# Application.java

* package com.cognizant;
* import org.springframework.context.ApplicationContext;
* import org.springframework.context.support.ClassPathXmlApplicationContext;
* /\*\*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* public class Application {
* public static void main(String[] args) {
* ApplicationContext applicationContext=new ClassPathXmlApplicationContext("applicationContext.xml");
* CommandLineRunner commandLineRunner = (CommandLineRunner) applicationContext.getBean("commandLineRunner");
* commandLineRunner.run();
* }
* }

# CommandLineRunner.java

* package com.cognizant;
* import java.io.BufferedReader;
* import java.io.IOException;
* import java.io.InputStreamReader;
* import java.time.LocalDate;
* import com.cognizant.model.Book;
* import com.cognizant.model.Subject;
* import com.cognizant.service.BookService;
* import com.cognizant.service.SubjectService;
* /\*\*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* public class CommandLineRunner {
* private SubjectService subjectService;
* private BookService bookService;
* public void run() {
* while (true) {
* System.out.println("\n");
* System.out.println("1.Add a Subject");
* System.out.println("2.Add a Book");
* System.out.println("3.Delete a Subject");
* System.out.println("4.Delete a book");
* System.out.println("5.Search for a subject");
* System.out.println("6.Search for a book");
* System.out.println("7.Show All");
* System.out.println("8.Exit");
* String input = gatValFromConsole("please select menu items::::::::::::::::");
* select(Integer.parseInt(input));
* }
* }
* public void select(int i) {
* switch (i) {
* case 1:
* addSubject();
* break;
* case 2:
* addBook();
* break;
* case 3:
* deleteSubject();
* break;
* case 4:
* deleteBook();
* break;
* case 5:
* searchSubject();
* break;
* case 6:
* searchBook();
* break;
* case 7:
* showAll();
* break;
* case 8:
* exit();
* break;
* }
* }
* private void addBook() {
* Book book = new Book();
* book.setBookId(Long.parseLong(gatValFromConsole("Enter book id")));
* book.setTitle(gatValFromConsole("Enter book title"));
* book.setPrice(Double.parseDouble(gatValFromConsole("Enter book price")));
* book.setVolume(Integer.parseInt(gatValFromConsole("Enter book volume")));
* book.setPublishDate(LocalDate.parse(gatValFromConsole("Enter book publish date(yyyy-mm-dd) ")));
* Book bookObj = bookService.addBook(book);
* if (null != bookObj) {
* System.out.println(bookObj.getTitle()+" has been added successfully");
* System.out.println(book.toString());
* } else {
* System.out.println("Encountered Issue while adding book");
* }
* }
* private void deleteBook() {
* long bookId = Long.parseLong(gatValFromConsole("Enter the book id to be deleted"));
* if (bookService.deleteBook(bookId))
* System.out.println("Book with id :" + bookId + " has been deleted successfully.");
* }
* private void searchBook() {
* long bookId = Long.parseLong(gatValFromConsole("Enter the book id for search"));
* Book book = bookService.searchBook(bookId);
* System.out.println("\n");
* System.out.println("<------Search Result------->");
* System.out.println(book);
* }
* private void addSubject() {
* Subject subject = new Subject();
* subject.setSubjectId(Long.parseLong(gatValFromConsole("Enter subject id")));
* subject.setSubTitle(gatValFromConsole("Enter subject title"));
* subject.setDurationInHours(Integer.parseInt(gatValFromConsole("Enter duration (In Hours)")));
* Subject subjectObj = subjectService.addSubject(subject);
* if (null != subjectObj) {
* System.out.println(subjectObj.getSubTitle()+" has been added successfully.");
* System.out.println(subject.toString());
* } else {
* System.out.println("Encountered Issue while adding subject");
* }
* }
* private void deleteSubject() {
* long subjectId = Long.parseLong(gatValFromConsole("Enter the subject id to be deleted"));
* if (subjectService.deleteSubject(subjectId))
* System.out.println("Subject with id :" + subjectId + " has been deleted successfully.");
* }
* private void searchSubject() {
* long subjectId = Long.parseLong(gatValFromConsole("Enter the subject id for search"));
* Subject subject = subjectService.searchSubject(subjectId);
* System.out.println("\n");
* System.out.println("<------Search Result------->");
* System.out.println(subject);
* }
* private void showAll() {
* System.out.println("\n");
* System.out.println("<------Subject List------->");
* subjectService.fetchAllSubject().stream().forEach(System.out::println);
* System.out.println("\n");
* System.out.println("<------Book List------->");
* bookService.fetchAllBook().stream().forEach(System.out::println);
* }
* private void exit() {
* System.exit(0);
* }
* private String gatValFromConsole(String log) {
* System.out.println(log);
* String inputString = "";
* try {
* BufferedReader bufferRead = new BufferedReader(new InputStreamReader(System.in));
* inputString = bufferRead.readLine();
* } catch (IOException ex) {
* ex.printStackTrace();
* }
* return inputString;
* }
* public void setSubjectService(SubjectService subjectService) {
* this.subjectService = subjectService;
* }
* public void setBookService(BookService bookService) {
* this.bookService = bookService;
* }
* }

