# Book.java

* **package** com.fsd.corejava.entities;
* **import** java.sql.Date;
* /\*\*
* \* **@author** Sudhir Kumar Thakur
* \*
* \*/
* **public** **class** Book {
* **private** **long** bookId;
* **private** String title;
* **private** **double** price;
* **private** Integer volume;
* **private** Date publishDate;
* **private** **long** subjectId;
* /\*\*
* \* **@return** the subjectId
* \*/
* **public** **long** getSubjectId() {
* **return** subjectId;
* }
* /\*\*
* \* **@param** subjectId the subjectId to set
* \*/
* **public** **void** setSubjectId(**long** subjectId) {
* **this**.subjectId = subjectId;
* }
* **public** Book() {
* };
* **public** Book(**long** bookId, String title, **double** price, Integer volume, Date publishDate,**long** subjectId) {
* **super**();
* **this**.bookId = bookId;
* **this**.title = title;
* **this**.price = price;
* **this**.volume = volume;
* **this**.publishDate = publishDate;
* **this**.subjectId = subjectId;
* }
* /\*\*
* \* **@return** the bookId
* \*/
* **public** **long** getBookId() {
* **return** bookId;
* }
* /\*\*
* \* **@param** bookId the bookId to set
* \*/
* **public** **void** setBookId(**long** bookId) {
* **this**.bookId = bookId;
* }
* /\*\*
* \* **@return** the title
* \*/
* **public** String getTitle() {
* **return** title;
* }
* /\*\*
* \* **@param** title the title to set
* \*/
* **public** **void** setTitle(String title) {
* **this**.title = title;
* }
* /\*\*
* \* **@return** the price
* \*/
* **public** **double** getPrice() {
* **return** price;
* }
* /\*\*
* \* **@param** price the price to set
* \*/
* **public** **void** setPrice(**double** price) {
* **this**.price = price;
* }
* /\*\*
* \* **@return** the volume
* \*/
* **public** Integer getVolume() {
* **return** volume;
* }
* /\*\*
* \* **@param** volume the volume to set
* \*/
* **public** **void** setVolume(Integer volume) {
* **this**.volume = volume;
* }
* /\*\*
* \* **@return** the publishDate
* \*/
* **public** Date getPublishDate() {
* **return** publishDate;
* }
* /\*\*
* \* **@param** publishDate the publishDate to set
* \*/
* **public** **void** setPublishDate(Date publishDate) {
* **this**.publishDate = publishDate;
* }
* /\*
* \* (non-Javadoc)
* \*
* \* @see java.lang.Object#toString()
* \*/
* @Override
* **public** String toString() {
* **return** "Book [bookId=" + bookId + ", title=" + title + ", price=" + price + ", volume=" + volume
* + ", publishDate=" + publishDate + "]";
* }
* }

# Subject.java

* **package** com.fsd.corejava.entities;
* **import** java.util.Set;
* /\*\*
* \* **@author** Sudhir Kumar Thakur
* \*
* \*/
* **public** **class** Subject {
* **private** **long** subjectId;
* **private** String subTitle;
* **private** **int** durationInHours;
* **private** Set<Book> references;
* **public** Subject() {
* };
* **public** Subject(**long** subjectId, String subTitle, **int** durationInHours, Set<Book> references) {
* **super**();
* **this**.subjectId = subjectId;
* **this**.subTitle = subTitle;
* **this**.durationInHours = durationInHours;
* **this**.references = references;
* }
* /\*\*
* \* **@return** the subjectId
* \*/
* **public** **long** getSubjectId() {
* **return** subjectId;
* }
* /\*\*
* \* **@param** subjectId the subjectId to set
* \*/
* **public** **void** setSubjectId(**long** subjectId) {
* **this**.subjectId = subjectId;
* }
* /\*\*
* \* **@return** the subTitle
* \*/
* **public** String getSubTitle() {
* **return** subTitle;
* }
* /\*\*
* \* **@param** subTitle the subTitle to set
* \*/
* **public** **void** setSubTitle(String subTitle) {
* **this**.subTitle = subTitle;
* }
* /\*\*
* \* **@return** the durationInHours
* \*/
* **public** **int** getDurationInHours() {
* **return** durationInHours;
* }
* /\*\*
* \* **@param** durationInHours the durationInHours to set
* \*/
* **public** **void** setDurationInHours(**int** durationInHours) {
* **this**.durationInHours = durationInHours;
* }
* /\*\*
* \* **@return** the references
* \*/
* **public** Set<Book> getReferences() {
* **return** references;
* }
* /\*\*
* \* **@param** references the references to set
* \*/
* **public** **void** setReferences(Set<Book> references) {
* **this**.references = references;
* }
* /\*
* \* (non-Javadoc)
* \*
* \* @see java.lang.Object#toString()
* \*/
* @Override
* **public** String toString() {
* **return** "Subject [subjectId=" + subjectId + ", subTitle=" + subTitle + ", durationInHours=" + durationInHours
* + ", references=" + references + "]";
* }
* }

