Comenzado el	domingo, 21 de abril de 2019, 12:58
Estado	Finalizado
Finalizado en	lunes, 29 de abril de 2019, 12:10
Tiempo empleado	7 días 23 horas
Puntos	0,00/6,00
Calificación	0,00 de 10,00 (0%)

Información

No os guiéis por las correcciones porque lo único que hacen es comparar vuestro código con una posible solución.

Desarrollar y ejecutad el programa en vuestro IDE y comparad la solución con la planteada.

Se usará el usuario desvan con la contraseña desvan. Recordad configurar la IP de BBDD con la que estáis trabajando.

Todas las tablas que se usan son de la tarea de aprendizaje 2 (Desvan) de BD05.

Para cualquier duda ya sabéis donde estoy.

Pregunta **1**

Incorrecta

Puntúa 0,00 sobre 1,00

Crea el método **empleadosRetencionMayor** dentro de la clase BBDD . Recibe un entero con el valor de la retención y devuelve un String con el número de cuenta, el nombre y la retención de los empleados cuya retención es mayor o igual a la indicada.

Para ello, se usará el usuario desvan con la contraseña desvan. Recuerda configurar correctamente la dirección IP de la BBDD.

La **consulta** que debemos realizar es:

```
SELECT cuenta, nombre, retencion FROM empleados WHERE retencion >= XX
```

donde retencion será un número entero y nombre y cuenta cadenas de caracteres

El formato del String devuelto por cada empleado será:

```
nombre: cuenta (retencion)
```

Si en el main del programa principal creamos un objeto BBDD y ejecutamos el método:

```
// Creamos un objeto para gestionar la BBDD
String driver = "oracle.jdbc.driver.OracleDriver";
String url = "jdbc:oracle:thin:@192.168.2.39:1521:xe ";
String usuario = "desvan";
String password = "desvan";
BBDD oracleBBDD = new BBDD(driver, url, usuario, password);

// Obtenemos información de los empleados con retención mayor que 10 y la mostramos
int retencion = 10;
System.out.println(oracleBBDD.empleadosRetencionMayor(retencion));
```

El **resultado** será:

```
Juan Ignacio Martinez: 12341234121234567890 (10)
José Luis Pérez: 12342233121122334455 (12)
Manuel Lopez Marín: 55443322110099887766 (10)
Alfonso Gutierrez Lopez: 12563478001234567890 (12)
Encarna Lopez Lopez: 99118822773344665500 (10)
Rosa Lorite Lopez: 52341234521214567890 (10)
Lola Martinez Contreras: 22341224121224567820 (11)
```

Solución: BBDD.java

Respuesta: (penalty regime: 10, 20, ... %)

```
1  // Clase que gestiona una BBDD
2
3  import java.sql.*;
4  import java.util.*;
5
6  public class BBDD {
7
8     // ATRIBUTOS
9     private String driver;
10    private String url;
11    private String usuario;
12    private String password;
13
14    // Constructor
15    public BBDD(String driver, String url, String usuario, String password) {
16       this.driver = driver;
17       this.url = url;
18       this.usuario = usuario;
19       this.password = password;
20
```

	Expected

Expected

```
public String empleadosRetencionMayor(int retencion) throws SQLException {

   Connection conn = conexion();

   // Consultar datos con una consulta preparada
   String consulta = "SELECT cuenta, nombre, retencion FROM empleados WHERE retencion >=
   PreparedStatement stat = conn.prepareStatement(consulta);

   // Introducimos los valores recibidos en la consulta y la ejecutamos
   stat.setInt(1, retencion);
   ResultSet rs = stat.executeQuery();
   String formato = "";
   while (rs.next()) {
      formato += rs.getString(2) + ": " + rs.getString(1) + " (" + rs.getString(3) +")\n"
   }

   // Cerramos la conexión
   conn.close();

   // Devolvemos los datos
   return formato;
}
```

Your code must pass all tests to earn any marks. Try again.