# BinaryBookStreamCRUD.java

* package com.cognizant.datastream;
* import java.io.BufferedOutputStream;
* import java.io.DataInputStream;
* import java.io.DataOutputStream;
* import java.io.File;
* import java.io.FileInputStream;
* import java.io.FileOutputStream;
* import java.io.IOException;
* import java.time.LocalDate;
* import java.time.format.DateTimeFormatter;
* import java.util.ArrayList;
* import java.util.List;
* import org.springframework.beans.factory.InitializingBean;
* import com.cognizant.model.Book;
* public class BinaryBookStreamCRUD implements InitializingBean{
* private String fileName = "C:\\Spring\\File\\book.dat";
* private DataOutputStream outStream = null;
* public DataOutputStream openOutputStream(String name) throws Exception {
* DataOutputStream out = null;
* File file = new File(name);
* out = new DataOutputStream(new BufferedOutputStream(new FileOutputStream(file)));
* return out;
* }
* public void writeBook(Book book){
* writeRecords(book,fileName);
* }
* private void writeRecords(Book book, String fileName ) {
* try {
* outStream.writeLong(book.getBookId());
* outStream.writeUTF(book.getTitle());
* outStream.writeInt(book.getVolume());
* outStream.writeDouble(book.getPrice());
* outStream.writeUTF(book.getPublishDate().toString());
* } catch (IOException e) {
* System.out.println("IOERROR: " + e.getMessage() + "\n");
* }
* } // writeRecords()
* public List<Book> readRecords(){
* List<Book> bookList=new ArrayList<Book>();
* try {
* DataInputStream inStream = new DataInputStream(new FileInputStream(fileName));
* while (true) { // Infinite loop
* Long bookId = inStream.readLong(); // Read a record
* String title = inStream.readUTF();
* Integer volume = inStream.readInt();
* Double price = inStream.readDouble();
* String publishDate = inStream.readUTF();
* Book book=new Book();
* book.setBookId(bookId);
* book.setTitle(title);
* book.setVolume(volume);
* book.setPrice(price);
* book.setPublishDate(stringToDate(publishDate));
* bookList.add(book);
* } // while
* } catch (Exception e) {
* System.out.println(bookList.size());
* System.err.println("eeeeeeee "+e);
* }
* return bookList;
* }
* public void open() throws Exception {
* this.outStream= new DataOutputStream(new FileOutputStream(fileName));
* }
* public void close() throws Exception {
* this.outStream.close();
* }
* @Override
* public void afterPropertiesSet() throws Exception {
* //this.out = openOutputStream("data.dat");
* }
* public LocalDate stringToDate(String date) {
* DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
* //convert String to LocalDate
* return LocalDate.parse(date, formatter);
* }
* }

# BinarySubjectStreamCRUD.java

* package com.cognizant.datastream;
* import java.io.BufferedOutputStream;
* import java.io.DataInputStream;
* import java.io.DataOutputStream;
* import java.io.File;
* import java.io.FileInputStream;
* import java.io.FileOutputStream;
* import java.io.IOException;
* import java.util.ArrayList;
* import java.util.List;
* import org.springframework.beans.factory.InitializingBean;
* import com.cognizant.model.Subject;
* public class BinarySubjectStreamCRUD implements InitializingBean {
* private DataOutputStream outStream =null;
* private String fileName = "C:\\Spring\\File\\subject.dat";
* public DataOutputStream openOutputStream(String name) throws Exception {
* DataOutputStream out = null;
* File file = new File(name);
* out = new DataOutputStream(new BufferedOutputStream(new FileOutputStream(file)));
* return out;
* }
* public void writeSubject(Subject subject){
* writeRecords(subject, fileName);
* }
* private void writeRecords(Subject subject, String fileName ) {
* try {
* // Open stream
* outStream.writeLong(subject.getSubjectId());
* outStream.writeUTF(subject.getSubTitle());
* outStream.writeInt(subject.getDurationInHours());
* //outStream.close(); // Close the stream
* } catch (IOException e) {
* System.out.println("IOERROR: " + e.getMessage() + "\n");
* }
* } // writeRecords()
* public List<Subject> readRecords(){
* List<Subject> subjectList=new ArrayList<Subject>();
* try {
* DataInputStream inStream = new DataInputStream(new FileInputStream(fileName));
* while (true) { // Infinite loop
* Long subjectId = inStream.readLong(); // Read a record
* String title = inStream.readUTF();
* Integer durationInHours = inStream.readInt();
* Subject subject=new Subject();
* subject.setSubjectId(subjectId);;
* subject.setSubTitle(title);
* subject.setDurationInHours(durationInHours);
* subjectList.add(subject);
* } // while
* } catch (Exception e) {
* System.out.println(subjectList.size());
* System.err.println("eeeeeeee "+e);
* }
* return subjectList;
* }
* public void open() throws Exception {
* this.outStream= new DataOutputStream(new FileOutputStream(fileName));
* }
* public void close() throws Exception {
* this.outStream.close();
* }
* @Override
* public void afterPropertiesSet() throws Exception {
* //this.out = openOutputStream("data.dat");
* }
* }

# Book.java

* /\*\*
* \*
* \*/
* package com.cognizant.model;
* import java.io.Serializable;
* import java.time.LocalDate;
* /\*\*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* public class Book implements Serializable{
* private static final long serialVersionUID = 3667779253735136971L;
* private long bookId;
* private String title;
* private double price;
* private Integer volume;
* private LocalDate publishDate;
* 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 Integer getVolume() {
* return volume;
* }
* public void setVolume(Integer volume) {
* this.volume = volume;
* }
* public LocalDate getPublishDate() {
* return publishDate;
* }
* public void setPublishDate(LocalDate publishDate) {
* this.publishDate = publishDate;
* }
* @Override
* public String toString() {
* return "Book [bookId=" + bookId + ", title=" + title + ", price=" + price + ", volume=" + volume
* + ", publishDate=" + publishDate + "]";
* }
* }

# Subject.java

* package com.cognizant.model;
* import java.io.Serializable;
* import java.util.Set;
* /\*\*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* public class Subject implements Serializable{
* private static final long serialVersionId = 1L;
* 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;
* }
* @Override
* public String toString() {
* return "Subject [subjectId=" + subjectId + ", subTitle=" + subTitle + ", durationInHours=" + durationInHours
* + "]";
* }
* }

# BookRepository.java

* package com.cognizant.repository;
* import java.util.List;
* import com.cognizant.model.Book;
* public interface BookRepository {
* public Book addBook(Book book);
* public boolean deleteBook(long bookId);
* public Book searchBook(long bookId);
* public List<Book> fetchAllBook();
* }

# BookRepositoryImpl.java

* package com.cognizant.repository;
* import java.util.ArrayList;
* import java.util.List;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.stereotype.Repository;
* import com.cognizant.datastream.BinaryBookStreamCRUD;
* import com.cognizant.model.Book;
* @Repository("bookRepository")
* public class BookRepositoryImpl implements BookRepository{
* private List<Book> bookList=new ArrayList<Book>();
* private BinaryBookStreamCRUD binaryBookStreamCRUD;
* @Override
* public Book addBook(Book book) {
* bookList.add(book);
* saveToBinaryFile(bookList);
* return book;
* }
* @Override
* public boolean deleteBook(long bookId) {
* boolean flag=bookList.removeIf(book->book.getBookId()==bookId);
* saveToBinaryFile(bookList);
* return flag;
* }
* @Override
* public Book searchBook(long bookId) {
* Book book=null;
* try {
* bookList=binaryBookStreamCRUD.readRecords();
* book= bookList.stream()
* .filter(b -> b.getBookId() == bookId).reduce((a, b) -> {
* throw new IllegalStateException("Multiple elements: " + a + ", " + b);
* }).get();
* }catch(Exception exception) {}
* return book;
* }
* public void saveToBinaryFile(List<Book> bookList) {
* try {
* binaryBookStreamCRUD.open();
* bookList.forEach(b->{
* binaryBookStreamCRUD.writeBook(b);
* });
* binaryBookStreamCRUD.close();
* } catch (Exception e) {
* e.printStackTrace();
* }
* }
* @Override
* public List<Book> fetchAllBook() {
* return bookList;
* }
* public BinaryBookStreamCRUD getBinaryBookStreamCRUD() {
* return binaryBookStreamCRUD;
* }
* public void setBinaryBookStreamCRUD(BinaryBookStreamCRUD binaryBookStreamCRUD) {
* this.binaryBookStreamCRUD = binaryBookStreamCRUD;
* }
* }

