

# 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 allowing focus on business logic.

# Contents

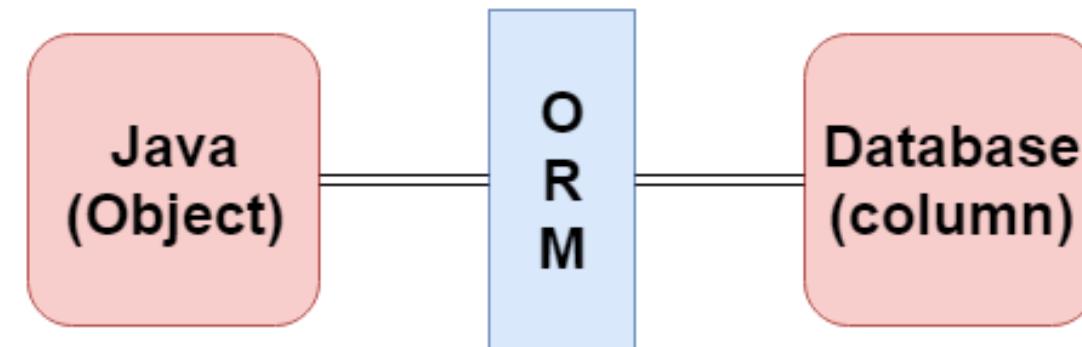


- ❖ Introduction
- ❖ Key Concepts
- ❖ Annotations
- ❖ Relationships
- ❖ Using JPA
- ❖ Demo
- ❖ Advantages and Disadvantages

# 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 application code (to some extent)

# Popular ORM Frameworks for Java

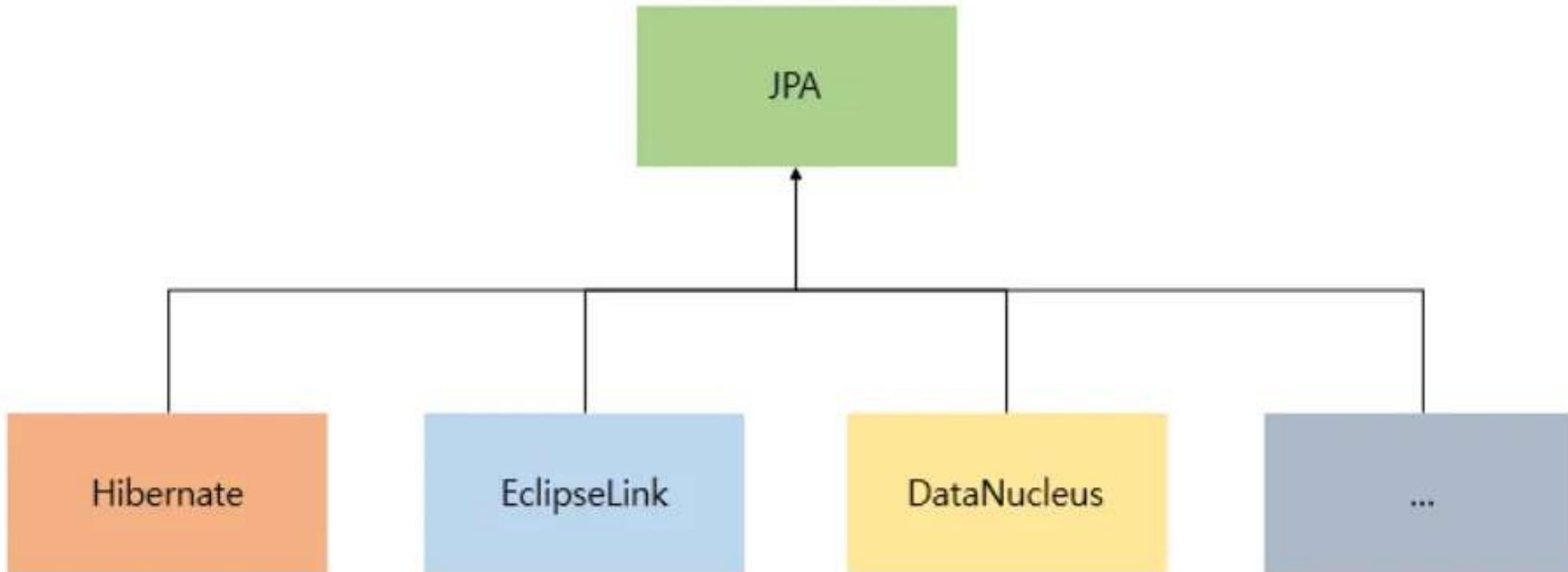
- ❖ **Hibernate:** The most widely used JPA implementation, offering a rich feature set and extensive customization options.
- ❖ **EclipseLink:** Another JPA implementation with a focus on standards compliance and portability.
- ❖ **Spring Data JPA:** An abstraction layer on top of JPA providers, simplifying data 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.

# 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 Java 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

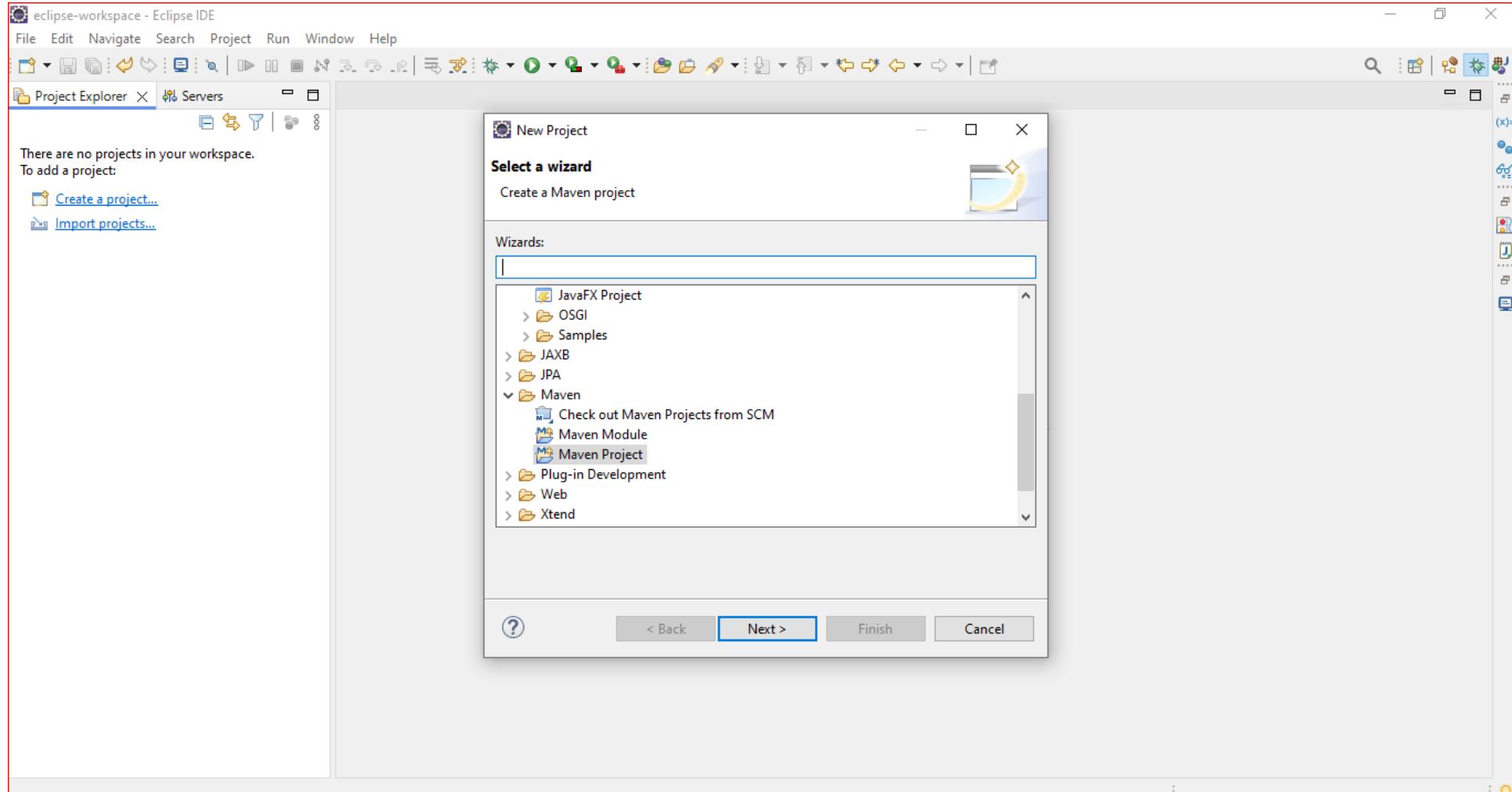
- ❖ The entity manager implements the API and encapsulates all of them within a single interface.
  - **Manages entity lifecycle:** Persist, find, merge, remove
  - **Controls persistence context:** Cache, flush, detach
  - **Executes queries:** JPQL, Criteria API
  - **Handles transactions:** Begin,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

# Key Operations of the Entity Manager

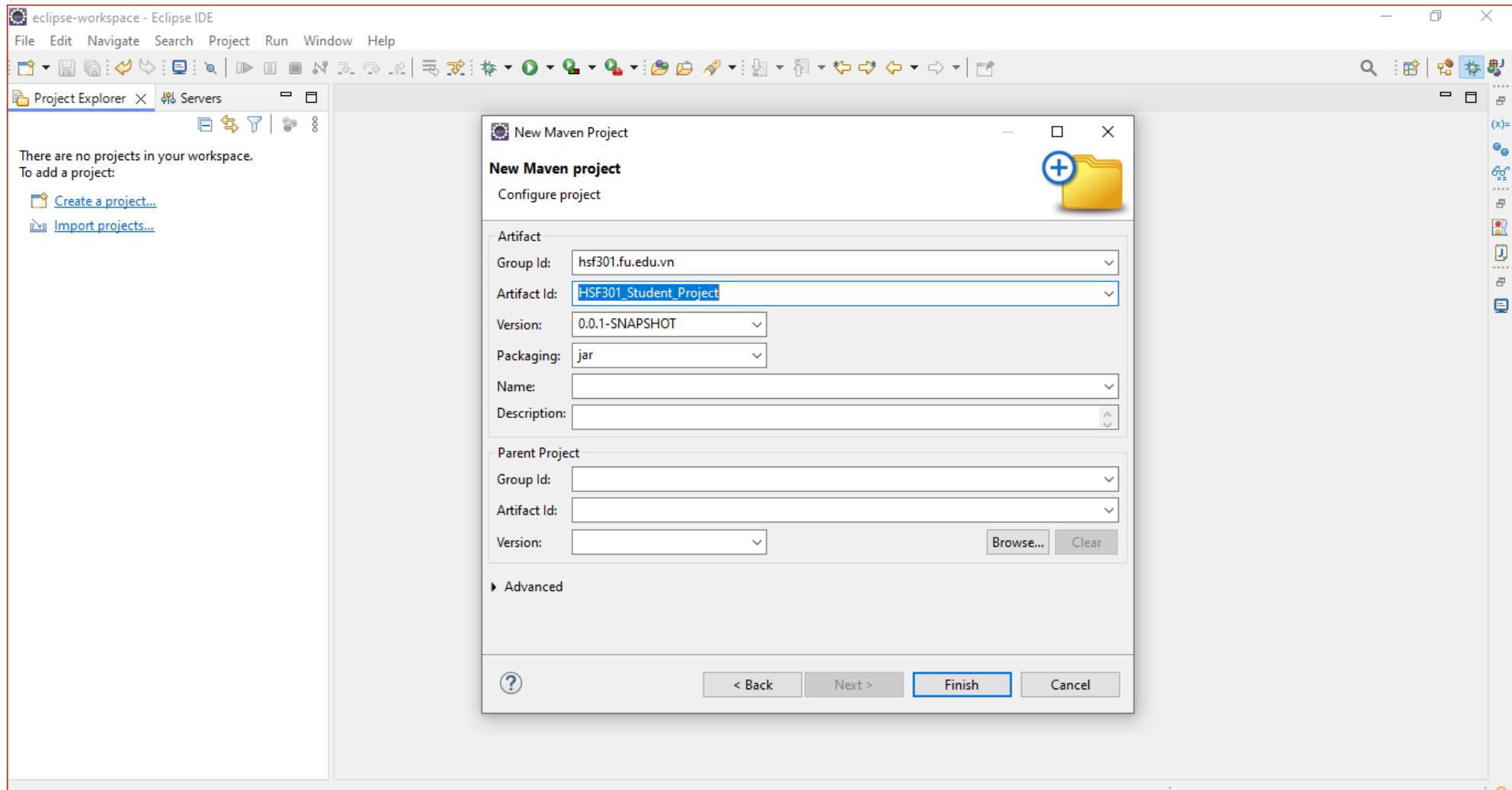
- ❖ **Persisting Entities:** `em.persist(entity)` makes a transient entity instance persistent
- ❖ **Finding Entities:** `em.find(entityClass,primarykey)` retrieves an entity by its primary key.
- ❖ **Merging Entities:** `em.merge(entity)` merges a detached entity instance into the current persistence 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.

# Demo Simple CRUD with JPA

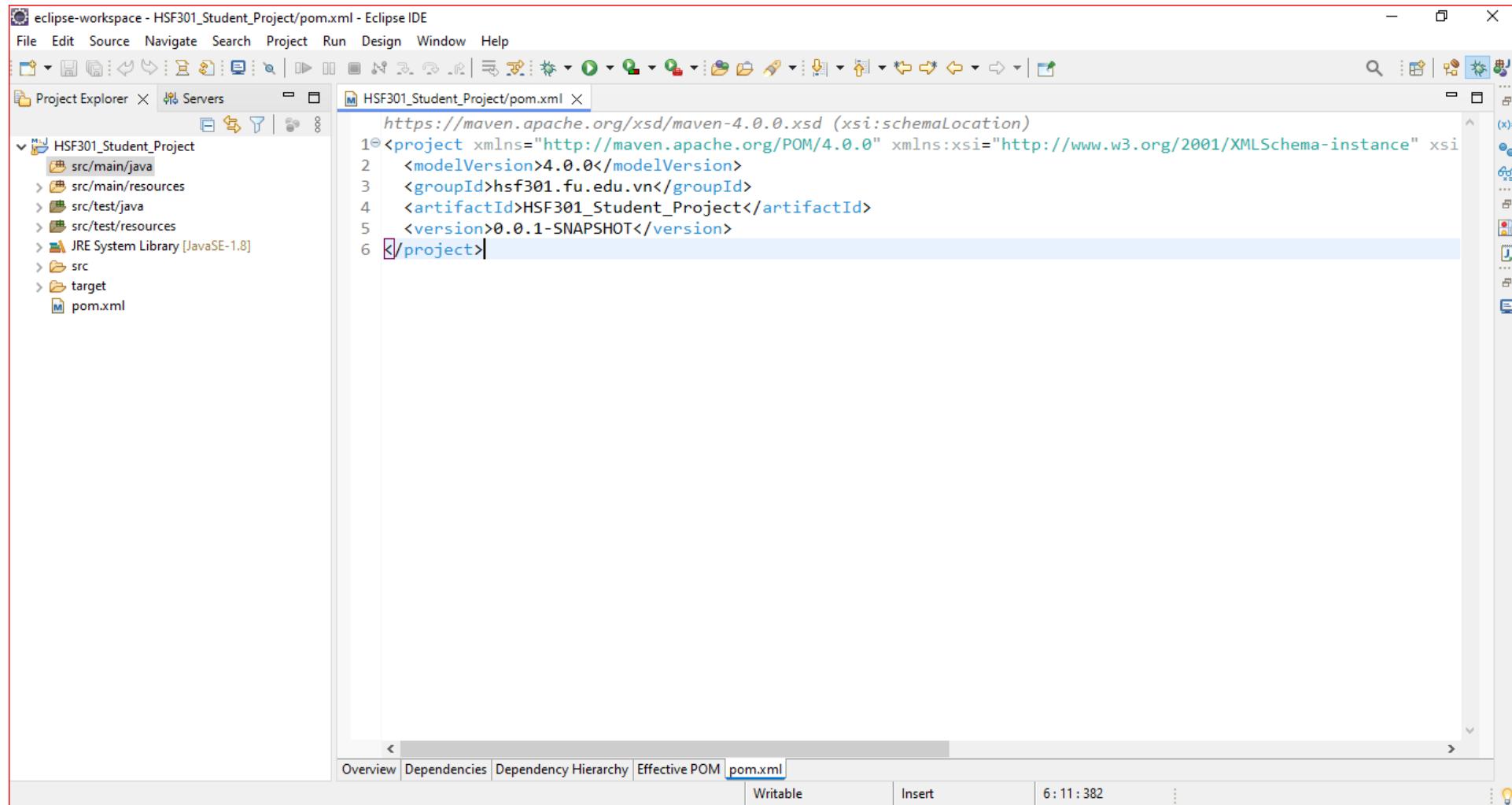
# 1. Open Eclipse, File | New | Maven Project



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



### 3. The New Project

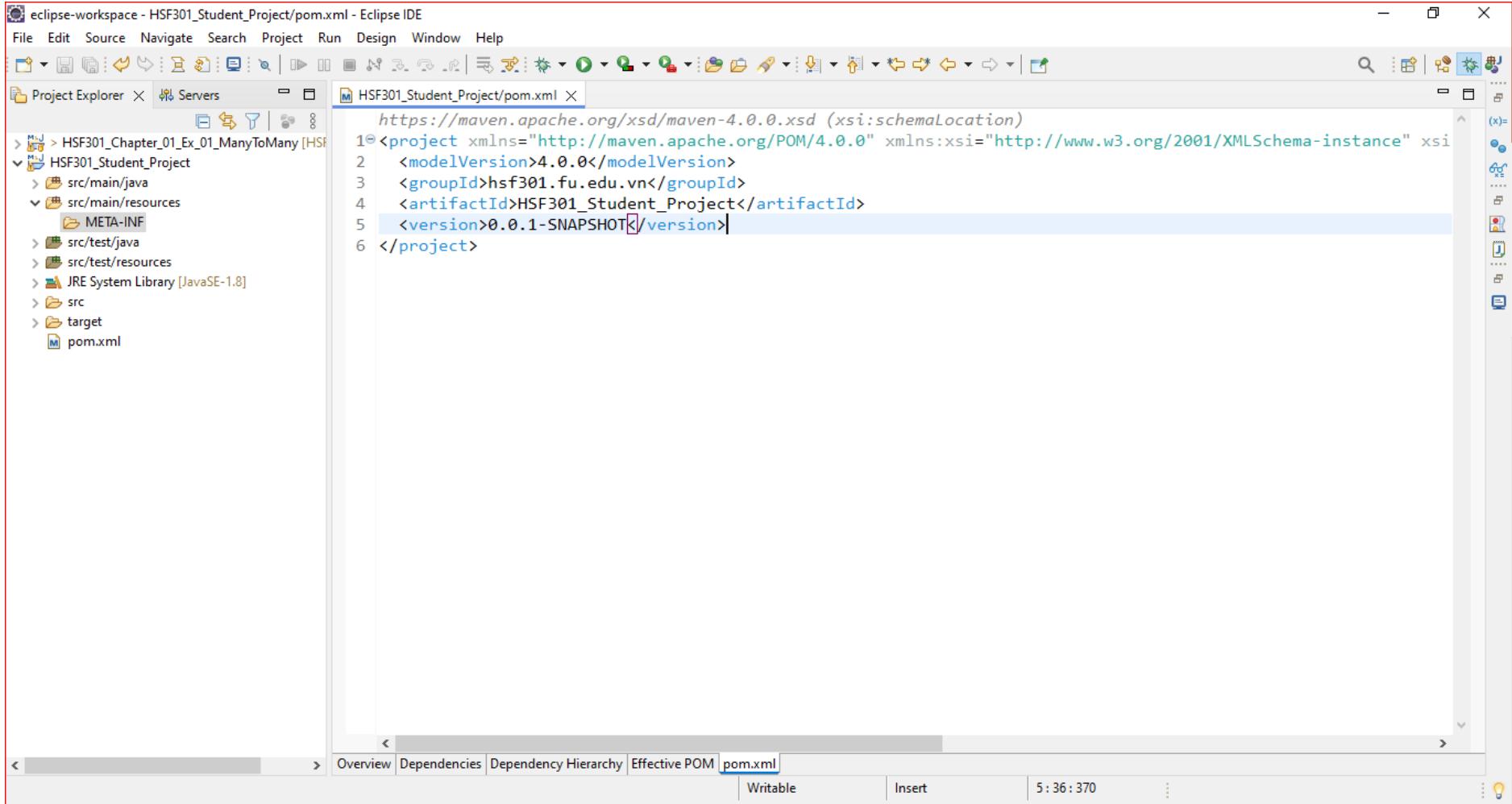


The screenshot shows the Eclipse IDE interface with a red border around the main window. On the left, the Project Explorer view displays a project named "HSF301\_Student\_Project" containing directories for Java source code, resources, and test code, along with a JRE System Library entry for JavaSE-1.8. In the center, the editor view shows the file "pom.xml" with the following XML content:

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi
2  <modelVersion>4.0.0</modelVersion>
3  <groupId>hsf301.fu.edu.vn</groupId>
4  <artifactId>HSF301_Student_Project</artifactId>
5  <version>0.0.1-SNAPSHOT</version>
6 </project>|
```

The bottom of the editor shows tabs for Overview, Dependencies, Dependency Hierarchy, Effective POM, and pom.xml, with the latter being the active tab. A status bar at the bottom indicates the file is Writable and shows the time as 6:11:382.

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

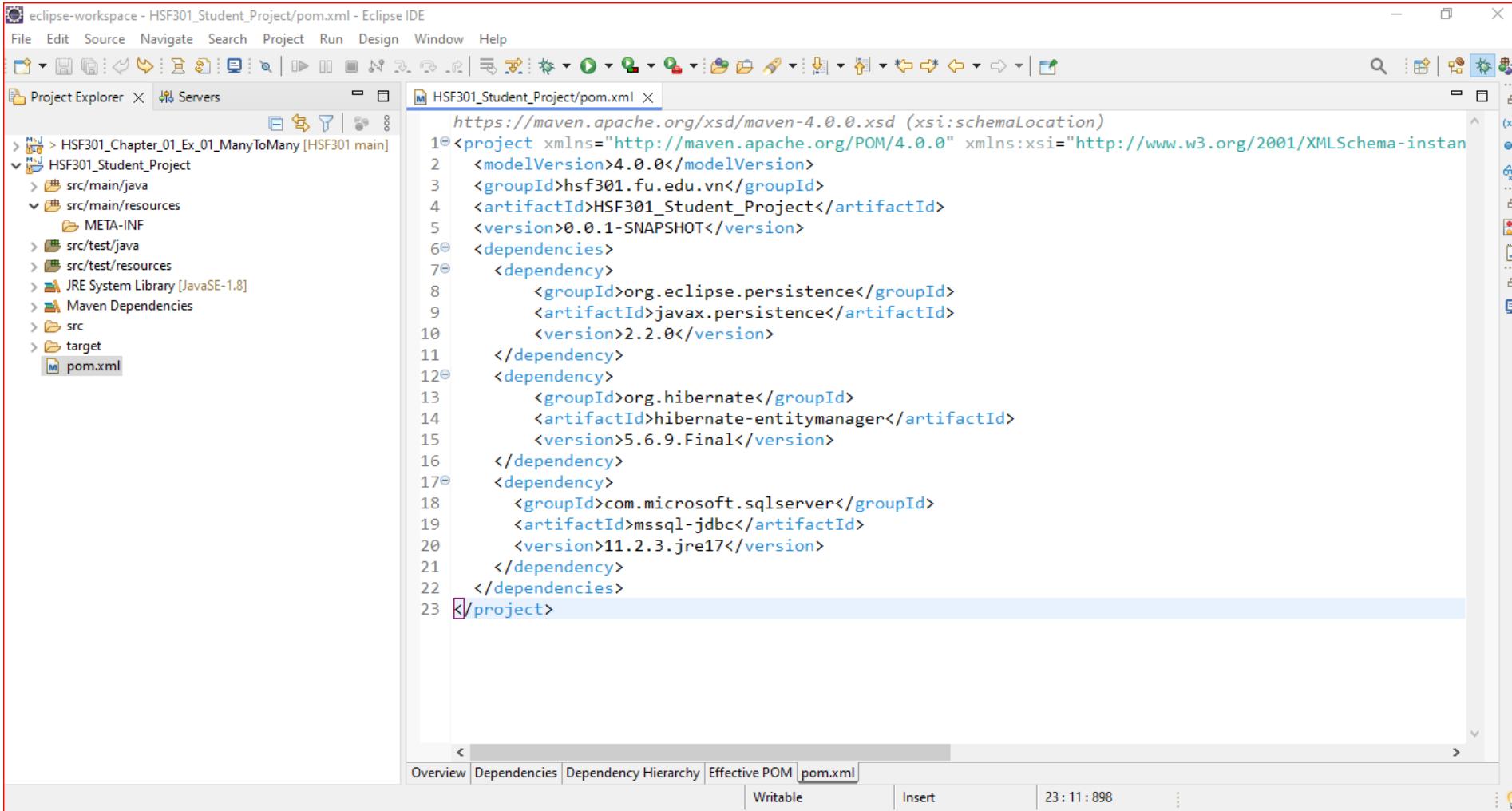


The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HSF301\_Student\_Project/pom.xml - Eclipse IDE
- Menu Bar:** File, Edit, Source, Navigate, Search, Project, Run, Design, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Project Explorer:** Shows the project structure:
  - HSF301\_Chapter\_01\_Ex\_01\_ManyToMany [HSF301 Chapter 01 Ex 01 ManyToMany]
  - HSF301\_Student\_Project
    - src/main/java
    - src/main/resources
      - META-INF
    - src/test/java
    - src/test/resources
    - JRE System Library [JavaSE-1.8]
    - src
    - target
    - pom.xml
- Code Editor:** Displays the pom.xml file content:

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi
2  <modelVersion>4.0.0</modelVersion>
3  <groupId>hsf301.fu.edu.vn</groupId>
4  <artifactId>HSF301_Student_Project</artifactId>
5  <version>0.0.1-SNAPSHOT</version>
6</project>
```
- Bottom Bar:** Navigation tabs (Overview, Dependencies, Dependency Hierarchy, Effective POM, pom.xml), status bar (Writable, Insert, 5:36:370).

