**CORE JAVA ASSIGNMENT 3**

**BooksMenuUI.java**

**package** com.books.app.ui;

**import** java.time.DateTimeException;

**import** java.time.LocalDate;

**import** java.time.format.DateTimeFormatter;

**import** java.util.ArrayList;

**import** java.util.Comparator;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.Scanner;

**import** java.util.Set;

**import** java.util.stream.Collectors;

**import** com.books.app.exception.BookAppException;

**import** com.books.app.model.Book;

**import** com.books.app.model.BooksAppMenu;

**import** com.books.app.model.Subject;

**import** com.books.app.service.BookAppServiceImpl;

**import** com.books.app.service.IBookAppService;

**public** **class** BooksMenuUI {

**static** IBookAppService *bookAppService*;

**static** Scanner *scan*;

**static** DateTimeFormatter *dtFormater*;

**public** **static** **void** main(String[] args) {

**try** {

*bookAppService* = **new** BookAppServiceImpl();

} **catch** (BookAppException exp) {

System.***err***.println(exp.getMessage());

System.*exit*(0);

}

*scan* = **new** Scanner(System.***in***);

*dtFormater* = DateTimeFormatter.*ofPattern*("dd-MM-yyyy");

BooksAppMenu menu = **null**;

**while** (menu != BooksAppMenu.***QUIT***) {

System.***out***.println("===========================");

System.***out***.println("Books App Menu Options");

System.***out***.println("===========================");

**for** (BooksAppMenu m : BooksAppMenu.*values*()) {

System.***out***.println(m.ordinal() + "\t" + m.name());

}

System.***out***.print("Select the action: ");

**int** action = -1;

**if** (*scan*.hasNextInt())

action = *scan*.nextInt();

**else** {

*scan*.next();

System.***out***.println("Please choose an action displayed");

**continue**;

}

**if** (action < 0 || action >= BooksAppMenu.*values*().length) {

System.***out***.println("Invalid Option");

} **else** {

menu = BooksAppMenu.*values*()[action];

**switch** (menu) {

**case** ***VIEW***:

*doView*();

**break**;

**case** ***ADD\_BOOK***:

*doAddBook*();

**break**;

**case** ***ADD\_SUBJECT***:

*doAddSubject*();

**break**;

**case** ***DELETE\_BOOK***:

*doDeleteBook*();

**break**;

**case** ***DELETE\_SUBJECT***:

*doDeleteSubject*();

**break**;

**case** ***SEARCH\_BOOK***:

*doSearchBook*();

**break**;

**case** ***SEARCH\_SUBJECT***:

*doSearchSubject*();

**break**;

**case** ***SORTBY\_SUBJECT\_TITLE***:

*doSortBySubjectTitle*();

**break**;

**case** ***SORTBY\_BOOK\_TITLE***:

*doSortByBookTitle*();

**break**;

**case** ***SORTBY\_BOOK\_PUBLISHED\_DATE***:

*doSortByBookPublishedDate*();

**break**;

**case** ***QUIT***:

System.***out***.println("Thanks for visiting!!");

**break**;

}

}

}

*scan*.close();

}

**private** **static** **void** doSortByBookPublishedDate() {

**try** {

Map<String, Subject> subjects = *bookAppService*.view();

List<Book> list = **new** ArrayList<Book>();

**if** (subjects != **null** && !subjects.isEmpty()) {

System.***out***.format("%-10s | %10s | %10s | %10s | %10s\n","Subject","BookTitle","Price","Volume","PublishedDate");

System.***out***.println("------------------------------------------------------------------");

*getAllBooksList*(subjects, list);

List<Book> sList = list.stream().sorted(Comparator.*comparing*(Book::getPublistDt)).collect(Collectors.*toList*());

sList.forEach(e -> System.***out***.format("%-10s | %10s | %10.2f | %10d | %10s\n",e.getSubject().getSubtitle(),e.getTitle(),e.getPrice(),e.getVolume(),e.getPublistDt()));

} **else** {

System.***out***.println("No books to view !!");

}

} **catch** (BookAppException e) {

e.printStackTrace();

