```
/**

    CODIGO JAVA

**/


import java.io.*;
import java.util.*;
import java.sql.*;

/**
*    Clase ProyectoBDJava2015 Implementa los metodos de Insercion, Eliminacion y Consultas
*    para la utilizacion de la base de datos del juego Cuatro En Linea.
*    Integrantes : Mauricio Delle Vedove, Francisco Roig, Matias Rondeau.
**/


public class ProyectoBDJava2015 {

    public ProyectoBDJava2015() {
        try {
            jbInit();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); // Buffer usado
para leer lo ingresado por teclado.
            String driver = "com.mysql.jdbc.Driver";
            String url = "jdbc:mysql://localhost/Proyecto";
            System.out.print("Nombre de Usuario: "); //root en casos generales
            String username = br.readLine();
            System.out.print("Clave: "); //root en caso generales
            String password = br.readLine();
            Class.forName(driver);  // Load database driver if not already loaded.
            Connection connection = DriverManager.getConnection(url, username, password); // Establish
network connection to database.
            Statement statement = connection.createStatement();
            char opcion;
            do {         //Eleccion del menu
                opcion = menu();
                switch (opcion) {
                    case 'a':
                        insertar(connection);
                        break;
                    case 'b' :
                        eliminar(statement);
                        break;
                    case 'c':
                        listar_partidas_ganadas(connection);
                        break;
                    case 'd' :
                        listar_partidas_mas_largas(connection);
                        break;
                    case 'e' :
                        listar_partida_de_jugador(statement);
                        break;
                    case 'f' :
                        listar(statement);
                        break;
                    case 'g' :
                        listar_partida_incompleta(connection);
                        break;
                    case 'h' :
                        jugadores_nocturnos(connection);
                        break;
                    case ' ' :
                        System.out.print("\n*** Debe ingresar un valor de los anteriores nombrados!!!
***\n");
```

```
                    break;
                case 's' :
                    break;
                default :
                    System.out.println("\n*** Opcion incorrecta!!!. Intente nuevamente. ***");
                    break;
            }
        } while (opcion != 's');
    }
    catch(ClassNotFoundException cnfe) { //Error si no utiliza java -cp .:mysql-connector-
java-5.1.33 java ProyectoBDJava2015.
        System.err.println("Error loading driver: " + cnfe);
    }
    catch(SQLException sqle) {
        System.err.println("Error connecting: " + sqle); //Cuando SQL Server returna un "warning"
o error.
    }
    catch (IOException ioe) {
        System.err.println("Error I/O"); //Error de Entrada/Salida.
    }
}

private void jbInit() throws Exception {
}

/**
*   Elimina un jugador de la base de datos del sistema.
*
*
**/

private static void eliminar(Statement statement) {
    try {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("\n- Ingrese el nick: ");
        String nick = br.readLine();
        String query = "DELETE FROM jugador WHERE nick='"+nick+"'";
        statement.executeUpdate(query);
    }
    catch (IOException ioe) {
        System.err.println("Error I/O");
    }
    catch (Exception e) {
        System.err.println(e);
    }
}

/**
*   Inserta un jugador dentro de la base de datos del sistema.
*
*
**/
private static void insertar(Connection connection) {
    try {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("\n- Ingrese el Nick: ");
        String nick = br.readLine();
        System.out.print("\n- Ingrese el email: ");
        String email = br.readLine();
        System.out.print("\n- Ingrese el Nombre y apellido: ");
        String nomAp = br.readLine();
        System.out.print("\n- Ingrese la Fecha de Nacimiento (aaaa-mm-dd): ");
        String bornDate = br.readLine();
        System.out.print("\n- Ingrese la edad: ");
        String age = br.readLine();
        String query = "insert into jugador (nick,email,nombreAPellido,fechaNac,edad) values
(?,?,?,?,?)";
        PreparedStatement preparedStmt = connection.prepareStatement(query);
        preparedStmt.setString (1, nick);
        preparedStmt.setString (2, email);
        preparedStmt.setString (3, nomAp);
        preparedStmt.setString (4, bornDate);
        preparedStmt.setString (5, age);
```

```
            preparedStmt.execute();

        }
        catch (IOException ioe) {
            System.err.println("Error I/O");
        }
        catch (Exception e) {
            System.err.println(e);
        }
    }

    /**
    *   Lista los jugadores del sistema con sus datos de miembro.