# 5. Edit the pom.xml

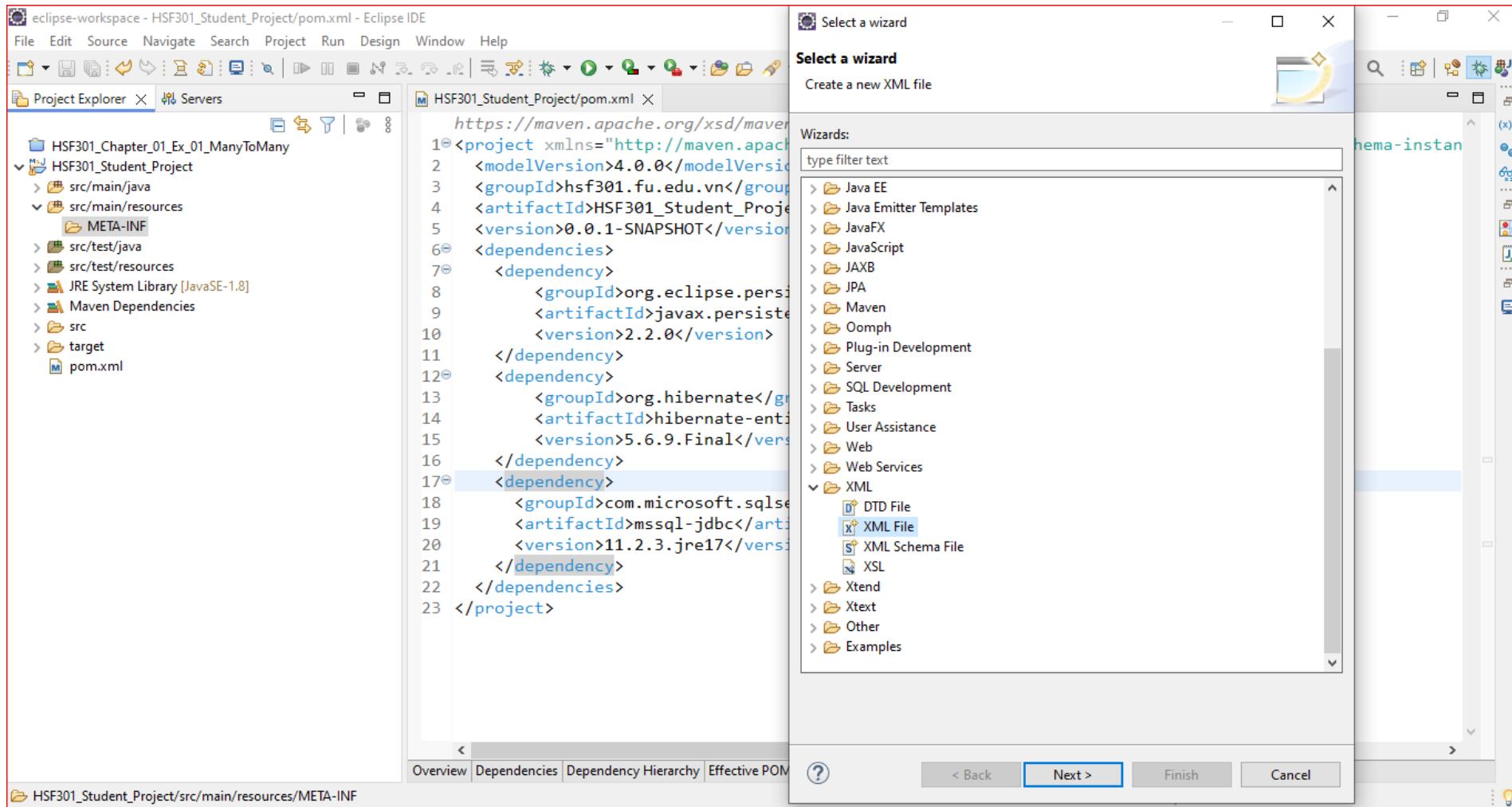


The screenshot shows the Eclipse IDE interface with a red border around the central workspace. On the left, the Project Explorer view shows a project named "HSF301\_Student\_Project" containing "src/main/java", "src/main/resources" (with "META-INF" and "JRE System Library [JavaSE-1.8]"), "src/test/java", "src/test/resources", "Maven Dependencies", "src", and "target". A file named "pom.xml" is selected in the Project Explorer. The main editor area displays the XML content of the "pom.xml" file:

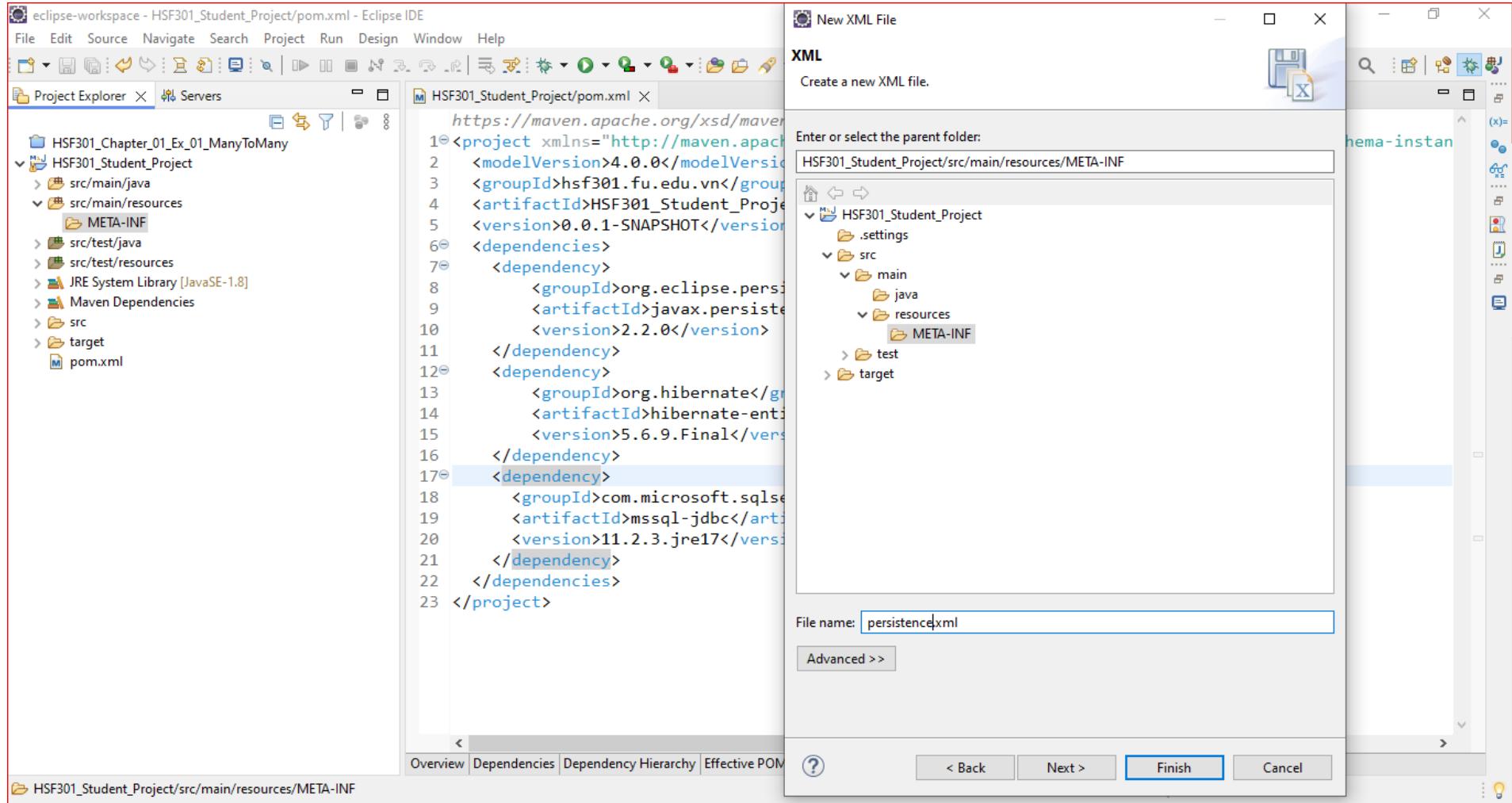
```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
2  <modelVersion>4.0.0</modelVersion>
3  <groupId>hsf301.fu.edu.vn</groupId>
4  <artifactId>HSF301_Student_Project</artifactId>
5  <version>0.0.1-SNAPSHOT</version>
6<dependencies>
7<dependency>
8    <groupId>org.eclipse.persistence</groupId>
9    <artifactId>javax.persistence</artifactId>
10   <version>2.2.0</version>
11</dependency>
12<dependency>
13    <groupId>org.hibernate</groupId>
14    <artifactId>hibernate-entitymanager</artifactId>
15    <version>5.6.9.Final</version>
16</dependency>
17<dependency>
18    <groupId>com.microsoft.sqlserver</groupId>
19    <artifactId>mssql-jdbc</artifactId>
20    <version>11.2.3.jre17</version>
21</dependency>
22</dependencies>
23</project>
```

The status bar at the bottom indicates the file is "Writable" and has 23 lines, 11:898 characters, and was last modified at 23:11:898.

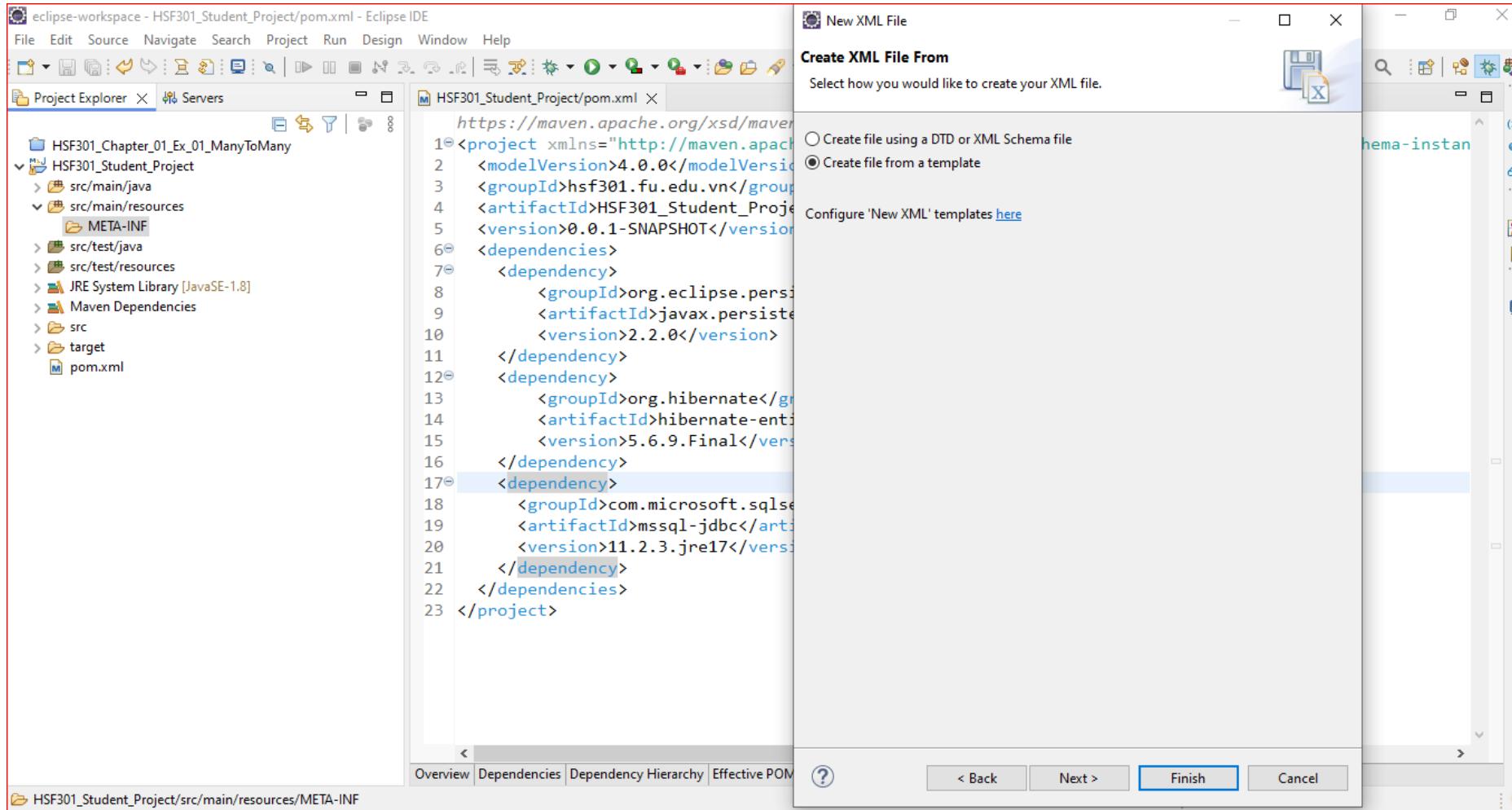
# 6. Create persistence.xml in META-INF folder



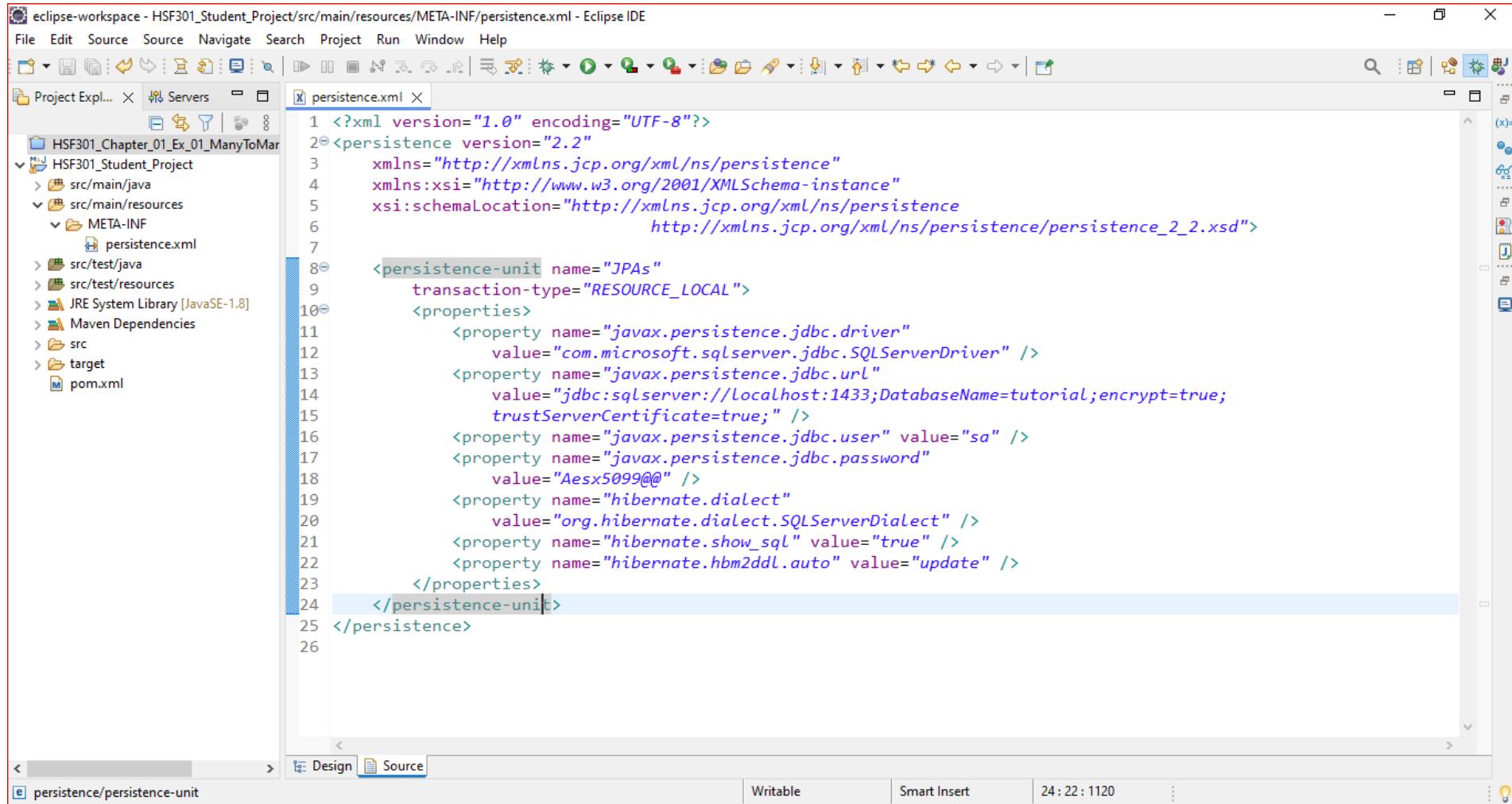
# 7. Create persistence.xml in META-INF folder



# 8. Create persistence.xml in META-INF folder



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



The screenshot shows the Eclipse IDE interface with a red border around the main window. The title bar reads "eclipse-workspace - HSF301\_Student\_Project/src/main/resources/META-INF/persistence.xml - Eclipse IDE". The menu bar includes File, Edit, Source, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left sidebar shows the "Project Explorer" with a tree view of the project structure:

- HSF301\_Chapter\_01\_Ex\_01\_ManyToMany
- HSF301\_Student\_Project
- src/main/java
- src/main/resources
  - META-INF
    - persistence.xml
  - src/test/java
  - src/test/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
- pom.xml

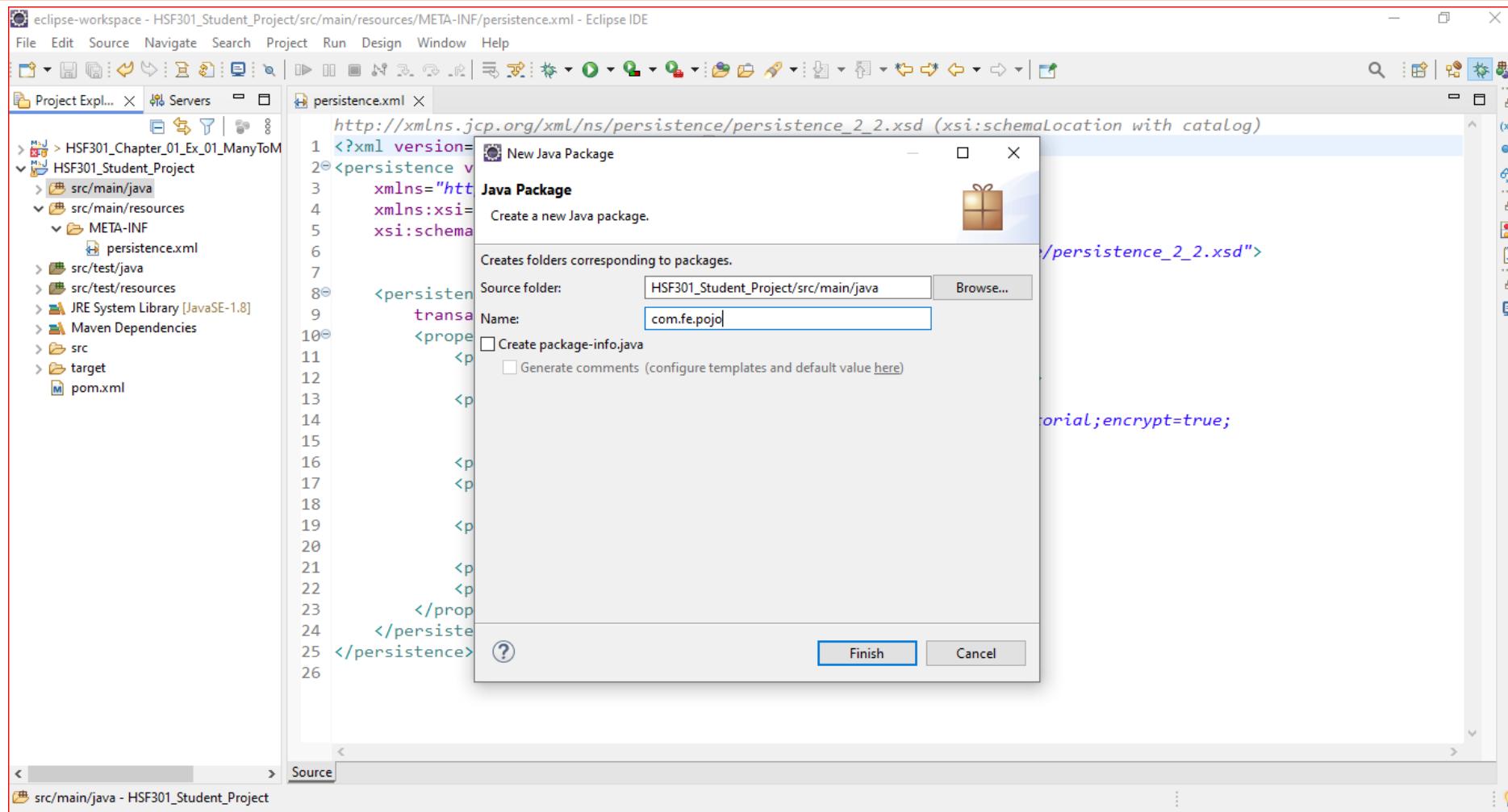
The central editor area displays the XML content of "persistence.xml":

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2"
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">