}

}

**private** **static** **void** doSortByBookTitle() {

**try** {

Map<String, Subject> subjects = *bookAppService*.view();

List<Book> list = **new** ArrayList<Book>();

**if** (subjects != **null** && !subjects.isEmpty()) {

System.***out***.format("%-10s | %10s | %10s | %10s | %10s\n","Subject","BookTitle","Price","Volume","PublishedDate");

System.***out***.println("------------------------------------------------------------------");

*getAllBooksList*(subjects, list);

List<Book> sList = list.stream().sorted(Comparator.*comparing*(Book::getTitle)).collect(Collectors.*toList*());

sList.forEach(e -> System.***out***.format("%-10s | %10s | %10.2f | %10d | %10s\n",e.getSubject().getSubtitle(),e.getTitle(),e.getPrice(),e.getVolume(),e.getPublistDt()));

} **else** {

System.***out***.println("No books to view !!");

}

} **catch** (BookAppException e) {

e.printStackTrace();

}

}

**private** **static** **void** doSortBySubjectTitle() {

**try** {

Map<String, Subject> subjects = *bookAppService*.view();

List<Book> list = **new** ArrayList<Book>();

**if** (subjects != **null** && !subjects.isEmpty()) {

System.***out***.format("%-10s | %10s | %10s | %10s | %10s\n","Subject","BookTitle","Price","Volume","PublishedDate");

System.***out***.println("------------------------------------------------------------------");

*getAllBooksList*(subjects, list);

List<Book> sList = list.stream().collect(Collectors.*toList*());

sList.forEach(e -> System.***out***.format("%-10s | %10s | %10.2f | %10d | %10s\n",e.getSubject().getSubtitle(),e.getTitle(),e.getPrice(),e.getVolume(),e.getPublistDt()));

} **else** {

System.***out***.println("No books to view !!");

}

} **catch** (BookAppException e) {

e.printStackTrace();

}

}

**private** **static** **void** getAllBooksList(Map<String, Subject> subjects, List<Book> list) {

**for** (Map.Entry<String, Subject> entry : subjects.entrySet()) {

Subject subj = entry.getValue();

**if** (subj != **null**) {

Set<Book> books = subj.getReferences();

**if** (books != **null** && !books.isEmpty()) {

**for** (Book book : books) {

Book sortBook = **new** Book();

**if** (book != **null**) {

sortBook.setBookId(book.getBookId());

sortBook.setTitle(book.getTitle());

sortBook.setPrice(book.getPrice());

sortBook.setVolume(book.getVolume());

sortBook.setPublistDt(book.getPublistDt());

sortBook.setSubject(book.getSubject());

list.add(sortBook);

}

}

}

}

}

}

**private** **static** **void** doSearchSubject() {

**try** {

System.***out***.print("Subject Title: ");

String subtitle = *scan*.next();

Subject subj = *bookAppService*.searchSubject(subtitle);

**if** (subj != **null**) {

System.***out***.println("Subject Found : " + subj.getSubtitle());

System.***out***.println("Available books are : ");

Set<Book> books = subj.getReferences();

**if** (**null** != books) {

**for** (Book book : books) {

**if** (book != **null**) {

System.***out***.println("-------------- " + book.getTitle());

}

}

}

}

} **catch** (BookAppException exp) {

System.***out***.println(exp.getMessage());

}

}

**private** **static** **void** doSearchBook() {

**try** {

System.***out***.print("Book Title: ");

String booktitle = *scan*.next();

Book book = *bookAppService*.search(booktitle);

**if** (book != **null**) {

System.***out***.println("Book Found : " + book.getTitle());

System.***out***.println("Title ----- " + book.getTitle());

System.***out***.println("Price ----- " + book.getPrice());

System.***out***.println("Volume ----- " + book.getVolume());

System.***out***.println("Published Date ----- " + book.getPublistDt());

}

} **catch** (BookAppException exp) {

System.***out***.println(exp.getMessage());

}

}

**private** **static** **void** doDeleteSubject() {

**try** {

System.***out***.print("Subject Title: ");

