Lu Peng

Create and initialize tables

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

public class BooksCreation {

	public static void main(String[] args) throws IOException {
		// load the JDBC driver
		try {
			Class.forName("oracle.jdbc.driver.OracleDriver");
		}
		catch(ClassNotFoundException e) {
			System.out.println("Unable to load the Driver class");
			System.exit(0);
		}

		// All database access is within a try/catch block. Connect to database,
		try {
			// get connection to a specific database using a username, and a password
			System.out.println("Connecting to the database...");
			Connection conn = DriverManager.getConnection(
					"jdbc:oracle:thin:@dagobah.engr.scu.edu:1521:db11g", "lpeng", "1985927pLPl");
			System.out.println("Database connected...");

			/* Create and execute SQL statement to create tables:
			 * the order in which the tables are created matters due to the referencing relations
			 * authors and publishers tables which referred by the other two tables are created first
			 * then titles table is created followed by the creation of authorISBN which refers to titles
			 */
			// create the authors table
			System.out.println("Creating table: authors...");
			Statement stmt1 = conn.createStatement();
			stmt1.executeUpdate(
					"CREATE TABLE authors(" // all columns are set to NOT NULL
					+ "authorID INTEGER PRIMARY KEY NOT NULL, " // authorID is primary key
					+ "firstName CHAR(20) NOT NULL, "
					+ "lastName CHAR(20) NOT NULL)");
			// set auto increment for authorID
			// create a sequence for auto increment
			String s1 = "CREATE SEQUENCE authors_seq "
					+ "START WITH 1 "
					+ "INCREMENT BY 1 NOMAXVALUE";
			// create a trigger for auto increment upon new row insertion
			String s2 = "CREATE OR REPLACE TRIGGER authors_trigger "
					+ "BEFORE INSERT ON authors "
					+ "FOR EACH ROW BEGIN SELECT authors_seq.nextval "
					+ "INTO :NEW.authorID FROM dual; end;";
			stmt1.executeUpdate(s1);
			stmt1.executeUpdate(s2);
			System.out.println("Table: authors created...");
			System.out.println();

			// create the publishers table
			System.out.println("Creating table: publishers...");
			Statement stmt4 = conn.createStatement();
			stmt4.executeUpdate(
					"CREATE TABLE publishers(" // all columns are set to NOT NULL
					+ "publisherID INTEGER NOT NULL, "
```

```
                        + "publisherName CHAR(100) NOT NULL, "
                        + "PRIMARY KEY(publisherID))"); // publisehrID is primary key
// set auto increment for authorID
// create a sequence for auto increment
s1 = "CREATE SEQUENCE publishers_seq "
                        + "START WITH 1 "
                        + "INCREMENT BY 1 NOMAXVALUE";
// create a trigger for auto increment upon new row insertion
s2 = "CREATE OR REPLACE TRIGGER publishers_trigger "
                        + "BEFORE INSERT ON publishers "
                        + "FOR EACH ROW BEGIN SELECT publishers_seq.nextval "
                        + "INTO :NEW.publisherID FROM dual; end;";
stmt4.executeUpdate(s1);
stmt4.executeUpdate(s2);
System.out.println("Table: publishers created...");
System.out.println();

// create the titles table
System.out.println("Creating table: titles...");
Statement stmt3 = conn.createStatement();
stmt3.executeUpdate(
                        "CREATE TABLE titles("
                        + "isbn CHAR(10) NOT NULL, "
                        + "title VARCHAR2(500) NOT NULL, "
                        + "editionNumber INTEGER NOT NULL, "
                        + "copyright CHAR(4) NOT NULL, "
                        + "publisherID INTEGER NOT NULL, "
                        + "price NUMBER(8,2) NOT NULL, " // NUMBER(8, -2) in description ???
                        + "PRIMARY KEY(isbn), "
                        + "FOREIGN KEY (publisherID) REFERENCES publishers(publisherID))"); // set foreign key
System.out.println("Table: titles created...");
System.out.println();

// create the authorISBN table
System.out.println("Creating table: authorISBN");
Statement stmt2 = conn.createStatement();
stmt2.executeUpdate(
                        "CREATE TABLE authorISBN("
                        + "authorID INTEGER NOT NULL, "
                        + "isbn CHAR(10) NOT NULL,"
                        + "FOREIGN KEY (authorID) REFERENCES authors (authorID), "  // set foreign key
                        + "FOREIGN KEY (isbn) REFERENCES titles (isbn))");
System.out.println("Table: authorISBN created...");
System.out.println();