# SubjectRepository.java

* package com.cognizant.repository;
* import java.util.List;
* import com.cognizant.model.Subject;
* public interface SubjectRepository {
* public Subject addSubject(Subject subject);
* public boolean deleteSubject(long subjectId);
* public Subject searchSubject(long subjectId);
* public List<Subject> fetchAllSubject();
* }

# SubjectRepositoryImpl.java

* package com.cognizant.repository;
* import java.util.ArrayList;
* import java.util.List;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.stereotype.Repository;
* import com.cognizant.datastream.BinarySubjectStreamCRUD;
* import com.cognizant.model.Subject;
* @Repository("subjectRepository")
* public class SubjectRepositoryImpl implements SubjectRepository{
* private List<Subject> subjectList=new ArrayList<Subject>();
* private BinarySubjectStreamCRUD binarySubjectStreamCRUD;
* @Override
* public Subject addSubject(Subject subject) {
* subjectList.add(subject);
* saveToBinaryFile(subjectList);
* return subject;
* }
* @Override
* public boolean deleteSubject(long subjectId) {
* boolean flag=subjectList.removeIf(sub->sub.getSubjectId()==subjectId);
* saveToBinaryFile(subjectList);
* return flag;
* }
* @Override
* public Subject searchSubject(long subjectId) {
* Subject subject=null;
* try {
* subjectList=binarySubjectStreamCRUD.readRecords();
* System.out.println("subjectList "+subjectList.size());
* subject = subjectList.stream()
* .filter(sub -> sub.getSubjectId() == subjectId).reduce((a, b) -> {
* throw new IllegalStateException("Multiple elements: " + a + ", " + b);
* }).get();
* }catch(Exception exception) {}
* return subject;
* }
* public void saveToBinaryFile(List<Subject> subjectList) {
* try {
* binarySubjectStreamCRUD.open();
* subjectList.forEach(sub->{
* binarySubjectStreamCRUD.writeSubject(sub);
* });
* binarySubjectStreamCRUD.close();
* } catch (Exception e) {
* e.printStackTrace();
* }
* }
* @Override
* public List<Subject> fetchAllSubject() {
* return subjectList;
* }
* public BinarySubjectStreamCRUD getBinarySubjectStreamCRUD() {
* return binarySubjectStreamCRUD;
* }
* public void setBinarySubjectStreamCRUD(BinarySubjectStreamCRUD binarySubjectStreamCRUD) {
* this.binarySubjectStreamCRUD = binarySubjectStreamCRUD;
* }
* }

# BookService.java

* package com.cognizant.service;
* import java.util.List;
* import com.cognizant.model.Book;
* public interface BookService {
* public Book addBook(Book book);
* public boolean deleteBook(long bookId);
* public Book searchBook(long bookId);
* public List<Book> fetchAllBook();
* }

# BookServiceImpl.java

* package com.cognizant.service;
* import java.util.List;
* import com.cognizant.model.Book;
* import com.cognizant.repository.BookRepository;
* public class BookServiceImpl implements BookService {
* private BookRepository bookRepository;
* public void setBookRepository(BookRepository bookRepository) {
* this.bookRepository = bookRepository;
* }
* @Override
* public Book addBook(Book book) {
* return bookRepository.addBook(book);
* }
* @Override
* public boolean deleteBook(long bookId) {
* return bookRepository.deleteBook(bookId);
* }
* @Override
* public Book searchBook(long bookId) {
* return bookRepository.searchBook(bookId);
* }
* @Override
* public List<Book> fetchAllBook() {
* return bookRepository.fetchAllBook();
* }
* }

# SubjectService.java

* package com.cognizant.service;
* import java.util.List;
* import com.cognizant.model.Subject;
* public interface SubjectService {
* public Subject addSubject(Subject subject);
* public boolean deleteSubject(long subjectId);
* public Subject searchSubject(long subjectId);
* public List<Subject> fetchAllSubject();
* }

# SubjectServiceImpl.java

* package com.cognizant.service;
* import java.util.List;
* import org.springframework.stereotype.Service;
* import com.cognizant.model.Subject;
* import com.cognizant.repository.SubjectRepository;
* @Service("subjectService")
* public class SubjectServiceImpl implements SubjectService{
* private SubjectRepository subjectRepository;
* public void setSubjectRepository(SubjectRepository subjectRepository) {
* this.subjectRepository = subjectRepository;
* }
* @Override
* public Subject addSubject(Subject subject) {
* return subjectRepository.addSubject(subject);
* }
* @Override
* public boolean deleteSubject(long subjectId) {
* return subjectRepository.deleteSubject(subjectId);
* }
* @Override
* public Subject searchSubject(long subjectId) {
* return subjectRepository.searchSubject(subjectId);
* }
* @Override
* public List<Subject> fetchAllSubject() {
* return subjectRepository.fetchAllSubject();
* }
* }

# applicationContext.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd"*>

<bean id=*"subjectService"* class=*"com.cognizant.service.SubjectServiceImpl"*>

<property name=*"subjectRepository"* ref=*"subjectRepository"* />

</bean>

<bean id=*"subjectRepository"* class=*"com.cognizant.repository.SubjectRepositoryImpl"* >

<property name=*"binarySubjectStreamCRUD"* ref=*"binarySubjectStreamCRUD"*/>

</bean>

<bean id=*"bookService"* class=*"com.cognizant.service.BookServiceImpl"*>

<property name=*"bookRepository"* ref=*"bookRepository"* />

</bean>

<bean id=*"bookRepository"* class=*"com.cognizant.repository.BookRepositoryImpl"* >

