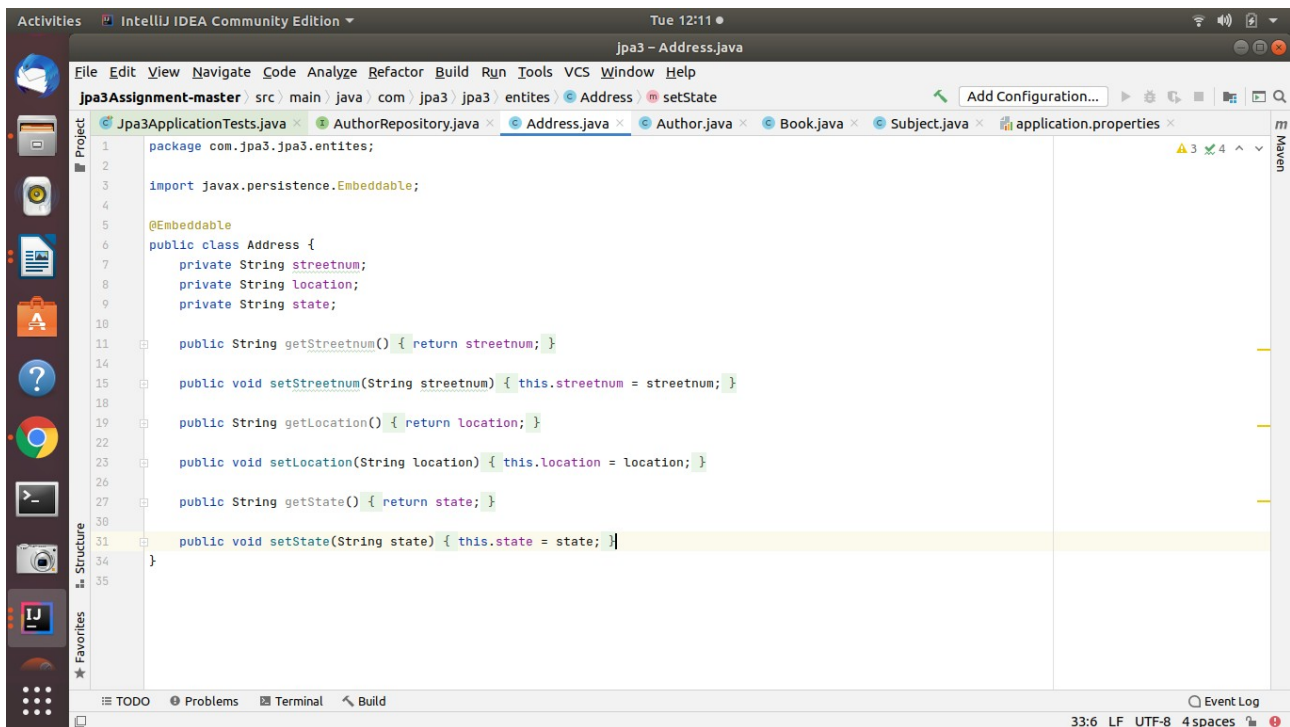


Spring Data JPA with Hibernate Part 3 Exercise

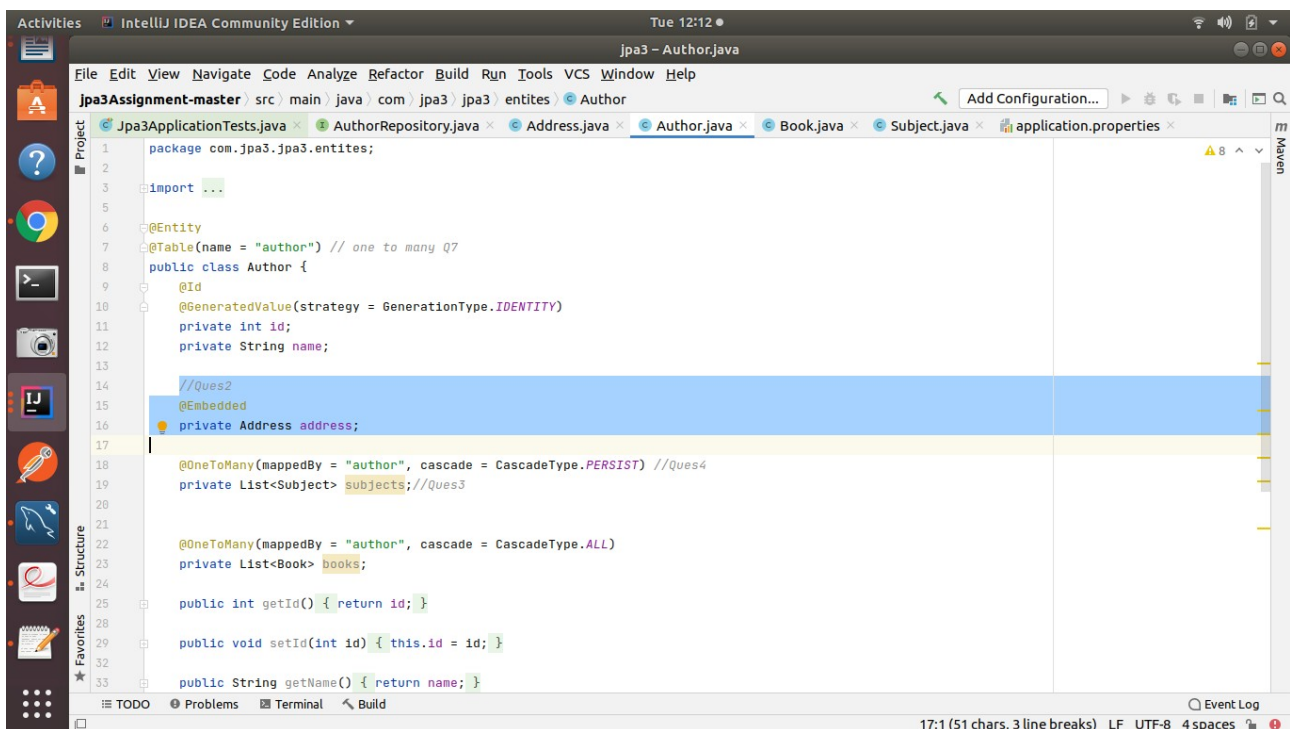
1. Create a class Address for Author with instance variables streetNumber, location, State.



