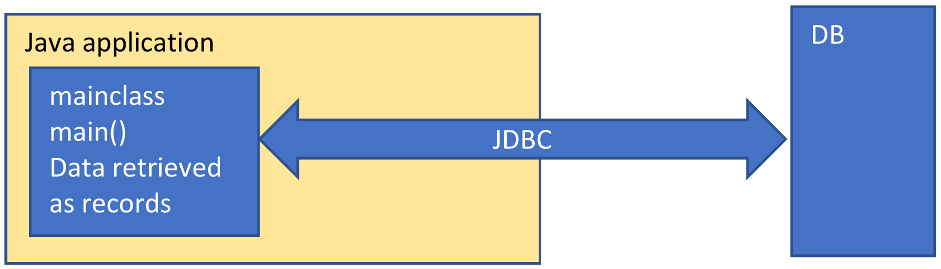
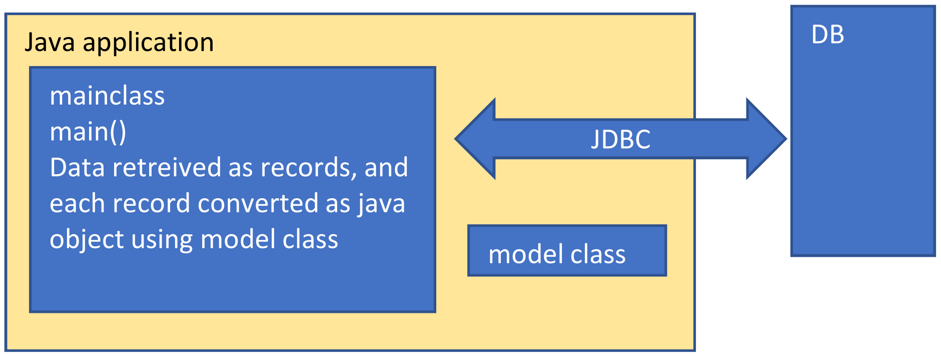
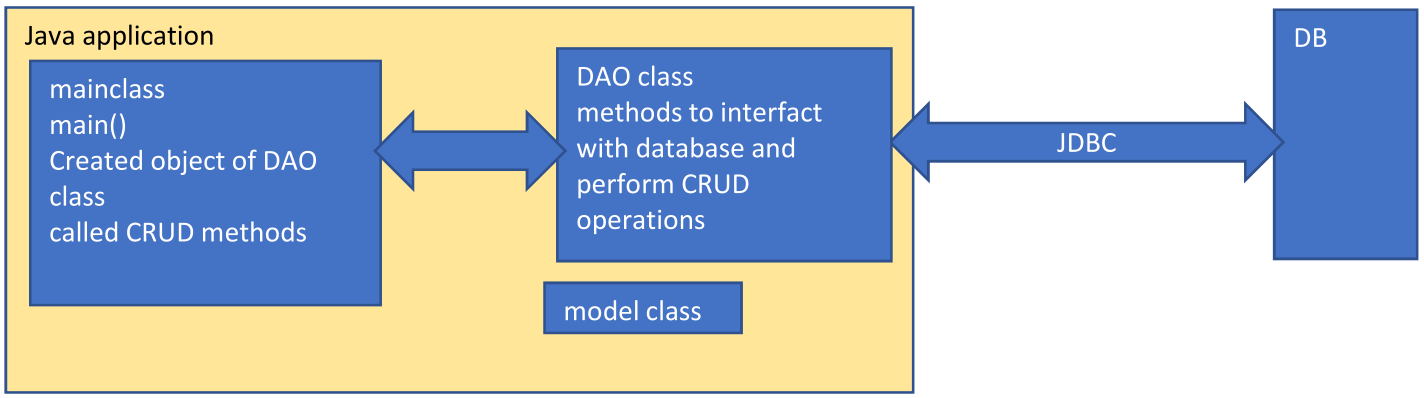
JDBC example
in corejava

Java Application <-> DB (MySQL)





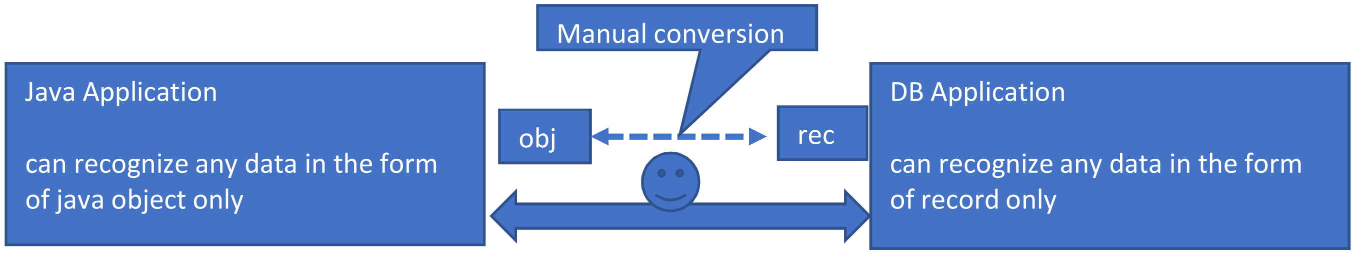


Methods in DAO

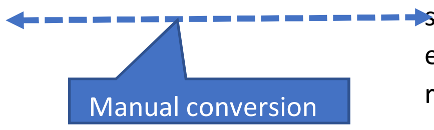
```
public boolean addBook(Book book){  }
public List<Book> getBooks() { }
public boolean updateBook(Book book){ }
public boolean deleteBook(int bookid){ }
public Book getBookById(int bookid) { }
```

load driver
statement (query)
fill params in prepared st (query)
execute respective method (ddl/dml/dql)
convert rec type data into object (viceversa)

In JDBC program
Table in DB and class in Java are mapped manually


Manual conversion
Java Application
can recognize any data in the form of java object only
obj
rec
DB Application
can recognize any data in the form of record only

Structure of data decided by model class
every data stored as object
object has member variables inside



structure of data decided by table
every data stored as record
record has fields

ORM tool
Hibernate

Advantages
1 java model class(es) are maped with table(s) in database automatically
2 rec type data <-> object type
3 comes with collection of methods to perform different operations on db

ex	select * from ….	delete from …	
	list()	delete()	select * from __ where bid =__
	insert into ….	update table set ….	get() /load()
	save()	update()	