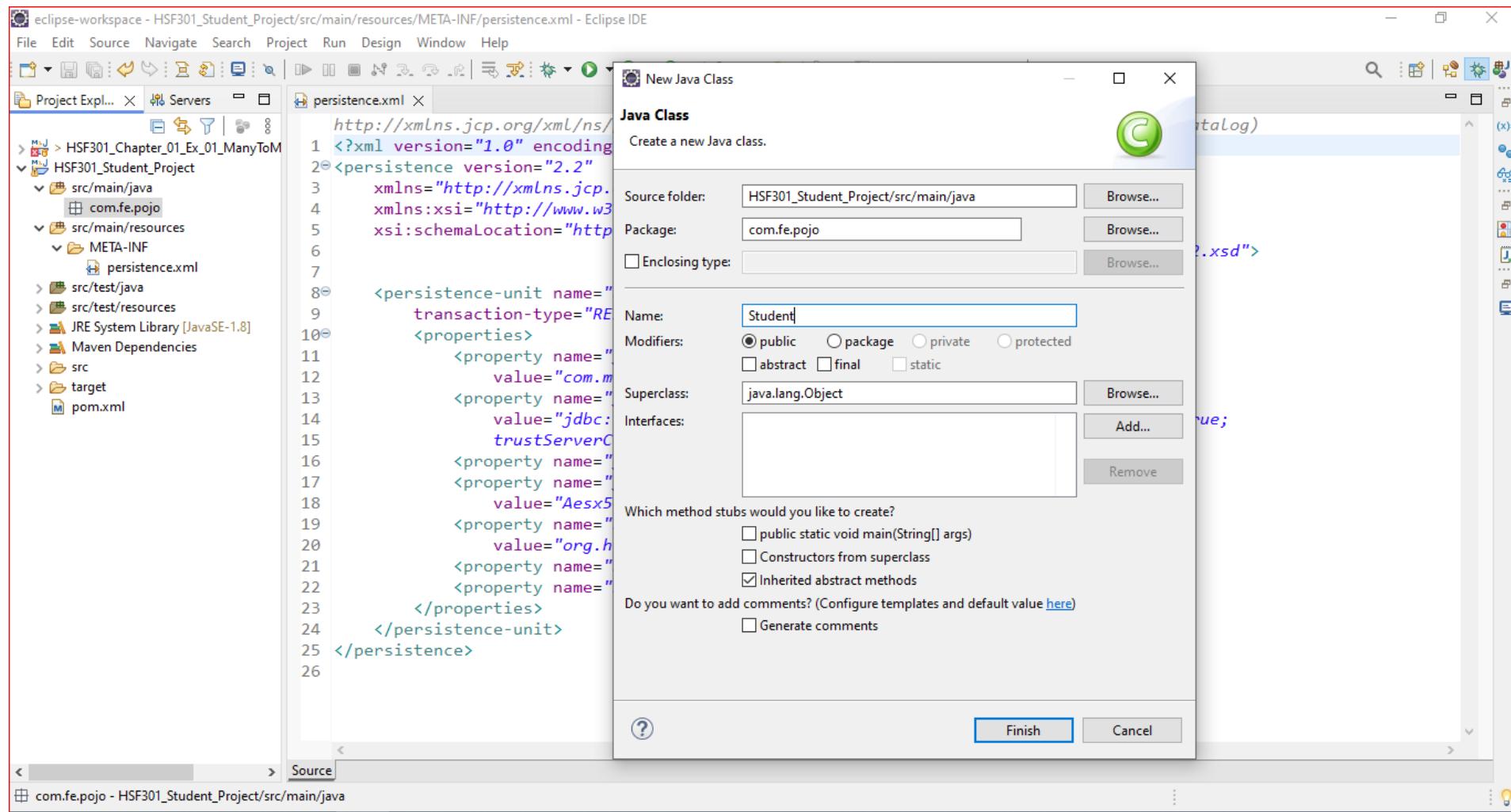
  <persistence-unit name="JPAs"
    transaction-type="RESOURCE_LOCAL">
    <properties>
      <property name="javax.persistence.jdbc.driver"
        value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
      <property name="javax.persistence.jdbc.url"
        value="jdbc:sqlserver://localhost:1433;DatabaseName=tutorial;encrypt=true;
        trustServerCertificate=true;" />
      <property name="javax.persistence.jdbc.user" value="sa" />
      <property name="javax.persistence.jdbc.password"
        value="Aesx5099@@" />
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.SQLServerDialect" />
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.hbm2ddl.auto" value="update" />
    </properties>
  </persistence-unit>
</persistence>
```

The bottom status bar shows "Design" selected, "Writable", "Smart Insert", and the date/time "24/22/1120".

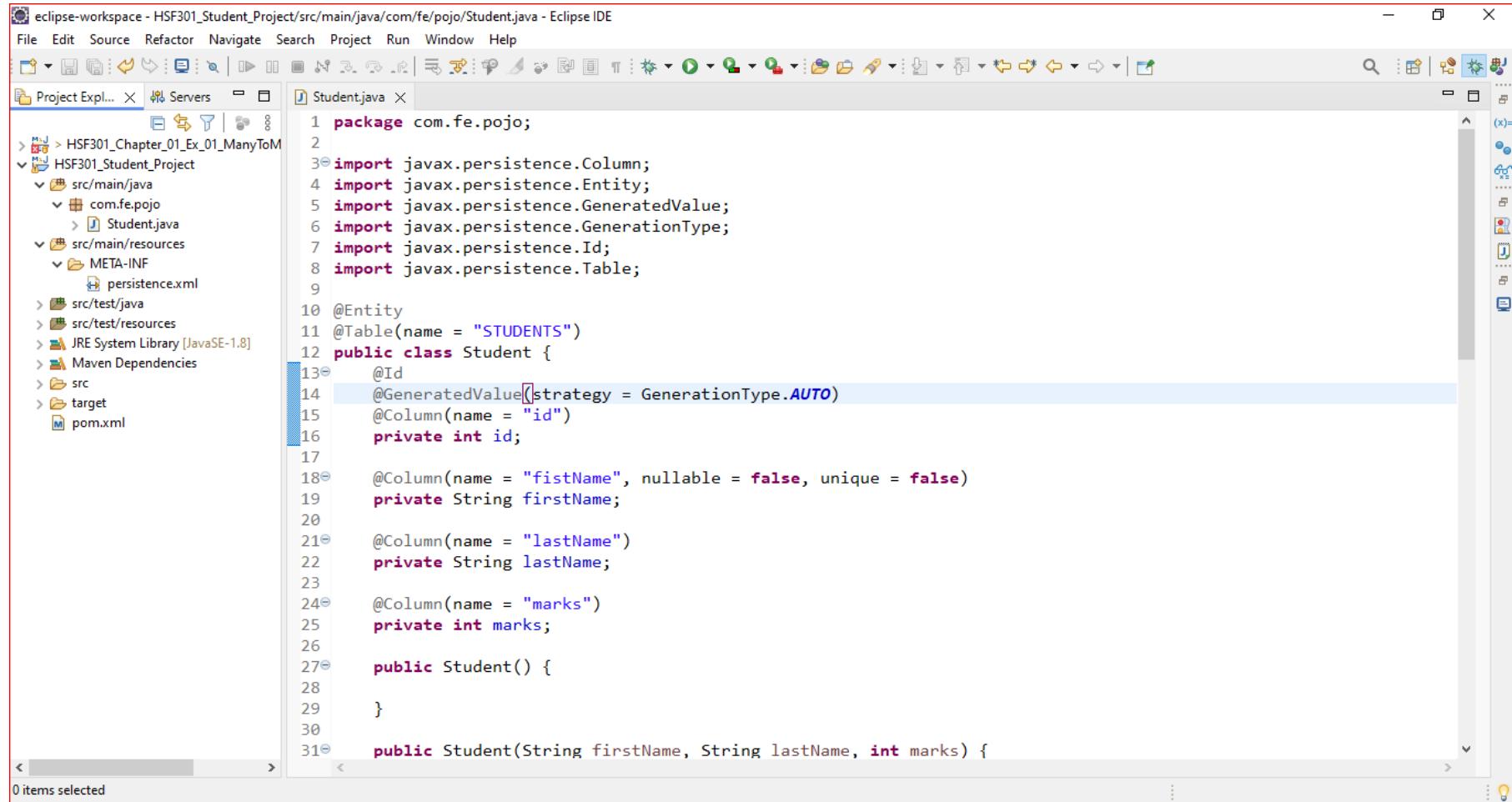
# 10. Add com.fe.pojo Package in src/main/java



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



# 12. Edit the Student.java

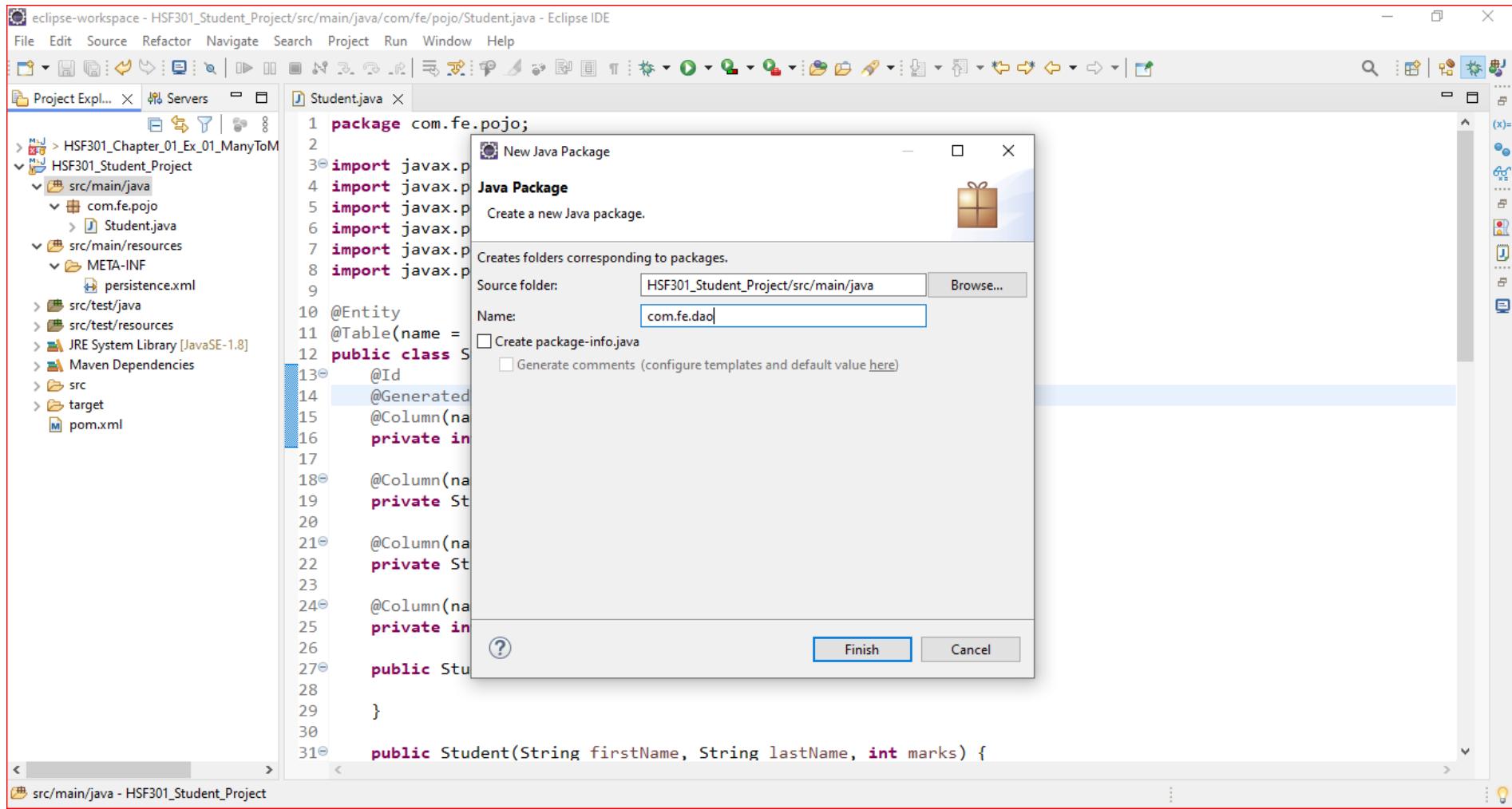


The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - HSF301\_Student\_Project/src/main/java/com/fe/pojo/Student.java - Eclipse IDE". The left sidebar displays the project structure for "HSF301\_Student\_Project" with "Student.java" selected. The main editor window shows the following Java code:

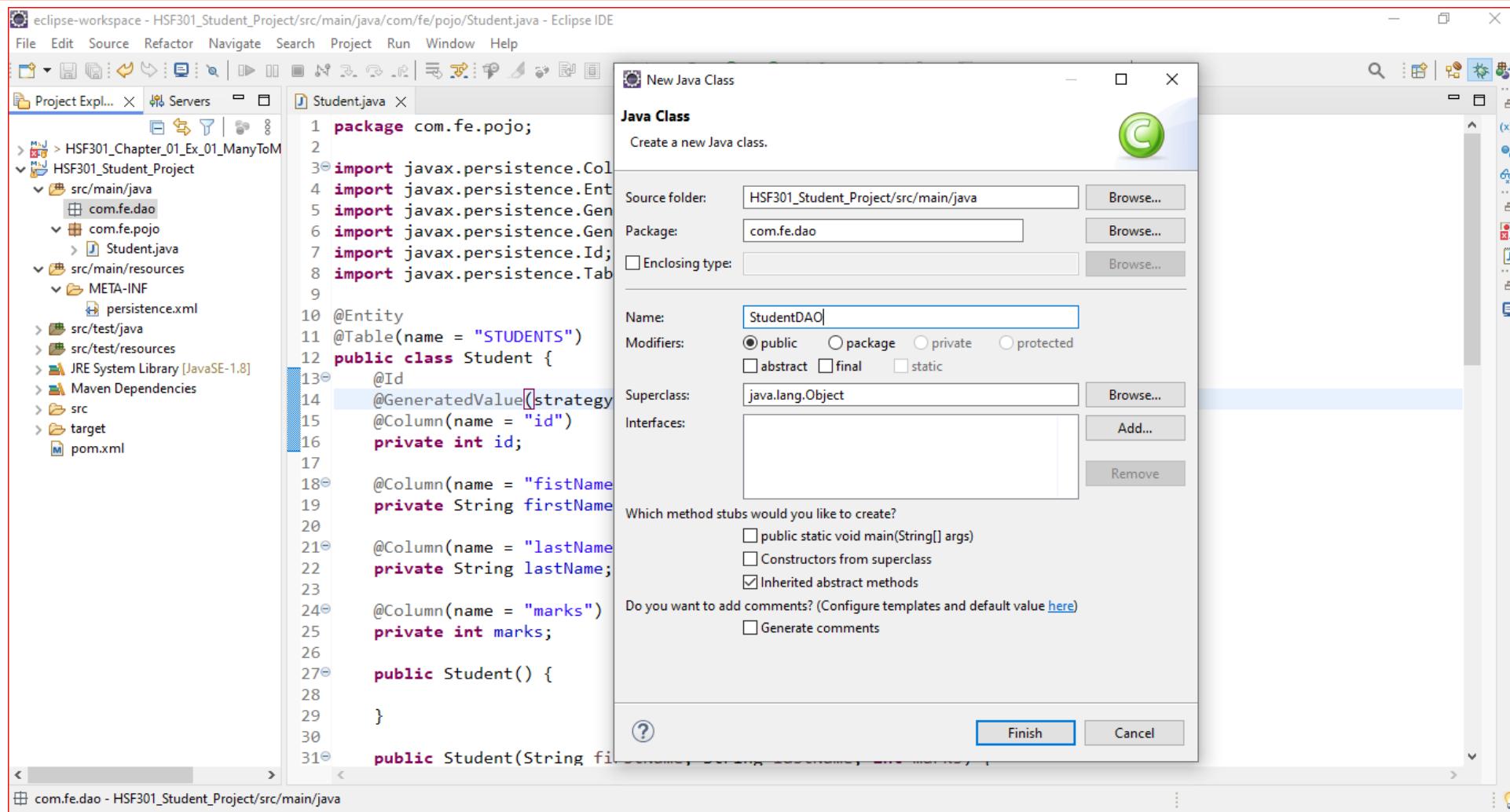
```
1 package com.fe.pojo;
2
3 import javax.persistence.Column;
4 import javax.persistence.Entity;
5 import javax.persistence.GeneratedValue;
6 import javax.persistence.GenerationType;
7 import javax.persistence.Id;
8 import javax.persistence.Table;
9
10 @Entity
11 @Table(name = "STUDENTS")
12 public class Student {
13     @Id
14     @GeneratedValue(strategy = GenerationType.AUTO)
15     @Column(name = "id")
16     private int id;
17
18     @Column(name = "firstName", nullable = false, unique = false)
19     private String firstName;
20
21     @Column(name = "lastName")
22     private String lastName;
23
24     @Column(name = "marks")
25     private int marks;
26
27     public Student() {
28
29     }
30
31     public Student(String firstName, String lastName, int marks) {
```

The code editor highlights the `@GeneratedValue(strategy = GenerationType.AUTO)` annotation with a blue background.

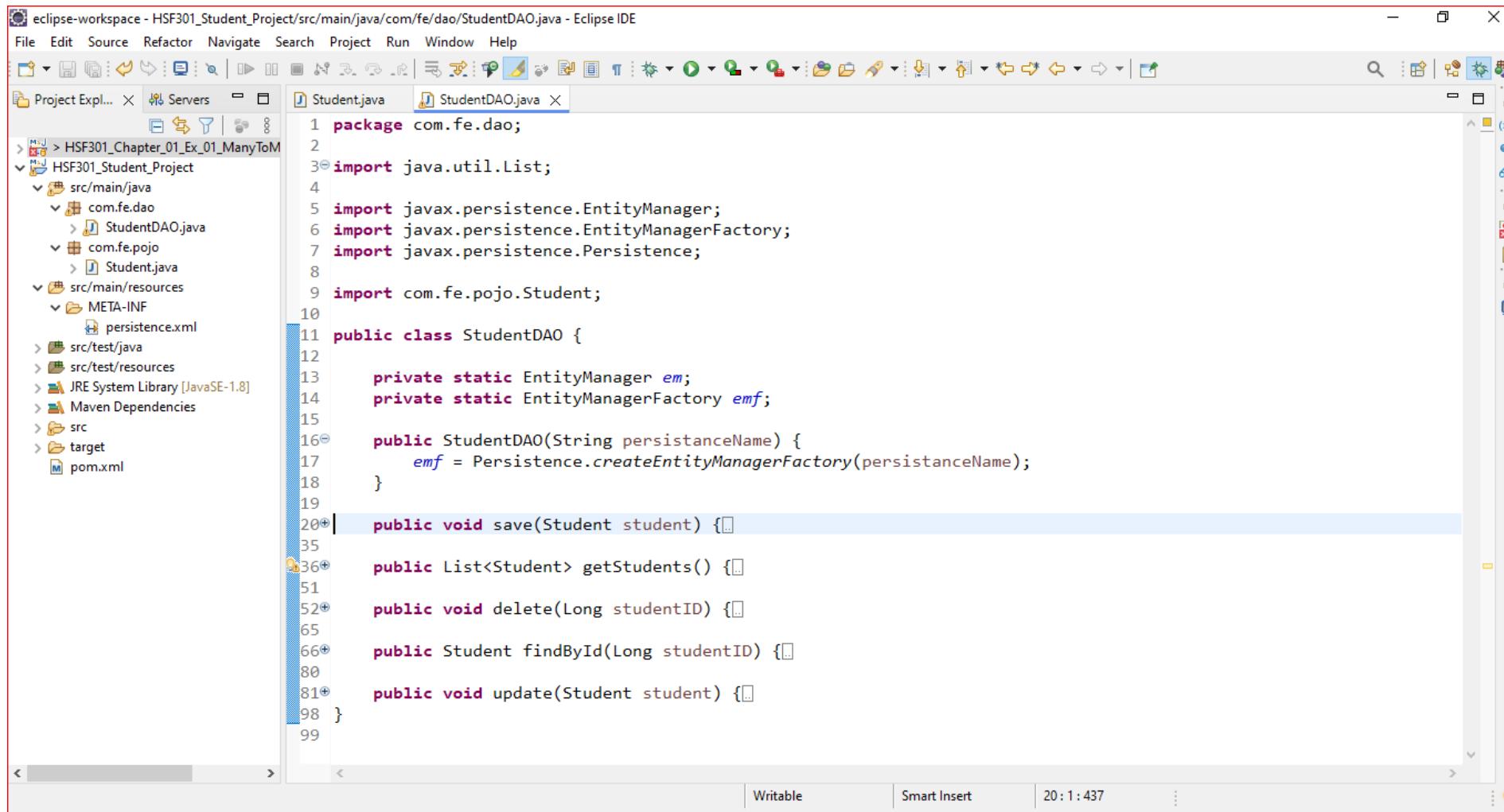
# 13. Add com.fe.dao Package in src/main/java



# 14. Create StudentDAO in com.fe.dao



# 15. Edit the StudentDAO in com.fe.dao



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HSF301\_Student\_Project/src/main/java/com/fe/dao/StudentDAO.java - Eclipse IDE
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbars:** Standard, Java, Database, XML, Web, CSS, JavaScript, CSS3, HTML, JSP, JSTL, JBoss Seam, JBoss Seam 3, JBoss Seam 4, JBoss Seam 5, JBoss Seam 6, JBoss Seam 7, JBoss Seam 8, JBoss Seam 9, JBoss Seam 10, JBoss Seam 11, JBoss Seam 12, JBoss Seam 13, JBoss Seam 14, JBoss Seam 15, JBoss Seam 16, JBoss Seam 17, JBoss Seam 18, JBoss Seam 19, JBoss Seam 20, JBoss Seam 21, JBoss Seam 22, JBoss Seam 23, JBoss Seam 24, JBoss Seam 25, JBoss Seam 26, JBoss Seam 27, JBoss Seam 28, JBoss Seam 29, JBoss Seam 30, JBoss Seam 31, JBoss Seam 32, JBoss Seam 33, JBoss Seam 34, JBoss Seam 35, JBoss Seam 36, JBoss Seam 37, JBoss Seam 38, JBoss Seam 39, JBoss Seam 40, JBoss Seam 41, JBoss Seam 42, JBoss Seam 43, JBoss Seam 44, JBoss Seam 45, JBoss Seam 46, JBoss Seam 47, JBoss Seam 48, JBoss Seam 49, JBoss Seam 50, JBoss Seam 51, JBoss Seam 52, JBoss Seam 53, JBoss Seam 54, JBoss Seam 55, JBoss Seam 56, JBoss Seam 57, JBoss Seam 58, JBoss Seam 59, JBoss Seam 60, JBoss Seam 61, JBoss Seam 62, JBoss Seam 63, JBoss Seam 64, JBoss Seam 65, JBoss Seam 66, JBoss Seam 67, JBoss Seam 68, JBoss Seam 69, JBoss Seam 70, JBoss Seam 71, JBoss Seam 72, JBoss Seam 73, JBoss Seam 74, JBoss Seam 75, JBoss Seam 76, JBoss Seam 77, JBoss Seam 78, JBoss Seam 79, JBoss Seam 80, JBoss Seam 81, JBoss Seam 82, JBoss Seam 83, JBoss Seam 84, JBoss Seam 85, JBoss Seam 86, JBoss Seam 87, JBoss Seam 88, JBoss Seam 89, JBoss Seam 90, JBoss Seam 91, JBoss Seam 92, JBoss Seam 93, JBoss Seam 94, JBoss Seam 95, JBoss Seam 96, JBoss Seam 97, JBoss Seam 98, JBoss Seam 99.
- Project Explorer:** Shows the project structure with packages like HSF301\_Chapter\_01\_Ex\_01\_ManyToMany, HSF301\_Student\_Project, and com.fe.dao containing StudentDAO.java and Student.java.
- Code Editor:** The StudentDAO.java file is open, showing Java code for a DAO class. The code includes imports for EntityManager, EntityManagerFactory, Persistence, and Student, and defines a static EntityManager em and EntityManagerFactory emf. It contains methods for save, getStudents, delete, findById, and update.
- Bottom Status Bar:** Writable, Smart Insert, 20:1:437

## 16. Save Student Method

The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - HSF301\_Student\_Project/src/main/java/com/fe/dao/StudentDAO.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, etc. The left sidebar is the "Project Explorer" showing the project structure:

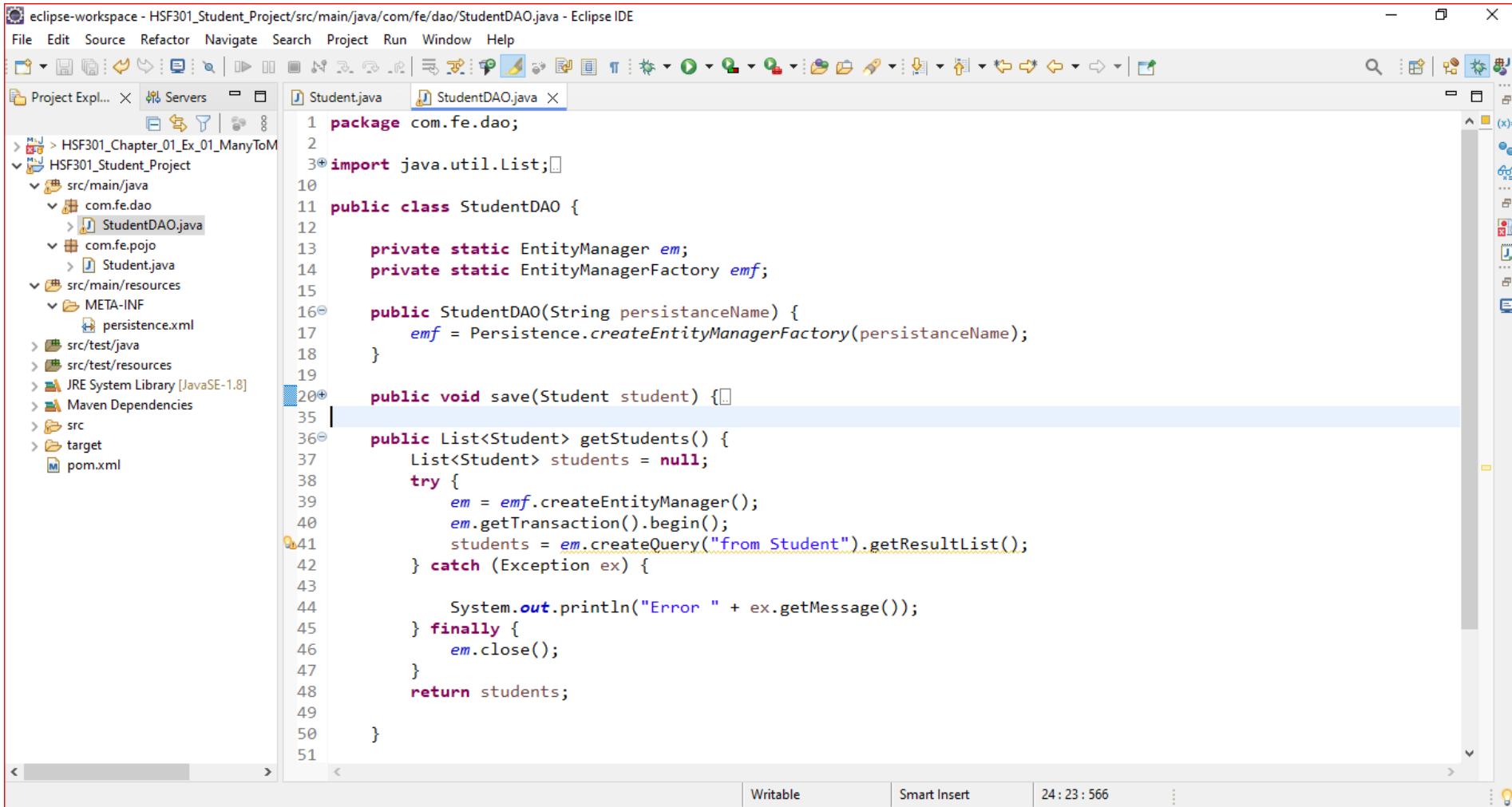
- HSF301 Chapter\_01\_Ex\_01\_ManyToM
- HSF301\_Student\_Project
- src/main/java
  - com.fe.dao
    - StudentDAO.java
  - com.fe.pojo
    - Student.java
- src/main/resources
  - META-INF
- src/test/java
- src/test/resources
- JRE System Library [JavaSE-1.8]
- Maven Dependencies
- src
- target
- pom.xml

The main editor window displays the code for `StudentDAO.java`:

```
3+import java.util.List;
10
11 public class StudentDAO {
12
13     private static EntityManager em;
14     private static EntityManagerFactory emf;
15
16     public StudentDAO(String persistanceName) {
17         emf = Persistence.createEntityManagerFactory(persistanceName);
18     }
19
20     public void save(Student student) {
21         try {
22             em = emf.createEntityManager();
23             em.getTransaction().begin();
24             em.persist(student);
25             em.getTransaction().commit();
26         } catch (Exception ex) {
27             em.getTransaction().rollback();
28             System.out.println("Error " + ex.getMessage());
29         } finally {
30             em.close();
31         }
32     }
33
34 }
35
36+ public List<Student> getStudents() { }
51
52+ public void delete(int studentID) { }
```

The code editor shows syntax highlighting for Java keywords and comments. A blue selection bar highlights the line `em.persist(student);`. The status bar at the bottom indicates "Writable", "Smart Insert", and the current position "24: 23 : 566".

# 17. Get All Students Method



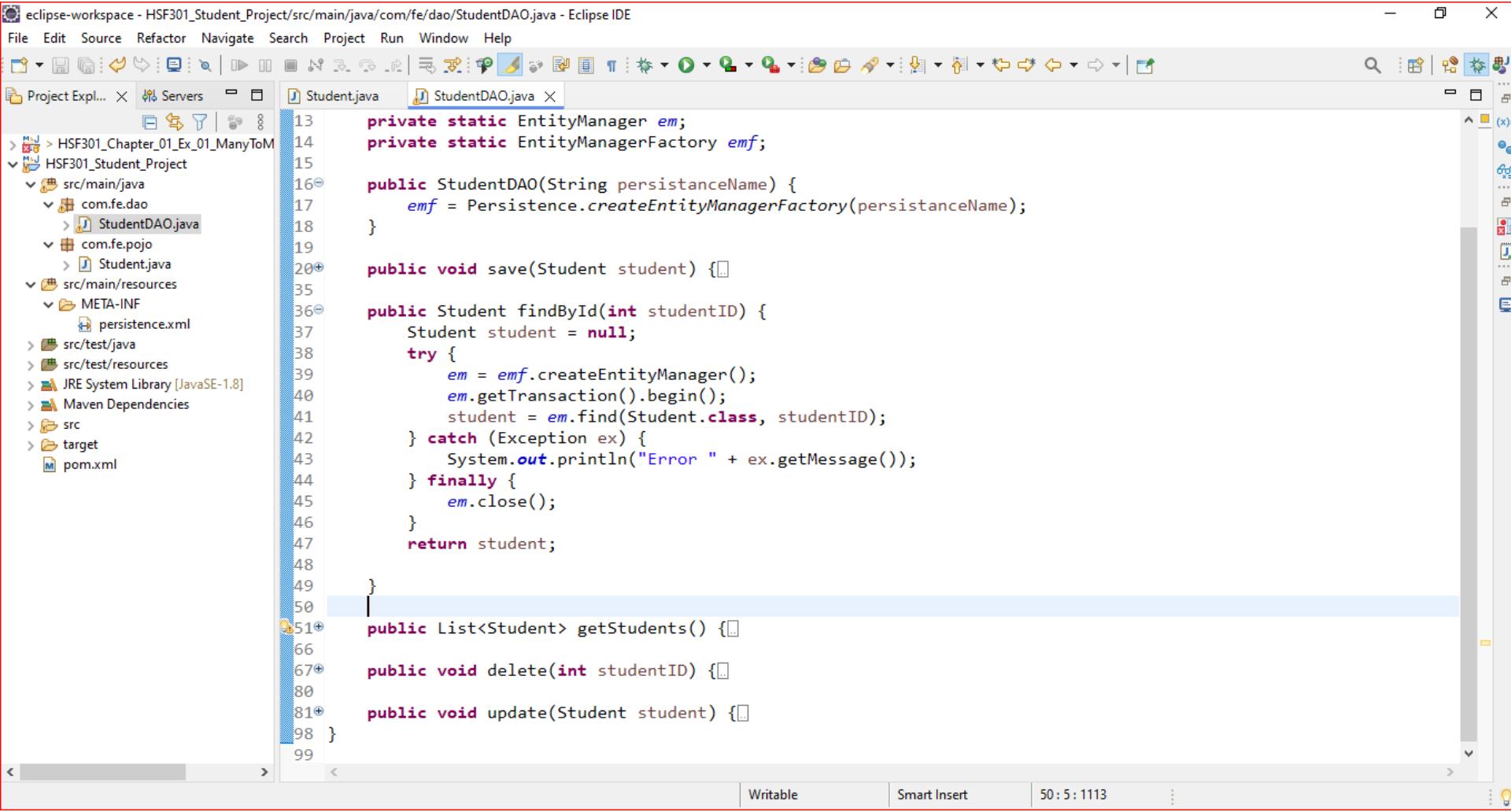
The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HSF301\_Student\_Project/src/main/java/com/fe/dao/StudentDAO.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbars:** Standard, Search, Navigator, Outline, Problems, Properties, Status Bar.
- Left Sidebar:** Project Explorer (HSF301\_Chapter\_01\_Ex\_01\_ManyToM, HSF301\_Student\_Project, src/main/java, com.fe.dao, StudentDAO.java, com.fe.pojo, Student.java, src/main/resources, META-INF, persistence.xml, src/test/java, src/test/resources, JRE System Library [JavaSE-1.8], Maven Dependencies, src, target, pom.xml).
- Central Area:** Editor tab for StudentDAO.java. The code is as follows:

```
1 package com.fe.dao;
2
3 import java.util.List;
4
5 public class StudentDAO {
6
7     private static EntityManager em;
8     private static EntityManagerFactory emf;
9
10    public StudentDAO(String persistanceName) {
11        emf = Persistence.createEntityManagerFactory(persistanceName);
12    }
13
14    public void save(Student student) {
15
16    }
17
18    public List<Student> getStudents() {
19        List<Student> students = null;
20        try {
21            em = emf.createEntityManager();
22            em.getTransaction().begin();
23            students = em.createQuery("from Student").getResultList();
24        } catch (Exception ex) {
25
26            System.out.println("Error " + ex.getMessage());
27        } finally {
28            em.close();
29        }
30        return students;
31    }
32}
```

The code implements a DAO (Data Access Object) for the Student entity. It uses EntityManager and EntityManagerFactory from the Persistence API to manage database transactions and queries. The `getStudents()` method retrieves all student records from the database.

# 18. Find a Student Method



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - HSF301\_Student\_Project/src/main/java/com/fe/dao/StudentDAO.java - Eclipse IDE". The left side features the Project Explorer with the project structure: HSF301\_Student\_Project, src/main/java, com.fe.dao, StudentDAO.java, com.fe.pojo, Student.java, and src/main/resources/META-INF/persistence.xml. The right side shows the code editor with the StudentDAO.java file open. The code implements a DAO interface for managing Student entities using EntityManager and EntityManagerFactory from Persistence.

```
private static EntityManager em;
private static EntityManagerFactory emf;

public StudentDAO(String persistanceName) {
    emf = Persistence.createEntityManagerFactory(persistanceName);
}

public void save(Student student) {}

public Student findById(int studentID) {
    Student student = null;
    try {
        em = emf.createEntityManager();
        em.getTransaction().begin();
        student = em.find(Student.class, studentID);
    } catch (Exception ex) {
        System.out.println("Error " + ex.getMessage());
    } finally {
        em.close();
    }
    return student;
}

public List<Student> getStudents() {}

public void delete(int studentID) {}

public void update(Student student) {}

}
```

## 19. Delete Student Method

The screenshot shows the Eclipse IDE interface with the title "eclipse-workspace - HSF301\_Student\_Project/src/main/java/com/fe/dao/StudentDAO.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, etc. The left sidebar is the "Project Explorer" showing the project structure:

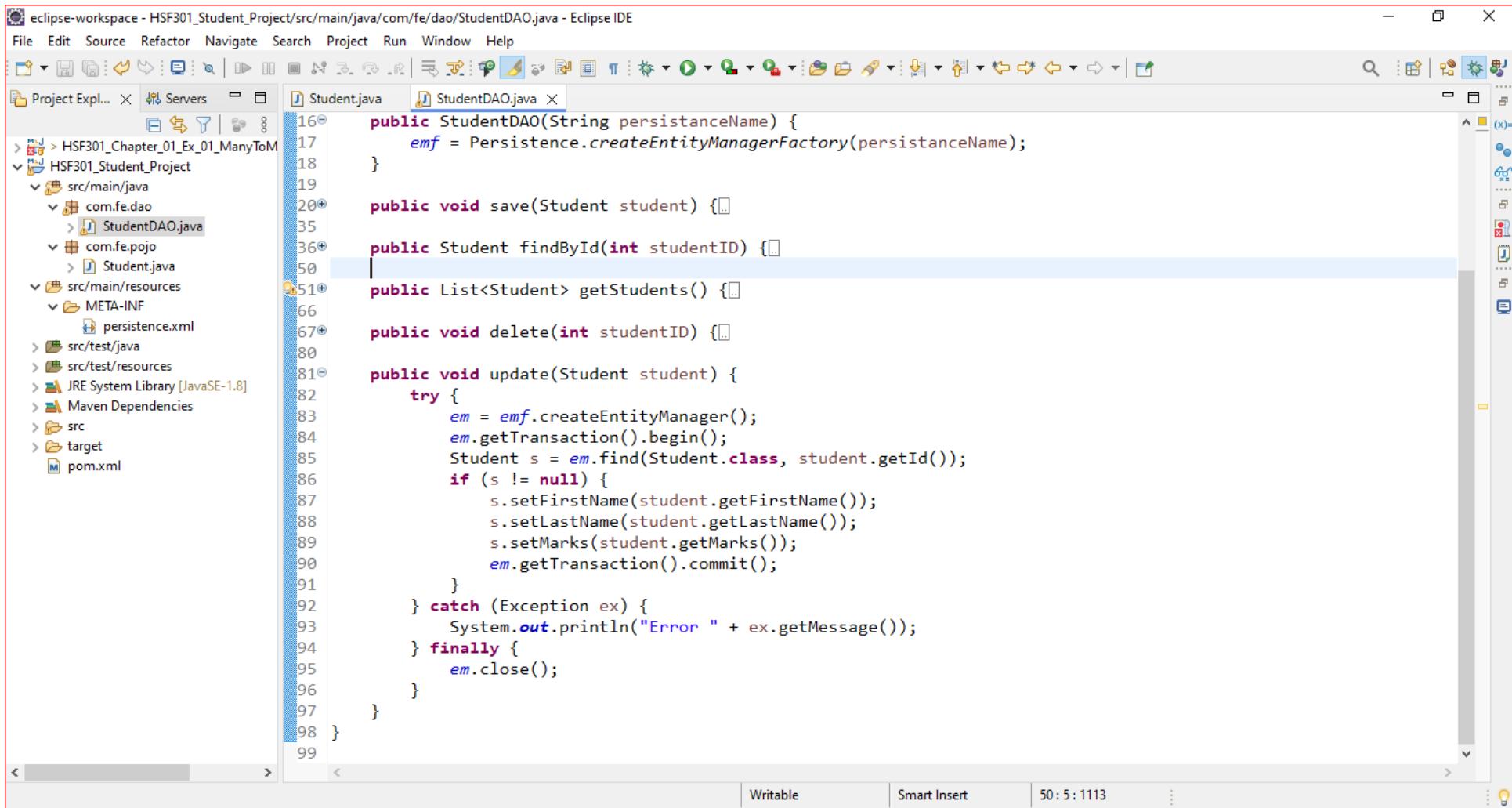
- HSF301\_Chapter\_01\_Ex\_01\_ManyToM
- HSF301\_Student\_Project
- src/main/java
  - com.fe.dao
    - StudentDAO.java
  - com.fe.pojo
    - Student.java
- src/main/resources
  - META-INF
    - persistence.xml
- src/test/java
- src/test/resources
- JRE System Library [JavaSE-1.8]
- Maven Dependencies
- src
- target
- pom.xml

The main editor window displays the code for `StudentDAO.java`:

```
12     private static EntityManager em;
13     private static EntityManagerFactory emf;
14
15     public StudentDAO(String persistanceName) {
16         emf = Persistence.createEntityManagerFactory(persistanceName);
17     }
18
19     public void save(Student student) {}
20
21     public Student findById(int studentID) {}
22
23     public List<Student> getStudents() {}
24
25     public void delete(int studentID) {
26         try {
27             em = emf.createEntityManager();
28             em.getTransaction().begin();
29             Student s = em.find(Student.class, studentID);
30             em.remove(s);
31             em.getTransaction().commit();
32         } catch (Exception ex) {
33             System.out.println("Error " + ex.getMessage());
34         } finally {
35             em.close();
36         }
37     }
38
39     public void update(Student student) {}
40 }
```

The status bar at the bottom indicates "Writable" and "Smart Insert" modes, and the current line is "50:5:1113".

# 20. Update Student Method



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - HSF301\_Student\_Project/src/main/java/com/fe/dao/StudentDAO.java - Eclipse IDE". The left side features the Project Explorer and Navigator. The main area displays the code for `StudentDAO.java`. The code implements methods for persisting and updating student data using Entity Manager.

