JDBC Project

Greg Violan
Sarah Han
CECS 323
Database Fundamentals

JAVA-

```
package cecs.pkg323.java.project;

import java.sql.*;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * This class connects to a database JDBC and manipulate the data inside it. It can update, add,
 * remove using prepared statements. The class connects to db first, then runs its prompts from
the user.
 * It is then manipulated by the user choices.
 * @author Greg Violan and Sarah Han
 */
public class CECS323JavaProject {
   //  Database credentials & private variables
   private static String DBNAME;
   private static boolean exit = false;
   private static Connection conn = null; //initialize the connection
   private static Statement stmt = null;  //initialize the statement that we're using
   private static ResultSet rs = null;

   //This is the specification for the printout that I'm doing:
   //each % denotes the start of a new field.
   //The - denotes left justification.
   //The number indicates how wide to make the field.
   //The "s" denotes that it's a string.  All of our output in this test are
   //strings, but that won't always be the case.
   private static final String displayFormat_1="%-20s%-25s%-20s%-20s\n";
   private static final String displayFormat_2="%-20s%-30s%-20s%-20s%-20s\n";


// JDBC driver name and database URL
   private static final String JDBC_DRIVER = "org.apache.derby.jdbc.ClientDriver"; // the
location of the database
   private static String DB_URL = "jdbc:derby://localhost:1527/"; // accessing the network of the
database
//          + "testdb;user=";

   /*
   * main class connects to the database and it prompts the user for
* input to manipulate the database.
*/
public static void main(String[] args) {
    try {
        // Connecting to DB
        connectToDB();

        // Workflow
        while(!exit){
            int menu_input = menu();
            input(menu_input);
            if(!exit){
                display();
            }
        }

        // Clean up
        if(rs != null || stmt != null){
            rs.close();
            stmt.close();
        }
        conn.close();
    } catch (SQLException | ClassNotFoundException ex) {
        Logger.getLogger(CECS323JavaProject.class.getName()).log(Level.SEVERE, null, ex);
    }  finally {
        //finally block used to close resources
        try {
            if (stmt != null) {
                stmt.close();
            }
        } catch (SQLException se2) {
        }// nothing we can do
        try {
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException se) {
        }//end finally try
    }//end try

    System.out.println("\nGoodbye!");
}//end main
/*
```

```
    * connects to the database
    */
    public static void connectToDB() throws SQLException, ClassNotFoundException{
        //Prompt the user for the database name, and the credentials.
        //If your database has no credentials, you can update this code to
        //remove that from the connection string.
        Scanner in = new Scanner(System.in);
        System.out.print("Name of the database (not the user account): ");
        DBNAME = in.nextLine();

        //Constructing the database URL connection string
        DB_URL = DB_URL + DBNAME;

        //STEP 2: Register JDBC driver
        Class.forName("org.apache.derby.jdbc.ClientDriver");

        //STEP 3: Open a connection
        System.out.println("Connecting to database...");
        conn = DriverManager.getConnection(DB_URL);
    }
    /*
    * menu choices for prompting the user
    */
    public static int menu(){
        Scanner scnr = new Scanner(System.in);
        System.out.println("\n1. List all writing groups \n2. List all the data for a group specified by
the user"
                + "\n3. List all publishers \n4. List all the data for a publisher specified by the user"
                + "\n5. List all books titles \n6. List all the data for a book specified by the user"
                + "\n7. Insert a new book \n8. Insert a new publisher and update all book published by
"
                + "one publisher to be published by the new publisher \n9. Remove a book specified
by the user"
                + "\n10. Exit");
        System.out.println("Please choose a command:");
        int input = checkInt(1,10);