String subtitle = *scan*.next();

**boolean** delsubj = *bookAppService*.deleteSubject(subtitle);

**if** (delsubj) {

System.***out***.println("Subject deleted successfully");

} **else** {

System.***out***.println("Subject cannot be deleted");

}

} **catch** (BookAppException exp) {

System.***out***.println(exp.getMessage());

}

}

**private** **static** **void** doDeleteBook() {

**try** {

System.***out***.print("Book Title: ");

String booktitle = *scan*.next();

**boolean** delbook = *bookAppService*.delete(booktitle);

**if** (delbook) {

System.***out***.println("Book deleted successfully");

} **else** {

System.***out***.println("Book cannot be deleted");

}

} **catch** (BookAppException exp) {

System.***out***.println(exp.getMessage());

}

}

**private** **static** **void** doAddSubject() {

String newSubject = **null**;

**try** {

Subject sub = **new** Subject();

System.***out***.print("Enter Subject ID: ");

sub.setSubjectId(Long.*parseLong*(*scan*.next()));

System.***out***.print("Enter Subject Title: ");

sub.setSubtitle(*scan*.next());

System.***out***.print("Enter Duration in hrs: ");

sub.setDurationInHours(Integer.*parseInt*(*scan*.next()));

**if** (*bookAppService*.searchSubject(sub.getSubtitle()) == **null**) {

newSubject = *bookAppService*.addSubject(sub);

}

System.***out***.println("Subject " + newSubject + " is added");

} **catch** (BookAppException exp) {

System.***out***.println(exp.getMessage());

}

}

**private** **static** **void** doView() {

**try** {

Map<String, Subject> subjects = *bookAppService*.view();

**if** (subjects != **null** && !subjects.isEmpty()) {

**for** (Map.Entry<String, Subject> entry : subjects.entrySet()) {

Subject subj = entry.getValue();

**if** (subj != **null**) {

System.***out***.println("Subject : " + subj.getSubtitle());

System.***out***.println("---- Books : ");

Set<Book> books = subj.getReferences();

**if** (books != **null** && !books.isEmpty()) {

**for** (Book book : books) {

**if** (book != **null**) {

System.***out***.println("-------- " + book.getTitle());

}

}

}

}

}

} **else** {

System.***out***.println("No books to view !!");

}

} **catch** (BookAppException e) {

e.printStackTrace();

}

}

**private** **static** **void** doAddBook() {

**try** {

Subject sub = **new** Subject();

Book book = **new** Book();

System.***out***.print("Enter Book Id: ");

book.setBookId(Long.*parseLong*(*scan*.next()));

System.***out***.print("Enter Book Title: ");

book.setTitle(*scan*.next());

System.***out***.print("Enter Price: ");

book.setPrice(Double.*parseDouble*(*scan*.next()));

System.***out***.print("Enter Volume: ");

book.setVolume(*scan*.nextInt());

System.***out***.print("Enter PublishDate(dd-MM-yyyy): ");

String pubDtStr = *scan*.next();

**try** {

book.setPublistDt(LocalDate.*parse*(pubDtStr, *dtFormater*));

} **catch** (DateTimeException exp) {

**throw** **new** BookAppException("Date must be in the format day(dd)-month(MM)-year(yyyy)");

}

System.***out***.print("Enter Subject Title: ");

sub.setSubtitle(*scan*.next());

book.setSubject(sub);

String bookTitle = *bookAppService*.add(book, sub.getSubtitle());

System.***out***.println("Book " + bookTitle + " is added");

} **catch** (BookAppException exp) {

System.***out***.println(exp.getMessage());

}

}

}

**IBookAppService.java**

**package** com.books.app.service;

**import** java.util.Map;

**import** com.books.app.exception.BookAppException;

**import** com.books.app.model.Book;

**import** com.books.app.model.Subject;

**public** **interface** IBookAppService {

String add(Book book, String subtitle) **throws** BookAppException;

String addSubject(Subject subject) **throws** BookAppException;

**boolean** delete(String booktitle) **throws** BookAppException;

**boolean** deleteSubject(String subtitle) **throws** BookAppException;

Book search(String booktitle) **throws** BookAppException;