<property name=*"binaryBookStreamCRUD"* ref=*"binaryBookStreamCRUD"*/>

</bean>

<bean id=*"commandLineRunner"* class=*"com.cognizant.CommandLineRunner"*>

<property name=*"subjectService"* ref=*"subjectService"* />

<property name=*"bookService"* ref=*"bookService"* />

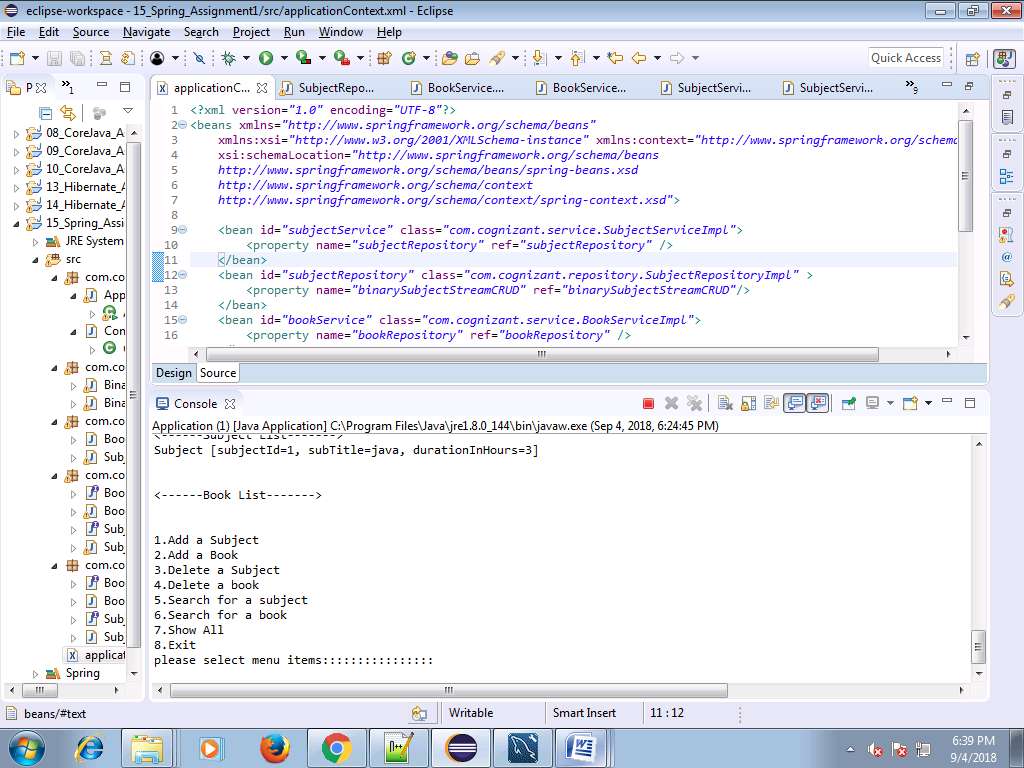
</bean>

<bean id=*"binaryBookStreamCRUD"* class=*"com.cognizant.datastream.BinaryBookStreamCRUD"*/>

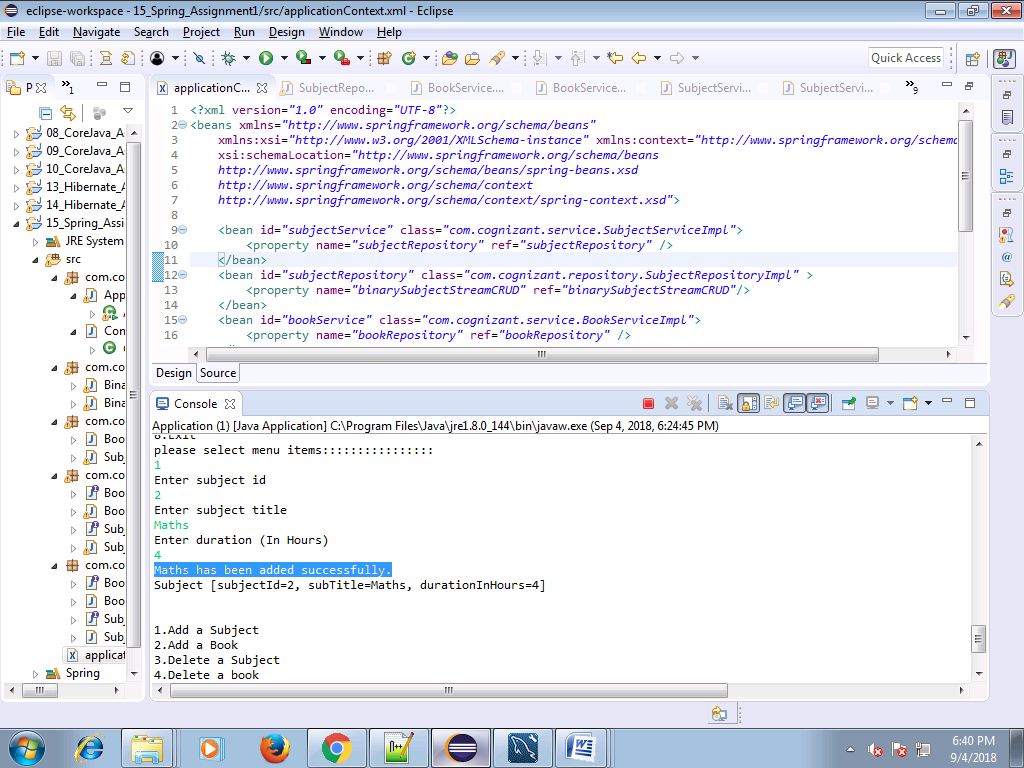
<bean id=*"binarySubjectStreamCRUD"* class=*"com.cognizant.datastream.BinarySubjectStreamCRUD"*/>