SessinFactory

Used to provide sessions for programmers

Session

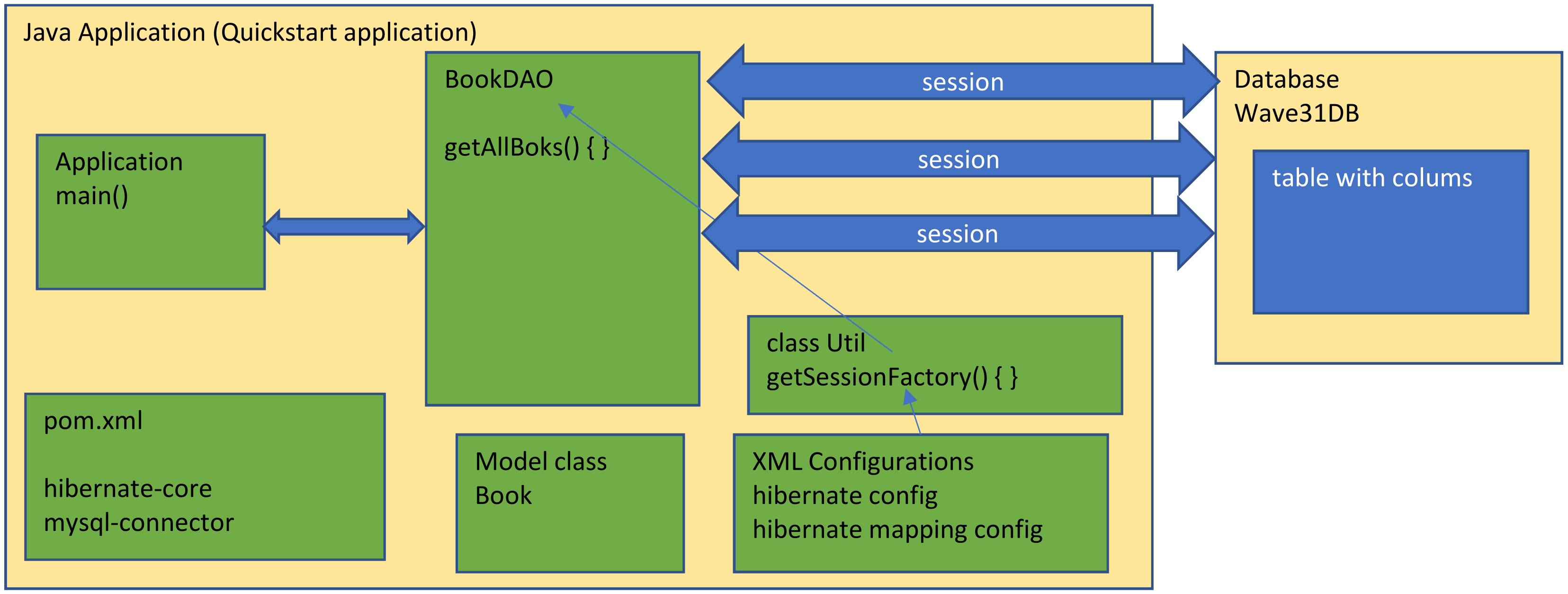
Temporary object for programmer to perfrom any CRUD opertation with database

Hibernate configuration

driver
db url, username, password
show queries / dialect

Hibernate mapping configuration

model <-> table

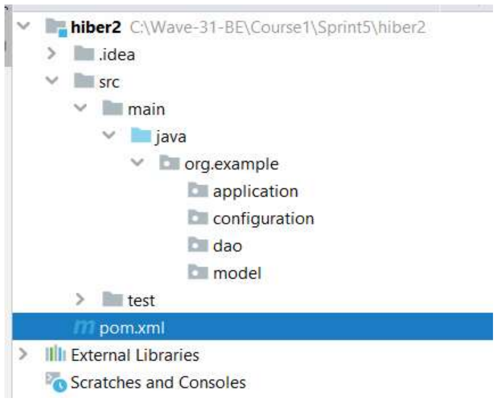


Steps to create new maven application, add hibernate

1 Create new Maven quickstart application

create required packages

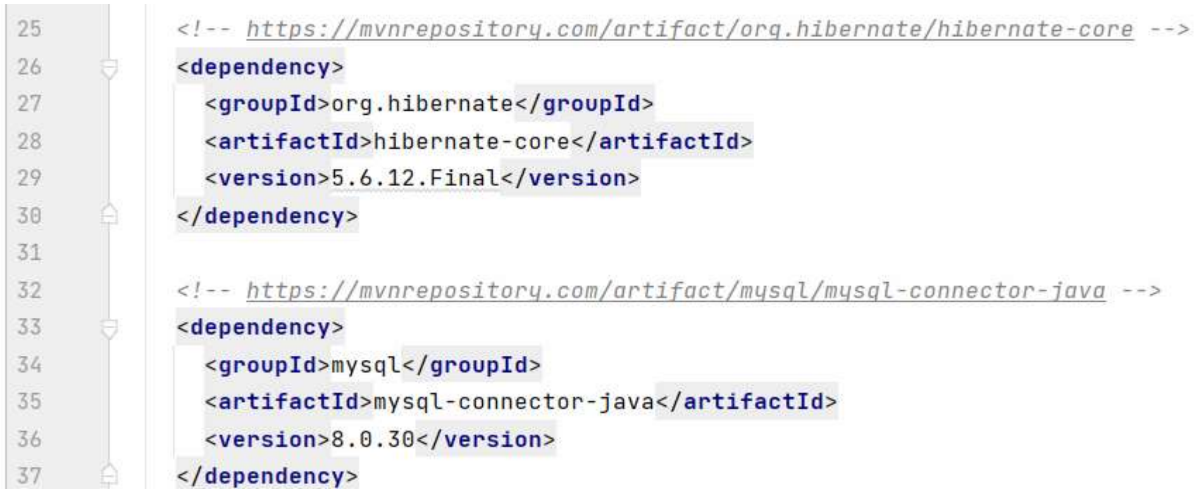
application/model/configuration/repository(dao)



