Class DbManager

java.lang.Object
    DbManager

```
public class DbManager
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`DbManager()`   init the dbmanager class

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`void`	`BeforeFirstResult()` resets current row for res (sql query) **Important:** Use if while loop is being implemented
`void`	`Close()`
`java.lang.String[]`	`ColumnNames()` generates the current returned SQL column names
`java.lang.String[]`	`Columnsreturn()` This function returns the heading of the columns returnted from the last passed SQL query
`void`	`createConnection()` creates a connection to the external database server uses the DBConfig.cfg file to get: host address = jdbc:{server type}://{server address}/{databasename}?usePipelineAuth=false username = {valid database manager user} password = {corresponding password for username} driver = org.mariadb.jdbc.Driver {for mariadb db only check your own}
`void`	`FirstResult()` resets to first row for res (sql query) **Important:** Use only if a while loop isn't being implemented
`java.util.ArrayList<java.lang.String[]>`	`fullArray()`

	creates an arraylist of all all rows and columns without the column names as keys but instead uses an array
`java.util.ArrayList<java.util.HashMap<java.lang.String,java.lang.String>>`	**`fullmap`**`()` creates an arraylist of all rows and columns for the current SQL query
`java.lang.String`	**`getColumn`**`(java.lang.String name)` getColumn returns the value from the selected column for the current selected row
`int`	**`getColumnInt`**`(java.lang.String name)` getColumn returns the value from the selected column for the current selected row
`java.lang.Boolean`	**`isempty`**`()` checks if the query fetched valid data from the database
`java.util.HashMap<java.lang.String,java.lang.String>`	**`mapresult`**`()` creates a hashmap variable for currently selected row.
`java.util.HashMap<java.lang.String,java.lang.String>`	**`Pop`**`()` returns the last selected row while also moving onto the next row
`void`	**`printresult`**`()` printns out all rows from the last passed valid SQL query
`void`	**`query`**`(java.lang.String SQL)` Query is used to pass a sql statement to the connected database, which than saves the resulting fetched data to a local ResultSet called res
`void`	**`query`**`(java.lang.String SQL, java.lang.Object[] variables)` Query is used to pass a sql statement with the ability to add variables into the SQL statement through passing a list of variables, along with placing '?' within the SQL string corresponding to where you want the variables inserted.
`java.sql.ResultSet`	**`result`**`()` returns res.
`java.lang.String[]`	**`resultArray`**`()`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

DbManager

```
public DbManager()
          throws java.sql.SQLException,
                 java.io.IOException,
                 java.lang.ClassNotFoundException
```

init the dbmanager class

Throws:

`java.sql.SQLException - ?`

`java.io.IOException - ?`

`java.lang.ClassNotFoundException - ?`

Method Detail

createConnection

```
public void createConnection()
                      throws java.io.IOException,
                             java.lang.ClassNotFoundException,
                             java.sql.SQLException
```

creates a connection to the external database server uses the DBConfig.cfg file to get:
host address = jdbc:{server type}://{server address}/{databasename}?usePipelineAuth=false
username = {valid database manager user}
password = {corresponding password for username}
driver = org.mariadb.jdbc.Driver {for mariadb db only check your own}

Throws:

`java.io.IOException - ?`

`java.lang.ClassNotFoundException - ?`

`java.sql.SQLException - ?`

query

```
public void query(java.lang.String SQL)
```

Query is used to pass a sql statement to the connected database, which than saves the resulting fetched data to a local ResultSet called res

Parameters:

`SQL: - valid sql string statement`

query

```
public void query(java.lang.String SQL,
                  java.lang.Object[] variables)
```

Query is used to pass a sql statement with the ability to add variables into the SQL statement through passing a list of variables, along with placing '?' within the SQL string corresponding to where you want the variables inserted. This function than constructs the completed SQL statement and pass it to the database, which than saves the resulting fetched data to a local ResultSet called res

Parameters:

`SQL: - valid sql string statement place '?' where you want a variable inserted`

`variables: - array of variables you want to insert into your SQL statement`

Columnsreturn

```
public java.lang.String[] Columnsreturn()
```

This function returns the heading of the columns returnted from the last passed SQL query

Returns:

`Columns: A String array of the column heading from the current sql query statement`

isempty

```
public java.lang.Boolean isempty()
                          throws java.sql.SQLException
```

checks if the query fetched valid data from the database

Returns:

`Boolean: True - if result is empty / false - if result has data`

Throws:

`java.sql.SQLException - ?`

printresult

```
public void printresult()
                 throws java.sql.SQLException
```

printns out all rows from the last passed valid SQL query

Throws:

`java.sql.SQLException - ?`

result

```
public java.sql.ResultSet result()
```

returns res. Data fetched from SQL query variable

Returns:

`ResultSet res ?`

getColumn

```
public java.lang.String getColumn(java.lang.String name)
                           throws java.sql.SQLException
```

getColumn returns the value from the selected column for the current selected row

Parameters:

`name: - String of a column name of the current returned data`

Returns:

`String of a selected column from the current selected row`

Throws:

`java.sql.SQLException - ?`

getColumnInt

```
public int getColumnInt(java.lang.String name)
                 throws java.sql.SQLException
```

getColumn returns the value from the selected column for the current selected row

Parameters:

`name: - String of a column name of the current returned data`

Returns:

`int of a selected column from the current selected row`

Throws:

`java.sql.SQLException - ?`

resultArray

```
public java.lang.String[] resultArray()
                               throws java.sql.SQLException
```

Throws:

`java.sql.SQLException`

ColumnNames

```
public java.lang.String[] ColumnNames()
                               throws java.sql.SQLException
```

generates the current returned SQL column names

Returns:

`String[] list of all Column names currently selected`

Throws:

`java.sql.SQLException - ?`

mapresult

```
public java.util.HashMap<java.lang.String,java.lang.String> mapresult()
                                                               throws java.sql.SQLException
```

creates a hashmap variable for currently selected row.

Returns:

`HashMap of String, String where the key is the column name and the data is the selected row's column data`

Throws:

`java.sql.SQLException - ?`

fullmap

```
public java.util.ArrayList<java.util.HashMap<java.lang.String,java.lang.String>> fullmap()
                                                                                  throws java.sql.SQLException
```

creates an arraylist of all rows and columns for the current SQL query

Returns:

`ArrayList of HashMap of String,String where the array list holds every row within a HashMap pf String,String where the key is the column name and the data is the corresponding cell`

Throws:

`java.sql.SQLException - ?`

Pop

```
public java.util.HashMap<java.lang.String,java.lang.String> Pop()
                                                         throws java.sql.SQLException
```

returns the last selected row while also moving onto the next row

Returns:

`HashMap of String,String, where the key is the column name and the data is the selected row's column data`

Throws:

`java.sql.SQLException - ?`

fullArray

```
public java.util.ArrayList<java.lang.String[]> fullArray()
                                                throws java.sql.SQLException
```

creates an arraylist of all all rows and columns without the column names as keys but instead uses an array

Returns:

`ArrayList of String[]`

Throws:

`java.sql.SQLException - ?`

BeforeFirstResult

```
public void BeforeFirstResult()
                       throws java.sql.SQLException
```

resets current row for res (sql query) **Important:** Use if while loop is being implemented

Throws:

`java.sql.SQLException - ?`

FirstResult

```
public void FirstResult()
                 throws java.sql.SQLException
```

resets to first row for res (sql query) **Important:** Use only if a while loop isn't being implemented

Throws:

`java.sql.SQLException - ?`

Close

```
public void Close()
```