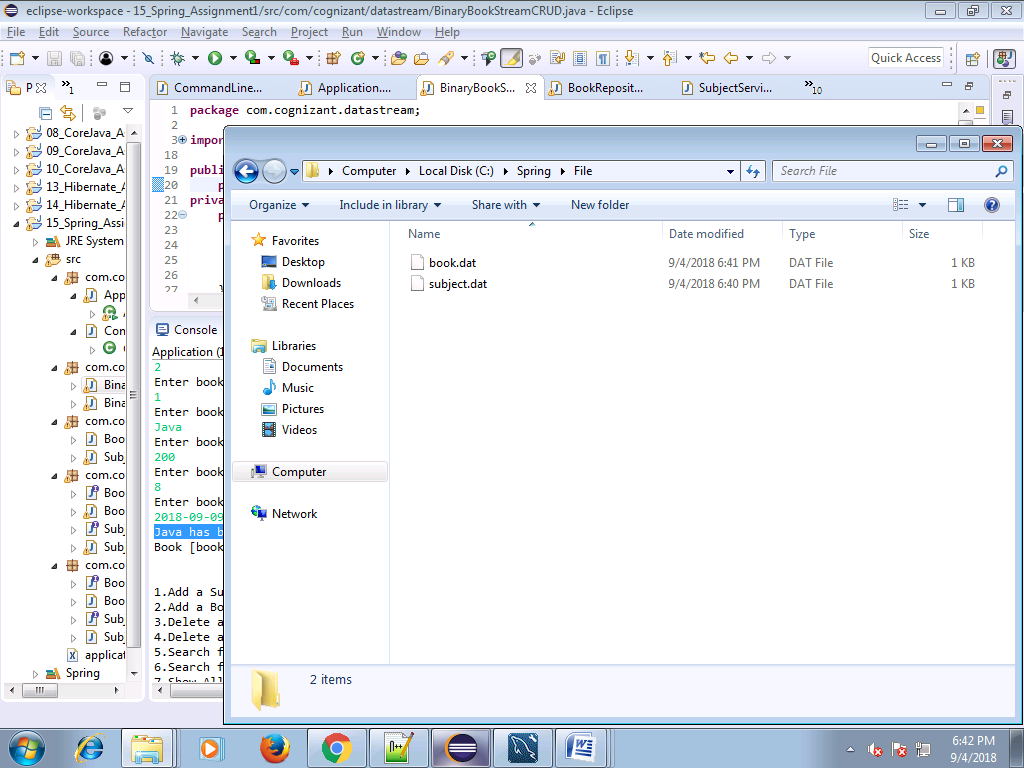
</beans>

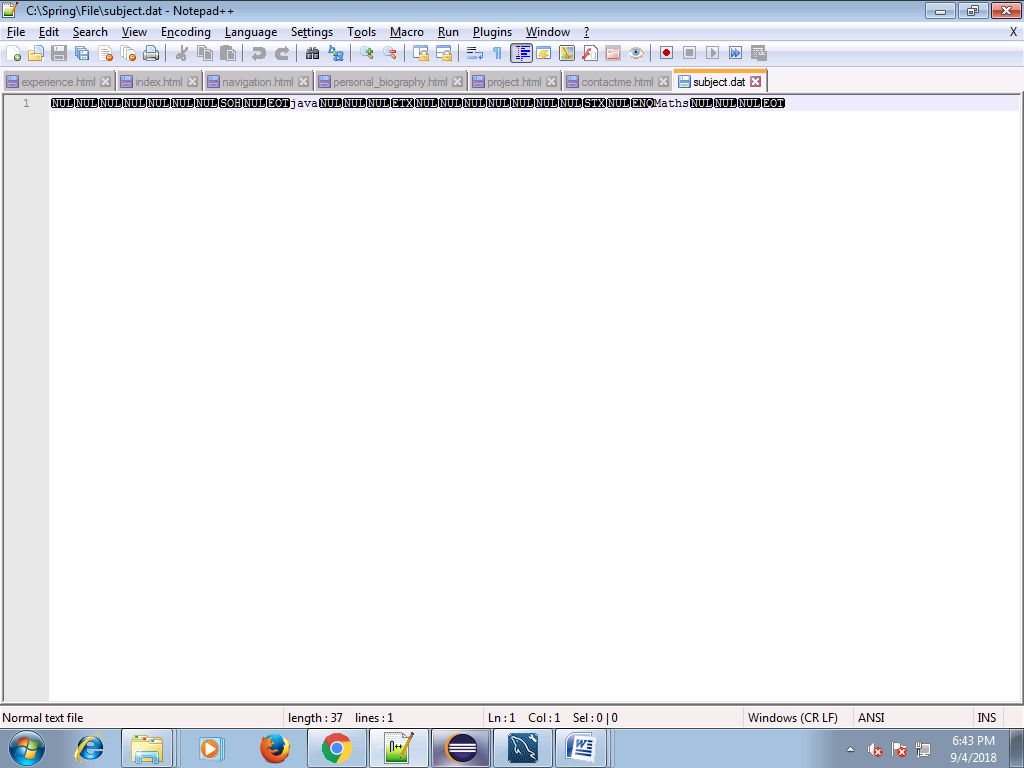
**Screenshots**

****

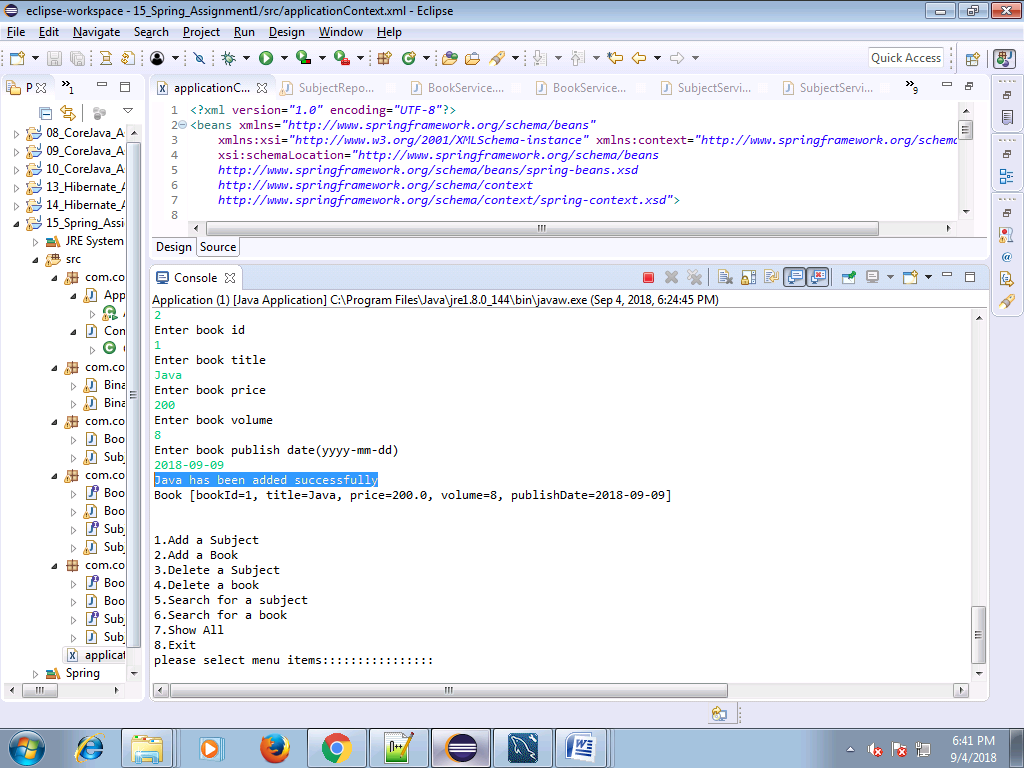
