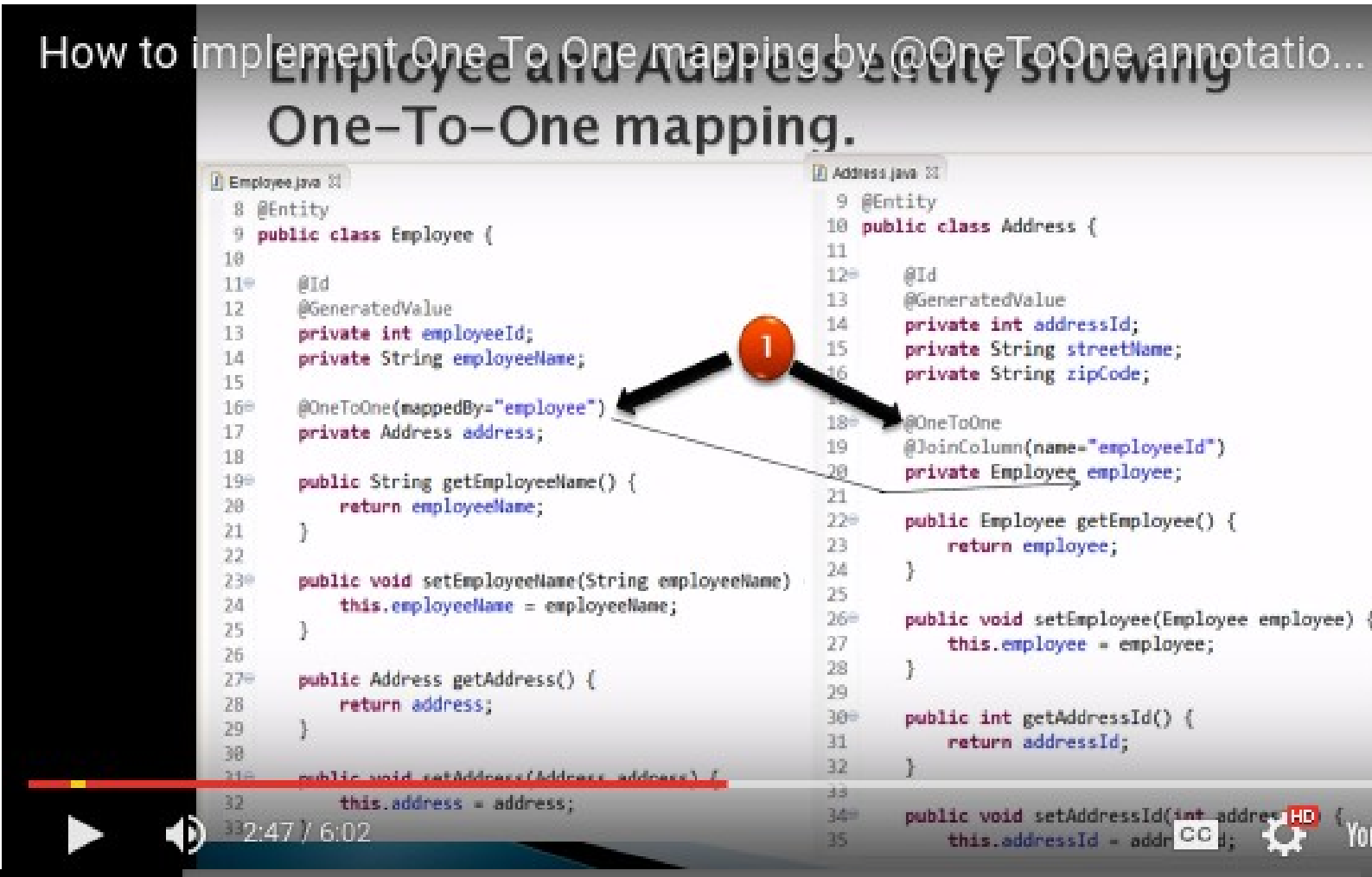
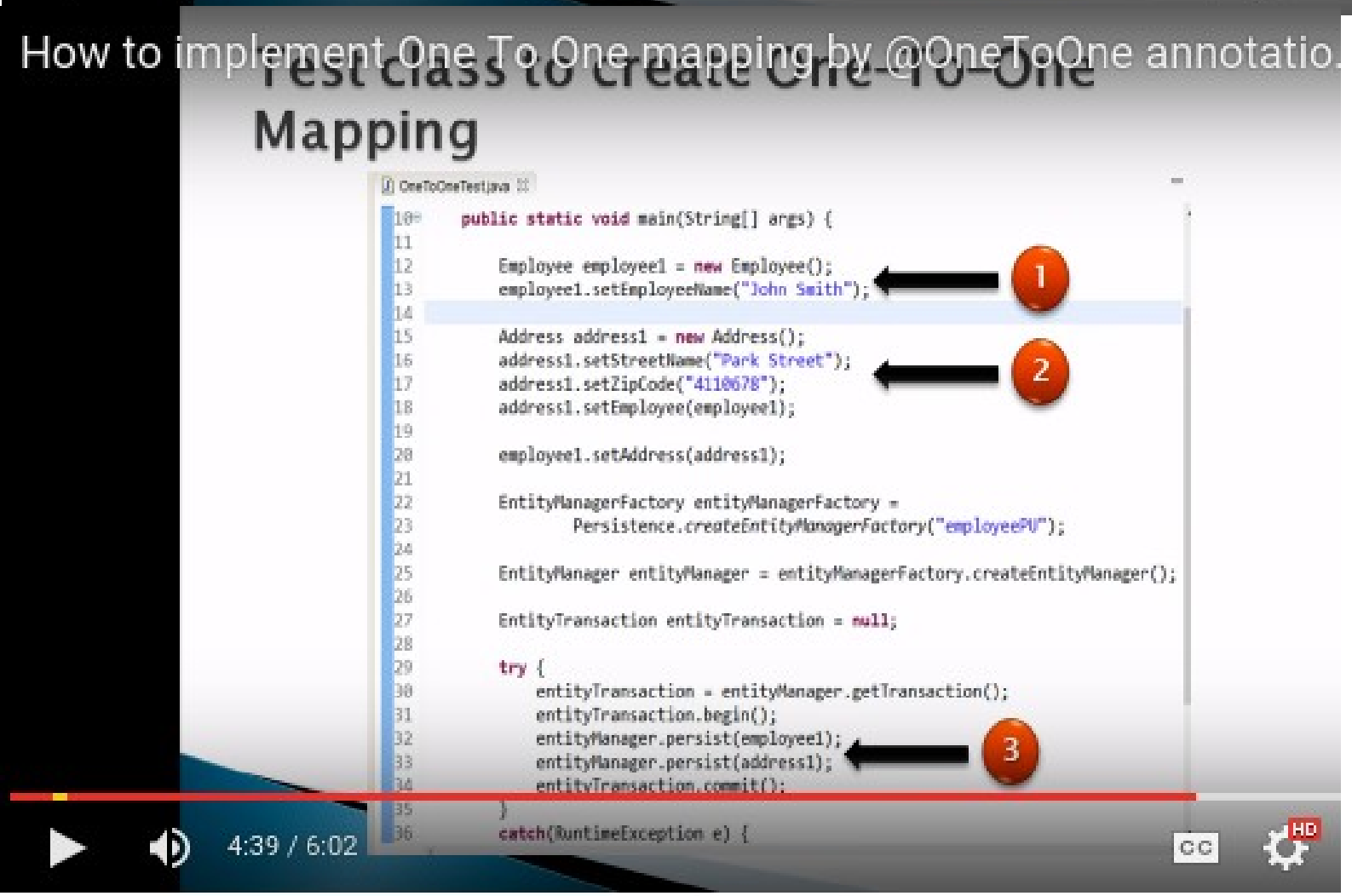

How to implement One To One mapping by @OneToOne annotatio...
Employee and Address entity showing One-To-One mapping.