Question author's solution:

```
public String empleadosRetencionMayor(int retencion) throws SQLException {

   Connection conn = conexion();

   // Consultar datos con una consulta preparada
   String consulta = "SELECT cuenta, nombre, retencion FROM empleados WHERE retencion >= ?";
   PreparedStatement stat = conn.prepareStatement(consulta);

   // Introducimos los valores recibidos en la consulta y la ejecutamos
   stat.setInt(1, retencion);
   ResultSet rs = stat.executeQuery();
   String formato = "";
   while (rs.next()) {
      formato += rs.getString(2) + ": " + rs.getString(1) + " (" + rs.getString(3) +")\n";
   }

   // Cerramos la conexión
   conn.close();

   // Devolvemos los datos
   return formato;
}
```

Incorrecta

Puntos para este envío: 0,00/1,00.

Pregunta **2**

Incorrecta

Puntúa 0,00 sobre 1,00

Crea el método **getEmpleado111** dentro de la clase BBDD . No recibe ningún parámetro y devuelve los datos del empleado con código 111 en un objeto de la clase Empleado .

Para ello, se usará el usuario desvan con la contraseña desvan. Recuerda configurar correctamente la dirección IP de la BBDD.

La **consulta** que debemos realizar es:

```
SELECT * FROM empleados WHERE codigo = 111
```

La consulta devolverá por cada empleado los siguientes datos en este mismo orden:

```
codigo: número entero
nombre: cadena de caracteres
hijos: número entero
retencion: número entero
cuenta: cadena de caracteres
fnacimiento: cadena de caracteres
```

Si en el main del programa principal creamos un objeto BBDD y ejecutamos el método:

```
// Creamos un objeto para gestionar la BBDD
String driver = "oracle.jdbc.driver.OracleDriver";
String url = "jdbc:oracle:thin:@192.168.2.39:1521:xe ";
String usuario = "desvan";
String password = "desvan";
BBDD oracleBBDD = new BBDD(driver, url, usuario, password);

// Obtenemos información sobre el empleado con código 111
Empleado elEmpleado = oracleBBDD.getEmpleado111();
System.out.println(elEmpleado);
```

El **resultado** será:

```
111   Encarna Lopez Lopez      0   10  99118822773344665500     1968-03-15 00:00:00.0
```

Solución: BBDD.java

Respuesta: (penalty regime: 10, 20, ... %)

```
1  import java.sql.*;
2  import java.util.*;
3
4  public class BBDD {
5
6     // ATRIBUTOS
7     private String driver;
8     private String url;
9     private String usuario;
10    private String password;
11
12    // Constructor
13    public BBDD(String driver, String url, String usuario, String password) {
14       this.driver = driver;
15       this.url = url;
16       this.usuario = usuario;
17       this.password = password;
18    }
19
20    // Realiza y devuelve la conexión con la BBDD
```

	Expected	Got

	Expected	Got

✗	`public Empleado getEmpleado111() throws SQLException {`
	
	`  Connection conn = conexion();`
	
	`  // Crear el objeto Statement, ejecutar consulta y obtener resultados`
	`  Statement stmt = conn.createStatement();`
	
	`  String consulta = "SELECT * FROM empleados WHERE codigo = 111";`
	`  ResultSet rs = stmt.executeQuery(consulta);`
	
	`  // Procesar el resultado y guardarlo en un objeto de la clase Empleado`
	`  Empleado empleado111 = null;`
	`  if (rs.next()) {`
	`  int codigo = rs.getInt("codigo");`
	`  String nombre = rs.getString("nombre");`
	`  int hijos = rs.getInt("hijos");`
	`  int retencion = rs.getInt("retencion");`
	`  String cuenta = rs.getString("cuenta");`
	`  String fecha = rs.getString("fnacimiento");`
	`  empleado111 = new Empleado(codigo, nombre, hijos, retencion, cuenta, fecha);`
	`  }`
	
	`  conn.close();`
	
	`  return empleado111;`
	`}`
		`***Error`
		`Tracebac`
		`File "`
		`comm`
		`IndexErr`

Your code must pass all tests to earn any marks. Try again.

