FPT Education
FPT UNIVERSITY

Java

Java Persistence API

Objectives

- Explain what JPA is and its role in data persistence for Java applications.
- Highlight the advantages of using JPA over traditional JDBC approaches.
- Define and illustrate core JPA concepts like entities, annotations, persistence context, entity manager and JPQL.
- Provide a clear understanding of how these components work together to manage data.
- Show how JPA simplifies development by reducing boilerplate code and allo wing focus on business logic.

Contents

- Introduction
- Key Concepts
- Annotations
- Relationships
- Using JPA
- Demo
- Advantages and Disadvantages

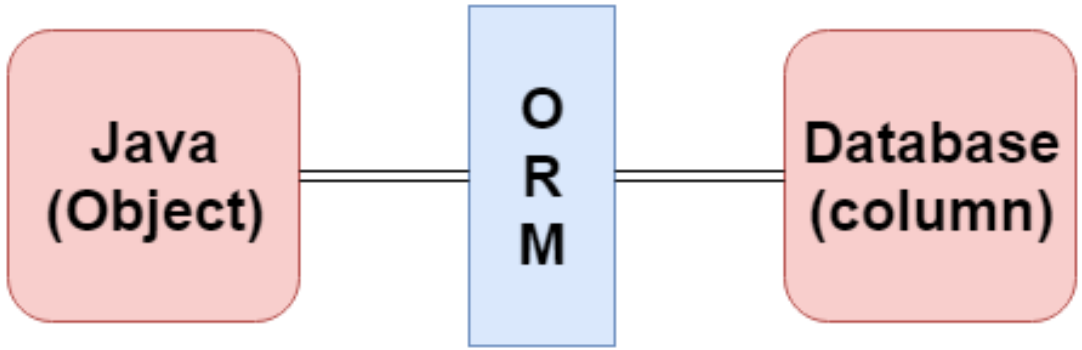
FPT Education
FPT UNIVERSITY

Java

Object Relational Mapping (ORM)

JPA Object Relational Mapping

- ORM makes it easier to work with databases in object-oriented applications, allowing you to focus on the business logic rather than the underlying data storage details.



Benefits of ORM

- **Increased Productivity:** You write less code because ORM handles the low-level data access tasks.
- **Improved Maintainability:** Your code is cleaner and easier to understand because it focuses on the business logic, not database details.
- **Enhanced Performance:** ORM frameworks can optimize data access and caching, potentially improving performance.
- **Database Independence:** You can switch databases without rewriting your a pplication code (to some extent)

Popular ORM Frameworks for Java

- **Hibernate:** The most widely used JPA implementation, offering a rich feature set and extensive customization options.
- **EclipseLink:** Another JPA implementation with a focus on standards complia nce and portability.
- **Spring Data JPA:** An abstraction layer on top of JPA providers, simplifying d ata access and reducing boilerplate code.
- **DataNucleus** providing easy persistence to RDBMS datastores. Comes with its own "SQL-like" JPQL query language, so you query your data in a language similar to what your datastore understands.

FPT Education
FPT UNIVERSITY

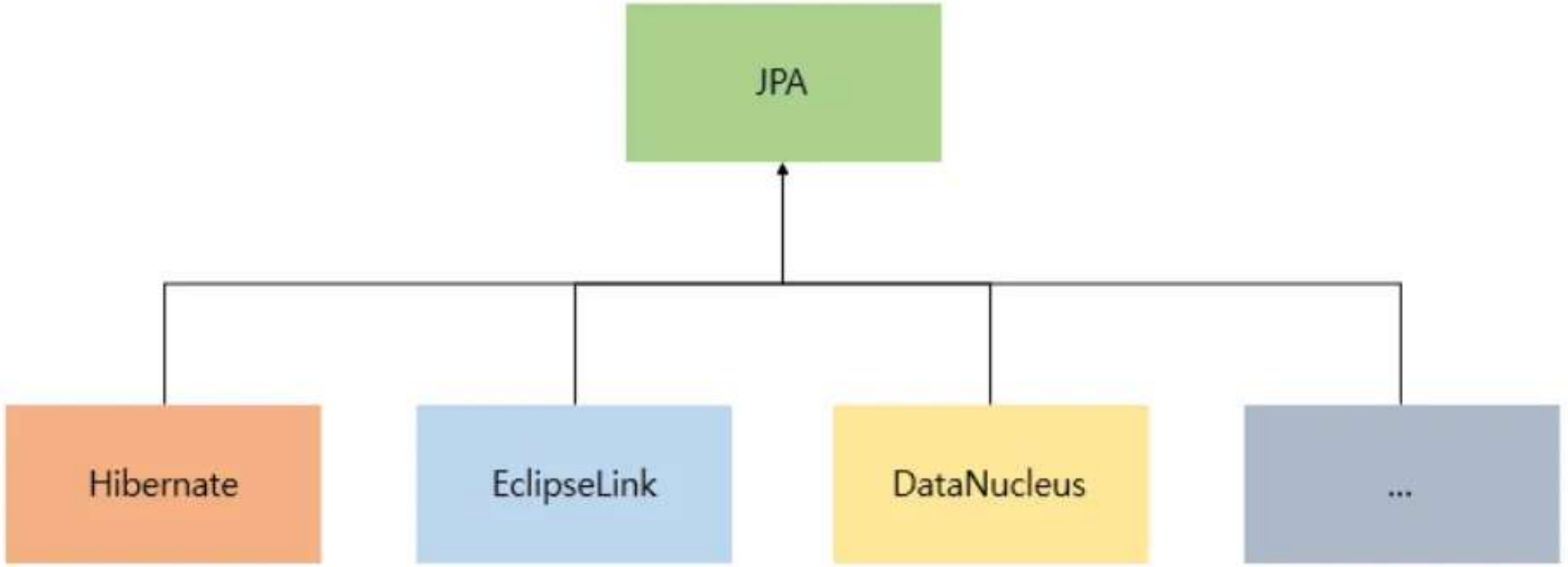
Java

Java Persistence API (JPA)

JPA Versions and Key Features

JPA Version	Release Year	Key Features
JPA 1.0	2006	Core ORM functionality: entities, mappings, relationships, inheritance, JPQL, EntityManager
JPA 2.0	2009	Enums, Criteria API, embeddable collections, derived properties, validation, metamodel
JPA 2.1	2013	Entity graphs, converters, stored procedures, Java 8 date/time
JPA 2.2	2019	Streamlined bootstrap, Java 9 modules
JPA 3.0	2022	Renamed to Jakarta Persistence, Java records, Java 17 support

JPA Implementation Options



What is JPA ?

- JPA stands for Java Persistence API. It is a Java programming interface that allows developers to manage relational data in Java applications using object oriented methodologies.
- JPA is a specification for managing relational data in Java applications. It provides a set of interfaces, annotations, and an object relational mapping (ORM) framework that simplifies the process of interacting with databases.

Key Feature of JPA

- **ORM**: JPA allows you to map Java objects (entities) to relational database tables, abstracting away the differences between the object-oriented and relational models.
- **Annotations**: You can use annotations to define entities, relationships and other persistence related metadata.
- **EntityManager**: The interface provides methods for persisting, retrieving, updating, and deleting entities.
- **JPQL** is an object oriented query language that allows you to query entities and their relationships.
- **Transaction Management**: JPA supports transaction management, ensuring data integrity and consistency.

Benefits of Using JPA

- **Simplified Data Access:** JPA makes it easier to work with relational data in Ja va applications by providing a higher-level abstraction.
- **Code Portability:** JPA is a standard specification, so code written using JPA can be portable across different JPA providers.
- **Reduced Boilerplate Code:** JPA annotations and the EntityManager interface reduce the amount of code needed for data access.
- **Improved Data Integrity:** JPA's transaction management and validation features help ensure data integrity.
- **Enhanced Performance:** Caching and other optimization techniques can improve the performance of JPA applications.

Entity Annotations in JPA

- **@Entity**: This annotation marks a Java class as an entity, meaning it represents a table in the database.
- **@Table:** This annotation allows you to specify the name of the database table that an entity maps to.
- **@Transient:** This annotation marks a property that should not be persisted to the database.
- **@NamedQueries and @NamedQuery:** These annotations allow you to define named queries that can be used to retrieve entities.

Mapping Annotations in JPA

- **@Id**: This annotation marks a Java class as an entity, meaning it represents a table in the database.
- **@Column:** This annotation defines the mapping between a property of an entity and a column in the database table. You can use it to specify the column name, data type, and other attributes.
- **@Basic:** This annotation specifies that a property is a basic type and should be persisted.
- **@Enumerated:** This annotation is used to map an Enum type to a database column.

JPA Entity Manager

- The entity manager implements the API and encapsulates all of them within a single interface.
  - **Manages entity lifecycle:** Persist, find, merge, remove
  - **Controls persistence context:** Cache, flush, detach
  - **Executes queries:** JPQL, Criteria API
  - **Handles transactions:** Begin, commit, rollback
  - **Provides access to entity metadata**

Key Operations of the Entity Manager

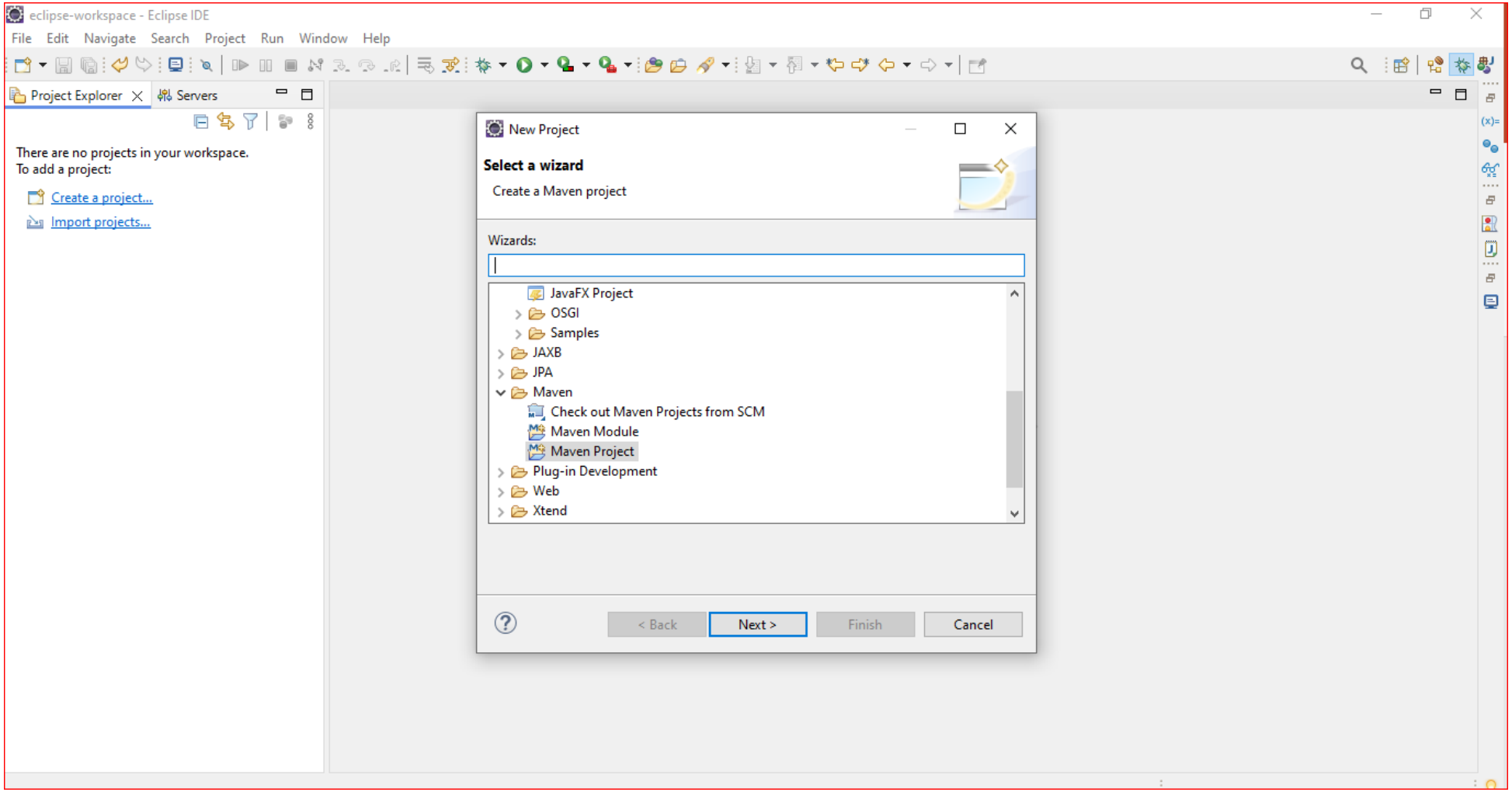
- **Persisting Entities:** *em.persist(entity)* makes a transient entity instance persistent
- **Finding Entities:** *em.find(entityClass,primaryKey)* retrieves an entity by its primary key.
- **Merging Entities:** *em.merge(entity)* merges a detached entity instance into the current persistence context.
- **Removing Entities:** *em.remove(entity)* removes a persistent entity instance.
- **Querying Data:** *em.createQuery(jpqlString)* creates a JPQL query to retrieve entities based on specified criteria.

FPT Education
FPT UNIVERSITY

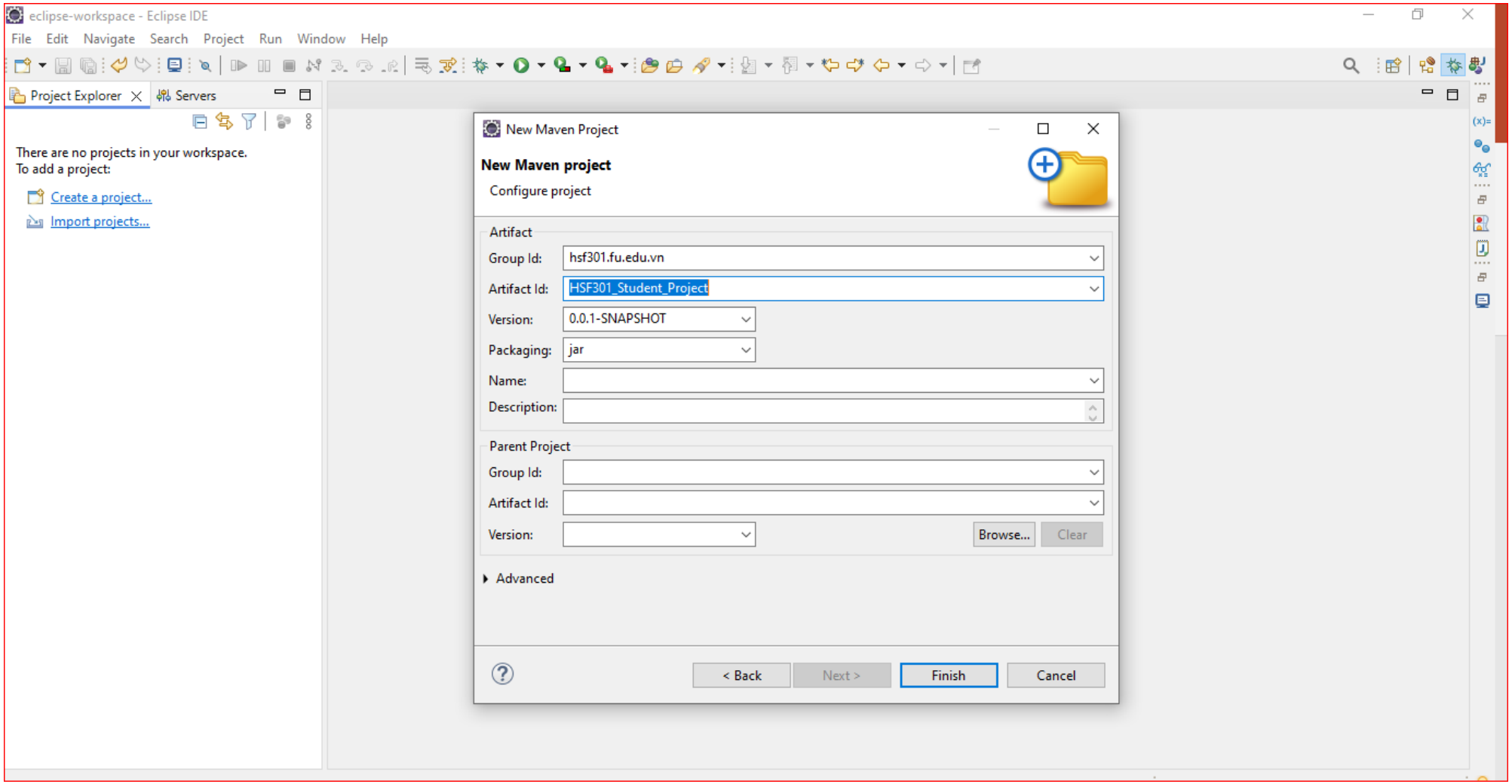
Java

Demo Simple CRUD with JPA

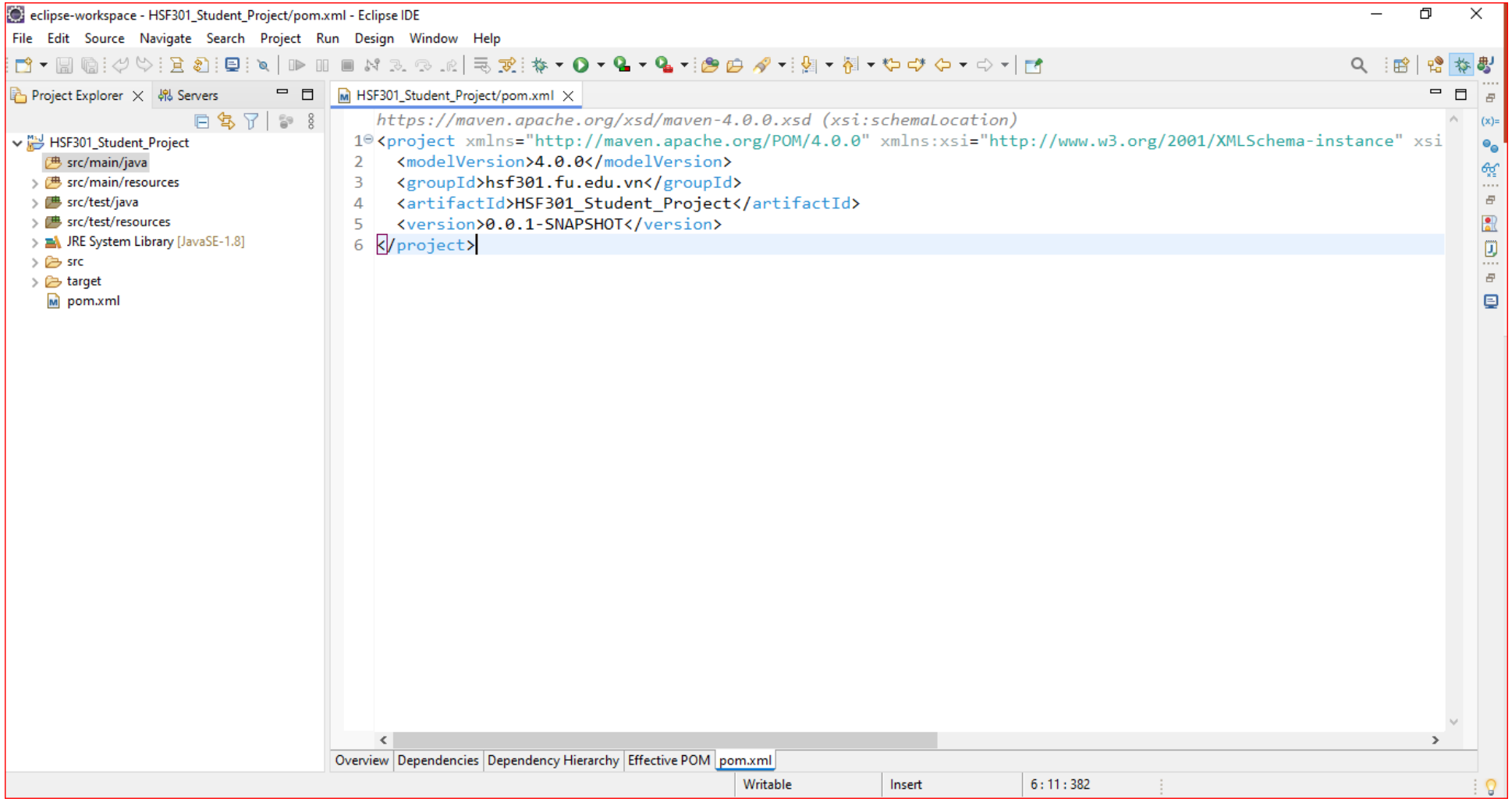
1. Open Eclipse, File | New | Maven Project



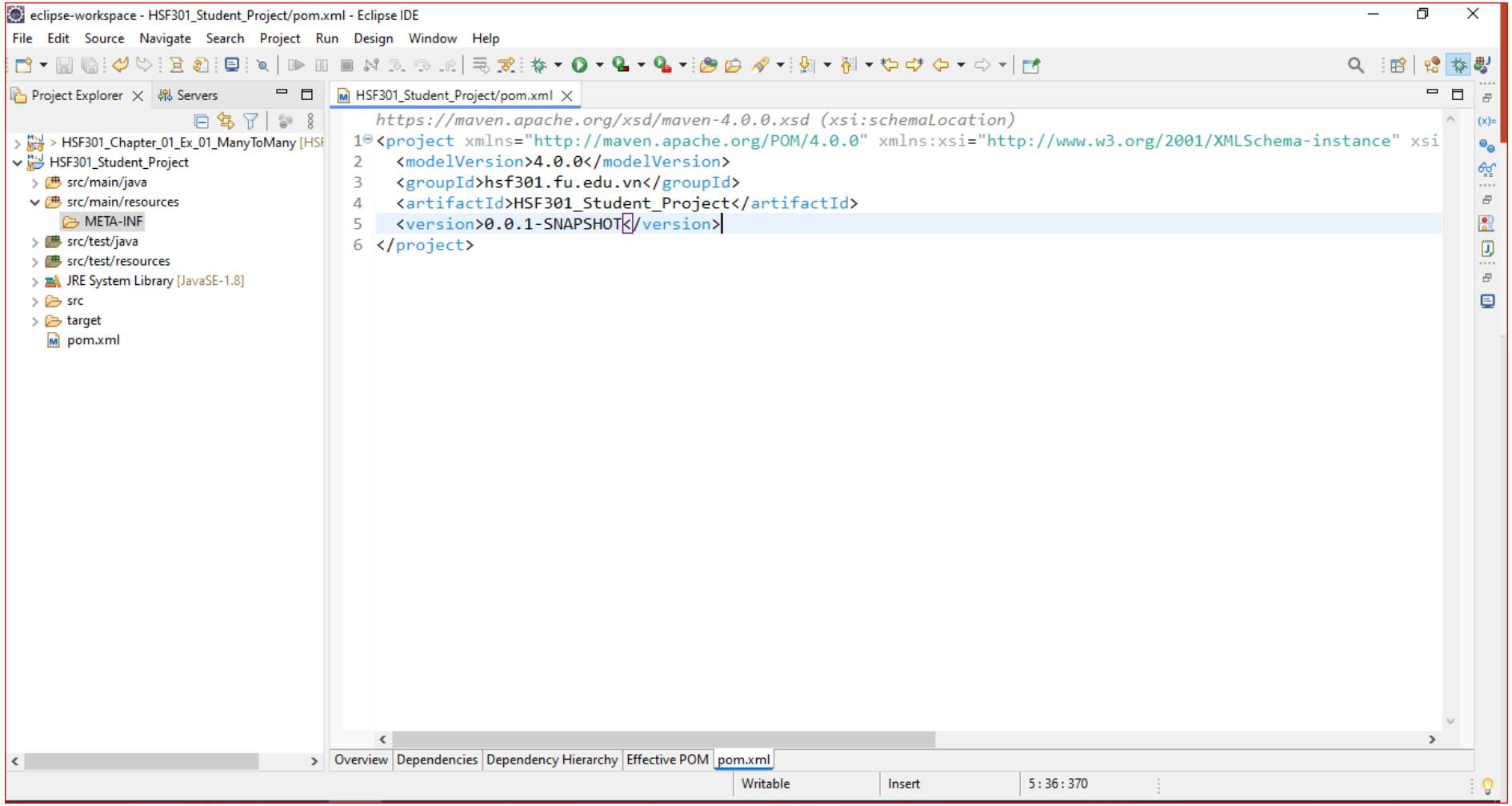
2. Check Create a simple project -> Browse Project -> Next



