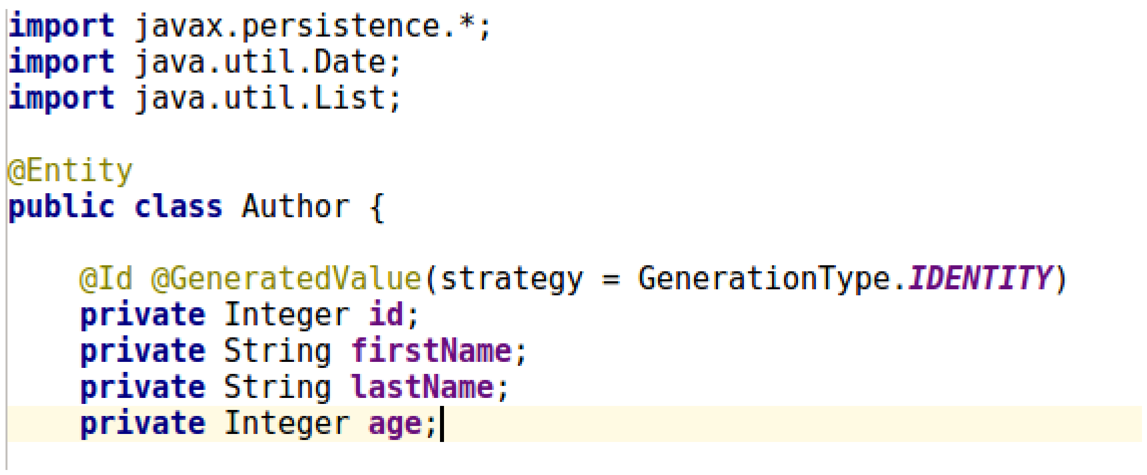
Hibernate

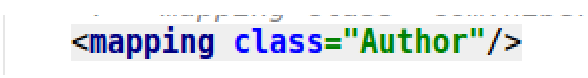
1.Create a class Author with instance variables firstName, lastName and age.

I create a class Author and make it an entity using @Entity annotation and mark its id using @Id annotation and provide the auto generation strategy as IDENTITY.



```
import javax.persistence.*;
import java.util.Date;
import java.util.List;

@Entity
public class Author {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String firstName;
    private String lastName;
    private Integer age;
```

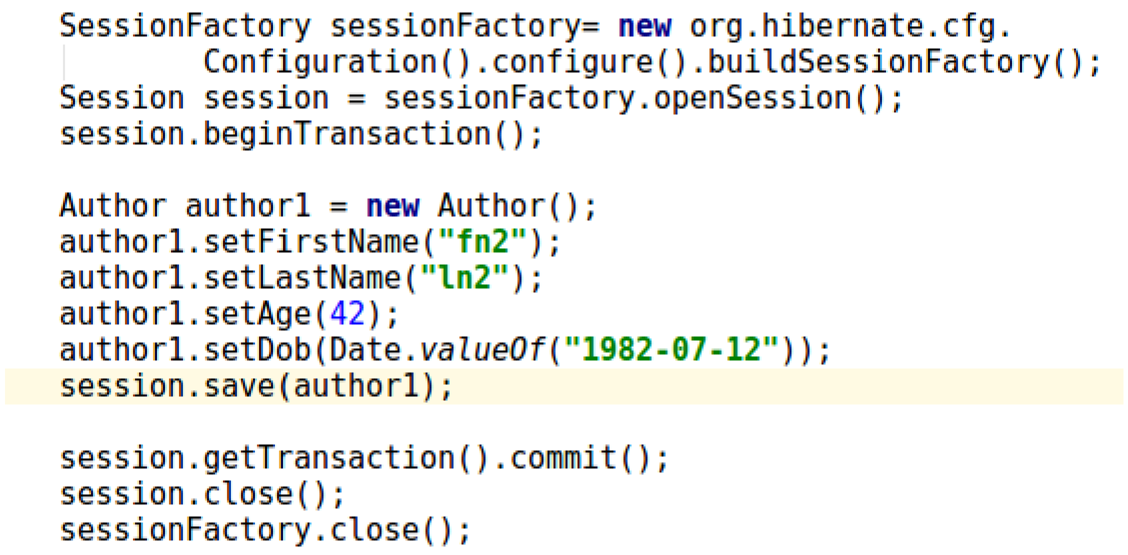
Added the Author classes’ mapping in hibernate.cfg.xml.



```
<mapping class="Author"/>
```

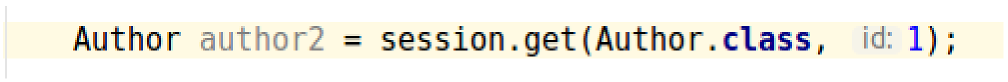
2.Perform CRUD operation for Author class.