The screenshot shows the IntelliJ IDEA Community Edition interface. The main editor window displays the `Address.java` file. The code defines an `Address` class that implements `Embeddable`. It has three private instance variables: `streetnum`, `location`, and `state`. Each variable has a corresponding getter and setter method. The class is annotated with `@Embeddable`. The package is `com.jp3.jp3.entites`.

```
1 package com.jp3.jp3.entites;
2
3 import javax.persistence.Embeddable;
4
5 @Embeddable
6 public class Address {
7     private String streetnum;
8     private String location;
9     private String state;
10
11     public String getStreetnum() { return streetnum; }
12
13     public void setStreetnum(String streetnum) { this.streetnum = streetnum; }
14
15     public String getLocation() { return location; }
16
17     public void setLocation(String location) { this.location = location; }
18
19     public String getState() { return state; }
20
21     public void setState(String state) { this.state = state; }
22 }
```

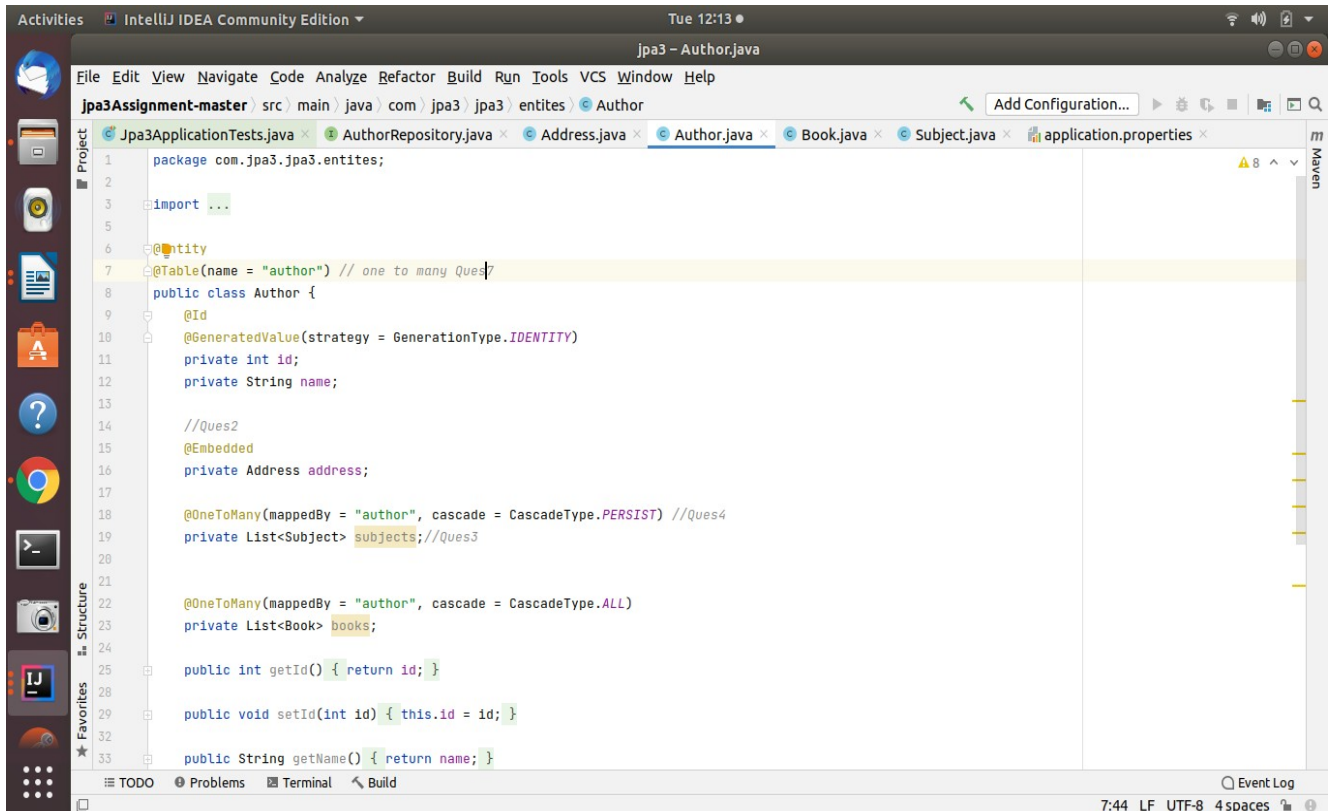
2. Create instance variable of Address class inside Author class and save it as embedded object.



The screenshot shows the IntelliJ IDEA Community Edition interface. The main editor window displays the `Author.java` file. The code defines an `Author` class that implements `Entity`. It has three private instance variables: `id`, `name`, and `address`. The `id` variable is annotated with `@Id` and `@GeneratedValue`. The `name` variable is annotated with `@Table`. The `address` variable is annotated with `@Embedded`. The class is annotated with `@Entity`. The package is `com.jp3.jp3.entites`.

```
1 package com.jp3.jp3.entites;
2
3 import ...
4
5 @Entity
6 @Table(name = "author") // one to many Q7
7 public class Author {
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10     private int id;
11     private String name;
12
13     //Ques2
14     @Embedded
15     private Address address;
16
17     @OneToMany(mappedBy = "author", cascade = CascadeType.PERSIST) //Ques4
18     private List<Subject> subjects; //Ques3
19
20     @OneToMany(mappedBy = "author", cascade = CascadeType.ALL)
21     private List<Book> books;
22
23     public int getId() { return id; }
24
25     public void setId(int id) { this.id = id; }
26
27     public String getName() { return name; }
28 }
```

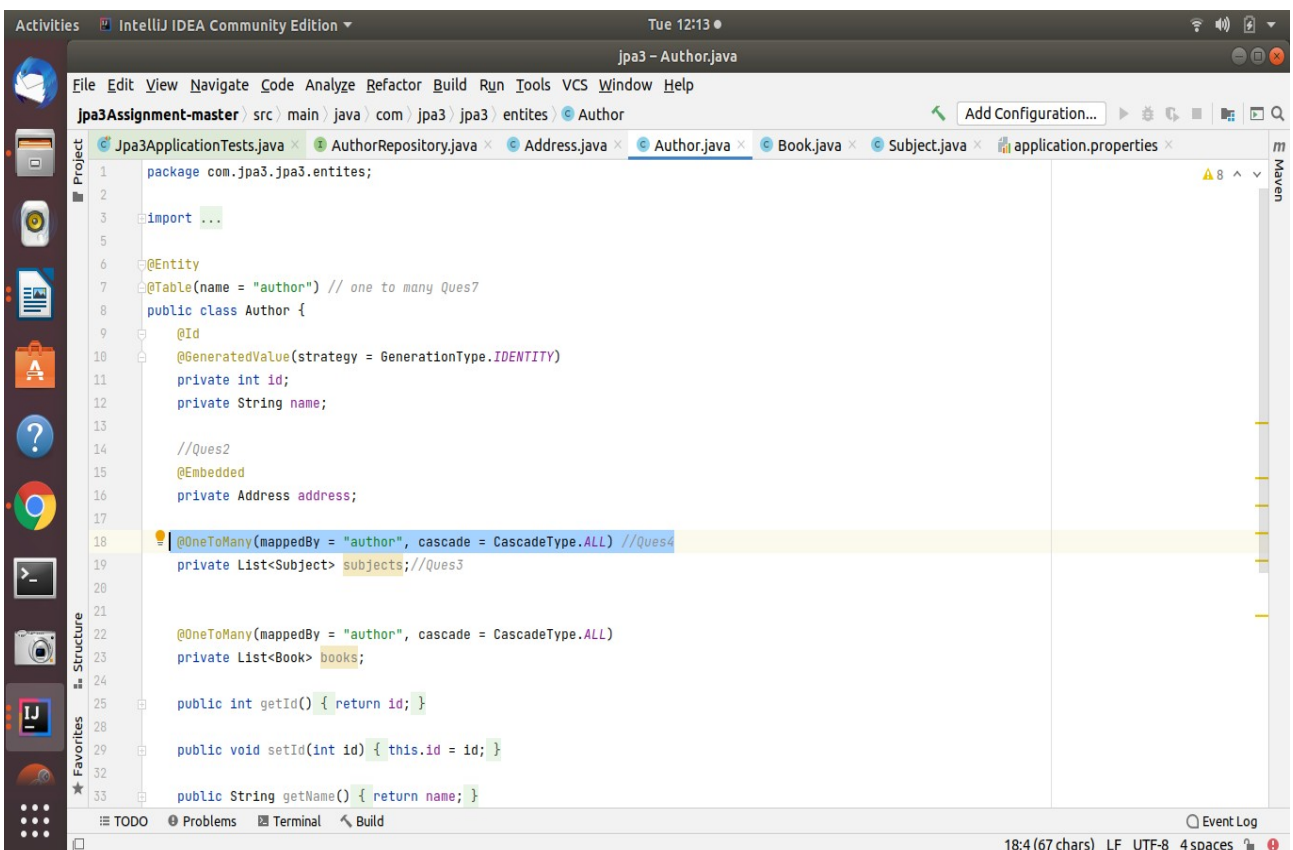
3. Introduce a List of subjects for author.