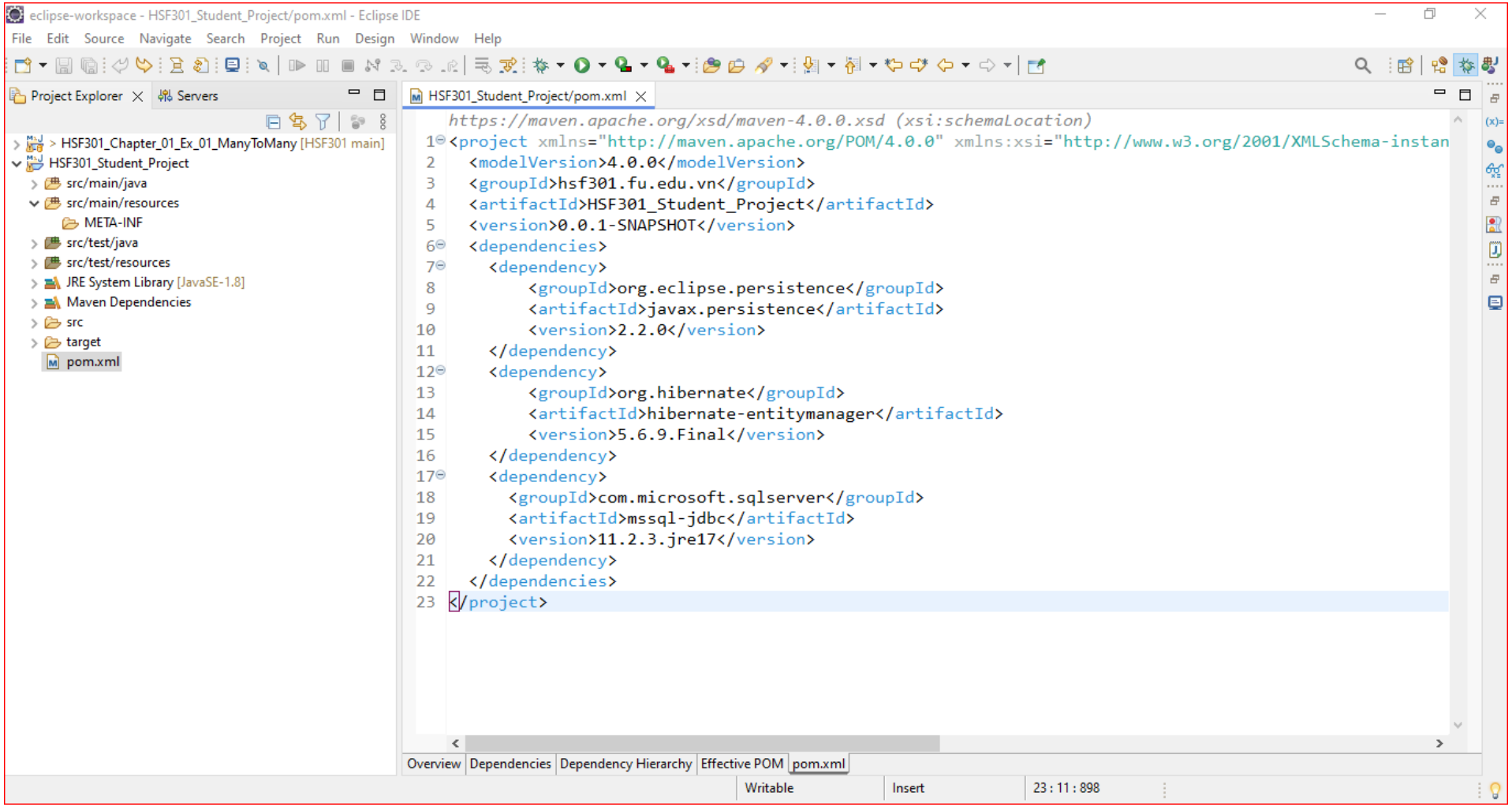
3. The New Project



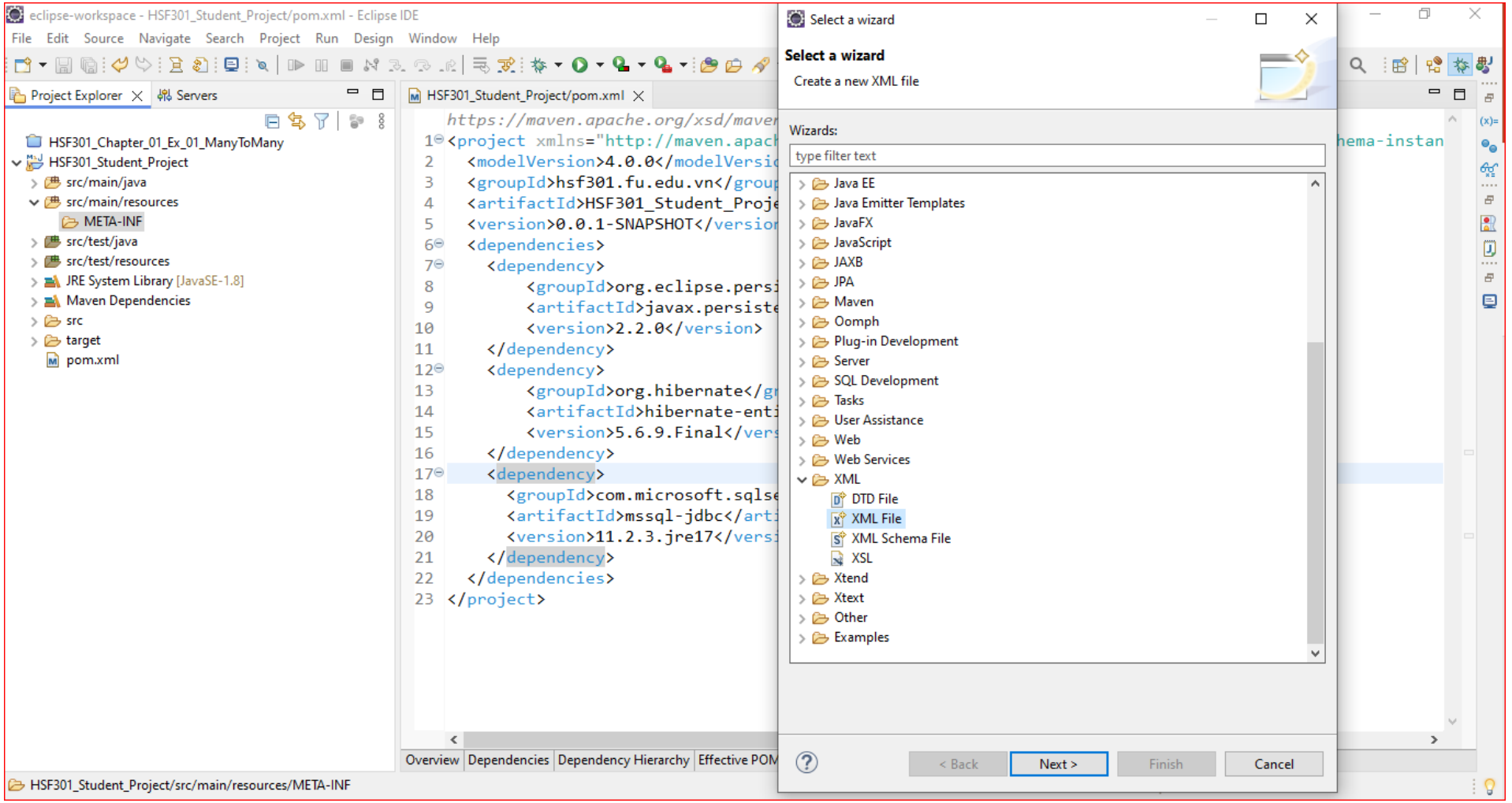
4. Create folder META-INF in src/main/resources



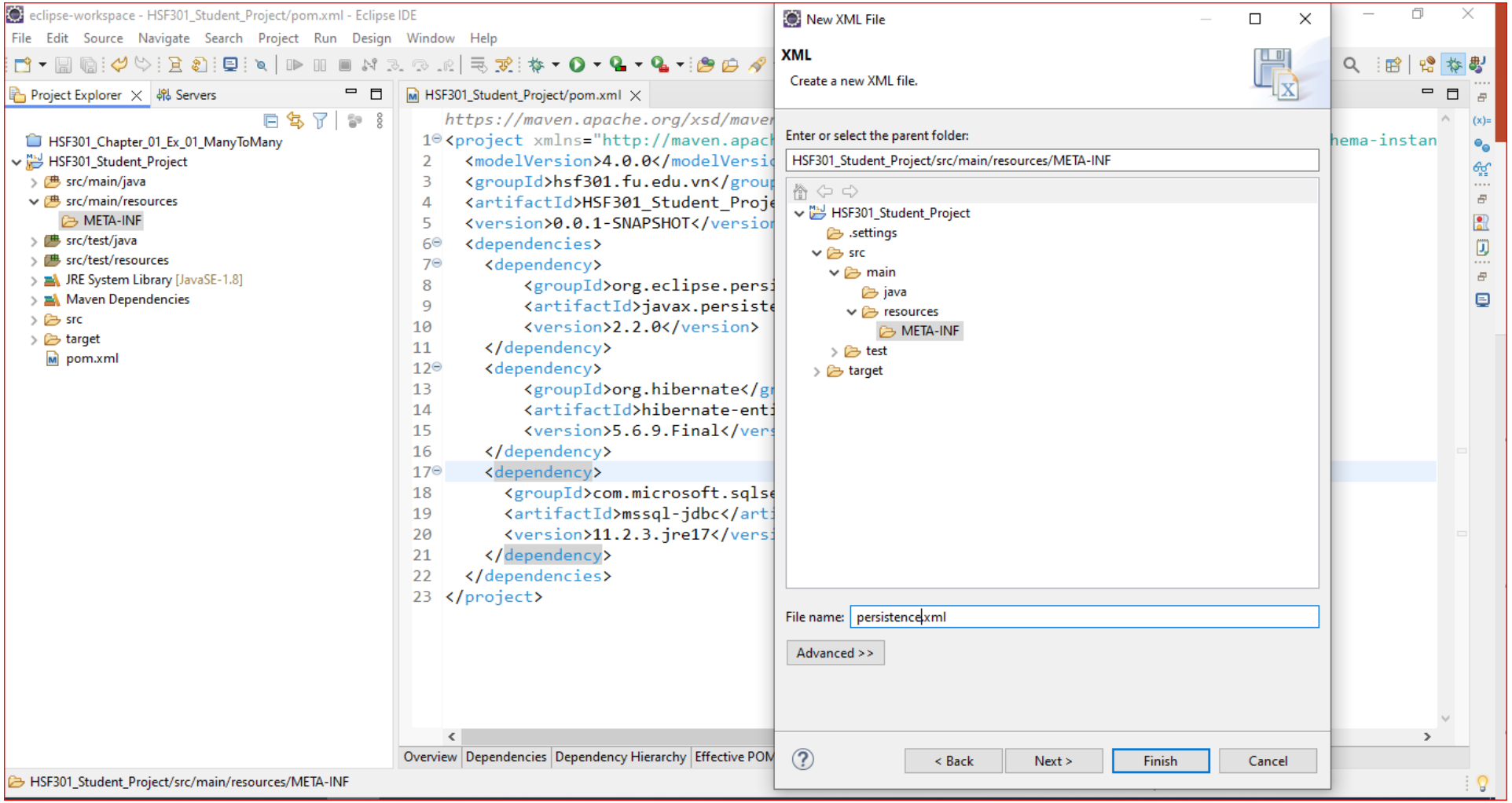
5. Edit the pom.xml



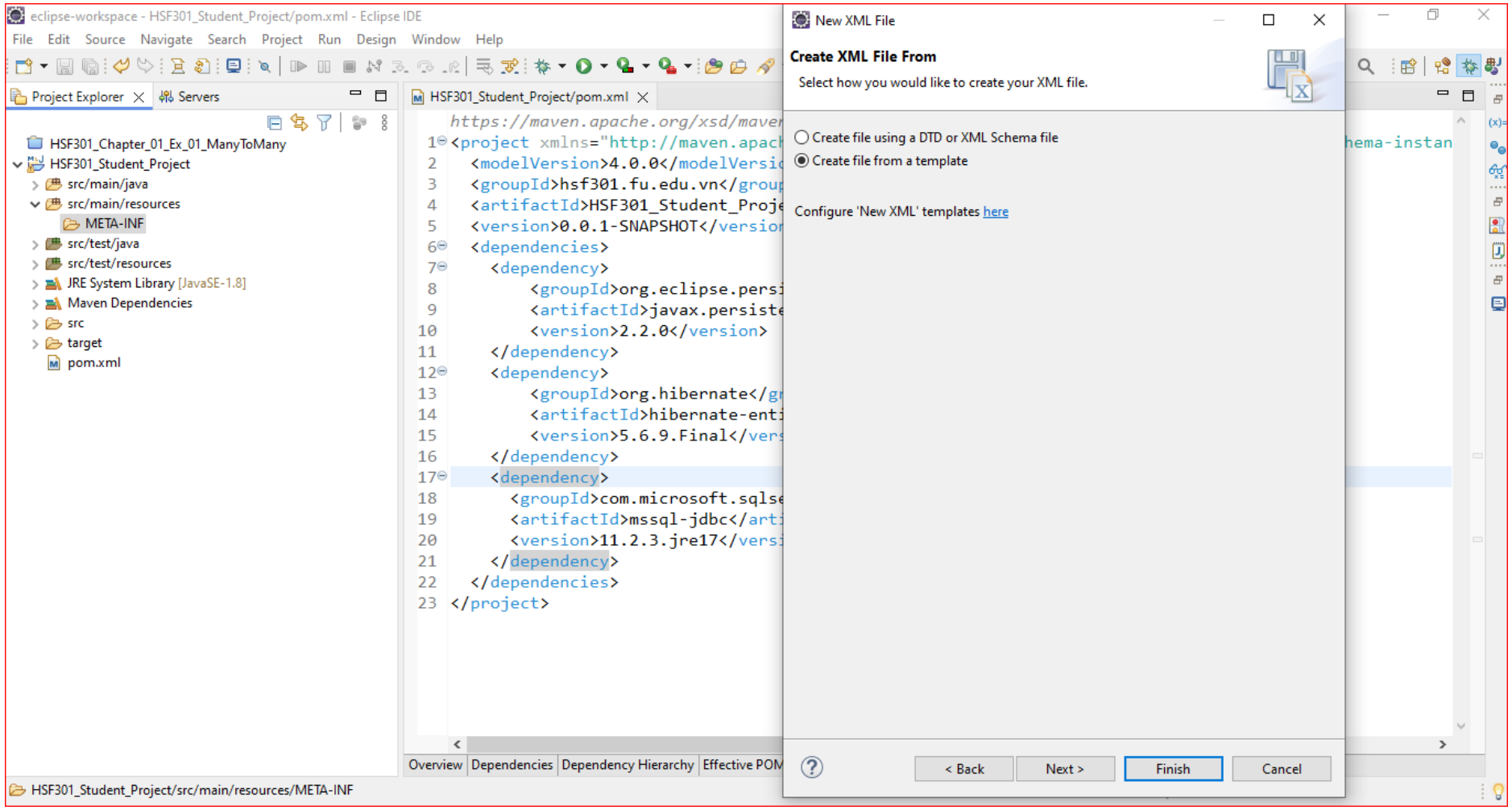
6. Create persistence.xml in META-INF folder



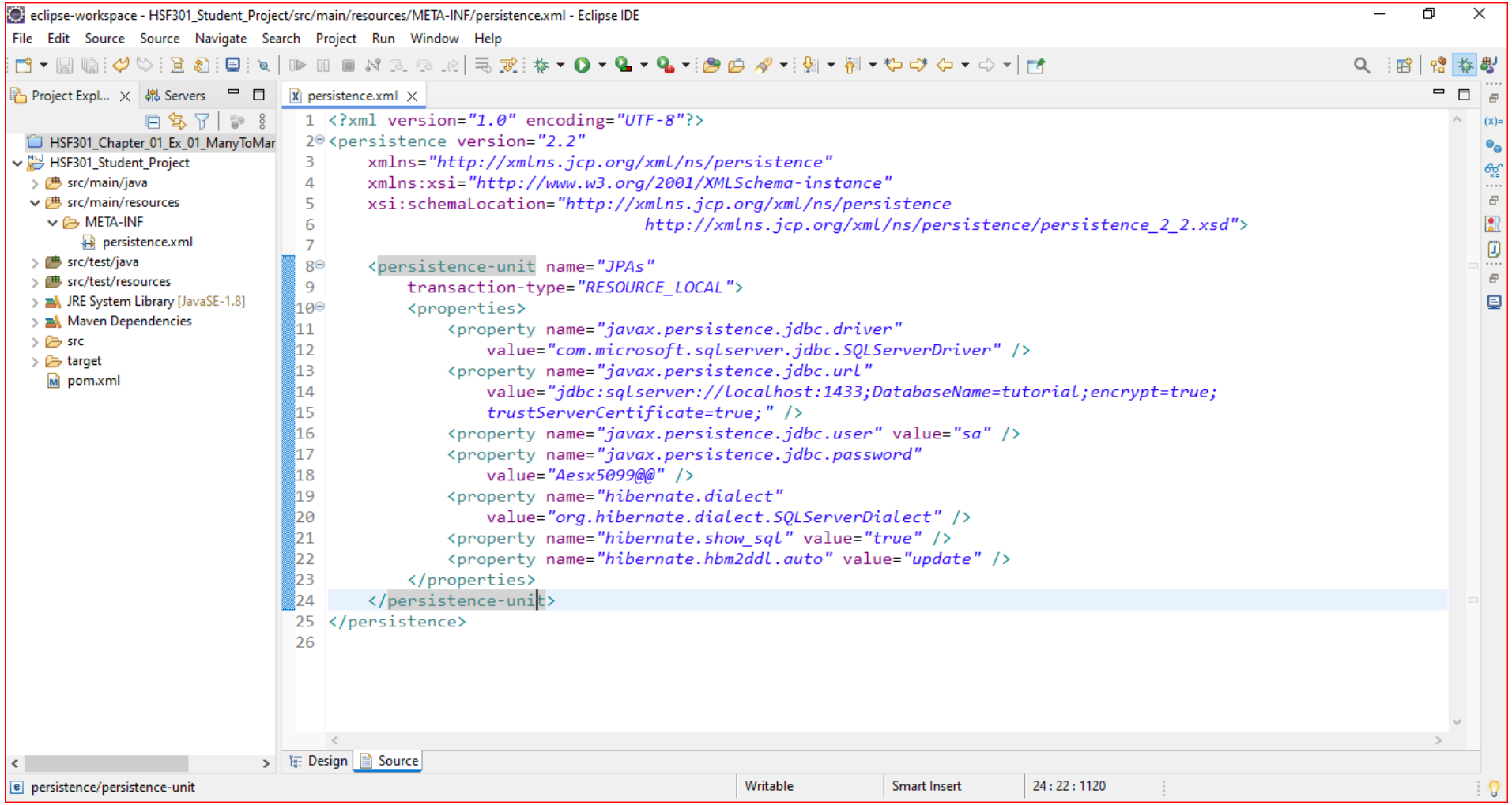
7. Create persistence.xml in META-INF folder



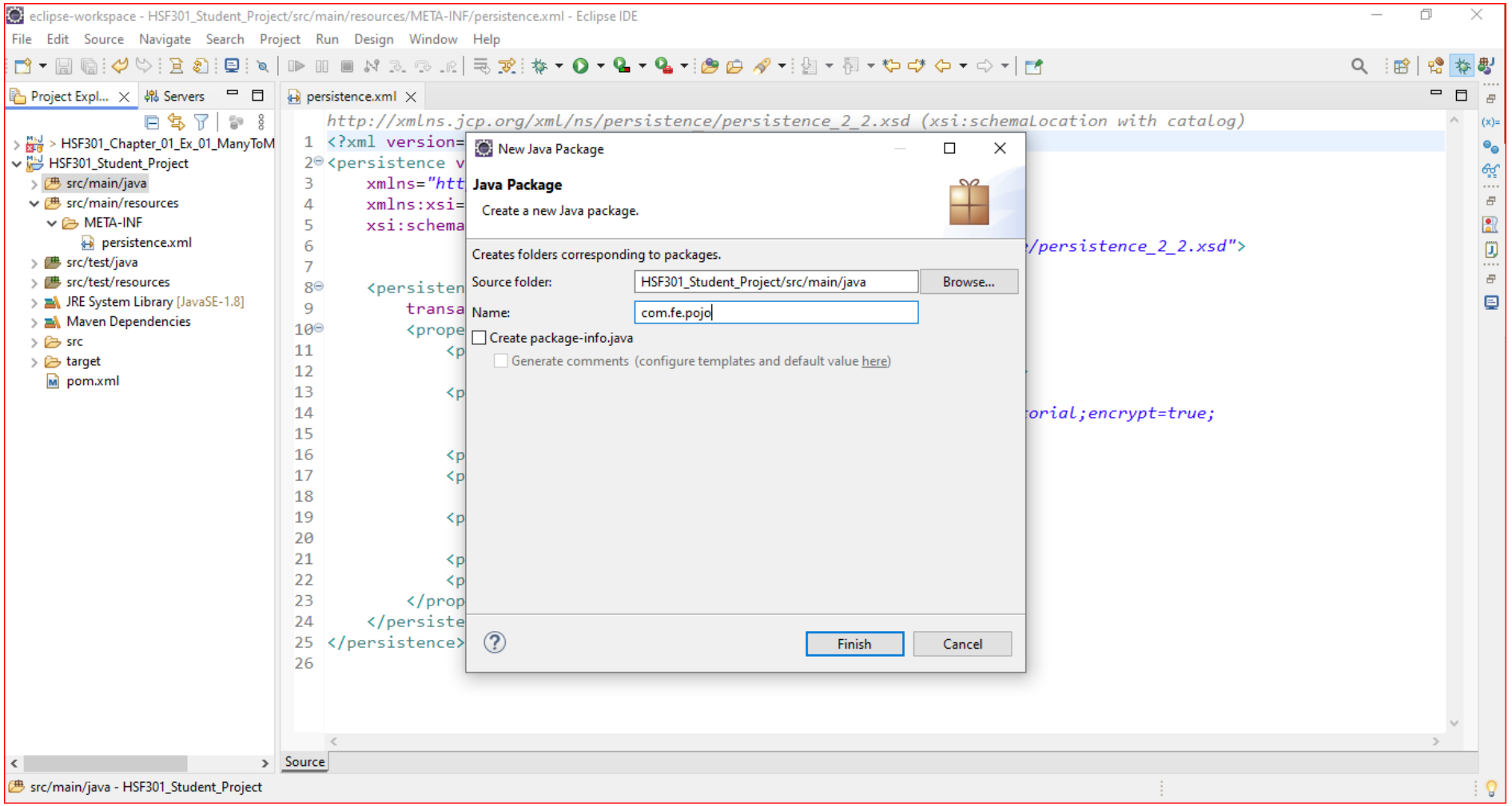
8. Create persistence.xml in META-INF folder



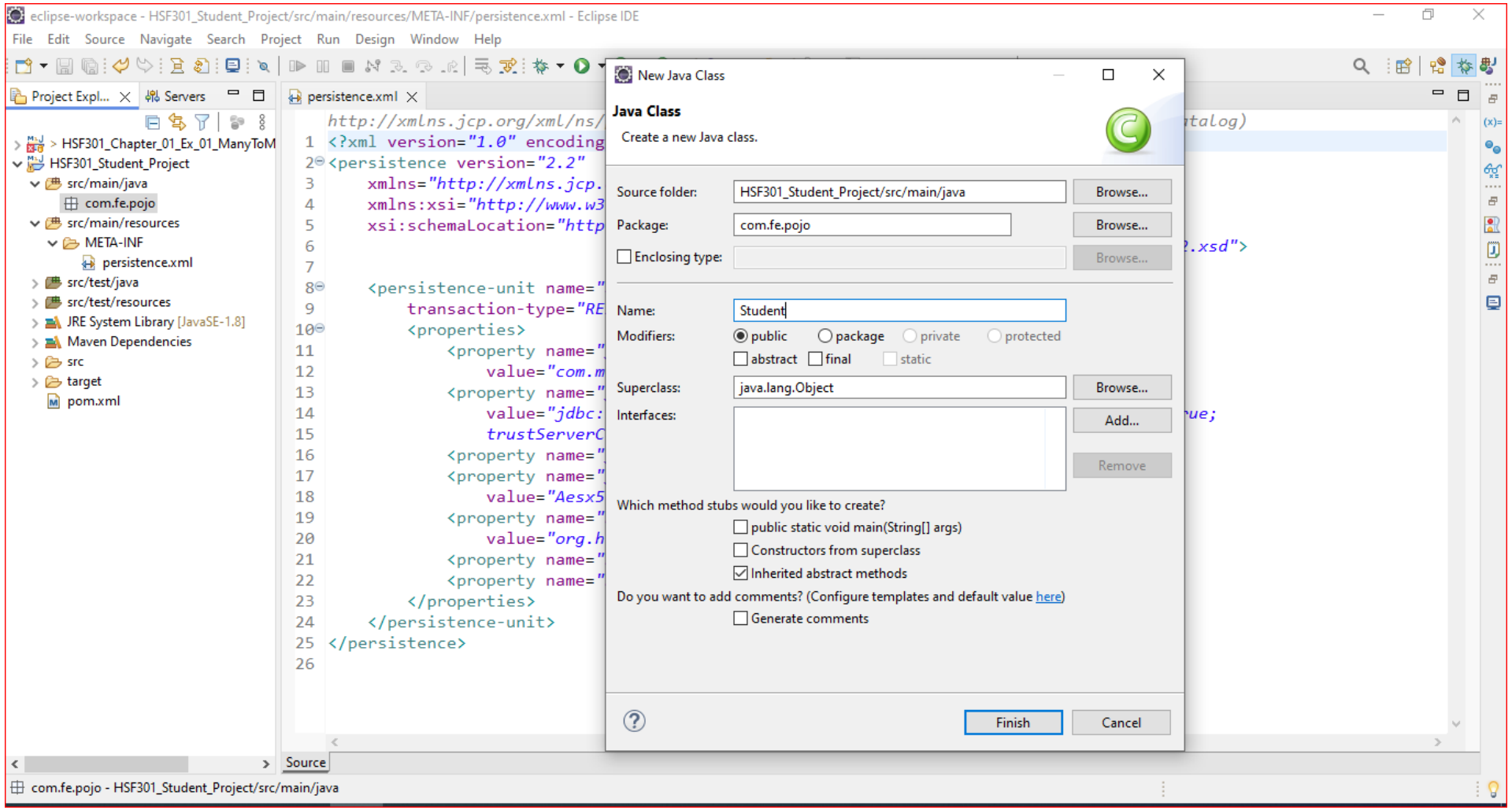
9. Edit the persistence.xml in META-INF folder



