Codes:

Book Entity:

package com.crudopearation;

import java.io.Serializable;

import java.time.LocalDate;

import java.util.Date;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="Book")

public class Book implements Serializable{

@Id

@Column(name="bookId")

int bookId;

@Column(name="title")

String title;

@Column(name="price")

double price;

@Column(name="volume")

Integer volume;

@Column(name="publishDate")

Date publishDate;

@Override

public String toString() {

return "Book [bookId=" + bookId + ", title=" + title + ", price="

+ price + ", volume=" + volume + ", publishDate=" + publishDate

+ "]";

}

public Book(int bookId, String title, double price, Integer volume,

Date publishDate) {

super();

this.bookId = bookId;

this.title = title;

this.price = price;

this.volume = volume;

this.publishDate = publishDate;

}

public Book(){

}

public long getBookId() {

return bookId;

}

public void setBookId(int 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 Integer getVolume() {

return volume;

}

public void setVolume(Integer volume) {

this.volume = volume;

}

public Date getPublishDate() {

return publishDate;

}

public void setPublishDate(Date publishDate) {

this.publishDate = publishDate;

}

}

Subject Entity:

package com.crudopearation;

import java.io.Serializable;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import javax.persistence.Table;

@Entity

@Table(name="Subject")

public class Subject implements Serializable {

@Id

@Column(name="subjectId")

long subjectId;

@Column(name="subtitle")

String subtitle;

@Column(name="durationInHours")

int durationInHours;

@Column(name="book\_ref\_id")

Long book\_ref\_id;

//@OneToOne

//Set<Book> references=new HashSet<Book>();

public Subject(){

}

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;

}\*/

public Subject(long subjectId, String subtitle, int durationInHours,

Long references) {

super();

this.subjectId = subjectId;

this.subtitle = subtitle;

this.durationInHours = durationInHours;

this.book\_ref\_id = references;

}

@Override

public String toString() {

return "Subject [subjectId=" + subjectId + ", subtitle=" + subtitle

+ ", durationInHours=" + durationInHours + ", references="

+ book\_ref\_id + "]";

}

public long getBook\_ref\_id() {

return book\_ref\_id;

}

public void setBook\_ref\_id(long book\_ref\_id) {

this.book\_ref\_id = book\_ref\_id;

}

}

DbOperation.Java:

**package** com.crudopearation;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectOutputStream;

**import** java.text.SimpleDateFormat;

**import** java.time.LocalDate;

**import** java.util.ArrayList;

**import** java.util.HashSet;

**import** java.util.List;

**import** java.util.Scanner;

**import** java.util.Set;

**import** javax.transaction.Transaction;

**import** org.apache.log4j.Logger;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** org.hibernate.cfg.Configuration;

**import** org.hibernate.service.ServiceRegistry;

**public** **class** DbOperation {

**static** Session *sessionObj*;

**static** SessionFactory *sessionFactoryObj*;

**public** **final** **static** Logger ***logger*** = Logger.*getLogger*(DbOperation.**class**);

// This Method Is Used To Create The Hibernate's SessionFactory Object

**private** **static** SessionFactory buildSessionFactory() {

// Creating Configuration Instance & Passing Hibernate Configuration File

Configuration configObj = **new** Configuration();

configObj.configure("hibernate.cfg.xml");

// Since Hibernate Version 4.x, ServiceRegistry Is Being Used

ServiceRegistry serviceRegistryObj = **new** StandardServiceRegistryBuilder().applySettings(configObj.getProperties()).build();

// Creating Hibernate SessionFactory Instance

*sessionFactoryObj* = configObj.buildSessionFactory(serviceRegistryObj);

**return** *sessionFactoryObj*;

}

// Method 1: This Method Used To Create A New Student Record In The Database Table

**public** **static** **void** addBook() {

**int** count = 0;

Book bookObj = **null**;

**try** {

// Getting Session Object From SessionFactory

*sessionObj* = *buildSessionFactory*().openSession();

// Getting Transaction Object From Session Object

*sessionObj*.beginTransaction();

// Creating Transaction Entities

**for**(**int** j = 101; j <= 101; j++) {

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

count = count + 1;

bookObj = **new** Book();

System.***out***.println("enter bookid");

**int** k = sc.nextInt();

bookObj.setBookId(k);