Subject searchSubject(String subtitle) **throws** BookAppException;

Map<String, Subject> view() **throws** BookAppException;

}

**BookAppServiceImpl.java**

**package** com.books.app.service;

**import** java.util.Map;

**import** com.books.app.dao.BookAppJDBCDAOImpl;

**import** com.books.app.dao.IBookAppDAO;

**import** com.books.app.exception.BookAppException;

**import** com.books.app.model.Book;

**import** com.books.app.model.Subject;

**public** **class** BookAppServiceImpl **implements** IBookAppService {

**private** IBookAppDAO bookAppDao;

**public** BookAppServiceImpl() **throws** BookAppException {

bookAppDao = **new** BookAppJDBCDAOImpl();

}

**public** IBookAppDAO getBookAppDAO() {

**return** bookAppDao;

}

@Override

**public** String add(Book book, String subtitle) **throws** BookAppException {

**return** bookAppDao.add(book, subtitle);

}

@Override

**public** String addSubject(Subject subject) **throws** BookAppException {

**return** bookAppDao.addSubject(subject);

}

@Override

**public** **boolean** delete(String booktitle) **throws** BookAppException {

**return** bookAppDao.delete(booktitle);

}

@Override

**public** **boolean** deleteSubject(String subtitle) **throws** BookAppException {

**return** bookAppDao.deleteSubject(subtitle);

}

@Override

**public** Book search(String booktitle) **throws** BookAppException {

**return** bookAppDao.search(booktitle);

}

@Override

**public** Subject searchSubject(String subtitle) **throws** BookAppException {

**return** bookAppDao.searchSubject(subtitle);

}

@Override

**public** Map<String, Subject> view() **throws** BookAppException{

**return** bookAppDao.view();

}

}

**IBookAppDAO.java**

**package** com.books.app.dao;

**import** java.util.Map;

**import** com.books.app.exception.BookAppException;

**import** com.books.app.model.Book;

**import** com.books.app.model.Subject;

**public** **interface** IBookAppDAO {

String add(Book book, String subtitle) **throws** BookAppException;

String addSubject(Subject subject) **throws** BookAppException;

**boolean** delete(String booktitle) **throws** BookAppException;

**boolean** deleteSubject(String subtitle) **throws** BookAppException;

Book search(String booktitle) **throws** BookAppException;

Subject searchSubject(String subtitle) **throws** BookAppException;

Map<String, Subject> view() **throws** BookAppException;

}

**BookAppJDBCDAOImpl.java**

**package** com.books.app.dao;

**import** java.io.IOException;

**import** java.sql.Connection;

**import** java.sql.Date;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.HashSet;

**import** java.util.Map;

**import** java.util.Set;

**import** java.util.TreeMap;

**import** com.books.app.exception.BookAppException;

**import** com.books.app.model.Book;

**import** com.books.app.model.Subject;

**import** com.books.app.util.ConnectionProvider;

**public** **class** BookAppJDBCDAOImpl **implements** IBookAppDAO {

ConnectionProvider conProvider;

**public** BookAppJDBCDAOImpl() **throws** BookAppException {

**try** {

conProvider = ConnectionProvider.*getInstance*();

} **catch** (ClassNotFoundException | IOException exp) {

**throw** **new** BookAppException("Database is not reachable");

}

}

@Override

**public** String add(Book book, String subtitle) **throws** BookAppException {

String bookTitle = **null**;

String subjectTitle = **null**;

**if** (book.getSubject() != **null**) {

Map<String, Subject> subjects = view();

**if** (subjects != **null** && !subjects.isEmpty() && subjects.containsKey(subtitle)) {

subjectTitle = subtitle;

} **else** {

subjectTitle = addSubject(book.getSubject());

}

**try** (Connection con = conProvider.getConnection();

PreparedStatement pInsert = con.prepareStatement(IQueryMapper.***ADD\_BOOK\_QRY***)) {

pInsert.setLong(1, book.getBookId());

pInsert.setString(2, book.getTitle());

pInsert.setDouble(3, book.getPrice());

pInsert.setInt(4, book.getVolume());

pInsert.setDate(5, Date.*valueOf*(book.getPublistDt()));

pInsert.setString(6, subjectTitle);