Question author's solution:

```
public Empleado getEmpleado111() throws SQLException {

   Connection conn = conexion();

   // Crear el objeto Statement, ejecutar consulta y obtener resultados
   Statement stmt = conn.createStatement();

   String consulta = "SELECT * FROM empleados WHERE codigo = 111";
   ResultSet rs = stmt.executeQuery(consulta);

   // Procesar el resultado y guardarlo en un objeto de la clase Empleado
   Empleado empleado111 = null;
   if (rs.next()) {
      int codigo = rs.getInt("codigo");
      String nombre = rs.getString("nombre");
      int hijos = rs.getInt("hijos");
      int retencion = rs.getInt("retencion");
      String cuenta = rs.getString("cuenta");
      String fecha = rs.getString("fnacimiento");
      empleado111 = new Empleado(codigo, nombre, hijos, retencion, cuenta, fecha);
   }

   conn.close();

   return empleado111;
}
```

Incorrecta

Puntos para este envío: 0,00/1,00.

Pregunta **3**

Incorrecta

Puntúa 0,00 sobre 1,00

Crea el método **getEmpleados** dentro de la clase BBDD . No recibe ningún parámetro y devuelve un ArrayList de objetos de la clase Empleado con la información de todos los empleados de la BBDD.

Para ello, se usará el usuario desvan con la contraseña desvan. Recuerda configurar correctamente la dirección IP de la BBDD.

La **consulta** que debemos realizar es:

```
SELECT * FROM empleados
```

La consulta devolverá por cada empleado los siguientes datos en este mismo orden:

```
codigo: número entero
nombre: cadena de caracteres
hijos: número entero
retencion: número entero
cuenta: cadena de caracteres
fnacimiento: cadena de caracteres
```

Si en el main del programa principal creamos un objeto BBDD y ejecutamos el método:

```
// Creamos un objeto para gestionar la BBDD
String driver = "oracle.jdbc.driver.OracleDriver";
String url = "jdbc:oracle:thin:@192.168.2.39:1521:xe ";
String usuario = "desvan";
String password = "desvan";
BBDD oracleBBDD = new BBDD(driver, url, usuario, password);

// Obtenemos información sobre todos los Empleados
ArrayList<Empleado> listaEmpleados = oracleBBDD.getEmpleados();
System.out.println(listaEmpleados);
```

El **resultado** será:

```
[11    Juan Ignacio Martinez    0   10  12341234121234567890     1960-02-01 00:00:00.0, 1     J
osé Luis Pérez          2   12  12342233121122334455     1964-04-12 00:00:00.0, ..., 64738 Andr
és Morales Martín    3   7   22341154116231563690     1964-01-20 00:00:00.0]
```

Solución: BBDD.java

Respuesta: (penalty regime: 10, 20, ... %)

```
1  import java.sql.*;
2  import java.util.*;
3
4  public class BBDD {
5
6     // ATRIBUTOS
7     private String driver;
8     private String url;
9     private String usuario;
10    private String password;
11
12    // Constructor
13    public BBDD(String driver, String url, String usuario, String password) {
14       this.driver = driver;
15       this.url = url;
16       this.usuario = usuario;
17       this.password = password;
18    }
19
20    // Realiza y devuelve la conexión con la BBDD
```

	Expected

Expected

```
public ArrayList<Empleado> getEmpleados() throws SQLException {

   Connection conn = conexion();

   // Crear el objeto Statement, ejecutar consulta y obtener resultados
   Statement stmt = conn.createStatement();

   String consulta = "SELECT * FROM empleados";
   ResultSet rs = stmt.executeQuery(consulta);

   // Procesar resultados y guardarlos en una lista de objetos Empleado
   ArrayList<Empleado> listaEmpleados = new ArrayList<Empleado>();
   while (rs.next()) {
      int codigo = rs.getInt("codigo");
      String nombre = rs.getString("nombre");
      int hijos = rs.getInt("hijos");
      int retencion = rs.getInt("retencion");
      String cuenta = rs.getString("cuenta");
      String fecha = rs.getString("fnacimiento");
      Empleado unEmpleado = new Empleado(codigo, nombre, hijos, retencion, cuenta, fecha);
      listaEmpleados.add(unEmpleado);
   }

   conn.close();

   return listaEmpleados;
}
```

Your code must pass all tests to earn any marks. Try again.