The screenshot shows the IntelliJ IDEA Community Edition interface. The main editor displays the `Author.java` file. The code defines an `Author` entity with a `name` attribute and a `subjects` list. The `subjects` list is annotated with `@OneToMany(mappedBy = "author", cascade = CascadeType.PERSIST)`, indicating a one-to-many relationship where the `Author` is the primary side and `Subject` is the secondary side. The `subjects` list is also annotated with `//Ques3`. The `Author` entity is also annotated with `@Entity` and `@Table(name = "author")`. The `Author` entity has a `getId()` method and a `setName()` method.

```
1 package com.jpa3.jpa3.entities;
2
3 import ...
4
5
6 @Entity
7 @Table(name = "author") // one to many Ques7
8 public class Author {
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    private int id;
12    private String name;
13
14    //Ques2
15    @Embedded
16    private Address address;
17
18    @OneToMany(mappedBy = "author", cascade = CascadeType.PERSIST) //Ques4
19    private List<Subject> subjects; //Ques3
20
21
22    @OneToMany(mappedBy = "author", cascade = CascadeType.ALL)
23    private List<Book> books;
24
25    public int getId() { return id; }
26
27
28    public void setId(int id) { this.id = id; }
29
30
31    public String getName() { return name; }
```

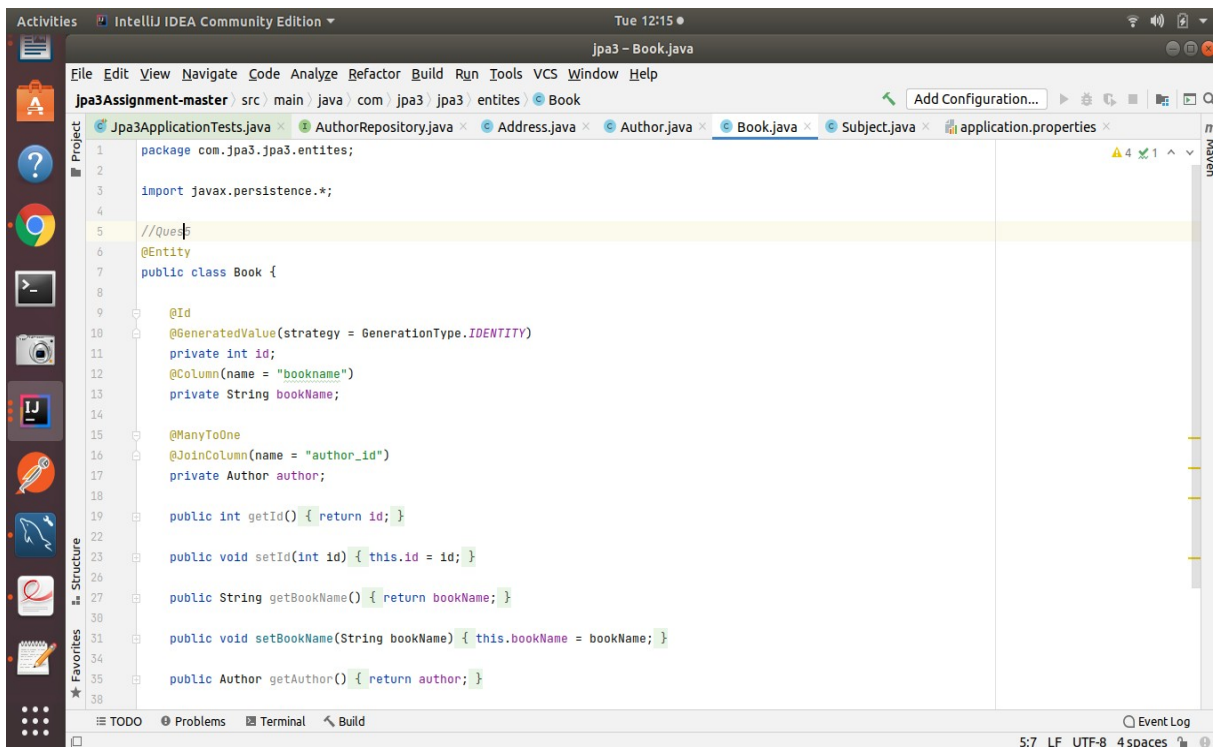
4. Persist 3 subjects for each author.



The screenshot shows the IntelliJ IDEA Community Edition interface. The main editor displays the `Author.java` file. The code defines an `Author` entity with a `name` attribute and a `subjects` list. The `subjects` list is annotated with `@OneToMany(mappedBy = "author", cascade = CascadeType.ALL)`, indicating a one-to-many relationship where the `Author` is the primary side and `Subject` is the secondary side. The `subjects` list is also annotated with `//Ques3`. The `Author` entity is also annotated with `@Entity` and `@Table(name = "author")`. The `Author` entity has a `getId()` method and a `setName()` method.