/* add contents to the tables:
 * first, four text file containing data of the four tables are created
 * then, file I/O (Scanner) and SQL statement are used to read and insert data into the tables
 */
// fill authors table
Scanner input = new Scanner(new File("authors.txt"));
while(input.hasNextLine()) {
        String str = input.nextLine();
        String[] name = str.split(" ");
        String first = name[0].trim();
        String last = name[1].trim();
        stmt1.executeUpdate("INSERT INTO authors (firstName,lastName) values ('" + first + "', '" + last + "')");
}
System.out.println("Table: authors filled...");
System.out.println();

// fill publishers table
input = new Scanner(new File("publishers.txt"));
while(input.hasNextLine()) {
        String publisherName = input.nextLine().trim();
```

```
                    stmt4.executeUpdate("INSERT INTO publishers (publisherName) values ('" + publisherName + "')");
                }
                System.out.println("Table: publishers filled...");
                System.out.println();


                // fill titles table
                input = new Scanner(new File("titles.txt"));
                while(input.hasNextLine()) {
                    String line = input.nextLine();
                    String[] tokens = line.split(",");
                    String isbn = tokens[0].trim();
                    String titles = tokens[1].trim();
                    int editionNumber = Integer.parseInt(tokens[2].trim());
                    String copyright = tokens[3].trim();
                    int publisherID = Integer.parseInt(tokens[4].trim());
                    double price = Double.parseDouble(tokens[5].trim());
                    stmt3.executeUpdate("INSERT INTO titles (isbn,title,editionNumber,copyright,publisherID,price) values ("
                            + "'" + isbn + "', '" + titles + "', "
                            + editionNumber + ", '" + copyright + "', "
                            + publisherID + ", " + price + ")");
                }
                System.out.println("Table: titles filled...");
                System.out.println();

                // fill authorISBN table
                input = new Scanner(new File("authorISBN.txt"));
                while(input.hasNextLine()) {
                    int authorID = input.nextInt();
                    String isbn = input.nextLine().trim();
                    stmt2.executeUpdate("INSERT INTO authorISBN (authorID,isbn) values (" + authorID + ", '" + isbn + "')");
                }
                System.out.println("Table: authorISBN filled...");
                System.out.println();

                // release the database resources
                input.close();
                stmt1.close();
                stmt2.close();
                stmt3.close();
                stmt4.close();
                conn.close();
            }
            catch (SQLException se) {
                System.out.println("SQL Exception: " + se.getMessage());
                se.printStackTrace();
                System.exit(0);
            }

        }

}
```

Query and update tables

```
import java.sql.*;
import java.util.*;


public class BooksQuery {

	public static void main(String[] args) {
		// load the JDBC driver
		try {
			Class.forName("oracle.jdbc.driver.OracleDriver");
		}
		catch(ClassNotFoundException e) {
			System.out.println("Unable to load the Driver class");
			System.exit(0);
		}

		// All database access is within a try/catch block. Connect to database,
		try {
			// get connection to a specific database using a username, and a password
			System.out.println("\nConnecting to the database...");
			Connection conn = DriverManager.getConnection(
					"jdbc:oracle:thin:@dagobah.engr.scu.edu:1521:db11g", "lpeng", "1985927pLPl");
			System.out.println("Database connected...\n");

			// display a set of options specified in the project description
			// ask user to select query operations
			Scanner input = new Scanner(System.in);
			boolean active = true;
			while(active) {
				System.out.println("Available operations include:"
						+ "\n\t0--Exit"
						+ "\n\t1--Select all authors from authors table ordered by last and first name"
						+ "\n\t2--Select all publishers from publishers table"
						+ "\n\t3--Select a specific publisher and list all books published by that publisher. "
						+ "\n\t   Include the title, year and ISBN number. Order the information alphabetically by title."
						+ "\n\t4--Add new author"
						+ "\n\t5--Edit/Update the existing information about an author"
						+ "\n\t6--Add a new title for an author"
						+ "\n\t7--Add new publisher"
						+ "\n\t8--Edit/Update the existing information about a publisher");
				System.out.print("\nEnter your choice: ");
				int choice = Integer.parseInt(input.nextLine());
				System.out.println();