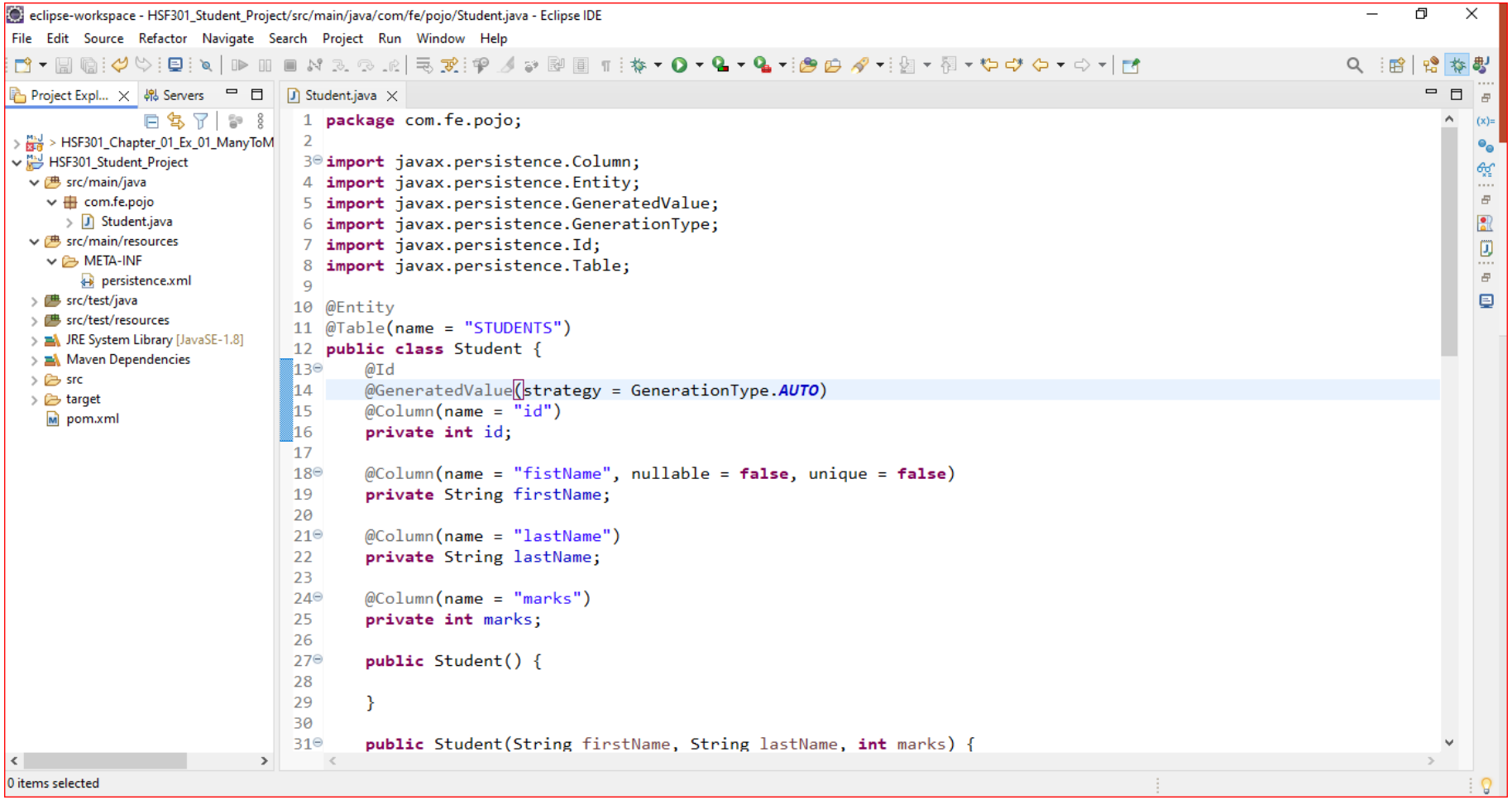
10. Add com.fe.pojo Package in src/main/java



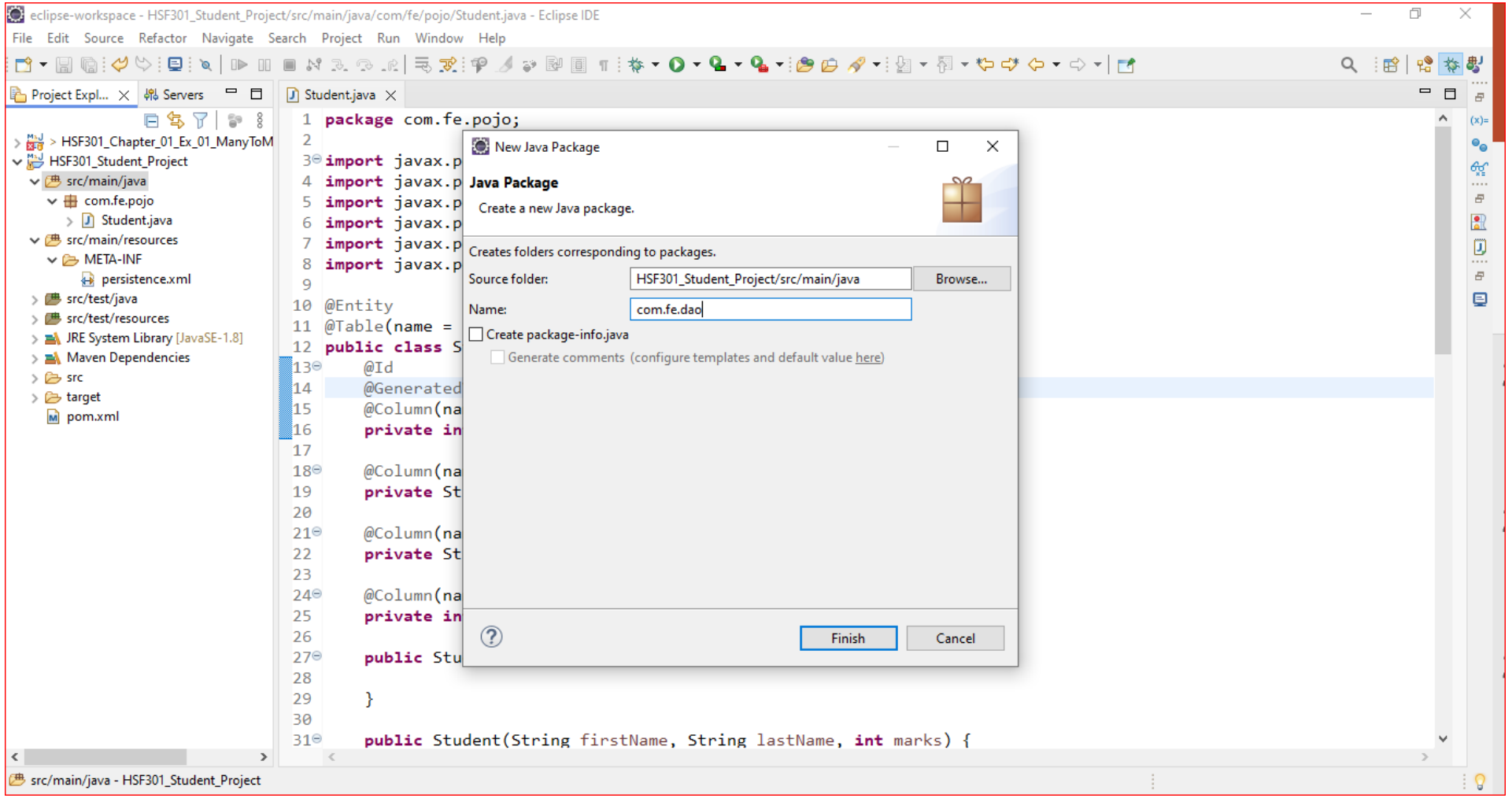
11. Add Student.java in com.fe.pojo package



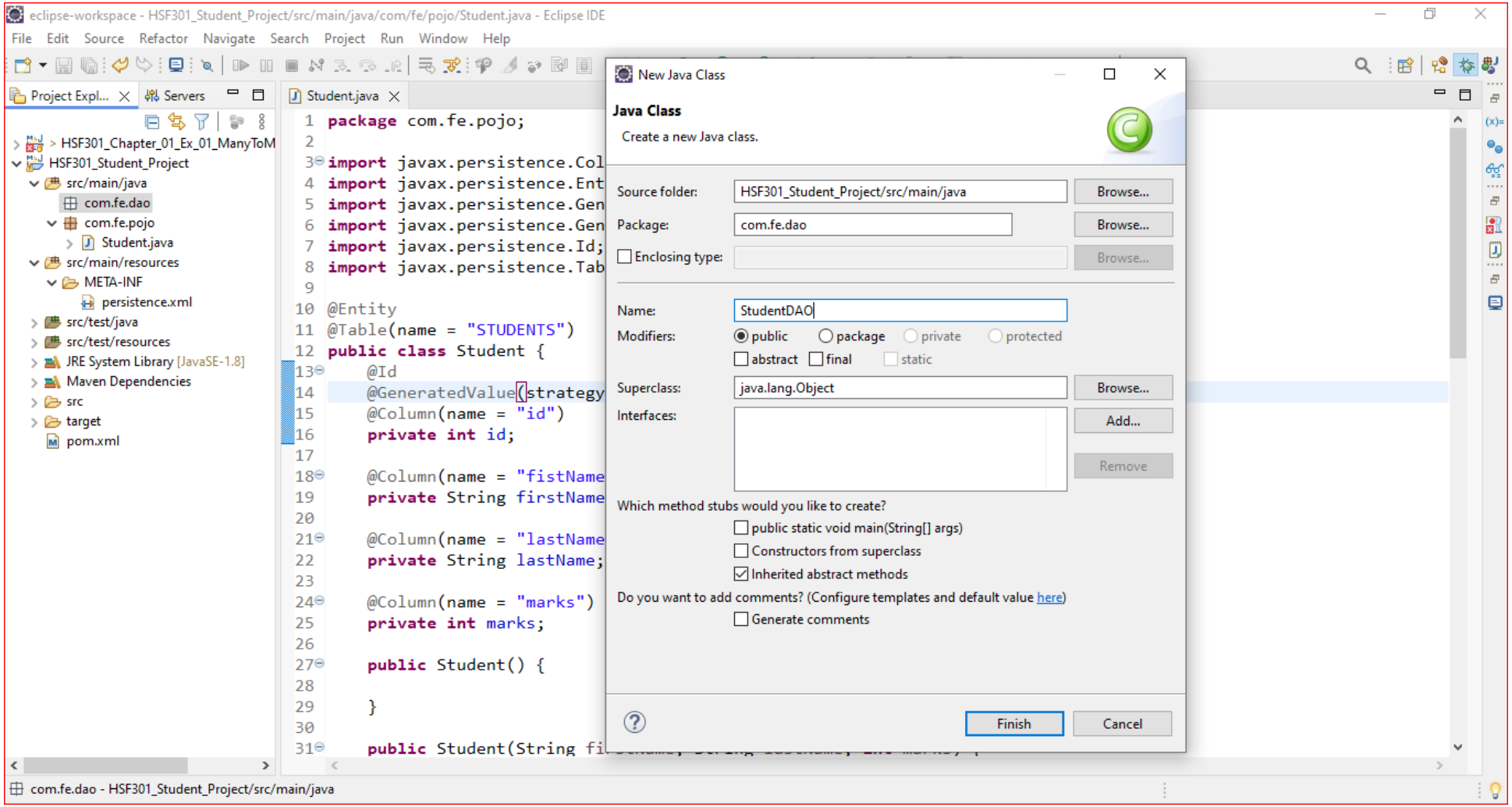
12. Edit the Student.java



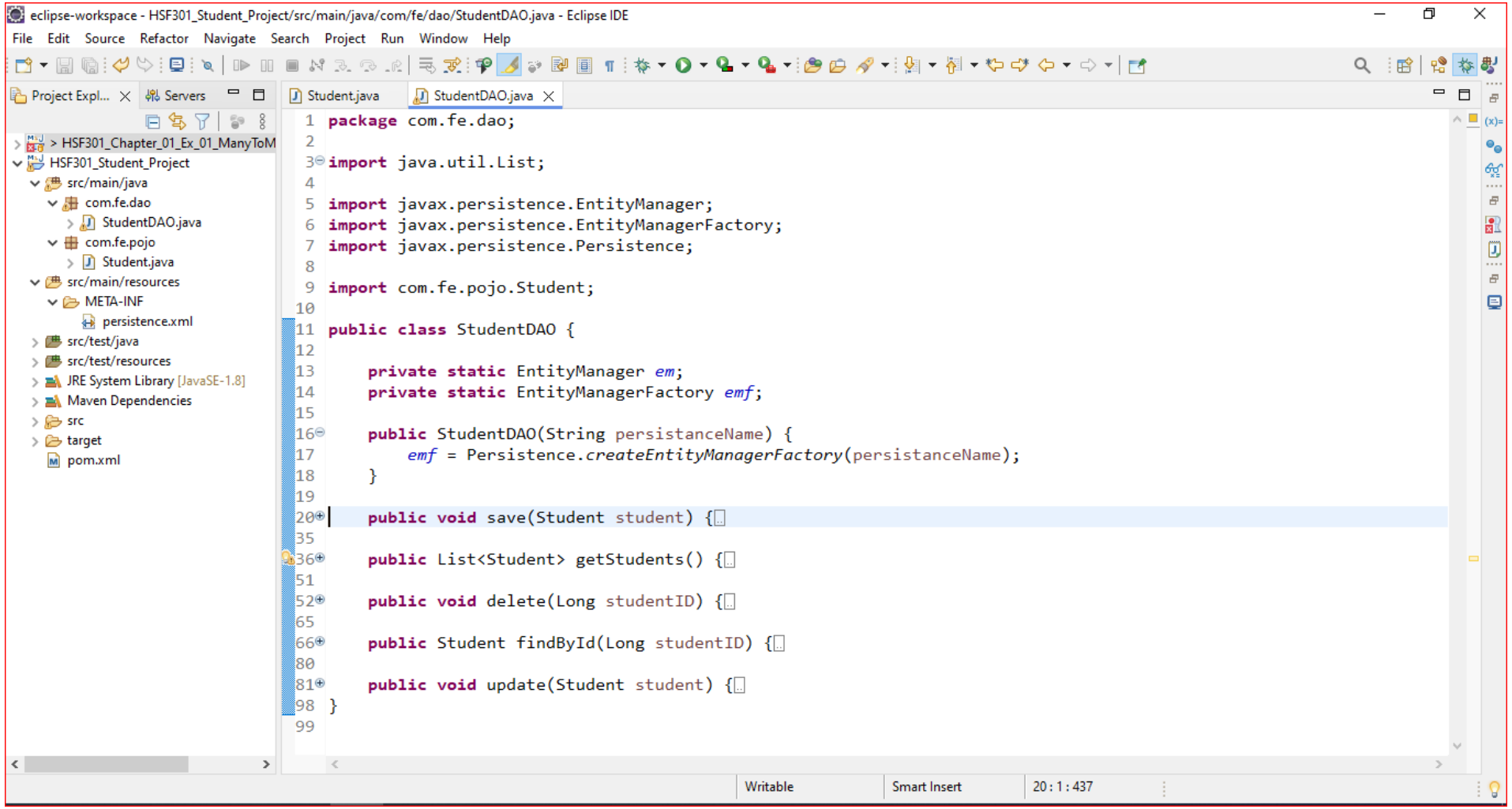
13. Add com.fe.dao Package in src/main/java