        return input;
    }
    /*
    * validated input that the user has made and do that business
    */
    public static void input(int i) throws SQLException{
        Scanner scnr = new Scanner(System.in);
```

```
String input = null, input2 = "";
stmt = conn.createStatement();

switch(i){
   case 1:
      rs = stmt.executeQuery("select * from writing_groups");
      break;
   case 2:
      System.out.println("Please enter the writing group name:"); // need to check input
      input = scnr.nextLine();
      select("writing_groups", "group_name", input);
      break;
   case 3:
      rs = stmt.executeQuery("select * from publishers");
      break;
   case 4:
      System.out.println("Please enter the publisher name:"); // need to check input
      input = scnr.nextLine();
      select("publishers", "pub_name", input);
      break;
   case 5:
      rs = stmt.executeQuery("select * from books");
      break;
   case 6:
      boolean check = true;
      while(check){
         System.out.println("Please enter the group name:");
         input = scnr.nextLine();
         check = checking("writing_groups", 1, input);
         if(!check){
            System.out.println("**invalid group name: does not exist**");
            check = true;
         }
         else{
            check = false;
         }
      }

      System.out.println("Please enter the book title:"); // need to check input
      input2 = scnr.nextLine();
      selectM("books", "group_name", "book_title", input, input2);
      break;
   case 7: // add book
```

```
                addBook();
                break;
            case 8: // add publisher
                addPublisher();
                break;
            case 9: // remove book
                removeBook();
                break;
            case 10:
                exit = true;
                break;
            default:
                System.out.println("default..");
                break;
        }
    }
    /*
    * displays the results of the table
    */
    public static void display() throws SQLException{
        ResultSetMetaData rsmd = null;
        String attribute_1, attribute_2, attribute_3, attribute_4, attribute_5 = "";

        System.out.println("\nCreating statement...");

        rsmd = rs.getMetaData();

        // Retrieve by column name
        attribute_1 = rsmd.getColumnName(1);
        attribute_2 = rsmd.getColumnName(2);
        attribute_3 = rsmd.getColumnName(3);
        attribute_4 = rsmd.getColumnName(4);
        if(rsmd.getColumnCount() == 5){
            attribute_5 = rsmd.getColumnName(5);
        }

        if(rsmd.getColumnCount() < 5)
            System.out.printf(displayFormat_1, attribute_1, attribute_2, attribute_3, attribute_4);
        else
            System.out.printf(displayFormat_2, attribute_1, attribute_2, attribute_3, attribute_4,
attribute_5);

        // Extracting data from result set
    // Display values
    while(rs.next()){
        if(rsmd.getColumnCount() < 5){
            System.out.printf(displayFormat_1,
            dispNull(rs.getString(attribute_1)), dispNull(rs.getString(attribute_2)),
            dispNull(rs.getString(attribute_3)), dispNull(rs.getString(attribute_4)));
        }
        else{
            System.out.printf(displayFormat_2,
            dispNull(rs.getString(attribute_1)), dispNull(rs.getString(attribute_2)),
            dispNull(rs.getString(attribute_3)), dispNull(rs.getString(attribute_4)),
            dispNull(rs.getString(attribute_5)));
        }
    }// end of while

}
/*
* this uses a prepared statement for selecting a table and a certain attribute
*/
public static void select(String table, String attribute, String input) throws SQLException{
    PreparedStatement pstmt = null;
    String prepare = "";

    prepare = "SELECT * FROM " + table + " WHERE " + attribute + " = ?";
    pstmt = conn.prepareStatement(prepare);
    pstmt.setString(1, input);

    rs = pstmt.executeQuery();
}
/*
* this uses a prepared statement for selecting a table and two attributes
*/
public static void selectM(String table, String attribute, String attribute2
        , String input, String input2) throws SQLException
{
    PreparedStatement pstmt = null;
    String prepare = "";

    prepare = "SELECT * FROM " + table + " WHERE " + attribute + " = ?"