Employee.java
@Entity
public class Employee {
@Id
@GeneratedValue
private int employeeId;
private String employeeName;
@OneToOne(mappedBy="employee")
private Address address;
public String getEmployeeName() {
return employeeName;
}
public void setEmployeeName(String employeeName)
this.employeeName = employeeName;
}
public Address getAddress() {
return address;
}
this.address = address;
Address.java
@Entity
public class Address {
@Id
@GeneratedValue
private int addressId;
private String streetName;
private String zipCode;
@OneToOne
@JoinColumn(name="employeeId")
private Employee employee;
public Employee getEmployee() {
return employee;
}
public void setEmployee(Employee employee)
this.employee = employee;
}
public int getAddressId() {
return addressId;
}
this.addressId =
1
2:47 / 6:02
HD
CC

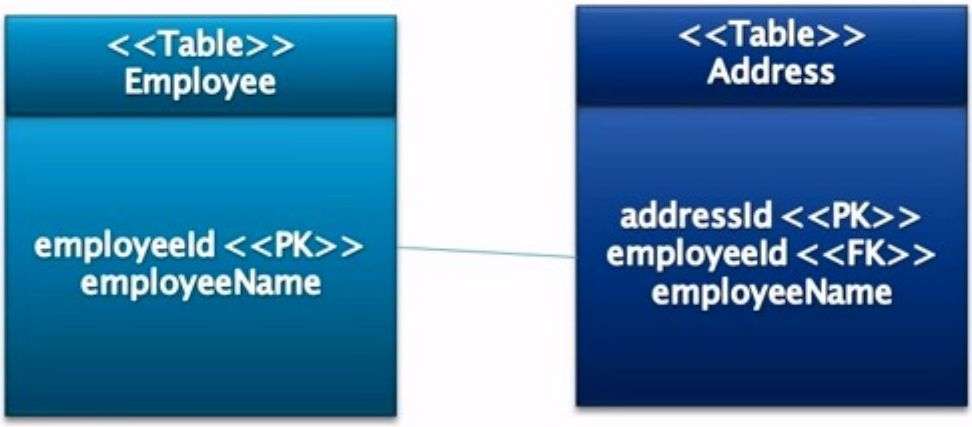

How to implement One To One mapping by @OneToOne annotatio...
Test class to create One-To-One Mapping
OneToOneTest.java
public static void main(String[] args) {
Employee employee1 = new Employee();
employee1.setEmployeeName("John Smith");
Address address1 = new Address();
address1.setStreetName("Park Street");
address1.setZipCode("411067B");
address1.setEmployee(employee1);
employee1.setAddress(address1);
EntityManagerFactory entityManagerFactory =
Persistence.createEntityManagerFactory("employeePU");
EntityManager entityManager = entityManagerFactory.createEntityManager();
EntityTransaction entityTransaction = null;
try {
entityTransaction = entityManager.getTransaction();
entityTransaction.begin();
entityManager.persist(employee1);
entityManager.persist(address1);
entityTransaction.commit();
}
catch(RuntimeException e) {
1
2
3
4:39 / 6:02
CC
HD

Persistence.xml file in Hibernate

persistence.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <persistence xmlns="http://java.sun.com/xml/ns/persistence"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persiste
5      version="1.0">
6
7      <persistence-unit name="employeePU" transaction-type="RESOURCE_LOCAL">
8          <provider>org.hibernate.ejb.HibernatePersistence</provider>
9
10         <properties>
11             <property name="hibernate.connection.driver_class" value="org.hsqldb.jdbcDriver" />
12             <property name="hibernate.connection.url" value="jdbc:hsqldb:hsql://localhost:9001" />
13             <property name="hibernate.connection.username" value="sa" />
14             <property name="hibernate.connection.password" value="" />
15             <property name="hibernate.dialect" value="org.hibernate.dialect.HSQLDialect" />
16             <property name="hibernate.hbm2ddl.auto" value="create"/>
17         </properties>
18     </persistence-unit>
19 </persistence>
20
```

One-To-One Mapping



How to use Enum types in Entity by @Enumerated annotation in Hi...

@Enumerated annotation in JPA

- Enumerations were introduced with Java 5.
- Enumerations are constant values, that have a implicit ordinal value associated with it. These ordinal values are assigned to each constant based on the order in which constants are declared.
- Enumerations can also be stored into database based on either ordinal value or constant value.
- Enumerations can be persisted into database by using @Enumerated annotation in entities.

How to use Enum types in Entity by @Enumerated annotation in Hi...

Enumerations and ordinal values

Role.java

```
1 public enum Role {
2
3     DEVELOPER,
4     TESTER,
5     MANAGER,
6     HR
7
8 }
```

- Role is a enum type. It has 4 enum constants with implicit ordinal value associated with it.
- 0 for Role.DEVELOPER
- 1 for Role.TESTER
- 2 for Role.MANAGER
- 3 for Role.HR

How to use Enum types in Entity by @Enumerated annotation in Hi...