# BookDao.java

* package com.fsd.corejava.dao;
* import java.sql.SQLException;
* import java.util.List;
* import com.fsd.corejava.entities.Book;
* /\*\*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* public interface BookDao {
* Book getBookById(long bookId)throws SQLException;
* List<Book> getAllBooks() throws SQLException;
* boolean addBook(Book book) throws SQLException;
* boolean removeBook(long bookId) throws SQLException;
* }

# SubjectDao.java

* package com.fsd.corejava.dao;
* import java.sql.SQLException;
* import java.util.List;
* import java.util.Set;
* import com.fsd.corejava.entities.Subject;
* /\*\*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* public interface SubjectDao {
* List<Subject> getSubjectById(long subjectId) throws SQLException;
* List<Subject> getAllSubjects() throws SQLException;
* boolean addSubject(Subject subject) throws SQLException;
* boolean removeSubject(long subjectId) throws SQLException;
* }

# BookDaoImpl.java

* package com.fsd.corejava.daoimpl;
* import java.sql.Connection;
* import java.sql.Date;
* import java.sql.PreparedStatement;
* import java.sql.ResultSet;
* import java.sql.SQLException;
* import java.util.ArrayList;
* import java.util.HashSet;
* import java.util.List;
* import java.util.Set;
* import com.fsd.corejava.dao.BookDao;
* import com.fsd.corejava.entities.Book;
* import com.fsd.corejava.test.TestApplication;
* /\*\*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* public class BookDaoImpl implements BookDao {
* @Override
* public Book getBookById(long bookId) throws SQLException {
* Book book = null;
* String sql = "SELECT \* FROM book WHERE book\_id = ?";
* TestApplication testApplication = new TestApplication();
* Connection connection = testApplication.connect();
* PreparedStatement statement = connection.prepareStatement(sql);
* statement.setLong(1, bookId);
* ResultSet resultSet = statement.executeQuery();
* if (resultSet.next()) {
* long bId = resultSet.getLong("book\_id");
* String title = resultSet.getString("title");
* double price = resultSet.getDouble("price");
* int volume = resultSet.getInt("volume");
* Date pubDate = resultSet.getDate("publish\_date");
* long subId = resultSet.getLong("subject\_id");
* book = new Book(bId, title, price, volume, pubDate, subId);
* }
* resultSet.close();
* statement.close();
* return book;
* }
* @Override
* public List<Book> getAllBooks() throws SQLException {
* //Book book = null;
* List<Book> bookList = new ArrayList<>();
* String sql = "SELECT \* FROM book";
* TestApplication testApplication = new TestApplication();
* Connection connection = testApplication.connect();
* PreparedStatement statement = connection.prepareStatement(sql);
* //statement.setLong(1, bookId);
* ResultSet resultSet = statement.executeQuery();
* while (resultSet.next()) {
* long bId = resultSet.getLong("book\_id");
* String title = resultSet.getString("title");
* double price = resultSet.getDouble("price");
* int volume = resultSet.getInt("volume");
* Date pubDate = resultSet.getDate("publish\_date");
* long subId = resultSet.getLong("subject\_id");
* Book book = new Book(bId, title, price, volume, pubDate, subId);
* bookList.add(book);
* }
* resultSet.close();
* statement.close();
* return bookList;
* }
* @Override
* public boolean addBook(Book book) throws SQLException {
* TestApplication testApplication = new TestApplication();
* Connection connection = testApplication.connect();
* PreparedStatement ps = connection.prepareStatement("INSERT INTO book VALUES (NULL, ?, ?, ?, ?, ?)");
* ps.setString(1, book.getTitle());
* ps.setDouble(2, book.getPrice());
* ps.setInt(3, book.getVolume());
* ps.setDate(4, (Date) book.getPublishDate());
* ps.setLong(5, book.getSubjectId());
* boolean insertUpdate = ps.executeUpdate() > 0;
* testApplication.disconnect();
* return insertUpdate;
* }
* @Override
* public boolean removeBook(long bookId) throws SQLException {
* String sql = "DELETE FROM book where book\_id = ?";
* TestApplication testApplication = new TestApplication();
* Connection connection = testApplication.connect();
* PreparedStatement statement = connection.prepareStatement(sql);
* statement.setLong(1, bookId);
* boolean rowDeleted = statement.executeUpdate() > 0;
* statement.close();
* testApplication.disconnect();
* return rowDeleted;
* }
* }