**Add a Subject**

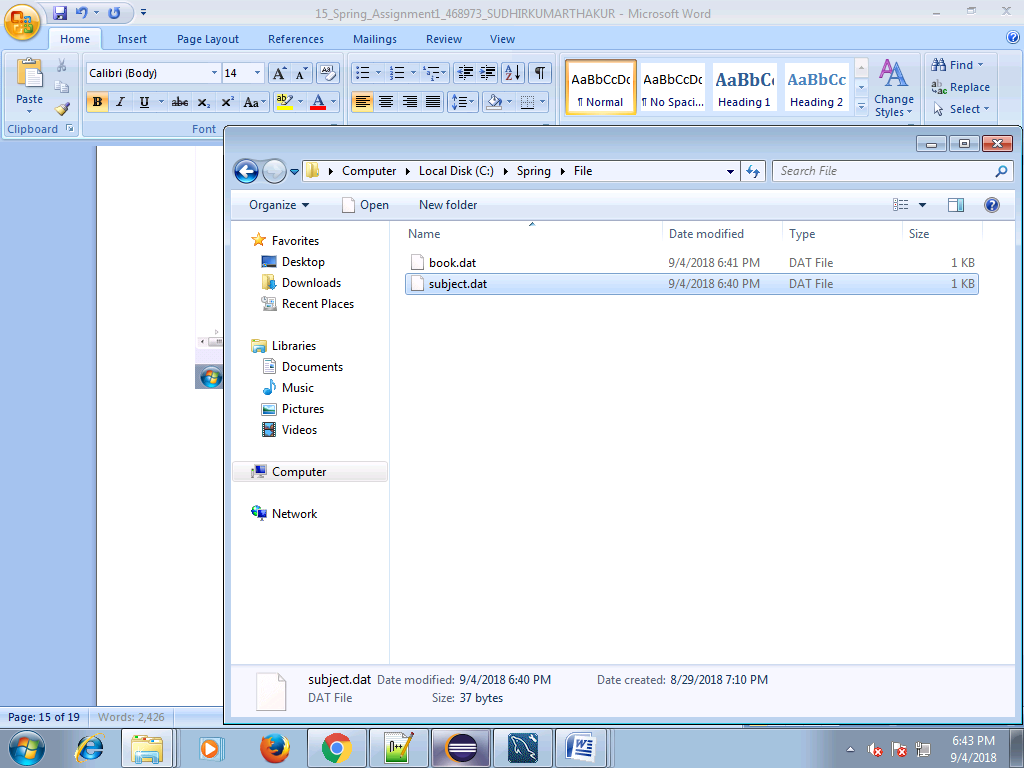
****

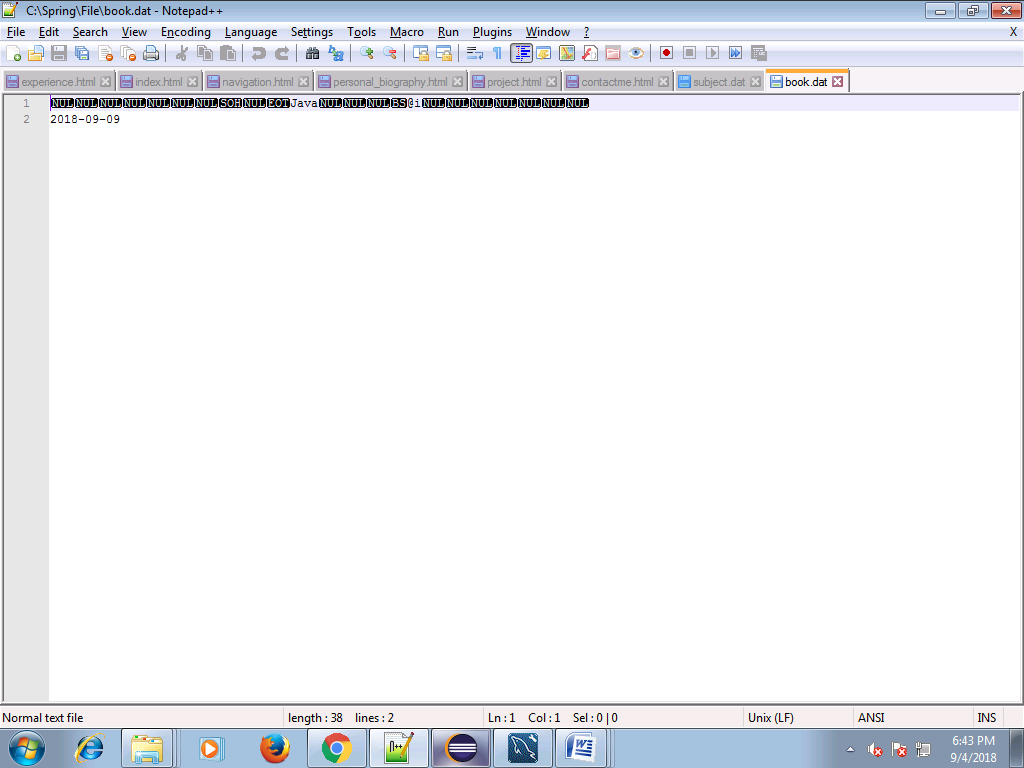
****

****

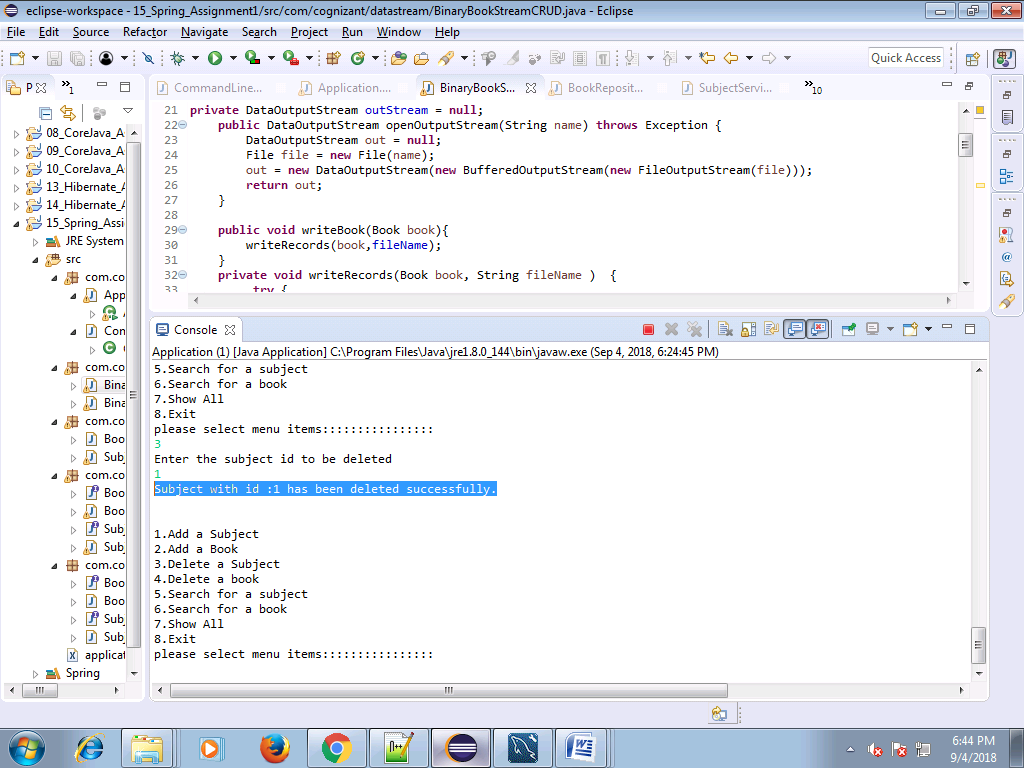