I wrote the following code in Application class for **creating** the record. I also wrote the getters and setters for the fields.



```
SessionFactory sessionFactory= new org.hibernate.cfg.
        Configuration().configure().buildSessionFactory();
Session session = sessionFactory.openSession();
session.beginTransaction();

Author author1 = new Author();
author1.setFirstName("fn2");
author1.setLastName("ln2");
author1.setAge(42);
author1.setDob(Date.valueOf("1982-07-12"));
session.save(author1);

session.getTransaction().commit();
session.close();
sessionFactory.close();
```

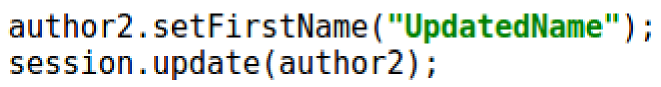
For **reading**,



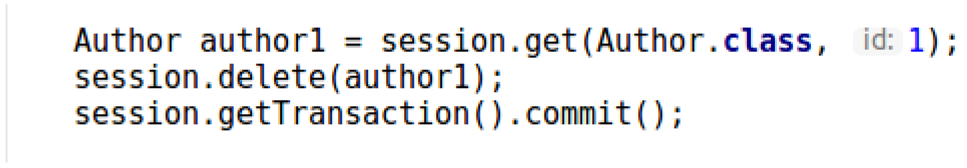
```
    Author author2 = session.get(Author.class,  id: 1);
```

For **updating**,

For **deletion**,



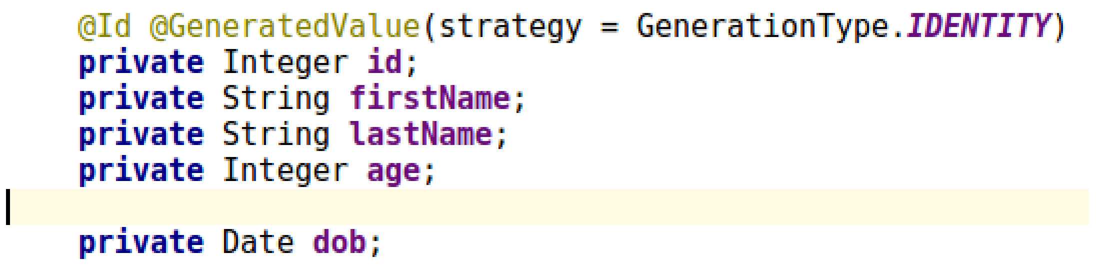
```
author2.setFirstName("UpdatedName");
session.update(author2);
```



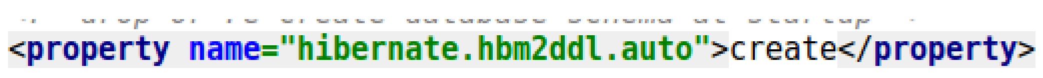
```
Author author1 = session.get(Author.class,  id: 1);
session.delete(author1);
session.getTransaction().commit();
```

3.Use hbm2ddl create to introduce Date of Birth for Author.

I added the dateOfBirth field in Author class and changed hbm2ddl to create.



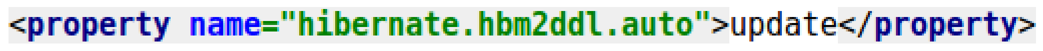
```
@Id @GeneratedValue(strategy = GenerationType.IDENTITY)
private Integer id;
private String firstName;
private String lastName;
private Integer age;

private Date dob;
```



```
<property name="hibernate.hbm2ddl.auto">create</property>
```

4.Use hbm2dll update to insert at least 4 records for Author.