```
public StudentDAO(String persistanceName) {
    emf = Persistence.createEntityManagerFactory(persistanceName);
}

public void save(Student student) {}

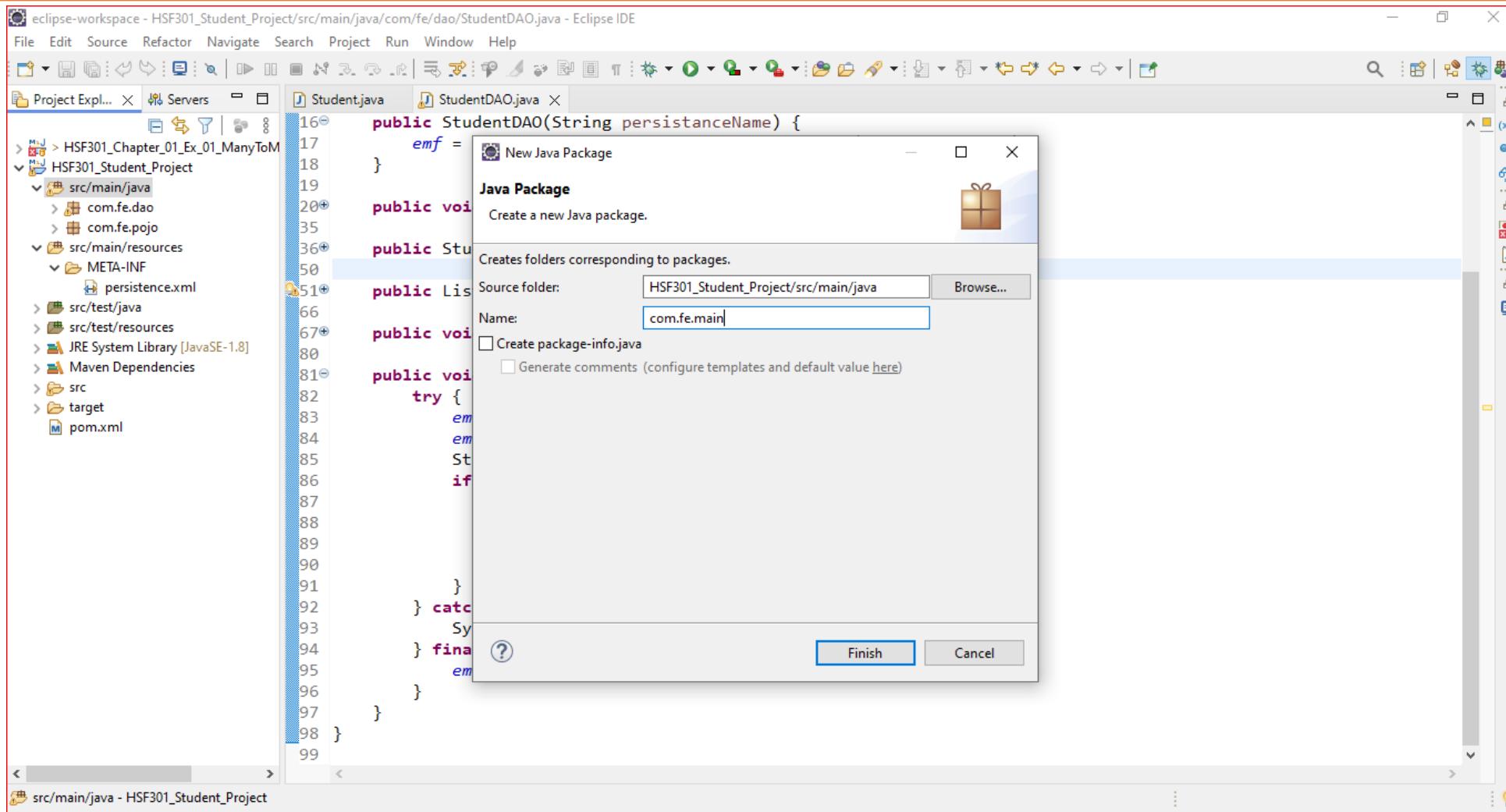
public Student findById(int studentID) {}

public List<Student> getStudents() {}

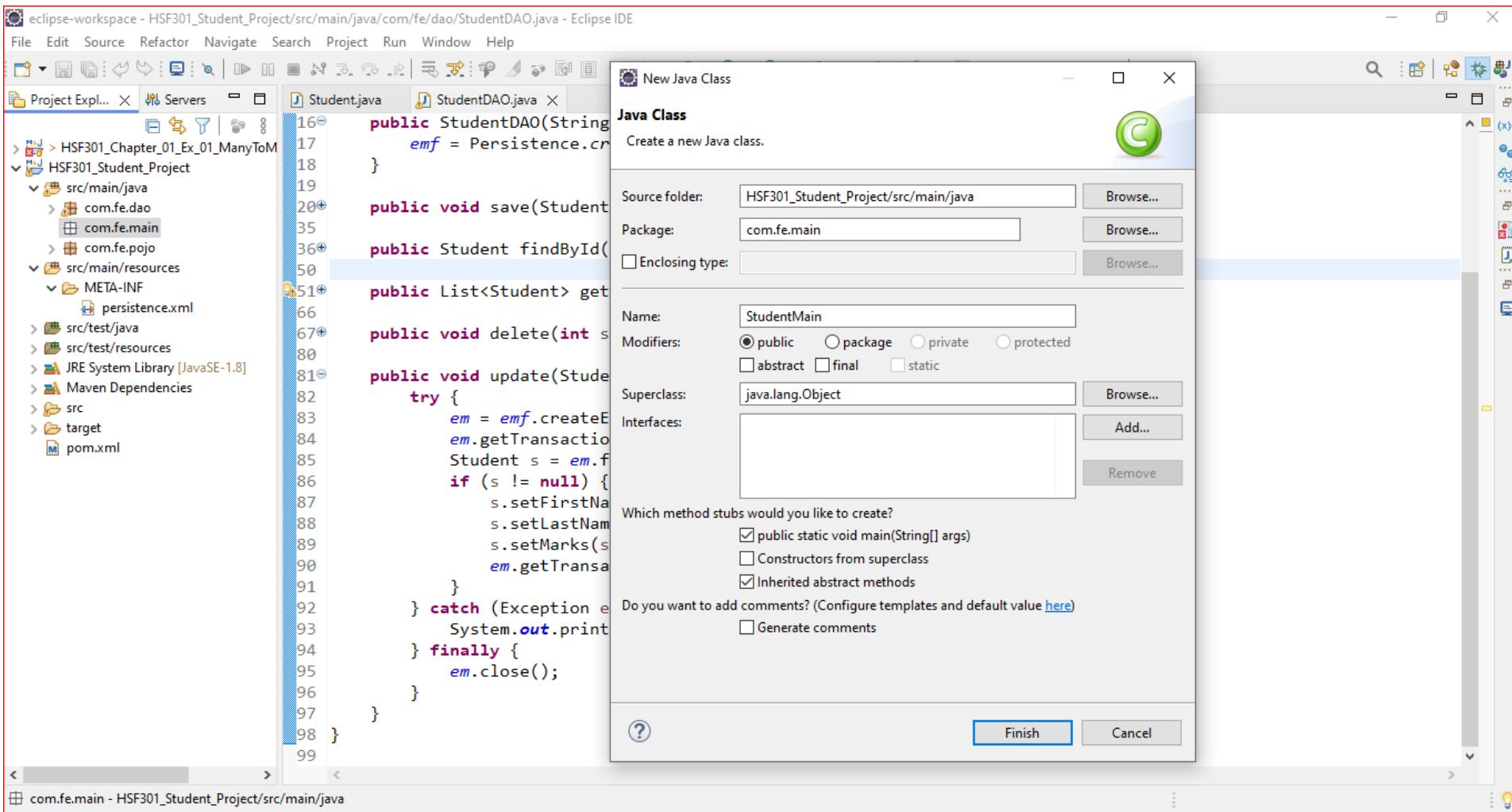
public void delete(int studentID) {}

public void update(Student student) {
    try {
        em = emf.createEntityManager();
        em.getTransaction().begin();
        Student s = em.find(Student.class, student.getId());
        if (s != null) {
            s.setFirstName(student.getFirstName());
            s.setLastName(student.getLastName());
            s.setMarks(student.getMarks());
            em.getTransaction().commit();
        }
    } catch (Exception ex) {
        System.out.println("Error " + ex.getMessage());
    } finally {
        em.close();
    }
}
```

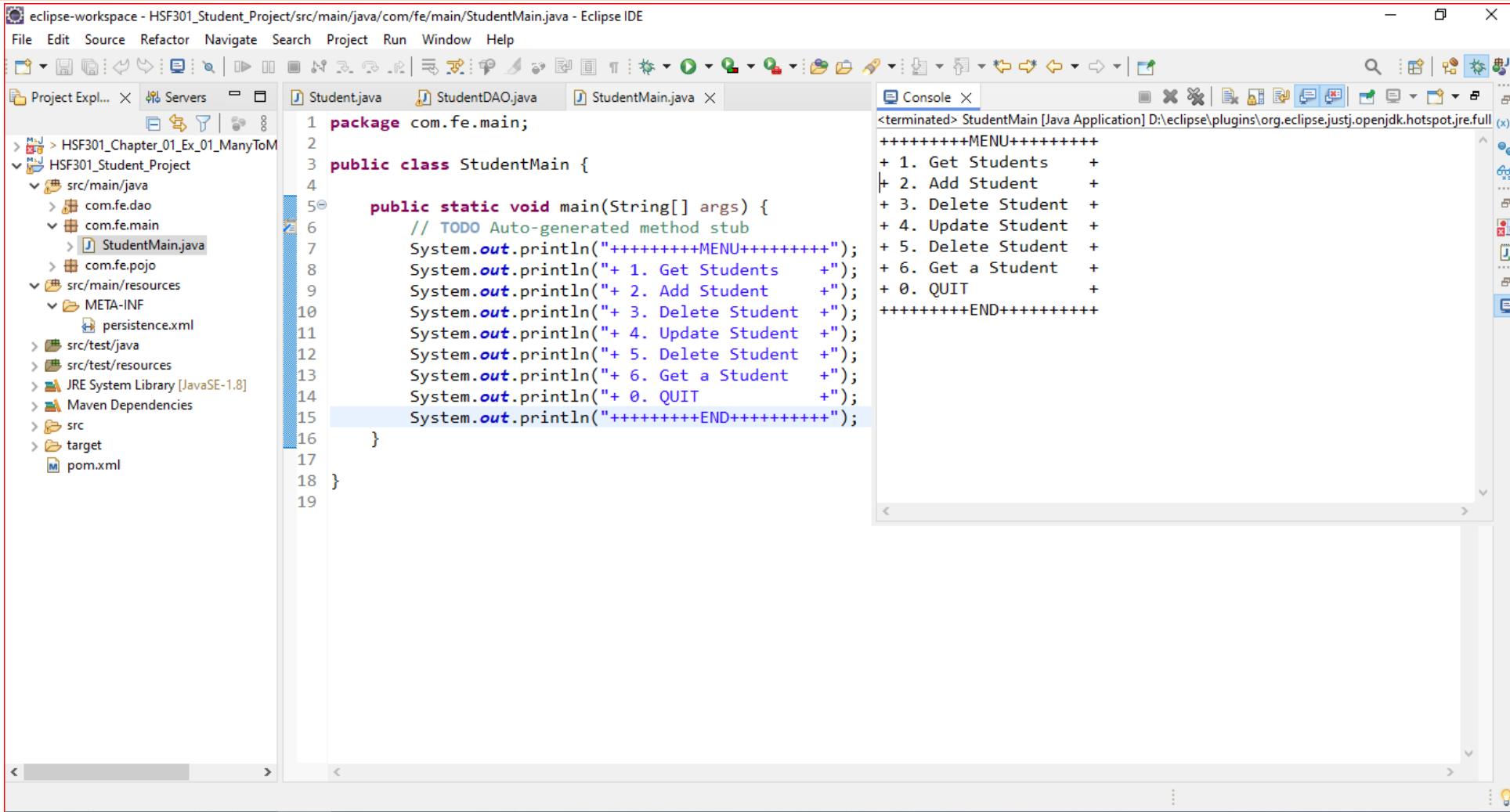
## 21. Add com.fe.main Package in src/main/java



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



## 23. Edit the StudentMain in com.fe.main



The screenshot shows the Eclipse IDE interface with the following details:

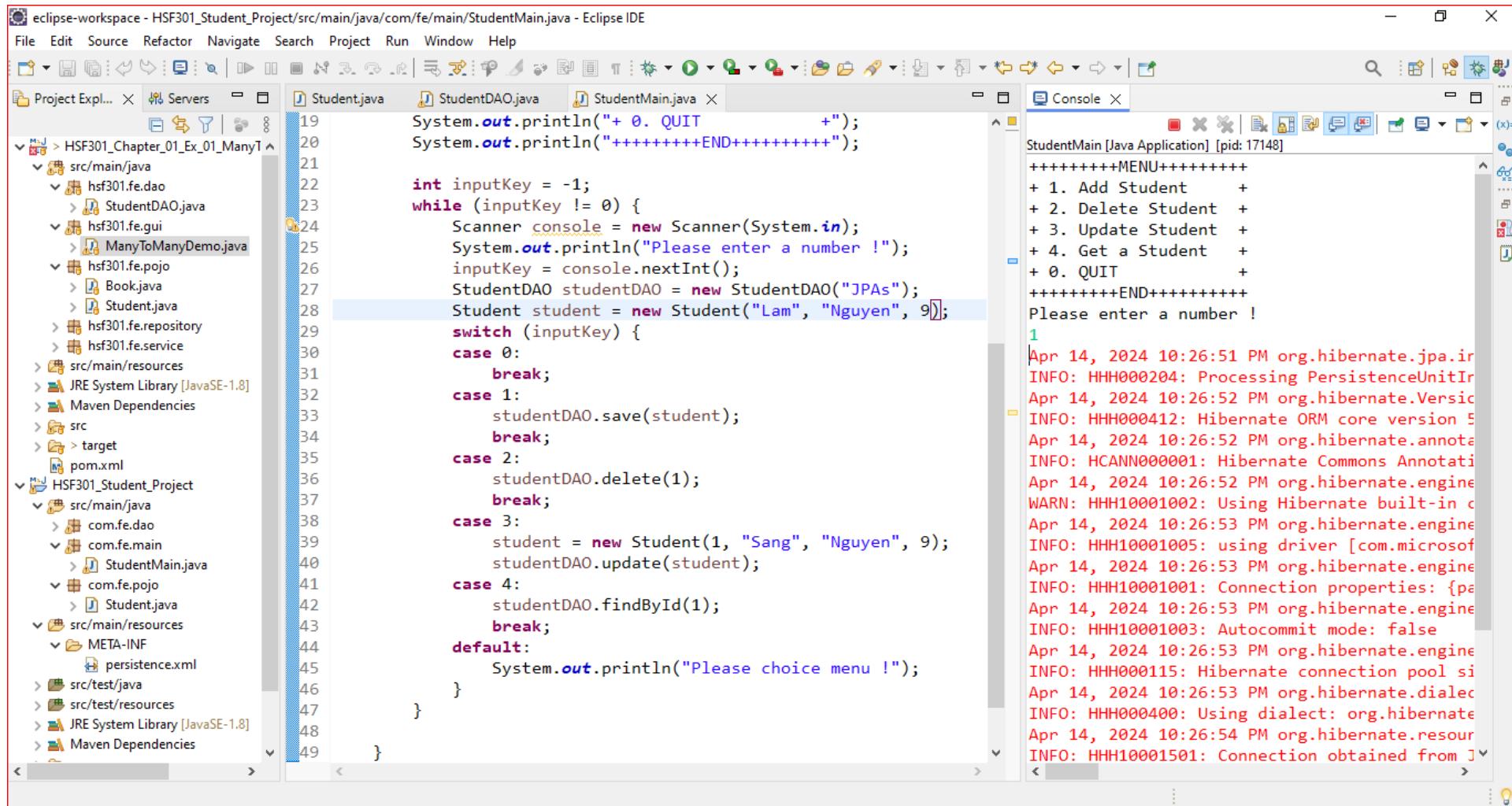
- Project Explorer:** Shows the project structure: HSF301 Chapter\_01\_Ex\_01\_ManyToM, HSF301\_Student\_Project, src/main/java (containing com.fe.dao, com.fe.main, com.fe.pojo), src/main/resources (containing META-INF/persistence.xml), src/test/java, src/test/resources, JRE System Library [JavaSE-1.8], Maven Dependencies, src, target, and pom.xml).
- Editor:** The StudentMain.java file is open. The code defines a public class StudentMain with a main method that prints a menu of operations (Get Students, Add Student, Delete Student, Update Student, Get a Student, QUIT) to the console.
- Console:** The output window shows the printed menu options.

```
package com.fe.main;

public class StudentMain {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("++++++MENU++++++");
        System.out.println("+ 1. Get Students +");
        System.out.println("+ 2. Add Student +");
        System.out.println("+ 3. Delete Student +");
        System.out.println("+ 4. Update Student +");
        System.out.println("+ 5. Delete Student +");
        System.out.println("+ 6. Get a Student +");
        System.out.println("+ 0. QUIT +");
        System.out.println("++++++END++++++");
    }
}
```

## 24. Edit the StudentMain in com.fe.main



The screenshot shows the Eclipse IDE interface with the following details:

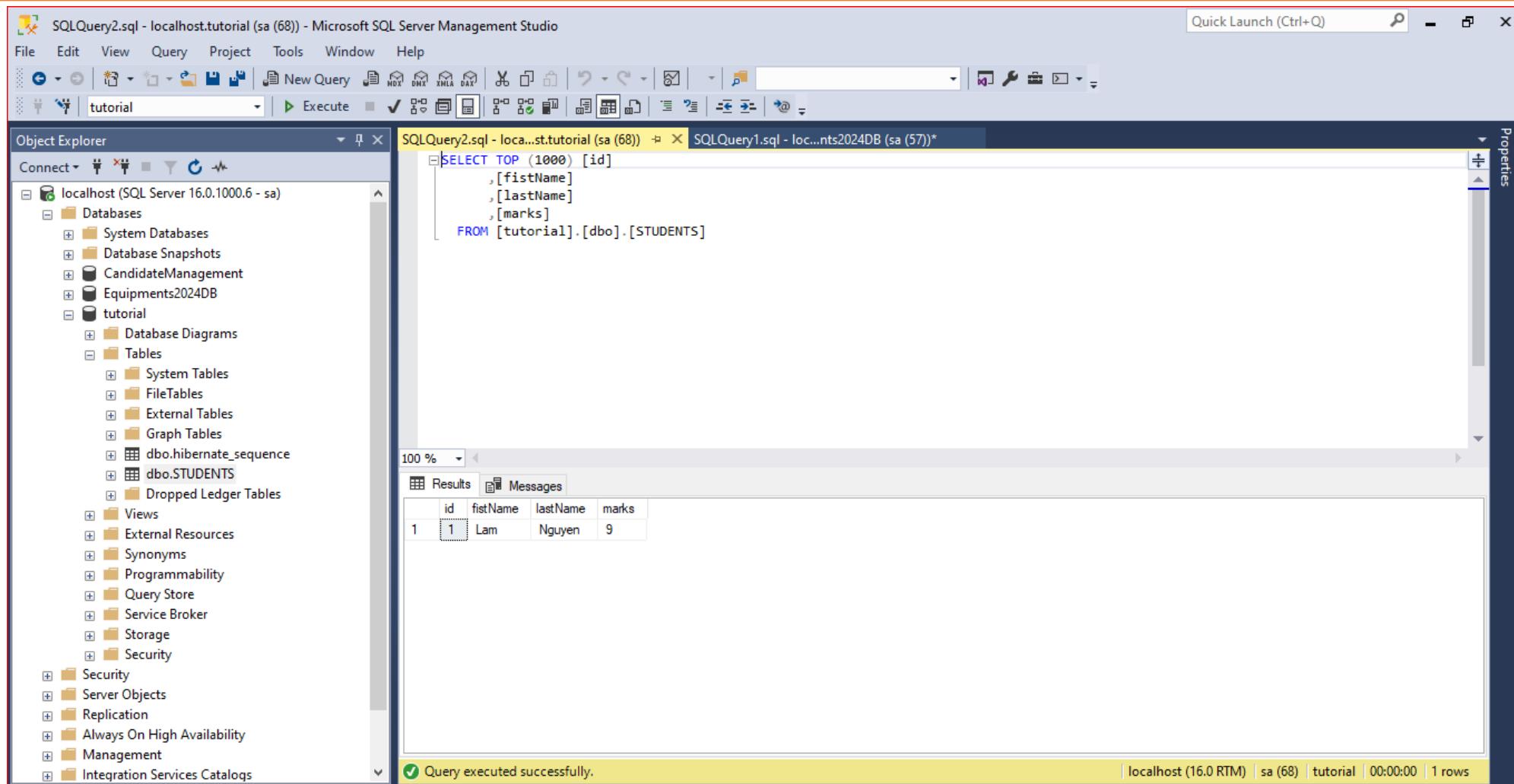
- Project Explorer:** Shows the project structure under "HSF301\_Chapter\_01\_Ex\_01\_ManyToMany".
- Editor:** Displays the code for `StudentMain.java`. The code implements a menu system for managing students using a `StudentDAO`.
- Console View:** Shows the output of the application's execution. It prints the menu options and prompts for user input. The application successfully adds a new student ("Lam", "Nguyen") and updates an existing student ("Sang", "Nguyen").

```
System.out.println("+ 0. QUIT      +");
System.out.println("++++++END++++++");

int inputKey = -1;
while (inputKey != 0) {
    Scanner console = new Scanner(System.in);
    System.out.println("Please enter a number !");
    inputKey = console.nextInt();
    StudentDAO studentDAO = new StudentDAO("JPAs");
    Student student = new Student("Lam", "Nguyen", 9);
    switch (inputKey) {
        case 0:
            break;
        case 1:
            studentDAO.save(student);
            break;
        case 2:
            studentDAO.delete(1);
            break;
        case 3:
            student = new Student(1, "Sang", "Nguyen", 9);
            studentDAO.update(student);
        case 4:
            studentDAO.findById(1);
            break;
        default:
            System.out.println("Please choice menu !");
    }
}

++++++MENU++++++
+ 1. Add Student +
+ 2. Delete Student +
+ 3. Update Student +
+ 4. Get a Student +
+ 0. QUIT +
++++++END+++++++
Please enter a number !
1
Apr 14, 2024 10:26:51 PM org.hibernate.jpa.ir
INFO: HHH000204: Processing PersistenceUnitIri
Apr 14, 2024 10:26:52 PM org.hibernate.Versio
INFO: HHH000412: Hibernate ORM core version S
Apr 14, 2024 10:26:52 PM org.hibernate.annota
INFO: HCANN000001: Hibernate Commons Annotati
Apr 14, 2024 10:26:52 PM org.hibernate.engine
WARN: HHH10001002: Using Hibernate built-in c
Apr 14, 2024 10:26:53 PM org.hibernate.engine
INFO: HHH10001005: using driver [com.microsof
Apr 14, 2024 10:26:53 PM org.hibernate.engine
INFO: HHH10001001: Connection properties: {pa
Apr 14, 2024 10:26:53 PM org.hibernate.engine
INFO: HHH10001003: Autocommit mode: false
Apr 14, 2024 10:26:53 PM org.hibernate.engine
INFO: HHH000115: Hibernate connection pool si
Apr 14, 2024 10:26:53 PM org.hibernate.dialect
INFO: HHH000400: Using dialect: org.hibernate
Apr 14, 2024 10:26:54 PM org.hibernate.resourc
INFO: HHH10001501: Connection obtained from J
```

# 25. Result



The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The title bar reads "SQLQuery2.sql - localhost.tutorial (sa (68)) - Microsoft SQL Server Management Studio". The left pane is the Object Explorer, showing the database structure for "localhost (SQL Server 16.0.1000.6 - sa)". The "tutorial" database is expanded, showing its tables, including "STUDENTS". The right pane contains two query panes: "SQLQuery2.sql" and "SQLQuery1.sql". The "SQLQuery2.sql" pane displays the following query:

```
SELECT TOP (1000) [id]
      ,[firstName]
      ,[lastName]
      ,[marks]
  FROM [tutorial].[dbo].[STUDENTS]
```

The results pane shows the output of the query:

	id	firstName	lastName	marks
1	1	Lam	Nguyen	9

A status bar at the bottom indicates "Query executed successfully." and provides connection information: "localhost (16.0 RTM) | sa (68) | tutorial | 00:00:00 | 1 rows".

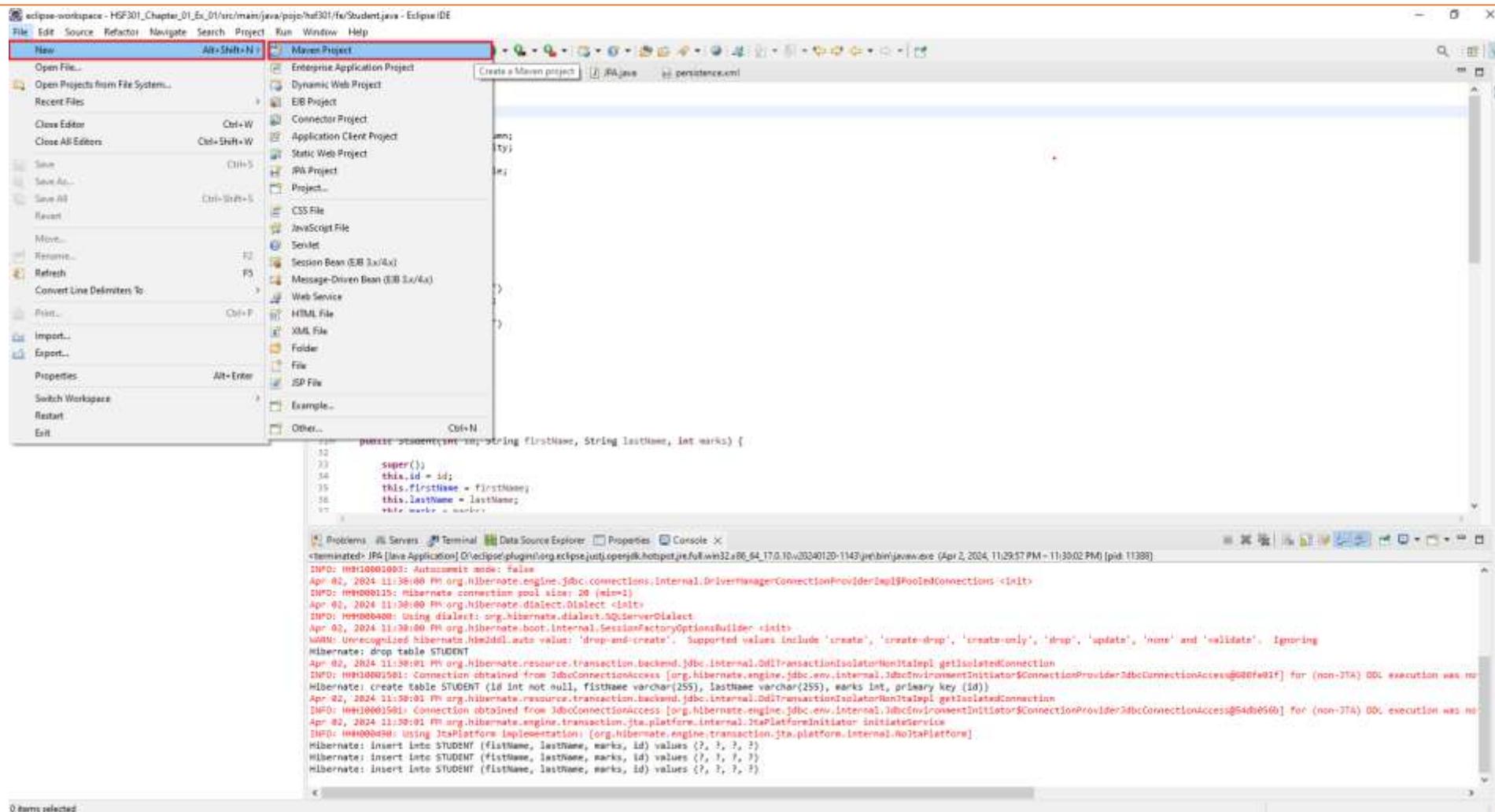
# Mapping in JPA

# Relationships Annotations in JPA

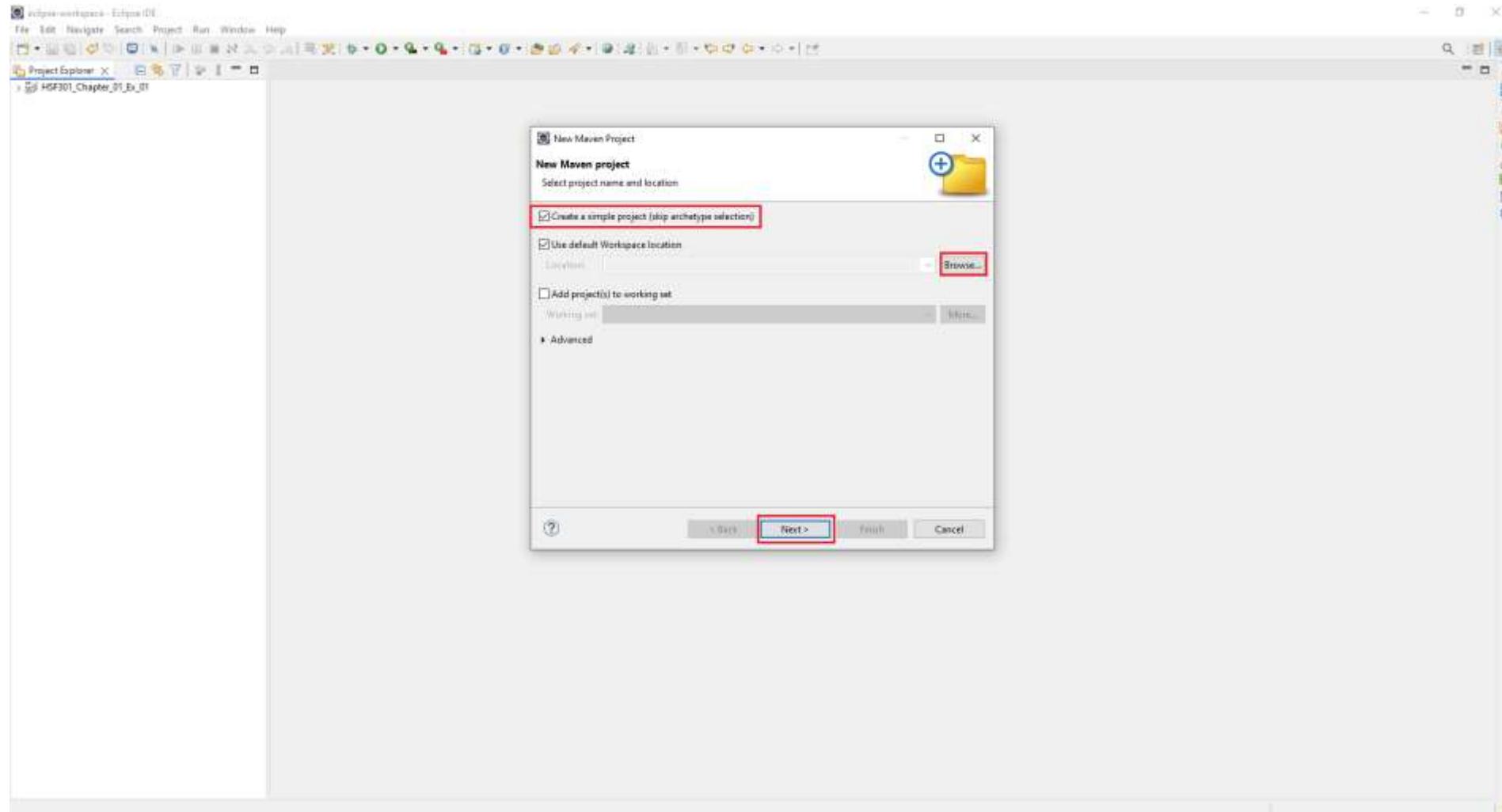
- ❖ **@ManyToOne:** This annotation defines a many-to-one relationship between two entities.
- ❖ **@OneToMany:** This annotation defines a one-to-many relationship between two entities.
- ❖ **@OneToOne:** This annotation defines a one-to-one relationship between two entities.
- ❖ **@ManyToMany:** This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

# Demo JPA (One To Many)

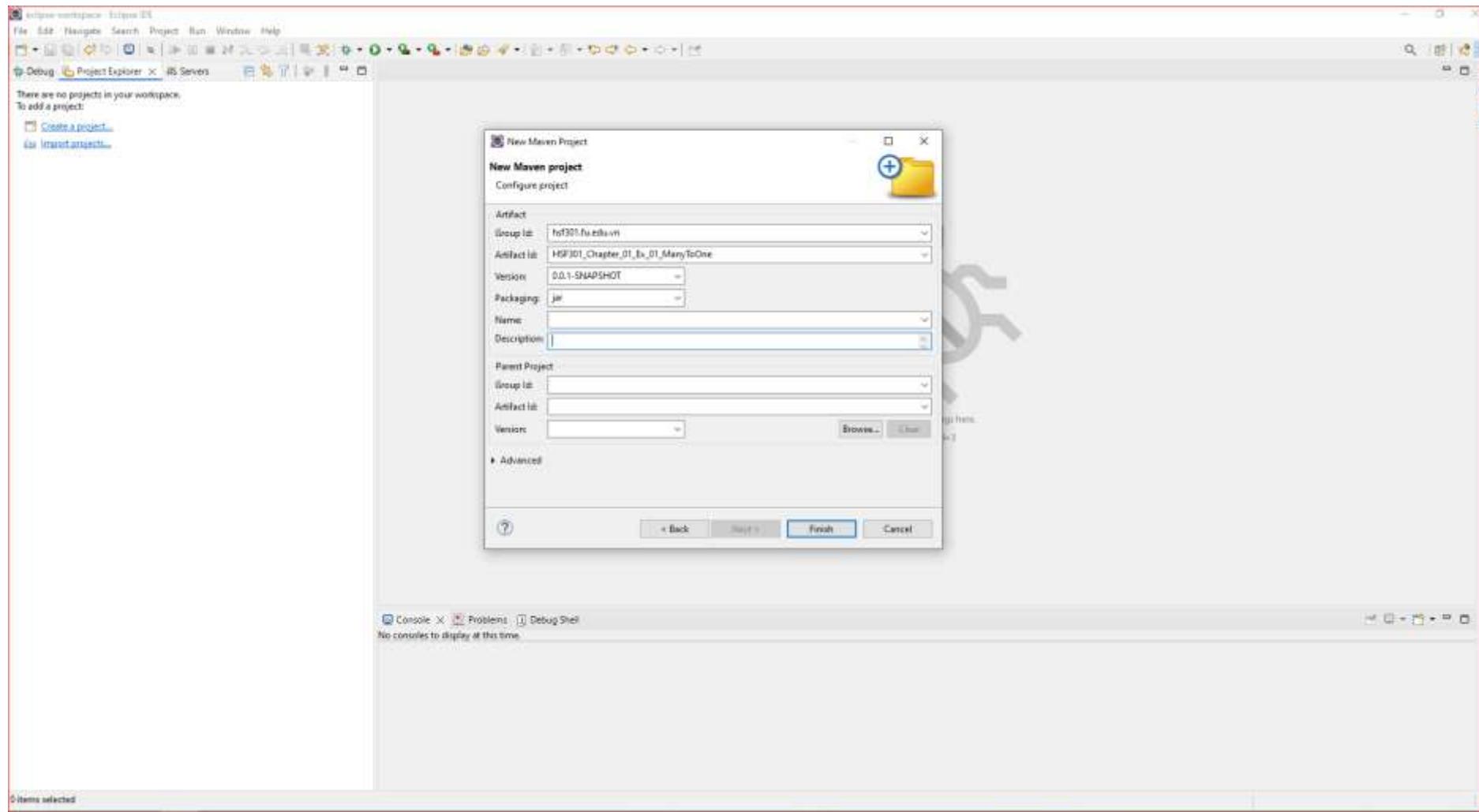
# 1. Open Eclipse, File | New | Maven Project



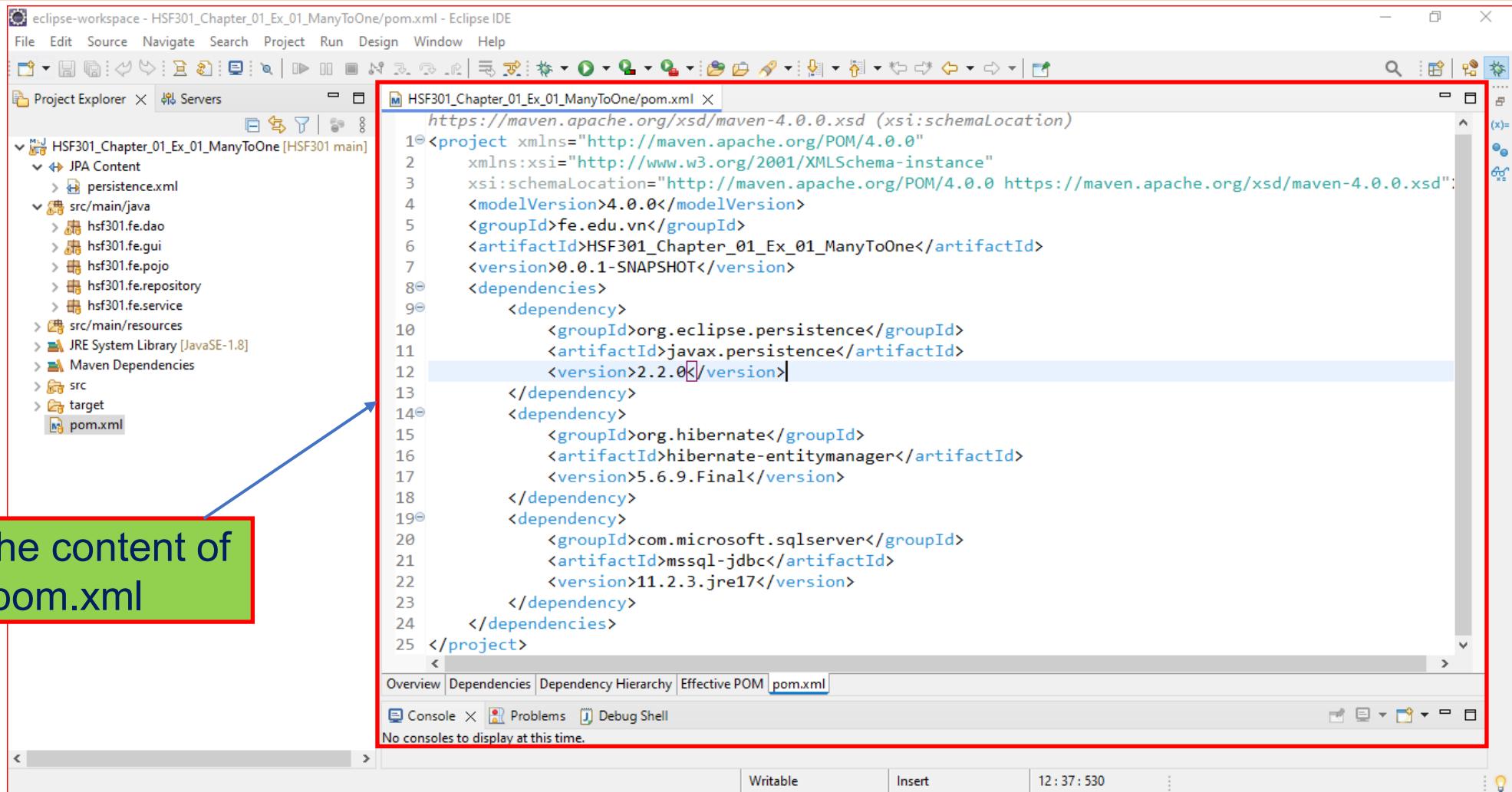
## 2. Check Create a simple project -> Browse Project -> Next



### 3. Fill the information Project -> Click Finish



## 4. Structure of Maven Project



The screenshot shows the Eclipse IDE interface with the title "eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/pom.xml - Eclipse IDE". The left side features the "Project Explorer" view, which lists the project structure:

- HSF301\_Chapter\_01\_Ex\_01\_ManyToOne [HSF301 main]
  - JPA Content
  - persistence.xml
  - src/main/java
    - hsf301.fe.dao
    - hsf301.fe.gui
    - hsf301.fe.pojo
    - hsf301.fe.repository
    - hsf301.fe.service
  - src/main/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - pom.xml

The right side displays the content of the "pom.xml" file, which is highlighted with a red border. A blue arrow points from a green box containing the text "Edit the content of pom.xml" towards the "pom.xml" tab in the bottom navigation bar.

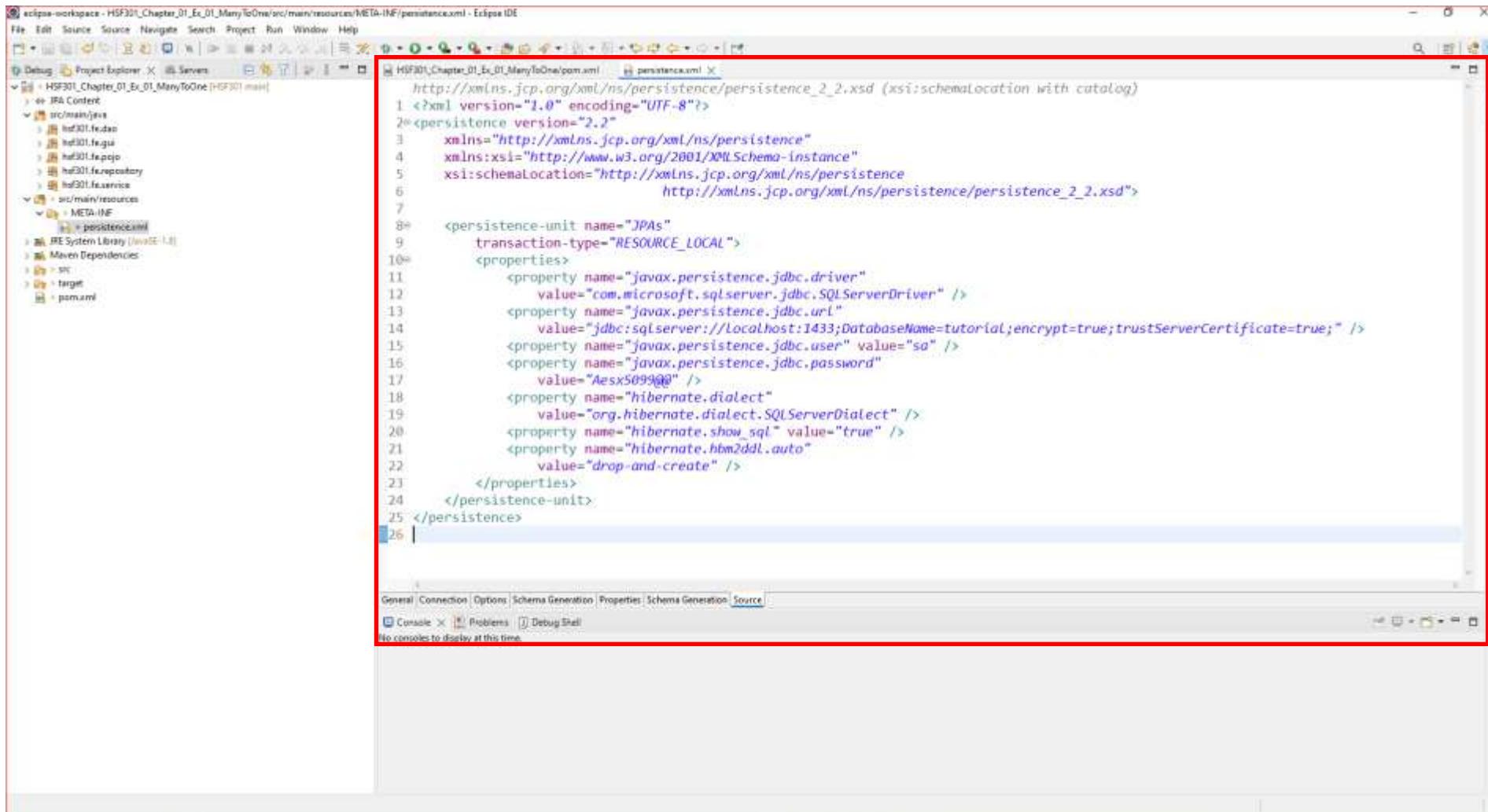
```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1<project xmlns="http://maven.apache.org/POM/4.0.0"
2  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4    <modelVersion>4.0.0</modelVersion>
5    <groupId>fe.edu.vn</groupId>
6    <artifactId>HSF301_Chapter_01_Ex_01_ManyToOne</artifactId>
7    <version>0.0.1-SNAPSHOT</version>
8    <dependencies>
9      <dependency>
10        <groupId>org.eclipse.persistence</groupId>
11        <artifactId>javax.persistence</artifactId>
12        <version>2.2.0</version>
13      </dependency>
14      <dependency>
15        <groupId>org.hibernate</groupId>
16        <artifactId>hibernate-entitymanager</artifactId>
17        <version>5.6.9.Final</version>
18      </dependency>
19      <dependency>
20        <groupId>com.microsoft.sqlserver</groupId>
21        <artifactId>mssql-jdbc</artifactId>
22        <version>11.2.3.jre17</version>
23      </dependency>
24    </dependencies>
25  </project>
```

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

Console X Problems Debug Shell

No consoles to display at this time.

## 5. Create persistence.xml in META-INF folder



The screenshot shows the Eclipse IDE interface with the following details:

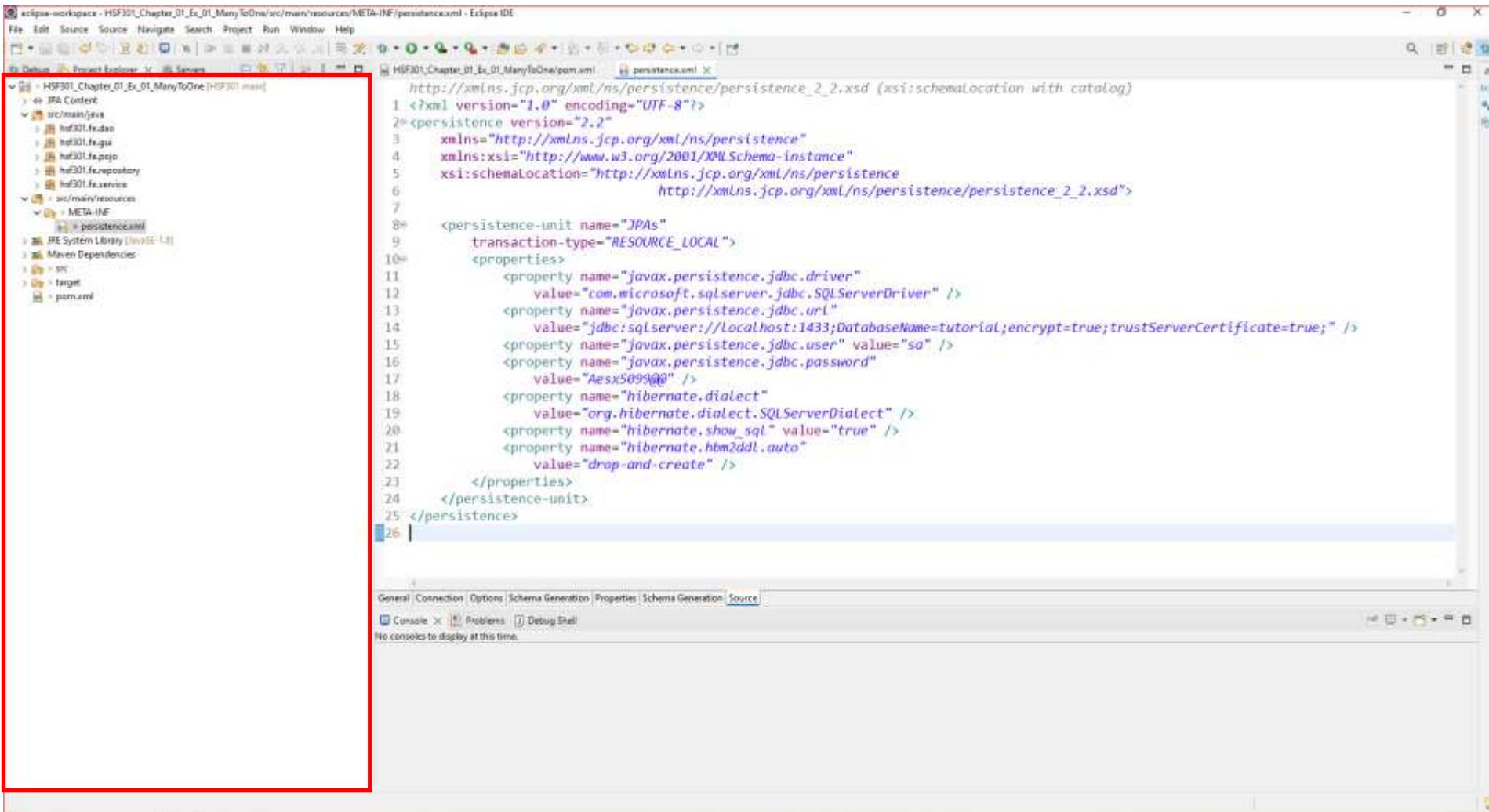
- Project Explorer:** Shows the project structure: `HFS301_Chapter_01_Ex_01_ManyToOne` (Maven) with subfolders `src/main/java`, `src/main/resources`, and `META-INF`.
- Editor:** Displays the `persistence.xml` file content.
- Content:** The XML code for `persistence.xml` is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2"
    xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
        http://xmlns.jcp.org/xml/ns/persistence_2_2.xsd">

    <persistence-unit name="JPAs"
        transaction-type="RESOURCE_LOCAL">
        <properties>
            <property name="javax.persistence.jdbc.driver"
                value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
            <property name="javax.persistence.jdbc.url"
                value="jdbc:sqlserver://localhost:1433;DatabaseName=tutorial;encrypt=true;trustServerCertificate=true;" />
            <property name="javax.persistence.jdbc.user" value="sa" />
            <property name="javax.persistence.jdbc.password"
                value="Aesx569980" />
            <property name="hibernate.dialect"
                value="org.hibernate.dialect.SQLServerDialect" />
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.hbm2ddl.auto"
                value="drop-and-create" />
        </properties>
    </persistence-unit>
</persistence>
```

The code is highlighted with syntax coloring, and the entire editor area is enclosed in a red border.

# 6. Create structure of project



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure: `HF301_Chapter_01_Ex_01_ManyToOne` (main) containing `src/main/java`, `src/main/resources`, and `META-INF`. The `META-INF` folder contains `persistence.xml`.
- Persistance.xml Content:** The `persistence.xml` file is open in the editor, showing the XML configuration for the persistence unit "JPAs".
- Editor Tab Bar:** The tabs include General, Connection, Options, Schema Generation, Properties, Schema Generation, and Source. The Source tab is selected.
- Status Bar:** Shows "No consoles to display at this time."

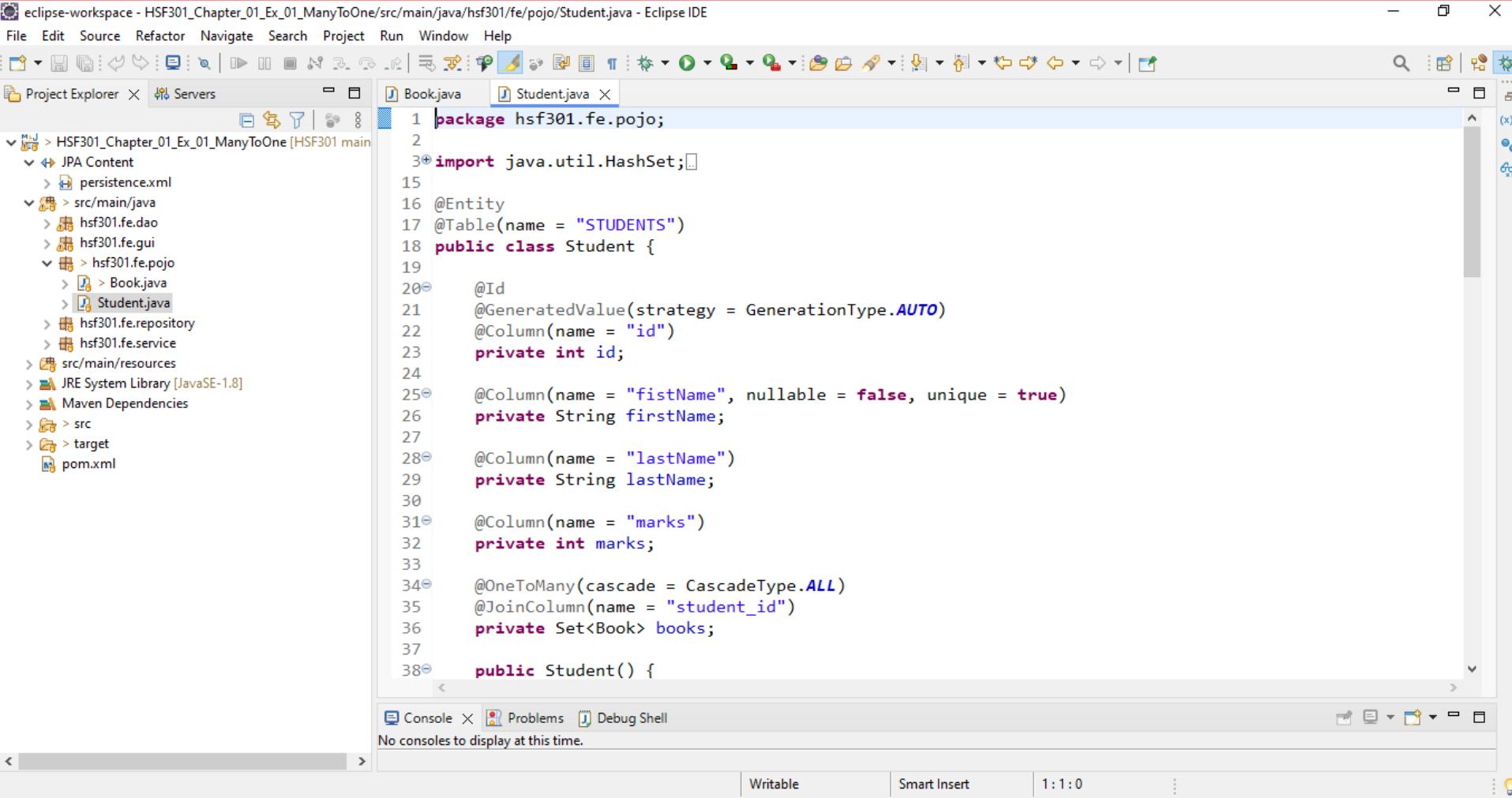
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2"
    xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
        http://xmlns.jcp.org/xml/ns/persistence_2_2.xsd">
    <persistence-unit name="JPAs"
        transaction-type="RESOURCE_LOCAL">
        <properties>
            <property name="javax.persistence.jdbc.driver"
                value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
            <property name="javax.persistence.jdbc.url"
                value="jdbc:sqlserver://localhost:1433;DatabaseName=tutorial;encrypt=true;trustServerCertificate=true;" />
            <property name="javax.persistence.jdbc.user" value="sa" />
            <property name="javax.persistence.jdbc.password"
                value="Aesx569980" />
            <property name="hibernate.dialect"
                value="org.hibernate.dialect.SQLServerDialect" />
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.hbm2ddl.auto"
                value="drop-and-create" />
        </properties>
    </persistence-unit>
</persistence>
```

## 7. Create Books in Pojo

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/pojo/Book.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar with various icons for file operations, search, and project management.
- Project Explorer:** Shows the project structure:
  - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne [HSF301 main]
  - JPA Content
  - persistence.xml
  - src/main/java
    - hsf301.fe.dao
    - hsf301.fe.gui
    - hsf301.fe.pojo
      - Book.java
      - Student.java
    - hsf301.fe.repository
    - hsf301.fe.service
  - src/main/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- Code Editor:** The active editor is Book.java, displaying Java code for a Book entity. The code includes annotations for Entity, Table, Id, Column, and ManyToOne. The line `@ManyToOne(cascade = CascadeType.ALL)` is highlighted with a blue selection bar.
- Bottom Status Bar:** Shows tabs for Console, Problems, and Debug Shell, and status indicators for Writable, Smart Insert, and the current timestamp (28 : 42 : 646).

# 8. Create Students in Pojo

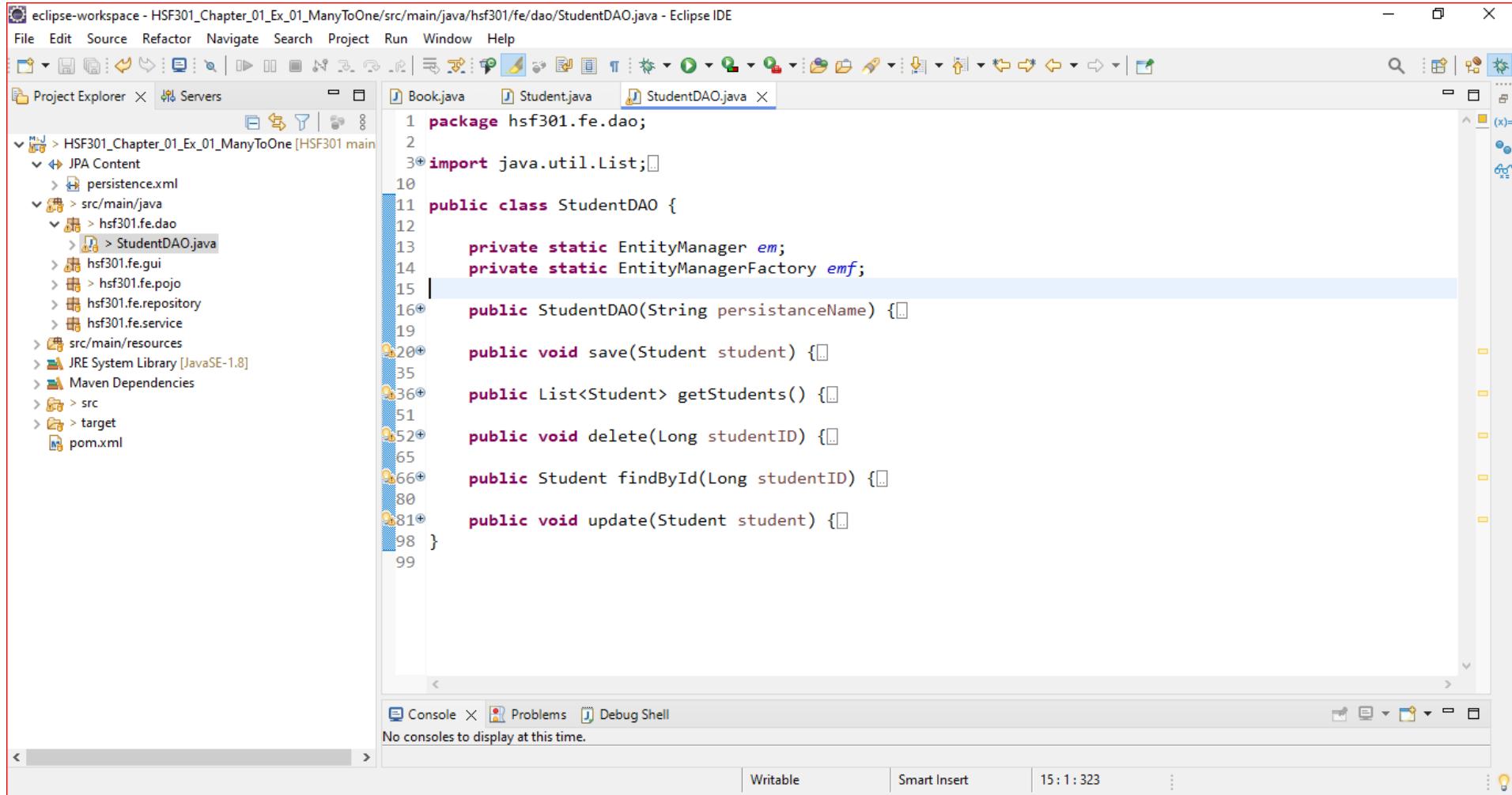


The screenshot shows the Eclipse IDE interface with a red border around the main window. The title bar reads "eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/pojo/Student.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The Project Explorer view on the left shows the project structure under "HSF301\_Chapter\_01\_Ex\_01\_ManyToOne": JPA Content, persistence.xml, src/main/java (containing hsf301.fe.dao, hsf301.fe.gui, hsf301.fe.pojo, Book.java, Student.java), src/main/resources, JRE System Library [JavaSE-1.8], Maven Dependencies, src, target, and pom.xml. The Student.java file is open in the editor tab, showing the following code:

```
1 package hsf301.fe.pojo;
2
3 import java.util.HashSet;
4
5
6 @Entity
7 @Table(name = "STUDENTS")
8 public class Student {
9
10    @Id
11    @GeneratedValue(strategy = GenerationType.AUTO)
12    @Column(name = "id")
13    private int id;
14
15    @Column(name = "firstName", nullable = false, unique = true)
16    private String firstName;
17
18    @Column(name = "lastName")
19    private String lastName;
20
21    @Column(name = "marks")
22    private int marks;
23
24    @OneToMany(cascade = CascadeType.ALL)
25    @JoinColumn(name = "student_id")
26    private Set<Book> books;
27
28    public Student() {
29
30    }
31}
```

The editor status bar at the bottom indicates "Writable", "Smart Insert", "1:1:0", and a lightbulb icon.

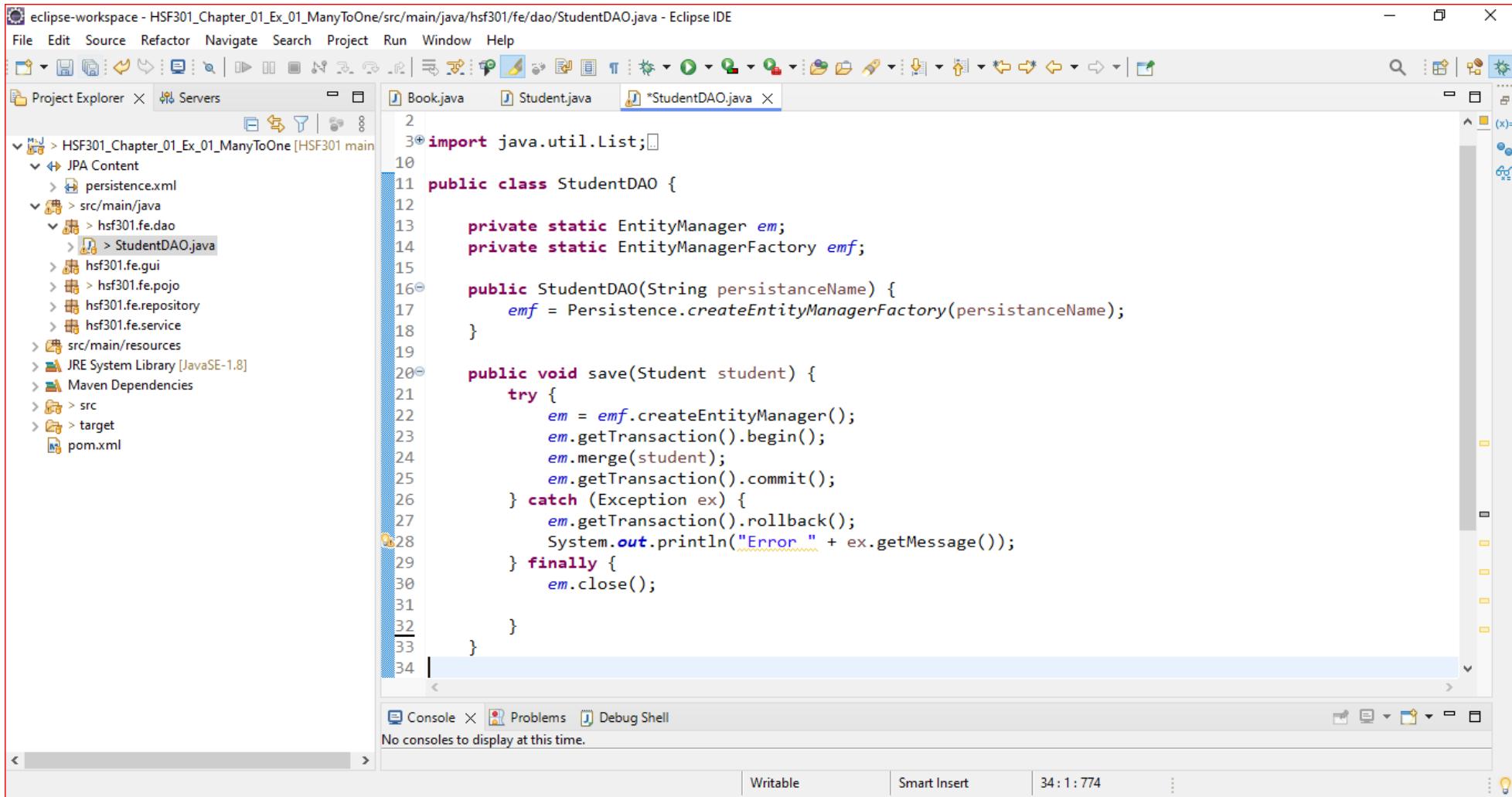
# 9. Create StudentDAO



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/dao/StudentDAO.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Project Explorer:** Shows the project structure:
  - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne [HSF301 main]
  - JPA Content
  - persistence.xml
  - src/main/java
    - hsf301.fe.dao
      - StudentDAO.java
    - hsf301.fe.gui
    - hsf301.fe.pojo
    - hsf301.fe.repository
    - hsf301.fe.service
  - src/main/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- Editor:** The StudentDAO.java file is open in the editor. The code defines a DAO class for managing students using Entity Manager.
- Bottom Bar:** Shows tabs for Console, Problems, and Debug Shell, with the message "No consoles to display at this time." The status bar shows "Writable", "Smart Insert", and the timestamp "15:1:323".

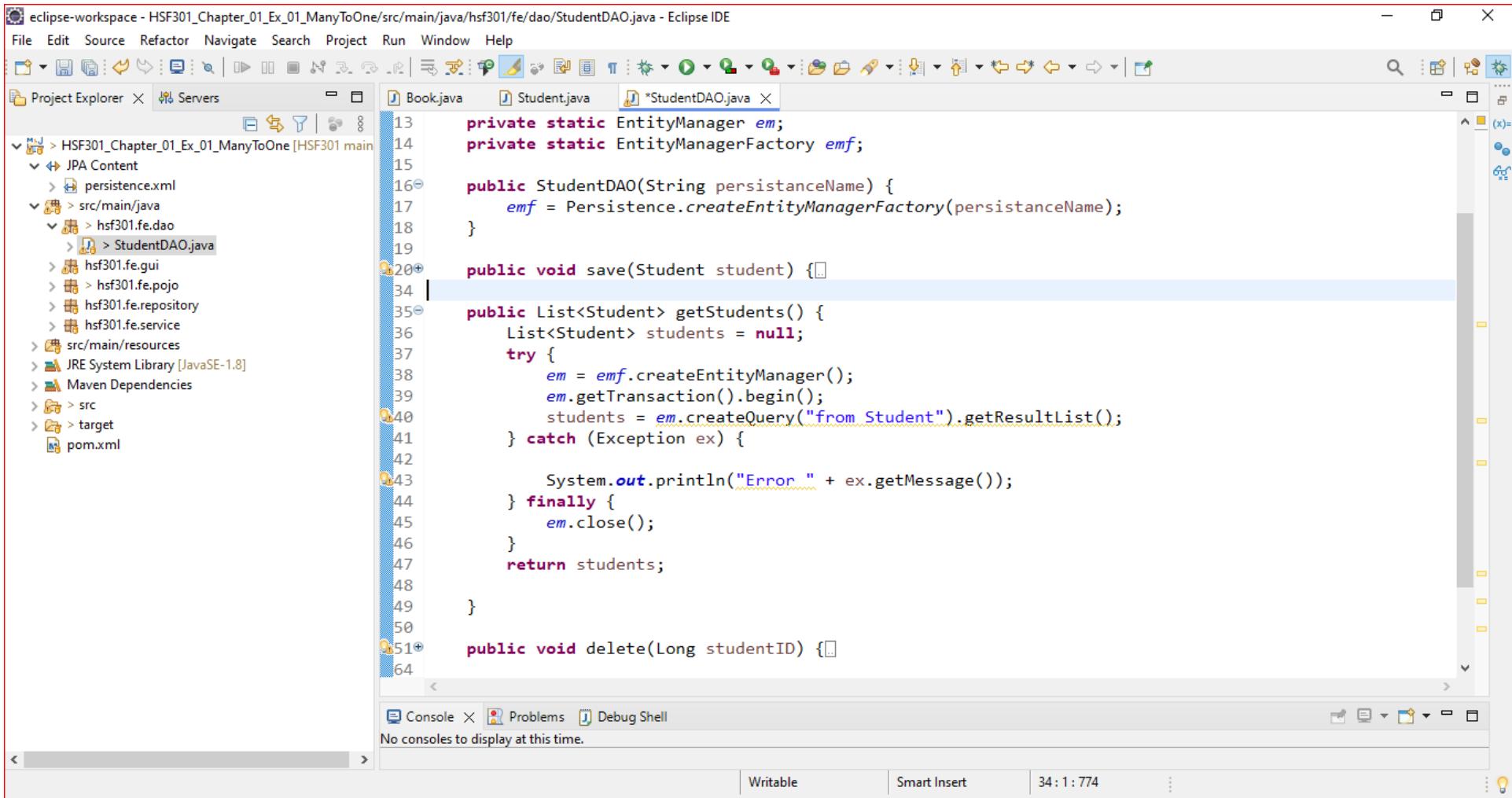
# 10. Save Student in StudentDAO



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/dao/StudentDAO.java - Eclipse IDE
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Project Explorer:** Shows the project structure:
  - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne [HSF301 main]
  - JPA Content
  - persistence.xml
  - src/main/java
    - hsf301.fe.dao
      - StudentDAO.java
    - hsf301.fe.gui
    - hsf301.fe.pojo
    - hsf301.fe.repository
    - hsf301.fe.service
  - src/main/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- Code Editor:** Displays the StudentDAO.java code. The code implements a DAO interface for saving students using JPA. It uses EntityManager and EntityManagerFactory to manage transactions and persist entities.
- Bottom Bar:** Shows tabs for Console, Problems, and Debug Shell, all of which are empty. Status bar at the bottom indicates Writable, Smart Insert, and the current line number (34 : 1 : 774).

# 11. Get All Students in StudentDAO



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/dao/StudentDAO.java - Eclipse IDE
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Project Explorer:** Shows the project structure:
  - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne [HSF301 main]
  - JPA Content
  - persistence.xml
  - src/main/java
    - hsf301.fe.dao
      - StudentDAO.java
    - hsf301.fe.gui
    - hsf301.fe.pojo
    - hsf301.fe.repository
    - hsf301.fe.service
  - src/main/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- Code Editor:** The StudentDAO.java file is open, showing Java code for a DAO layer. The code includes EntityManager and EntityManagerFactory, persistenceName, save, getStudents, and delete methods. The getStudents method uses JPA to query the database.

```
private static EntityManager em;
private static EntityManagerFactory emf;

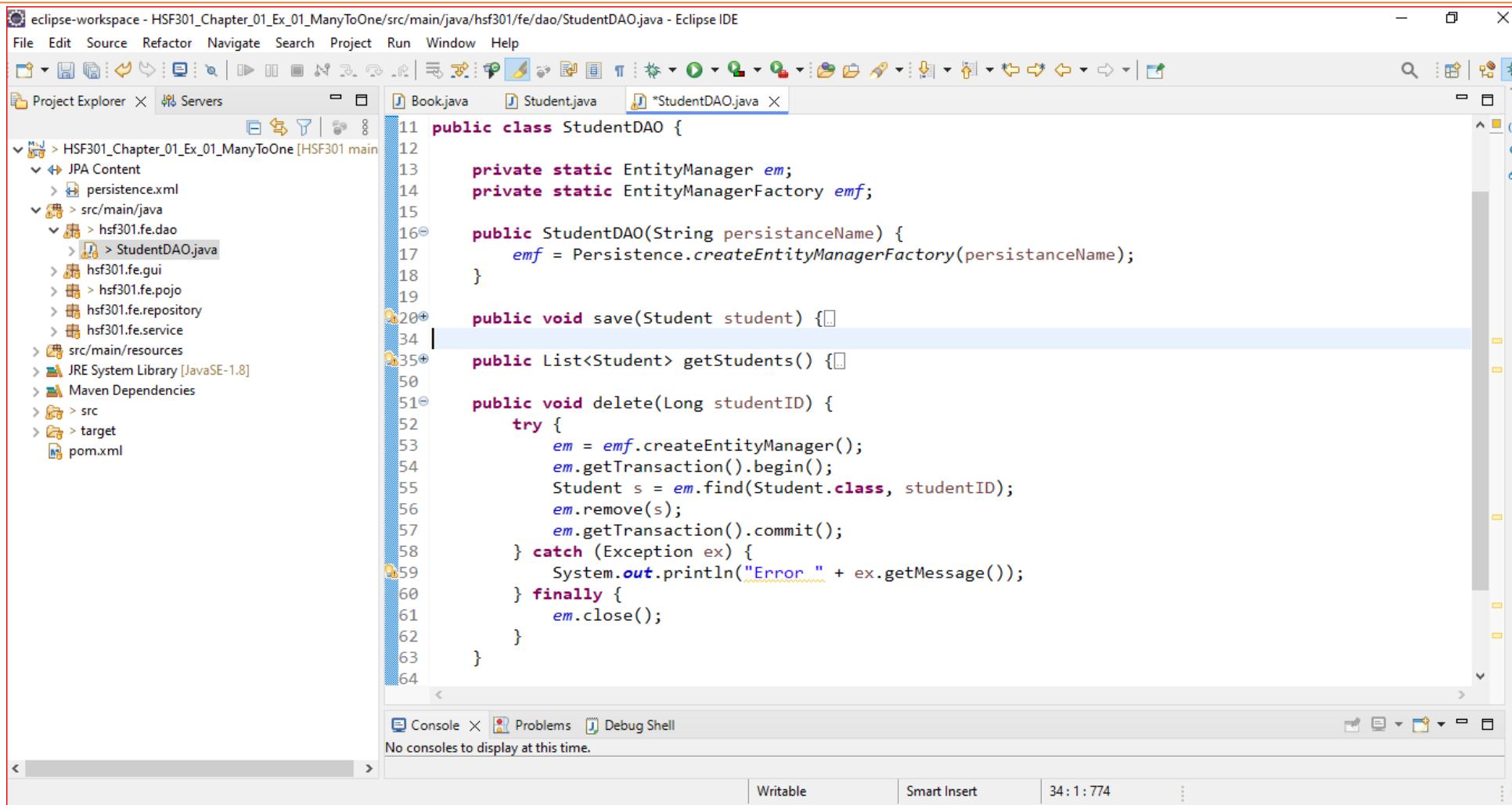
public StudentDAO(String persistanceName) {
    emf = Persistence.createEntityManagerFactory(persistanceName);
}

public void save(Student student) {}

public List<Student> getStudents() {
    List<Student> students = null;
    try {
        em = emf.createEntityManager();
        em.getTransaction().begin();
        students = em.createQuery("from Student").getResultList();
    } catch (Exception ex) {
        System.out.println("Error " + ex.getMessage());
    } finally {
        em.close();
    }
    return students;
}

public void delete(Long studentID) {}
```
- Console:** No consoles to display at this time.
- Status Bar:** Writable, Smart Insert, 34:1:774

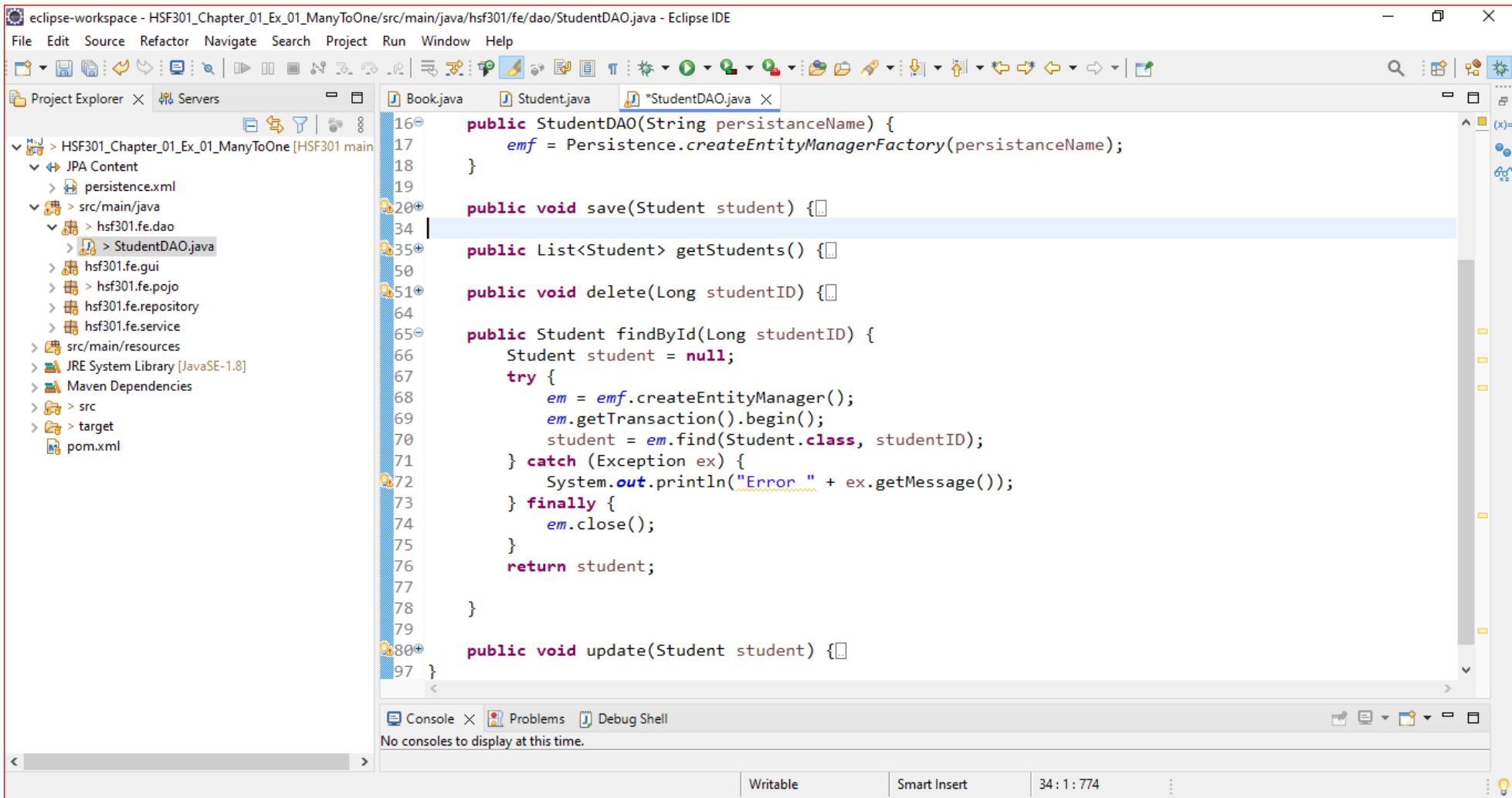
# 12. Delete Student in StudentDAO



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/dao/StudentDAO.java - Eclipse IDE
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbars:** Standard toolbar, Java Development toolbar, Persistence toolbar, Persistence Navigator toolbar, Database toolbar, Database Navigator toolbar, JBoss Seam toolbar, Seam Persistence toolbar, Seam Navigation toolbar, Seam Faces toolbar, Seam Faces Persistence toolbar, Seam Faces Navigation toolbar, Seam Faces Persistence Navigator toolbar.
- Project Explorer:** Shows the project structure:
  - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne [HSF301 main]
    - JPA Content
    - persistence.xml
    - src/main/java
      - hsf301.fe.dao
        - StudentDAO.java
      - hsf301.fe.gui
      - hsf301.fe.pojo
      - hsf301.fe.repository
      - hsf301.fe.service
    - src/main/resources
    - JRE System Library [JavaSE-1.8]
    - Maven Dependencies
    - src
    - target
    - pom.xml
- Editor:** Displays the StudentDAO.java code. The code implements a DAO interface for managing Student entities using JPA. It includes methods for saving, getting all students, and deleting a student by ID. Error handling is provided using try-catch-finally blocks.
- Bottom Bar:** Console X, Problems, Debug Shell. The message "No consoles to display at this time." is shown.
- Status Bar:** Writable, Smart Insert, 34:1:774

# 13. Get a Student in StudentDAO



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/dao/StudentDAO.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Project Explorer:** Shows the project structure:
  - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne [HSF301 main]
  - JPA Content
  - persistence.xml
  - src/main/java
    - hsf301.fe.dao
      - StudentDAO.java
    - hsf301.fe.gui
    - hsf301.fe.pojo
    - hsf301.fe.repository
    - hsf301.fe.service
  - src/main/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- Code Editor:** Displays the `StudentDAO.java` file content:

```
public StudentDAO(String persistanceName) {
    emf = Persistence.createEntityManagerFactory(persistanceName);
}

public void save(Student student) {}

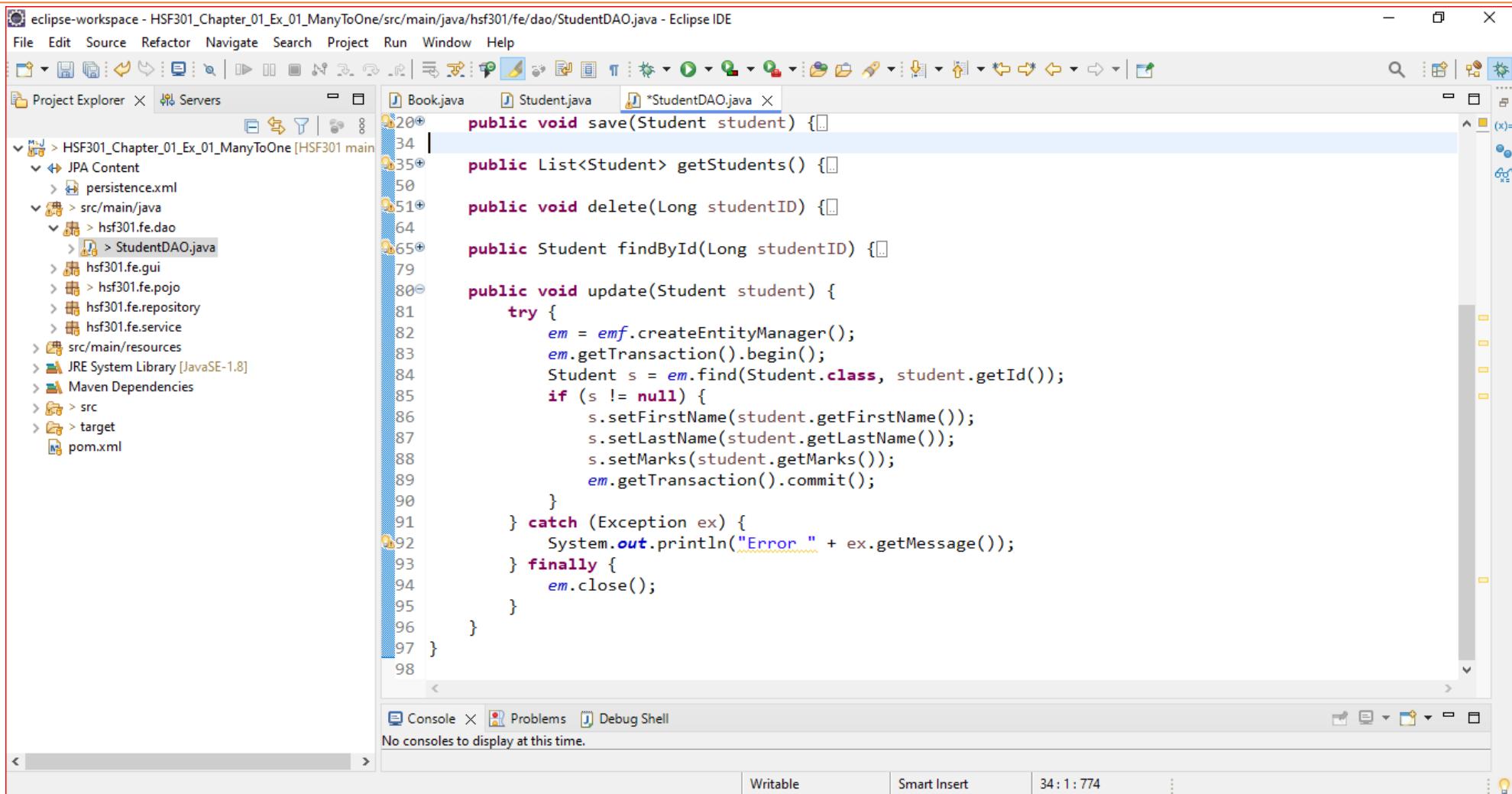
public List<Student> getStudents() {}

public void delete(Long studentID) {}

public Student findById(Long studentID) {
    Student student = null;
    try {
        em = emf.createEntityManager();
        em.getTransaction().begin();
        student = em.find(Student.class, studentID);
    } catch (Exception ex) {
        System.out.println("Error " + ex.getMessage());
    } finally {
        em.close();
    }
    return student;
}

public void update(Student student) {}
```
- Bottom Bar:** Console X, Problems, Debug Shell. A message says "No consoles to display at this time."
- Status Bar:** Writable, Smart Insert, 34:1:774

# 14. Get a Student in StudentDAO

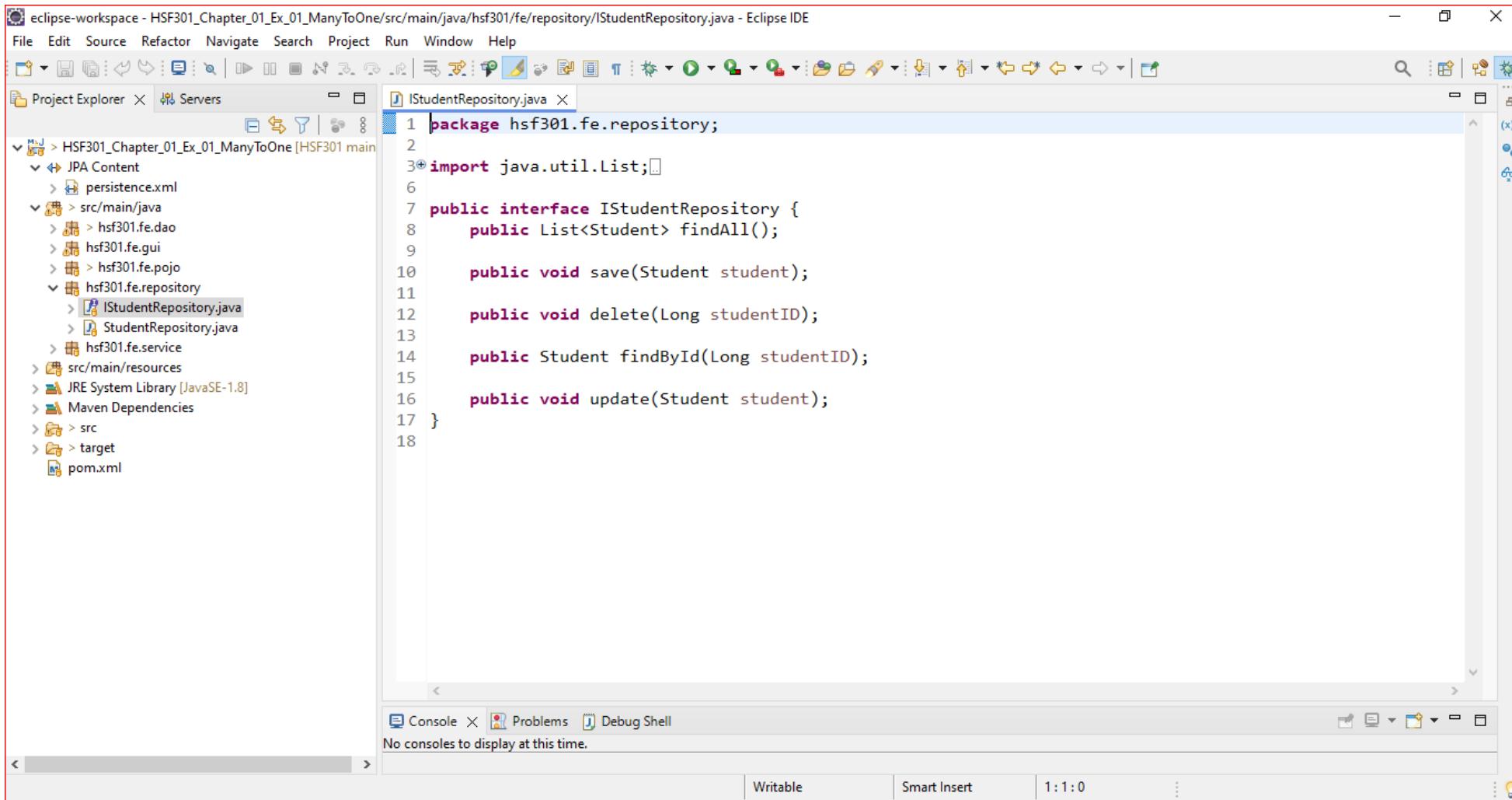


The screenshot shows the Eclipse IDE interface with a red border around the main window. The title bar reads "eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/dao/StudentDAO.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The Project Explorer view on the left shows the project structure under "HSF301\_Chapter\_01\_Ex\_01\_ManyToOne": JPA Content, src/main/java (containing hsf301.fe.dao, hsf301.fe.gui, hsf301.fe.pojo, hsf301.fe.repository, hsf301.fe.service), src/main/resources, JRE System Library [JavaSE-1.8], Maven Dependencies, src, target, and pom.xml. The central editor pane displays the code for StudentDAO.java:

```
public void save(Student student) {  
    public List<Student> getStudents() {}  
    public void delete(Long studentID) {}  
    public Student findById(Long studentID) {}  
    public void update(Student student) {  
        try {  
            em = emf.createEntityManager();  
            em.getTransaction().begin();  
            Student s = em.find(Student.class, student.getId());  
            if (s != null) {  
                s.setFirstName(student.getFirstName());  
                s.setLastName(student.getLastName());  
                s.setMarks(student.getMarks());  
                em.getTransaction().commit();  
            }  
        } catch (Exception ex) {  
            System.out.println("Error " + ex.getMessage());  
        } finally {  
            em.close();  
        }  
    }  
}
```

The code uses EntityManager (em) and EntityManagerFactory (emf) from the persistence.xml configuration. The DAO methods handle saving, deleting, finding by ID, and updating students. The update method uses a try-catch block to handle exceptions and prints error messages to the console. The finally block ensures the EntityManager is closed.

# 15. Create IStudentRepository

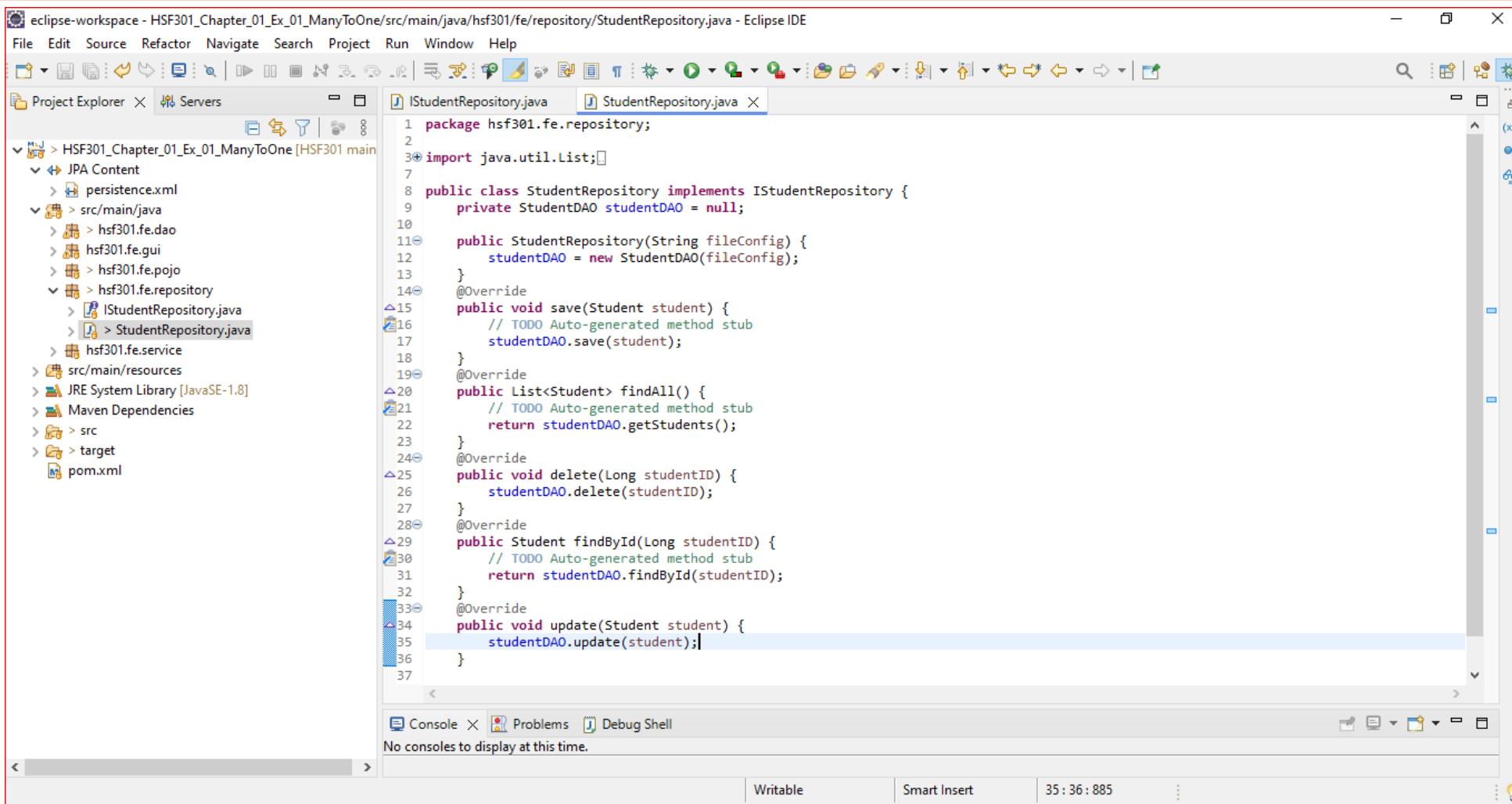


The screenshot shows the Eclipse IDE interface with a red border around the main workspace. The title bar reads "eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/repository/IStudentRepository.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The Project Explorer view on the left shows a project structure under "HSF301\_Chapter\_01\_Ex\_01\_ManyToOne": JPA Content, src/main/java (containing hsf301.fe.dao, hsf301.fe.gui, hsf301.fe.pojo, hsf301.fe.repository, hsf301.fe.service), src/main/resources, JRE System Library [JavaSE-1.8], Maven Dependencies, src, target, pom.xml. The src/main/java/hsf301/fe/repository folder contains IStudentRepository.java and StudentRepository.java. The IStudentRepository.java file is open in the editor, showing the following code:

```
1 package hsf301.fe.repository;
2
3 import java.util.List;
4
5 public interface IStudentRepository {
6     public List<Student> findAll();
7
8     public void save(Student student);
9
10    public void delete(Long studentID);
11
12    public Student findById(Long studentID);
13
14    public void update(Student student);
15
16 }
17
18 }
```

The editor status bar at the bottom shows "Writable", "Smart Insert", "1:1:0", and a lightbulb icon. Below the editor is a toolbar with Console, Problems, and Debug Shell tabs, and a message "No consoles to display at this time." The status bar also shows "1:1:0" and a lightbulb icon.

# 16. Create StudentRepository

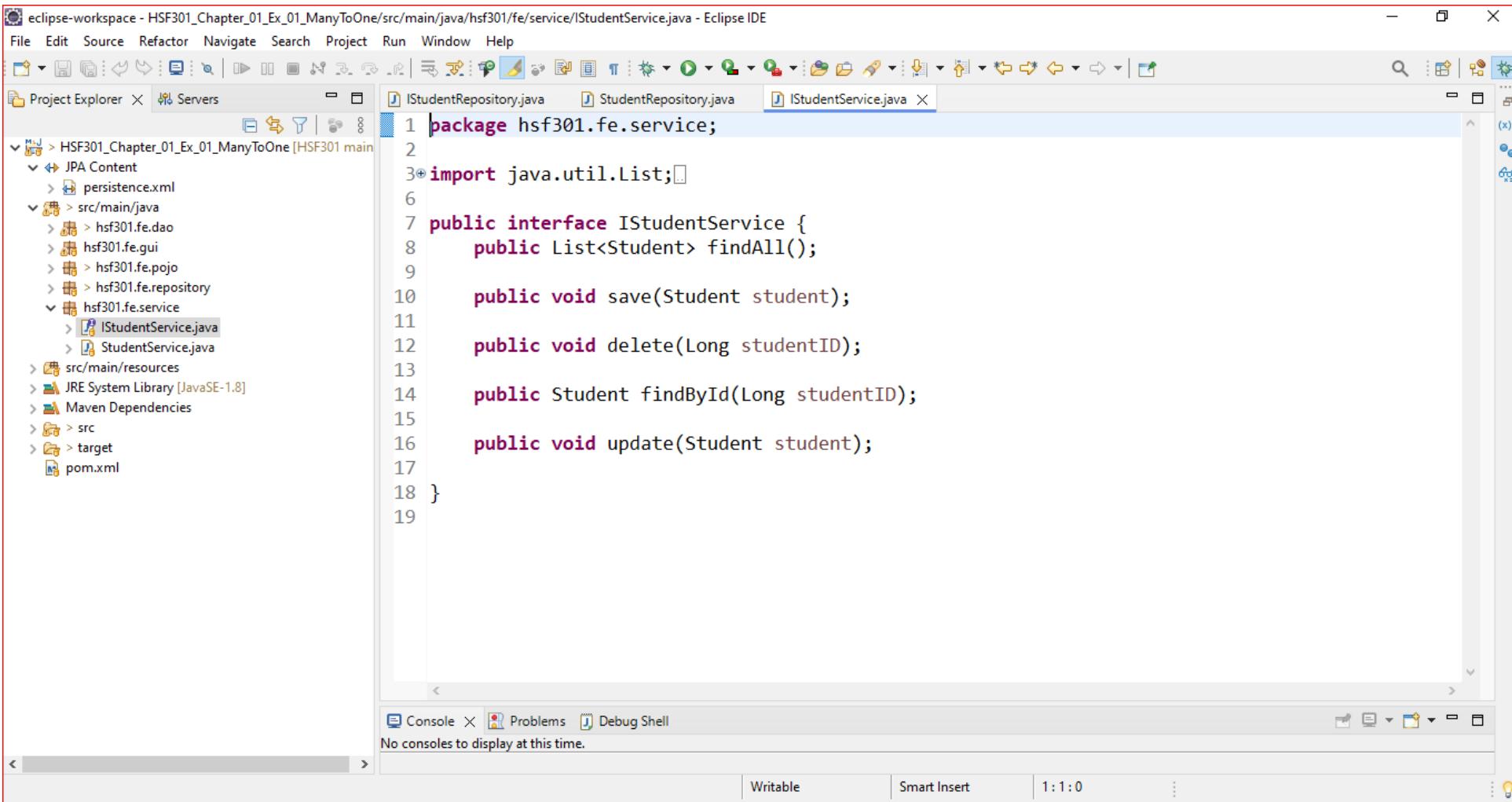


The screenshot shows the Eclipse IDE interface with a red border around the main workspace. The title bar reads "eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/repository/StudentRepository.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Find, and Run. The Project Explorer view on the left shows the project structure under "HSF301\_Chapter\_01\_Ex\_01\_ManyToOne [HSF301 main]". The "src/main/java" folder contains packages like "hsf301.fe.dao", "hsf301.fe.gui", "hsf301.fe.pojo", "hsf301.fe.repository", and "hsf301.fe.service". The "hsf301.fe.repository" package contains files "IStudentRepository.java" and "StudentRepository.java". The "IStudentRepository.java" file is currently selected in the tabs. The "StudentRepository.java" file is open in the editor, showing the following code:

```
1 package hsf301.fe.repository;
2
3 import java.util.List;
4
5 public class StudentRepository implements IStudentRepository {
6     private StudentDAO studentDAO = null;
7
8     public StudentRepository(String fileConfig) {
9         studentDAO = new StudentDAO(fileConfig);
10    }
11    @Override
12    public void save(Student student) {
13        // TODO Auto-generated method stub
14        studentDAO.save(student);
15    }
16    @Override
17    public List<Student> findAll() {
18        // TODO Auto-generated method stub
19        return studentDAO.getStudents();
20    }
21    @Override
22    public void delete(Long studentID) {
23        studentDAO.delete(studentID);
24    }
25    @Override
26    public Student findById(Long studentID) {
27        // TODO Auto-generated method stub
28        return studentDAO.findById(studentID);
29    }
30    @Override
31    public void update(Student student) {
32        studentDAO.update(student);
33    }
34}
```

The "Console" tab at the bottom shows "No consoles to display at this time." The status bar at the bottom right indicates "Writable", "Smart Insert", and the current line and column as "35 : 36 : 885".

# 17. Create IStudentService

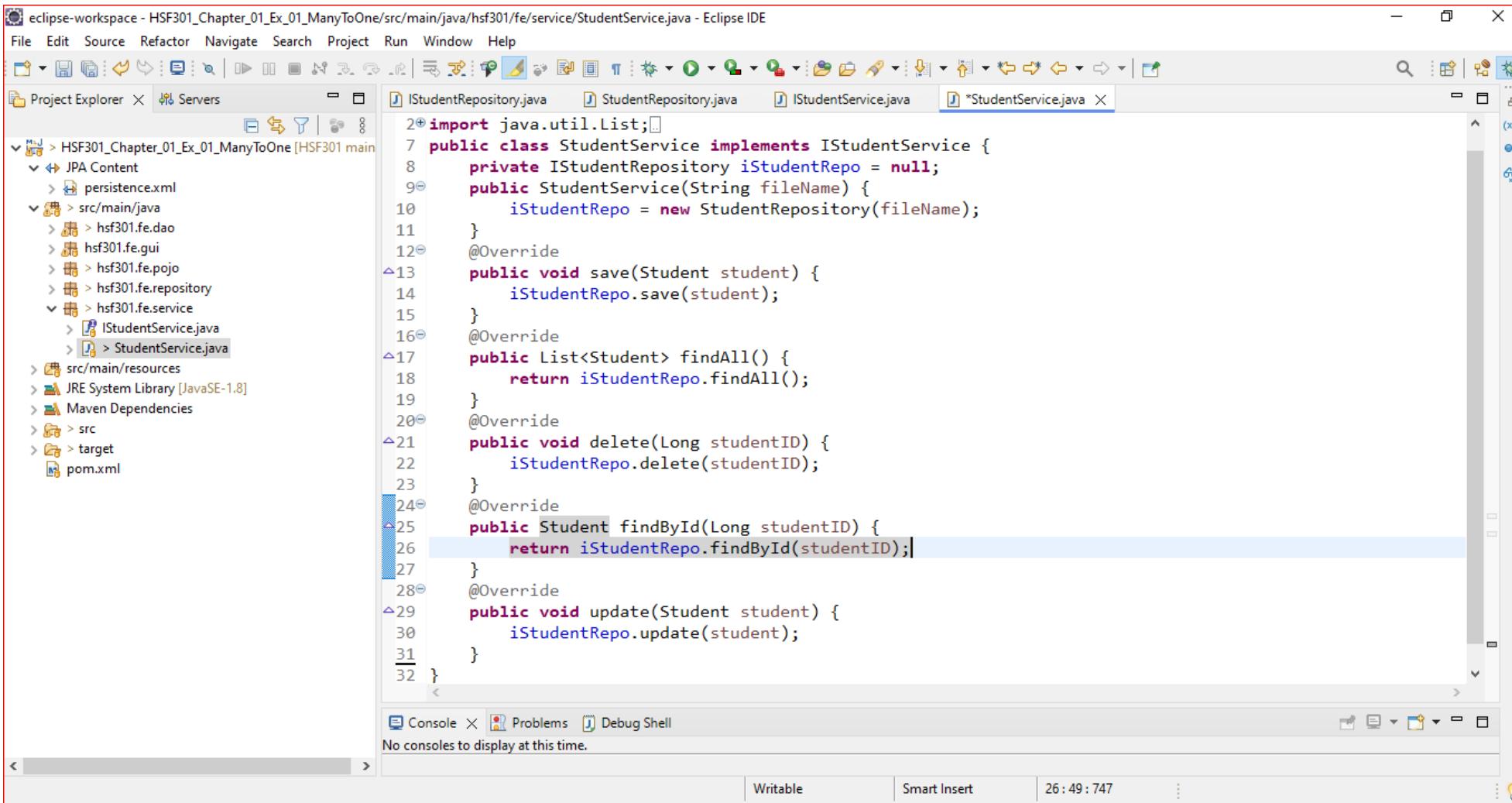


The screenshot shows the Eclipse IDE interface with a red border around the main window. The title bar reads "eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/service/IStudentService.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like New, Open, Save, Cut, Copy, Paste, Find, and Run. The Project Explorer view on the left shows a project structure under "HSF301\_Chapter\_01\_Ex\_01\_ManyToOne [HSF301 main]". It includes JPA Content (persistence.xml), src/main/java (hsf301.fe.dao, hsf301.fe.gui, hsf301.fe.pojo, hsf301.fe.repository, hsf301.fe.service, IStudentService.java, StudentService.java), src/main/resources, JRE System Library [JavaSE-1.8], Maven Dependencies, src, target, and pom.xml. The IStudentService.java tab is selected in the editor area, which contains the following code:

```
1 package hsf301.fe.service;
2
3 import java.util.List;
4
5 public interface IStudentService {
6     public List<Student> findAll();
7
8     public void save(Student student);
9
10    public void delete(Long studentID);
11
12    public Student findById(Long studentID);
13
14    public void update(Student student);
15
16 }
17
18 }
```

The bottom of the screen shows the Eclipse status bar with tabs for Console, Problems, and Debug Shell, and a message "No consoles to display at this time." The status bar also displays Writable, Smart Insert, and a timestamp of 1:1:0.

# 18. Create StudentService

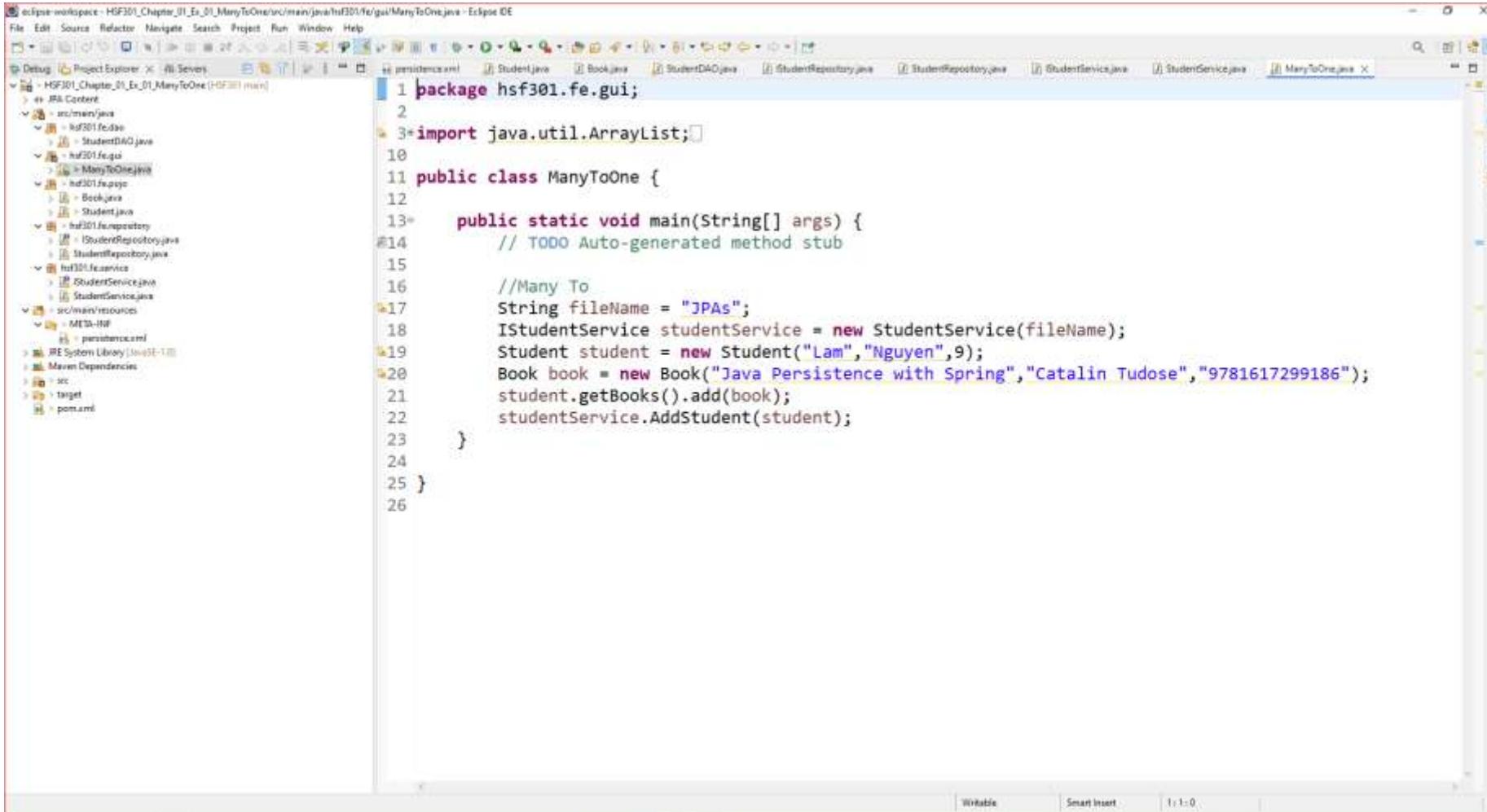


The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/service/StudentService.java - Eclipse IDE
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Project Explorer:** Shows the project structure:
  - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne [HSF301 main]
  - JPA Content
  - persistence.xml
  - src/main/java
    - hsf301.fe.dao
    - hsf301.fe.gui
    - hsf301.fe.pojo
    - hsf301.fe.repository
    - hsf301.fe.service
      - IStudentService.java
      - StudentService.java
  - src/main/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- Code Editor:** Displays the `StudentService.java` file content:

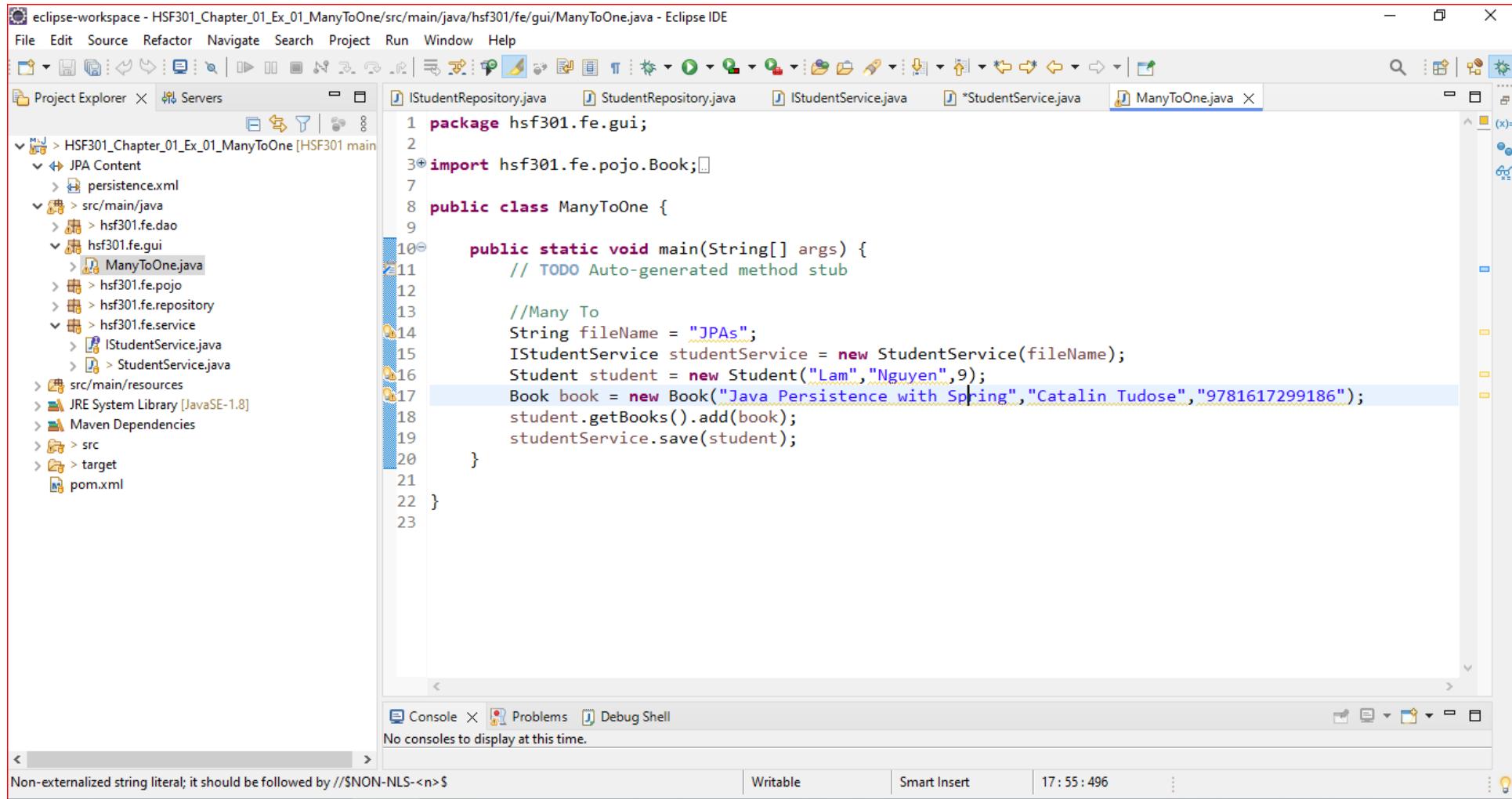
```
2+ import java.util.List;
3  public class StudentService implements IStudentService {
4      private IStudentRepository iStudentRepo = null;
5      public StudentService(String fileName) {
6          iStudentRepo = new StudentRepository(fileName);
7      }
8      @Override
9      public void save(Student student) {
10         iStudentRepo.save(student);
11     }
12     @Override
13     public List<Student> findAll() {
14         return iStudentRepo.findAll();
15     }
16     @Override
17     public void delete(Long studentID) {
18         iStudentRepo.delete(studentID);
19     }
20     @Override
21     public Student findById(Long studentID) {
22         return iStudentRepo.findById(studentID);
23     }
24     @Override
25     public void update(Student student) {
26         iStudentRepo.update(student);
27     }
28     @Override
29     public void deleteAll() {
30         iStudentRepo.deleteAll();
31     }
32 }
```
- Bottom Status Bar:** Writable, Smart Insert, 26:49 : 747
- Console:** No consoles to display at this time.

# 19. Create Main function