**Add a Book**

****

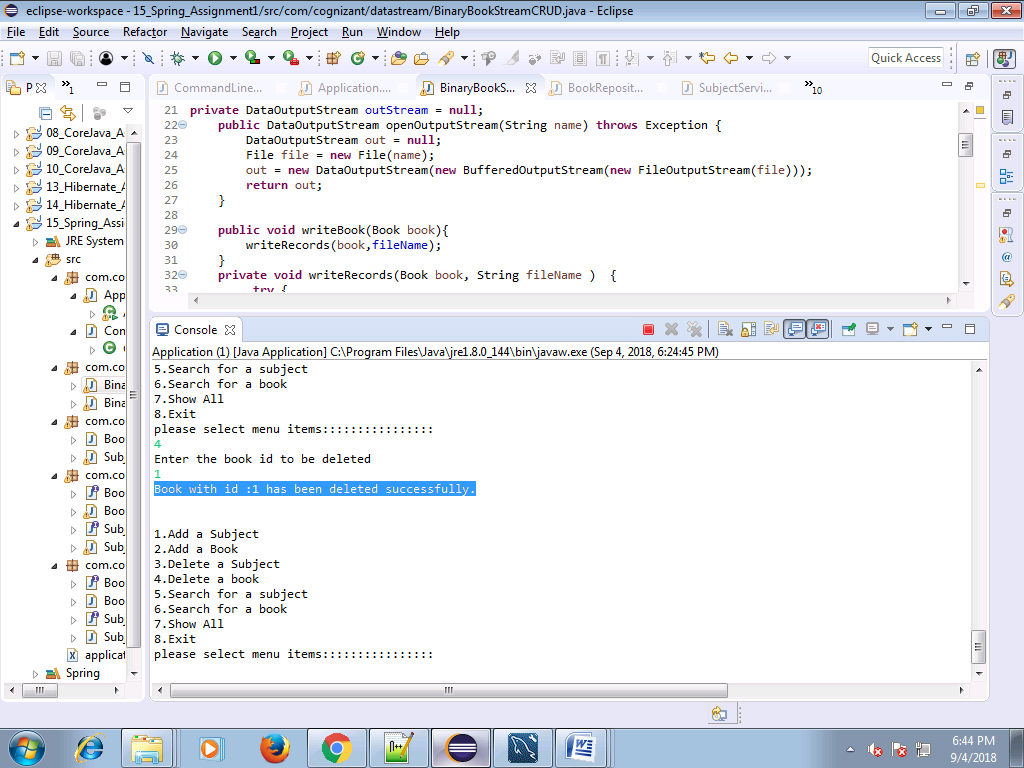
****

****

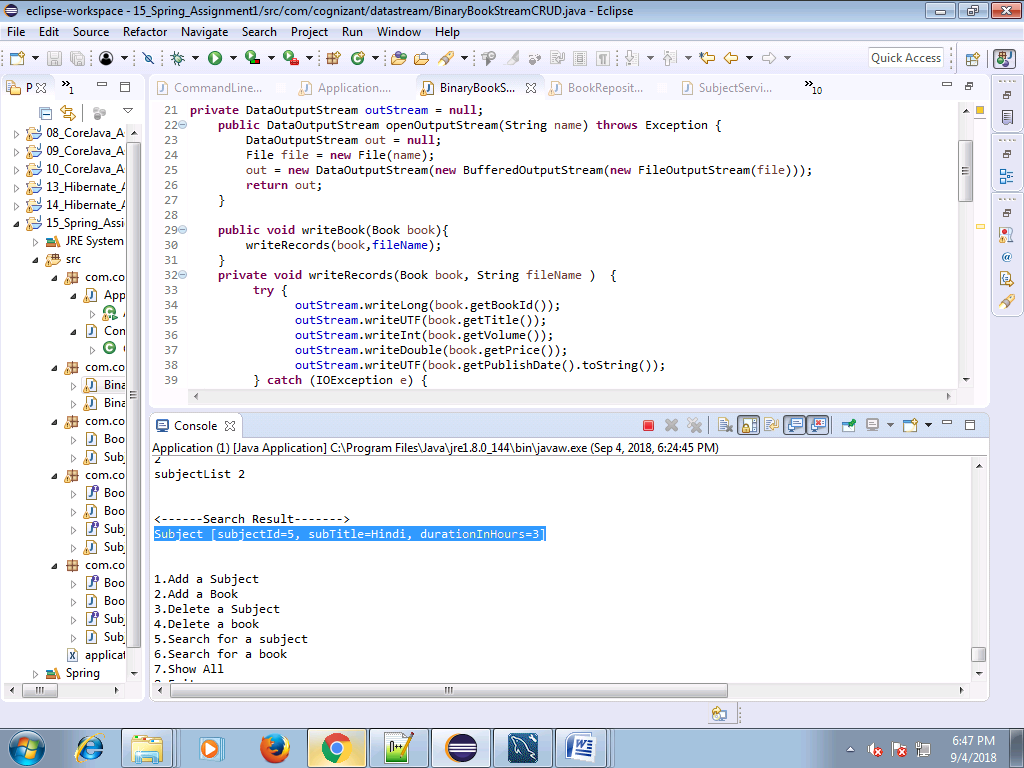
**Delete a subject**

****

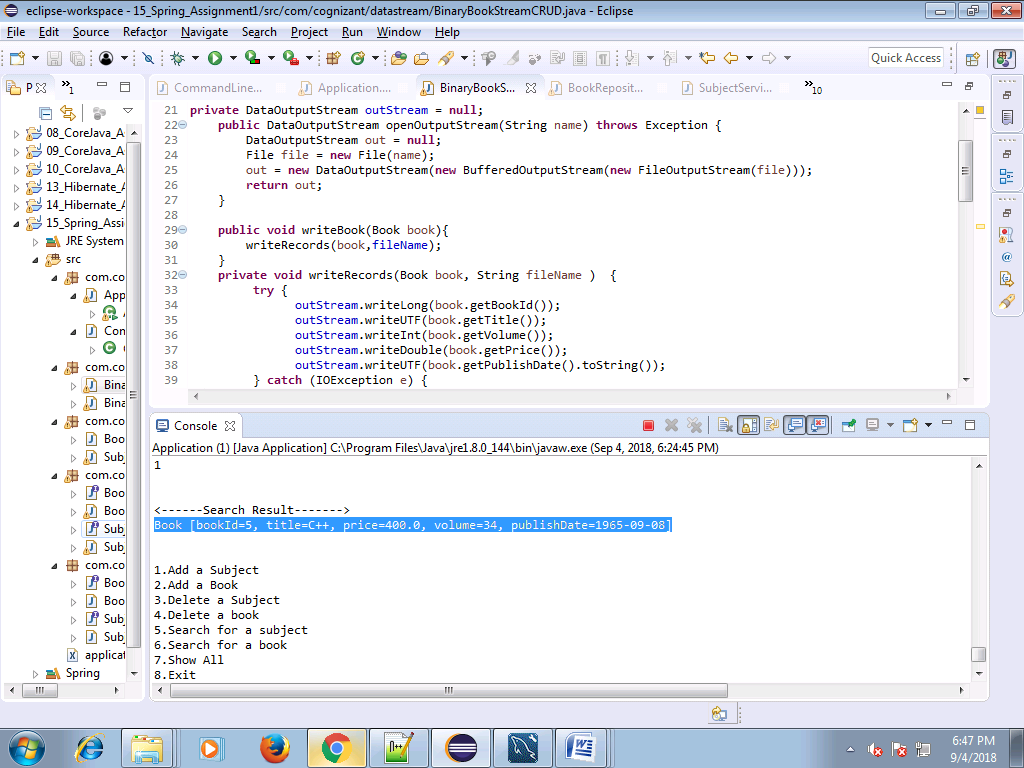