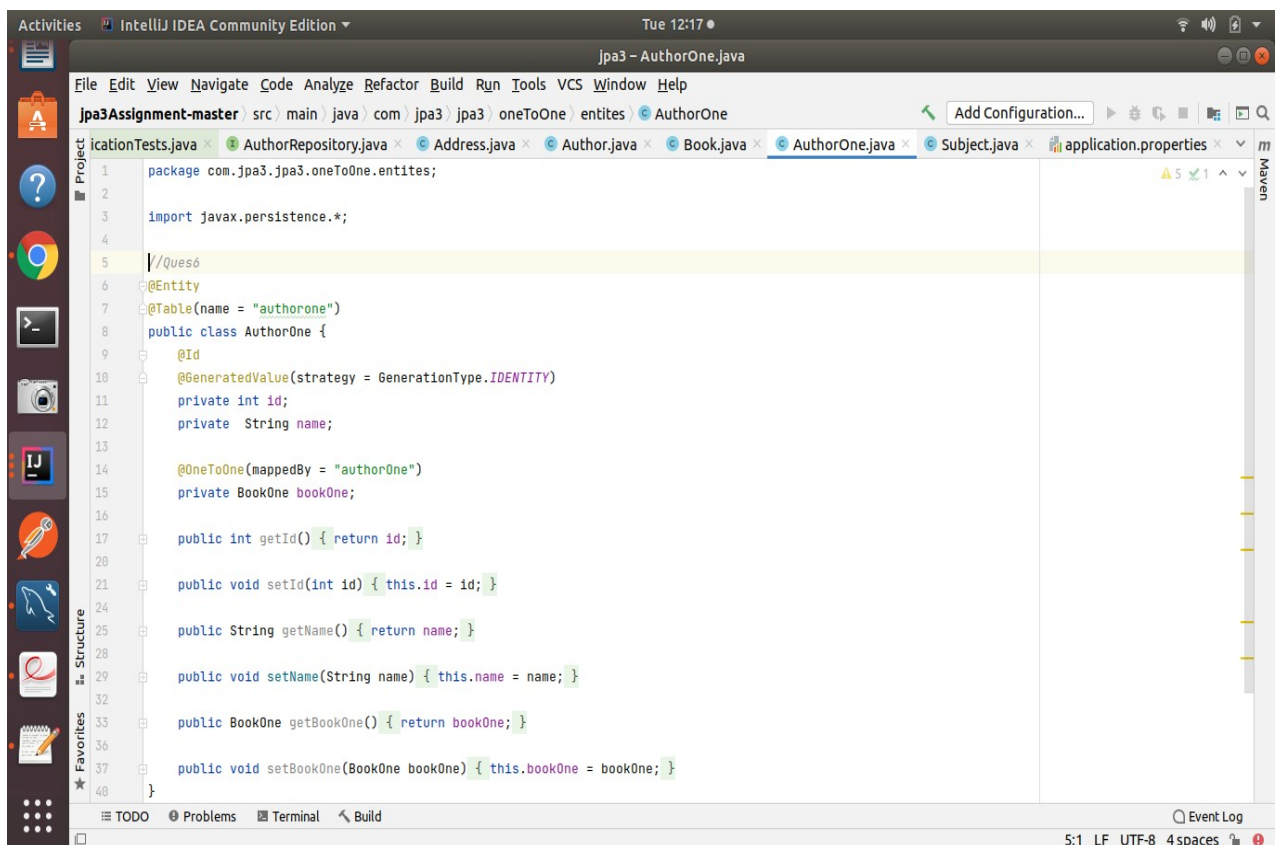
```
1 package com.jpa3.jpa3.entities;
2
3 import ...
4
5
6 @Entity
7 @Table(name = "author") // one to many Ques7
8 public class Author {
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    private int id;
12    private String name;
13
14    //Ques2
15    @Embedded
16    private Address address;
17
18    @OneToMany(mappedBy = "author", cascade = CascadeType.ALL) //Ques4
19    private List<Subject> subjects; //Ques3
20
21
22    @OneToMany(mappedBy = "author", cascade = CascadeType.ALL)
23    private List<Book> books;
24
25    public int getId() { return id; }
26
27
28    public void setId(int id) { this.id = id; }
29
30
31    public String getName() { return name; }
```