Question author's solution:

```
public ArrayList<Empleado> getEmpleados() throws SQLException {

   Connection conn = conexion();

   // Crear el objeto Statement, ejecutar consulta y obtener resultados
   Statement stmt = conn.createStatement();

   String consulta = "SELECT * FROM empleados";
   ResultSet rs = stmt.executeQuery(consulta);

   // Procesar resultados y guardarlos en una lista de objetos Empleado
   ArrayList<Empleado> listaEmpleados = new ArrayList<Empleado>();
   while (rs.next()) {
      int codigo = rs.getInt("codigo");
      String nombre = rs.getString("nombre");
      int hijos = rs.getInt("hijos");
      int retencion = rs.getInt("retencion");
      String cuenta = rs.getString("cuenta");
      String fecha = rs.getString("fnacimiento");
      Empleado unEmpleado = new Empleado(codigo, nombre, hijos, retencion, cuenta, fecha);
      listaEmpleados.add(unEmpleado);
   }

   conn.close();

   return listaEmpleados;
}
```

Incorrecta

Puntos para este envío: 0,00/1,00.

Pregunta **4**

Incorrecta

Puntúa 0,00 sobre 1,00

Crea el método **getEmpleadosEmpiezan** dentro de la clase BBDD . Recibe una letra y devuelve un array de objetos de la clase Empleado con la información de todos los empleados cuyo nombre empieza por la letra indicada.

Para ello, se usará el usuario desvan con la contraseña desvan. Recuerda configurar correctamente la dirección IP de la BBDD.

La **consulta** que debemos realizar es:

```
SELECT * FROM empleados WHERE nombre LIKE 'X%'
```

La consulta devolverá por cada empleado los siguientes datos en este mismo orden:

```
codigo: número entero
nombre: cadena de caracteres
hijos: número entero
retencion: número entero
cuenta: cadena de caracteres
fnacimiento: cadena de caracteres
```

Si en el main del programa principal creamos un objeto BBDD y ejecutamos el método:

```
// Creamos un objeto para gestionar la BBDD
String driver = "oracle.jdbc.driver.OracleDriver";
String url = "jdbc:oracle:thin:@192.168.2.39:1521:xe ";
String usuario = "desvan";
String password = "desvan";
BBDD oracleBBDD = new BBDD(driver, url, usuario, password);

// Obtenemos información sobre los Empleados cuyo nombre empieza por la letra indicada
String letra = "A";
Empleado[] arrayEmpleados = oracleBBDD.getEmpleadosEmpiezan(letra);
System.out.println(Arrays.toString(arrayEmpleados));
```

El **resultado** será:

```
[67890 Alfonso Gutierrez Lopez  1   12  12563478001234567890     1967-11-05 00:00:00.0, 64738 A
ndrés Morales Martín    3   7   22341154116231563690     1964-01-20 00:00:00.0]
```

Solución: BBDD.java

Respuesta: (penalty regime: 10, 20, ... %)

```
1  import java.sql.*;
2  import java.util.*;
3
4  public class BBDD {
5
6     // ATRIBUTOS
7     private String driver;
8     private String url;
9     private String usuario;
10    private String password;
11
12    // Constructor
13    public BBDD(String driver, String url, String usuario, String password) {
14       this.driver = driver;
15       this.url = url;
16       this.usuario = usuario;
17       this.password = password;
18    }
19
20    // Realiza y devuelve la conexión con la BBDD
```

	Expected

Expected

```
public Empleado[] getEmpleadosEmpiezan(String letra) throws SQLException {

   Connection conn = conexion();

   // Consultar datos con una consulta preparada
   String consulta = "SELECT * FROM empleados WHERE nombre LIKE ?";
   PreparedStatement stat = conn.prepareStatement(consulta);

   // Introducimos los valores recibidos en la consulta y la ejecutamos
   stat.setString(1, letra + "%");
   ResultSet rs = stat.executeQuery();

   // Procesar resultados y guardarlos en una lista de objetos Empleado
   ArrayList<Empleado> listaEmpleados = new ArrayList<Empleado>();
   while (rs.next()) {
      int codigo = rs.getInt("codigo");
      String nombre = rs.getString("nombre");
      int hijos = rs.getInt("hijos");
      int retencion = rs.getInt("retencion");
      String cuenta = rs.getString("cuenta");
      String fecha = rs.getString("fnacimiento");
      Empleado unEmpleado = new Empleado(codigo, nombre, hijos, retencion, cuenta, fecha);
      listaEmpleados.add(unEmpleado);
   }

   // Guardamos el ArrayList en un array
   int longi = listaEmpleados.size();
   Empleado[] arrayEmpleados = new Empleado[longi];
   listaEmpleados.toArray(arrayEmpleados);

   // Cerramos la conexión
   conn.close();

   // Devolvemos el array
   return arrayEmpleados;
}
```

Your code must pass all tests to earn any marks. Try again.