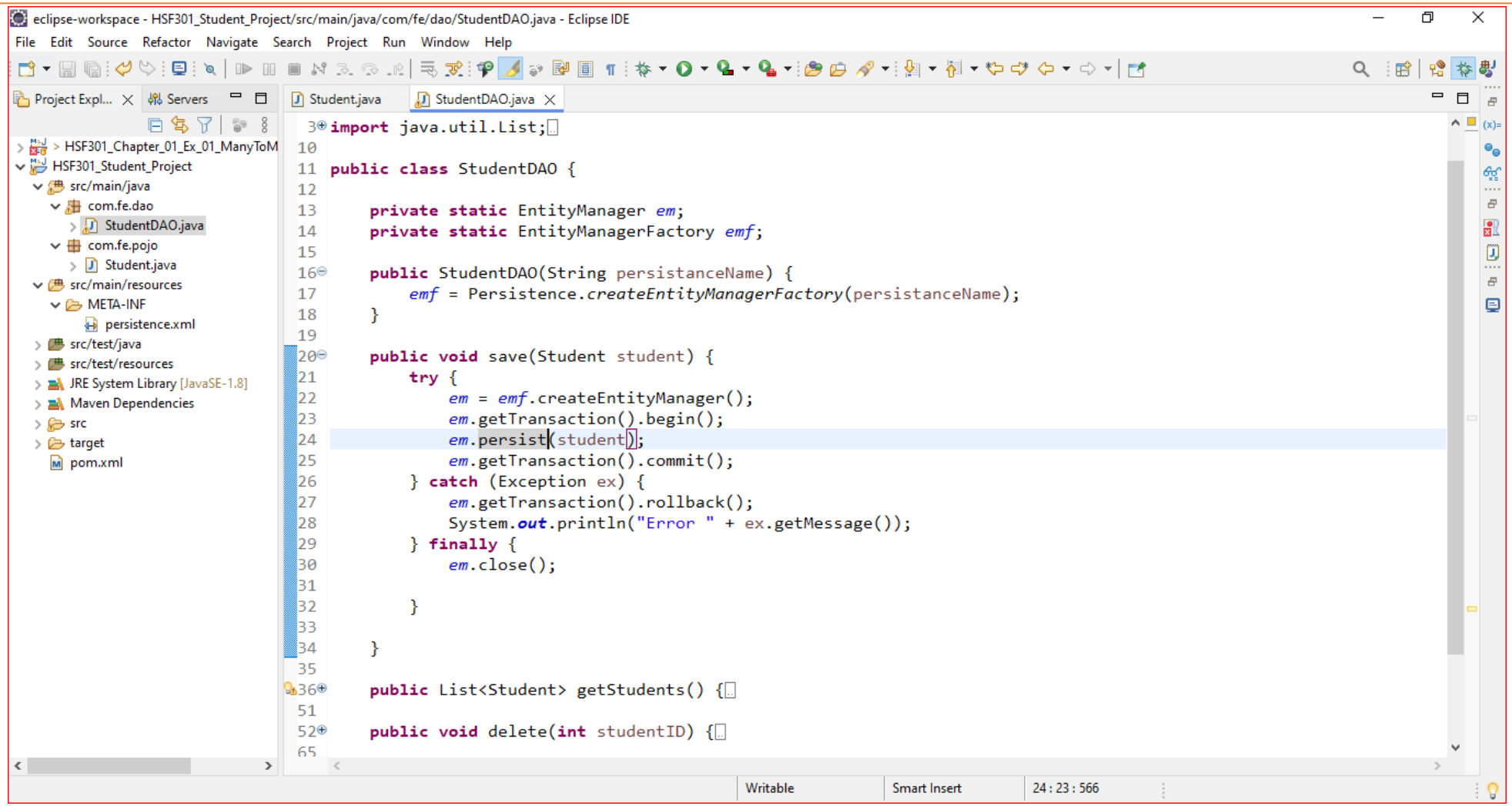
14. Create StudentDAO in com.fe.dao



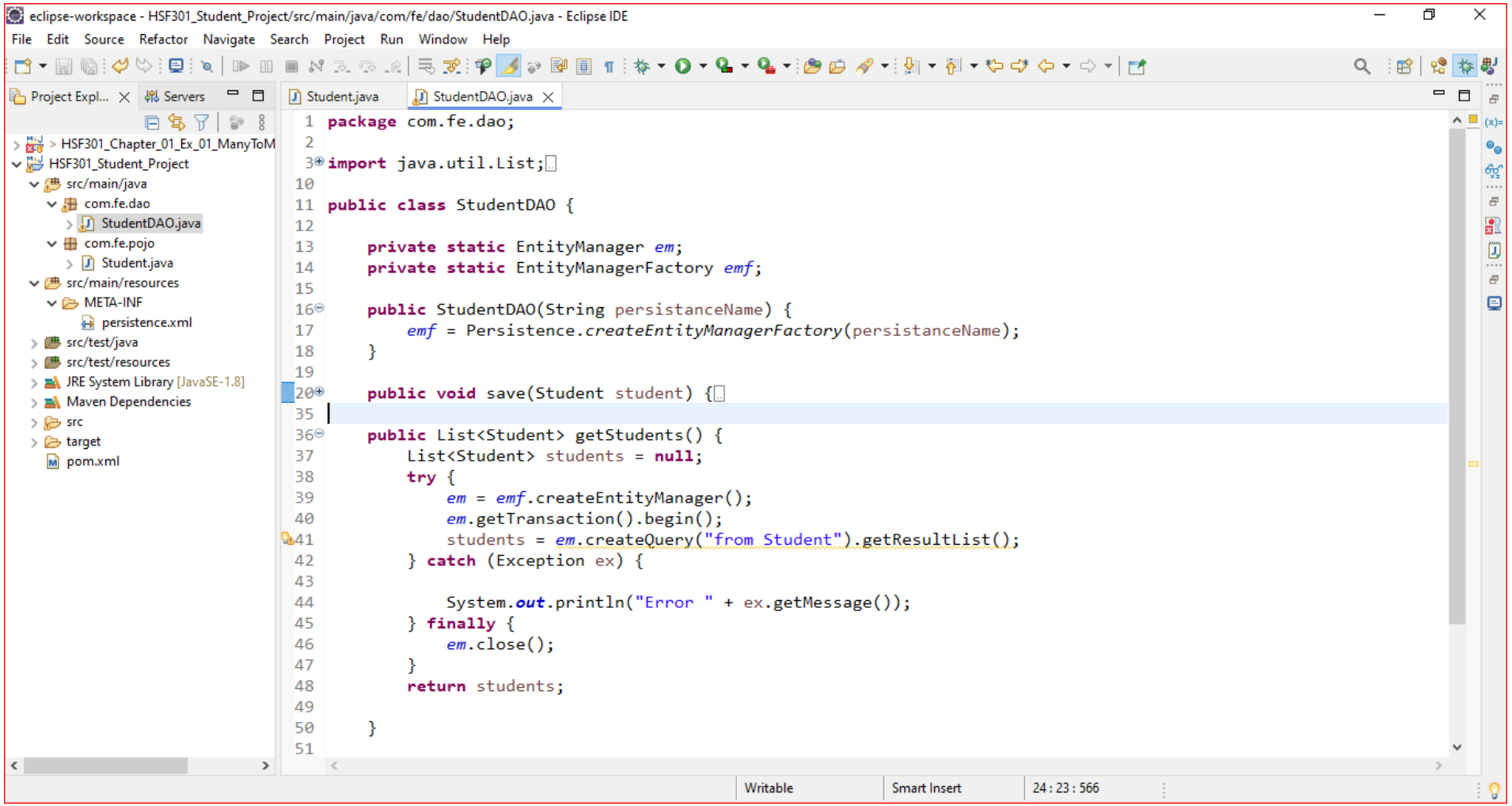
15. Edit the StudentDAO in com.fe.dao



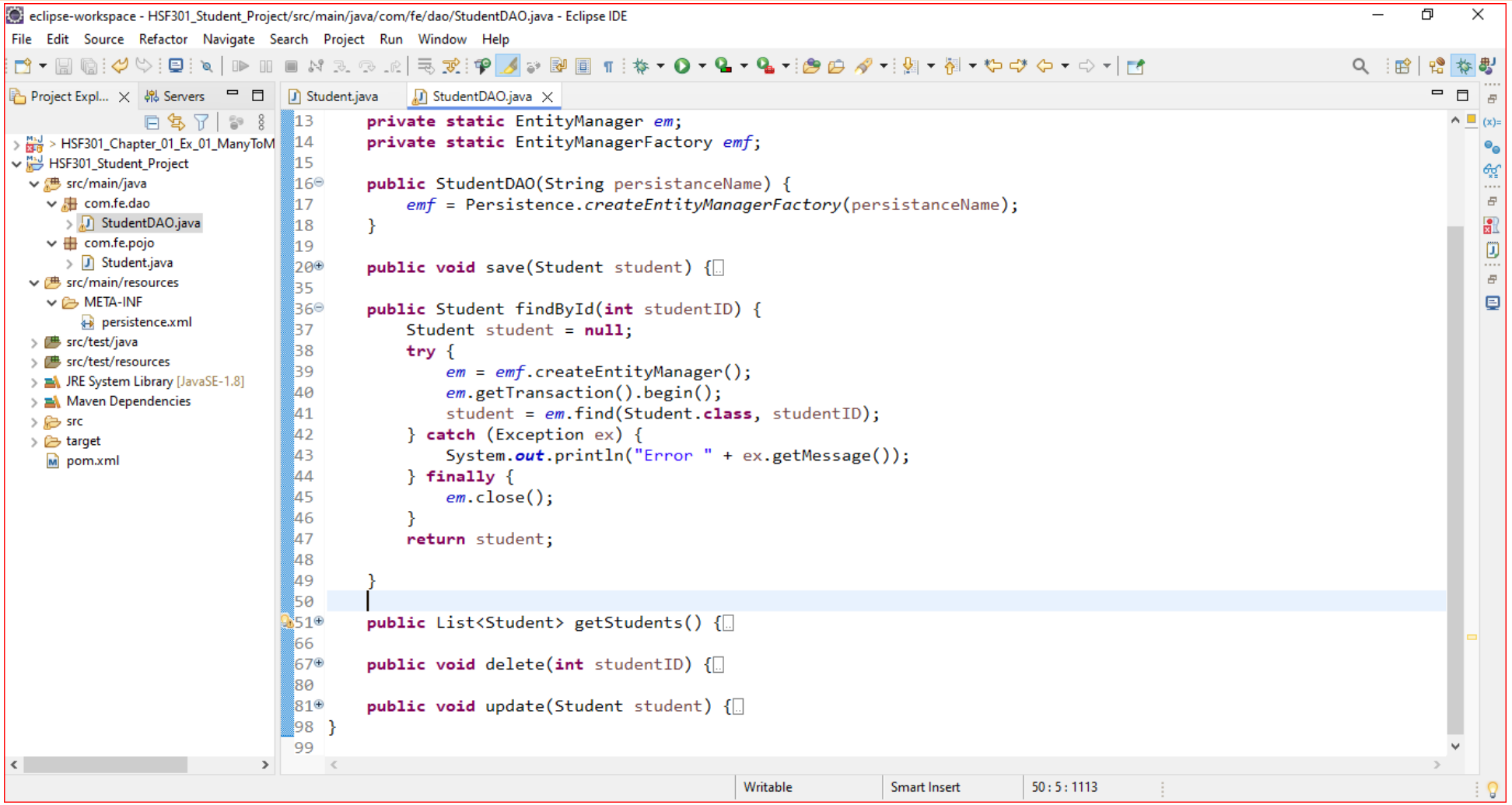
16. Save Student Method



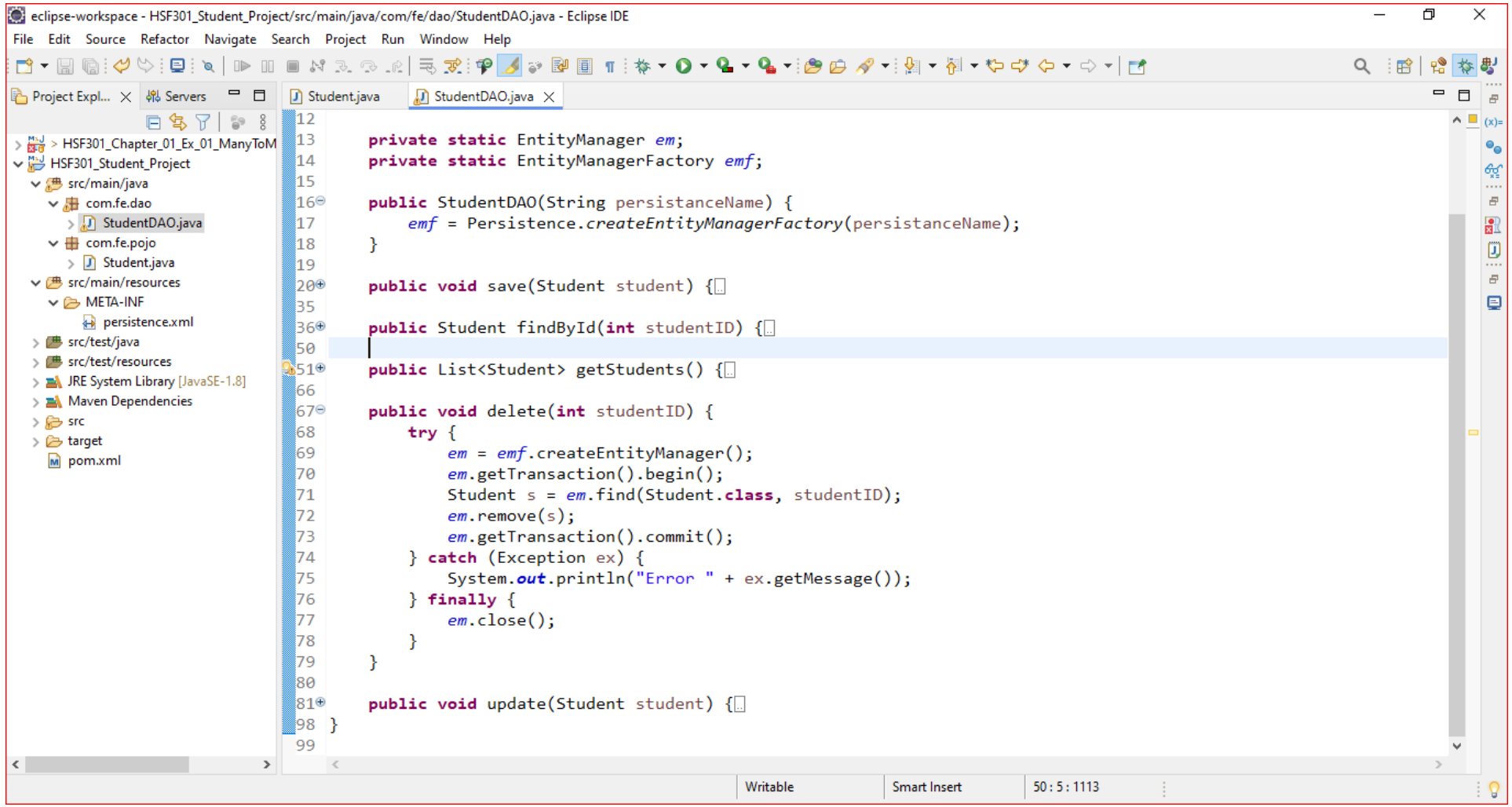
17. Get All Students Method



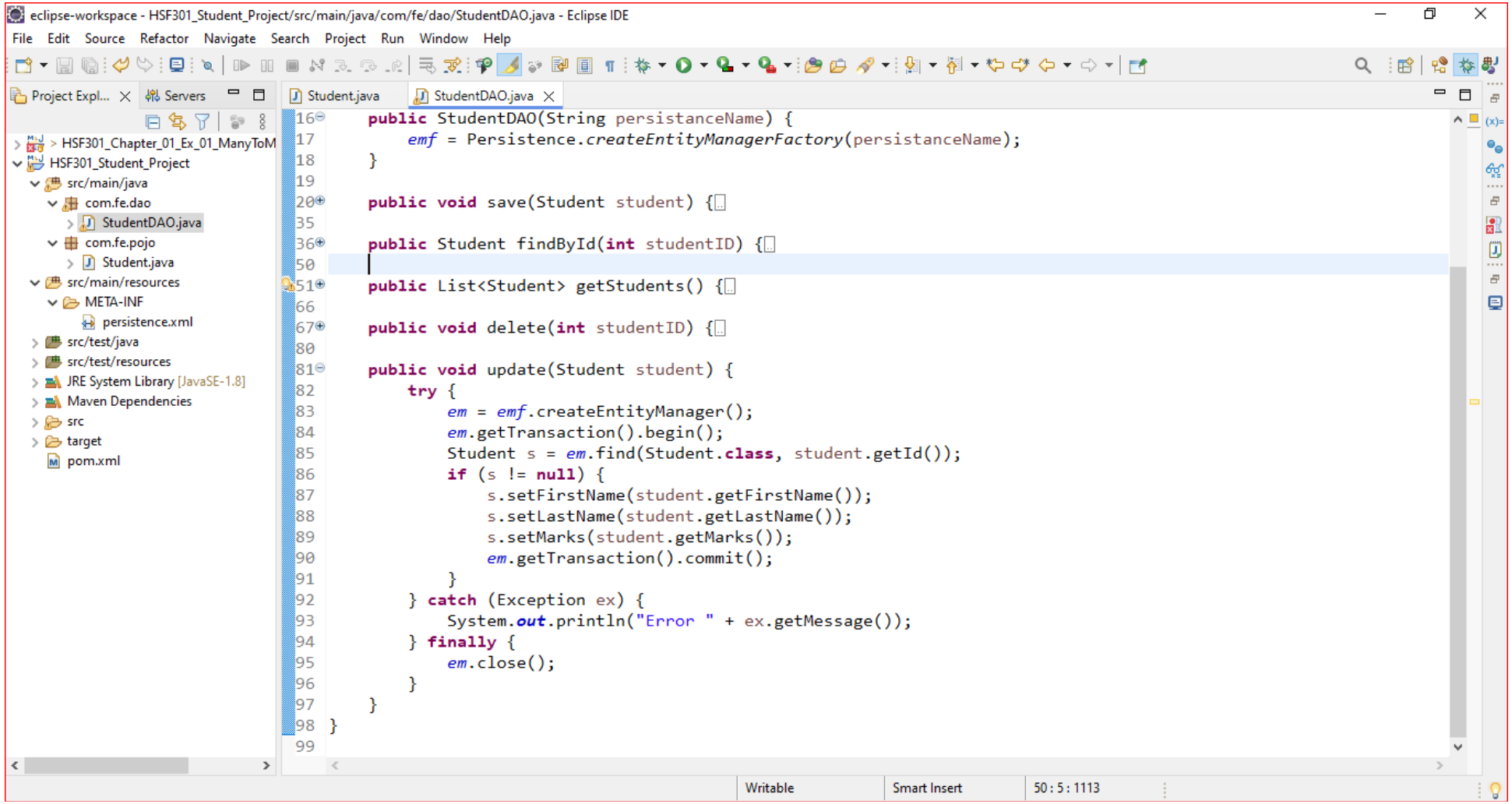
18. Find a Student Method



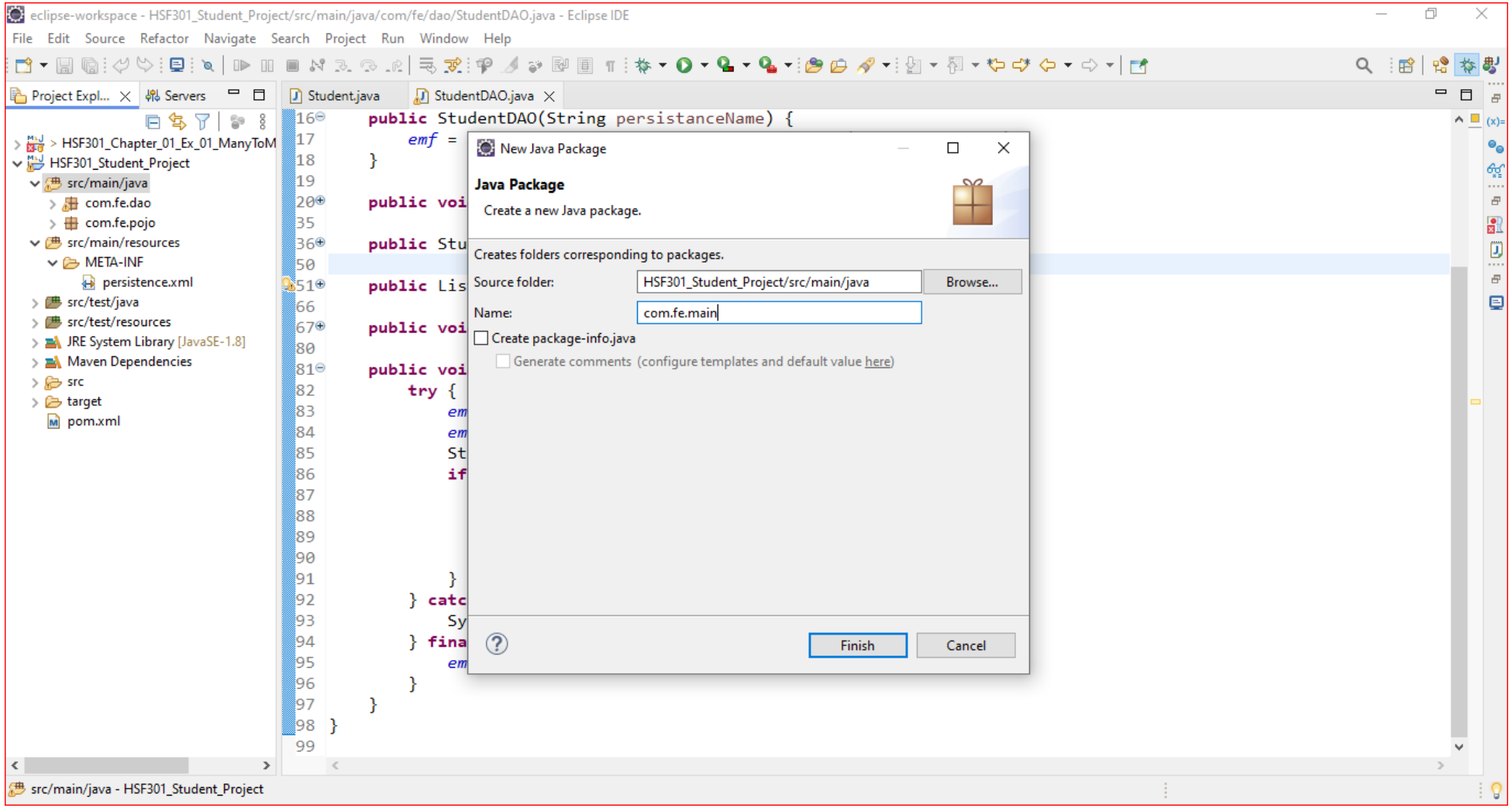
19. Delete Student Method



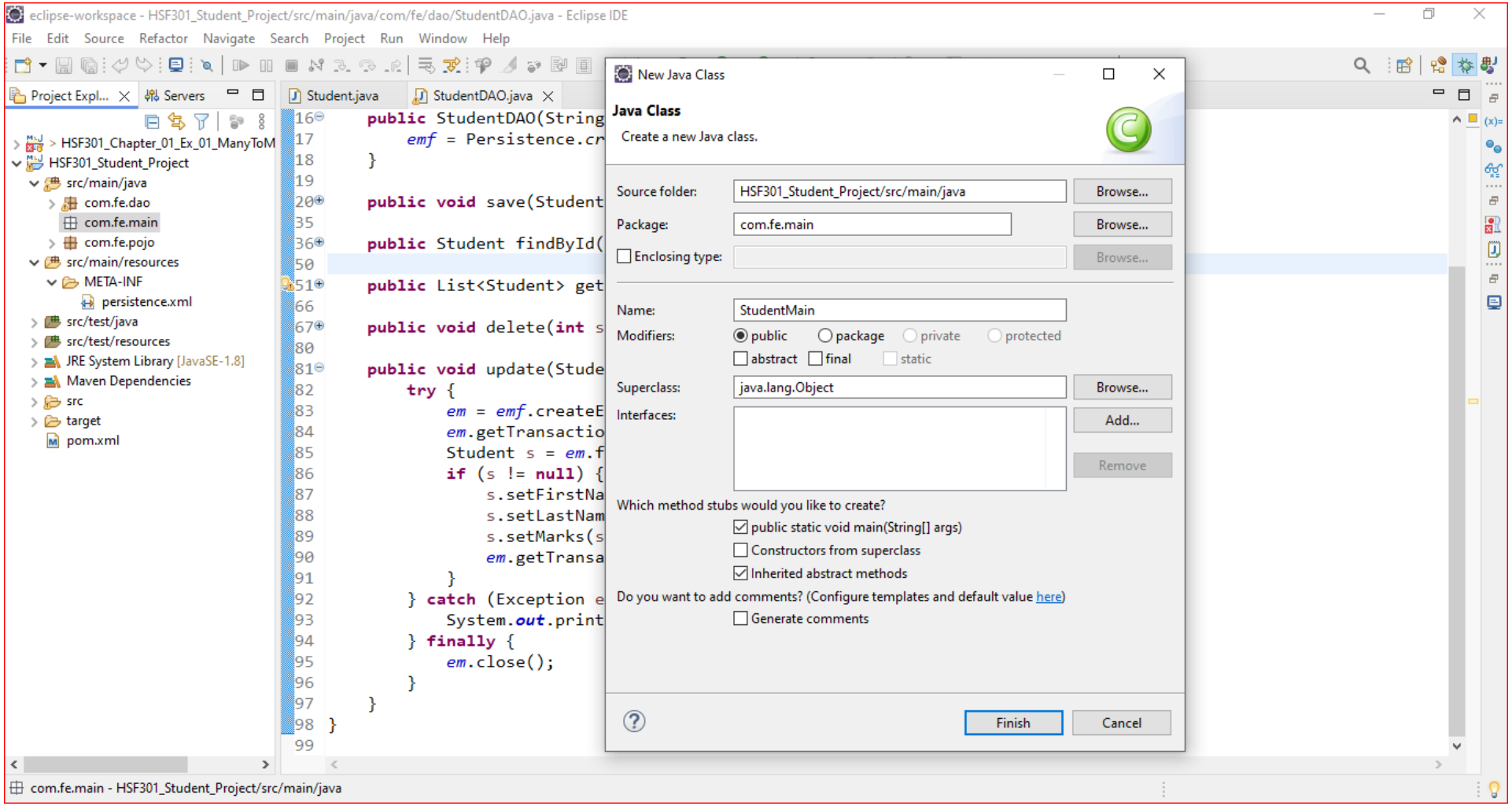
20. Update Student Method



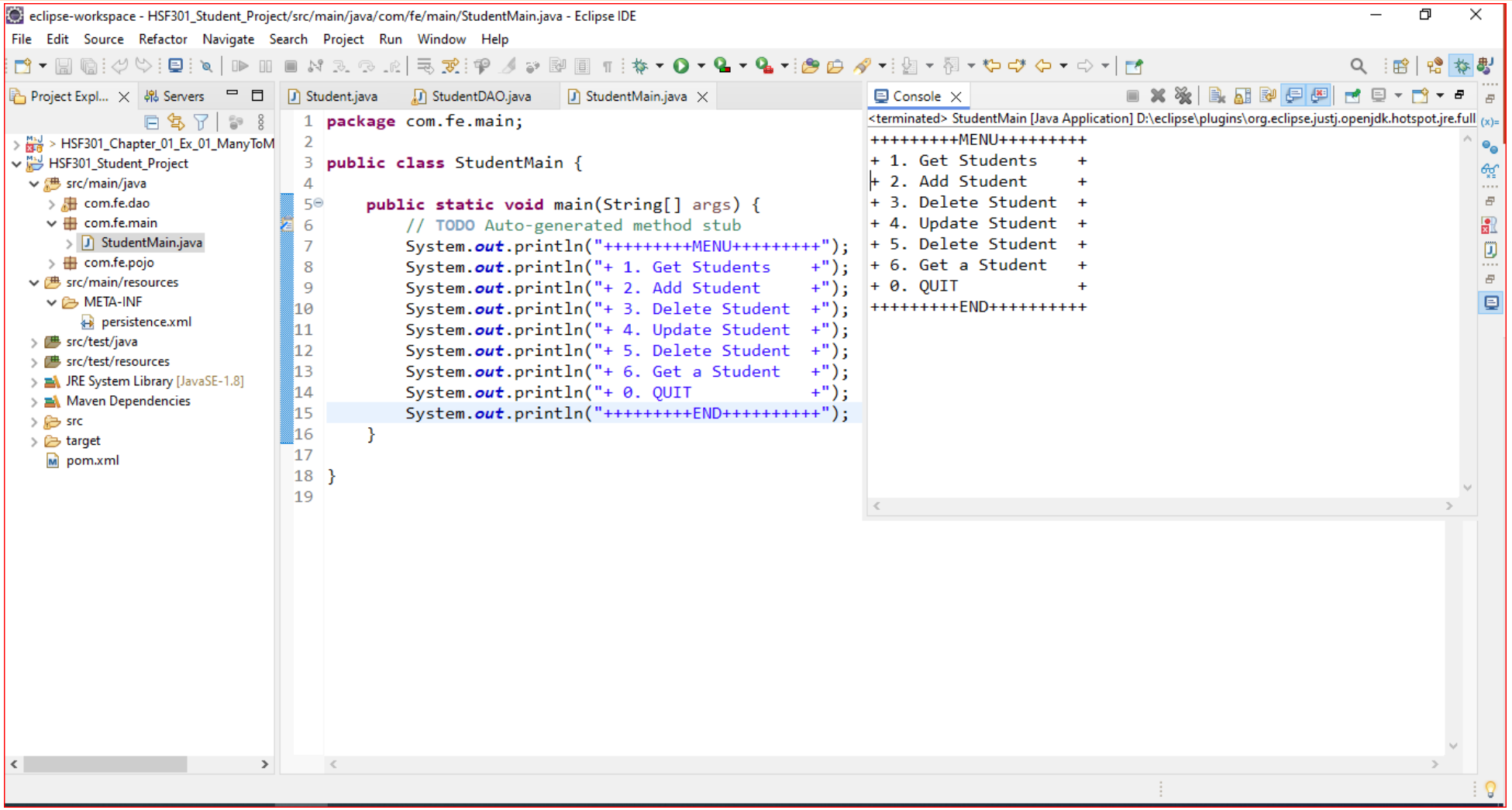
21. Add com.fe.main Package in src/main/java



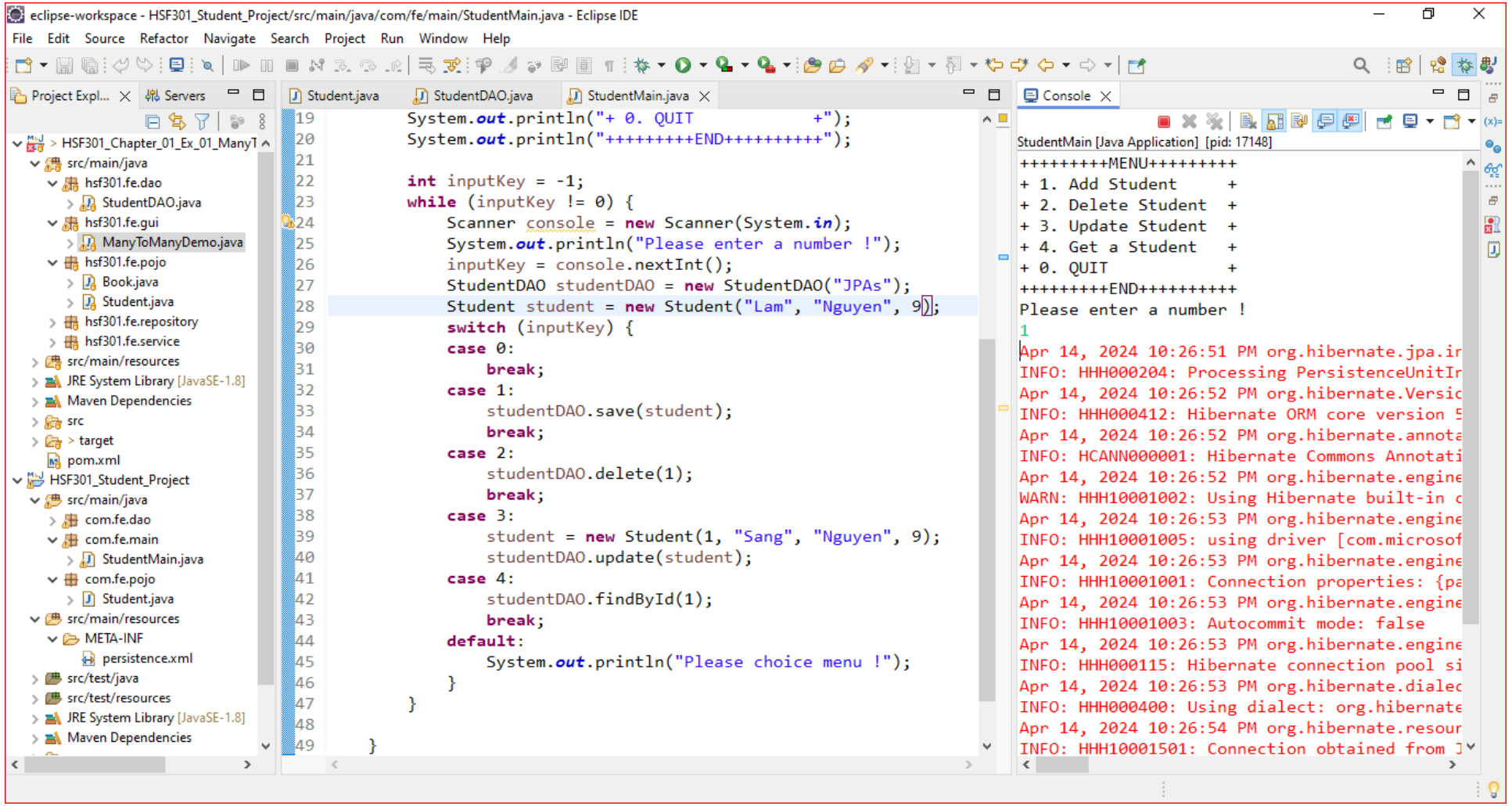
22. Create StudentMain class in com.fe.main Package



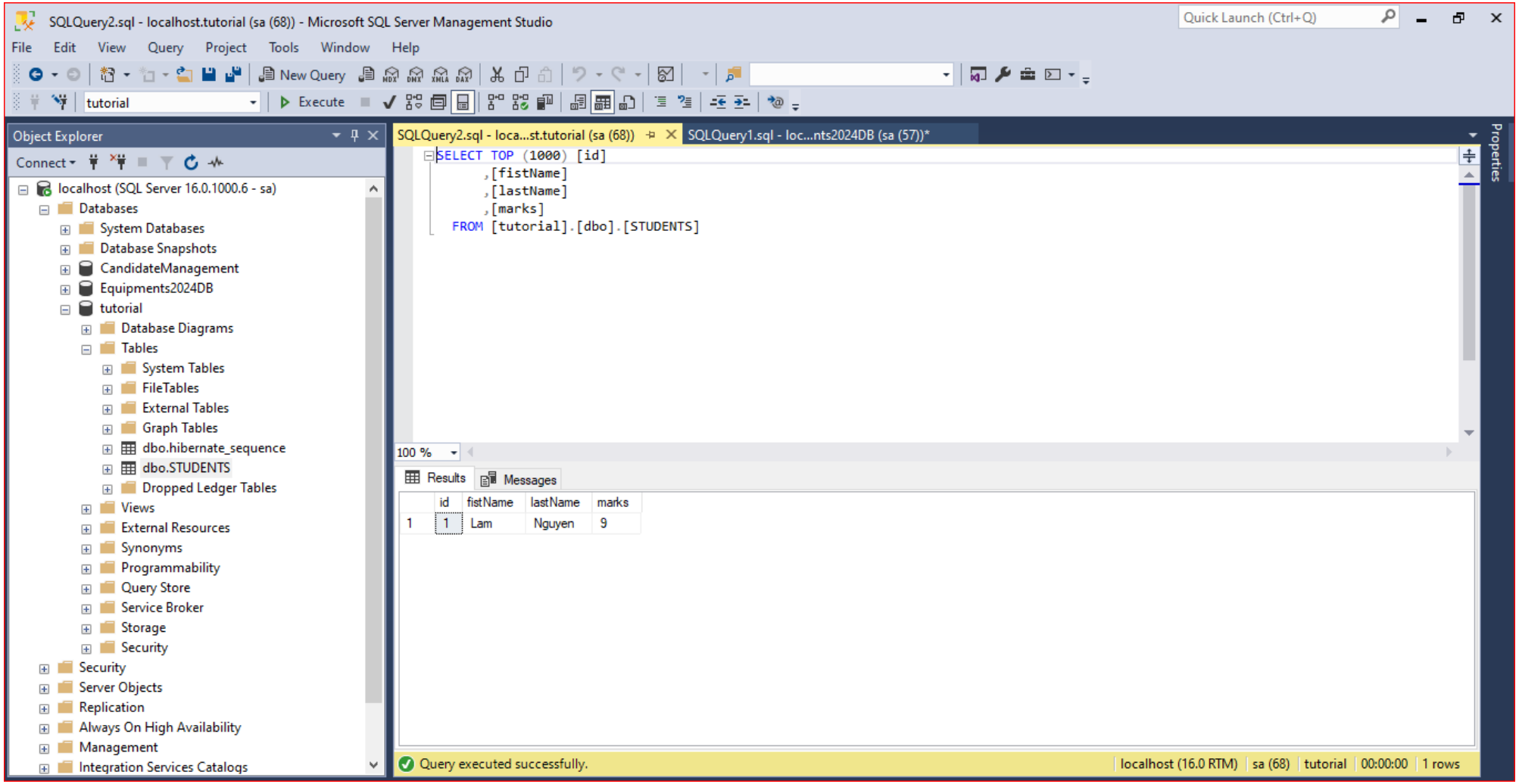
23. Edit the StudentMain in com.fe.main



24. Edit the StudentMain in com.fe.main



25. Result


SQLQuery2.sql - localhost.tutorial (sa (68)) - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
Execute
tutorial
Object Explorer
Connect
localhost (SQL Server 16.0.1000.6 - sa)
Databases
System Databases
Database Snapshots
CandidateManagement
Equipments2024DB
tutorial
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.hibernate_sequence
dbo.STUDENTS
Dropped Ledger Tables
Views
External Resources
Synonyms
Programmability
Query Store
Service Broker
Storage
Security
Security
Server Objects
Replication
Always On High Availability
Management
Integration Services Catalogs
SQLQuery2.sql - loca...st.tutorial (sa (68))
SQLQuery1.sql - loc...nts2024DB (sa (57))*
SELECT TOP (1000) [id]
,[fistName]
,[lastName]
,[marks]
FROM [tutorial].[dbo].[STUDENTS]
Properties
100 %
Results
Messages
id fistName lastName marks
1 1 Lam Nguyen 9
Query executed successfully.
localhost (16.0 RTM) sa (68) tutorial 00:00:00 1 rows