            + " AND " + attribute2 + " = ?";
    pstmt = conn.prepareStatement(prepare);
    pstmt.setString(1, input);
    pstmt.setString(2, input2);
    rs = pstmt.executeQuery();
}
/*
* this adds a book into the database, also handles all the invalid inputs
*/
public static void addBook() throws SQLException{
    Scanner scnr = new Scanner(System.in);
    String input, gn = null, bt = null, pn = null, yr = "";
    int pg = 0;
    PreparedStatement pstmt = null;
    boolean check = true;

    System.out.println("Adding book..");
    while(check){
        System.out.println("Please enter group name:");
        gn = scnr.nextLine();
        check = checking("writing_groups", 1, gn);
        if(!check){
            System.out.println("**invalid writing group name: does not exist**");
            check = true;
        } else {
            check = false;
        }
    }

    check = true;
    while(check){
        System.out.println("Please enter book title:");
        bt = scnr.nextLine();
        check = checking("books", 2, bt);
        if(!check){
            check = false;
        } else {
            System.out.println("**invalid book's name: it already exist**");
            check = true;
        }
    }

    check = true;
    while(check){
        System.out.println("Please enter publisher's name:");
        pn = scnr.nextLine();
```

```
            check = checking("publishers", 1, pn);
            if(!check){
                System.out.println("**invalid publisher's name: does not exist**");
                check = true;
            } else {
                check = false;
            }
        }

        check = true;
        while(check){
            System.out.println("Please enter year of publish:");
            yr = scnr.nextLine();
            if(yr.length() > 4 || !isInteger(yr)){
                System.out.println("**invalid year**");
            } else {
                check = false;
            }
        }

        System.out.println("Please enter number of pages:");
        pg = checkInt(0, 999999);
        input = "INSERT INTO books"
        + "(group_name, book_title, pub_name, year_published, pages) VALUES"
        + "(?,?,?,?,?)";

        pstmt = conn.prepareStatement(input);
        pstmt.setString(1, gn);
        pstmt.setString(2, bt);
        pstmt.setString(3, pn);
        pstmt.setString(4, yr);
        pstmt.setInt(5, pg);

        pstmt.executeUpdate();
        rs = stmt.executeQuery("SELECT * FROM books");
    }
    /*
    * removes a book in the database
    */
    public static void removeBook() throws SQLException{
        Scanner scnr = new Scanner(System.in);
        PreparedStatement pstmt = null;
        String temp, input = "";
        boolean check = true;

        while(check){
            System.out.println("Please enter the book title ('c' to cancel):");
            input = scnr.nextLine();
            check = checking("books", 2, input);
            if(input.equalsIgnoreCase("c")){
                check = false;
            }
            else if(!check){
                System.out.println("**invalid book title: does not exist**");
                check = true;
            }
            else{
                check = false;
                temp = "DELETE FROM books WHERE book_title = ?";
                pstmt = conn.prepareStatement(temp);
                pstmt.setString(1, input);

                pstmt.executeUpdate();
                rs = stmt.executeQuery("select * from books");
            }
        }


    }
    /**
     * adds a publisher in the database
     * @throws SQLException
     */
    public static void addPublisher() throws SQLException{
        Scanner scnr = new Scanner(System.in);
        String input, pn = null, pa, pp = null, pe, pu = "";
        PreparedStatement pstmt = null;
        boolean check = true;

        while(check){
            System.out.println("Adding publisher..\nPlease enter the publisher's name:");
            pn = scnr.nextLine();
            check = checking("publishers", 1, pn);
            if(!check){
                check = false;
            }
            else{
                System.out.println("**invalid publisher: already exist**");
                check = true;
            }
        }

        System.out.println("Please enter the publisher's address title:");
        pa = scnr.nextLine();


        System.out.println("Please enter the publisher's phone:");
        pp = scnr.nextLine();

        System.out.println("Please enter the publisher's email");
        pe = scnr.nextLine();

        check = true;
        while(check){
            System.out.println("Please enter which publisher you want to update it on");
            pu = scnr.nextLine();
            check = checking("publishers", 1, pu);
            if(!check){
                System.out.println("**invalid publisher name**");
                check = true;
            }
            else{
                check = false;
            }
        }

        input = "INSERT INTO publishers"
        + "(pub_name, pub_address, pub_phone, pub_email) VALUES"
        + "(?,?,?,?)";

        pstmt = conn.prepareStatement(input);
        pstmt.setString(1, pn);
        pstmt.setString(2, pa);
        pstmt.setString(3, pp);
        pstmt.setString(4, pe);

        pstmt.executeUpdate();

        input = "update books"
```

```
        + " set pub_name = ?"