    *
    *
    **/


    private static void listar(Statement statement) {
        try {
            String query = "SELECT * FROM jugador";
            ResultSet resultSet1 = statement.executeQuery(query);
            while(resultSet1.next()) {
                System.out.print("\nID: " + resultSet1.getString(1));
                System.out.print("; Nick: " + resultSet1.getString(2));
                System.out.print("; Email: " + resultSet1.getString(3));
                System.out.print("; Nombre y Apellido: " + resultSet1.getString(4));
                System.out.print("; Fecha de Nacimiento: " + resultSet1.getString(5));
                System.out.print("; Edad: " + resultSet1.getString(6));
                System.out.print("\n");

            }
            resultSet1.close();
        }

        catch (Exception e) {
            System.err.println(e);
        }
    }

    /**
    *   Lista la partida de un jugador ingresado (El codigo de partida, la fecha y el
contrincante)
    *
    *
    **/

    private static void listar_partida_de_jugador(Statement statement) {
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("\n- Ingrese el Nick: ");
            String nick = br.readLine();
            String query = "SELECT idpartida,fecha,id2,id,id1,nick FROM partida NATURAL JOIN jugador
WHERE (id=id1 OR id=id2) AND (nick='"+nick+"')";
            ResultSet resultSet1 = statement.executeQuery(query);
            int i=0;
                while(resultSet1.next()) {
                    System.out.print("\nCodigo de partida: " + resultSet1.getString(1));
                    System.out.print("; Fecha: " + resultSet1.getString(2));
                    System.out.print("; Contrincante: " + resultSet1.getString(3));
                    System.out.print("\n");
                    i++;
                }
                if (i==0){
                    System.out.print("\n -"+nick+", No tiene partidas \n");
                }
                resultSet1.close();
            }

        catch (Exception e) {
            System.err.println(e);
        }
    }
```

```java
    /**
    *   Punto 6 Primera Consulta Compuesta:
    *   Seleccionar el o los jugadores (nicks) que aun no han finalizado alguna partida.
    *
    *
    **/


    private static void listar_partida_incompleta (Connection connection){
        try {

            Statement stat1 = connection.createStatement();
            String query = "SELECT DISTINCT nick FROM jugador NATURAL JOIN partida WHERE"
                           +"(id=id1 AND id IN(SELECT id1 FROM partida WHERE estado=false))"
                           +"OR id=id2 AND id IN (SELECT id2 FROM partida WHERE estado=false); ";
            ResultSet resultSet1= stat1.executeQuery(query);
            System.out.print("\n Jugador/es con partida/s incompleta/s : \n");
            int i= 0;
            while (resultSet1.next()){
                System.out.print("\n -"+resultSet1.getString("nick")+"\n");
                i++;
                }
            if (i==0){
                System.out.print("\n - Todas las partidas estan finalizadas. \n");
            }
            resultSet1.close();
        }
        catch (Exception e){
        System.err.println(e);
        }
    }



    /**
    *
    *    Lista los jugadores que han jugador alguna vez luego de las 23:59
    *    Los Jugadores Nocturnos.