Question author's solution:

```
public Empleado[] getEmpleadosEmpiezan(String letra) throws SQLException {
   Connection conn = conexion();

   // Consultar datos con una consulta preparada
   String consulta = "SELECT * FROM empleados WHERE nombre LIKE ?";
   PreparedStatement stat = conn.prepareStatement(consulta);

   // Introducimos los valores recibidos en la consulta y la ejecutamos
   stat.setString(1, letra + "%");
   ResultSet rs = stat.executeQuery();

   // Procesar resultados y guardarlos en una lista de objetos Empleado
   ArrayList<Empleado> listaEmpleados = new ArrayList<Empleado>();
   while (rs.next()) {
      int codigo = rs.getInt("codigo");
      String nombre = rs.getString("nombre");
      int hijos = rs.getInt("hijos");
      int retencion = rs.getInt("retencion");
      String cuenta = rs.getString("cuenta");
      String fecha = rs.getString("fnacimiento");
      Empleado unEmpleado = new Empleado(codigo, nombre, hijos, retencion, cuenta, fecha);
      listaEmpleados.add(unEmpleado);
   }

   // Guardamos el ArrayList en un array
   int longi = listaEmpleados.size();
   Empleado[] arrayEmpleados = new Empleado[longi];
   listaEmpleados.toArray(arrayEmpleados);

   // Cerramos la conexión
   conn.close();

   // Devolvemos el array
   return arrayEmpleados;
}
```

Incorrecta

Puntos para este envío: 0,00/1,00.

Pregunta **5**

Incorrecta

Puntúa 0,00 sobre 1,00

Crea el método **insertarSistemas** dentro de la clase BBDD . No recibe ningún parámetro y trata de insertar el departamento Sistemas con el código 7 en la BBDD. Devuelve true si la operación se ejecuta correctamente, false en caso contrario.

Para ello, se usará el usuario desvan con la contraseña desvan. Recuerda configurar correctamente la dirección IP de la BBDD.

La **consulta** que debemos realizar es:

```
INSERT INTO departamentos (codigo, nombre) VALUES (7, 'Sistemas')
```

Si en el main del programa principal creamos un objeto BBDD y ejecutamos el método:

```
// Creamos un objeto para gestionar la BBDD
String driver = "oracle.jdbc.driver.OracleDriver";
String url = "jdbc:oracle:thin:@192.168.2.39:1521:xe ";
String usuario = "desvan";
String password = "desvan";
BBDD oracleBBDD = new BBDD(driver, url, usuario, password);

// Insertamos un nuevo departamento
if (oracleBBDD.insertarSistemas()) {
   System.out.println("Los datos se han introducido correctamente");
} else {
   System.out.println("Problemas al introducir los datos");
}
```

La primera vez, el **resultado** será:

```
Los datos se han introducido correctamente
```

Las siguientes, el programa fallará porque estamos tratando de introducir un departamento con un código que ya existe (clave primaria). Tendremos que borrar el departamento para volverlo a introducir o cambiar el código.

Solución: BBDD.java

Respuesta: (penalty regime: 10, 20, ... %)

```
1  /*
2     Clase que gestiona una BBDD
3  */
4
5  import java.sql.*;
6  import java.util.*;
7
8  public class BBDD {
9
10    // ATRIBUTOS
11    private String driver;
12    private String url;
13    private String usuario;
14    private String password;
15
16    // Constructor
17    public BBDD(String driver, String url, String usuario, String password) {
18       this.driver = driver;
19       this.url = url;
20       this.usuario = usuario;
```

	Expected	G

Expected	G

✗ `public boolean insertarSistemas() throws SQLException {`
 	
`Connection conn = conexion();`
 	
`// Creamos la consulta y la ejecutamos`
`String consulta = "INSERT INTO departamentos (codigo, nombre) VALUES (7, 'Sistemas')";`
`Statement stat = conn.createStatement();`
`int resultado = stat.executeUpdate(consulta);`
 	
`conn.close();`
 	
`return resultado == 1;`
`}`
	*
	T
	
	I

Your code must pass all tests to earn any marks. Try again.