# SubjectDaoImpl.java

* package com.fsd.corejava.daoimpl;
* import java.sql.Connection;
* import java.sql.Date;
* import java.sql.PreparedStatement;
* import java.sql.ResultSet;
* import java.sql.SQLException;
* import java.util.ArrayList;
* import java.util.HashSet;
* import java.util.List;
* import java.util.Set;
* import com.fsd.corejava.dao.SubjectDao;
* import com.fsd.corejava.entities.Book;
* import com.fsd.corejava.entities.Subject;
* import com.fsd.corejava.test.TestApplication;
* /\*\*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* public class SubjectDaoImpl implements SubjectDao {
* @Override
* public List<Subject> getSubjectById(long subjectId) throws SQLException {
* //Subject subject = null;
* List<Subject> subList = new ArrayList<Subject>();
* String sql = "SELECT \* FROM fsd\_db.subject sub inner join fsd\_db.book book on sub.subject\_id = book.subject\_id and sub.subject\_id = ?";//"SELECT \* FROM subject WHERE subject\_id = ?";
* TestApplication testApplication = new TestApplication();
* Connection connection = testApplication.connect();
* PreparedStatement statement = connection.prepareStatement(sql);
* statement.setLong(1, subjectId);
* ResultSet resultSet = statement.executeQuery();
* while (resultSet.next()) {
* long subId = resultSet.getLong("subject\_id");
* String title = resultSet.getString("subject\_title");
* int duration = resultSet.getInt("duration");
* long bookId = resultSet.getLong("book\_id");
* String bookTitle = resultSet.getString("title");
* double price = resultSet.getDouble("price");
* int volume = resultSet.getInt("volume");
* Date pubDate = resultSet.getDate("publish\_date");
* Set<Book> references = new HashSet<>();
* references.add(new Book(bookId,bookTitle,price,volume,pubDate,0));
* Subject subject = new Subject(subId, title, duration,references);
* subList.add(subject);
* }
* resultSet.close();
* statement.close();
* return subList;
* }
* @Override
* public List<Subject> getAllSubjects() throws SQLException {
* List<Subject> subList = new ArrayList<>();
* String sql = "SELECT \* FROM subject";
* TestApplication testApplication = new TestApplication();
* Connection connection = testApplication.connect();
* PreparedStatement statement = connection.prepareStatement(sql);
* //statement.setLong(1, bookId);
* ResultSet resultSet = statement.executeQuery();
* while (resultSet.next()) {
* long subId = resultSet.getLong("subject\_id");
* String title = resultSet.getString("subject\_title");
* int duration = resultSet.getInt("duration\_in\_hours");
* Subject subject = new Subject(subId, title, duration,null);
* subList.add(subject);
* }
* resultSet.close();
* statement.close();
* return subList;
* }
* @Override
* public boolean addSubject(Subject subject) throws SQLException {
* // TODO Auto-generated method stub
* TestApplication testApplication = new TestApplication();
* Connection connection = testApplication.connect();
* PreparedStatement ps = connection.prepareStatement("INSERT INTO subject VALUES (NULL, ?, ?)");
* ps.setString(1, subject.getSubTitle());
* ps.setInt(2, subject.getDurationInHours());
* boolean insertUpdate = ps.executeUpdate() > 0;
* testApplication.disconnect();
* return insertUpdate;
* }
* @Override
* public boolean removeSubject(long subjectId) throws SQLException {
* String sql = "DELETE FROM subject where subject\_id = ?";
* TestApplication testApplication = new TestApplication();
* Connection connection = testApplication.connect();
* PreparedStatement statement = connection.prepareStatement(sql);
* statement.setLong(1, subjectId);
* boolean rowDeleted = statement.executeUpdate() > 0;
* statement.close();
* testApplication.disconnect();
* return rowDeleted;
* }
* }