@Enumerated(EnumType.ORDINAL)

Employee.java

```
 9 @Entity
10 public class Employee {
11
12     @Id
13     @GeneratedValue
14     private int employeeId;
15     private String employeeName;
16
17     @Enumerated(EnumType.ORDINAL)
18     private Role employeeRole;
19
20     public int getEmployeeId() {
21         return employeeId;
22     }
23
24     public void setEmployeeId(int employeeId) {
25         this.employeeId = employeeId;
26     }
27
28     public Role getEmployeeRole() {
29         return employeeRole;
30     }
31
32     public void setEmployeeRole(Role employeeRole) {
33         this.employeeRole = employeeRole;
34     }
```

1

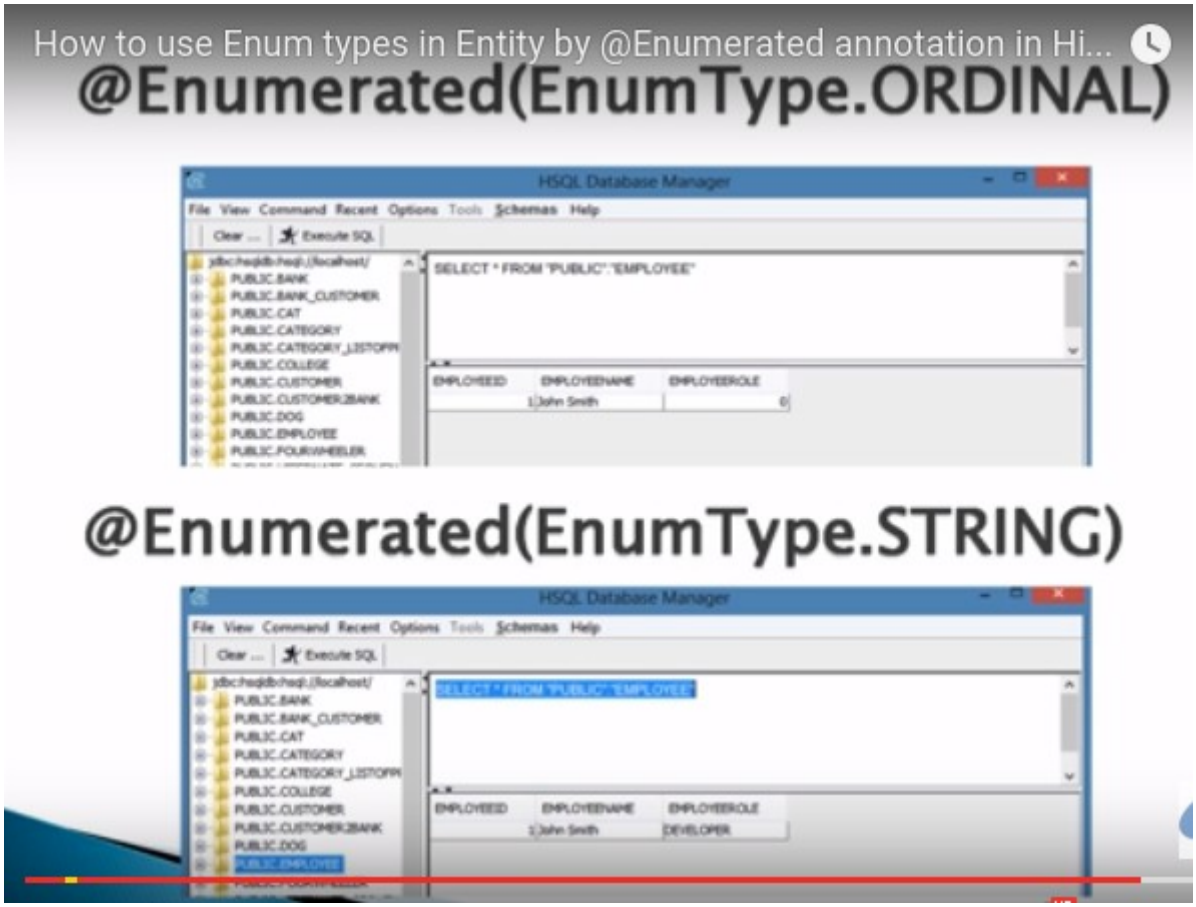
How to use Enum types in Entity by @Enumerated annotation in Hi...