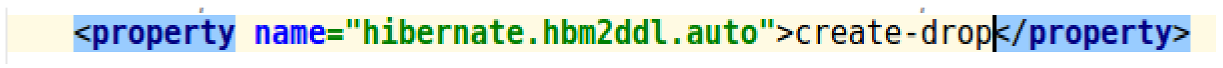
I changed the hbm2ddl to update which results in the current schema being updated and the records present in the database are not dropped.



```
<property name="hibernate.hbm2ddl.auto">update</property>
```

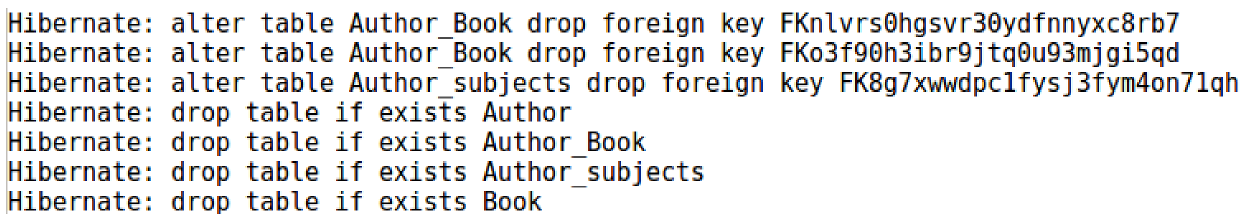
And then added four records of author similarly as I did in question 2.

5.Perform hbm2dll create-drop by closing session factory.



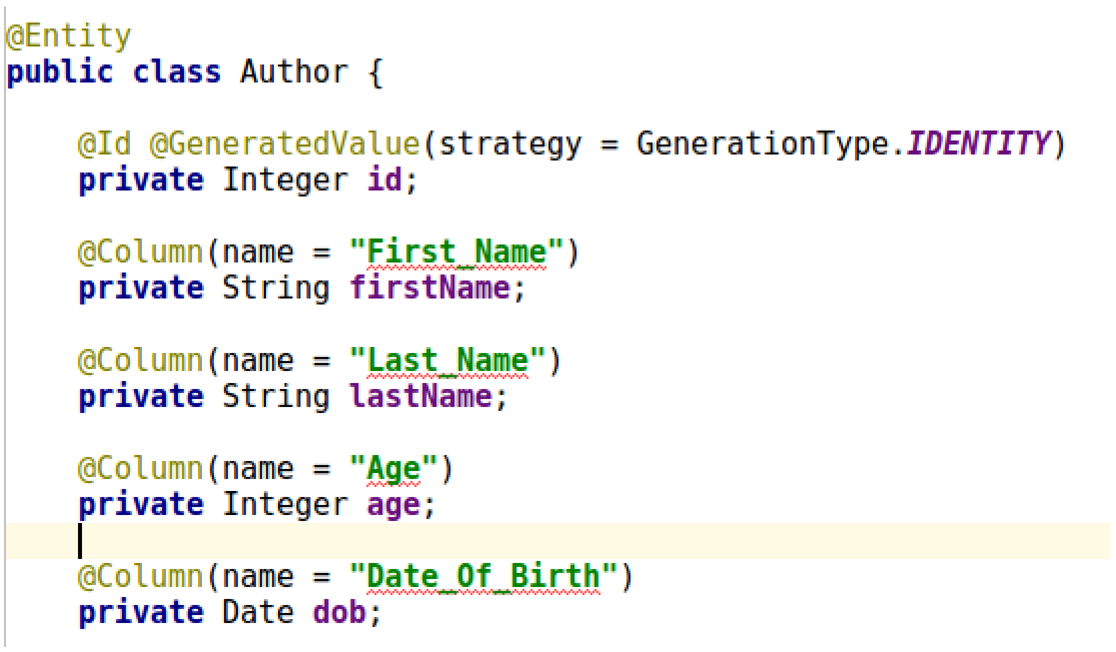
```
<property name="hibernate.hbm2ddl.auto">create-drop</property>
```

The schema is dropped after the sessionfactory is closed.



```
Hibernate: alter table Author_Book drop foreign key FKnlvrs0hgsvr30ydfnnyxc8rb7
Hibernate: alter table Author_Book drop foreign key FKo3f90h3ibr9jtq0u93mjgi5qd
Hibernate: alter table Author_subjects drop foreign key FK8g7xwwdpc1fysj3fym4on71qh
Hibernate: drop table if exists Author
Hibernate: drop table if exists Author_Book
Hibernate: drop table if exists Author_subjects
Hibernate: drop table if exists Book
```