				// use switch block to process user selection
				switch(choice) {
					case 0: active = false; break; // exit at 0
					case 1: getAllAuthors(conn); break;
					case 2: getAllPublishers(conn); break;
					case 3: getAllBooksFromOnePublisher(conn); break;
					case 4: addNewAuthor(conn); break;
					case 5: updateAuthor(conn); break;
					case 6: addNewTitleForAuthor(conn); break;
					case 7: addNewPublisher(conn); break;
					case 8: updatePublisher(conn); break;
				}
				System.out.println("-----------------------------------------------");
			}
			input.close();
			conn.close();
			System.out.println("Database disconnected");
```

```
		}
		catch (SQLException se) {
			System.out.println("SQL Exception: " + se.getMessage());
			se.printStackTrace();
			System.exit(0);
		}
}


// 8--Edit/Update the existing information about a publisher
private static void updatePublisher(Connection conn) {
		try {
			Statement stmt = conn.createStatement();
			ResultSet rs = stmt.executeQuery("SELECT * FROM publishers");

			if(rs == null) {
				System.out.println("Empty table");
				return;
			}

			// display the publishers table for user convenience
			System.out.println("Available publishers include: ");
			System.out.printf("%-15s%-100s\n"
					+ "-----------------------------------------------------------------\n",
					"publisherID", "publisherName");
			while(rs.next()) {
				System.out.printf("%-15s%-100s\n",
						rs.getInt("publisherID"), rs.getString("publisherName"));
			}
			System.out.println("-----------------------------------"
					+ "--------------------------");

			// user selects a publisher ID and enters a new publisher name to update the table
			Scanner input = new Scanner(System.in);
			System.out.print("Enter the publisher ID: ");
			String publisherID = input.nextLine();
			System.out.print("Enter the new publisher name: ");
			String publisherName = input.nextLine();

			// confirm with user for update
			System.out.print("Do you want to commit update? (yes/no): ");
			String ans = input.nextLine();
			//input.close(); read from System.in, same as the one in main; if this is closed, main scanner alse close
			if(!ans.equalsIgnoreCase("yes")) return;

			// commit update
			int count = stmt.executeUpdate("UPDATE publishers SET publisherName = '"
					+ publisherName + "' WHERE publisherID = " + publisherID);

			// display update result
			if(count > 0) {
				System.out.println("publisher ID: " + publisherID + " updated");
			}
			else {
				System.out.println("update failed");
			}

			stmt.close();
			rs.close();
		}
		catch (SQLException e) {
			e.printStackTrace();
			return;
		}
```

```
}

// 7--Add new publisher
private static void addNewPublisher(Connection conn) {
	try {
		// ask user to enter a new entry for publishers table
		Scanner input = new Scanner(System.in);
		System.out.print("Enter a new publisher name: ");
		String publisherName = input.nextLine();

		// confirm with user for update
		System.out.print("Do you want to commit update? (yes/no): ");
		String ans = input.nextLine();
		//input.close();
		if(!ans.equalsIgnoreCase("yes")) return;

		// check duplicate publisher name
		Statement stmt = conn.createStatement();
		ResultSet rs = stmt.executeQuery("SELECT publisherName FROM publishers");
		while(rs.next()) {
			if(rs.getString("publisherName").equalsIgnoreCase(publisherName)) {
				System.out.println("Duplicate publisher name detected: " + publisherName);
				return;
			}
		}

		// insert the new publisher
		int count = stmt.executeUpdate("INSERT INTO publishers (publisherName) VALUES ('" + publisherName + "')");
		if(count > 0) {
			System.out.println("publisher name: " + publisherName + " added");
		}
		else {
			System.out.println("Insertion failed");
		}

		stmt.close();
		rs.close();
	}
	catch(SQLException e) {
		e.printStackTrace();
		return;
	}

}

// 6--Add a new title for an author
private static void addNewTitleForAuthor(Connection conn) {
	try {
		Statement stmt = conn.createStatement();
		ResultSet rs = stmt.executeQuery("SELECT * FROM authors");

		// first select the author from the authors table
		System.out.println("Available authors include: \n");
		System.out.printf("%-10s%-25s%-25s\n"
				+ "---------------------------------------------------------------\n",
				"authorID", "First_Name", "Last_Name");
		while(rs.next()) {
			System.out.printf("%-10s%-25s%-25s\n",
					rs.getInt("authorID"), rs.getString("firstName"), rs.getString("lastName"));
		}
		System.out.println("---------------------------------------------------------------");
		Scanner input = new Scanner(System.in);
		System.out.print("Enter book author ID: ");
		int authorID = Integer.parseInt(input.nextLine());
```