Question author's solution:

```
public boolean insertarSistemas() throws SQLException {

   Connection conn = conexion();

   // Creamos la consulta y la ejecutamos
   String consulta = "INSERT INTO departamentos (codigo, nombre) VALUES (7, 'Sistemas')";
   Statement stat = conn.createStatement();
   int resultado = stat.executeUpdate(consulta);

   conn.close();

   return resultado == 1;
}
```

Incorrecta

Puntos para este envío: 0,00/1,00.

Pregunta **6**
Incorrecta

Puntúa 0,00 sobre 1,00

Crea el método **borrarSistemas** dentro de la clase BBDD . No recibe ningún parámetro y trata de borrar el departamento Sistemas con el código 7 de la BBDD. Devuelve true si la operación se ejecuta correctamente, false en caso contrario.

Para ello, se usará el usuario desvan con la contraseña desvan. Recuerda configurar correctamente la dirección IP de la BBDD.

La **consulta** que debemos realizar es:

```
DELETE FROM departamentos WHERE nombre = 'Sistemas'
```

Si en el main del programa principal creamos un objeto BBDD y ejecutamos el método:

```
// Creamos un objeto para gestionar la BBDD
String driver = "oracle.jdbc.driver.OracleDriver";
String url = "jdbc:oracle:thin:@192.168.2.39:1521:xe ";
String usuario = "desvan";
String password = "desvan";
BBDD oracleBBDD = new BBDD(driver, url, usuario, password);

// Borramos el departamento
if (oracleBBDD.borrarSistemas()) {
   System.out.println("Los datos se han borrado correctamente");
} else {
   System.out.println("Problemas al borrar los datos");
}
```

La primera vez, el **resultado** será:

```
Los datos se han borrado correctamente
```

Las siguientes, el mensaje mostrado será:

```
Problemas al borrar los datos
```

Solución: BBDD.java

Respuesta: (penalty regime: 10, 20, ... %)

```
1  import java.sql.*;
2  import java.util.*;
3
4  public class BBDD {
5
6     // ATRIBUTOS
7     private String driver;
8     private String url;
9     private String usuario;
10    private String password;
11
12    // Constructor
13    public BBDD(String driver, String url, String usuario, String password) {
14       this.driver = driver;
15       this.url = url;
16       this.usuario = usuario;
17       this.password = password;
18    }
19
20    // Realiza y devuelve la conexión con la BBDD
```

	Expected	Got

Expected	Got

✗ `public boolean borrarSistemas() throws SQLException {`
 	
`  Connection conn = conexion();`
 	
`  // Creamos la consulta y la ejecutamos`
`  String consulta = "DELETE FROM departamentos WHERE nombre = 'Sistemas'";`
`  Statement stat = conn.createStatement();`
`  int resultado = stat.executeUpdate(consulta);`
 	
`  conn.close();`
 	
`  return resultado == 1;`
`}`
	`***Error***`
	`Traceback (most`
	`  File "prog.py`
	`  comment = "`
	`IndexError: tup`

Your code must pass all tests to earn any marks. Try again.

Question author's solution:

```
public boolean borrarSistemas() throws SQLException {

   Connection conn = conexion();

   // Creamos la consulta y la ejecutamos
   String consulta = "DELETE FROM departamentos WHERE nombre = 'Sistemas'";
   Statement stat = conn.createStatement();
   int resultado = stat.executeUpdate(consulta);

   conn.close();

   return resultado == 1;
}
```

Incorrecta

Puntos para este envío: 0,00/1,00.

Contacta con nosotros

Dirección: Calle Álava 41, interior - Vitoria-Gasteiz
Telefono : 945 567 953
E-mail: ulhi@ulhi.net
Twitter: @UrrutikoLH