2 add required dependencies

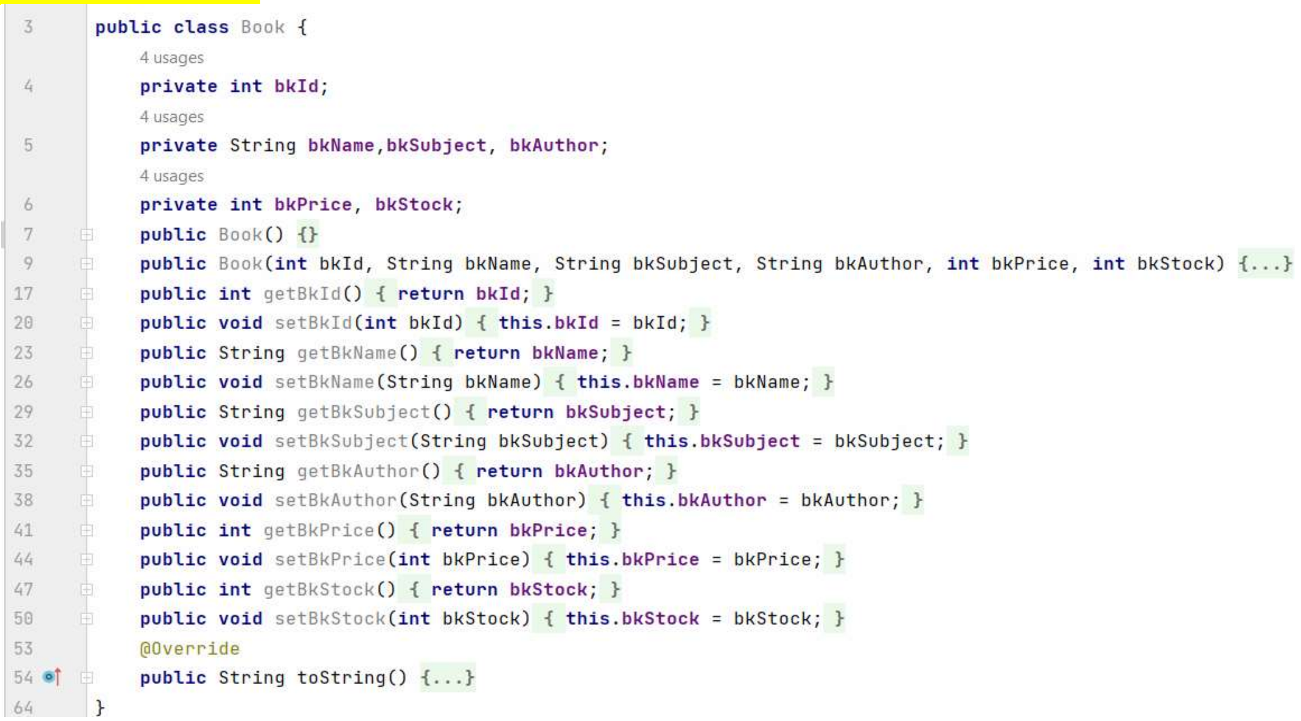
hibernate-core

mysql-connector 8.0.30 C:\Program Files (x86)\MySQL\Connector J 8.0



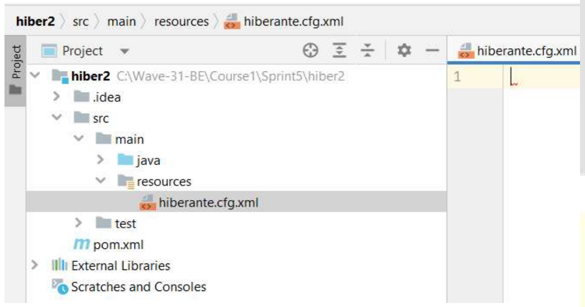
```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.12.Final</version>
</dependency>

<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.30</version>
</dependency>
```

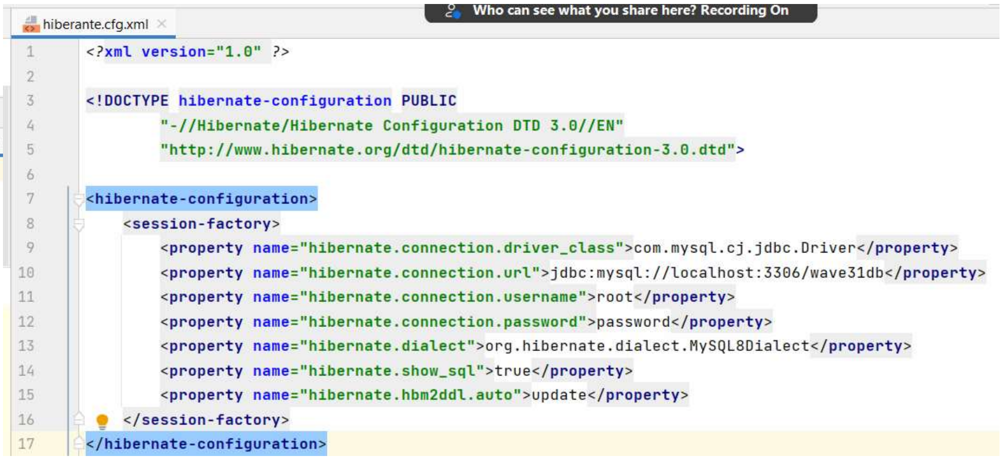
3 Create model class



```java
public class Book {
    4 usages
    private int bkId;
    4 usages
    private String bkName,bkSubject, bkAuthor;
    4 usages
    private int bkPrice, bkStock;
    public Book() {}
    public Book(int bkId, String bkName, String bkSubject, String bkAuthor, int bkPrice, int bkStock) {...}
    public int getBkId() { return bkId; }
    public void setBkId(int bkId) { this.bkId = bkId; }
    public String getBkName() { return bkName; }
    public void setBkName(String bkName) { this.bkName = bkName; }
    public String getBkSubject() { return bkSubject; }
    public void setBkSubject(String bkSubject) { this.bkSubject = bkSubject; }
    public String getBkAuthor() { return bkAuthor; }
    public void setBkAuthor(String bkAuthor) { this.bkAuthor = bkAuthor; }
    public int getBkPrice() { return bkPrice; }
    public void setBkPrice(int bkPrice) { this.bkPrice = bkPrice; }
    public int getBkStock() { return bkStock; }
    public void setBkStock(int bkStock) { this.bkStock = bkStock; }
    @Override
    public String toString() {...}
}
```

4 Create hibernate.cfg.xml under resources folder, define properties for session factory

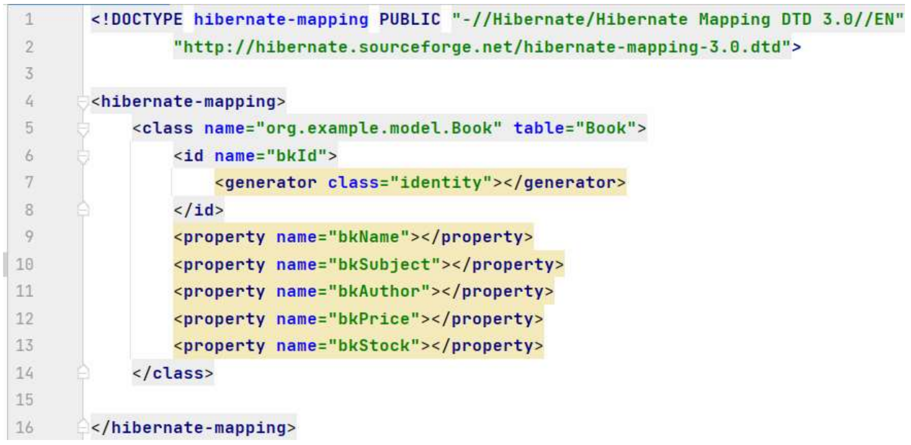




```xml
<?xml version="1.0" ?>

<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/wave31db</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">password</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
    </session-factory>
</hibernate-configuration>
```

