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

FPT Education
FPT UNIVERSITY

Java

Object Relational Mapping (ORM)

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 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.

FPT Education
FPT UNIVERSITY

Java

Java Persistence API (JPA)

What is JPA ?

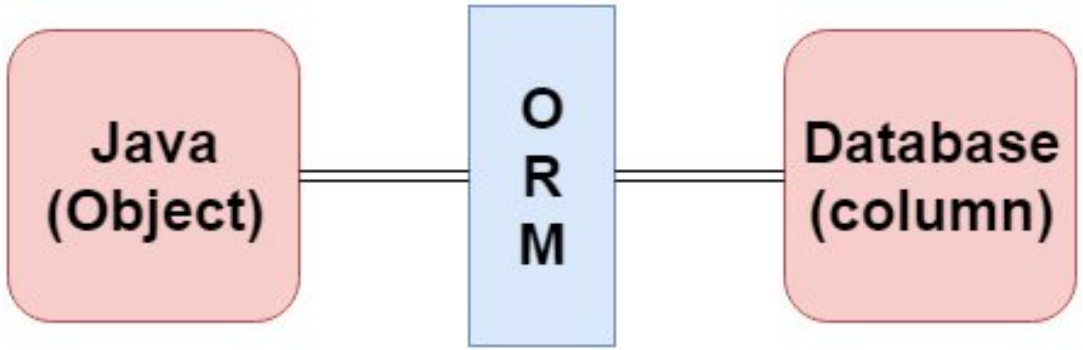
- JPA stands for Java Persistence API. It is a Java programming interface that allows developers to manage relational data in Java applications using object oriented methodologies.
- JPA is a specification for managing relational data in Java applications. It provides a set of interfaces, annotations, 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.

JPA Object Relational Mapping

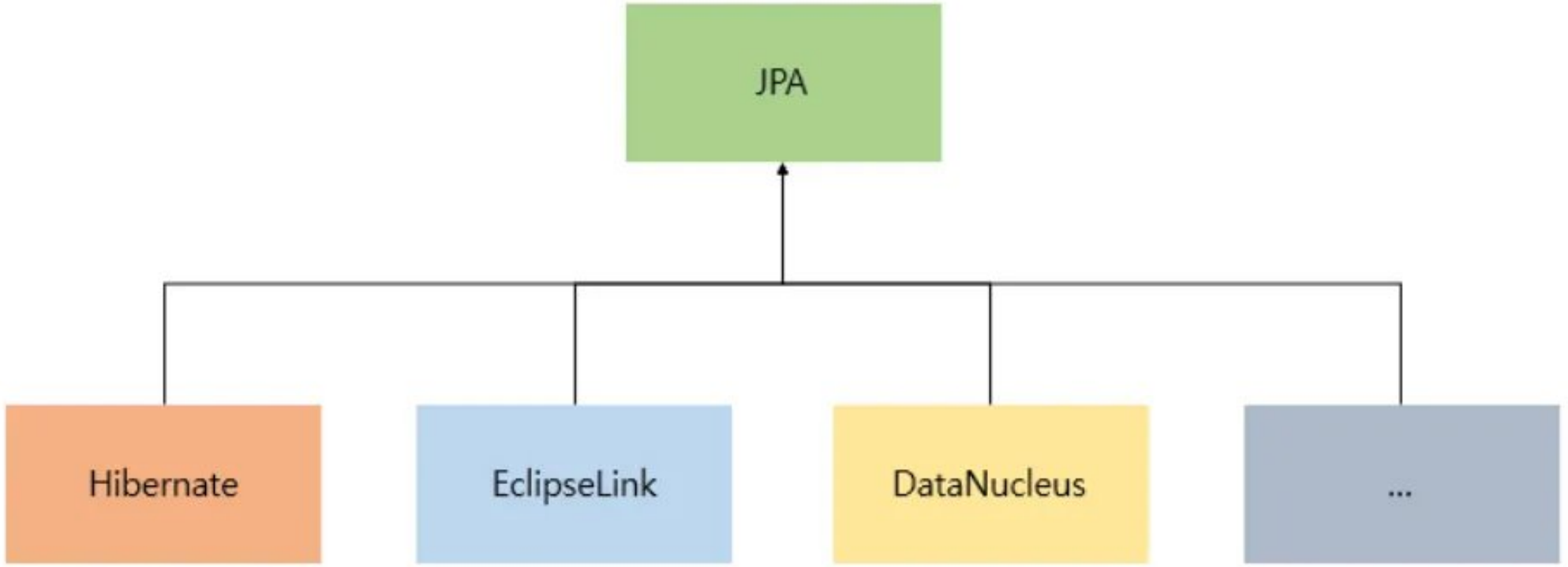
- ORM makes it easier to work with databases in object-oriented applications, allowing you to focus on the business logic rather than the underlying data storage details.



