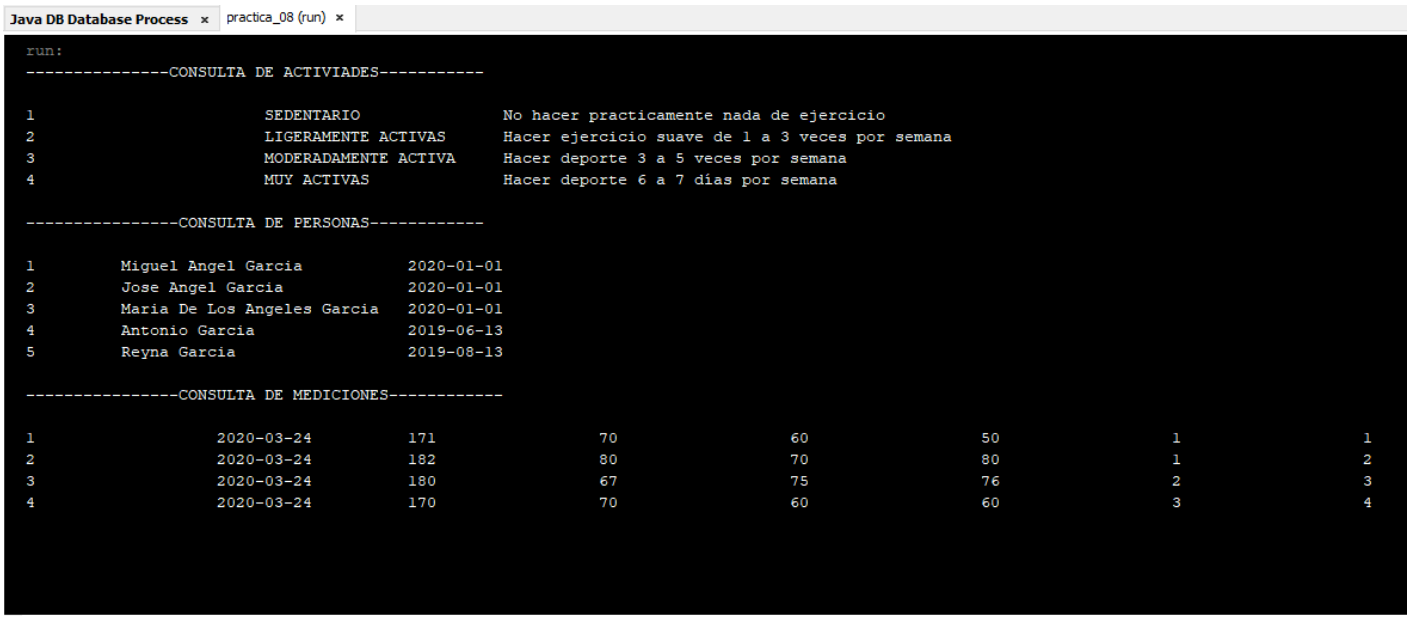
Tabla de Mediciones

SELECT * FROM CHEPE.MEDIC... ×

Page Size: 20 Total Rows: 4 Page: 1 of 1 Matching Rows:

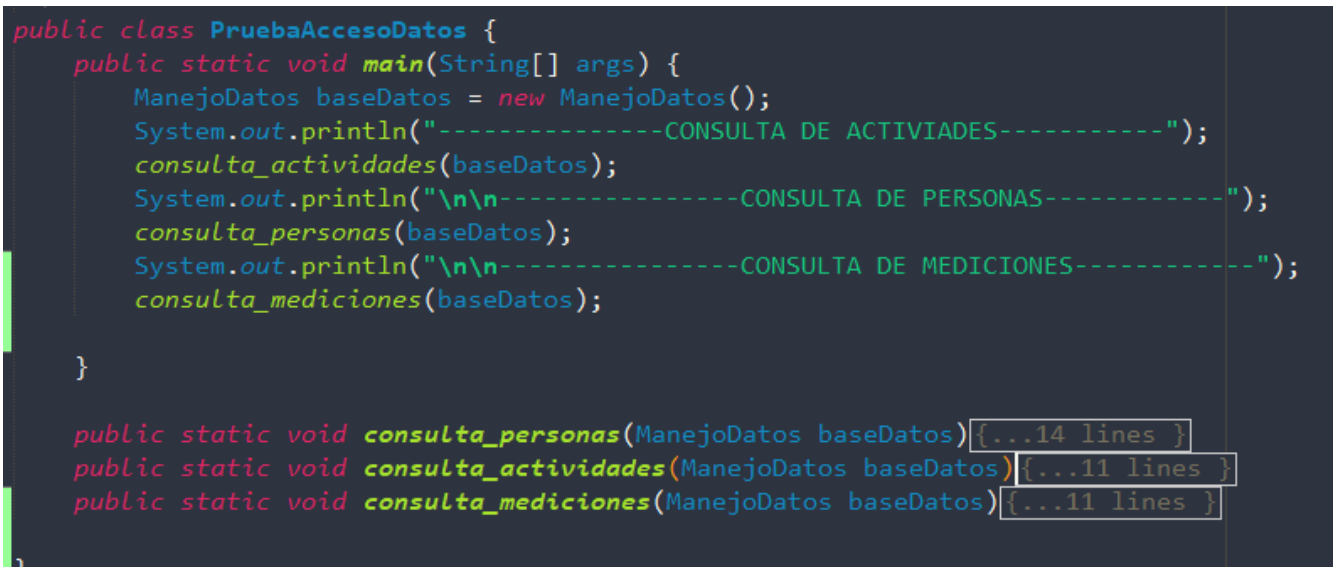
#	IDMEDIDAS	FECHA	ESTATURA	PESO	CINTURA	CADERA	IDTIPOACT	IDPERSONA
1	1	2020-03-24	171	70.0	60	50	1	1
2	2	2020-03-24	182	80.0	70	80	1	2
3	3	2020-03-24	180	67.0	75	76	2	3
4	4	2020-03-24	170	70.0	60	60	3	4