FPT Education
FPT UNIVERSITY

Java

Mapping in JPA

Relationships Annotations in JPA

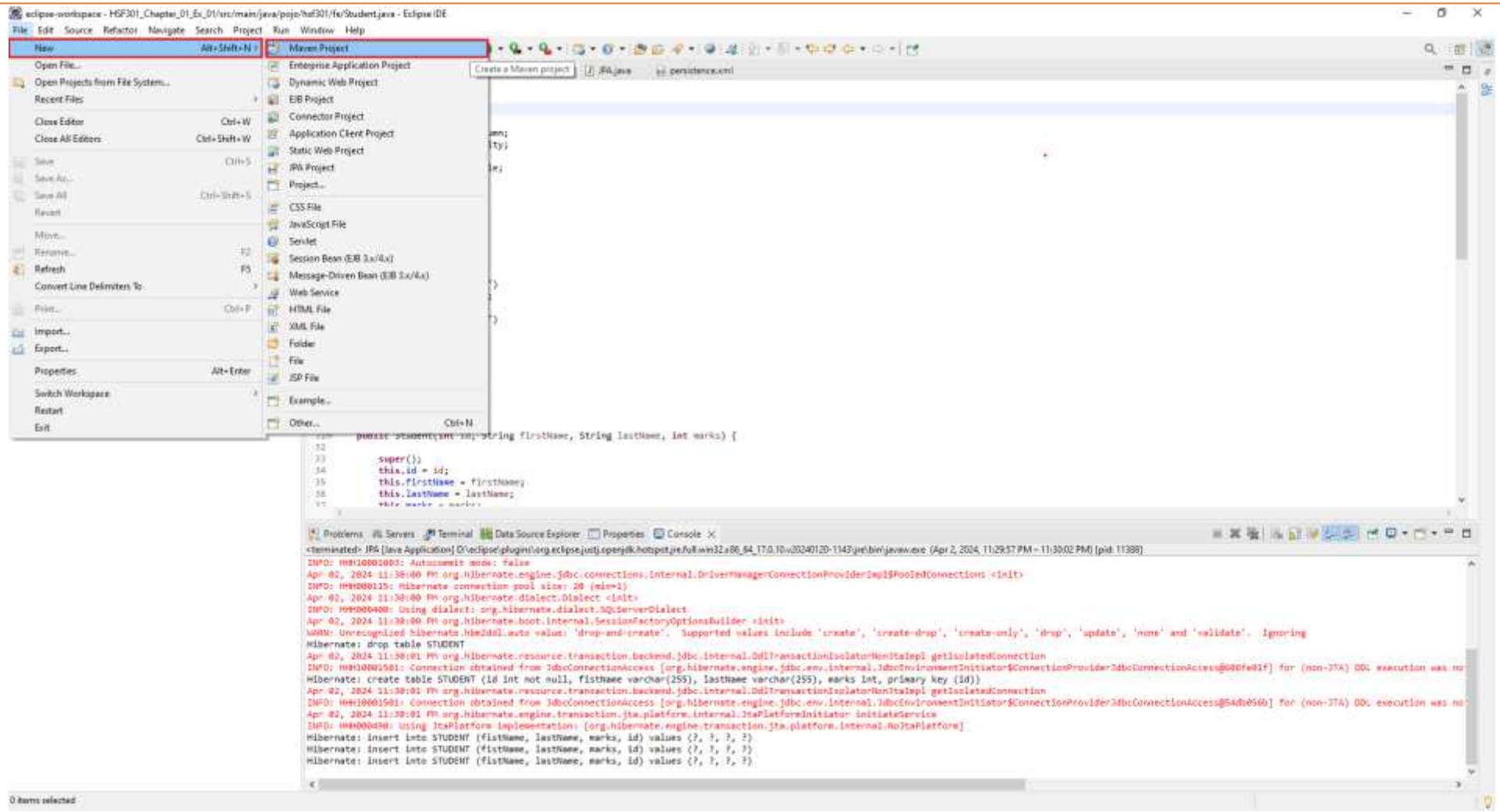
- **@ManyToOne:** This annotation defines a many-to-one relationship between two entities.
- **@OneToMany:** This annotation defines a one-to-many relationship between two entities.
- **@OneToOne:** This annotation defines a one-to-one relationship between two entities.
- **@ManyToMany:** This annotation defines a many-to-many relationship between two entities.

FPT Education
FPT UNIVERSITY

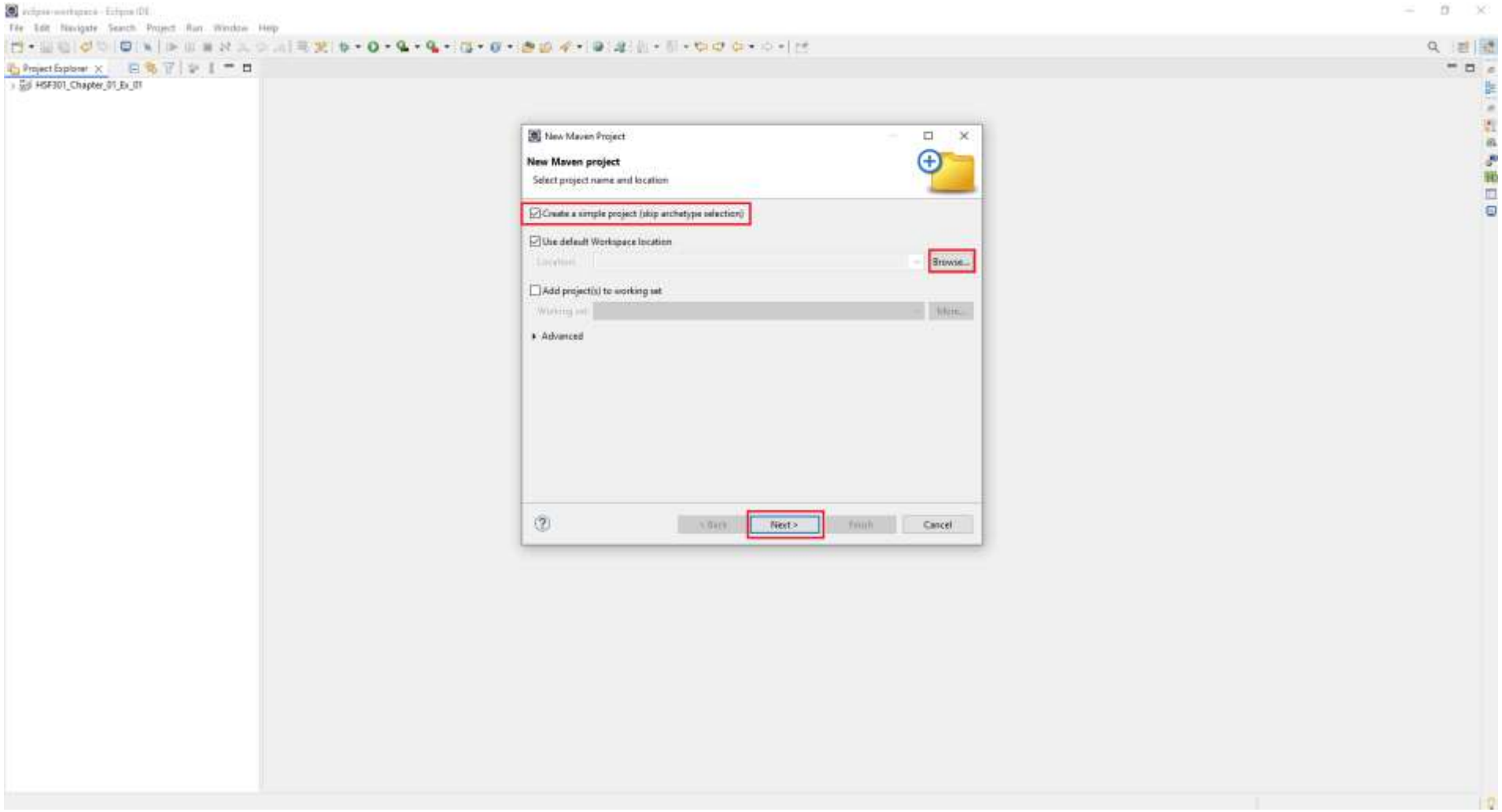
Java

Demo JPA
(One To Many)

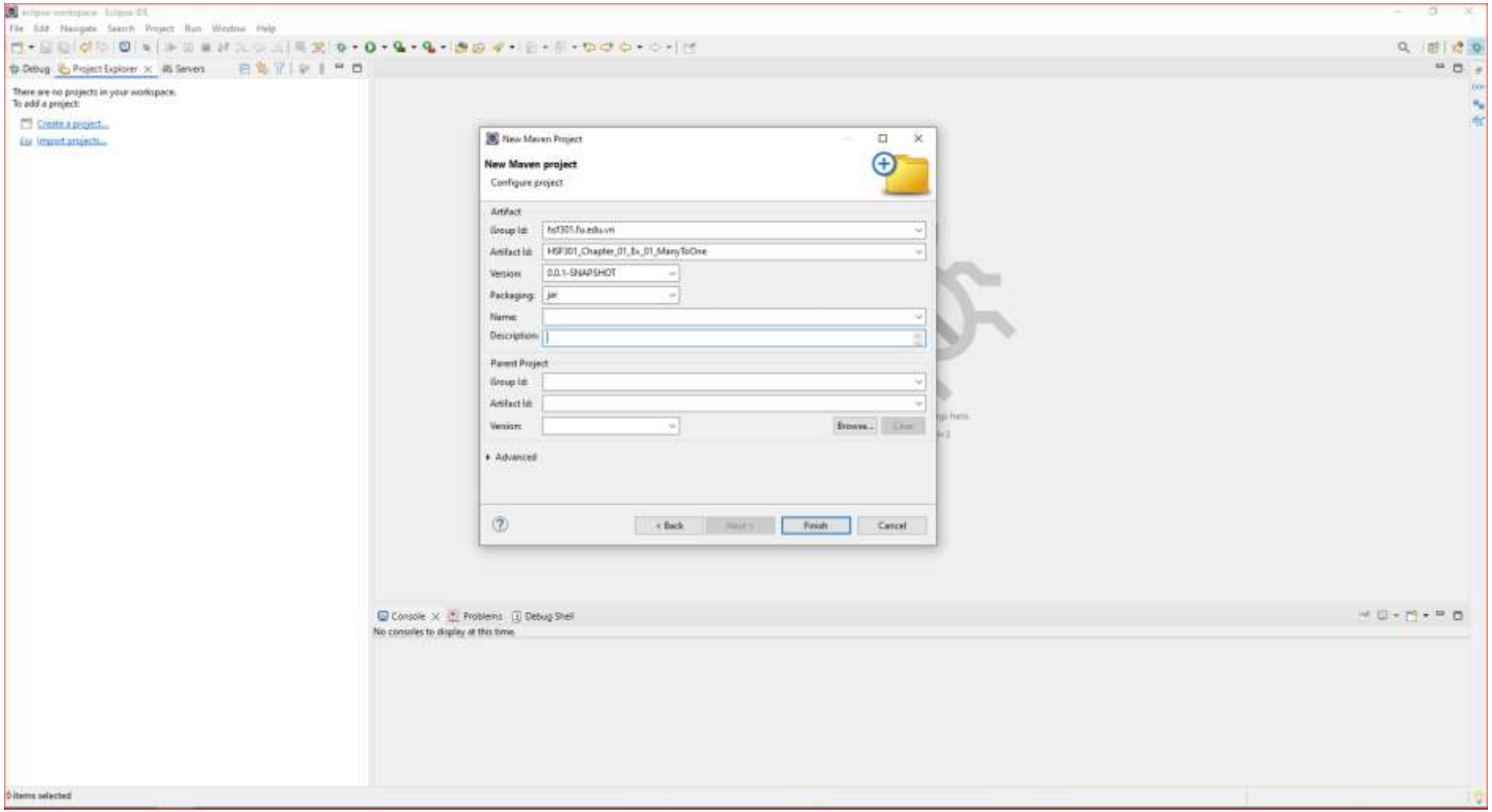
1. Open Eclipse, File | New | Maven Project



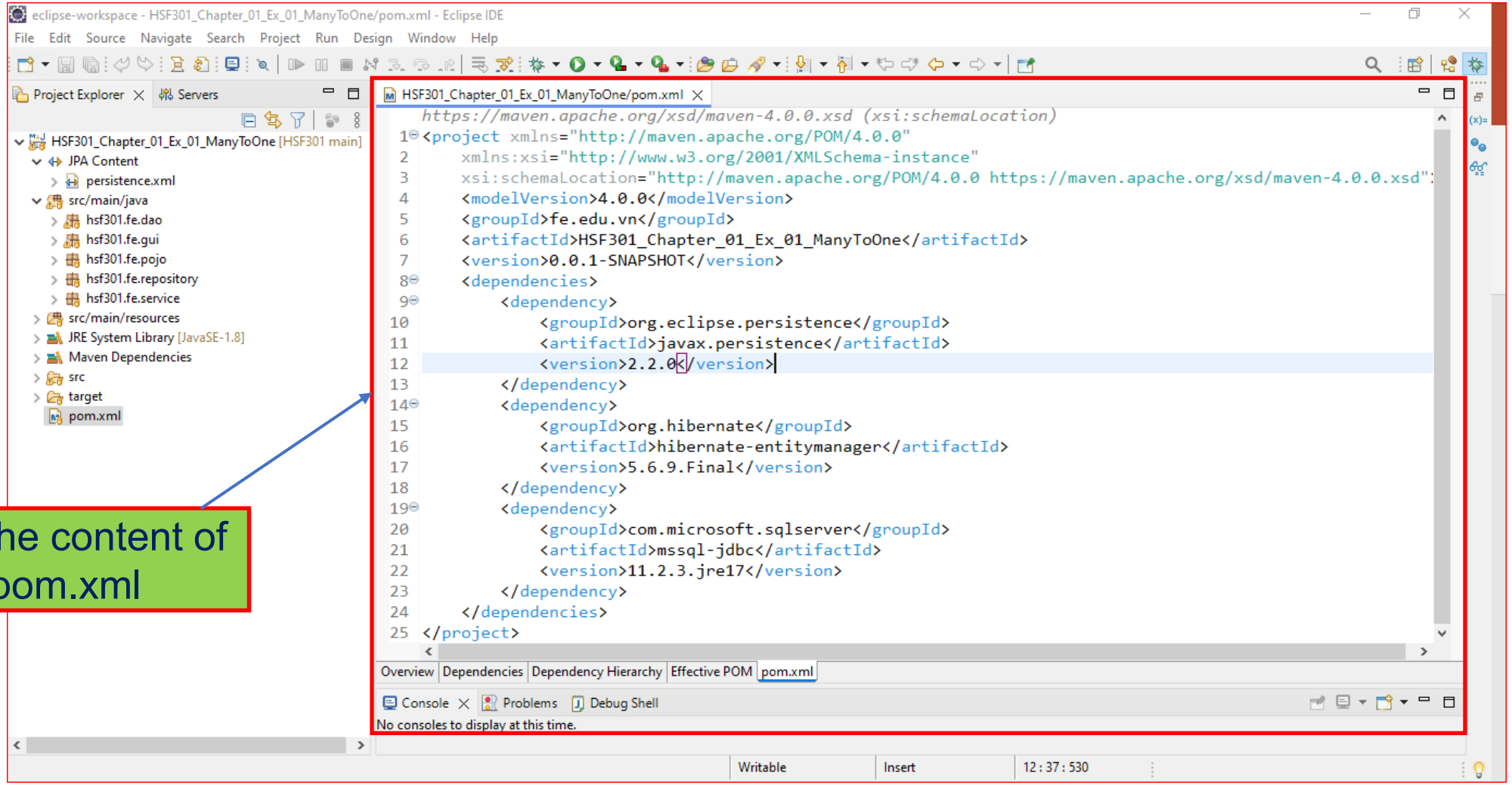
2. Check Create a simple project -> Browse Project -> Next



3. Fill the information Project -> Click Finish

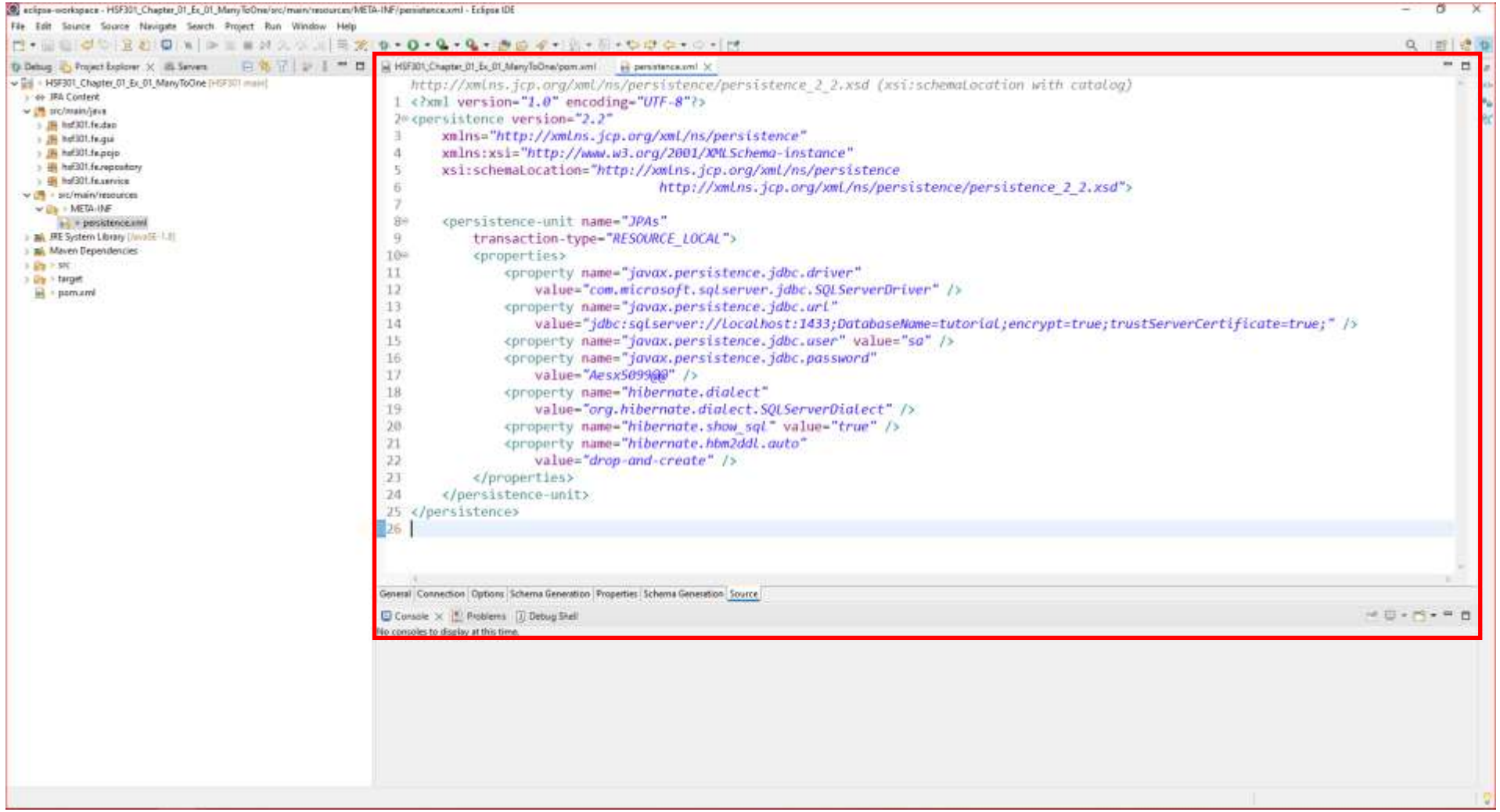


4. Structure of Maven Project

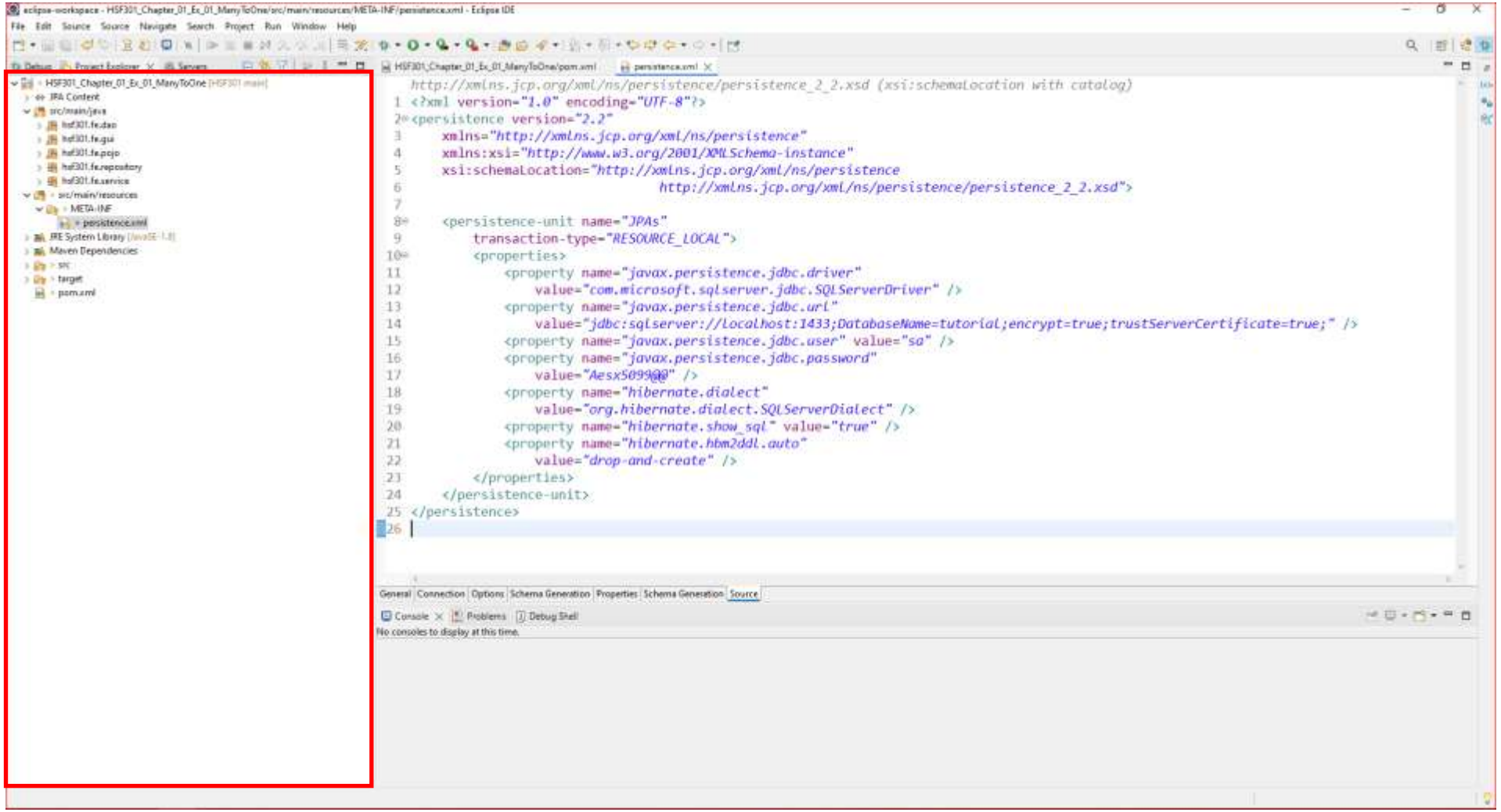


Edit the content of pom.xml

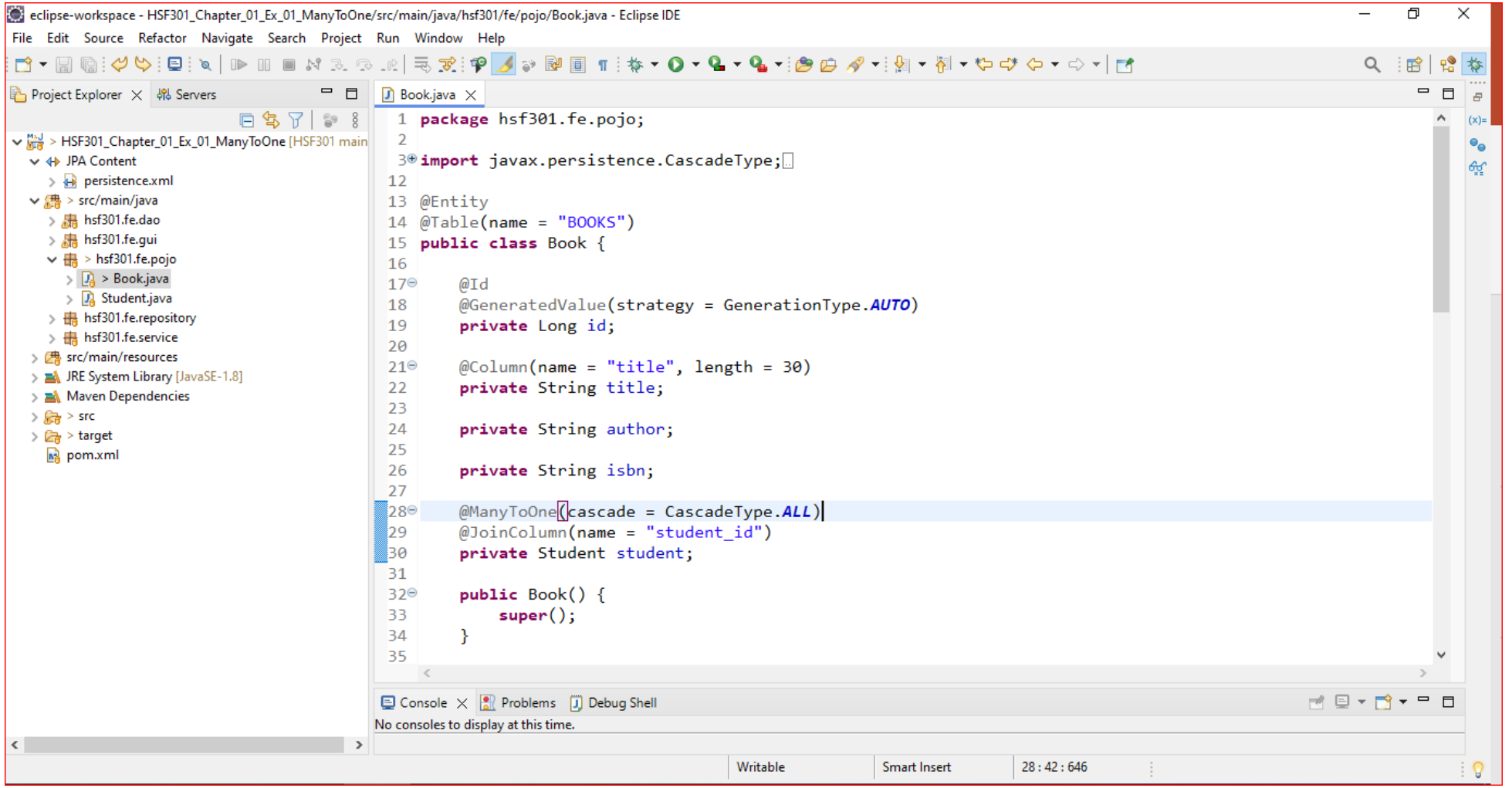
5. Create persistence.xml in META-INF folder