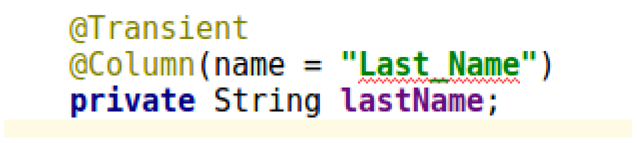
6. Rename all the fields using column annotation.



```
@Entity
public class Author {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "First_Name")
    private String firstName;

    @Column(name = "Last_Name")
    private String lastName;

    @Column(name = "Age")
    private Integer age;

    @Column(name = "Date_Of_Birth")
    private Date dob;
```

7.Mark lastName as @Transient.

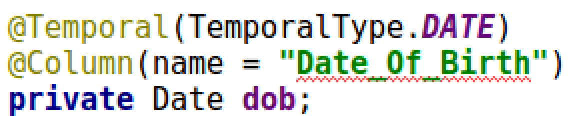
Fields marked as Transient will not be saved in the database.



```
@Transient
@Column(name = "Last_Name")
private String lastName;
```

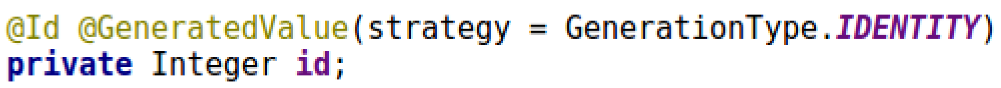
8.Use @Temporal for date of birth of Author.

Using temporal type as DATE will result in time not being saved in the database.

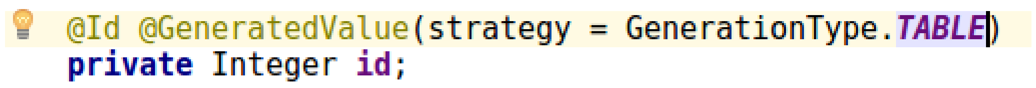


```
@Temporal(TemporalType.DATE)
@Column(name = "Date_Of_Birth")
private Date dob;
```

9.Generate Id for Author Using IDENTITY and TABLE starategy.

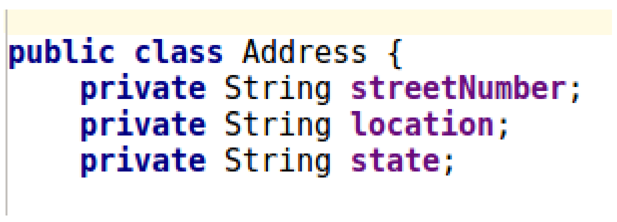


```
@Id @GeneratedValue(strategy = GenerationType.IDENTITY)
private Integer id;
```



```
@Id @GeneratedValue(strategy = GenerationType.TABLE)
private Integer id;
```

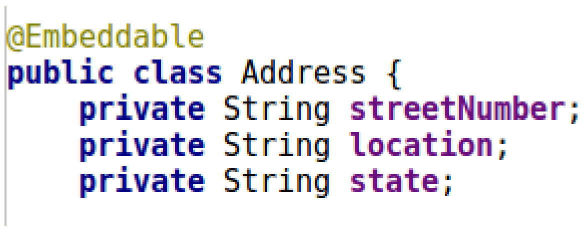
10.Create a class Address for Author with instance variables streetNumber, location, State.



```java
public class Address {
    private String streetNumber;
    private String location;
    private String state;
```

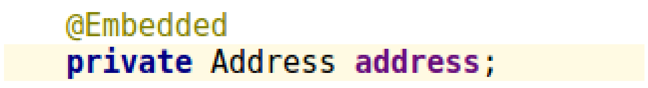
I also wrote the getters and setters for the fields.

11.Create instance variable of Address class inside Author class and save it as embedded object.