```
1 package hsf301.fe.gui;
2
3 import java.util.ArrayList;
4
5 public class ManyToOne {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9
10        //Many To
11        String fileName = "JPAs";
12        IStudentService studentService = new StudentService(fileName);
13        Student student = new Student("Lam", "Nguyen", 9);
14        Book book = new Book("Java Persistence with Spring", "Catalin Tudose", "9781617299186");
15        student.getBooks().add(book);
16        studentService.AddStudent(student);
17    }
18
19 }
20
21 }
```

# 20. Run Program

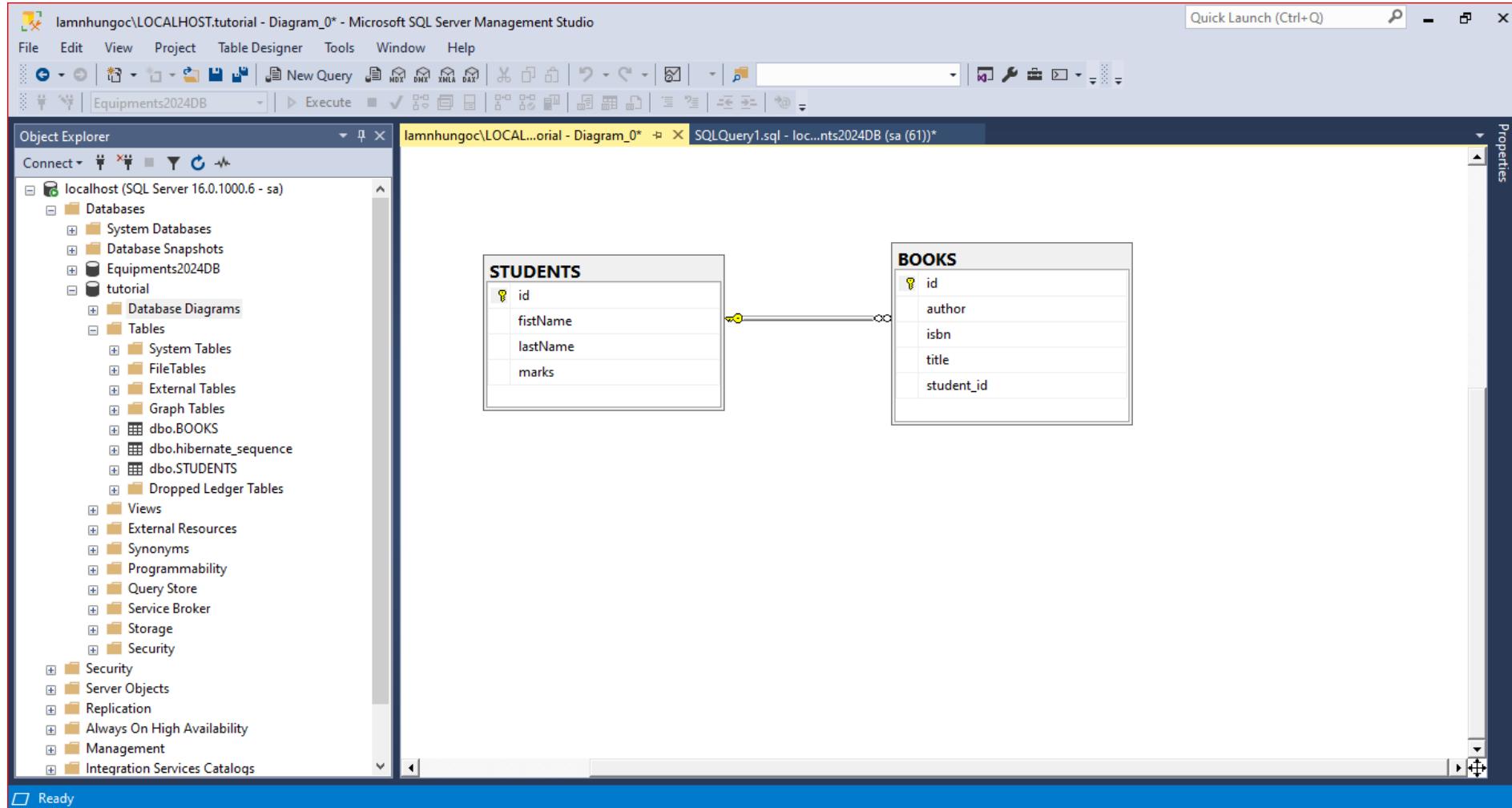


The screenshot shows the Eclipse IDE interface with the title "eclipse-workspace - HSF301\_Chapter\_01\_Ex\_01\_ManyToOne/src/main/java/hsf301/fe/gui/ManyToOne.java - Eclipse IDE". The Project Explorer view on the left shows the project structure under "HSF301\_Chapter\_01\_Ex\_01\_ManyToOne". The code editor on the right displays the content of ManyToOne.java:

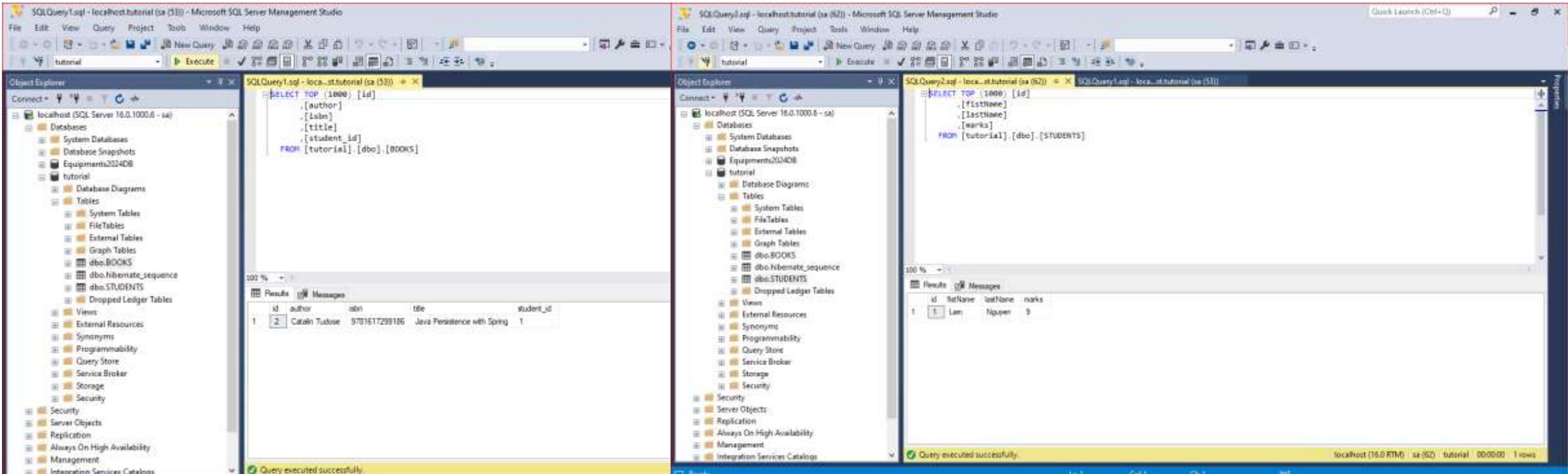
```
1 package hsf301.fe.gui;
2
3 import hsf301.fe.pojo.Book;
4
5 public class ManyToOne {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9
10        //Many To
11        String fileName = "JPAs";
12        IStudentService studentService = new StudentService(fileName);
13        Student student = new Student("Lam", "Nguyen", 9);
14        Book book = new Book("Java Persistence with Spring", "Catalin Tudose", "9781617299186");
15        student.getBooks().add(book);
16        studentService.save(student);
17    }
18
19
20}
21
22}
```

The code uses Java Persistence API (JPA) annotations like `@Entity` and `@ManyToOne` to map entities. It creates a `Student` object with a name ("Lam", "Nguyen") and age (9), adds a `Book` object ("Java Persistence with Spring" by Catalin Tudose, ISBN "9781617299186") to its books collection, and saves the student to a database.

# 21. Result



## 22. Result



The screenshot shows two Microsoft SQL Server Management Studio windows side-by-side, both connected to the 'tutorial' database on 'localhost'.

**Left Window:**

```
SELECT TOP (1000) [id]
      ,[author]
      ,[isbn]
      ,[title]
      ,[student_id]
  FROM [tutorial].[dbo].[BOOKS]
```

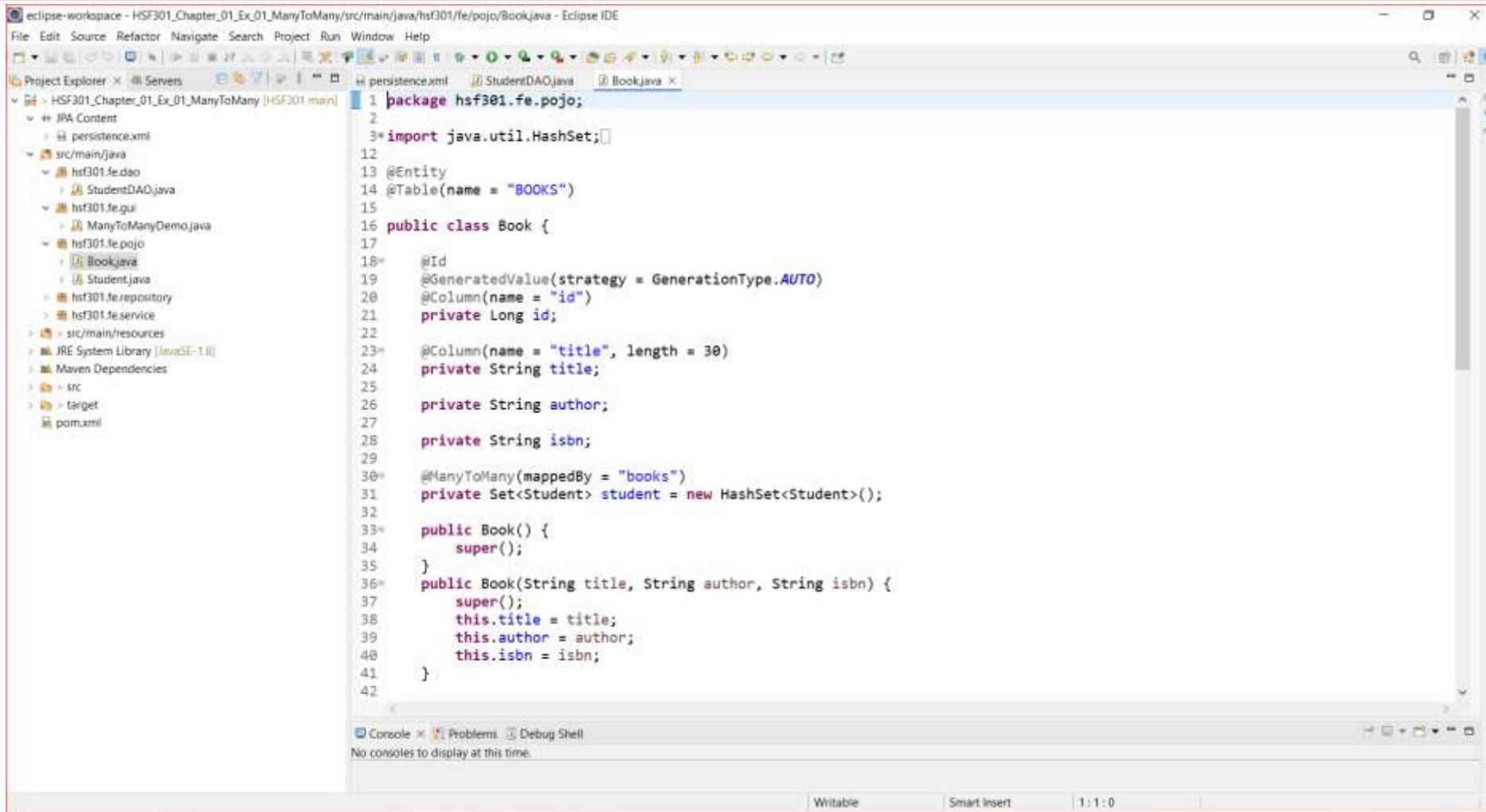
**Right Window:**

```
SELECT TOP (1000) [id]
      ,[firstname]
      ,[lastname]
      ,[marks]
  FROM [tutorial].[dbo].[STUDENTS]
```

Both queries executed successfully, as indicated by the status bar at the bottom of each window.

# Demo JPA (Many To Many)

# 1. Create Books in Pojo's Package



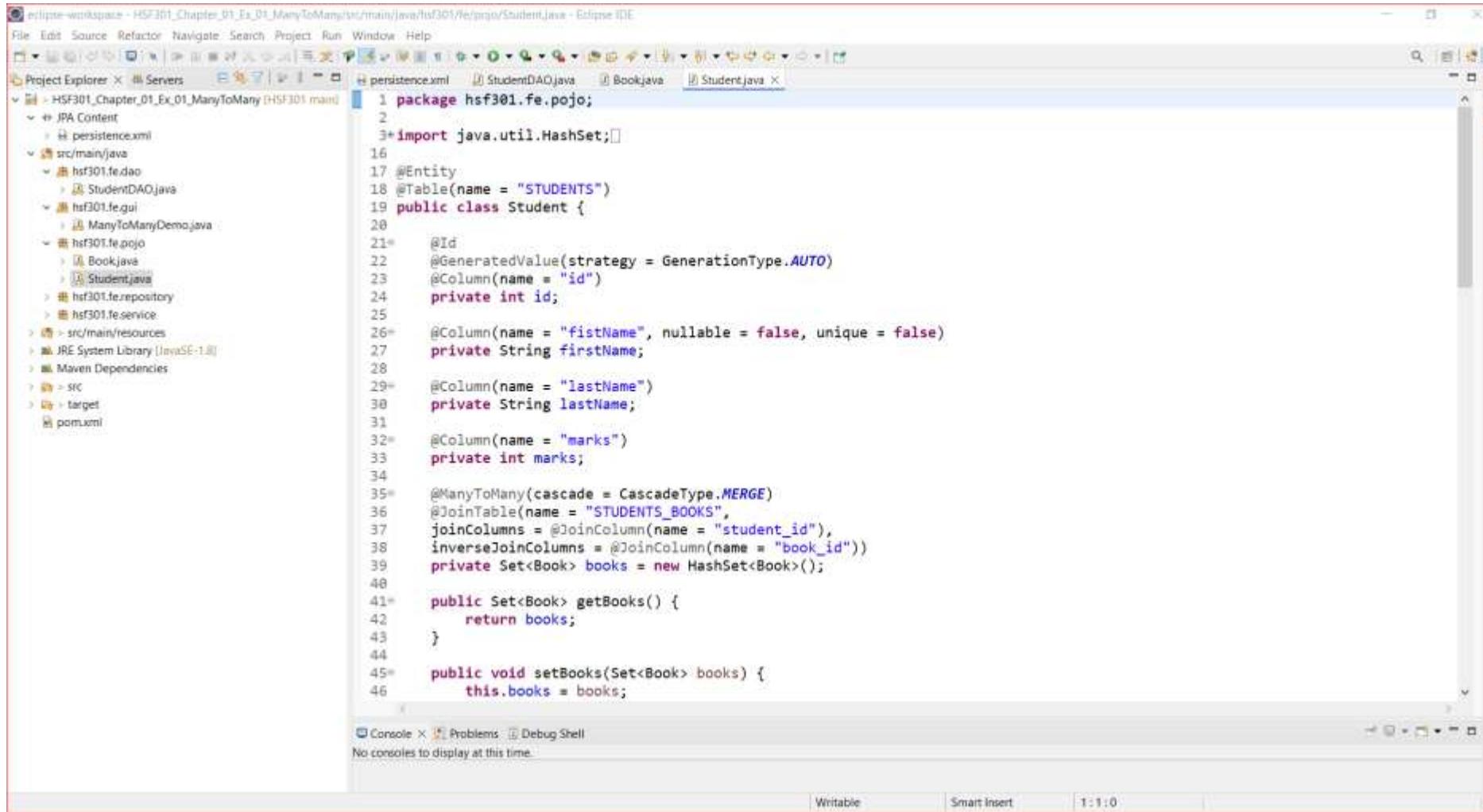
The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure: HSF301\_Chapter\_01\_Ex\_01\_ManyToMany, containing JPA Content, src/main/java, and various DAO, POJO, and service classes.
- Editor Tab:** The active tab is Book.java, which contains the following Java code:

```
1 package hsf301.fe.pojo;
2
3 import java.util.HashSet;
4
5 @Entity
6 @Table(name = "BOOKS")
7
8 public class Book {
9
10    @Id
11    @GeneratedValue(strategy = GenerationType.AUTO)
12    @Column(name = "id")
13    private Long id;
14
15    @Column(name = "title", length = 30)
16    private String title;
17
18    private String author;
19
20    private String isbn;
21
22    @ManyToMany(mappedBy = "books")
23    private Set<Student> student = new HashSet<Student>();
24
25
26    public Book() {
27        super();
28    }
29
30    public Book(String title, String author, String isbn) {
31        super();
32        this.title = title;
33        this.author = author;
34        this.isbn = isbn;
35    }
36}
```

The code defines a `Book` class with attributes `id`, `title`, `author`, and `isbn`. It uses `@ManyToMany` annotation to map to the `student` set in the `Student` class.

## 2. Create Students in Pojo's Package

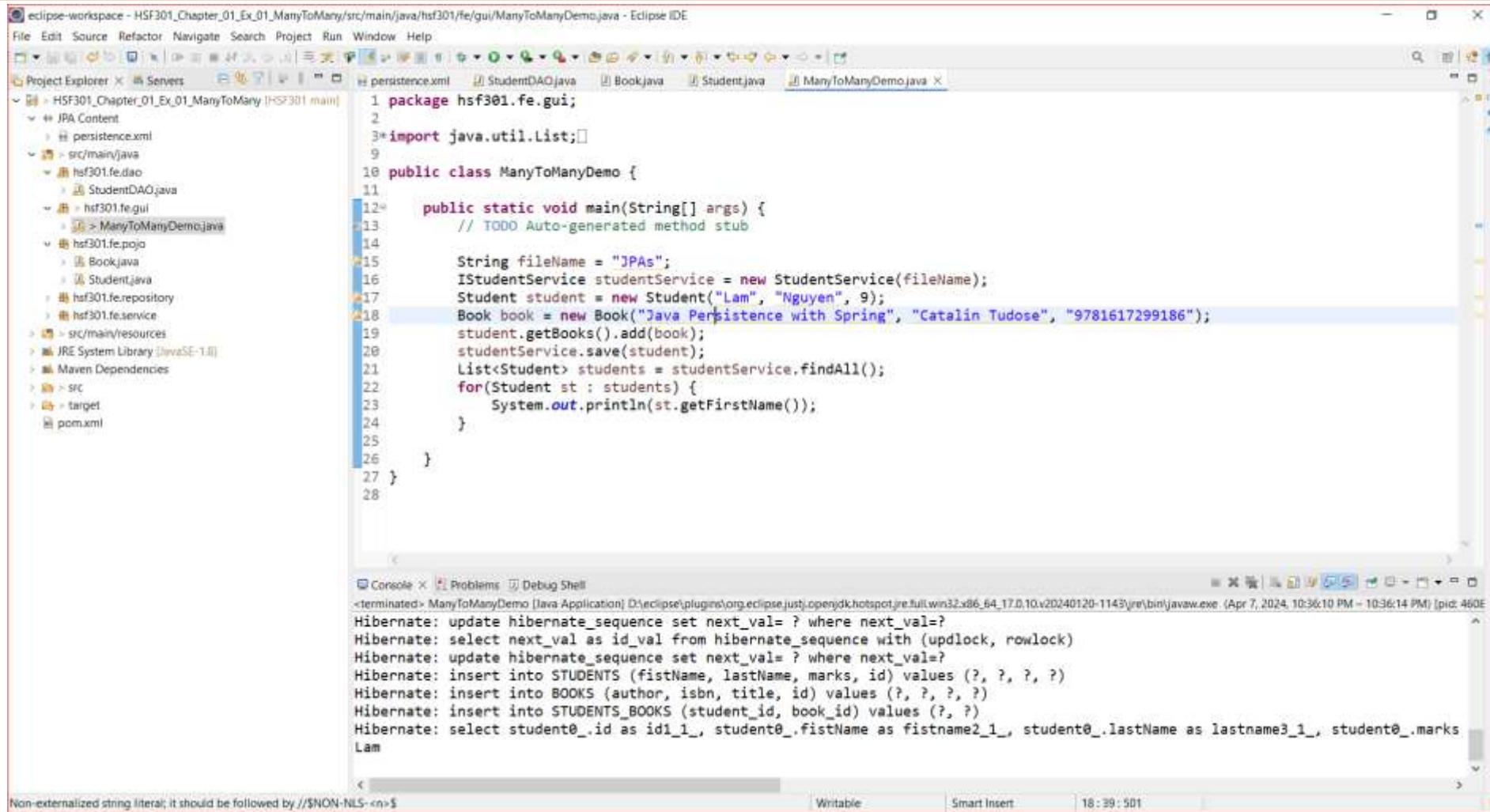


The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure: `HSF301_Chapter_01_Ex_01_ManyToMany` (HSF301 main), which contains JPA Content (persistence.xml), src/main/java (hsf301.fe.dao, hsf301.fe.gui, hsf301.fe.pojo, hsf301.fe.repository, hsf301.fe.service), src/main/resources, JRE System Library [JavaSE-1.8], Maven Dependencies, and SRC.
- Editor:** Displays the `Student.java` file content, which defines a POJO class `Student` with attributes `id`, `firstName`, `lastName`, and `marks`, and a many-to-many relationship `books` with cascade type `MERGE`.
- Bottom Status Bar:** Shows "Writable" and "Smart Insert".
- Bottom Navigation:** Shows tabs for Console, Problems, and Debug Shell, with a note: "No consoles to display at this time."

```
1 package hsf301.fe.pojo;
2
3 import java.util.HashSet;
4
5
6 @Entity
7 @Table(name = "STUDENTS")
8 public class Student {
9
10     @Id
11     @GeneratedValue(strategy = GenerationType.AUTO)
12     @Column(name = "id")
13     private int id;
14
15     @Column(name = "firstName", nullable = false, unique = false)
16     private String firstName;
17
18     @Column(name = "lastName")
19     private String lastName;
20
21     @Column(name = "marks")
22     private int marks;
23
24     @ManyToMany(cascade = CascadeType.MERGE)
25     @JoinTable(name = "STUDENTS_BOOKS",
26     joinColumns = @JoinColumn(name = "student_id"),
27     inverseJoinColumns = @JoinColumn(name = "book_id"))
28     private Set<Book> books = new HashSet<Book>();
29
30     public Set<Book> getBooks() {
31         return books;
32     }
33
34     public void setBooks(Set<Book> books) {
35         this.books = books;
36     }
37 }
```

### 3. Run Program



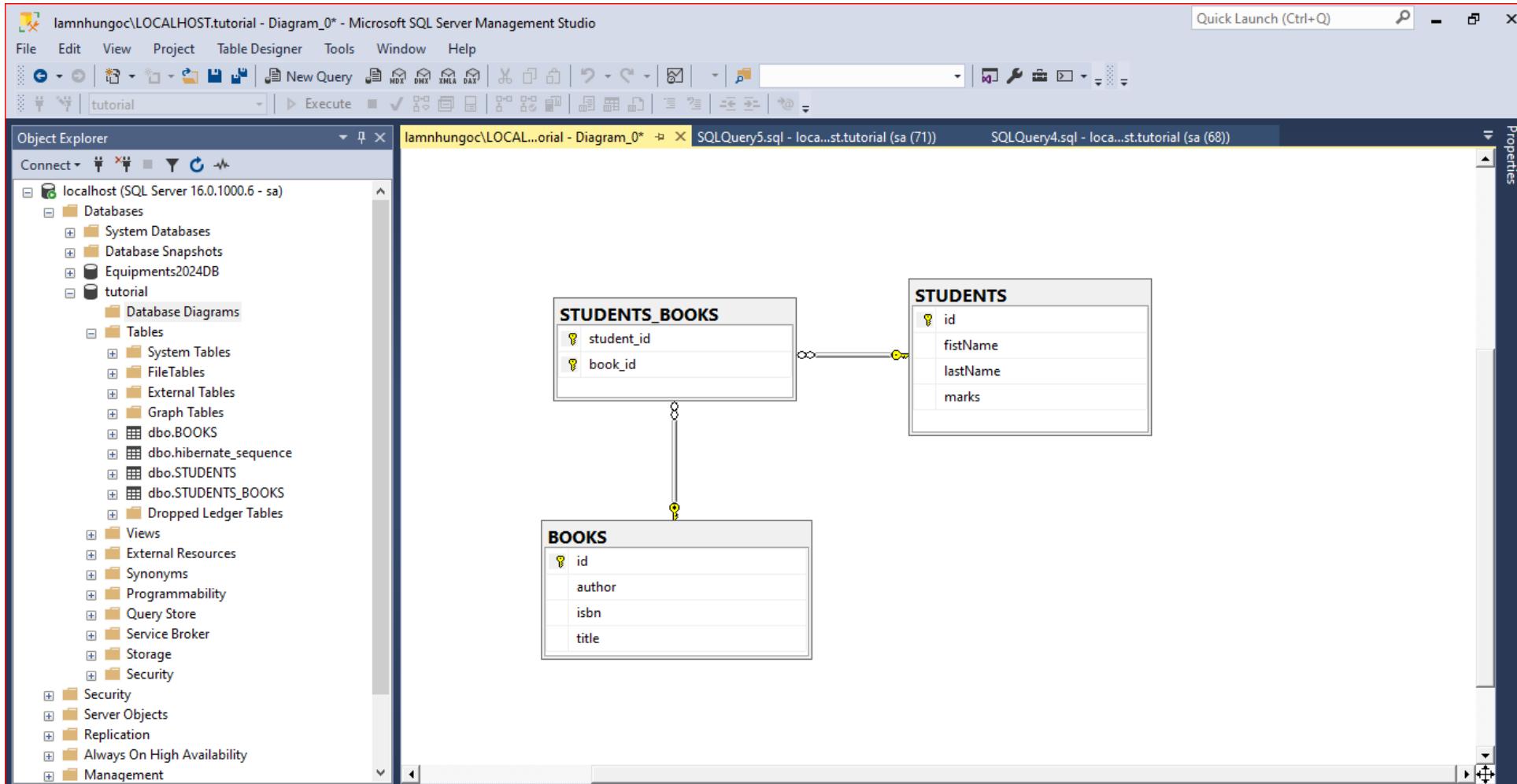
The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Displays the project structure for "HSF301\_Chapter\_01\_Ex\_01\_ManyToMany". It includes JPA Content, persistence.xml, src/main/java (containing hsf301.fe.dao.StudentDAO.java, hsf301.fe.gui.ManyToManyDemo.java, hsf301.fe.pojo.Book.java, hsf301.fe.pojo.Student.java, hsf301.fe.repository, and hsf301.fe.service), src/main/resources, JRE System Library [JavaSE-1.8], Maven Dependencies, SRC, target, and pom.xml.
- Editor:** Shows the code for ManyToManyDemo.java. The code creates a Student object with name "Lam" and last name "Nguyen", adds a Book object ("Java Persistence with Spring" by Catalin Tudose) to its books list, and then saves the student to a database.
- Console:** Displays the execution output of the program. It shows Hibernate SQL statements for updating the hibernate\_sequence table, inserting data into STUDENTS and BOOKS tables, and selecting student information. The output ends with the name "Lam".

```
1 package hsf301.fe.gui;
2
3 import java.util.List;
4
5 public class ManyToManyDemo {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9
10        String fileName = "JPAs";
11        IStudentService studentService = new StudentService(fileName);
12        Student student = new Student("Lam", "Nguyen", 9);
13        Book book = new Book("Java Persistence with Spring", "Catalin Tudose", "9781617299186");
14        student.getBooks().add(book);
15        studentService.save(student);
16        List<Student> students = studentService.findAll();
17        for(Student st : students) {
18            System.out.println(st.getFirstName());
19        }
20    }
21
22 }
23
24
25
26
27 }
28

Console < Problems < Debug Shell
<terminated> ManyToManyDemo [Java Application] D:\eclipse\plugins\org.eclipse.jdt.core\openjdk\hotspot\ref\full\win32\x86_64_17.0.10.v20240120-1143\jre\bin\javaw.exe (Apr 7, 2024, 10:36:10 PM - 10:36:14 PM) [pid: 460E]
Hibernate: update hibernate_sequence set next_val=? where next_val=?
Hibernate: select next_val as id_val from hibernate_sequence with (updlock, rowlock)
Hibernate: update hibernate_sequence set next_val=? where next_val=?
Hibernate: insert into STUDENTS (firstName, lastName, marks, id) values (?, ?, ?, ?)
Hibernate: insert into BOOKS (author, isbn, title, id) values (?, ?, ?, ?)
Hibernate: insert into STUDENTS_BOOKS (student_id, book_id) values (?, ?)
Hibernate: select student0_.id as id1_1_, student0_.firstName as firstname2_1_, student0_.lastName as lastname3_1_, student0_.marks
Lam
```

## 4. Result



# 5. Result

The image displays three separate windows of Microsoft SQL Server Management Studio (SSMS) showing the results of a database query. Each window has a title bar indicating the connection is to 'localhost (16.0 RTM) - sa (53) - tutorial'.

**Top Window:** Shows the results of the following query:

```
SELECT TOP (1000) [student_id]
FROM [tutorial].[dbo].[STUDENTS_BOOKS]
```

The results show two rows:

student_id	book_id
1	2

**Middle Window:** Shows the results of the following query:

```
SELECT TOP (1000) [id]
FROM [author].[dbo].[BOOKS]
```

The results show one row:

id	author	isbn	title	student_id
1	Catalin Tudose	9781617299185	Java Persistence with Spring	1

**Bottom Window:** Shows the results of the following query:

```
SELECT id, surname, lastname, marks
FROM [tutorial].[dbo].[STUDENTS]
```

The results show one row:

id	surname	lastname	marks
1	Lam	Nguyen	9

# 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