Benefits of 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.

JPA Implementation Options



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.

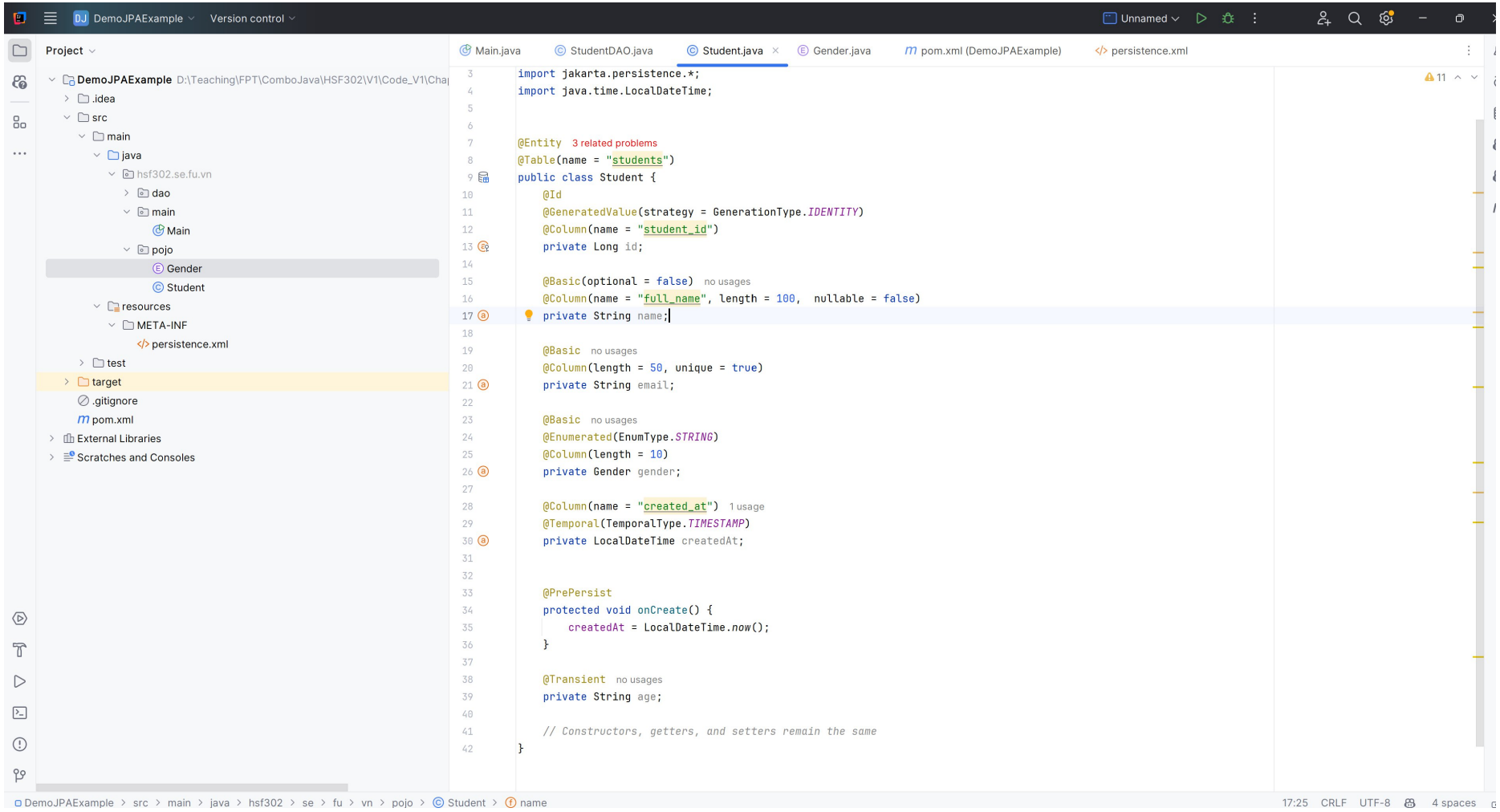
Mapping Annotations in JPA

- **@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.