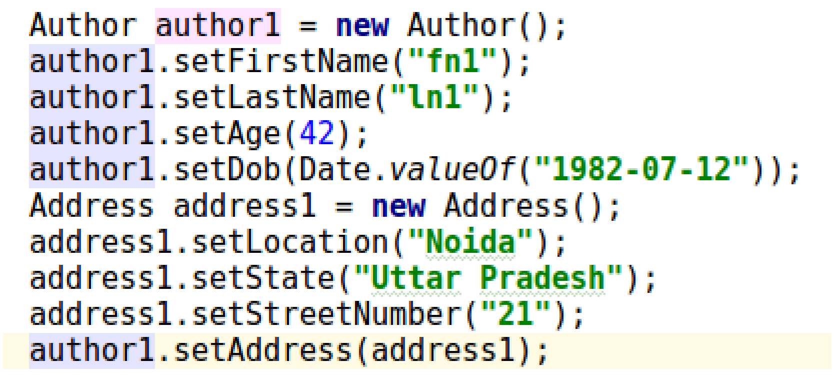
```java
@Embeddable
public class Address {
    private String streetNumber;
    private String location;
    private String state;
```

In Author class I added the field address with annotation @Embedded and its getter and setter.



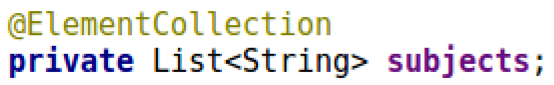
```java
    @Embedded
    private Address address;
```

The address is saved for a Author as following,



```java
Author author1 = new Author();
author1.setFirstName("fn1");
author1.setLastName("ln1");
author1.setAge(42);
author1.setDob(Date.valueOf("1982-07-12"));
Address address1 = new Address();
address1.setLocation("Noida");
address1.setState("Uttar Pradesh");
address1.setStreetNumber("21");
author1.setAddress(address1);
```

12. Introduce a List of subjects for author.

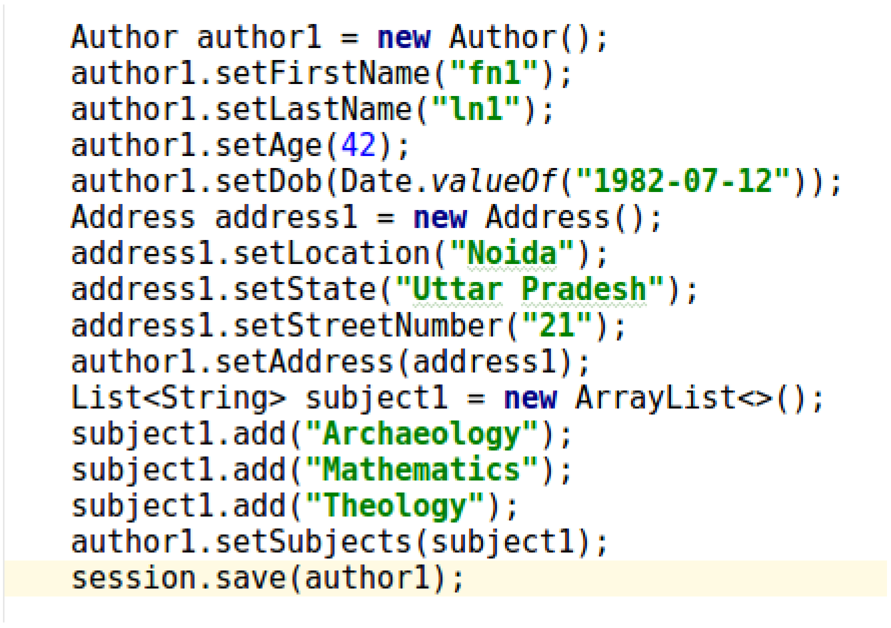


```java
@ElementCollection
private List<String> subjects;
```

This results in the creation of another table in the database which stores the collection elements alongwith its parent id.

13.Persist 3 subjects for each author.