        + " where pub_name = ?";

        pstmt = conn.prepareStatement(input);
        pstmt.setString(1, pn);
        pstmt.setString(2, pu);

        pstmt.executeUpdate();

        rs = stmt.executeQuery("SELECT * FROM books");
    }
    /**
     * checks the database's table if a data already exist
     * @param table
     * @param col
     * @param input
     * @return check
     * @throws SQLException
     */
    public static boolean checking(String table, int col, String input) throws SQLException{
        ArrayList<String> arr = new ArrayList<String>();
        PreparedStatement pstmt = null;
        boolean check = false;
        String prepare, s = "";

        prepare = "select * from " + table;
        pstmt = conn.prepareStatement(prepare);
        rs = pstmt.executeQuery();
        ResultSetMetaData rsmd = rs.getMetaData();

        while(rs.next()){
            s = dispNull(rs.getString(rsmd.getColumnName(col)));
            arr.add(s);
        }

        if(arr.contains(input)){
            check = true;
        }else{
            check = false;
        }

        return check;
    }
```

```
   /**
* Takes the input string and outputs "N/A" if the string is empty or null.
* @param input The string to be mapped.
* @return  Either the input string or "N/A" as appropriate.
*/
   public static String dispNull (String input) {
      //because of short circuiting, if it's null, it never checks the length.
      if (input == null || input.length() == 0)
         return "NULL";
      else
         return input;
   }
   /**
    * checks the input of the user
    * @param low, lowest input choice
    * @param high, highest input choice
    * @return the integer
    */
   public static int checkInt( int low, int high ) {
      Scanner in = new Scanner(System.in);
      boolean valid = false;
      int validNum = 0;

      while( !valid ) {
            if(in.hasNextInt()) {
                  validNum = in.nextInt();
                  if( validNum >= low && validNum <= high ){
                        valid = true;
                  }
                  else{
                        System.out.println("**invalid input**");
                  }
            }
            else{
                  //clear buffer of junk input
                  in.next();
                  System.out.println("**invalid input**");
            }
      }
      return validNum;
   }
   /**
    * checks if the string is an integer
```

```
     * @param s, the string
     * @return true or false
     */
    public static boolean isInteger(String s) {
        return isInteger(s,10);
    }
    public static boolean isInteger(String s, int radix) {
        if(s.isEmpty()) return false;
        for(int i = 0; i < s.length(); i++) {
            if(i == 0 && s.charAt(i) == '-') {
                if(s.length() == 1) return false;
                else continue;
            }
            if(Character.digit(s.charAt(i),radix) < 0) return false;
        }
        return true;
    }
}//end FirstExample}
```

DDL-

```
create table writing_groups
(
   group_name varchar(30) not null,
   head_writer varchar(30) not null,
   year_formed char(4),
   subject varchar(20),
   constraint pk_writing_groups primary key (group_name)
);

create table publishers
(
   pub_name varchar(30) not null,
   pub_address varchar(50),
   pub_phone varchar(15),
   pub_email varchar(40),
   constraint pk_publishers primary key(pub_name)
);

create table books
(
   group_name varchar(30) not null,
   book_title varchar(30) not null,
   pub_name varchar(30) not null,
   year_published char(4) not null,
   pages integer not null,
   constraint pk_books primary key(group_name, book_title),
   constraint ck_books unique (book_title, pub_name),
   constraint books_writing_groups_fk
      foreign key(group_name)
      references writing_groups(group_name),
   constraint books_publishers_fk
      foreign key(pub_name)
      references publishers(pub_name)
);

insert into writing_groups values
('Winterspring', 'Gray Fullbuster', '2012', 'Cyrogenics'),
('Hotsprings', 'Natsu Dragneel', '2012', 'Pyrogenics'),
('Mystics', 'Lucy Heartfilia', '2013', 'Spiritology'),
('Metallics', 'Erza Scarlet', null, 'Metallurgy'),
('One Piece', 'Monkey D. Luffy', '2016', null),
('Logstone', 'Kirito Kirigaya', '2023', 'Sci-Fi');
```

```
insert into publishers values
('Stormwind Press', '3415 Stormwind Blvd.', '808-608-5709', 'stormwind@press.com'),
('Darnassus Press', '819 Darnassus St.', '707-584-1857', 'darnassus@press.com'),
('Undercity Press', '666 Undercity Ave.', '606-338-0107', null),
('Dalaran Press', '100 Sky St.', null, 'dalaran@press.com');

insert into books values
('Winterspring', 'Vanilla', 'Darnassus Press', '2012', 2344),
('Hotsprings', 'Burning Legion', 'Stormwind Press', '2013', 3891),
('Mystics', 'Wrath of The Lich King', 'Undercity Press', '2013', 1743),
('Metallics', 'Cataclysm', 'Dalaran Press', '2014', 598),
('Logstone', 'Pandarean', 'Dalaran Press', '2015', 819),
('Hotsprings', 'Legion', 'Undercity Press', '2016', 1293);
```

Example of Output-

```
run:
Name of the database (not the user account): jdbc_
Connecting to database...