# TestApplication.java

* **package** com.fsd.corejava.test;
* **import** java.io.BufferedReader;
* **import** java.io.IOException;
* **import** java.io.InputStreamReader;
* **import** java.sql.Connection;
* **import** java.sql.DriverManager;
* **import** java.sql.SQLException;
* **import** java.text.ParseException;
* **import** java.text.SimpleDateFormat;
* **import** java.util.Comparator;
* **import** java.util.HashSet;
* **import** java.util.LinkedHashSet;
* **import** java.util.List;
* **import** java.util.Set;
* **import** java.util.stream.Collectors;
* **import** com.fsd.corejava.daoimpl.BookDaoImpl;
* **import** com.fsd.corejava.daoimpl.SubjectDaoImpl;
* **import** com.fsd.corejava.entities.Book;
* **import** com.fsd.corejava.entities.Subject;
* /\*\*
* \* **@author** Sudhir Kumar Thakur
* \*
* \*/
* **public** **class** TestApplication {
* **public** **static** **final** String ***DB\_URL*** = "jdbc:mysql://localhost:3306/fsd\_db";
* **public** **static** **final** String ***DB\_USERNAME*** = "root";
* **public** **static** **final** String ***DB\_PASSWORD*** = "pass@word1";
* **private** **static** Connection *connection*;
* /\*\*
* \* **@param** args
* \* **@throws** ParseException
* \*/
* **public** **static** **void** main(String[] args) **throws** ParseException {
* // **TODO** Auto-generated method stub
* **try** {
* **char** select;
* **do** {
* System.***out***.println("please select menu items::::::::::::::::");
* System.***out***.println("a.Add a Subject");
* System.***out***.println("b.Add a Book");
* System.***out***.println("c.Delete a Subject");
* System.***out***.println("d.Delete a book");
* System.***out***.println("e.Search for a book");
* System.***out***.println("f.Search for a subject");
* System.***out***.println("g.Sort Book By Title");
* System.***out***.println("h.Sort Subject By Subject Title");
* System.***out***.println("i.Sort Books by publish Date");
* System.***out***.println("j.Exit");
* BookDaoImpl bookDaoImpl = **new** BookDaoImpl();
* SubjectDaoImpl subjectDaoImpl = **new** SubjectDaoImpl();
* BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));
* select = br.readLine().charAt(0);
* **switch** (select) {
* **case** 'a':
* System.***out***.println("<---Add a Subject--->");
* Subject subObj = **new** Subject();
* System.***out***.println("Enter Subject Title: ");
* String subjectTitle = br.readLine();
* subObj.setSubTitle(subjectTitle);
* System.***out***.println("Enter Subject durationInHours: ");
* String durationInHours = br.readLine();
* subObj.setDurationInHours(Integer.*parseInt*(durationInHours));
* **boolean** isSubAdded = subjectDaoImpl.addSubject(subObj);
* **if** (isSubAdded) {
* System.***out***.println("Subject is added successfully");
* } **else** {
* System.***out***.println("Please try again");
* }
* **break**;
* **case** 'b':
* System.***out***.println("<---Add a Book--->");
* Book bookObj = **new** Book();
* System.***out***.println("Enter Book Title: ");
* String bookTitle = br.readLine();
* bookObj.setTitle(bookTitle);
* System.***out***.println("Enter price: ");
* **double** price = Double.*parseDouble*(br.readLine());
* System.***out***.println("price: " + price);
* bookObj.setPrice(price);
* System.***out***.println("Enter volume: ");
* String volume = br.readLine();
* bookObj.setVolume(Integer.*parseInt*(volume));
* System.***out***.println("volume: " + volume);