5 create hbm.xml file under resources folder

define model class maping with table

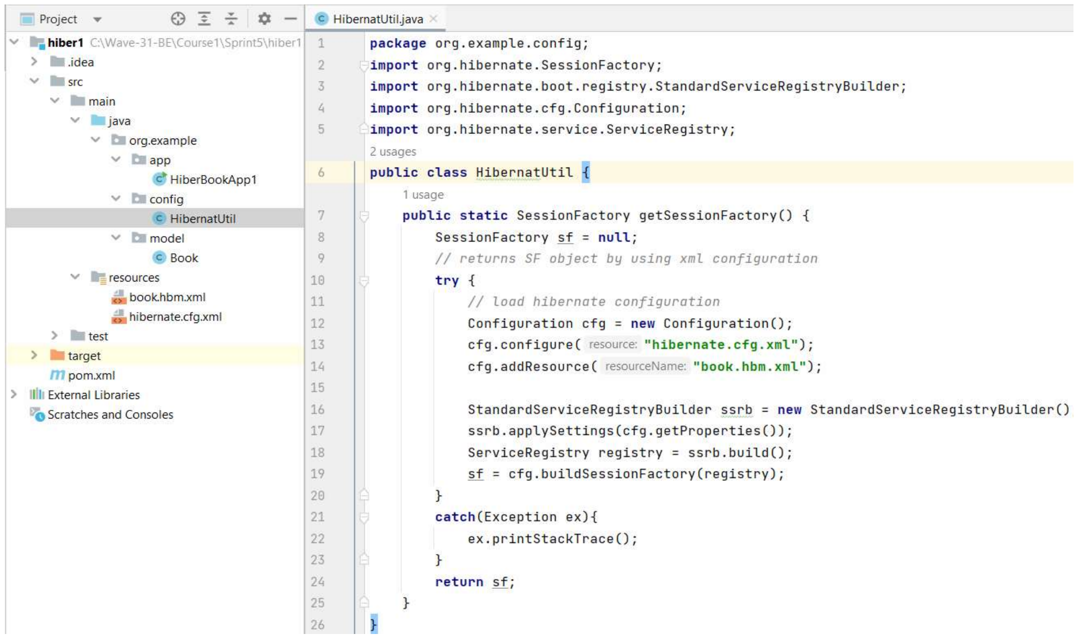


```xml
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="org.example.model.Book" table="Book">
        <id name="bkId">
            <generator class="identity"></generator>
        </id>
        <property name="bkName"></property>
        <property name="bkSubject"></property>
        <property name="bkAuthor"></property>
        <property name="bkPrice"></property>
        <property name="bkStock"></property>
    </class>

</hibernate-mapping>
```

6 Define util class under configuration packege

define method to gernerate and return SessionFactory object

by using xml file configurations



Project | HibernatUtil.java

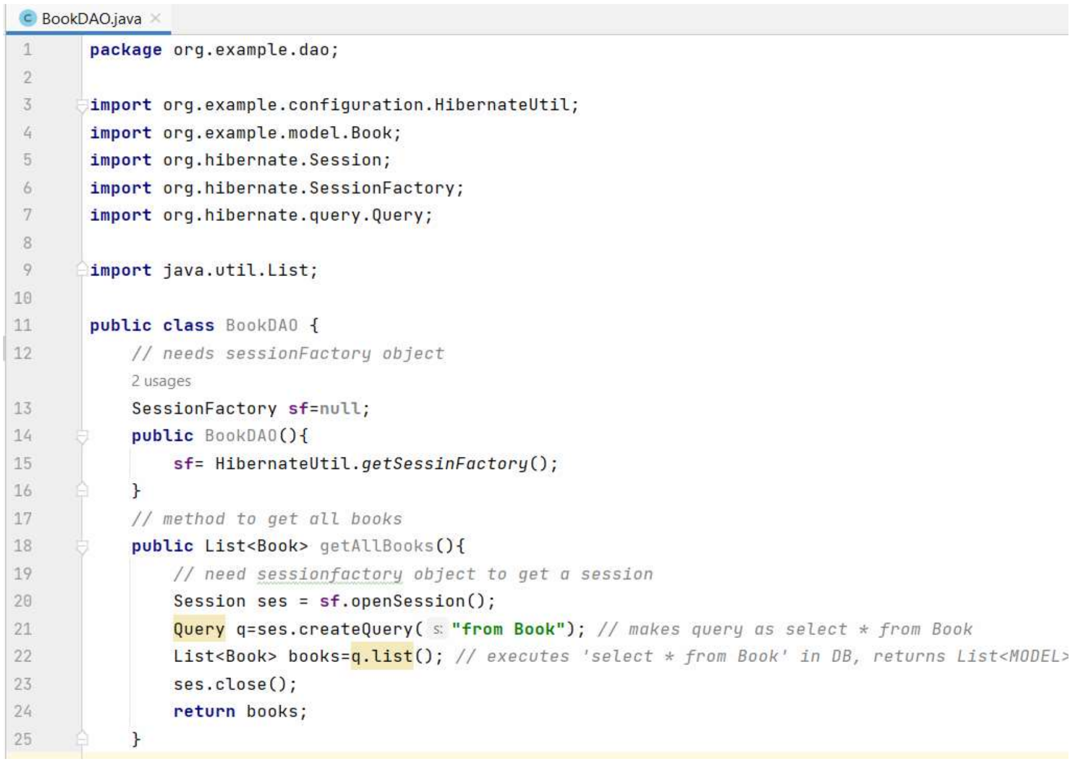
```
hiber1 C:\Wave-31-BE\Course1\Sprint5\hiber1
  .idea
  src
    main
      java
        org.example
          app
            HiberBookApp1
          config
            HibernatUtil
          model
            Book
      resources
        book.hbm.xml
        hibernate.cfg.xml
    test
  target
  pom.xml
External Libraries
Scratches and Consoles
```

```java
package org.example.config;
import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;
2 usages
public class HibernatUtil {
    1 usage
    public static SessionFactory getSessionFactory() {
        SessionFactory sf = null;
        // returns SF object by using xml configuration
        try {
            // load hibernate configuration
            Configuration cfg = new Configuration();
            cfg.configure( resource: "hibernate.cfg.xml");
            cfg.addResource( resourceName: "book.hbm.xml");

            StandardServiceRegistryBuilder ssrb = new StandardServiceRegistryBuilder()
            ssrb.applySettings(cfg.getProperties());
            ServiceRegistry registry = ssrb.build();
            sf = cfg.buildSessionFactory(registry);
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
        return sf;
    }
}
```