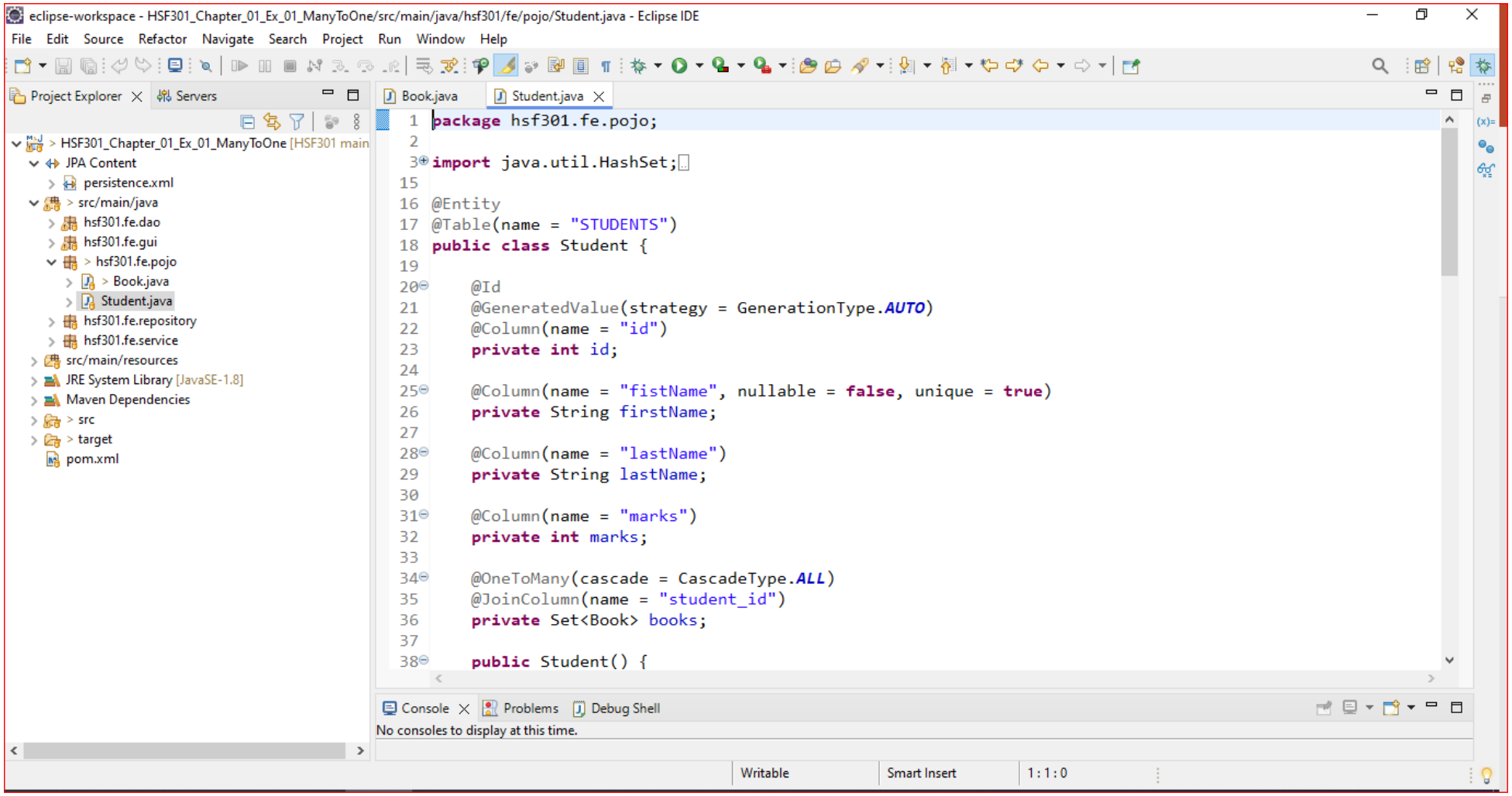
6. Create structure of project



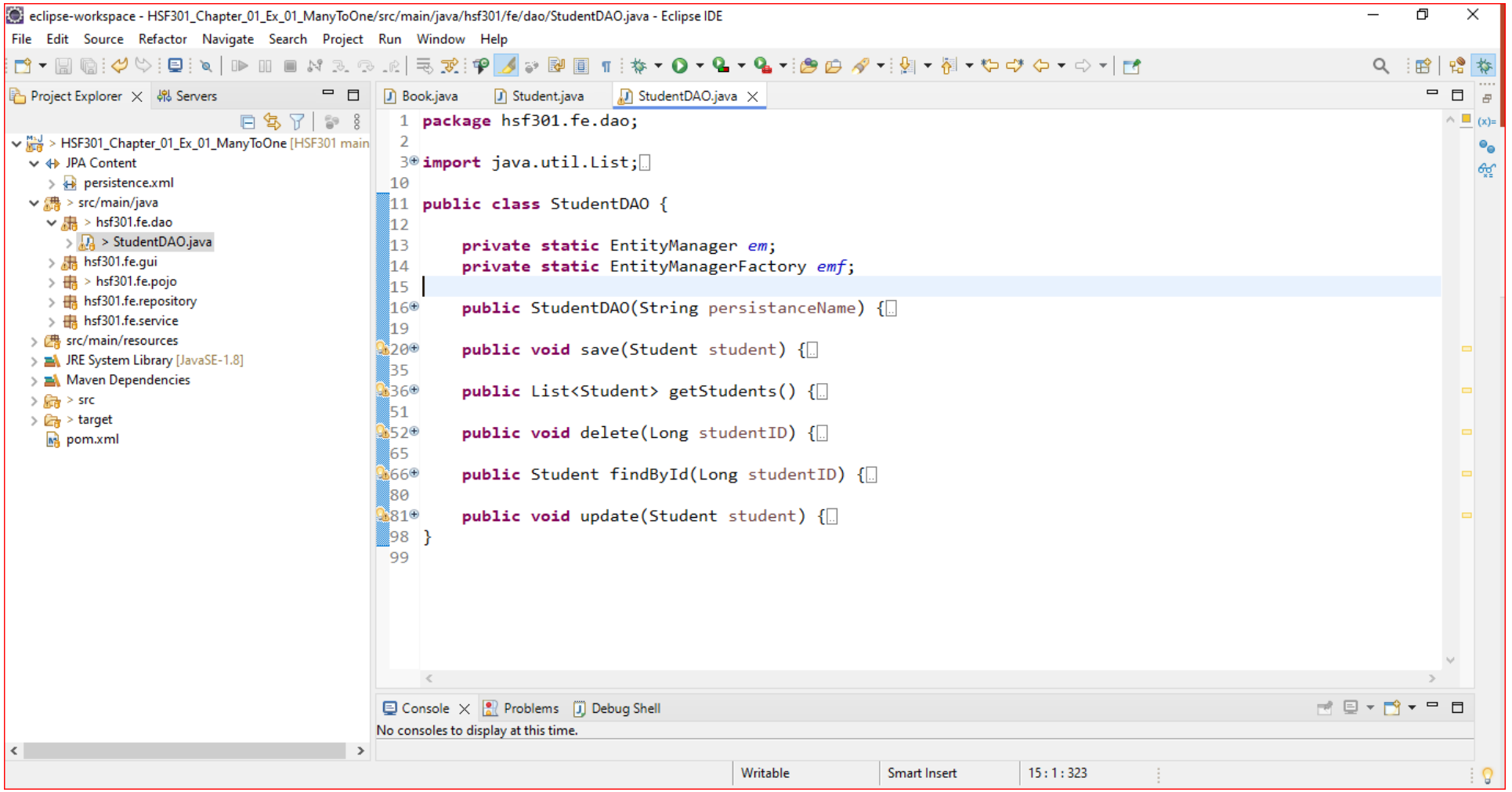
7. Create Books in Pojo



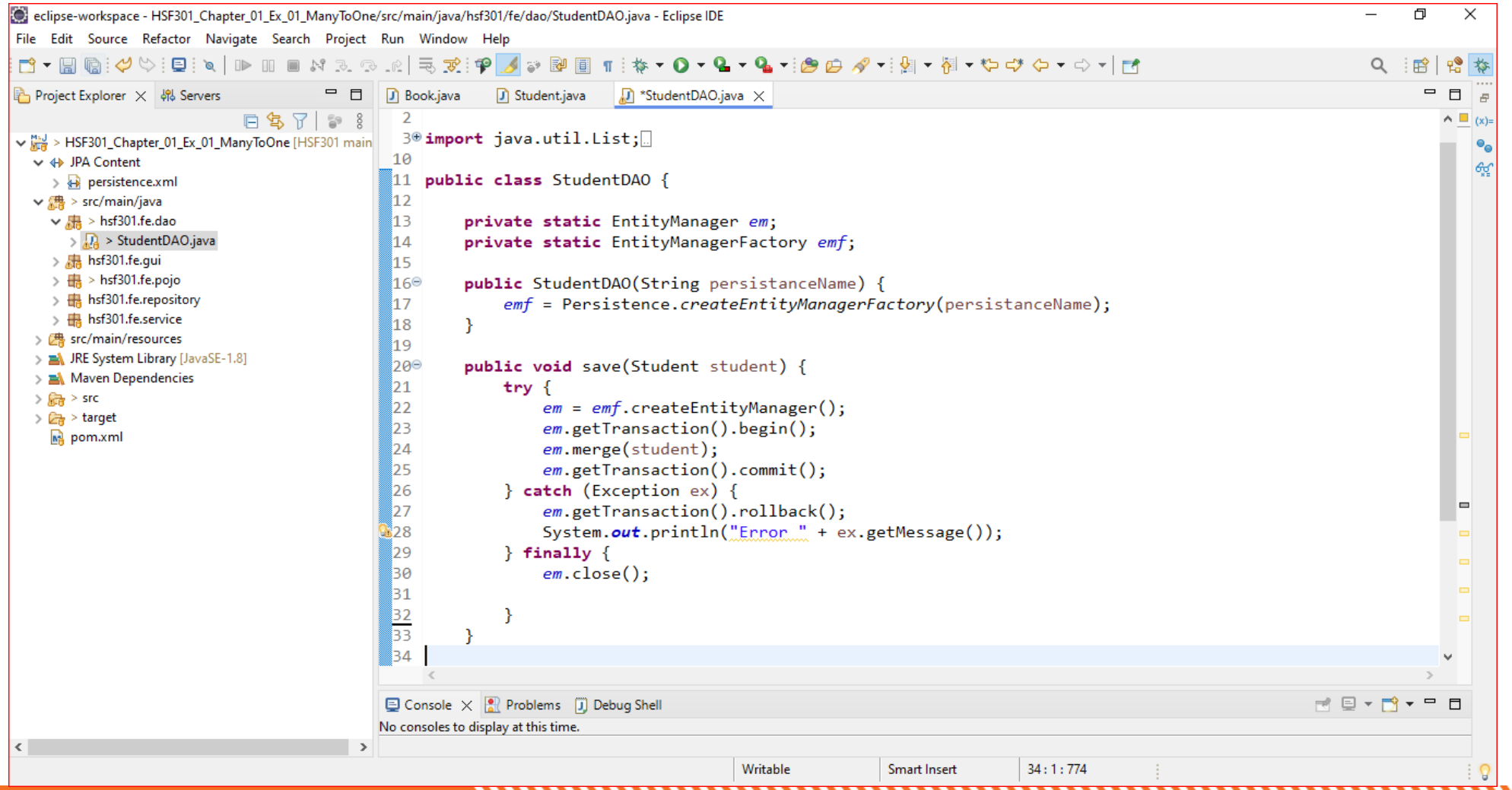
8. Create Students in Pojo



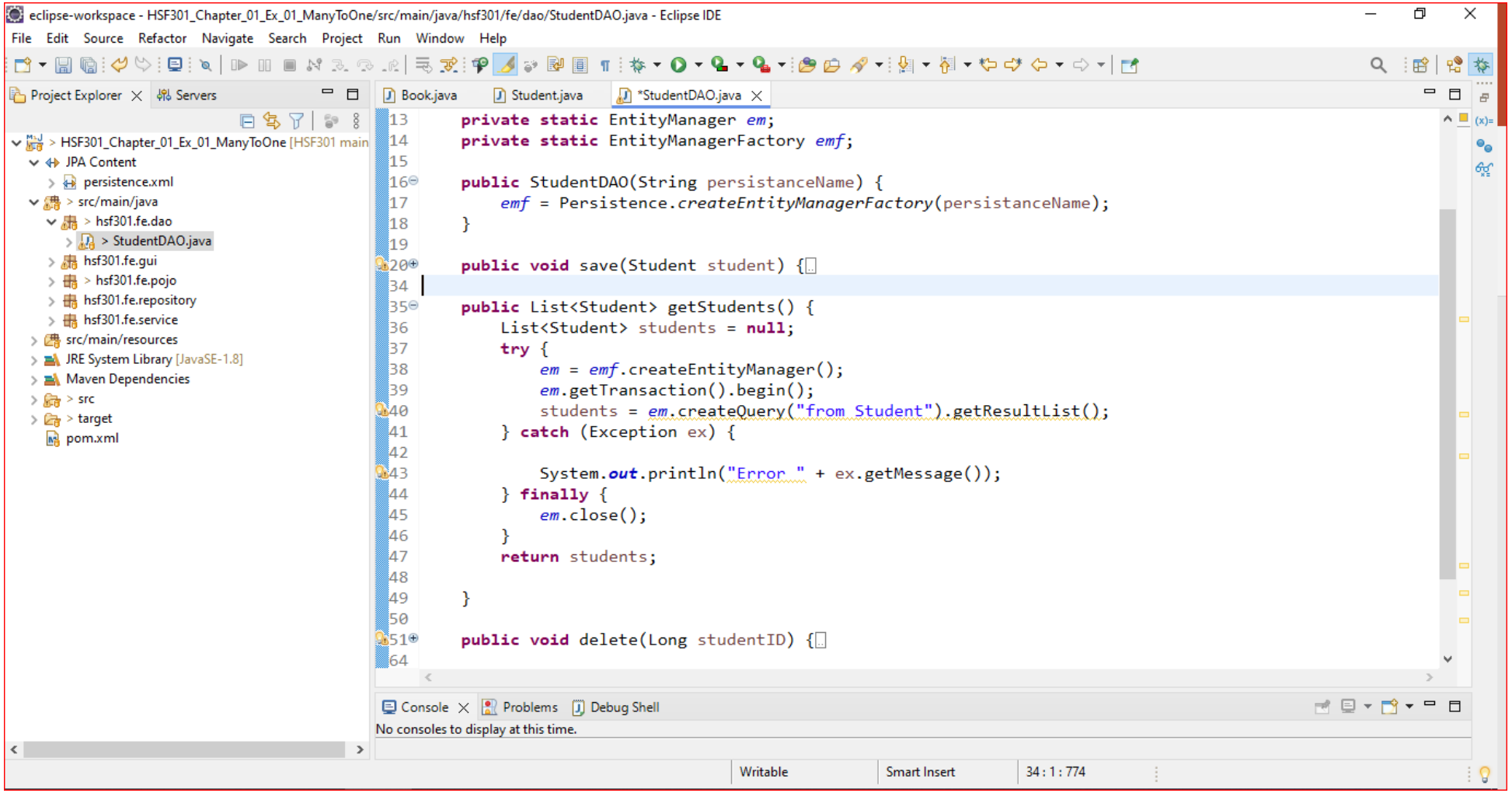
9. Create StudentDAO



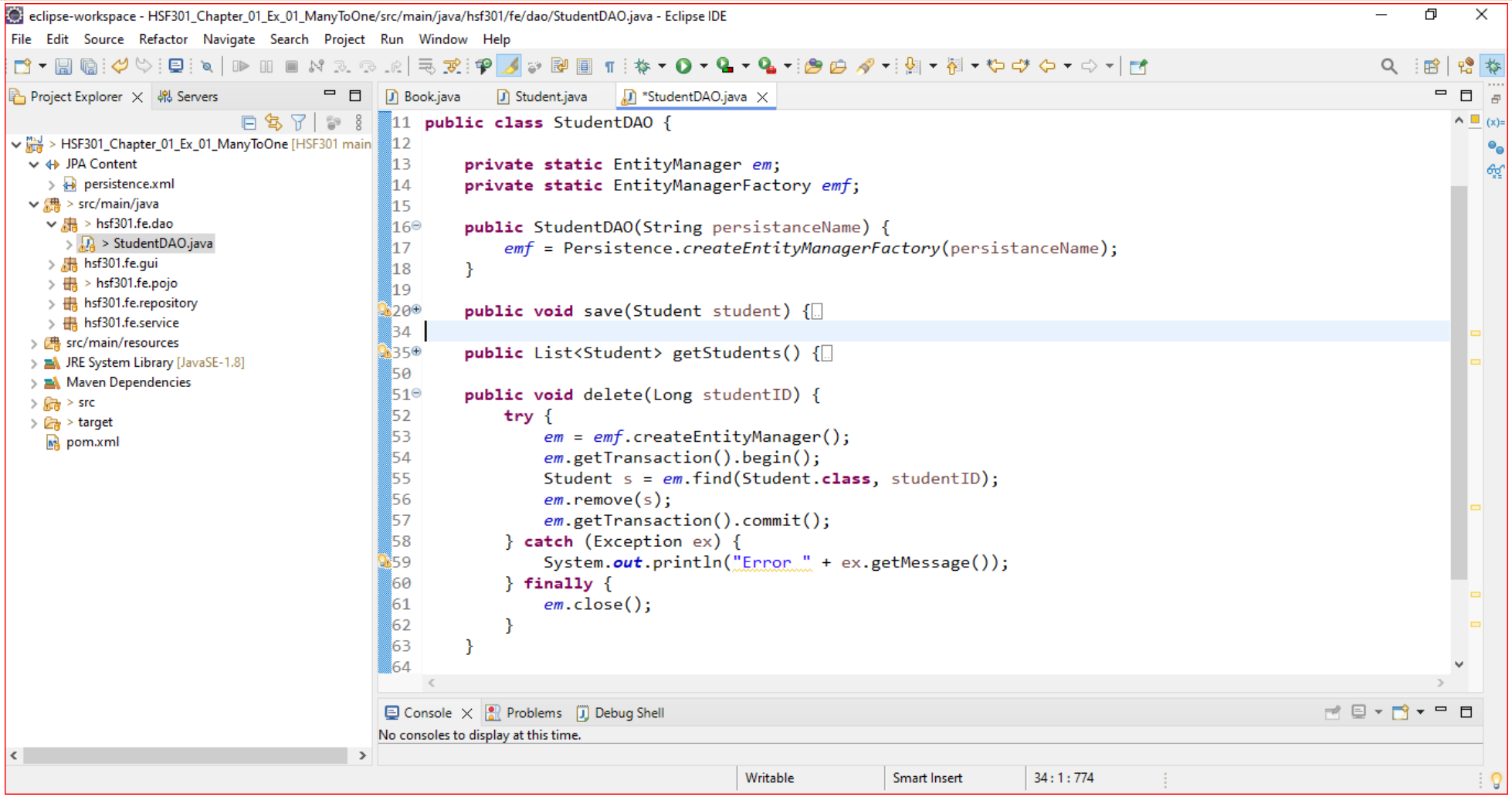
10. Save Student in StudentDAO



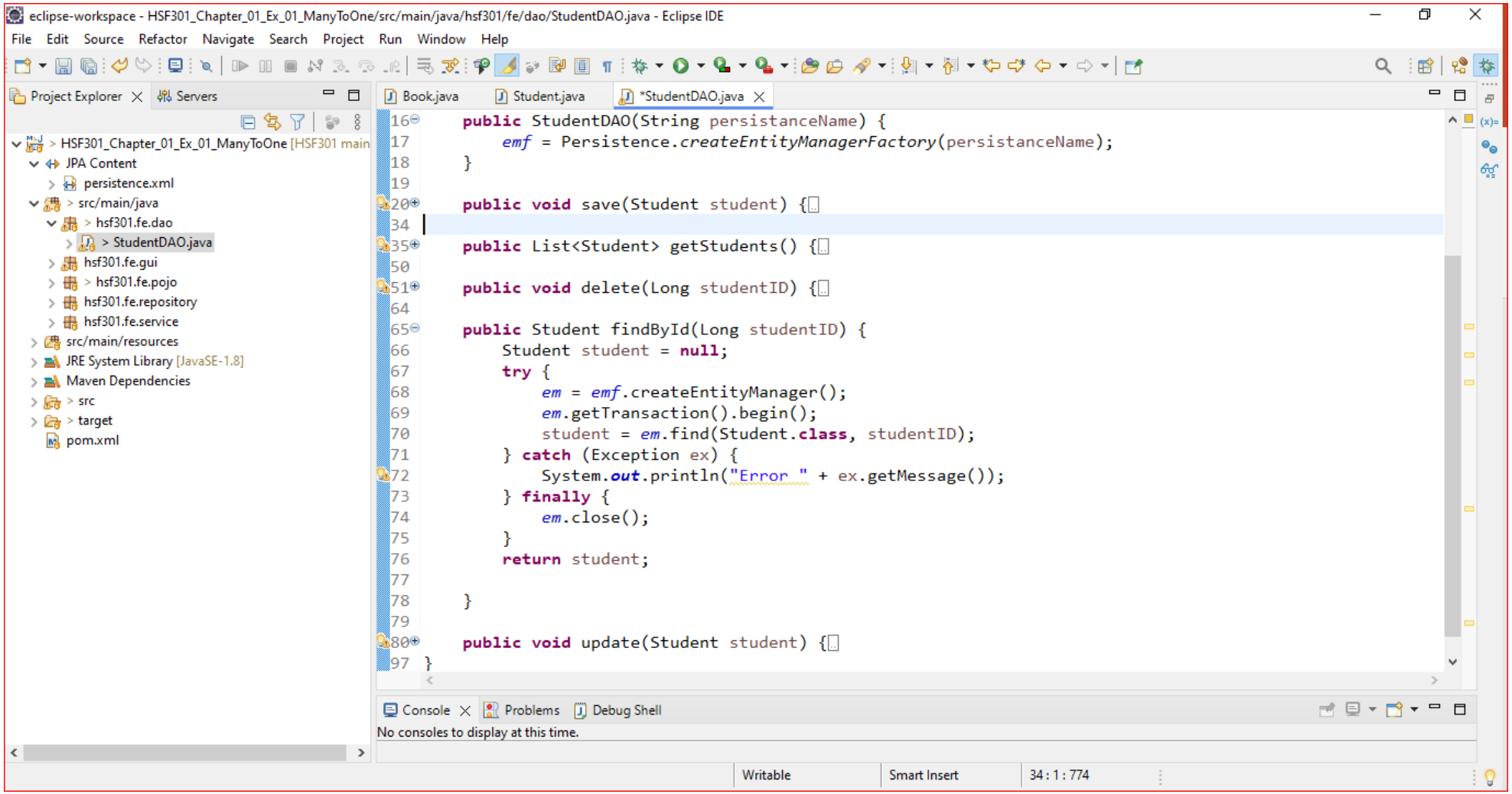
11. Get All Students in StudentDAO



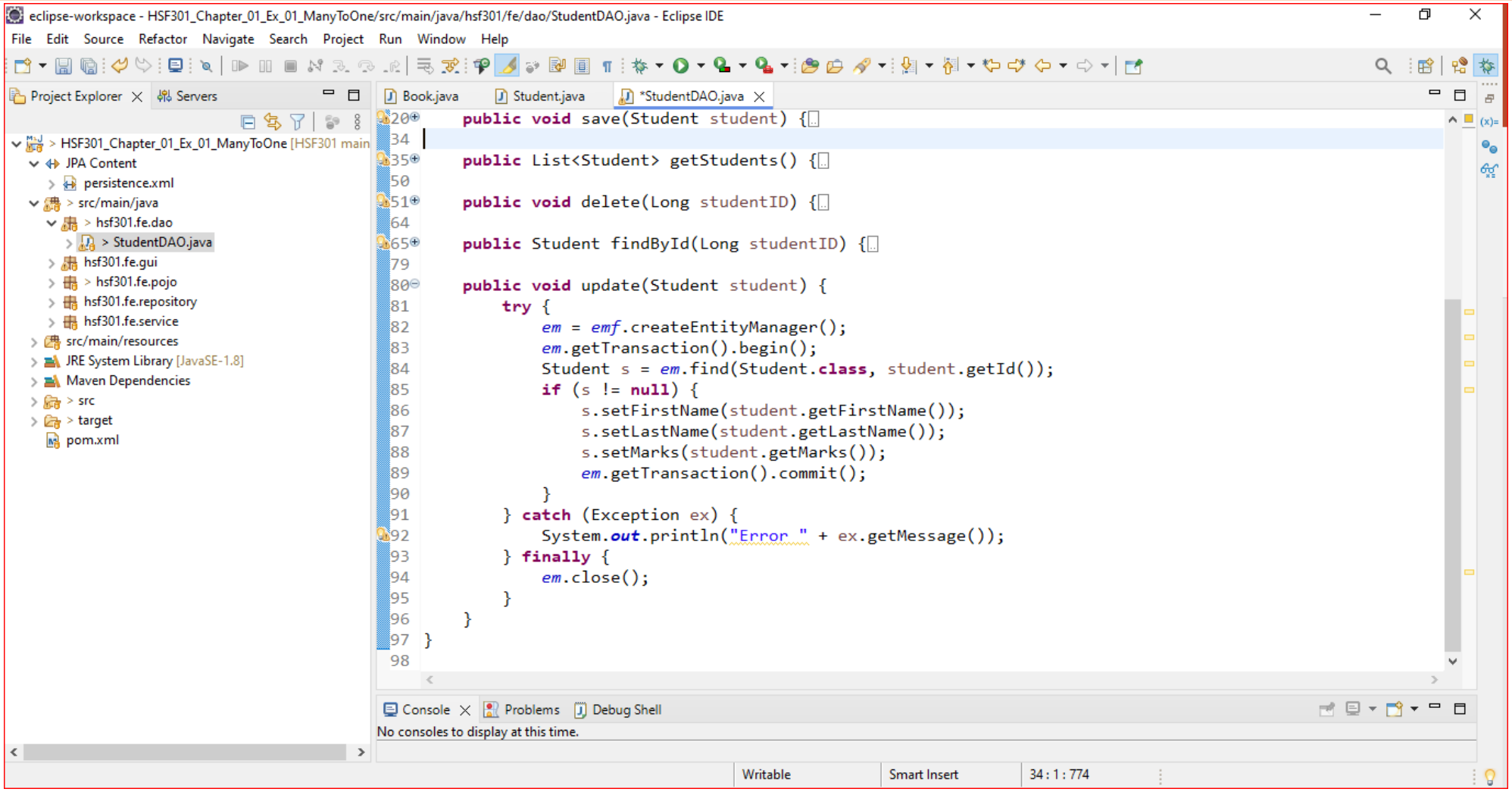
12. Delete Student in StudentDAO



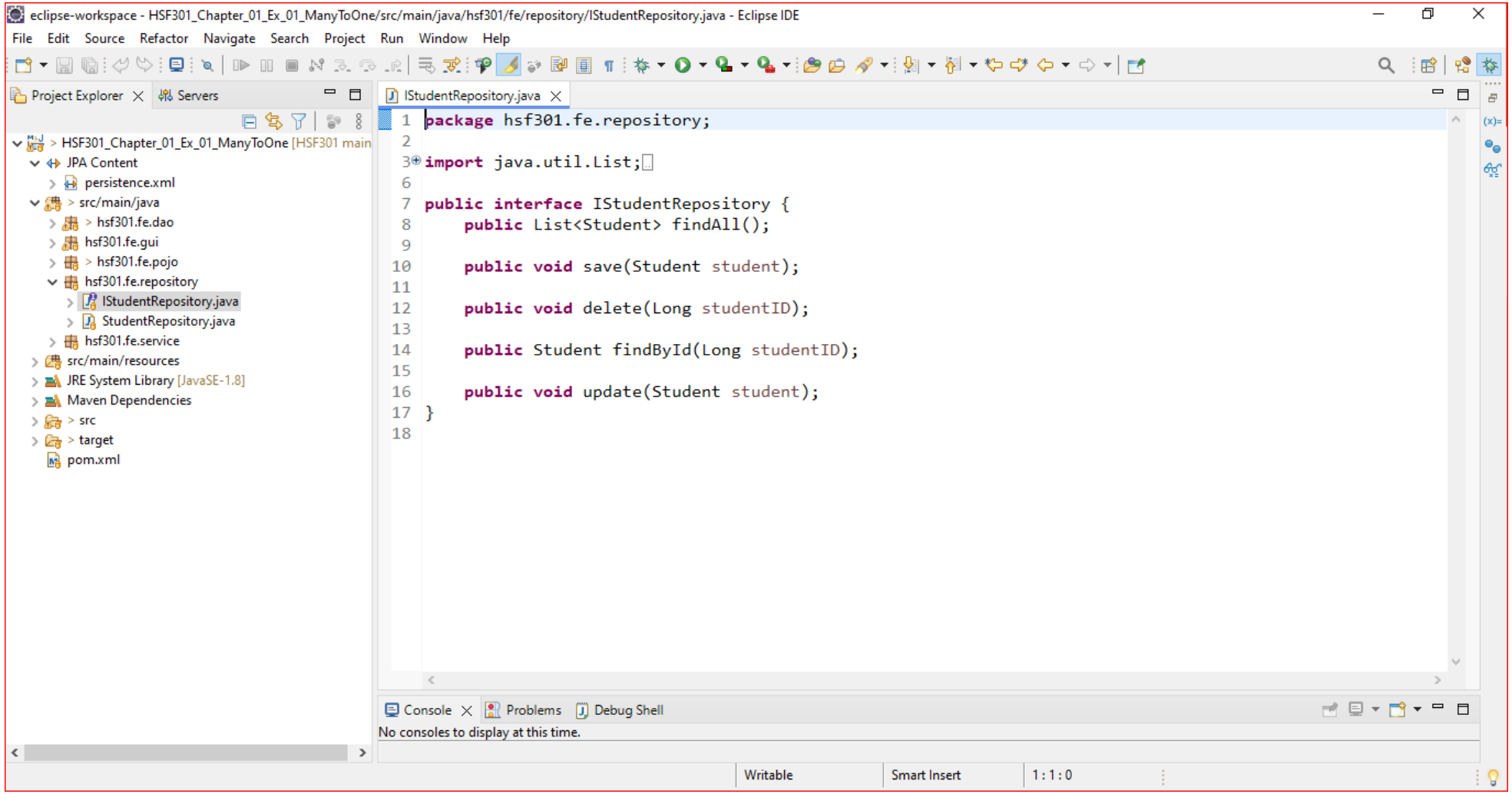
13. Get a Student in StudentDAO



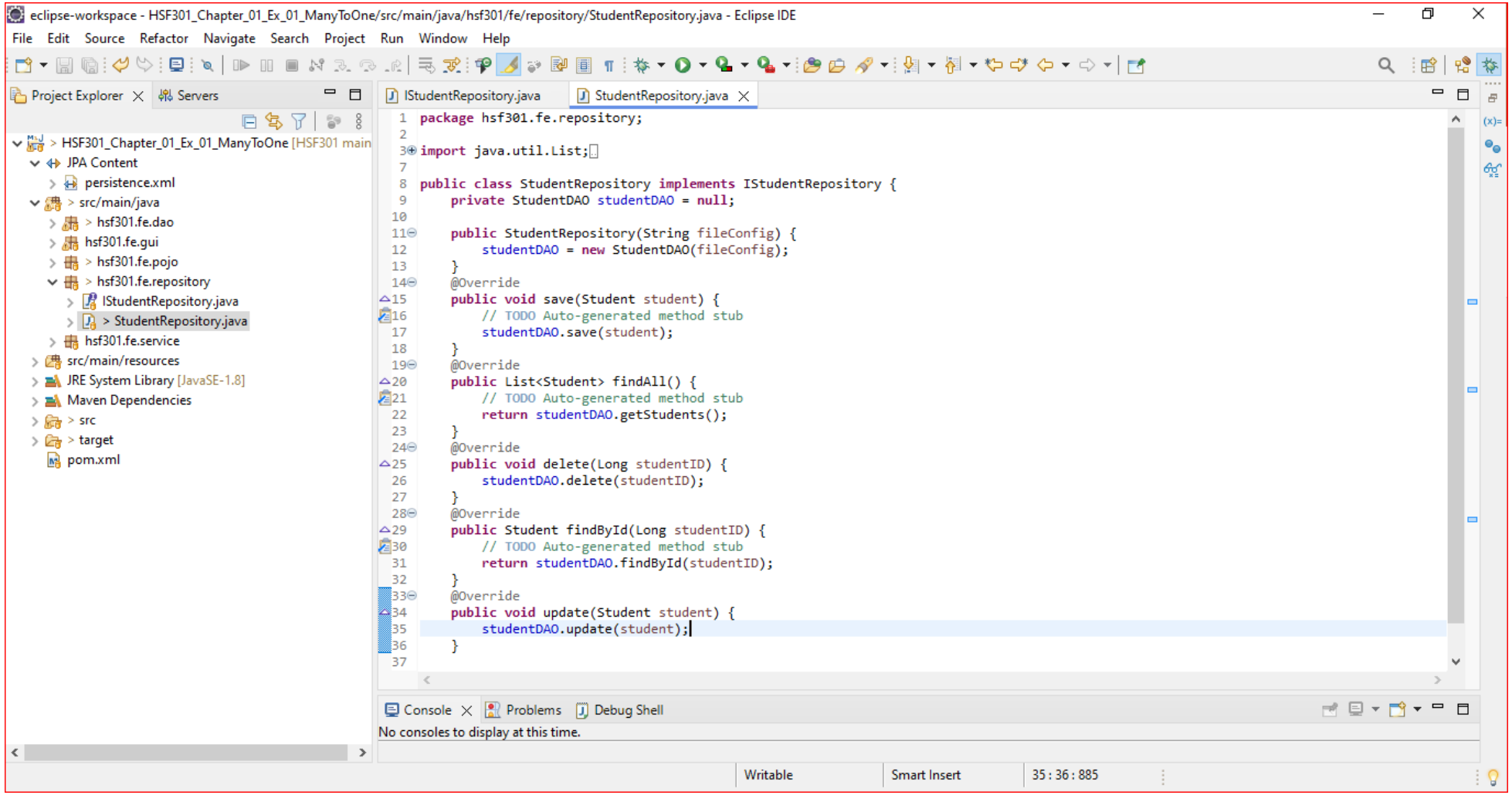
14. Get a Student in StudentDAO



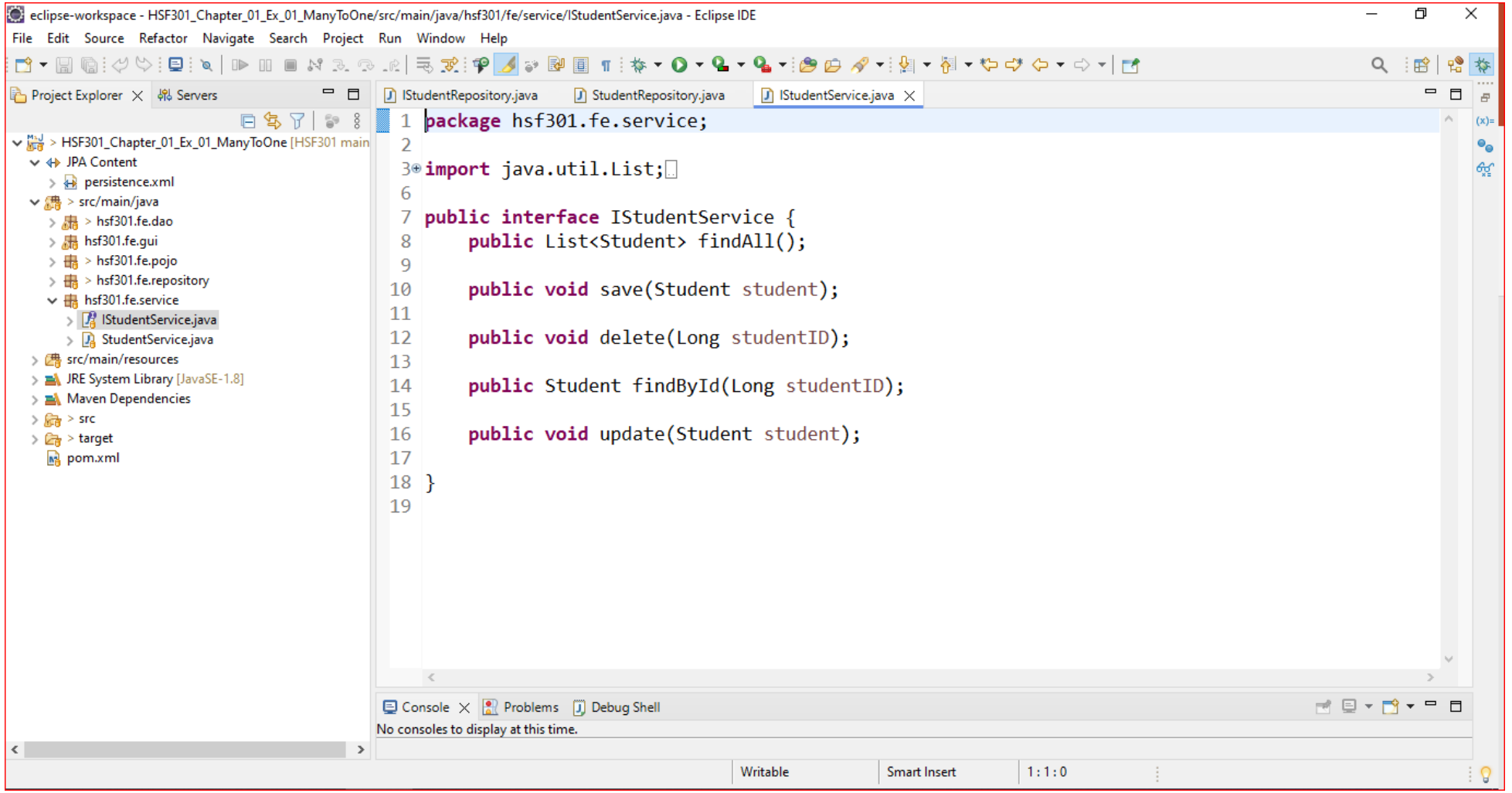
15. Create IStudentRepository



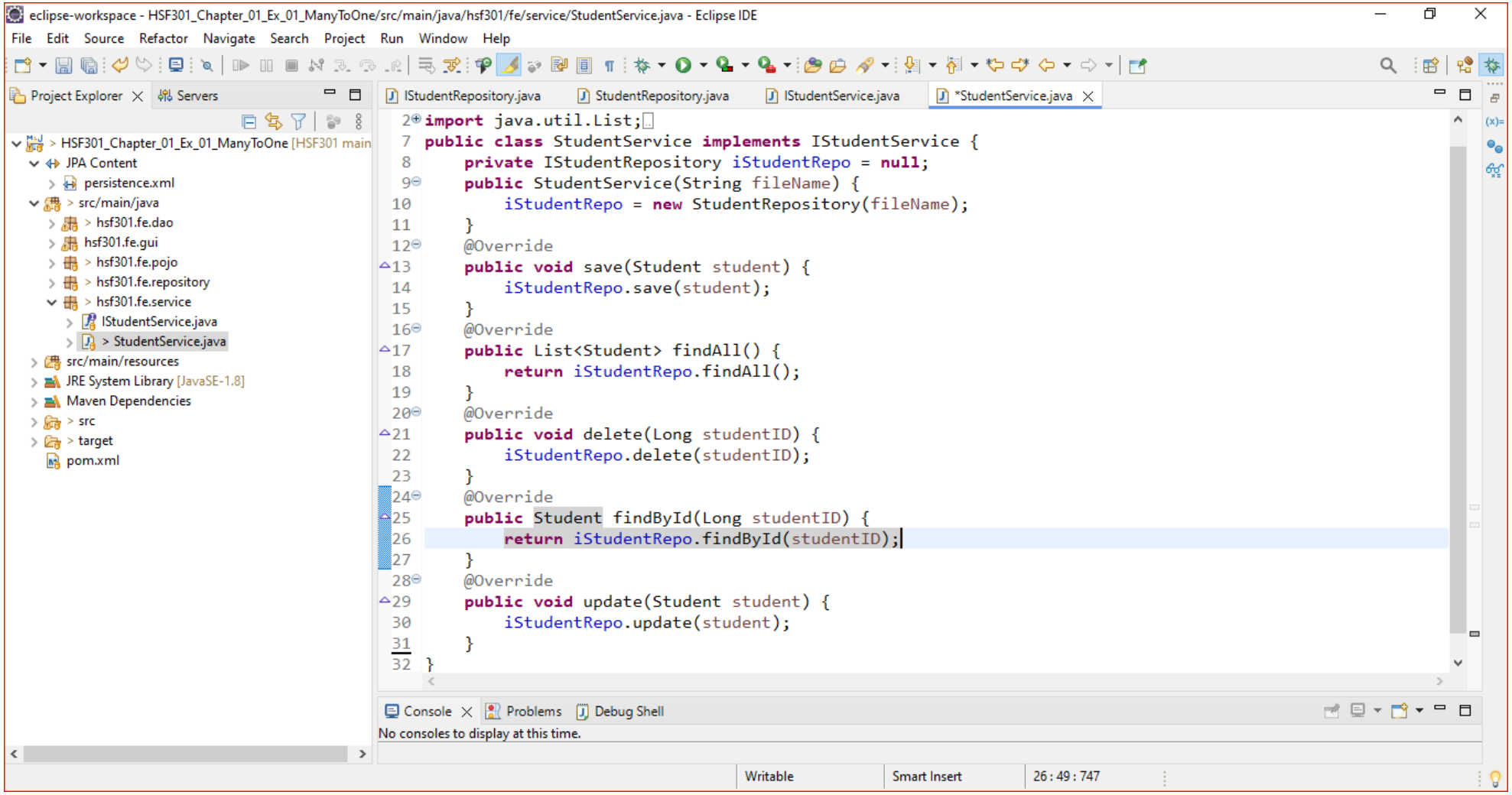
16. Create StudentRepository