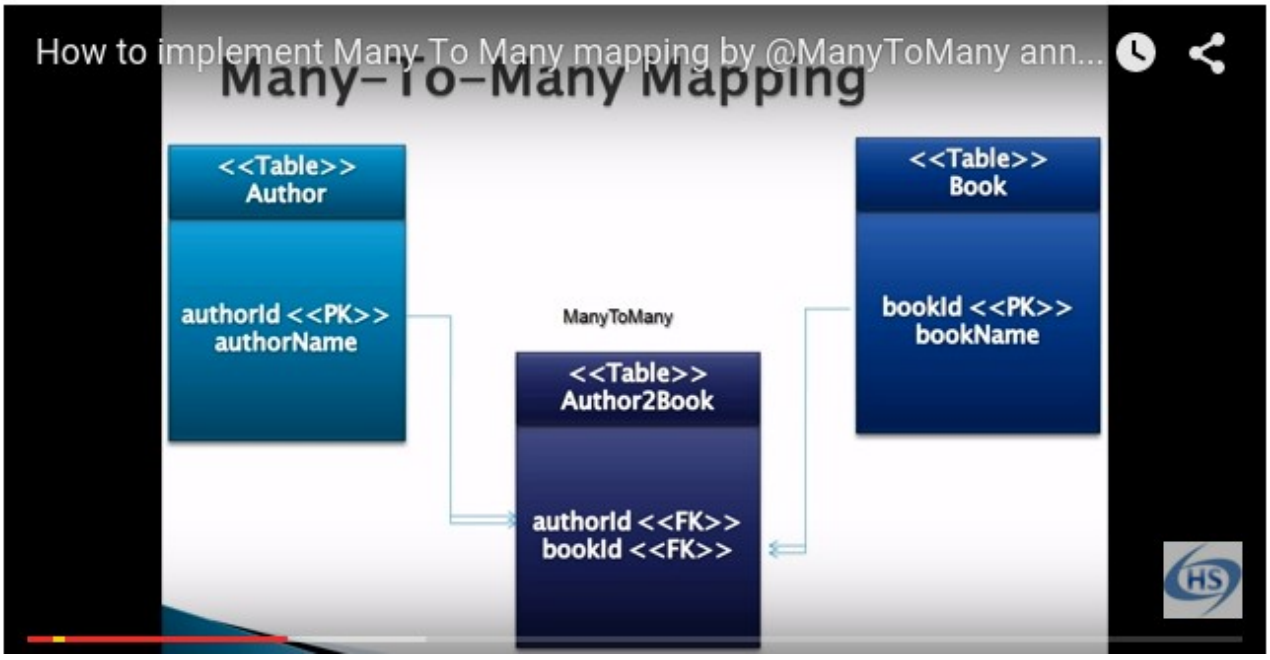
Test class to demonstrate @Enumerated

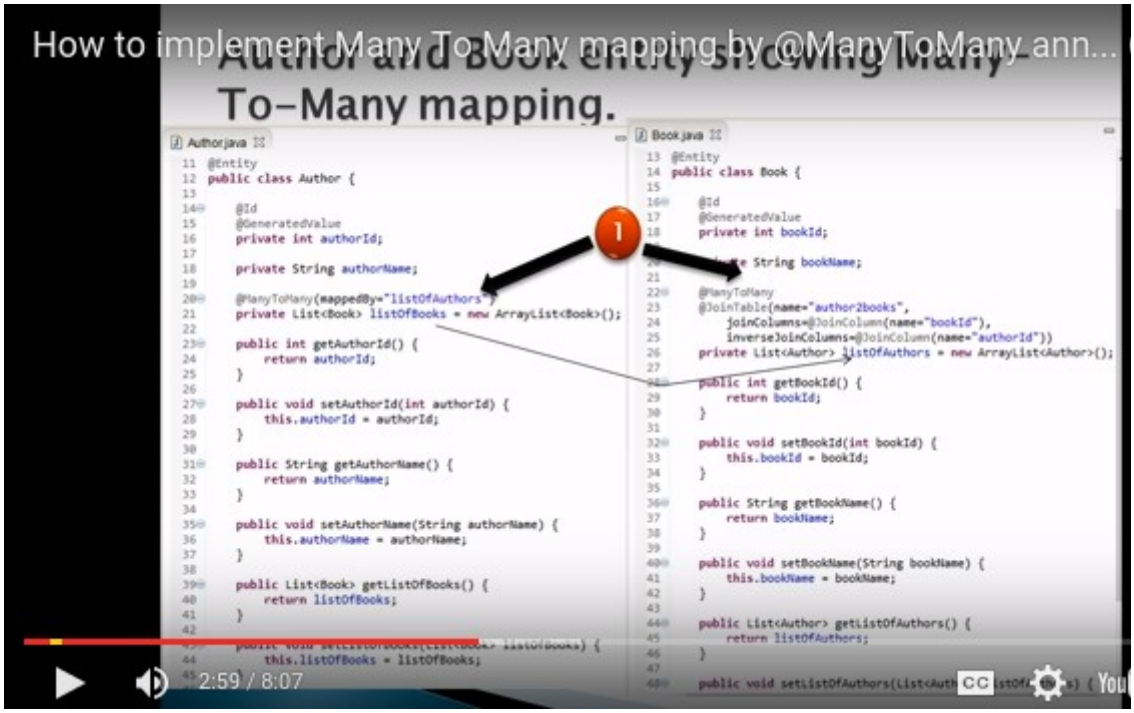
EnumeratedTest.java

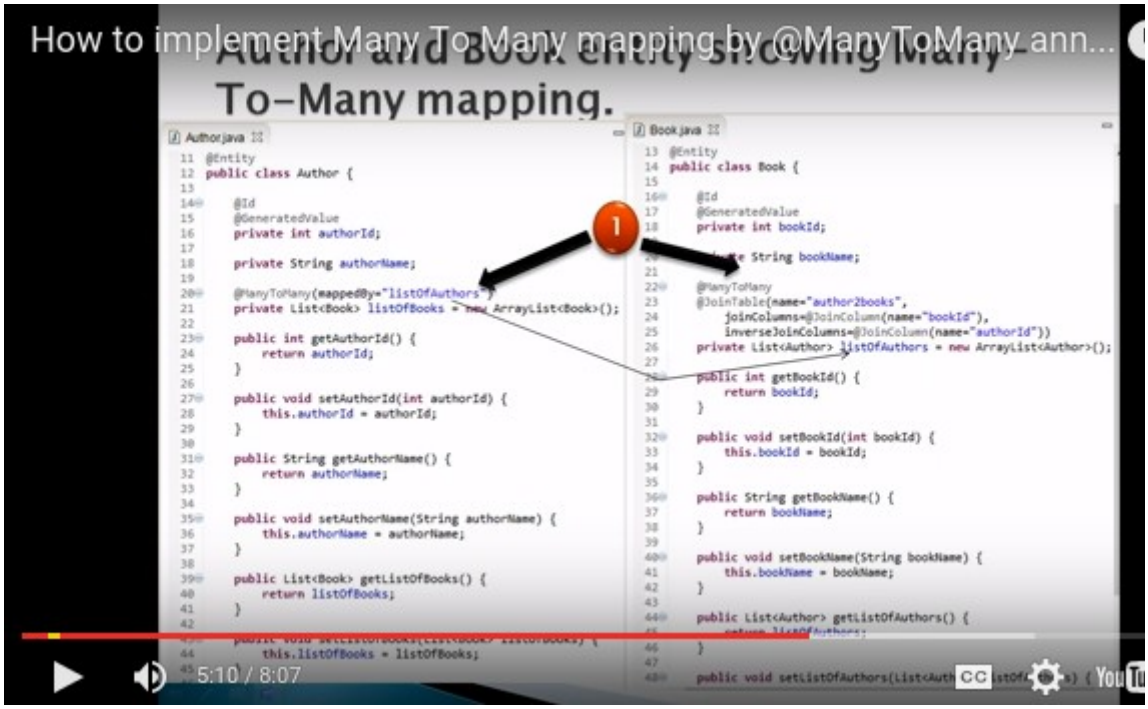
```
10     public static void main(String[] args) {
11
12         Employee employee1 = new Employee();
13         employee1.setEmployeeName("John Smith");
14
15         Role role1 = Role.DEVELOPER;
16
17         employee1.setEmployeeRole(role1);