```
                    // add a new title entry to titles table
                    System.out.print("Enter book ISBN: ");
                    String isbn = input.nextLine();
                    System.out.print("Enter book title: ");
                    String title = input.nextLine();
                    System.out.print("Enter book edition number: ");
                    int editionNumber = Integer.parseInt(input.nextLine().trim());
                    System.out.print("Enter book copyright year: ");
                    String copyright = input.nextLine();
                    System.out.print("Enter book publisher ID: ");
                    int publisherID = Integer.parseInt(input.nextLine().trim());
                    System.out.print("Enter book price: ");
                    double price = Double.parseDouble(input.nextLine().trim());

                    // confirm with user for update
                    System.out.print("Do you want to commit update? (yes/no): ");
                    String ans = input.nextLine();
                    //input.close();
                    if(!ans.equalsIgnoreCase("yes")) return;

                    // insert into titles table
                    int count1 = stmt.executeUpdate("INSERT INTO titles (isbn, title, editionNumber, "
                                    + "copyright, publisherID, price) values ("
                                    + "'" + isbn + "', '" + title + "', "
                                    + editionNumber + ", '" + copyright + "', "
                                    + publisherID + ", " + price + ")");
                    if(count1 > 0) {
                            System.out.println("one row inserted into titles");
                    }
                    else {
                            System.out.println("insertion failed");
                    }

                    // insert into authorISBN table
                    int count2 = stmt.executeUpdate("INSERT INTO authorISBN (authorID,isbn) values (" + authorID + ", '" + isbn + "')");
                    if(count2 > 0) {
                            System.out.println("one row inserted into authorISBN");
                    }
                    else {
                            System.out.println("insertion failed");
                    }

                    stmt.close();
                    rs.close();
            }
            catch (SQLException e) {
                    e.printStackTrace();
                    return;
            }
}

// 5--Edit/Update the existing information about an author
private static void updateAuthor(Connection conn) {
            try {
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery("SELECT * FROM authors");

                    if(rs == null) {
                            System.out.println("Empty table");
                            return;
                    }

                    // display the authors table
                    System.out.println("Available authors include: \n");
                    System.out.printf("%-10s%-25s%-25s\n"
```

```
                                + "----------------------------------------------------------------\n",
                                "authorID", "First_Name", "Last_Name");
                while(rs.next()) {
                        System.out.printf("%-10s%-25s%-25s\n",
                                        rs.getInt("authorID"), rs.getString("firstName"), rs.getString("lastName"));
                }

                // user selects an author ID and enters the new author first and last name to update the table
                Scanner input = new Scanner(System.in);
                System.out.print("\nEnter the author ID: ");
                String authorID = input.nextLine();
                System.out.print("Enter the new publisher name: ");
                String firstName = input.next().trim();
                String lastName = input.nextLine().trim();

                // confirm with user for update
                System.out.print("Do you want to commit update? (yes/no): ");
                String ans = input.nextLine().trim();
                //input.close();
                if(!ans.equalsIgnoreCase("yes")) return;

                // commit update
                int count = stmt.executeUpdate("UPDATE authors SET firstName = '"
                                + firstName + "', lastName = '" + lastName + "' WHERE authorID = " + authorID);
                if(count > 0) {
                        System.out.println("author ID: " + authorID + " updated");
                }
                else {
                        System.out.println("update failed");
                }

                stmt.close();
                rs.close();
        }
        catch (SQLException e) {
                e.printStackTrace();
                return;
        }

}

// 4--Add new author
private static void addNewAuthor(Connection conn) {
        try {
                // ask user to enter a new entry for authors table
                Scanner input = new Scanner(System.in);
                System.out.print("Enter a new author name, eg. Lu Peng: ");
                String str = input.nextLine();
                String[] name = str.split(" ");
                if(name.length != 2) {
                        System.out.println("Wrong name format");
                        return;
                }
                String firstName = name[0].trim();
                String lastName = name[1].trim();

                // check duplicate author names
                Statement stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery("SELECT firstName, lastName FROM authors");
                while(rs.next()) {
                        if(rs.getString("firstName").equalsIgnoreCase(firstName)
                                        && rs.getString("lastName").equalsIgnoreCase(lastName)) {
                                System.out.println("Duplicate author: " + firstName + " " + lastName);
                                return;
                        }
```