System.***out***.println("enter book title");

String t = sc.next();

bookObj.setTitle(t);

System.***out***.println("enter book price");

**double** p = sc.nextDouble();

bookObj.setPrice(p);

System.***out***.println("enter book volume");

**int** v = sc.nextInt();

bookObj.setVolume(v);

System.***out***.println("enter book publish date");

String date = sc.next();

//LocalDate d = LocalDate.parse(date);

SimpleDateFormat formatter = **new** SimpleDateFormat("yyyy-MM-dd"); // your template here

java.util.Date dateStr = formatter.parse(date);

bookObj.setPublishDate(dateStr);

*sessionObj*.save(bookObj);

}

// Committing The Transactions To The Database

*sessionObj*.getTransaction().commit();

System.***out***.println("\nSuccessfully Created '" + count + "' Records In The Database!\n");

***logger***.info("\nSuccessfully Created '" + count + "' Records In The Database!\n");

} **catch**(Exception sqlException) {

**if**(**null** != *sessionObj*.getTransaction()) {

***logger***.info("\n.......Transaction Is Being Rolled Back.......\n");

*sessionObj*.getTransaction().rollback();

}

sqlException.printStackTrace();

}

}

**public** **static** List<Book> searchBook() {

List bookList = **new** ArrayList();

**try** {

// Getting Session Object From SessionFactory

*sessionObj* = *buildSessionFactory*().openSession();

// Getting Transaction Object From Session Object

*sessionObj*.beginTransaction();

bookList = *sessionObj*.createQuery("FROM Book").list();

} **catch**(Exception sqlException) {

**if**(**null** != *sessionObj*.getTransaction()) {

***logger***.info("\n.......Transaction Is Being Rolled Back.......\n");

*sessionObj*.getTransaction().rollback();

}

sqlException.printStackTrace();

}

**return** bookList;

}

**public** **static** **void** addSubject() {

**try**{

List<Book> list=*searchBook*();

Long bookrefid=**null**;

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

bookrefid=book.getBookId();

}

System.***out***.println("before save subject>>>>>>>>>>>");

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

sub.setSubjectId(1);

sub.setSubtitle("physics");

sub.setDurationInHours(12);

sub.setBook\_ref\_id(bookrefid);

System.***out***.println("bookrefid>>>>>>>>>"+bookrefid);

*sessionObj* = *buildSessionFactory*().openSession();

// Getting Transaction Object From Session Object

*sessionObj*.beginTransaction();

*sessionObj*.save(sub);

*sessionObj*.getTransaction().commit();

System.***out***.println("After save subject>>>>>>>>>>>");

}**catch**(Exception exp){

exp.printStackTrace();

}

}

**public** **static** List<Subject> searchSubject() {

List subjectList = **new** ArrayList();

**try** {

// Getting Session Object From SessionFactory

*sessionObj* = *buildSessionFactory*().openSession();

// Getting Transaction Object From Session Object

*sessionObj*.beginTransaction();

subjectList = *sessionObj*.createQuery("FROM Subject").list();

} **catch**(Exception sqlException) {

/\*if(null != sessionObj.getTransaction()) {

logger.info("\n.......Transaction Is Being Rolled Back.......\n");

sessionObj.getTransaction().rollback();

}\*/

sqlException.printStackTrace();

}

**return** subjectList;

}

**public** **static** **void** deleteBook(**int** bookId) {

**try** {

// Getting Session Object From SessionFactory

*sessionObj* = *buildSessionFactory*().openSession();

// Getting Transaction Object From Session Object

org.hibernate.Transaction t=(org.hibernate.Transaction) *sessionObj*.beginTransaction();

Book book=(Book)*sessionObj*.get(Book.**class**,bookId);

System.***out***.println("Book for delete>>>>>>>>>>>"+book);

*sessionObj*.delete(book);

t.commit();

System.***out***.println("After delete>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>");

} **catch**(Exception sqlException) {

/\*if(null != sessionObj.getTransaction()) {

logger.info("\n.......Transaction Is Being Rolled Back.......\n");

sessionObj.getTransaction().rollback();

}\*/

sqlException.printStackTrace();

}

//return subjectList;

}

// Method 4(a): This Method Is Used To Delete A Particular Record From The Database Table