7 Make DAO/Repository layer

inject SessionFactory dependency

define getAllBooks() using sessionFactory object

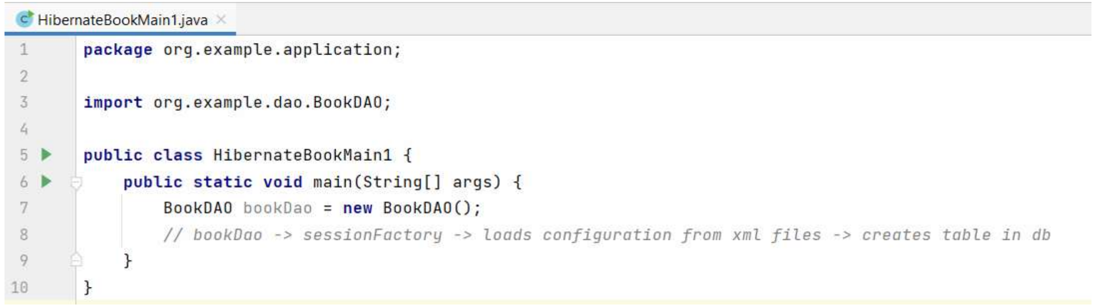

BookDAO.java

```java
package org.example.dao;

import org.example.configuration.HibernateUtil;
import org.example.model.Book;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;

import java.util.List;

public class BookDAO {
    // needs sessionFactory object
    2 usages
    SessionFactory sf=null;
    public BookDAO(){
        sf= HibernateUtil.getSessinFactory();
    }
    // method to get all books
    public List<Book> getAllBooks(){
        // need sessionfactory object to get a session
        Session ses = sf.openSession();
        Query q=ses.createQuery( s: "from Book"); // makes query as select * from Book
        List<Book> books=q.list(); // executes 'select * from Book' in DB, returns List<MODEL>
        ses.close();
        return books;
    }
```

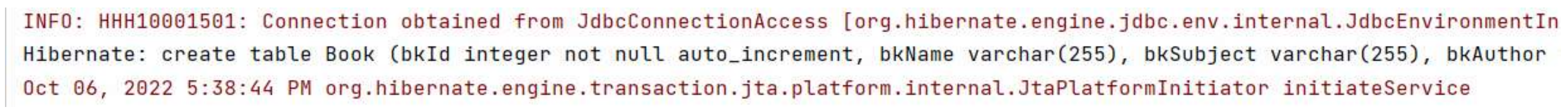
8 Define application class

define main()

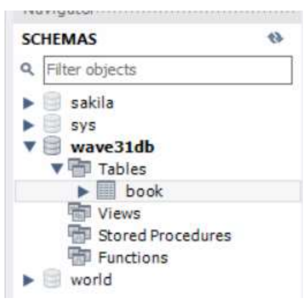
create DAO object   Make sure hiberate is creating table in db as per model class


HibernateBookMain1.java

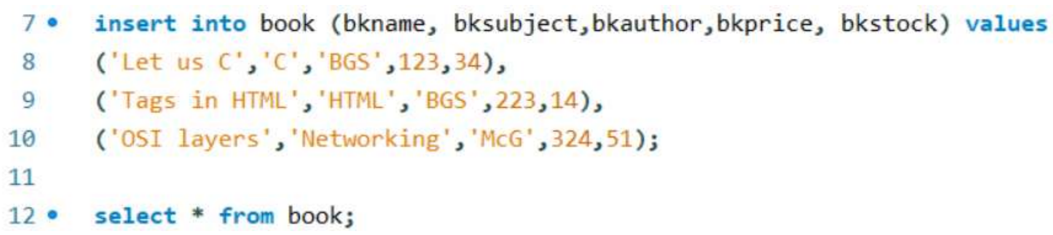
```java
package org.example.application;

import org.example.dao.BookDAO;

public class HibernateBookMain1 {
    public static void main(String[] args) {
        BookDAO bookDao = new BookDAO();
        // bookDao -> sessionFactory -> loads configuration from xml files -> creates table in db
    }
}
```



```
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentIn
Hibernate: create table Book (bkId integer not null auto_increment, bkName varchar(255), bkSubject varchar(255), bkAuthor
Oct 06, 2022 5:38:44 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
```

