

Relational Databases with MySQL Week 10 Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document to the repository. Additionally, push an .sql file with all your queries and your Java project code to the same repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

In this week's coding activity, you will create a menu driven application backed by a MySQL database.

To start, choose one item that you like. It could be vehicles, sports, foods, etc....

Create a new Java project in Eclipse.

Create a SQL script in the project to create a database with one table. The table should be the item you picked.

Write a Java menu driven application that allows you to perform all four CRUD operations on your table.

Tips:

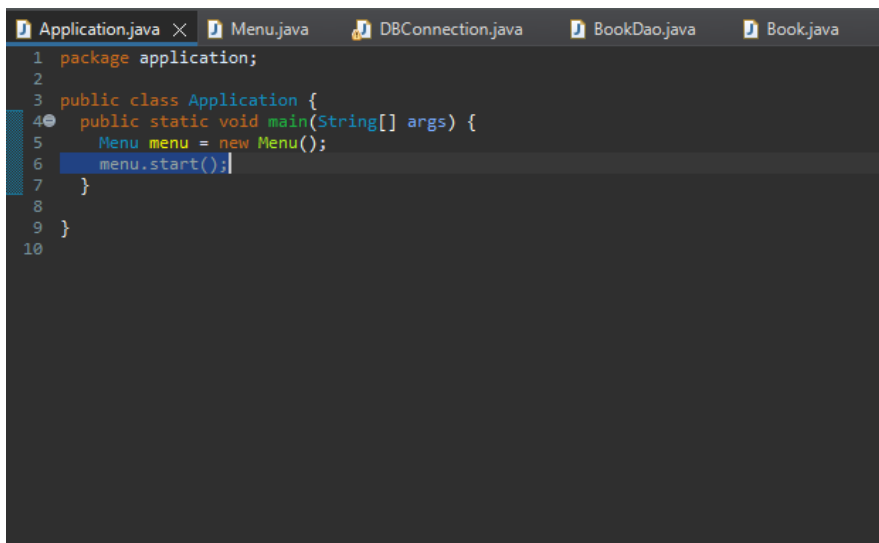
The application does not need to be as complex as the example in the video curriculum.

You need an option for each of the CRUD operations (Create, Read, Update, and Delete).

Remember that `PreparedStatement.executeQuery()` is only for Reading data and `.executeUpdate()` is used for Creating, Updating, and Deleting data.

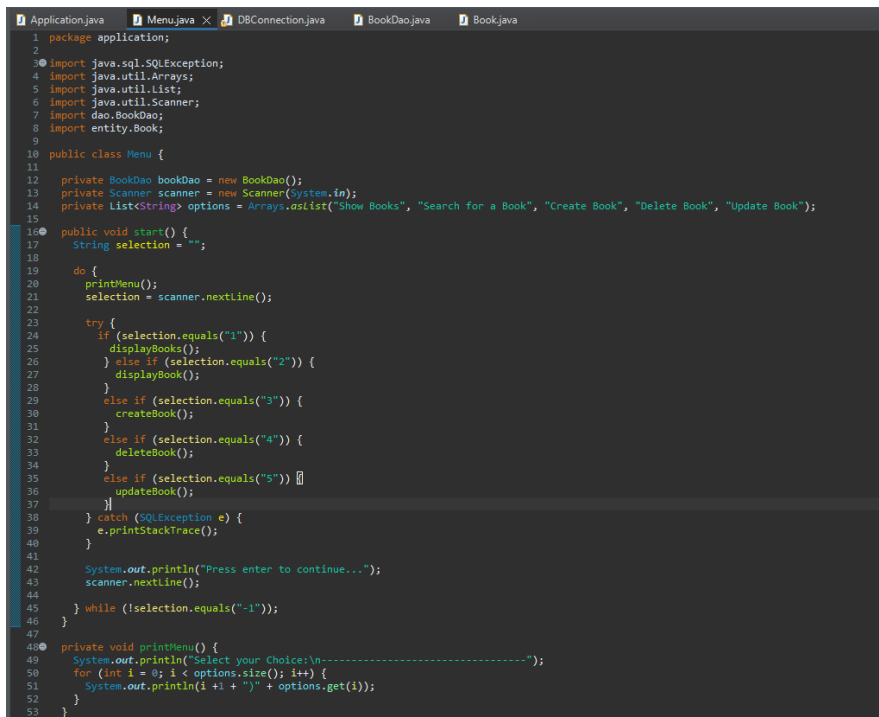
Remember that both parameters on `PreparedStatements` and the `ResultSet` columns are based on indexes that start with 1, not 0.