Prueba de la salida de las consultas.



```
Java DB Database Process × | practica_08 (run) ×

run:
---------------CONSULTA DE ACTIVIADES-----------

1                       SEDENTARIO              No hacer practicamente nada de ejercicio
2                       LIGERAMENTE ACTIVAS     Hacer ejercicio suave de 1 a 3 veces por semana
3                       MODERADAMENTE ACTIVA    Hacer deporte 3 a 5 veces por semana
4                       MUY ACTIVAS             Hacer deporte 6 a 7 días por semana

----------------CONSULTA DE PERSONAS------------

1       Miguel Angel Garcia             2020-01-01
2       Jose Angel Garcia               2020-01-01
3       Maria De Los Angeles Garcia     2020-01-01
4       Antonio Garcia                  2019-06-13
5       Reyna Garcia                    2019-08-13

----------------CONSULTA DE MEDICIONES------------

1               2020-03-24      171             70              60              50              1               1
2               2020-03-24      182             80              70              80              1               2
3               2020-03-24      180             67              75              76              2               3
4               2020-03-24      170             70              60              60              3               4
```

Prueba de que la clase **PruebaAccesoDatos** está consultando a la misma base de datos. Lo que realice fue la creación de dos métodos que realizan la impresión de la consulta independientemente. Agregue la consulta de tabla Mediciones.

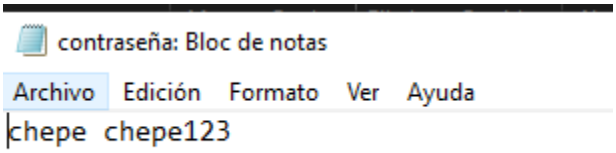


```
public class PruebaAccesoDatos {
    public static void main(String[] args) {
        ManejoDatos baseDatos = new ManejoDatos();
        System.out.println("---------------CONSULTA DE ACTIVIADES-----------");
        consulta_actividades(baseDatos);
        System.out.println("\n\n----------------CONSULTA DE PERSONAS------------");
        consulta_personas(baseDatos);
        System.out.println("\n\n----------------CONSULTA DE MEDICIONES------------");
        consulta_mediciones(baseDatos);

    }

    public static void consulta_personas(ManejoDatos baseDatos){...14 lines }
    public static void consulta_actividades(ManejoDatos baseDatos){...11 lines }
    public static void consulta_mediciones(ManejoDatos baseDatos){...11 lines }

}
```

La parte de seguridad que implemente fue guardar la contraseña en un archivo. Entonces la parte del código configurada es:

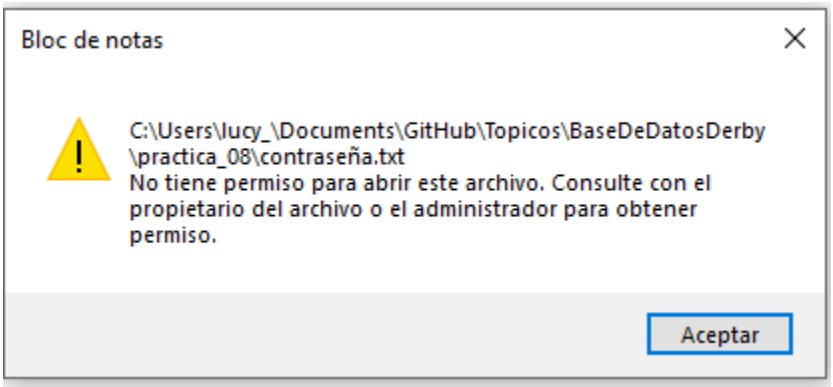
```
public ManejoDatos(){
    try{
    String usuario = "",contraseña = "";
    In leer = new In("contraseña.txt");
    String[] line = leer.readLine().split(" ");
    usuario = line[0];
    contraseña = line[1];
    crearConexion = crearConexion.getConexion("jdbc:derby://localhost:1527/mediciones_personas",usuario,contraseña);
    conexion = crearConexion.getConeccion();
    }catch(Exception e){
    }
}
```

Lo que realiza es la lectura de lo que contenga el archivo. Este archivo contiene lo siguiente:



El primero corresponde al usuario y lo segundo a la contraseña, estos deben estar separados por un espacio en blanco. La seguridad inicia cuando encriptamos el archivo o cambiamos los permisos de este, para que solo ciertos clientes tengan acceso a él o inclusive el único que creo el archivo, en fin, existen diferentes formas de poner seguridad a un archivo para que no pueda ser abierto.

Un ejemplo de cuando aseguramos nuestro archivo es poniendo permisos para abrir, entonces se nos muestra un mensaje como el de la imagen.



Pero algo importante es asegurarnos que desde donde ejecutamos el proyecto se encuentre entre los Privilegios para poder acceder a él, de lo contrario tendríamos problemas. Lo recomendable es tener el privilegio para un grupo de usuarios.