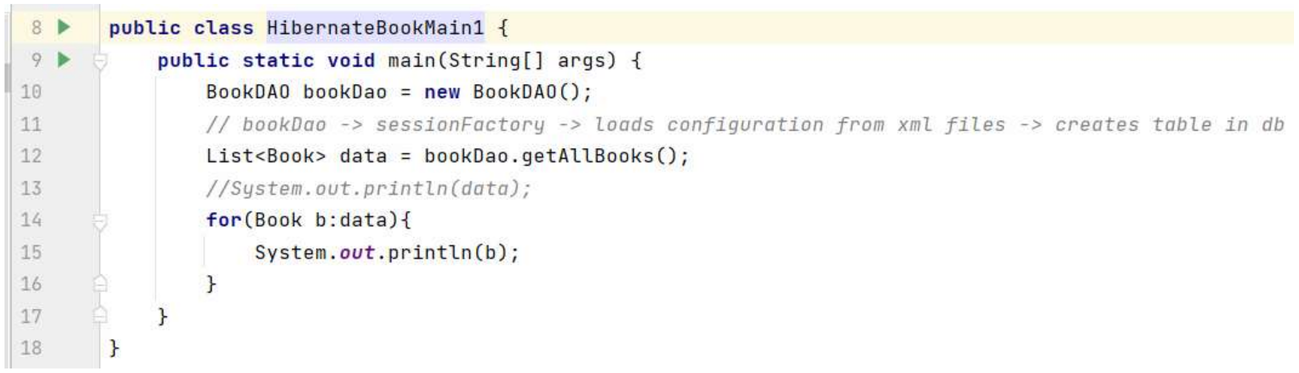


insert few records

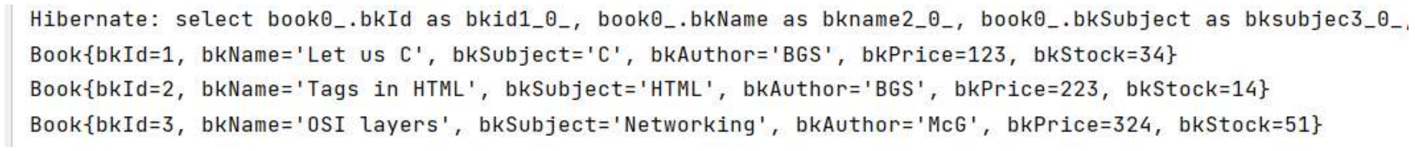


```sql
insert into book (bkname, bksubject,bkauthor,bkprice, bkstock) values
('Let us C','C','BGS',123,34),
('Tags in HTML','HTML','BGS',223,14),
('OSI layers','Networking','McG',324,51);

select * from book;
```

9 in main()
call dao.getAllBooks()



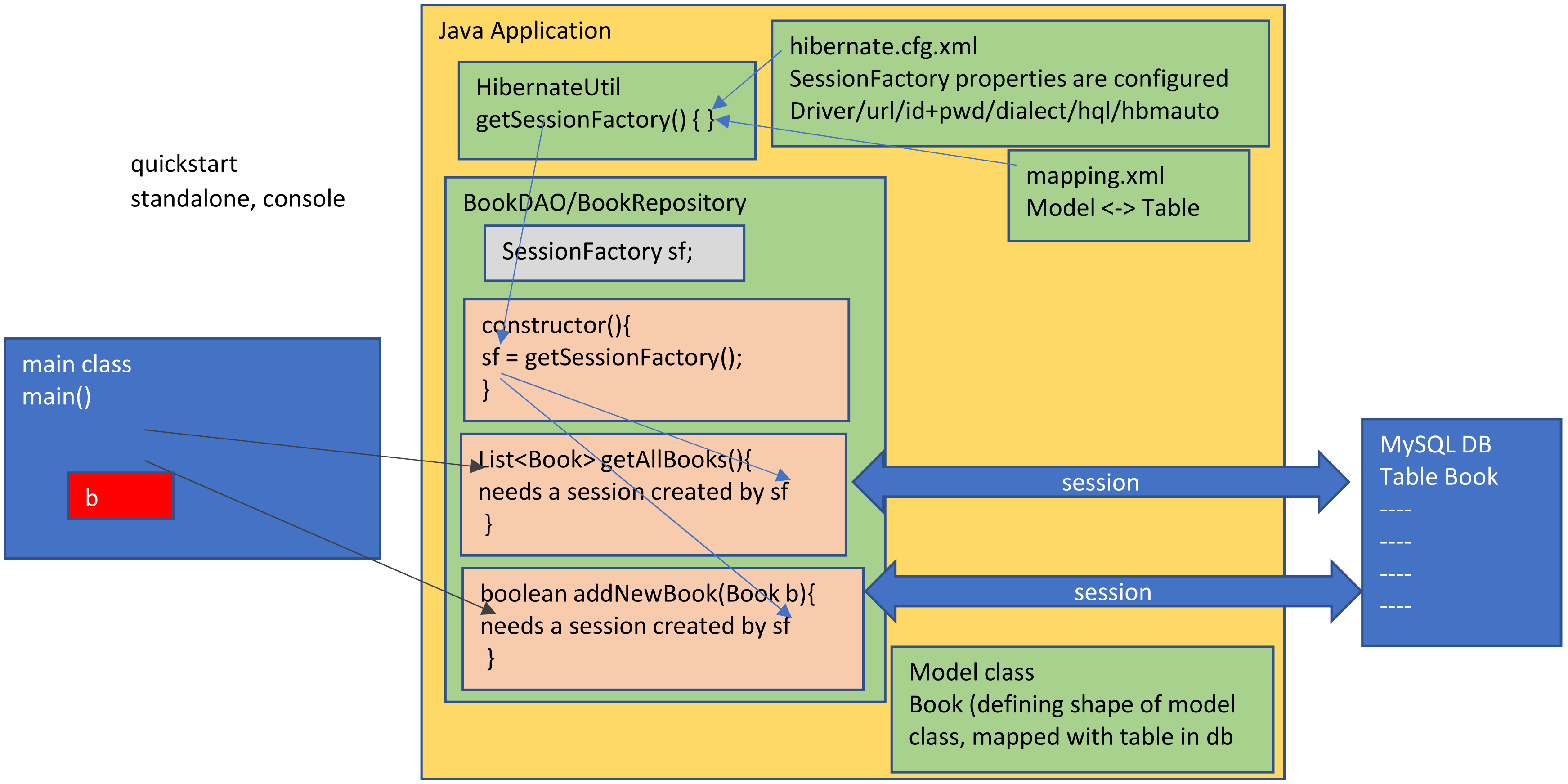
```java
public class HibernateBookMain1 {
    public static void main(String[] args) {
        BookDAO bookDao = new BookDAO();
        // bookDao -> sessionFactory -> loads configuration from xml files -> creates table in db
        List<Book> data = bookDao.getAllBooks();
        //System.out.println(data);
        for(Book b:data){
            System.out.println(b);
        }
    }
}
```



```
Hibernate: select book0_.bkId as bkid1_0_, book0_.bkName as bkname2_0_, book0_.bkSubject as bksubjec3_0_,
Book{bkId=1, bkName='Let us C', bkSubject='C', bkAuthor='BGS', bkPrice=123, bkStock=34}
Book{bkId=2, bkName='Tags in HTML', bkSubject='HTML', bkAuthor='BGS', bkPrice=223, bkStock=14}
Book{bkId=3, bkName='OSI layers', bkSubject='Networking', bkAuthor='McG', bkPrice=324, bkStock=51}
```

DAY2

quickstart
standalone, console



Java Application
HibernateUtil
getSessionFactory() { }
hibernate.cfg.xml
SessionFactory properties are configured
Driver/url/id+pwd/dialect/hql/hbmauto
mapping.xml
Model <-> Table
BookDAO/BookRepository
SessionFactory sf;
constructor(){
sf = getSessionFactory();
}
List<Book> getAllBooks(){
needs a session created by sf
}
boolean addNewBook(Book b){
needs a session created by sf
}
Model class
Book (defining shape of model class, mapped with table in db
main class
main()
b
session
session
MySQL DB
Table Book