Screenshots of Code:



```
1 package application;
2
3 public class Application {
4     public static void main(String[] args) {
5         Menu menu = new Menu();
6         menu.start();
7     }
8
9 }
10
```

This screenshot shows the `Application.java` file in an IDE. It contains a simple `main` method that creates a `Menu` object and calls its `start` method. The package is `application`.



```
1 package application;
2
3 import java.sql.SQLException;
4 import java.util.Arrays;
5 import java.util.List;
6 import java.util.Scanner;
7 import dao.BookDao;
8 import entity.Book;
9
10 public class Menu {
11
12     private BookDao bookDao = new BookDao();
13     private Scanner scanner = new Scanner(System.in);
14     private List<String> options = Arrays.asList("Show Books", "Search for a Book", "Create Book", "Delete Book", "Update Book");
15
16     public void start() {
17         String selection = "";
18
19         do {
20             printMenu();
21             selection = scanner.nextLine();
22
23             try {
24                 if (selection.equals("1")) {
25                     displayBooks();
26                 } else if (selection.equals("2")) {
27                     displayBook();
28                 }
29                 else if (selection.equals("3")) {
30                     createBook();
31                 }
32                 else if (selection.equals("4")) {
33                     deleteBook();
34                 }
35                 else if (selection.equals("5")) {
36                     updateBook();
37                 }
38             } catch (SQLException e) {
39                 e.printStackTrace();
40             }
41
42             System.out.println("Press enter to continue...");
43             scanner.nextLine();
44         } while (!selection.equals("-1"));
45     }
46
47     private void printMenu() {
48         System.out.println("Select your Choice:\n-----");
49         for (int i = 0; i < options.size(); i++) {
50             System.out.println(i + 1 + " " + options.get(i));
51         }
52     }
53 }
```

This screenshot shows the `Menu.java` file in an IDE. It contains a `Menu` class with a `start` method that implements a menu loop. The menu options are "Show Books", "Search for a Book", "Create Book", "Delete Book", and "Update Book". The `start` method uses a `do-while` loop to repeatedly show the menu and process user input until the user enters "-1". The `printMenu` method is a private helper method that displays the menu options.

```

Application.java  Menu.java  DBConnection.java  BookDao.java  Book.java
48 private void printMenu() {
49     System.out.println("Select your Choice:\n-----");
50     for (int i = 0; i < options.size(); i++) {
51         System.out.println(i + " " + options.get(i));
52     }
53 }
54
55 private void displayBooks() throws SQLException {
56     List<Book> books = bookDao.getBooks();
57     for (Book book : books) {
58         System.out.println(book.getBookId() + " : " + book.getBookName() + "\n      Written By: " + book.getAuthorLastName() + "\n      Published In: " + book.get
59     }
60 }
61
62 private void displayBook() throws SQLException {
63     System.out.println("Enter book id: ");
64     int id = Integer.parseInt(scanner.nextLine());
65     Book book = bookDao.getBookById(id);
66     System.out.println(book.getBookId() + " : " + book.getBookName() + "\n      Written By: " + book.getAuthorLastName() + "\n      Published In: " + book.get
67 }
68
69 private void createBook() throws SQLException {
70     System.out.println("Enter Book Title: ");
71     String bookName = scanner.nextLine();
72     System.out.println("Enter the Last Name of the Author:");
73     String authorLastName = scanner.nextLine();
74     System.out.println("Enter the year the book was Published:");
75     String publishDate = scanner.nextLine();
76     bookDao.createNewBook(bookName, authorLastName, publishDate);
77 }
78
79 public void deleteBook() throws SQLException {
80     System.out.println("Enter id to delete: ");
81     int id = Integer.parseInt(scanner.nextLine());
82     bookDao.deleteBookById(id);
83     System.out.println("Book at id: " + id + " has been deleted.");
84 }
85
86 public void updateBook() throws SQLException {
87     System.out.println("Enter id to update: ");
88     int id = Integer.parseInt(scanner.nextLine());
89     System.out.println("Enter the book name you would like to use: ");
90     String bookName = scanner.next();
91     System.out.println("Enter the Author name you would like to use: ");
92     String authorLastName = scanner.next();
93     System.out.println("Enter the Publishing Year you would like to use: ");
94     String publishDate = scanner.next();
95     bookDao.updateBookById(id, bookName, authorLastName, publishDate);
96     System.out.println("Book at id: " + id + " updated.");
97 }
98 }
99 }
100