1. List all writing groups
2. List all the data for a group specified by the user
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
10. Exit
Please choose a command:
1

Creating statement...
GROUP_NAME       HEAD_WRITER          YEAR_FORMED      SUBJECT
Winterspring      Gray Fullbuster       2012            Cyrogenics
Hotsprings       Natsu Dragneel        2012            Pyrogenics
Mystics          Lucy Heartfilia       2013            Spiritology
Metallics        Erza Scarlet          NULL            Metallurgy
One Piece         Monkey D. Luffy       2016            NULL
Logstone          Kirito Kirigaya       2023            Sci-Fi

1. List all writing groups
2. List all the data for a group specified by the user
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
10. Exit
Please choose a command:
11
```

```
**invalid input**
2
Please enter the writing group name:
Winterspring

Creating statement...
GROUP_NAME       HEAD_WRITER         YEAR_FORMED       SUBJECT
Winterspring      Gray Fullbuster       2012             Cyrogenics

1. List all writing groups
2. List all the data for a group specified by the user
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
10. Exit
Please choose a command:
3

Creating statement...
PUB_NAME          PUB_ADDRESS           PUB_PHONE         PUB_EMAIL
Stormwind Press    3415 Stormwind Blvd.    808-608-5709      stormwind@press.com
Darnassus Press    819 Darnassus St.      707-584-1857      darnassus@press.com
Undercity Press    666 Undercity Ave.     606-338-0107      NULL
Dalaran Press      100 Sky St.           NULL             dalaran@press.com

1. List all writing groups
2. List all the data for a group specified by the user
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
10. Exit
Please choose a command:
4
```

```
Please enter the publisher name:
Stormwind Press

Creating statement...
PUB_NAME          PUB_ADDRESS            PUB_PHONE          PUB_EMAIL
Stormwind Press     3415 Stormwind Blvd.     808-608-5709       stormwind@press.com

1. List all writing groups
2. List all the data for a group specified by the user
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
10. Exit
Please choose a command:
5

Creating statement...
GROUP_NAME        BOOK_TITLE                  PUB_NAME          YEAR_PUBLISHED
PAGES
Winterspring       Vanilla                   Darnassus Press     2012              2344
Hotsprings         Burning Legion              Stormwind Press     2013              3891
Mystics           Wrath of The Lich King       Undercity Press     2013              1743
Metallics          Cataclysm                  Dalaran Press      2014              598
Logstone           Pandarean                  Dalaran Press      2015              819
Hotsprings         Legion                    Undercity Press     2016              1293
Winterspring       Greg's Life                 Darnassus Press     2016              2
Winterspring       Sarah's Life                Darnassus Press     2016              3

1. List all writing groups
2. List all the data for a group specified by the user
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
```

```
10. Exit
Please choose a command:
6
Please enter the group name:
Winterspring
Please enter the book title:
Greg's Life

Creating statement...