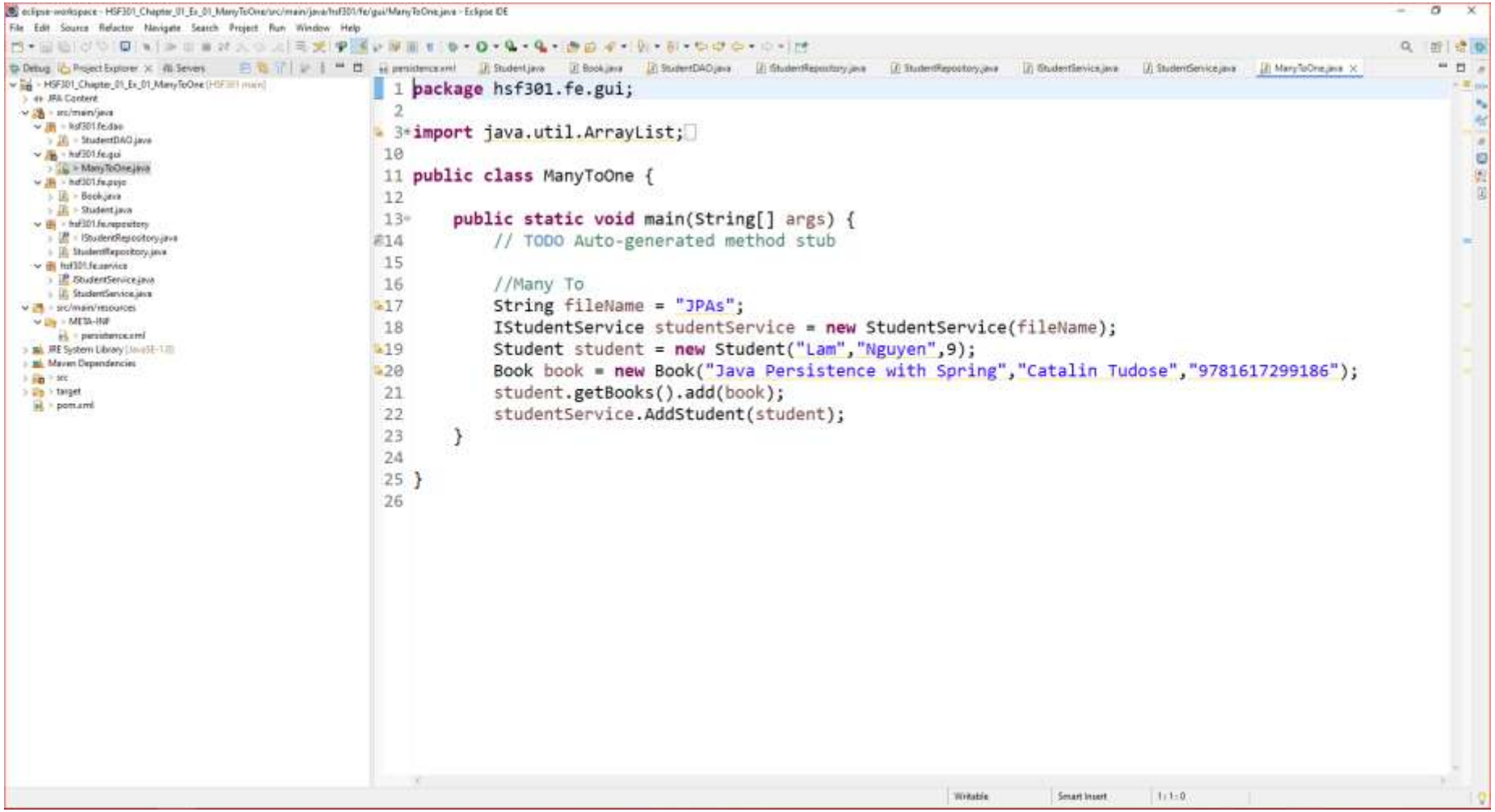
17. Create IStudentService



18. Create StudentService

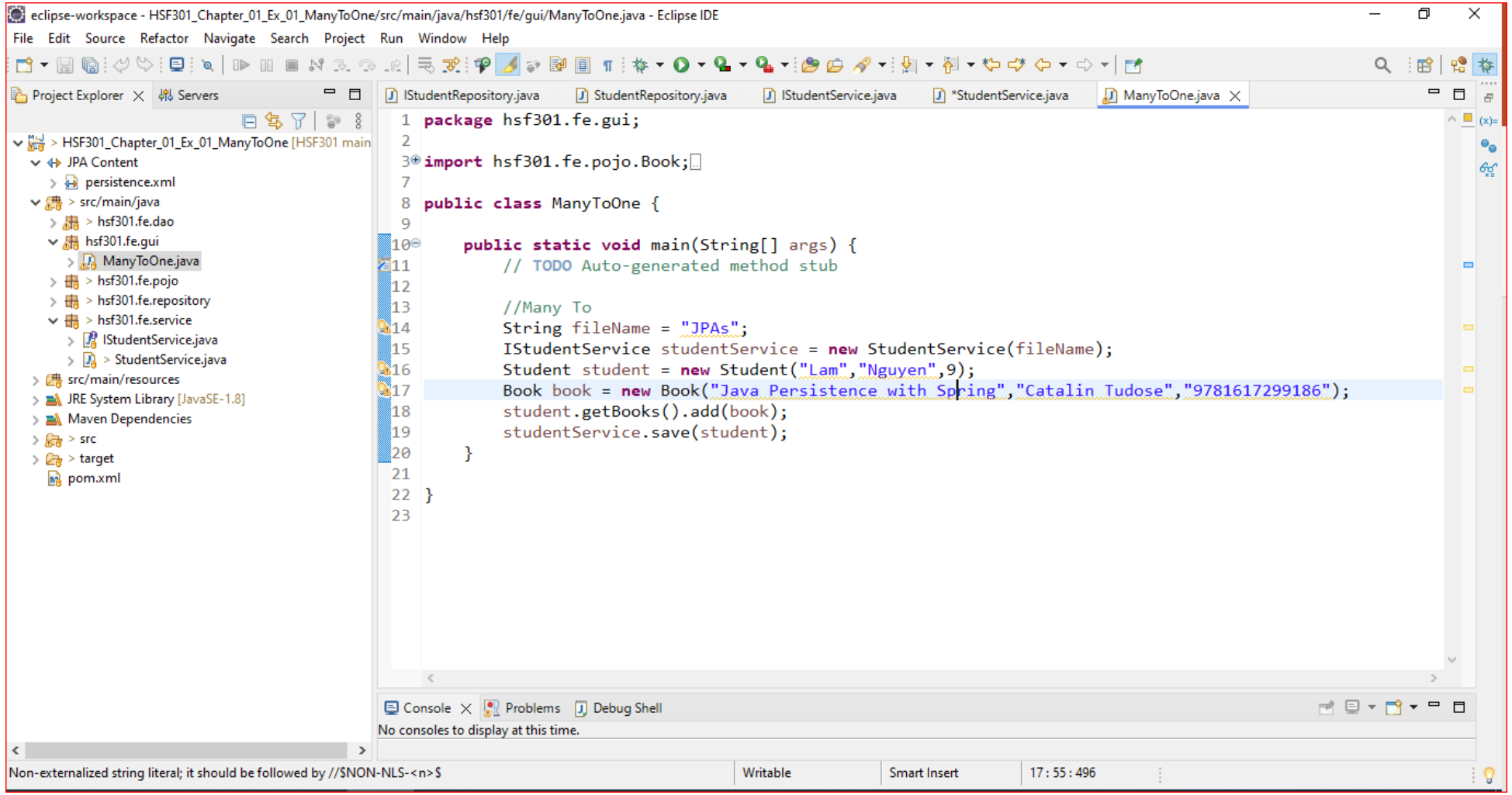


19. Create Main function

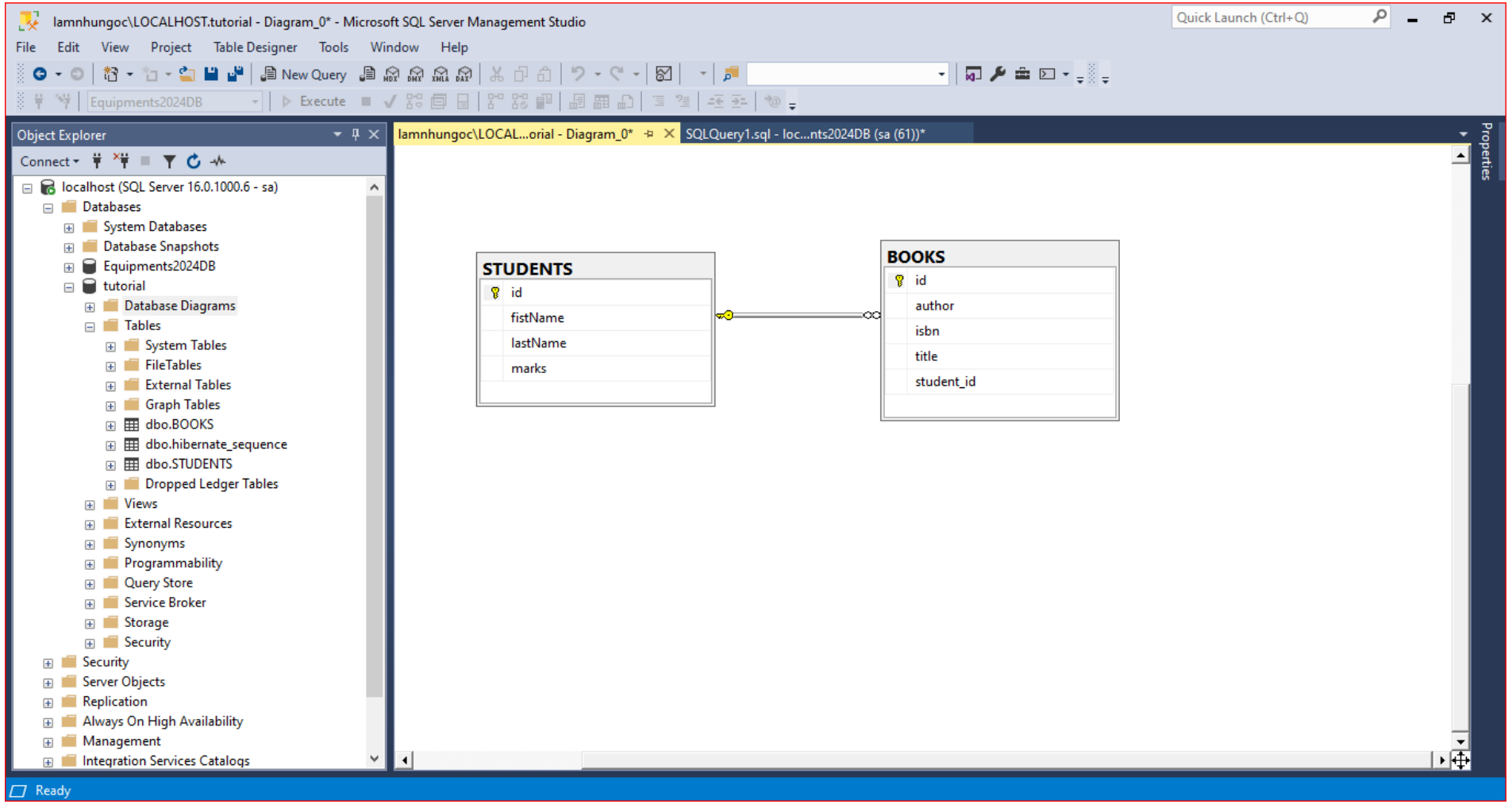


```
1  package hsf301.fe.gui;
2
3  import java.util.ArrayList;
10
11 public class ManyToOne {
12
13     public static void main(String[] args) {
14         // TODO Auto-generated method stub
15
16         //Many To
17         String fileName = "JPAs";
18         IStudentService studentService = new StudentService(fileName);
19         Student student = new Student("Lam","Nguyen",9);
20         Book book = new Book("Java Persistence with Spring","Catalin Tudose","9781617299186");
21         student.getBooks().add(book);
22         studentService.AddStudent(student);
23     }
24
25 }
26
```