**int** rowCount = pInsert.executeUpdate();

**if** (rowCount > 0) {

bookTitle = book.getTitle();

}

} **catch** (SQLException exp) {

**throw** **new** BookAppException("Book is not inserted");

}

}

**return** bookTitle;

}

@Override

**public** String addSubject(Subject subject) **throws** BookAppException {

String subjectTitle = **null**;

**if** (subject != **null**) {

**try** (Connection con = conProvider.getConnection();

PreparedStatement pInsertSub = con.prepareStatement(IQueryMapper.***ADD\_SUBJECT\_QRY***)) {

pInsertSub.setLong(1, subject.getSubjectId());

pInsertSub.setString(2, subject.getSubtitle());

pInsertSub.setInt(3, subject.getDurationInHours());

**int** subRow = pInsertSub.executeUpdate();

**if** (subRow > 0) {

subjectTitle = subject.getSubtitle();

}

} **catch** (SQLException exp) {

**throw** **new** BookAppException("Subject is not inserted");

}

}

**return** subjectTitle;

}

@Override

**public** **boolean** deleteSubject(String subtitle) **throws** BookAppException {

**try** (Connection con = conProvider.getConnection();

PreparedStatement pDelete = con.prepareStatement(IQueryMapper.***DEL\_SUBJECT\_QRY***)) {

pDelete.setString(1, subtitle);

**int** count = pDelete.executeUpdate();

**if**(count >0) {

**return** **true**;

}

} **catch** (SQLException exp) {

**throw** **new** BookAppException("Book not deleted");

}

**return** **false**;

}

@Override

**public** Book search(String booktitle) **throws** BookAppException {

Book book = **new** Book();

**try** (Connection con = conProvider.getConnection();

PreparedStatement pSelect = con.prepareStatement(IQueryMapper.***GET\_BOOK\_QRY***)) {

pSelect.setString(1, booktitle);

ResultSet rs = pSelect.executeQuery();

**while** (rs.next()) {

book.setBookId(rs.getLong("bookid"));

book.setTitle(rs.getString("title"));

book.setPrice(rs.getDouble("price"));

book.setVolume(rs.getInt("vol"));

book.setPublistDt(rs.getDate("pdate").toLocalDate());

Subject subject = **new** Subject();

subject.setSubtitle(rs.getString("stitle"));

book.setSubject(subject);

}

} **catch** (SQLException exp) {

**throw** **new** BookAppException("Book not retrieved");

}

**return** book;

}

@Override

**public** Subject searchSubject(String subtitle) **throws** BookAppException {

Map<String, Subject> subjects = view();

**if** (subjects != **null** && !subjects.isEmpty() && subjects.containsKey(subtitle)) {

**return** subjects.get(subtitle);

}

**return** **null**;

}

@Override

**public** Map<String, Subject> view() **throws** BookAppException {

Map<String, Subject> subjects = **new** TreeMap<String, Subject>();

getAllSubjects(subjects);

getAllBooks(subjects);

**return** subjects;

}

**private** **void** getAllBooks(Map<String, Subject> subjects) **throws** BookAppException {

**try** (Connection con = conProvider.getConnection();

PreparedStatement pSelect = con.prepareStatement(IQueryMapper.***GET\_ALL\_BOOKS\_QRY***)) {

ResultSet rs = pSelect.executeQuery();

**while** (rs.next()) {

Book book = **new** Book();

book.setBookId(rs.getLong("bookid"));

book.setTitle(rs.getString("title"));

book.setPrice(rs.getDouble("price"));

book.setVolume(rs.getInt("vol"));

book.setPublistDt(rs.getDate("pdate").toLocalDate());

Subject subject = **new** Subject();

subject.setSubtitle(rs.getString("stitle"));

book.setSubject(subject);

**if** (subjects != **null** && !subjects.isEmpty() && subjects.containsKey(subject.getSubtitle())) {

Set<Book> books = subjects.get(subject.getSubtitle()).getReferences();