GROUP_NAME         BOOK_TITLE                  PUB_NAME           YEAR_PUBLISHED
PAGES
Winterspring       Greg's Life                 Darnassus Press     2016              2

1. List all writing groups
2. List all the data for a group specified by the user
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
10. Exit
Please choose a command:
7
Adding book..
Please enter group name:
CSULB
**invalid writing group name: does not exist**
Please enter group name:
Mystics
Please enter book title:
Database FUNdamentals
Please enter publisher's name:
A to Z Publishing
**invalid publisher's name: does not exist**
Please enter publisher's name:
Undercity Press
Please enter year of publish:
2010
Please enter number of pages:
4
```

```
Creating statement...
GROUP_NAME        BOOK_TITLE                PUB_NAME          YEAR_PUBLISHED
PAGES
Winterspring      Vanilla                   Darnassus Press     2012            2344
Hotsprings        Burning Legion            Stormwind Press     2013            3891
Mystics           Wrath of The Lich King       Undercity Press     2013            1743
Metallics         Cataclysm                 Dalaran Press       2014            598
Logstone          Pandarean                 Dalaran Press       2015            819
Hotsprings        Legion                    Undercity Press     2016            1293
Winterspring      Greg's Life               Darnassus Press     2016            2
Winterspring      Sarah's Life              Darnassus Press     2016            3
Mystics           Database FUNdamentals        Undercity Press     2010            4

1. List all writing groups
2. List all the data for a group specified by the user
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
10. Exit
Please choose a command:
8
Adding publisher..
Please enter the publisher's name:
Undercity Press
**invalid publisher: already exist**
Adding publisher..
Please enter the publisher's name:
Undercity Company
Please enter the publisher's address title:
1004 Cloudview Way
Please enter the publisher's phone:
213-222-3456
Please enter the publisher's email
cloud@view.com
Please enter which publisher you want to update it on
Undercity Press
```

```
Creating statement...
GROUP_NAME        BOOK_TITLE                PUB_NAME          YEAR_PUBLISHED
PAGES
Winterspring      Vanilla                 Darnassus Press     2012             2344
Hotsprings        Burning Legion            Stormwind Press     2013             3891
Mystics          Wrath of The Lich King      Undercity Company   2013             1743
Metallics        Cataclysm                 Dalaran Press      2014             598
Logstone         Pandarean                  Dalaran Press      2015             819
Hotsprings        Legion                  Undercity Company   2016             1293
Winterspring      Greg's Life                Darnassus Press     2016             2
Winterspring      Sarah's Life               Darnassus Press     2016             3
Mystics          Database FUNdamentals        Undercity Company   2010             4

1. List all writing groups
2. List all the data for a group specified by the user
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
10. Exit
Please choose a command:
9
Please enter the book title ('c' to cancel):
Database FUNdamentals

Creating statement...
GROUP_NAME        BOOK_TITLE                PUB_NAME          YEAR_PUBLISHED
PAGES
Winterspring      Vanilla                 Darnassus Press     2012             2344
Hotsprings        Burning Legion            Stormwind Press     2013             3891
Mystics          Wrath of The Lich King      Undercity Company   2013             1743
Metallics        Cataclysm                 Dalaran Press      2014             598
Logstone         Pandarean                  Dalaran Press      2015             819
Hotsprings        Legion                  Undercity Company   2016             1293
Winterspring      Greg's Life                Darnassus Press     2016             2
Winterspring      Sarah's Life               Darnassus Press     2016             3

1. List all writing groups
2. List all the data for a group specified by the user
```

```
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
10. Exit
Please choose a command:
9
Please enter the book title ('c' to cancel):
c

Creating statement...
GROUP_NAME       BOOK_TITLE               PUB_NAME         YEAR_PUBLISHED
PAGES

1. List all writing groups
2. List all the data for a group specified by the user
3. List all publishers
4. List all the data for a publisher specified by the user
5. List all books titles
6. List all the data for a book specified by the user
7. Insert a new book
8. Insert a new publisher and update all book published by one publisher to be published by the
new publisher
9. Remove a book specified by the user
10. Exit
Please choose a command:
10

Goodbye!
BUILD SUCCESSFUL (total time: 4 minutes 23 seconds)
```