**public** **static** **void** deleteSubject(**int** subjectId) {

**try** {

// Getting Session Object From SessionFactory

*sessionObj* = *buildSessionFactory*().openSession();

// Getting Transaction Object From Session Object

*sessionObj*.beginTransaction();

List<Subject> subObj = *searchSubject*();

**for** (Subject s1:subObj)

*sessionObj*.delete(s1);

// Committing The Transactions To The Database

*sessionObj*.getTransaction().commit();

***logger***.info("\nStudent With Id?= " + subjectId + " Is Successfully Deleted From The Database!\n");

} **catch**(Exception sqlException) {

**if**(**null** != *sessionObj*.getTransaction()) {

***logger***.info("\n.......Transaction Is Being Rolled Back.......\n");

*sessionObj*.getTransaction().rollback();

}

sqlException.printStackTrace();

} **finally** {

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

*sessionObj*.close();

}

}

}

}

Test class:

package corejavaassigbnment;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.time.Instant;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.HashSet;

import java.util.List;

import java.util.Map;

import java.util.Scanner;

import java.util.Set;

public class AppTest {

public static void main(String[] args) throws IOException,

ClassNotFoundException {

Scanner sc = new Scanner(System.in);

int ch, k, v;

String t, d;

double p;

while (true) {

System.out.println("01. Add a Book");

System.out.println("02. Add a Subject");

System.out.println("03. Delete a Book");

System.out.println("04. Delete a Subject");

System.out.println("05. Search for a book");

System.out.println("06. Search for a subject");

System.out.println("07. Exit");

System.out.print("Enter Your Choice : ");

ch = sc.nextInt();

switch (ch) {

case 1:

System.out.println("01. Add a Book");

AddBook();

break;

case 2:

System.out.println("02. Add a Subject");

AddSubject();

break;

case 3:

System.out.println("03. Delete a Book");

Scanner sc3 = new Scanner(System.in);

System.out.println("Enter Book tittle");

String tittle1 = sc3.next();

deleteBook(tittle1);

break;

case 4:

System.out.println("03. Delete a Subject");

Scanner sc4 = new Scanner(System.in);

System.out.println("Enter Book tittle");

String tittle3 = sc4.next();

deleteSubject(tittle3);

break;

case 5:

System.out.println("05. Search for a book");

try {

Scanner sc6 = new Scanner(System.in);

System.out.println("Enter Book tittle");

String tittle5 = sc6.next();

Set<Book> sb1 = retriveBook(tittle5);

System.out.println("searched book =====" + sb1);

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

break;

case 6:

System.out.println("06. Search for a subject");

try {

// Set<Book> sb1=retriveBook("www");

Scanner sc2 = new Scanner(System.in);

System.out.println("Enter subject tittle");

String tittle = sc2.next();

retriveSubject(tittle);

// System.out.println("sb1====="+sb1);

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

break;

case 7:

System.out.println("Exited");

System.exit(0);

default:

System.out.println("Wrong Entry");

}

}

}

private static void deleteSubject(String subjecttittle)

throws ClassNotFoundException, IOException {

Set<Book> setbook = new HashSet<Book>();

Map<Integer, Subject> map = new HashMap<Integer, Subject>();

FileInputStream fint = new FileInputStream("D:\\f2.txt");

ObjectInputStream oint = new ObjectInputStream(fint);

Subject subject = (Subject) oint.readObject();

System.out.println("Before remove subject>>>>>>>>>>>>>>>>>>"

+ subject.toString());

if (subject.getSubtitle().equals(subjecttittle)) {

setbook = subject.getReferences();

}

for (Book book : setbook) {

if (book.getTitle().equals(subjecttittle)) {

setbook.remove(book);

}

}

System.out.println("After remove subject>>>>>>>>>>>>>>>>>>" + setbook);

}

private static void deleteBook(String title) throws ClassNotFoundException,

IOException {

Set<Book> s2 = retriveBook(title);

System.out.println("before remove " + s2);

// Scanner sc=new Scanner(System.in);

// System.out.println("enter bookid to delete");

// int bid=sc.nextInt();

for (Book book : s2) {

s2.remove(book);

}

System.out.println("after remove" + s2);

}

private static void AddSubject() throws IOException {

List<Subject> s1 = new ArrayList<Subject>();