* System.***out***.println("Enter Publish date in dd/MM/yyyy format: ");
* String publishDate = br.readLine();
* System.***out***.println("publishDate: " + publishDate);
* SimpleDateFormat sdf = **new** SimpleDateFormat("dd/MM/yyyy");
* java.util.Date localPublishDate = sdf.parse(publishDate);
* java.sql.Date sqlPublishDate = **new** java.sql.Date(localPublishDate.getTime());
* // LocalDate localPublishDate = LocalDate.parse(publishDate,
* // DateTimeFormatter.ofPattern("dd/MM/yyyy"));
* bookObj.setPublishDate(sqlPublishDate);
* System.***out***.println("Enter Subject Id: ");
* String subjectId = br.readLine();
* bookObj.setSubjectId(Long.*parseLong*(subjectId));
* **boolean** isBookAdded = Boolean.***FALSE***;
* **try** {
* isBookAdded = bookDaoImpl.addBook(bookObj);
* } **catch** (Exception e) {
* }
* **if** (isBookAdded) {
* System.***out***.println("Book is added successfully");
* } **else** {
* System.***out***.println("Something went wrong please try again");
* }
* **break**;
* **case** 'c':
* System.***out***.println("Delete a Subject");
* System.***out***.println("Enter Subject Id to delete: ");
* **long** subId = Long.*parseLong*(br.readLine());
* Boolean isSubRemoved = Boolean.***FALSE***;
* **try** {
* isSubRemoved = subjectDaoImpl.removeSubject(subId);
* } **catch** (Exception e) {
* // **TODO**: handle exception
* }
* **if** (isSubRemoved) {
* System.***out***.println("Subject has been removed successfully");
* } **else** {
* System.***out***.println("Something went wrong, Please try again...");
* }
* **break**;
* **case** 'd':
* System.***out***.println("Delete a Book");
* System.***out***.println("Enter Book Id to delete: ");
* **long** bookId = Long.*parseLong*(br.readLine());
* Boolean isBookRemoved = Boolean.***FALSE***;
* **try** {
* isBookRemoved = bookDaoImpl.removeBook(bookId);
* } **catch** (Exception e) {
* // **TODO**: handle exception
* }
* **if** (isBookRemoved) {
* System.***out***.println("Book has been Removed successfully");
* } **else** {
* System.***out***.println("Something went wrong, Please try again..");
* }
* **break**;
* **case** 'e':
* System.***out***.println("<---Search for a Book--->");
* System.***out***.println("Enter Book Id to Search: ");
* **long** bookIdSearch = Long.*parseLong*(br.readLine());
* Book book = bookDaoImpl.getBookById(bookIdSearch);
* **if** (book != **null**) {
* System.***out***.println("Search book details: " + book);
* } **else** {
* System.***out***.println("No data found");
* }
* **break**;
* **case** 'f':
* System.***out***.println("<---Search for a Subject--->");
* System.***out***.println("Enter Subject Id to Search: ");
* **long** SubIdSearch = Long.*parseLong*(br.readLine());
* List<Subject> subjectList = subjectDaoImpl.getSubjectById(SubIdSearch);
* Set<Book> references = **new** HashSet<>();
* Set set = **new** LinkedHashSet<>();
* **for** (Subject sub : subjectList) {
* set.add(sub.getSubjectId());
* set.add(sub.getSubTitle());
* set.add(sub.getDurationInHours());
* references.addAll(sub.getReferences());
* }
* set.add(references);
* **if** (set != **null**) {
* System.***out***.println("Search Subject details: " + set);
* } **else** {
* System.***out***.println("No data found");
* }
* **break**;
* **case** 'g':
* System.***out***.println("<---Sort Book By Title--->");
* List<Book> bookList = bookDaoImpl.getAllBooks();
* StringBuilder sbBeforeSortBook = **new** StringBuilder();
* **for**(Book book1: bookList) {