18
19         EntityManagerFactory entityManagerFactory =
20                 Persistence.createEntityManagerFactory("employeePU");
21
22         EntityManager entityManager = entityManagerFactory.createEntityManager();
23
24         EntityTransaction entityTransaction = null;
25
26         try {
27             entityTransaction = entityManager.getTransaction();
28             entityTransaction.begin();
29             entityManager.persist(employee1);
30             entityTransaction.commit();
31         }
32         catch(RuntimeException e) {
33             if(entityTransaction.isActive())
34                 entityTransaction.rollback();
35             throw e;
```

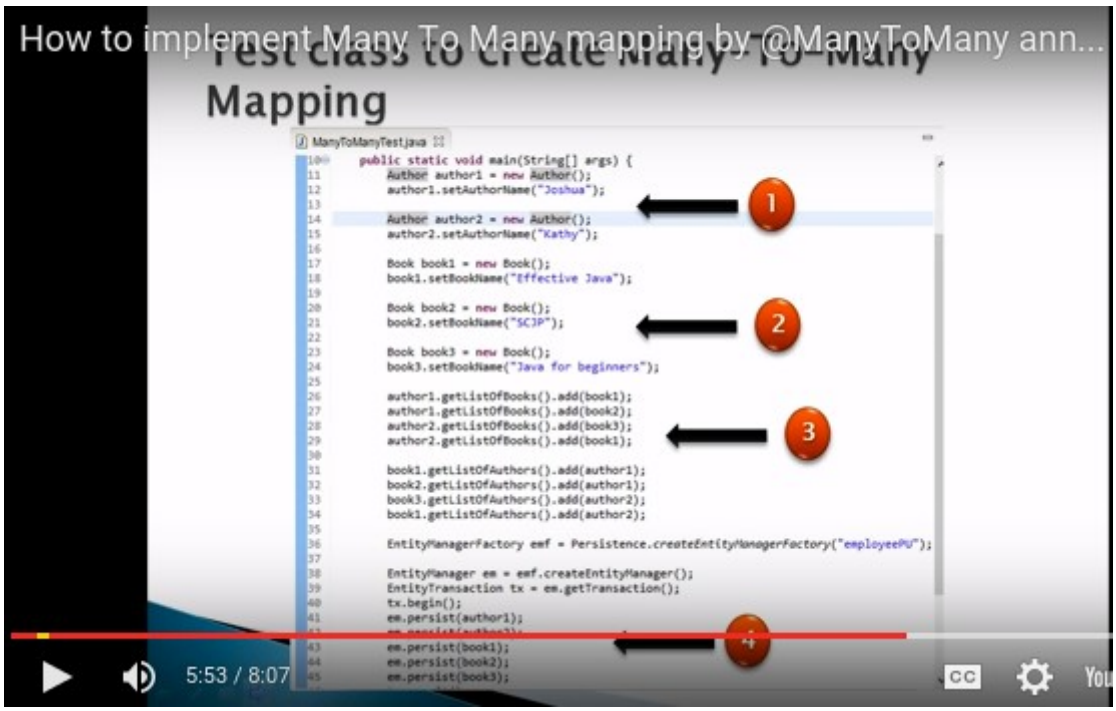
1 2 3 4

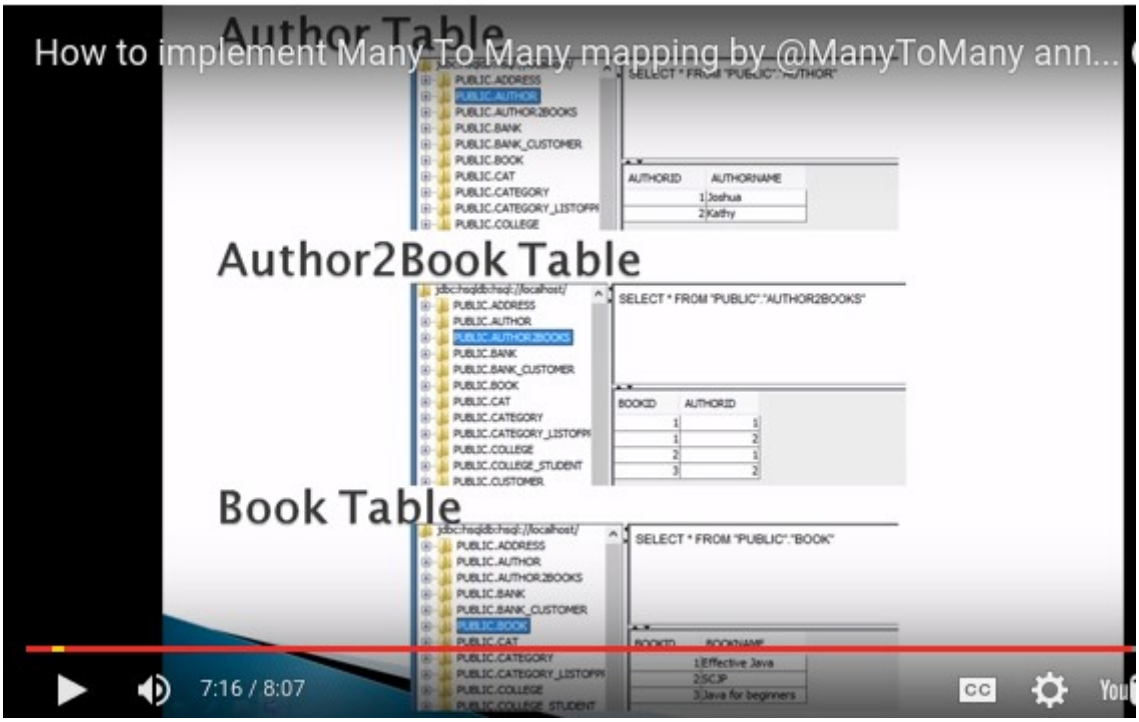

How to use Enum types in Entity by @Enumerated annotation in Hi...