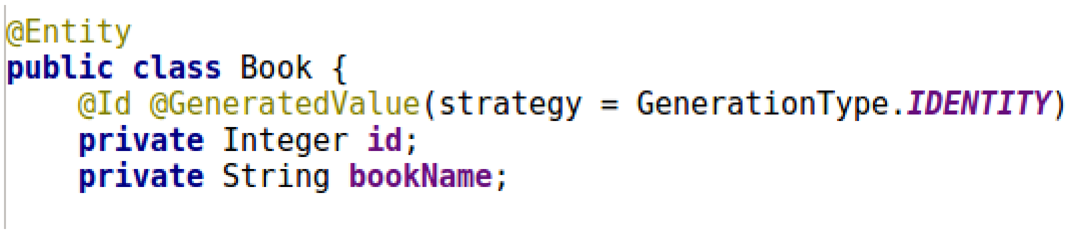
For one author



```
Author author1 = new Author();
author1.setFirstName("fn1");
author1.setLastName("ln1");
author1.setAge(42);
author1.setDob(Date.valueOf("1982-07-12"));
Address address1 = new Address();
address1.setLocation("Noida");
address1.setState("Uttar Pradesh");
address1.setStreetNumber("21");
author1.setAddress(address1);
List<String> subject1 = new ArrayList<>();
subject1.add("Archaeology");
subject1.add("Mathematics");
subject1.add("Theology");
author1.setSubjects(subject1);
session.save(author1);
```

Similarly, I added twor more author records.

14.Create an Entity book with an instance variable bookName.

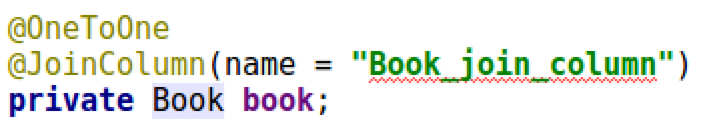


```
@Entity
public class Book {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String bookName;
```

I also wrote the getters and setters for the fields.

15.Implement One to One mapping between Author and Book.

We add the following code to Author class, a variable of Book type and also its getter and setter.

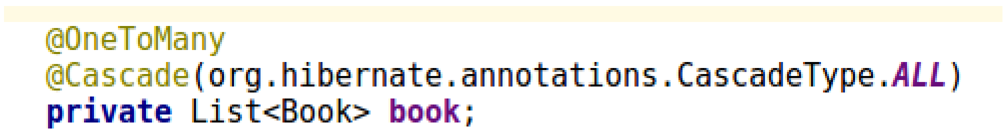


```
@OneToOne
@JoinColumn(name = "Book_join_column")
private Book book;
```

16.Implement One to Many Mapping between Author and Book(Unidirectional, BiDirectional and without additional table) and implement cascade save.

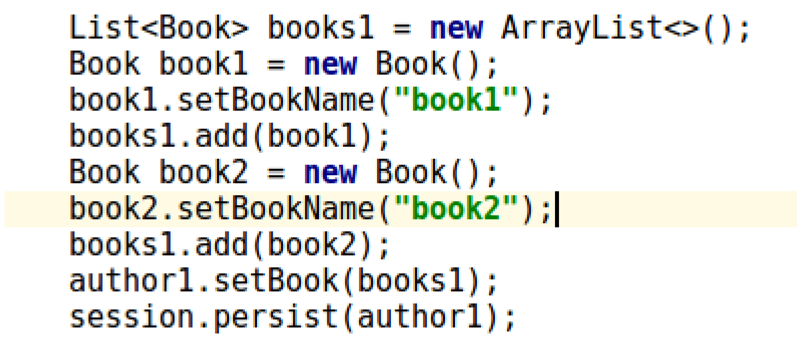
Uniderctional One to Many mapping:

In Author class we add,



```
@OneToMany
@Cascade(org.hibernate.annotations.CascadeType.ALL)
private List<Book> book;
```

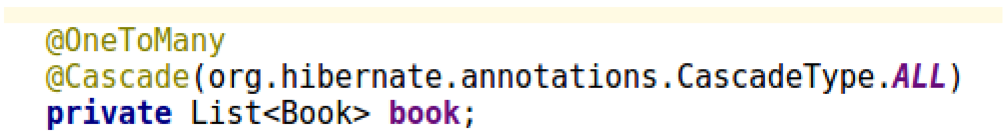
Following is the code to set fields of author and save the record in database.



```
List<Book> books1 = new ArrayList<>();
Book book1 = new Book();
book1.setBookName("book1");
books1.add(book1);
Book book2 = new Book();
book2.setBookName("book2");
books1.add(book2);
author1.setBook(books1);
session.persist(author1);
```