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

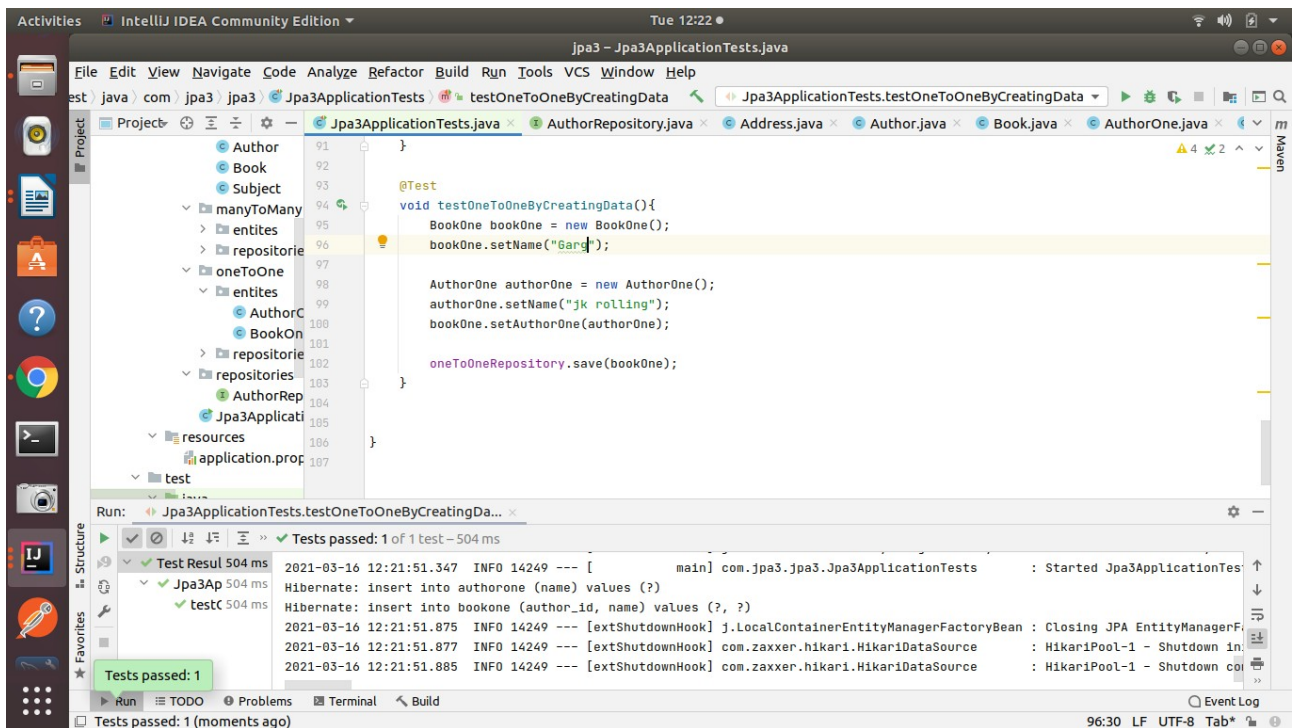
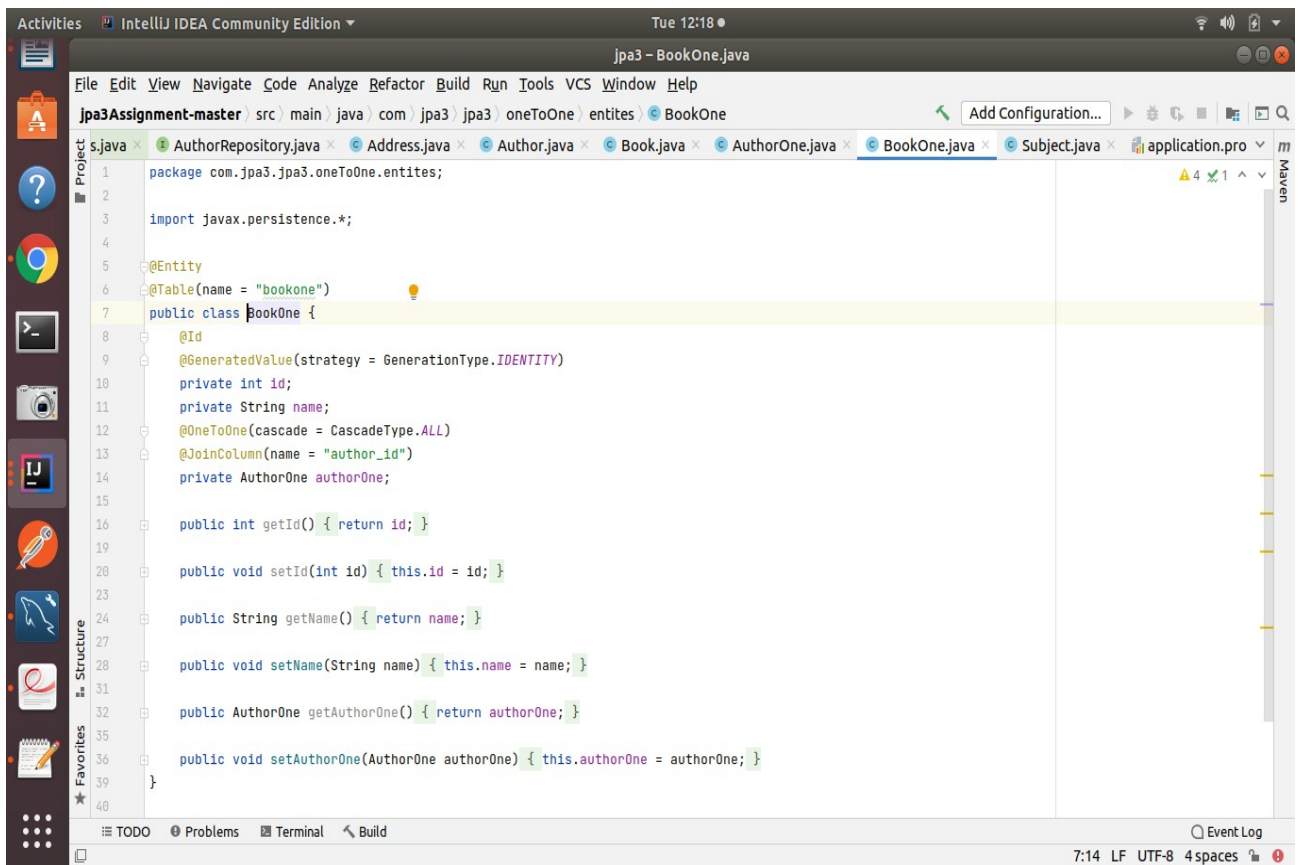


```
1 package com.jpaa3.jpaa3.entities;
2
3 import javax.persistence.*;
4
5 //ques5
6 @Entity
7 public class Book {
8
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    private int id;
12    @Column(name = "bookname")
13    private String bookName;
14
15    @ManyToOne
16    @JoinColumn(name = "author_id")
17    private Author author;
18
19    public int getId() { return id; }
20
21    public void setId(int id) { this.id = id; }
22
23    public String getBookName() { return bookName; }
24
25    public void setBookName(String bookName) { this.bookName = bookName; }
26
27    public Author getAuthor() { return author; }
```

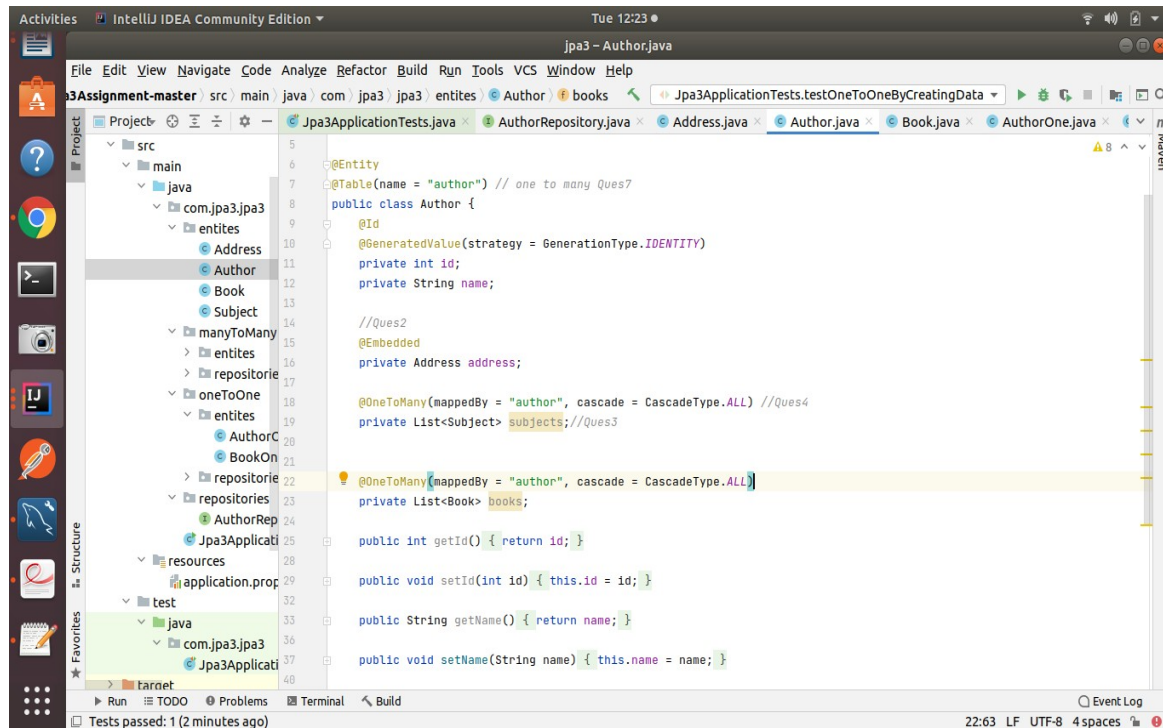
6. Implement One to One mapping between Author and Book.



```
1 package com.jpaa3.jpaa3.oneToOne.entities;
2
3 import javax.persistence.*;
4
5 //ques6
6 @Entity
7 @Table(name = "authorone")
8 public class AuthorOne {
9
10    @Id
11    @GeneratedValue(strategy = GenerationType.IDENTITY)
12    private int id;
13    private String name;
14
15    @OneToOne(mappedBy = "authorOne")
16    private BookOne bookOne;
17
18    public int getId() { return id; }
19
20    public void setId(int id) { this.id = id; }
21
22    public String getName() { return name; }
23
24    public void setName(String name) { this.name = name; }
25
26    public BookOne getBookOne() { return bookOne; }
27
28    public void setBookOne(BookOne bookOne) { this.bookOne = bookOne; }
29 }
```