**if** (books != **null** ) {

subjects.get(subject.getSubtitle()).getReferences().add(book);

}**else** {

Set<Book> nbooks = **new** HashSet<Book>();

nbooks.add(book);

subjects.get(subject.getSubtitle()).setReferences(nbooks);

}

}

}

} **catch** (SQLException exp) {

**throw** **new** BookAppException("Books not retrieved");

}

}

**private** **void** getAllSubjects(Map<String, Subject> subjects) **throws** BookAppException {

**try** (Connection con = conProvider.getConnection();

PreparedStatement pSelect = con.prepareStatement(IQueryMapper.***GET\_ALL\_SUBJECTS\_QRY***)) {

ResultSet rs = pSelect.executeQuery();

**while** (rs.next()) {

Subject subject = **new** Subject();

subject.setSubjectId(rs.getLong("subjectid"));

subject.setSubtitle(rs.getString("subtitle"));

subject.setDurationInHours(rs.getInt("durationInHours"));

subjects.put(subject.getSubtitle(), subject);

}

} **catch** (SQLException exp) {

**throw** **new** BookAppException("Subjects not retrieved");

}

}

@Override

**public** **boolean** delete(String booktitle) **throws** BookAppException {

**try** (Connection con = conProvider.getConnection();

PreparedStatement pDelete = con.prepareStatement(IQueryMapper.***DEL\_BOOK\_QRY***)) {

pDelete.setString(1, booktitle);

**int** count = pDelete.executeUpdate();

**if**(count >0) {

**return** **true**;

}

} **catch** (SQLException exp) {

**throw** **new** BookAppException("Book not deleted");

}

**return** **false**;

}

}

**BookAppException.java**

**package** com.books.app.exception;

@SuppressWarnings("serial")

**public** **class** BookAppException **extends** Exception {

**public** BookAppException(String errMsg) {

**super**(errMsg);

}

}

**BooksAppMenu.java**

**package** com.books.app.model;

**public** **enum** BooksAppMenu {

***VIEW***,***ADD\_BOOK***,***ADD\_SUBJECT***,***DELETE\_BOOK***,***DELETE\_SUBJECT***,***SEARCH\_BOOK***,***SEARCH\_SUBJECT***,***SORTBY\_SUBJECT\_TITLE***,***SORTBY\_BOOK\_TITLE***,***SORTBY\_BOOK\_PUBLISHED\_DATE***,***QUIT***;