@Enumerated(EnumType.ORDINAL)
HSQL Database Manager
File View Command Recent Options Tools Schemas Help
Clear ... Execute SQL
SELECT * FROM "PUBLIC"."EMPLOYEE"
EMPLOYEEID EMPLOYEENAME EMPLOYEEROLE
1 John Smith 0
@Enumerated(EnumType.STRING)
HSQL Database Manager
File View Command Recent Options Tools Schemas Help
Clear ... Execute SQL
SELECT * FROM "PUBLIC"."EMPLOYEE"
EMPLOYEEID EMPLOYEENAME EMPLOYEEROLE
1 John Smith DEVELOPER


How to implement Many To Many mapping by @ManyToMany ann...
Many-To-Many Mapping
<<Table>>
Author
authorId <<PK>>
authorName
ManyToMany
<<Table>>
Author2Book
authorId <<FK>>
bookId <<FK>>
<<Table>>
Book
bookId <<PK>>
bookName


How to implement Many To Many mapping by @ManyToMany ann...
To-Many mapping.
Author.java
Book.java
2:59 / 8:07


How to implement Many To Many mapping by @ManyToMany ann...
To-Many mapping.
Author.java
Book.java
5:10 / 8:07


How to implement Many To Many mapping by @ManyToMany ann...
Test class to create Many To Many Mapping
ManyToManyTest.java
public static void main(String[] args) {
Author author1 = new Author();
author1.setAuthorName("Joshua");
Author author2 = new Author();
author2.setAuthorName("Kathy");
Book book1 = new Book();
book1.setBookName("Effective Java");
Book book2 = new Book();
book2.setBookName("SCJP");
Book book3 = new Book();
book3.setBookName("Java for beginners");
author1.getListOfBooks().add(book1);
author1.getListOfBooks().add(book2);
author2.getListOfBooks().add(book3);
author2.getListOfBooks().add(book1);
book1.getListOfAuthors().add(author1);
book2.getListOfAuthors().add(author1);
book3.getListOfAuthors().add(author2);
book1.getListOfAuthors().add(author2);
EntityManagerFactory emf = Persistence.createEntityManagerFactory("employeePU");
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
em.persist(book1);
em.persist(book2);
em.persist(book3);
1
2
3
4
5:53 / 8:07
CC


How to implement Many To Many mapping by @ManyToMany ann...
Author Table
PUBLIC.ADDRESS
PUBLIC.AUTHOR
PUBLIC.AUTHOR2BOOKS
PUBLIC.BANK
PUBLIC.BANK_CUSTOMER
PUBLIC.BOOK
PUBLIC.CAT
PUBLIC.CATEGORY
PUBLIC.CATEGORY_LISTOFPR
PUBLIC.COLLEGE
SELECT * FROM "PUBLIC"."AUTHOR"
AUTHORID
AUTHORNAME
1 Joshua
2 Kathy
Author2Book Table
PUBLIC.ADDRESS
PUBLIC.AUTHOR
PUBLIC.AUTHOR2BOOKS
PUBLIC.BANK
PUBLIC.BANK_CUSTOMER
PUBLIC.BOOK
PUBLIC.CAT
PUBLIC.CATEGORY
PUBLIC.CATEGORY_LISTOFPR
PUBLIC.COLLEGE
PUBLIC.COLLEGE_STUDENT
PUBLIC.CUSTOMER
SELECT * FROM "PUBLIC"."AUTHOR2BOOKS"
BOOKID
AUTHORID
1 1
1 2
2 1
3 2
Book Table
PUBLIC.ADDRESS
PUBLIC.AUTHOR
PUBLIC.AUTHOR2BOOKS
PUBLIC.BANK
PUBLIC.BANK_CUSTOMER
PUBLIC.BOOK
PUBLIC.CAT
PUBLIC.CATEGORY
PUBLIC.CATEGORY_LISTOFPR
PUBLIC.COLLEGE
PUBLIC.COLLEGE_STUDENT
SELECT * FROM "PUBLIC"."BOOK"
1 Effective Java
2 SCJP
3 Java for beginners
7:16 / 8:07
CC