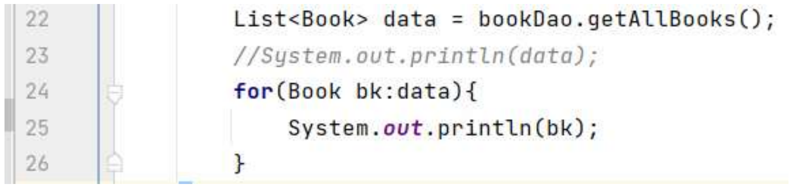
----

get all records
add new record
update record
delete record
get record by id

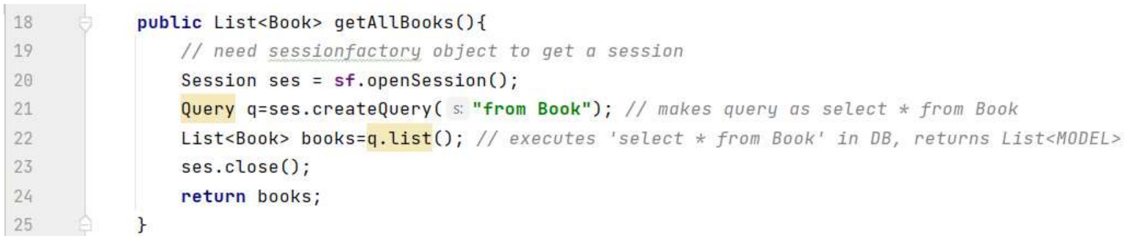
BookDAO methods

public List<Book> getAllBooks() {   }

gets all book records, returns as collection on book objects



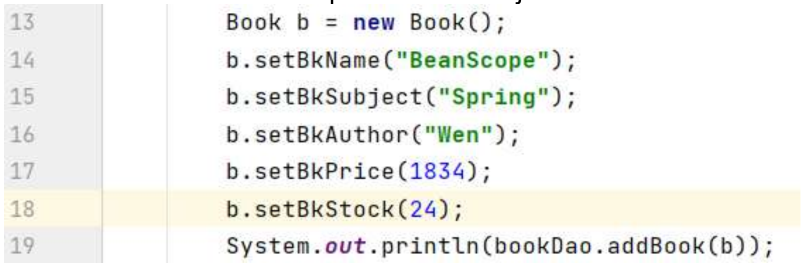
```java
List<Book> data = bookDao.getAllBooks();
//System.out.println(data);
for(Book bk:data){
    System.out.println(bk);
}
```



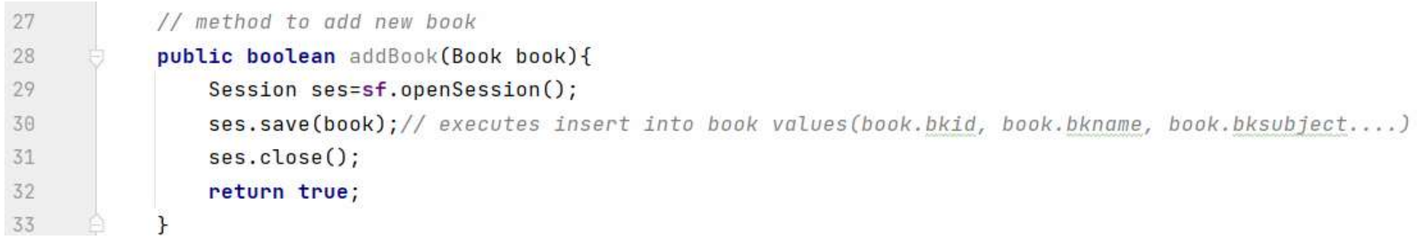
```java
public List<Book> getAllBooks(){
    // need sessionfactory object to get a session
    Session ses = sf.openSession();
    Query q=ses.createQuery( s: "from Book"); // makes query as select * from Book
    List<Book> books=q.list(); // executes 'select * from Book' in DB, returns List<MODEL>
    ses.close();
    return books;
}
```

public boolean addBook(Book book) { }

adds passed book object data as record in table



```java
Book b = new Book();
b.setBkName("BeanScope");
b.setBkSubject("Spring");
b.setBkAuthor("Wen");
b.setBkPrice(1834);
b.setBkStock(24);
System.out.println(bookDao.addBook(b));
```



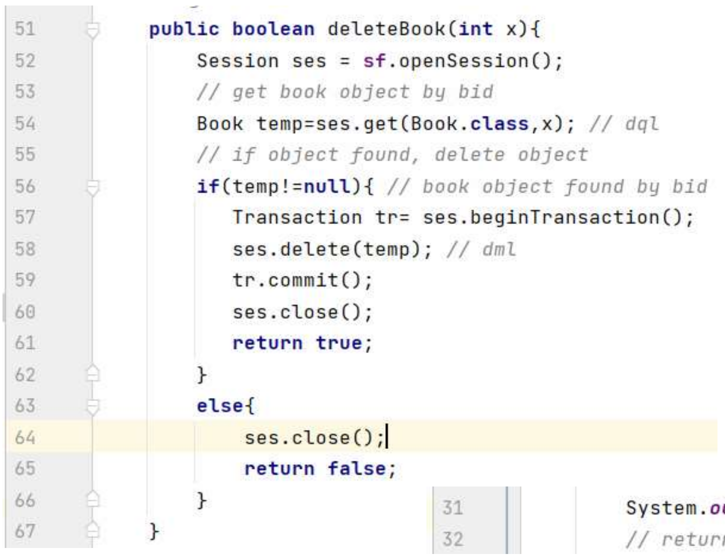
```java
// method to add new book
public boolean addBook(Book book){
    Session ses=sf.openSession();
    ses.save(book);// executes insert into book values(book.bkid, book.bkname, book.bksubject....)
    ses.close();
    return true;
}
```

public boolean deleteBook(int bid) { } 1

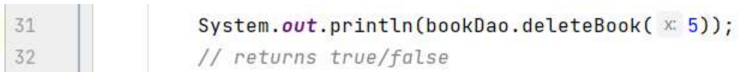
deletes book record based on passed id

get book object by id
if book object found
        ses.delete(object)

select * from book where bkid = __
list()



```
public boolean deleteBook(int x){
    Session ses = sf.openSession();
    // get book object by bid
    Book temp=ses.get(Book.class,x); // dql
    // if object found, delete object
    if(temp!=null){ // book object found by bid
       Transaction tr= ses.beginTransaction();
       ses.delete(temp); // dml
       tr.commit();
       ses.close();
       return true;
    }
    else{
        ses.close();
        return false;
    }
}
```