```

```
Application.java Menu.java DBConnection.java X BookDao.java Book.java
1 package dao;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class DBConnection {
8     private final static String URL = "jdbc:mysql://localhost:3306/books";
9     private final static String USERNAME = "root";
10    private final static String PASSWORD = "password";
11    private static Connection connection;
12    private static DBConnection instance;
13
14    private DBConnection(Connection connection) {
15        this.connection = connection;
16    }
17
18    public static Connection getConnection() {
19        if (instance == null) {
20            try {
21                connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
22                instance = new DBConnection(connection);
23                System.out.println("Connected");
24            } catch (SQLException e) {
25                e.printStackTrace();
26            }
27        }
28        return DBConnection.connection;
29    }
30
31 }
32
```

```
Application.java Menu.java DBConnection.java X BookDao.java Book.java
1 package dao;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.util.ArrayList;
8 import java.util.List;
9 import entity.Book;
10
11 public class BookDao {
12
13     private Connection connection;
14     private final String GET_BOOKS = "SELECT * FROM books";
15     private final String GET_BOOK_BY_ID = "SELECT * FROM books WHERE id = ?";
16     private final String CREATE_NEW_BOOK = "INSERT INTO books(name, author_lastname, publish_date) values (?, ?, ?)";
17     private final String DELETE_BOOK_BY_ID = "DELETE FROM books WHERE id = ?";
18     private final String UPDATE_BOOKS_BY_ID = "UPDATE books SET name = ?, author_lastname = ?, publish_date = ? WHERE id = ?";
19
20     public BookDao() {
21         connection = DBConnection.getConnection();
22     }
23
24     public List<Book> getBooks() throws SQLException {
25         ResultSet rs = connection.prepareStatement(GET_BOOKS).executeQuery();
26         List<Book> books = new ArrayList<Book>();
27
28         while (rs.next()) {
29             books.add(populateBook(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4)));
30         }
31
32         return books;
33     }
34
35     public Book getBookById(int id) throws SQLException {
36         PreparedStatement ps = connection.prepareStatement(GET_BOOK_BY_ID);
37         ps.setInt(1, id);
38         ResultSet rs = ps.executeQuery();
39         rs.next();
40         return populateBook(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4));
41     }
42
43     public void createNewBook(String bookName, String authorLastName, String publishedYear) throws SQLException {
44         PreparedStatement ps = connection.prepareStatement(CREATE_NEW_BOOK);
45         ps.setString(1, bookName);
46         ps.setString(2, authorLastName);
47         ps.setString(3, publishedYear);
48         ps.executeUpdate();
49     }
50
51     private Book populateBook(int id, String bookName, String authorLastName, String publishDate) {
52         return new Book(id, bookName, authorLastName, publishDate);
53     }
54 }
55
```

```

19
20 public BookDao() {
21     connection = DBConnection.getConnection();
22 }
23
24 public List<Book> getBooks() throws SQLException {
25     ResultSet rs = connection.prepareStatement(GET_BOOKS).executeQuery();
26     List<Book> books = new ArrayList<Book>();
27
28     while (rs.next()) {
29         books.add(populateBook(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4)));
30     }
31
32     return books;
33 }
34
35 public Book getBookById(int id) throws SQLException {
36     PreparedStatement ps = connection.prepareStatement(GET_BOOK_BY_ID);
37     ps.setInt(1, id);
38     ResultSet rs = ps.executeQuery();
39     rs.next();
40     return populateBook(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4));
41 }
42
43 public void createNewBook(String bookName, String authorLastName, String publishedYear) throws SQLException {
44     PreparedStatement ps = connection.prepareStatement(CREATE_NEW_BOOK);
45     ps.setString(1, bookName);
46     ps.setString(2, authorLastName);
47     ps.setString(3, publishedYear);
48     ps.executeUpdate();
49 }
50
51 private Book populateBook(int id, String bookName, String authorLastName, String publishDate) {
52     return new Book(id, bookName, authorLastName, publishDate);
53 }
54
55 public void deleteBookById(int id) throws SQLException {
56     PreparedStatement ps = connection.prepareStatement(DELETE_BOOK_BY_ID);
57     ps.setInt(1, id);
58     ps.executeUpdate();
59 }
60
61 public void updateBookById(int id, String bookName, String authorLastName, String publishDate) throws SQLException {
62     PreparedStatement ps = connection.prepareStatement(UPDATE_BOOKS_BY_ID);
63     ps.setString(1, bookName);
64     ps.setString(2, authorLastName);
65     ps.setString(3, publishDate);
66     ps.setInt(4, id);
67     ps.executeUpdate();
68 }
69
70 }
71