**Delete a book**

****

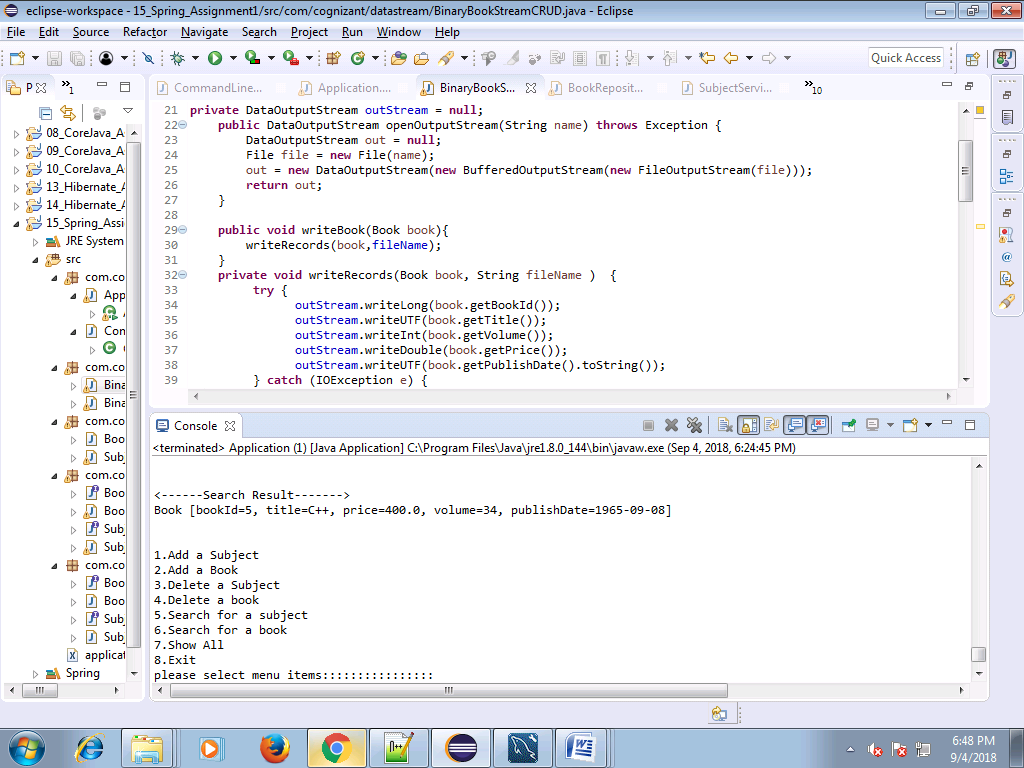
**Search for a subject**

****

**Search for a book**

****

**Exit**

****