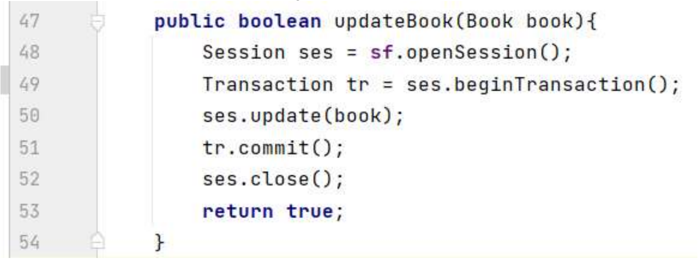


```
System.out.println(bookDao.deleteBook( x: 5));
// returns true/false
```

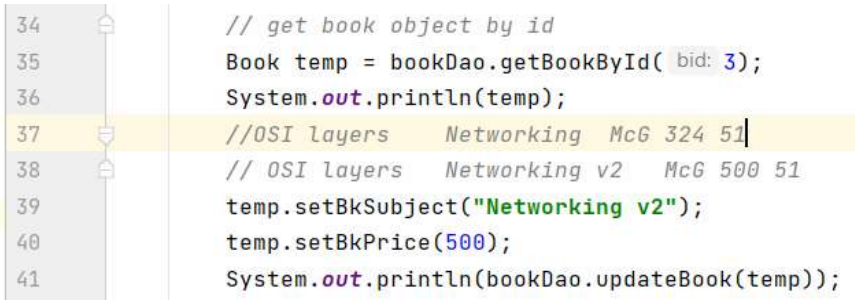
public boolean updateBook(Book book) { }

updates passed book object into db

get complete object which to be edited
update required fields in received object
send modilfied object to dao



```
public boolean updateBook(Book book){
    Session ses = sf.openSession();
    Transaction tr = ses.beginTransaction();
    ses.update(book);
    tr.commit();
    ses.close();
    return true;
}
```



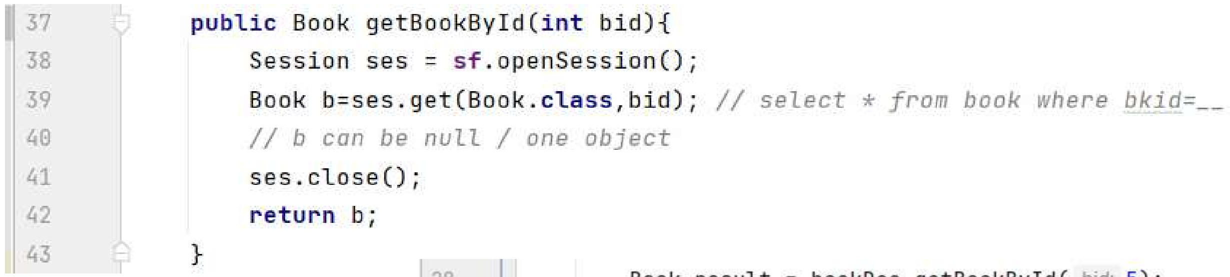
```
// get book object by id
Book temp = bookDao.getBookById( bid: 3);
System.out.println(temp);
//OSI layers    Networking   McG 324 51
// OSI layers    Networking v2   McG 500 51
temp.setBkSubject("Networking v2");
temp.setBkPrice(500);
System.out.println(bookDao.updateBook(temp));
```

public Book getBookById(int bkid) { }

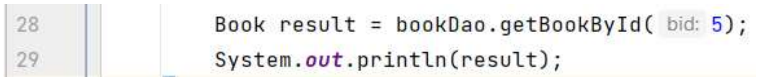
returns book object by filtering by passed id

ses.get()   returns null if object not found by id

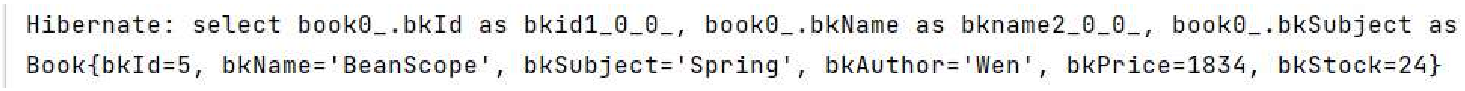
ses.load()   throws exception if object not found by id



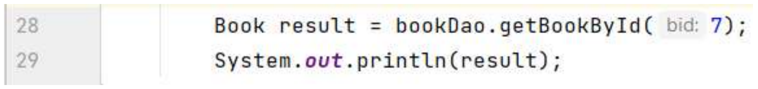
```
public Book getBookById(int bid){
    Session ses = sf.openSession();
    Book b=ses.get(Book.class,bid); // select * from book where bkid=__
    // b can be null / one object
    ses.close();
    return b;
}
```



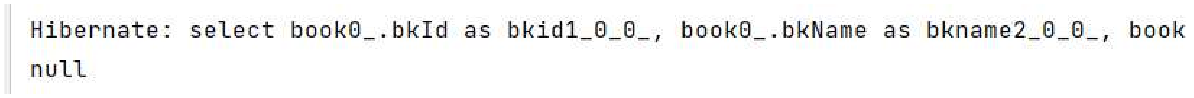
```
Book result = bookDao.getBookById( bid: 5);
System.out.println(result);
```



```
Hibernate: select book0_.bkId as bkid1_0_0_, book0_.bkName as bkname2_0_0_, book0_.bkSubject as
Book{bkId=5, bkName='BeanScope', bkSubject='Spring', bkAuthor='Wen', bkPrice=1834, bkStock=24}
```



```
Book result = bookDao.getBookById( bid: 7);
System.out.println(result);
```



```
Hibernate: select book0_.bkId as bkid1_0_0_, book0_.bkName as bkname2_0_0_, book
null
```

Note

when performing DML operations

start transaction

call dml method thru session

commit transaction

Common operations to be performed in Java collections

get all items from collection
add new item to collection
update an item in collection
delete an item from collection
get filtered items from collection
sort : a particular field
sort : any random field

```
          song
          id,name,duration,artist

List<Song> songs = new ArrayList<Song>();
          songs.add(s1);              S001, "abcd","1:2:2","xyz"
          songs.add(s2);              S002, "mnop","1:2:2","ab"
          songs.add(s3);              S003, "pqrs","1:2:2","mn"
          songs.add(s4);              S004, "ijkl","1:2:2","xy"
```

I need to remove song object from above collection which has song name as "pqrs"

```
filter song objects by name
    if any song found as name matching with "pqrs"
        fetch complete song object, store to temp

songs.remove(temp)
```