* sbBeforeSortBook = sbBeforeSortBook.append(book1.getTitle()).append(",");
* }
* System.***out***.println("Book List before sorting: "+sbBeforeSortBook.toString().substring(0, sbBeforeSortBook.length()-1));
* List<Book> sBookList = bookList.stream().sorted(Comparator.*comparing*(Book::getTitle)).collect(Collectors.*toList*());
* StringBuilder sbAfterSortBook = **new** StringBuilder();
* **for**(Book book1: sBookList) {
* sbAfterSortBook = sbAfterSortBook.append(book1.getTitle()).append(",");
* }
* System.***out***.println("Book List after sorting: "+sbAfterSortBook.toString().substring(0, sbAfterSortBook.length()-1));
* **break**;
* **case** 'h':
* System.***out***.println("<---Sort Subject By Subject Title--->");
* List<Subject> subList = subjectDaoImpl.getAllSubjects();
* StringBuilder sbBeforeSort = **new** StringBuilder();
* **for**(Subject subject: subList) {
* sbBeforeSort = sbBeforeSort.append(subject.getSubTitle()).append(",");
* }
* System.***out***.println("Subject List before sorting: "+sbBeforeSort.toString().substring(0, sbBeforeSort.length()-1));
* List<Subject> sSubList = subList.stream().sorted(Comparator.*comparing*(Subject::getSubTitle)).collect(Collectors.*toList*());
* //sBookList.forEach(e -> System.out.println(e.toString()));
* StringBuilder sbAfterSort = **new** StringBuilder();
* **for**(Subject subject: sSubList) {
* sbAfterSort = sbAfterSort.append(subject.getSubTitle()).append(",");
* }
* System.***out***.println("Subject List after sorting:::"+sbAfterSort.toString().substring(0, sbAfterSort.length()-1));
* **break**;
* **case** 'i':
* System.***out***.println("<---Sort Books by publish Date--->");
* List<Book> bookList1 = bookDaoImpl.getAllBooks();
* StringBuilder sbBeforeSortByDate = **new** StringBuilder();
* **for**(Book bookDate: bookList1) {
* sbBeforeSortByDate = sbBeforeSortByDate.append(bookDate.getTitle()).append(",");
* }
* System.***out***.println("Book List By Publish Date before sorting: "+sbBeforeSortByDate.toString().substring(0, sbBeforeSortByDate.length()-1));
* List<Book> sBookList1 = bookList1.stream().sorted(Comparator.*comparing*(Book::getPublishDate)).collect(Collectors.*toList*());
* StringBuilder sbAfterSortByDate = **new** StringBuilder();
* **for**(Book bookDate: sBookList1) {
* sbAfterSortByDate = sbAfterSortByDate.append(bookDate.getTitle()).append(",");
* }
* System.***out***.println("Book List By Publish Date after sorting: "+sbAfterSortByDate.toString().substring(0, sbAfterSortByDate.length()-1));
* **break**;
* }
* } **while** (select != 'j');
* } **catch** (SQLException e) {
* // **TODO** Auto-generated catch block
* e.printStackTrace();
* } **catch** (IOException e) {
* // **TODO** Auto-generated catch block
* e.printStackTrace();
* }
* }
* // getting Database Connection
* **public** **static** Connection connect() **throws** SQLException {
* **if** (*connection* == **null** || *connection*.isClosed()) {
* **try** {
* Class.*forName*("com.mysql.jdbc.Driver");
* } **catch** (ClassNotFoundException e) {
* **throw** **new** SQLException(e);
* }
* *connection* = DriverManager.*getConnection*(***DB\_URL***, ***DB\_USERNAME***, ***DB\_PASSWORD***);
* }
* **return** *connection*;
* }
* // Closing Database connection
* **public** **static** **void** disconnect() **throws** SQLException {
* **if** (*connection* != **null** && !*connection*.isClosed()) {
* *connection*.close();
* }
* }
* }

**Screenshots**