    **/
    private static void jugadores_nocturnos (Connection connection){
        try {

            Statement stat1 = connection.createStatement();
            String query =  "select distinct nick from jugador where"
                           +"(id in (select id1 from partida where horaInicio>'23:59') "
                           +"or id in (select id2 from partida where horaInicio>'23:59')) group by
nick asc; ";
            ResultSet resultSet1= stat1.executeQuery(query);
            System.out.print("\n Jugadores Nocturnos : \n");
            int i= 0;
            while (resultSet1.next()){
                System.out.print("\n -"+resultSet1.getString("nick")+"\n");
                i++;
                }
            if (i==0){
                System.out.print("\n - No hay jugadores nocturnos  \n");
            }
            resultSet1.close();
        }
        catch (Exception e){
        System.err.println(e);
        }
    }




    /**
    *   Lista las partidas ganadas de un jugador de la base de datos
    *
    *
    **/
```

```
    private static void listar_partidas_ganadas(Connection connection) {
        try {
            Statement stat1 = connection.createStatement();
            Statement stat2 = connection.createStatement();
            String query = "SELECT nick FROM jugador";
            ResultSet resultSet1 = stat1.executeQuery(query);
            while(resultSet1.next()) {
                String nick = resultSet1.getString(1);
                String query1 = "SELECT COUNT(nick) FROM partida NATURAL JOIN jugador WHERE (id=id1)
AND resultado='Ganador' AND nick='"+nick+"'";
                ResultSet resultSet2 = stat2.executeQuery(query1);
                while(resultSet2.next()){
                    System.out.print("\n- "+nick+" tiene "+resultSet2.getString(1)+" partidas
ganadas");
                }
                resultSet2.close();
            }
            resultSet1.close();
        }
        catch (Exception e) {
            System.err.println(e);
        }
    }
    /**
    *   Lista los jugadores con su partida que mas duracion tuvo en encuentro.
    *
    *
    **/


    private static void listar_partidas_mas_largas(Connection connection) {
        try {
            Statement stat1 = connection.createStatement();
            Statement stat2 = connection.createStatement();
            String query = "SELECT DISTINCT nick FROM partida NATURAL JOIN jugador WHERE id=id1";
            ResultSet resultSet1 = stat1.executeQuery(query);
            while(resultSet1.next()) {
                String nick = resultSet1.getString(1);
                String query1 = "SELECT MAX(TIMEDIFF(horaFin,HoraInicio)) FROM partida NATURAL JOIN
jugador WHERE (id=id1) AND nick='"+nick+"'";
                ResultSet resultSet2 = stat2.executeQuery(query1);
                while(resultSet2.next()){
                    System.out.println("- "+nick+" su partida mas larga fue de "+resultSet2.getString
(1)+"");
                }
                resultSet2.close();
            }
            resultSet1.close();
        }
        catch (Exception e) {
            System.err.println(e);
        }
    }

    /**
     * Muestra por pantalla el menu y lee por teclado la opcion ingresada por el usuario.
     * @throws IOException cuando el metodo readLine() falla.
     * @pre. true.
     * @post. Muestra por pantalla el menu y lee por teclado la opcion ingresada por el usuario.
     */
    private static char menu() throws IOException {
        char op = ' ';
        BufferedReader b = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("\n \t -----------------------------------------");
        System.out.println("\t|*************** Menu ****************       |");
        System.out.println("\t -----------------------------------------");
        System.out.println("\t| - a: Insertar Jugador                     |");
        System.out.println("\t| - b: Eliminar Jugador                     |");
        System.out.println("\t| - c: Partidas ganadas por jugador         |");
        System.out.println("\t| - d: Partidas mas largas por jugador      |");
        System.out.println("\t| - e: Listar partidas por jugador          |");
        System.out.println("\t| - f: Listar Jugador/es                    |");
```

```
        System.out.println("\t| - g: Jugadores con partidas incompletas     |");
        System.out.println("\t| - h: Jugadores Nocturnos                     |");
        System.out.println("\t| - s: Salir                                   |");
        System.out.println("\t ---------------------------------------------- ");
        System.out.print("\t   Seleccione la opcion deseada: ");

        try {
            op = Character.toLowerCase((b.readLine()).charAt(0)); // Toma el 1er caracter del string
ingresado
        }                                                         // por el usuario (como minuscula).
        catch (IOException ioe) {
            throw new IOException("Error I/O");
        }
        return op;
    }
}
```