}

**Subject.java**

**package** com.books.app.model;

**import** java.io.Serializable;

**import** java.util.Set;

**public** **class** Subject **implements** Serializable {

**private** **long** subjectId;

**private** String subtitle;

**private** **int** durationInHours;

**private** Set<Book> references;

**public** long getSubjectId() {

**return** subjectId;

}

**public** void setSubjectId(long subjectId) {

**this**.subjectId = subjectId;

}

**public** String getSubtitle() {

**return** subtitle;

}

**public** void setSubtitle(String subtitle) {

**this**.subtitle = subtitle;

}

**public** int getDurationInHours() {

**return** durationInHours;

}

**public** void setDurationInHours(int durationInHours) {

**this**.durationInHours = durationInHours;

}

**public** Set<Book> getReferences() {

**return** references;

}

**public** void setReferences(Set<Book> references) {

**this**.references = references;

}

}

**Book.java**

**package** com.books.app.model;

**import** java.io.Serializable;

**import** java.time.LocalDate;

**public** **class** Book **implements** Serializable {

**private** **long** **bookId**;

**private** String title;

**private** **double** price;

**private** **int** volume;

**private** LocalDate publistDt;

**private** Subject subject;

**public** Subject getSubject() {

**return** subject;

}

**public** **void** setSubject(Subject subject) {

**this**.subject = subject;

}

**public** **long** getBookId() {

**return** bookId;

}

**public** **void** setBookId(**long** bookId) {

**this**.bookId = bookId;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

**public** **int** getVolume() {

**return** volume;

}

**public** **void** setVolume(**int** volume) {

**this**.volume = volume;

}

**public** LocalDate getPublistDt() {

**return** publistDt;

}

**public** **void** setPublistDt(LocalDate publistDt) {

**this**.publistDt = publistDt;

}

}

**jdbc.properties**

db.driver=com.mysql.jdbc.Driver

db.unm=root

db.pwd=root

db.url=jdbc:mysql://localhost:3306/tmdb

**dbscript.sql**

CREATE TABLE books(

bookid CHAR(4) NOT NULL,

title VARCHAR(20) NOT NULL,

price double(7,2) NOT NULL,

vol int(2) NOT NULL,

pdate date NOT NULL,

stitle VARCHAR(20) NOT NULL,

index pn\_subject\_index(stitle),

foreign key(stitle) references subjects(subtitle) on delete cascade,

primary key(bookid)

);

CREATE TABLE subjects(

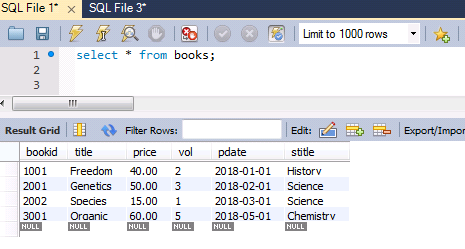
subjectid CHAR(4) NOT NULL,

subtitle VARCHAR(20) PRIMARY KEY,

durationInHoursparent\_task\_table int(2)

);

**RESULTS :**



===========================

Books App Menu Options

===========================

0 VIEW

1 ADD\_BOOK

2 ADD\_SUBJECT

3 DELETE\_BOOK

4 DELETE\_SUBJECT

5 SEARCH\_BOOK

6 SEARCH\_SUBJECT

7 SORTBY\_SUBJECT\_TITLE

8 SORTBY\_BOOK\_TITLE

9 SORTBY\_BOOK\_PUBLISHED\_DATE

10 QUIT

**Select the action: 7**

**Subject | BookTitle | Price | Volume | PublishedDate**

**--------------------------------------------------------------------------------------**

**Chemistry | Organic | 60.00 | 5 | 2018-05-01**

**History | Freedom | 40.00 | 2 | 2018-01-01**

**Science | Species | 15.00 | 1 | 2018-03-01**

**Science | Genetics | 50.00 | 3 | 2018-02-01**

===========================

Books App Menu Options

===========================

0 VIEW

1 ADD\_BOOK

2 ADD\_SUBJECT

3 DELETE\_BOOK

4 DELETE\_SUBJECT

5 SEARCH\_BOOK

6 SEARCH\_SUBJECT

7 SORTBY\_SUBJECT\_TITLE

8 SORTBY\_BOOK\_TITLE

9 SORTBY\_BOOK\_PUBLISHED\_DATE

10 QUIT

**Select the action: 8**

**Subject | BookTitle | Price | Volume | PublishedDate**

**------------------------------------------------------------------**

**History | Freedom | 40.00 | 2 | 2018-01-01**

**Science | Genetics | 50.00 | 3 | 2018-02-01**

**Chemistry | Organic | 60.00 | 5 | 2018-05-01**

**Science | Species | 15.00 | 1 | 2018-03-01**

===========================

Books App Menu Options

===========================

0 VIEW

1 ADD\_BOOK

2 ADD\_SUBJECT

3 DELETE\_BOOK

4 DELETE\_SUBJECT

5 SEARCH\_BOOK

6 SEARCH\_SUBJECT

7 SORTBY\_SUBJECT\_TITLE

8 SORTBY\_BOOK\_TITLE

9 SORTBY\_BOOK\_PUBLISHED\_DATE

10 QUIT

**Select the action: 9**

**Subject | BookTitle | Price | Volume | PublishedDate**

**------------------------------------------------------------------**

**History | Freedom | 40.00 | 2 | 2018-01-01**

**Science | Genetics | 50.00 | 3 | 2018-02-01**

**Science | Species | 15.00 | 1 | 2018-03-01**

**Chemistry | Organic | 60.00 | 5 | 2018-05-01**