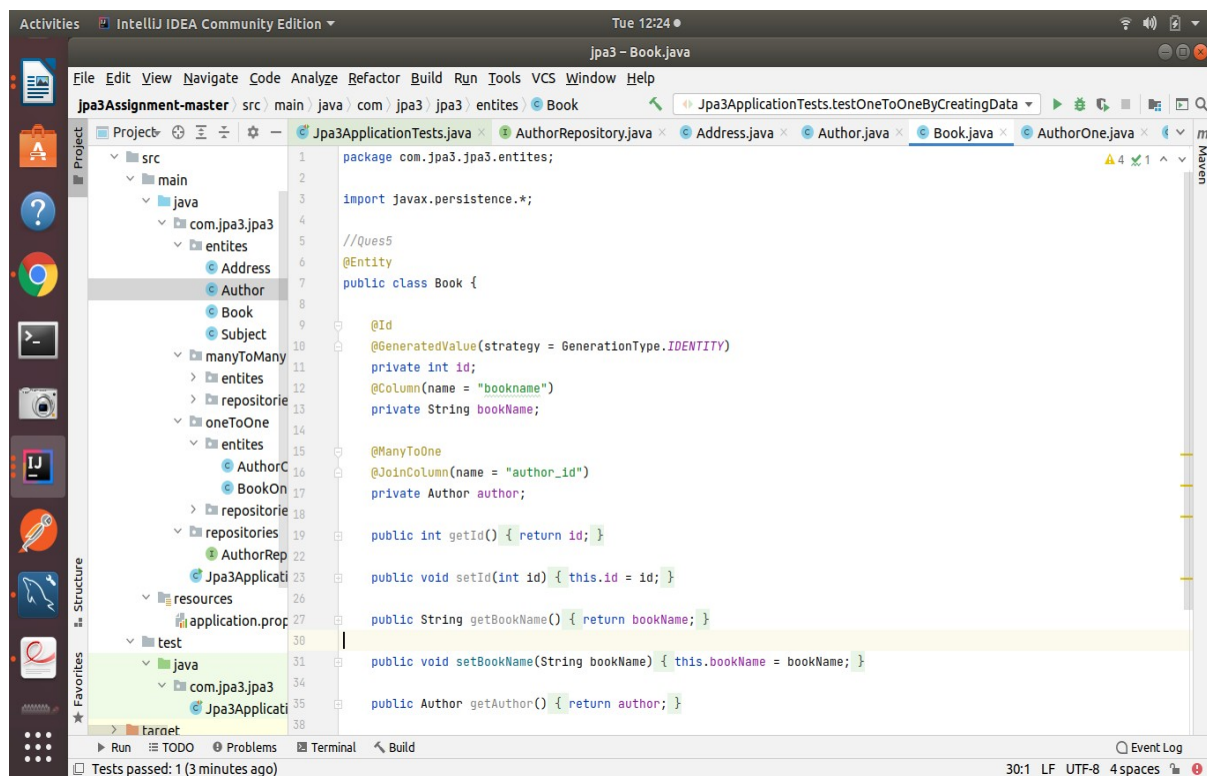



7. Implement One to Many Mapping between Author and Book(Unidirectional, BiDirectional and without additional table) and implement cascade save.



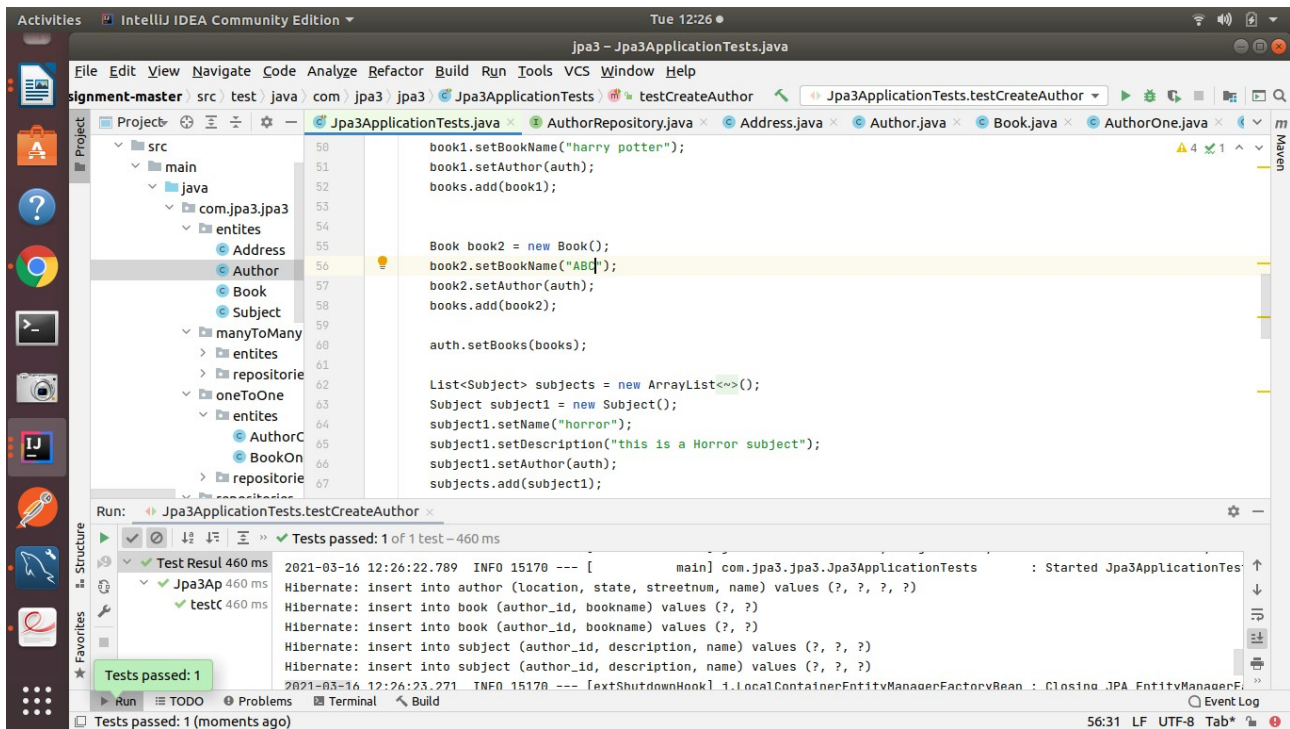
The screenshot shows the IntelliJ IDEA Community Edition interface. The main editor displays the `Author.java` file. The code defines an `Author` entity with a one-to-many relationship to the `Book` entity. The annotations used are `@Entity`, `@Table(name = "author")`, `@Id`, `@GeneratedValue(strategy = GenerationType.IDENTITY)`, `@Embedded`, `@OneToOne(mappedBy = "author", cascade = CascadeType.ALL)`, and `@OneToMany(mappedBy = "author", cascade = CascadeType.ALL)`. The `Author` class has private fields for `id` and `name`, and a `List<Subject>` field named `subjects`. The `Book` class is also visible in the project structure, showing a `bookName` field and a `Author` field.

```
5 @Entity
6 @Table(name = "author") // one to many Ques7
7 public class Author {
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10    private int id;
11    private String name;
12
13    //Ques2
14    @Embedded
15    private Address address;
16
17    @OneToOne(mappedBy = "author", cascade = CascadeType.ALL) //Ques4
18    private List<Subject> subjects; //Ques3
19
20    @OneToOne(mappedBy = "author", cascade = CascadeType.ALL)
21    private List<Book> books;
22
23    public int getId() { return id; }
24    public void setId(int id) { this.id = id; }
25
26    public String getName() { return name; }
27    public void setName(String name) { this.name = name; }
28
29    //Ques5
30    @Entity
31    public class Book {
32        @Id
33        @GeneratedValue(strategy = GenerationType.IDENTITY)
34        private int id;
35        @Column(name = "bookname")
36        private String bookName;
37
38        @ManyToOne
39        @JoinColumn(name = "author_id")
40        private Author author;
41
42        public int getId() { return id; }
43        public void setId(int id) { this.id = id; }
44
45        public String getBookName() { return bookName; }
46        public void setBookName(String bookName) { this.bookName = bookName; }
47
48        public Author getAuthor() { return author; }
49    }
50}
```