Scanner sc0 = new Scanner(System.in);

System.out.println("enter total entry--------\n");

int total = sc0.nextInt();

// Book b1=null;

for (int i = 0; i < total; i++) {

Scanner sc = new Scanner(System.in);

System.out.println("enter subjectId");

long k = sc.nextInt();

System.out.println("enter subtitle");

String t = sc.next();

System.out.println("enter durationInHours");

int p = sc.nextInt();

Set<Book> mapsubject = null;

try {

mapsubject = retriveBook(t);

System.out.println("mapsubject>>>>>>>>>>>>>" + mapsubject);

// mapsubject = getBook(t);

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

FileOutputStream fout = new FileOutputStream("D:\\f2.txt");

ObjectOutputStream out = new ObjectOutputStream(fout);

out.writeObject(new Subject(k, t, p, mapsubject));

out.flush();

System.out.println("success");

}

}

private static Set<Book> getBook(String t) throws IOException,

ClassNotFoundException {

Set<Book> sb = new HashSet<Book>();

FileInputStream fint = new FileInputStream("D:\\f.txt");

ObjectInputStream oint = new ObjectInputStream(fint);

Map<String, Book> sbook = (Map<String, Book>) oint.readObject();

System.out.println("Set of books----------" + sbook);

Book book = new Book();

Set<String> set = sbook.keySet();

for (String str : set) {

if (set.contains(t)) {

sb.add(sbook.get(str));

}

}

fint.close();

oint.close();

return sb;

}

private static Set<Book> retriveSubject(String tittle)

throws ClassNotFoundException, IOException {

Set<Book> sb = new HashSet<Book>();

Set<Book> sb1 = new HashSet<Book>();

FileInputStream fint = new FileInputStream("D:\\f2.txt");

ObjectInputStream oint = new ObjectInputStream(fint);

Subject subject = (Subject) oint.readObject();

System.out.println("subject----------" + subject.toString());

if (subject.getSubtitle().equals(tittle)) {

System.out.println("retriveSubject>>>>>>>>>>>>>>"

+ subject.getSubjectId() + "----"

+ subject.getDurationInHours() + "--------"

+ subject.getSubtitle() + "----" + subject.getReferences());

}

fint.close();

oint.close();

return sb;

}

private static Set<Book> retriveBook(String tittle)

throws ClassNotFoundException, IOException {

Set<Book> sb = new HashSet<Book>();

FileInputStream fint = new FileInputStream("D:\\f.txt");

ObjectInputStream oint = new ObjectInputStream(fint);

Map<String, Book> sbook = (Map<String, Book>) oint.readObject();

System.out.println("Set of books----------" + sbook);

Book book = new Book();

Set<String> set = sbook.keySet();

for (String str : set) {

if (str.equals(tittle)) {

sb.add(sbook.get(str));

}

}

fint.close();

oint.close();

return sb;

}

public static Map<String, Book> AddBook() throws IOException {

// Set<Book> s1 = new HashSet<Book>();

Map<String, Book> map = new HashMap<String, Book>();

Scanner sc0 = new Scanner(System.in);

System.out.println("enter total entry--------");

int total = sc0.nextInt();

// Book b1=null;

for (int i = 0; i < total; i++) {

Scanner sc = new Scanner(System.in);

System.out.println("enter bookid");

int k = sc.nextInt();

System.out.println("enter book title");

String t = sc.next();

System.out.println("enter book price");

double p = sc.nextDouble();

System.out.println("enter book volume");

int v = sc.nextInt();

System.out.println("enter book publish date");

String date = sc.next();

LocalDate d = LocalDate.parse(date);

map.put(t, new Book(k, t, p, v, d));

}

FileOutputStream fout = new FileOutputStream("D:\\f.txt");

ObjectOutputStream out = new ObjectOutputStream(fout);

out.writeObject(map);

out.flush();

System.out.println("success");

return map;

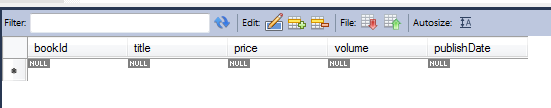
}

}

Db:demoabc

Before insertion in db:

Select \* from book;



**Add a Book in data base:-**

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

enter bookid

121

enter book title

physicsone

enter book price

450

enter book volume

2

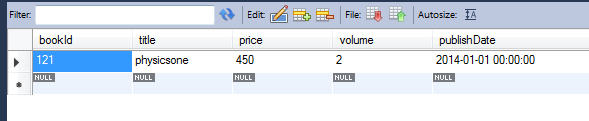
enter book publish date

2014-01-01 10:00:00

Hibernate: insert into Book (price, publishDate, title, volume, bookId) values (?, ?, ?, ?, ?)