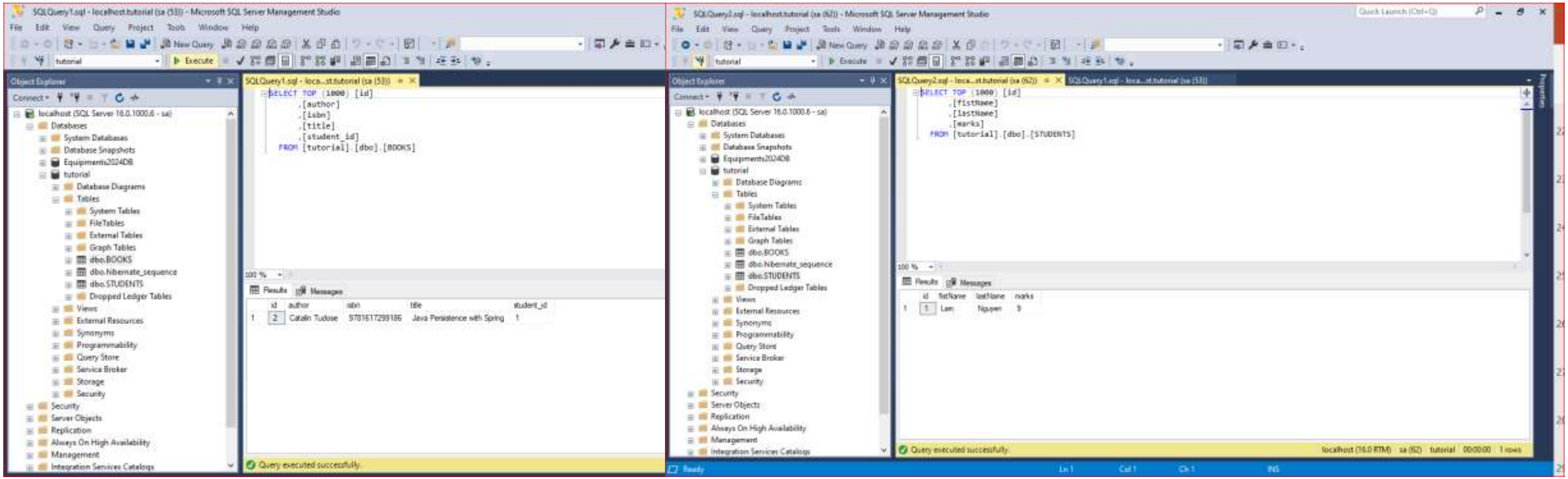
20. Run Program



21. Result


lamnhungoc\LOCALHOST.tutorial - Diagram_0* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Project Table Designer Tools Window Help
New Query
Equipments2024DB
Execute
Object Explorer
Connect
localhost (SQL Server 16.0.1000.6 - sa)
Databases
System Databases
Database Snapshots
Equipments2024DB
tutorial
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.BOOKS
dbo.hibernate_sequence
dbo.STUDENTS
Dropped Ledger Tables
Views
External Resources
Synonyms
Programmability
Query Store
Service Broker
Storage
Security
Security
Server Objects
Replication
Always On High Availability
Management
Integration Services Catalogs
lamnhungoc\LOCAL...orial - Diagram_0*
SQLQuery1.sql - loc...nts2024DB (sa (61))*
STUDENTS
id
fistName
lastName
marks
BOOKS
id
author
isbn
title
student_id
Properties
Ready

22. Result

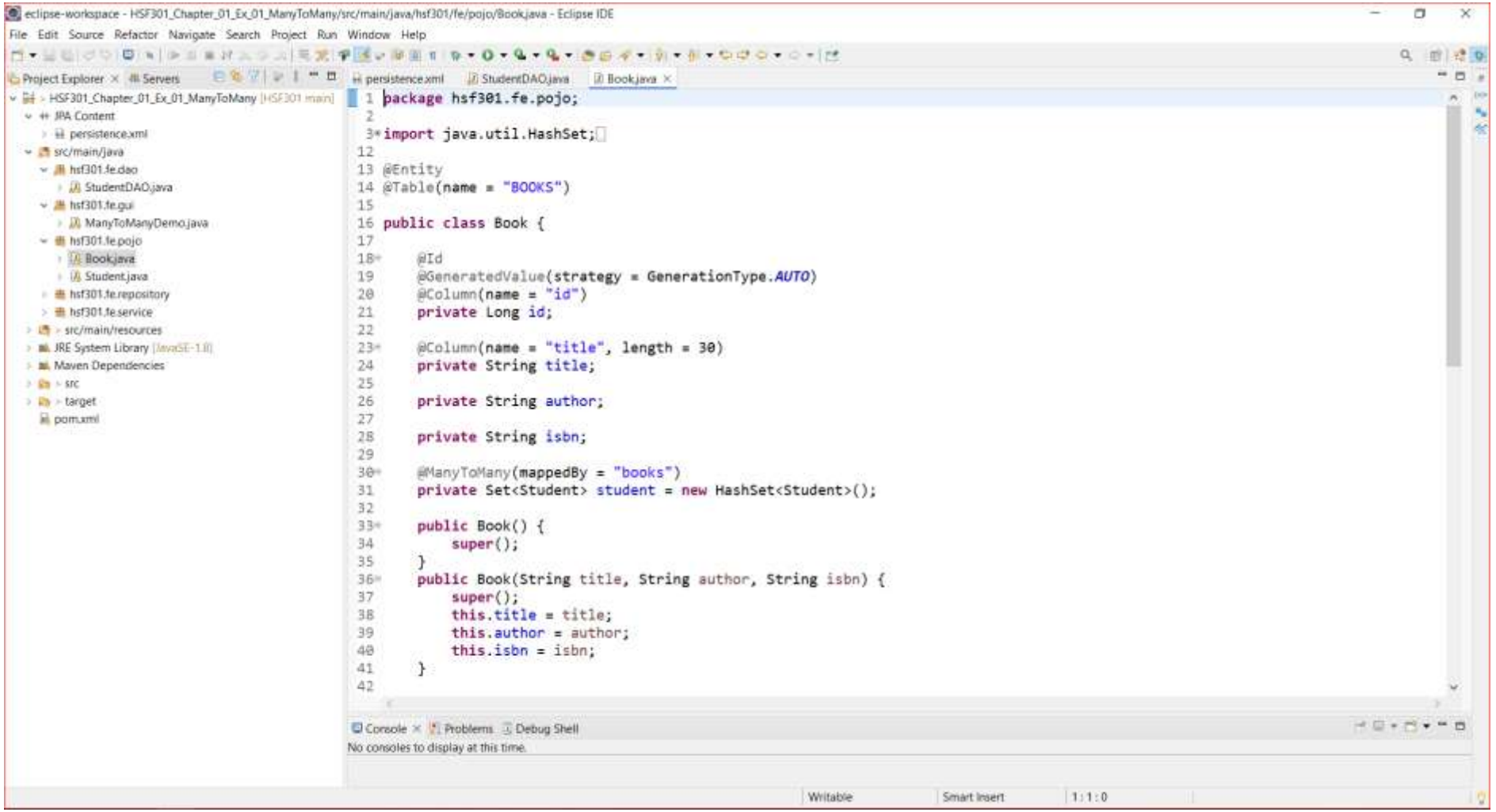

SELECT TOP (1000) [id]
,[author]
,[isbn]
,[title]
,[student_id]
FROM [tutorial].[dbo].[BOOKS]
SELECT TOP (1000) [id]
,[fistNaee]
,[lastNaee]
,[marks]
FROM [tutorial].[dbo].[STUDENTS]
Query executed successfully.

FPT Education
FPT UNIVERSITY

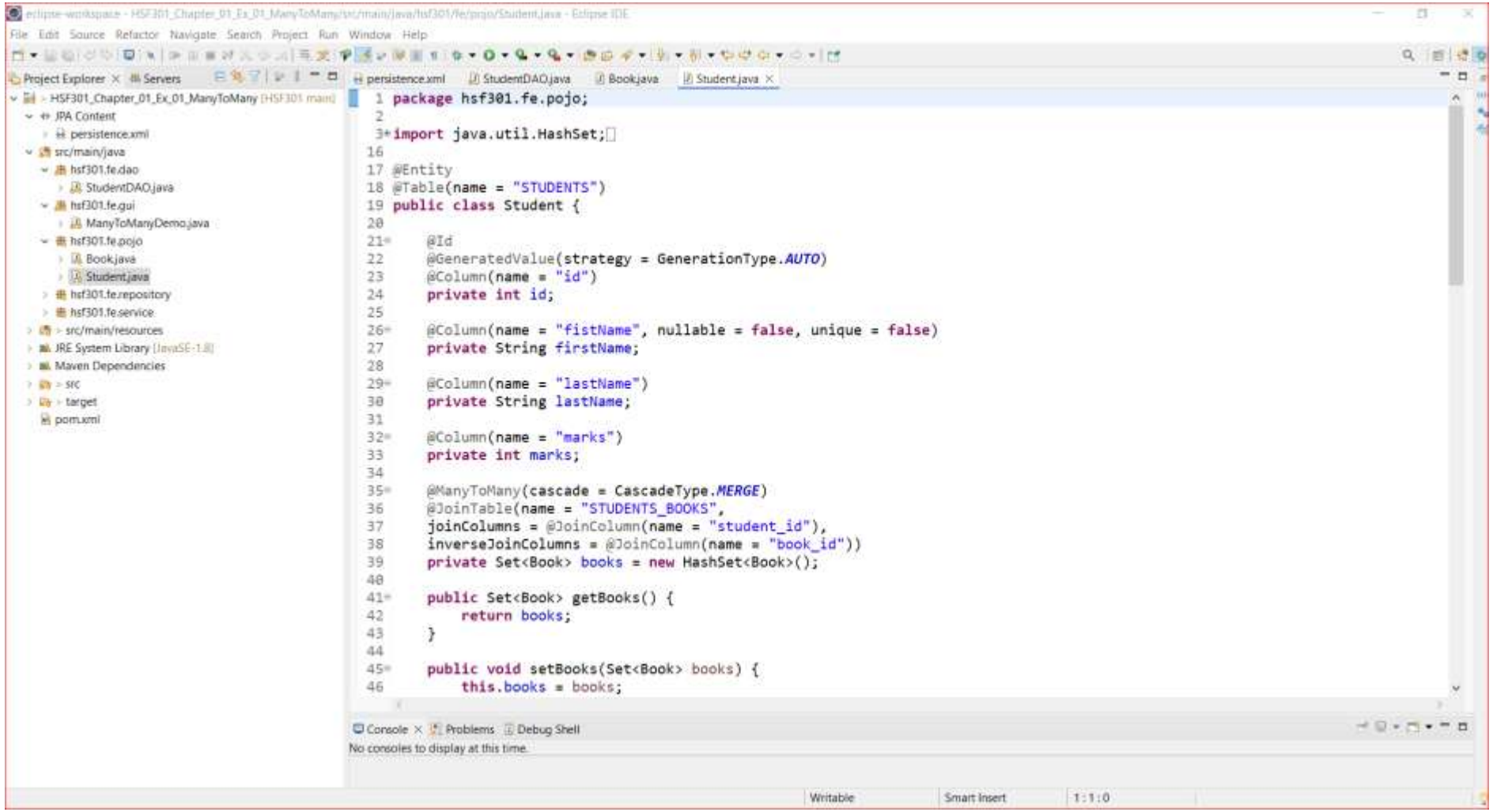
Java

Demo JPA
(Many To Many)

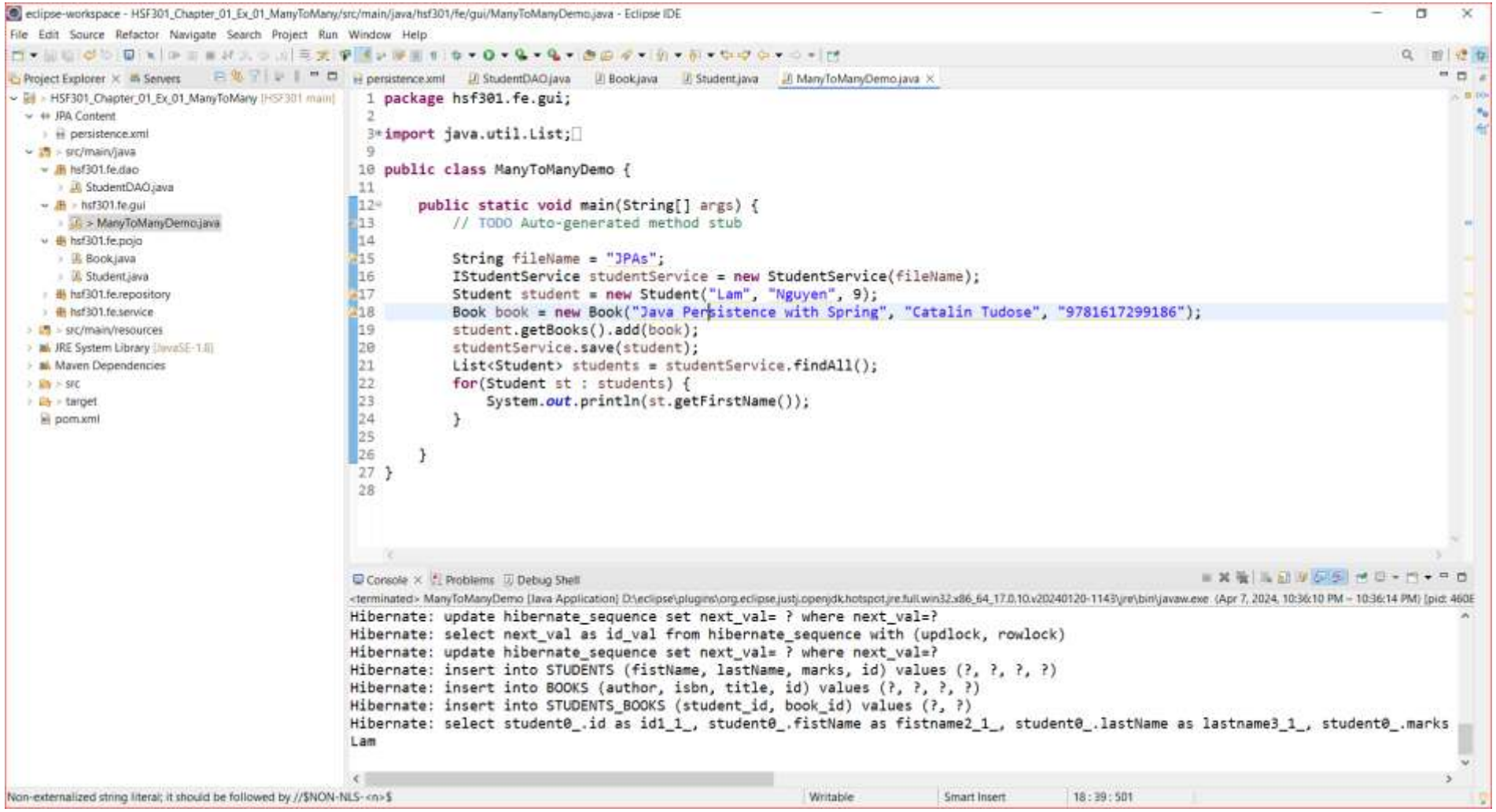
1. Create Books in Pojo's Package



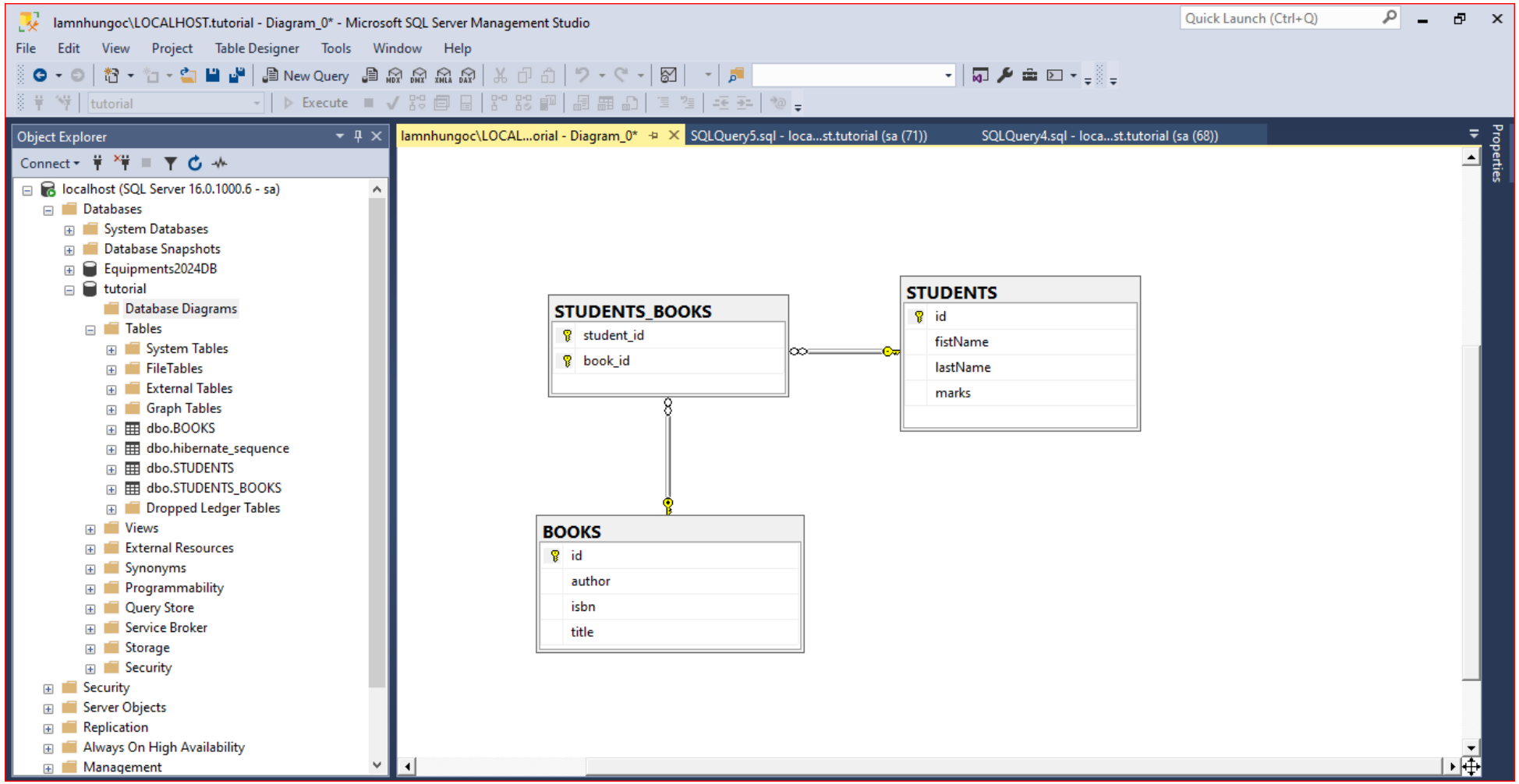
2. Create Students in Pojo's Package



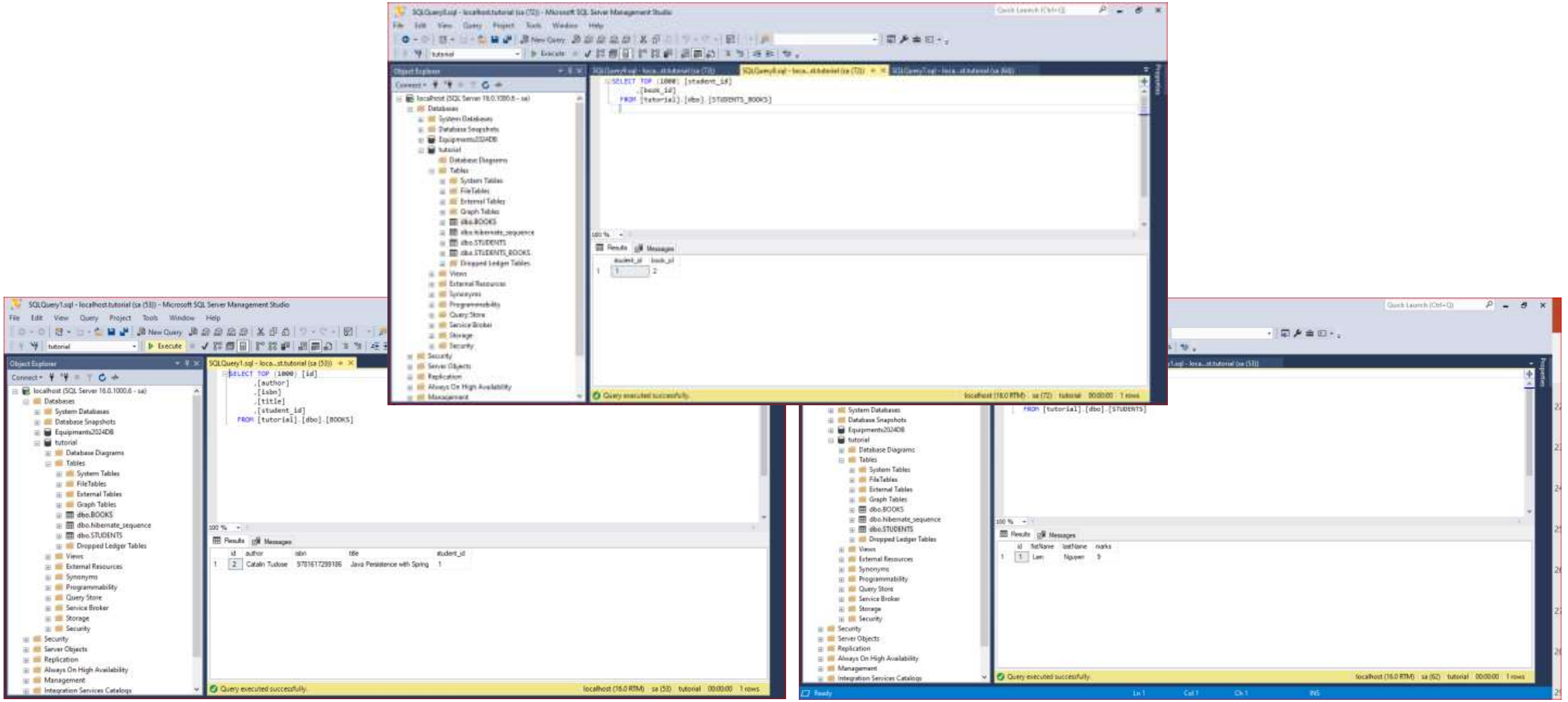
3. Run Program



4. Result


lamnhungoc\LOCALHOST.tutorial - Diagram_0* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Project Table Designer Tools Window Help
New Query
Execute
tutorial
Object Explorer
Connect
localhost (SQL Server 16.0.1000.6 - sa)
Databases
System Databases
Database Snapshots
Equipments2024DB
tutorial
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.BOOKS
dbo.hibernate_sequence
dbo.STUDENTS
dbo.STUDENTS_BOOKS
Dropped Ledger Tables
Views
External Resources
Synonyms
Programmability
Query Store
Service Broker
Storage
Security
Security
Server Objects
Replication
Always On High Availability
Management
lamnhungoc\LOCAL...orial - Diagram_0*
SQLQuery5.sql - loca...st.tutorial (sa (71))
SQLQuery4.sql - loca...st.tutorial (sa (68))
Properties
STUDENTS_BOOKS
student_id
book_id
STUDENTS
id
fistName
lastName
marks
BOOKS
id
author
isbn
title

5. Result



Summary

- Concepts were introduced:
  - Overview about JPA
  - Architecture Overview new features of JPA
  - Why JPA is selected as develop application?
  - Explain and demo using Eclipse IDE to create JPA Console App
  - Create and Run cross-platform Console application with Java connect to MSSQL with Repository Pattern