```
                }

                // confirm with user for update
                System.out.print("Do you want to commit update? (yes/no): ");
                String ans = input.nextLine();
                //input.close();
                if(!ans.equalsIgnoreCase("yes")) return;

                // insert the new publisher into the table
                int count = stmt.executeUpdate("INSERT INTO authors (firstName, lastName) "
                                + "VALUES ('" + firstName + "', '" + lastName + "')");
                if(count > 0) {
                        System.out.println("author: " + firstName + " " + lastName + " added");
                }
                else {
                        System.out.println("Update failed");
                }

                stmt.close();
                rs.close();
        }
        catch(SQLException e) {
                e.printStackTrace();
                return;
        }

}

// 3--Select a specific publisher and list all books published by that publisher.
// Include the title, year and ISBN number. Order the information alphabetically by title.
private static void getAllBooksFromOnePublisher(Connection conn) {
        try {
                Statement stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery("SELECT * FROM publishers");

                if(rs == null) {
                        System.out.println("Empty table");
                        return;
                }

                // display the publishers table for user convenience
                System.out.println("Available publishers include: ");
                System.out.printf("%-15s%-100s\n"
                                + "-----------------------------------------------------------------\n",
                                "publisherID", "publisherName");
                while(rs.next()) {
                        System.out.printf("%-15s%-100s\n",
                                        rs.getInt("publisherID"), rs.getString("publisherName"));
                }
                System.out.println("-----------------------------------"
                                + "----------------------------");

                // user selects a publisher ID
                Scanner input = new Scanner(System.in);
                System.out.print("Enter the publisher ID: ");
                String publisherID = input.nextLine();

                // get the publisher name
                rs = stmt.executeQuery("SELECT publisherName FROM publishers "
                                + "WHERE publisherID = '" + publisherID + "'");
                rs.next();
                String publisherName = rs.getString("publisherName");

                // get result set and display
                rs = stmt.executeQuery("SELECT title, copyright, isbn "
```

```
                                        + "FROM titles "
                                        + "WHERE publisherID = '" + publisherID
                                        + "' ORDER BY title");
                        System.out.println("Books published by " + publisherID + ", " + publisherName);
                        System.out.printf("%-50s%-10s%-10s\n"
                                        + "-----------------------------------------------------------\n",
                                        "title", "year", "ISBN");
                        while(rs.next()) {
                                System.out.printf("%-50s%-10s%-10s\n",
                                                rs.getString("title"), rs.getString("copyright"), rs.getString("isbn"));
                        }

                        stmt.close();
                        rs.close();
                }
                catch(SQLException e) {
                        e.printStackTrace();
                        return;
                }

}

// 2--Select all publishers from publishers table
private static void getAllPublishers(Connection conn) {
                try {
                        Statement stmt = conn.createStatement();
                        ResultSet rs = stmt.executeQuery("SELECT * FROM publishers");

                        if(rs == null) {
                                System.out.println("Empty table");
                                return;
                        }

                        // display the publishers table for user convenience
                        System.out.println("Available publishers include: ");
                        System.out.printf("%-15s%-100s\n"
                                        + "-----------------------------------------------------------\n",
                                        "publisherID", "publisherName");
                        while(rs.next()) {
                                System.out.printf("%-15s%-100s\n",
                                                rs.getInt("publisherID"), rs.getString("publisherName"));
                        }

                        stmt.close();
                        rs.close();
                }
                catch(SQLException e) {
                        e.printStackTrace();
                        return;
                }

}

// 1--Select all authors from authors table ordered by last and first name
private static void getAllAuthors(Connection conn) {
                try {
                        Statement stmt = conn.createStatement();
                        ResultSet rs = stmt.executeQuery("SELECT * FROM authors ORDER BY lastName, firstName");

                        if(rs == null) {
                                System.out.println("Empty table");
                                return;
                        }

                        // display the authors table
```

```
                    System.out.println("Available authors include: \n");
                    System.out.printf("%-10s%-25s%-25s\n"
                                    + "-----------------------------------------------------------------\n",
                                    "authorID", "First_Name", "Last_Name");
                    while(rs.next()) {
                            System.out.printf("%-10s%-25s%-25s\n",
                                            rs.getInt("authorID"), rs.getString("firstName"), rs.getString("lastName"));
                    }

                    stmt.close();
                    rs.close();
            }
            catch (SQLException e) {
                    e.printStackTrace();
                    return;
            }

    }

}
```