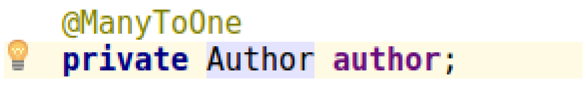
Biderctional One to Many mapping:

Added code in Author class:



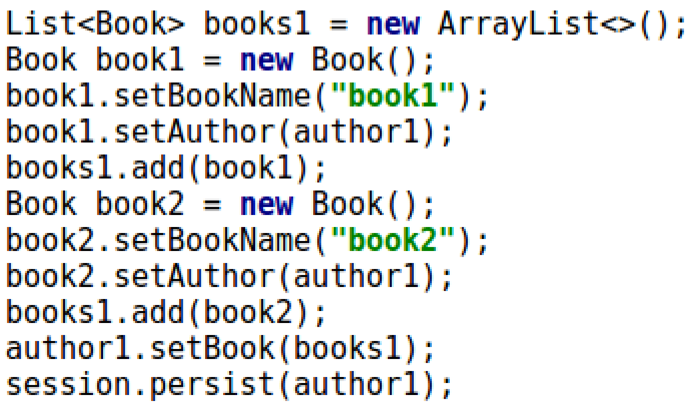
```
@OneToMany
@Cascade(org.hibernate.annotations.CascadeType.ALL)
private List<Book> book;
```

Added code in Book class:



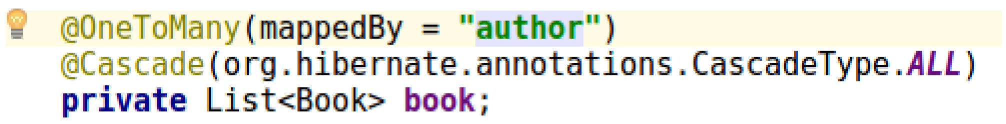
```
@ManyToOne
private Author author;
```

Also set the book field when saving the author and book records in database:



```
List<Book> books1 = new ArrayList<>();
Book book1 = new Book();
book1.setBookName("book1");
book1.setAuthor(author1);
books1.add(book1);
Book book2 = new Book();
book2.setBookName("book2");
book2.setAuthor(author1);
books1.add(book2);
author1.setBook(books1);
session.persist(author1);
```

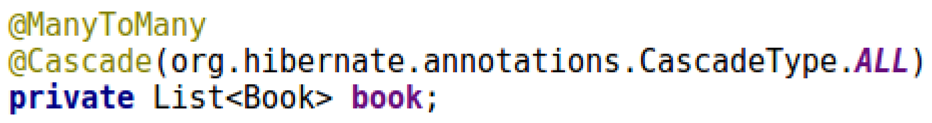
One to Many without using additional table:



```
@OneToMany(mappedBy = "author")
@Cascade(org.hibernate.annotations.CascadeType.ALL)
private List<Book> book;
```

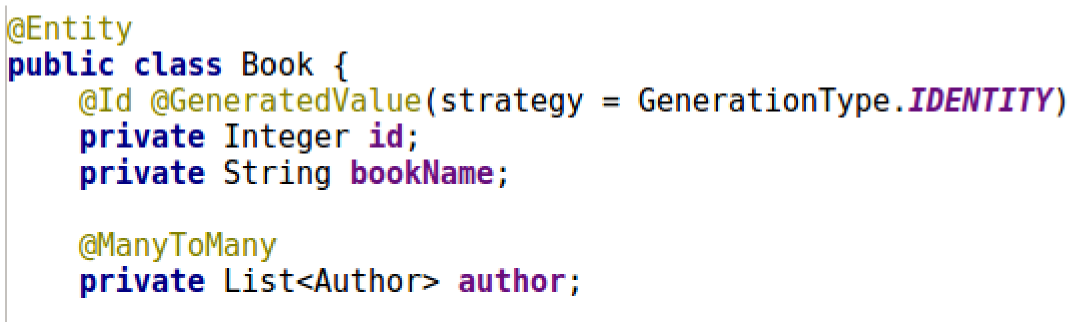
17.Implement Many to Many Mapping between Author and Book.

Added following code in Author class:



```java
@ManyToMany
@Cascade(org.hibernate.annotations.CascadeType.ALL)
private List<Book> book;
```

and the following in Book class:



```java
@Entity
public class Book {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String bookName;

    @ManyToMany
    private List<Author> author;
```