```

```

1 package entity;
2
3 public class Book {
4     private int bookId;
5     private String bookName;
6     private String authorLastName;
7     private String publishDate;
8
9     public Book(int bookId, String bookName, String authorLastName, String publishDate) {
10         this.setBookId(bookId);
11         this.setBookName(bookName);
12         this.setAuthorLastName(authorLastName);
13         this.setPublishDate(publishDate);
14     }
15
16     public int getBookId() {
17         return bookId;
18     }
19
20     public void setBookId(int bookId) {
21         this.bookId = bookId;
22     }
23
24     public String getBookName() {
25         return bookName;
26     }
27
28     public void setBookName(String bookName) {
29         this.bookName = bookName;
30     }
31
32     public String getAuthorLastName() {
33         return authorLastName;
34     }
35
36     public void setAuthorLastName(String authorLastName) {
37         this.authorLastName = authorLastName;
38     }
39
40     public String getPublishDate() {
41         return publishDate;
42     }
43
44     public void setPublishDate(String publishDate) {
45         this.publishDate = publishDate;
46     }
47
48 }
49
50
51

```

Screenshots of Running Application:

```
2)Search for a Book
3)Create Book
4)Delete Book
5)Update Book
4
Enter id to delete:
2
Book at id: 2 has been deleted.
Press enter to continue...

Select your Choice:
-----
1)Show Books
2)Search for a Book
3)Create Book
4)Delete Book
5)Update Book
1
1: Caesar
   Written By: Goldworthy
   Published In: 2012-01-01
Press enter to continue...

Select your Choice:
-----
1)Show Books
2)Search for a Book
3)Create Book
4)Delete Book
5)Update Book
3
Enter Book Title:
Napoleon: A Life
Enter the Last Name of the Author:
Roberts
Enter the Year the book was Published:
2014
Press enter to continue...

Select your Choice:
-----
1)Show Books
2)Search for a Book
3)Create Book
4)Delete Book
5)Update Book
1
1: Caesar
   Written By: Goldworthy
   Published In: 2012-01-01
3: Napoleon: A Life
   Written By: Roberts
<
```

URL to GitHub Repository:

<https://github.com/jaredInElit/SQL-Menu-Application>