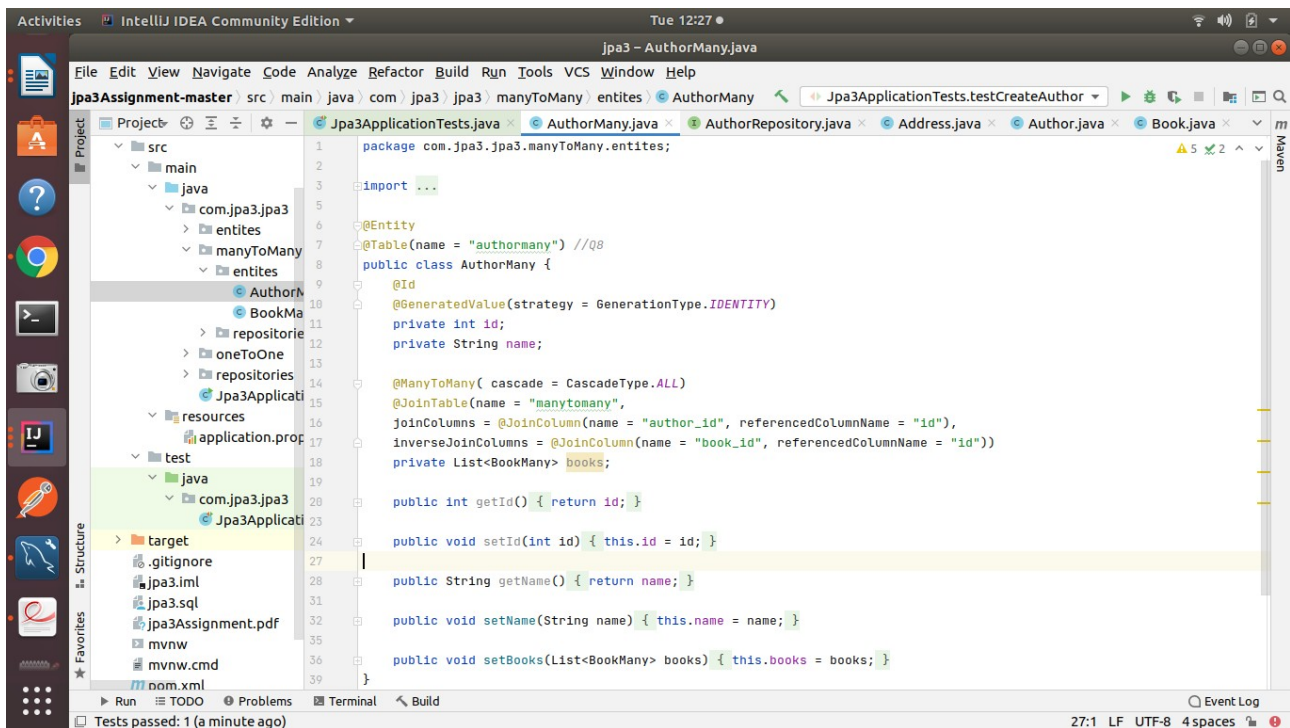


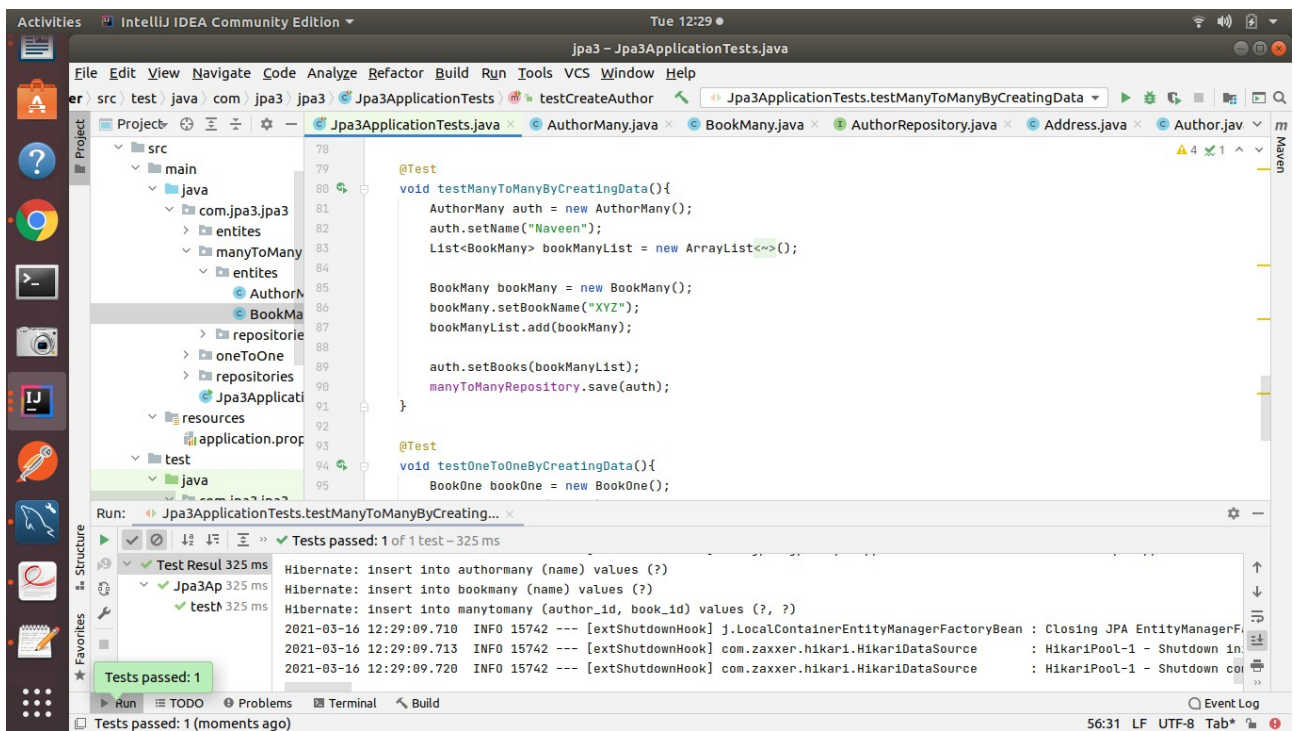
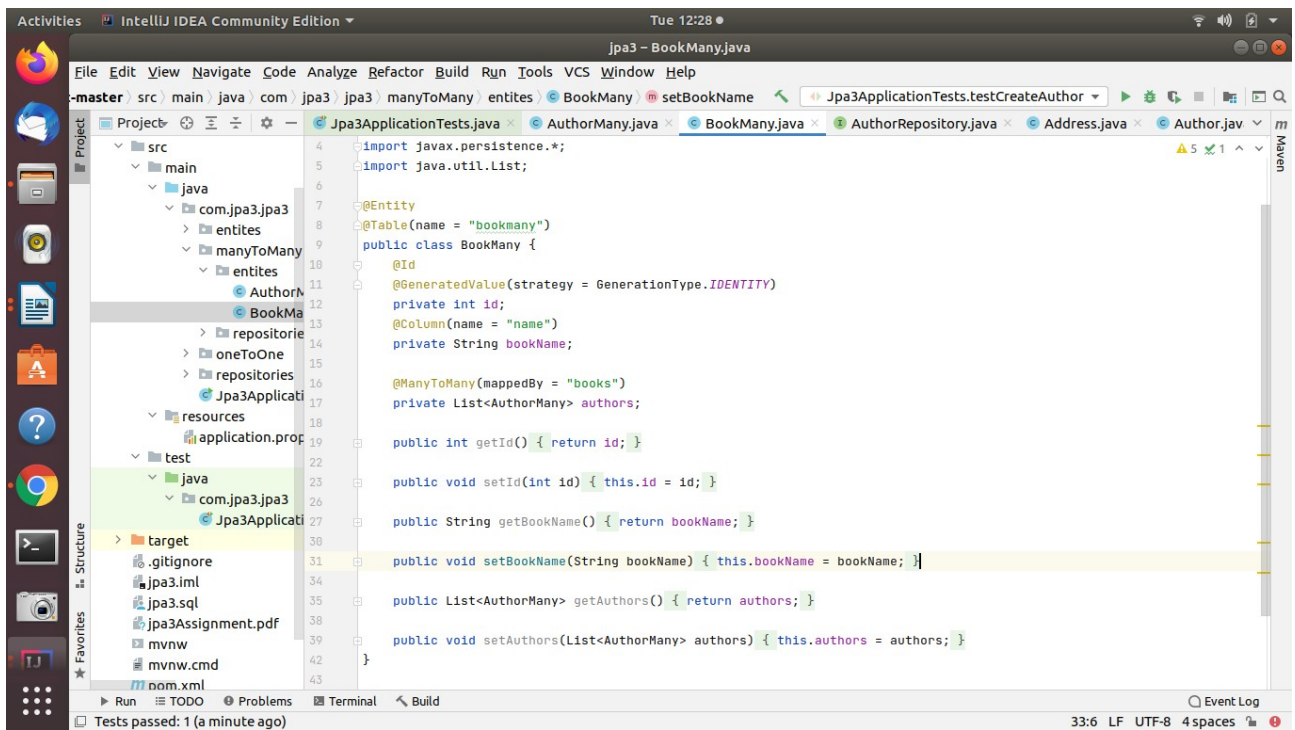
The screenshot shows the IntelliJ IDEA Community Edition interface. The main editor displays the `Book.java` file. The code defines a `Book` entity with a one-to-many relationship to the `Author` entity. The annotations used are `@Entity`, `@Table(name = "book")`, `@Id`, `@GeneratedValue(strategy = GenerationType.IDENTITY)`, `@Column(name = "bookname")`, `@ManyToOne`, and `@JoinColumn(name = "author_id")`. The `Book` class has private fields for `id` and `bookName`, and an `Author` field named `author`. The `Author` class is also visible in the project structure, showing an `id` field and a `name` field.

```
1 package com.jpa3.jpa3.entities;
2
3 import javax.persistence.*;
4
5 //Ques5
6 @Entity
7 public class Book {
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10    private int id;
11    @Column(name = "bookname")
12    private String bookName;
13
14    @ManyToOne
15    @JoinColumn(name = "author_id")
16    private Author author;
17
18    public int getId() { return id; }
19    public void setId(int id) { this.id = id; }
20
21    public String getBookName() { return bookName; }
22    public void setBookName(String bookName) { this.bookName = bookName; }
23
24    public Author getAuthor() { return author; }
25}
```



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





9. Which method on the session object can be used to remove an object from the cache?

Ans:

- 1) evict() is used to remove a particular object from cache associated with session.
- 2) You cannot disable the first level cache, it always enabled
- 3) Hibernate entities or database rows remain in cache only until Session is open, once Session is closed, all associated cached data is lost

Eg.

```
@Test
@Transactional
public void testCaching(){
    Session session = entityManager.unwrap(Session.class);
    Author author = repository.findAllById(1);
    repository.findAllById(1);
    session.evict(author);
    repository.findAllById(1);
}
```

10. What does @transactional annotation do?

Ans:

The @Transactional annotation is metadata that specifies that an interface, class, or method must have transactional semantics; for example, "start a brand new read-only transaction when this method is invoked, suspending any existing transaction".

Eg:

```
@Transactional
//whole transaction won't commit due to transactional
public void testTransfer(int amount){
    Account account1 = new Account();
    Account account2 = new Account();
    account1.setBalance(account1.getBalance() - amount);
    if(true){
        throw new RuntimeException();
    }
    account2.setBalance(account2.getBalance() + amount);
}
```