Successfully Created '1' Records In The Database!

After insertion in db:-



**Search a book:**

01. Add a Book

02. Add a Subject

03. Delete a Book

04. Delete a Subject

05. Search for a book

06. Search for a subject

07. Exit

Enter Your Choice : 05

05. Search for a book

log4j:WARN No appenders could be found for logger (org.jboss.logging).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Hibernate: select book0\_.bookId as bookId1\_0\_, book0\_.price as price2\_0\_, book0\_.publishDate as publishD3\_0\_, book0\_.title as title4\_0\_, book0\_.volume as volume5\_0\_ from Book book0\_

searched book =====[Book [bookId=121, title=physicsone, price=450.0, volume=2, publishDate=2014-01-01 00:00:00.0]]

01. Add a Book

02. Add a Subject

03. Delete a Book

04. Delete a Subject

05. Search for a book

06. Search for a subject

07. Exit

Enter Your Choice :06

**Search for subject**

01. Add a Book

02. Add a Subject

03. Delete a Book

04. Delete a Subject

05. Search for a book

06. Search for a subject

07. Exit

Enter Your Choice : 06

06. Search for a subject

log4j:WARN No appenders could be found for logger (org.jboss.logging).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Hibernate: select subject0\_.subjectId as subjectI1\_1\_, subject0\_.book\_ref\_id as book\_ref2\_1\_, subject0\_.durationInHours as duration3\_1\_, subject0\_.subtitle as subtitle4\_1\_ from Subject subject0\_

subjectID:1

SubejctTittle:physics

SubjectDuration:12

SubjectBookrefid:121

01. Add a Book

02. Add a Subject

03. Delete a Book

04. Delete a Subject

05. Search for a book

06. Search for a subject

07. Exit

Enter Your Choice :

**Delete a subject:**

01. Add a Book

02. Add a Subject

03. Delete a Book

04. Delete a Subject

05. Search for a book

06. Search for a subject

07. Exit

Enter Your Choice : 04

03. Delete a Subject

Enter Book tittle

1

log4j:WARN No appenders could be found for logger (org.jboss.logging).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Hibernate: select subject0\_.subjectId as subjectI1\_1\_, subject0\_.book\_ref\_id as book\_ref2\_1\_, subject0\_.durationInHours as duration3\_1\_, subject0\_.subtitle as subtitle4\_1\_ from Subject subject0\_

Hibernate: delete from Subject where subjectId=?

01. Add a Book

02. Add a Subject

03. Delete a Book

04. Delete a Subject

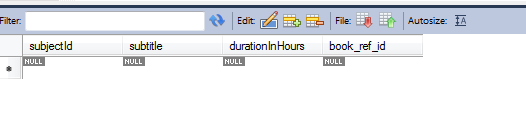
05. Search for a book

06. Search for a subject

07. Exit

Enter Your Choice :

After deletion:



**Exit**

01. Add a Book

02. Add a Subject

03. Delete a Book

04. Delete a Subject

05. Search for a book

06. Search for a subject

07. Exit

Enter Your Choice : 04

03. Delete a Subject

Enter Book tittle

1

log4j:WARN No appenders could be found for logger (org.jboss.logging).

log4j:WARN Please initialize the log4j system properly.

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Hibernate: select subject0\_.subjectId as subjectI1\_1\_, subject0\_.book\_ref\_id as book\_ref2\_1\_, subject0\_.durationInHours as duration3\_1\_, subject0\_.subtitle as subtitle4\_1\_ from Subject subject0\_

Hibernate: delete from Subject where subjectId=?

01. Add a Book

02. Add a Subject

03. Delete a Book

04. Delete a Subject

05. Search for a book

06. Search for a subject

07. Exit

Enter Your Choice : 07

Exited