Example

```
package hsf302.se.fu.vn.pojo;

public enum Gender {
    MALE,
    FEMALE,
    OTHER
}
```



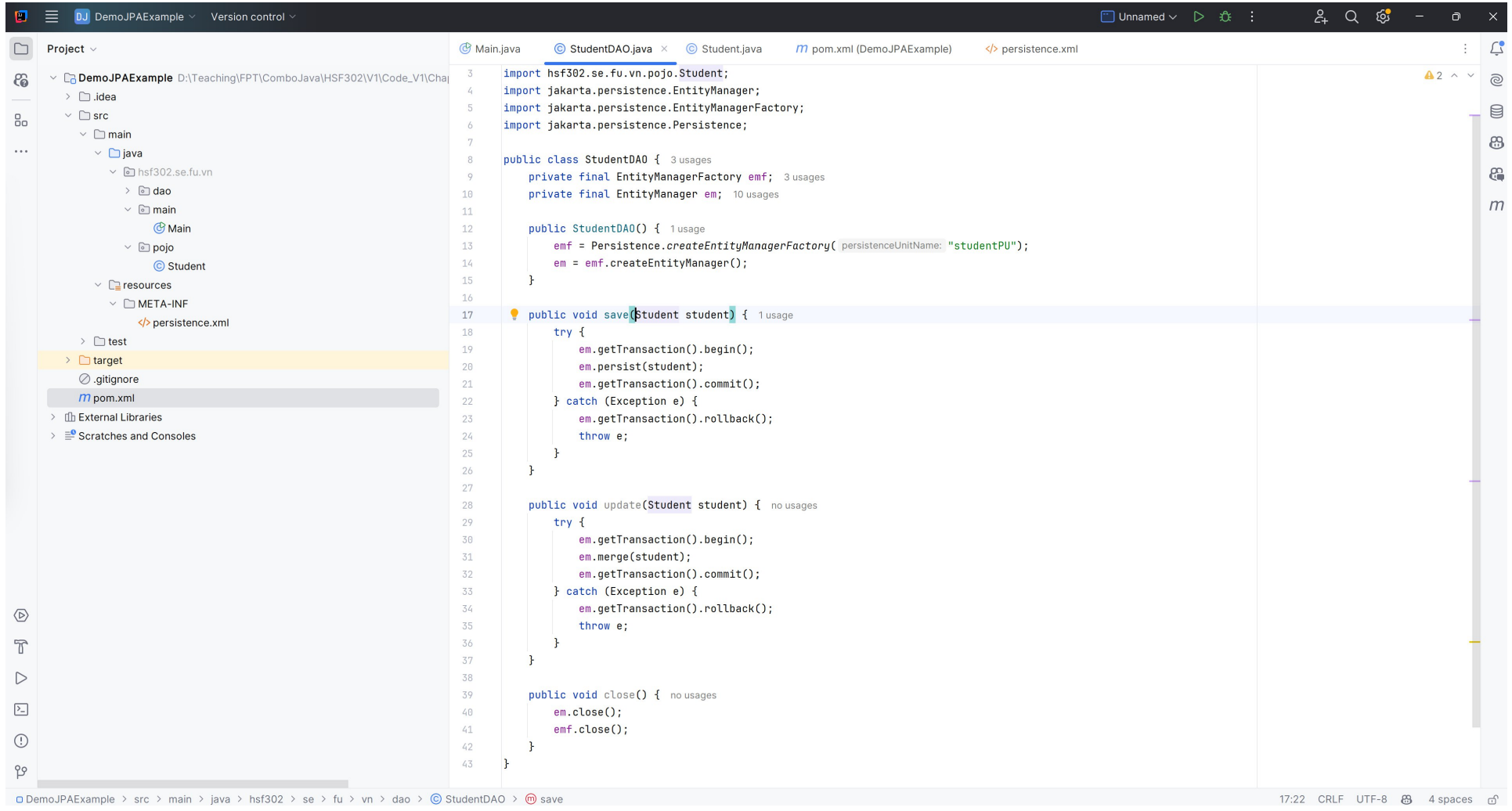
JPA Entity Manager

- 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**

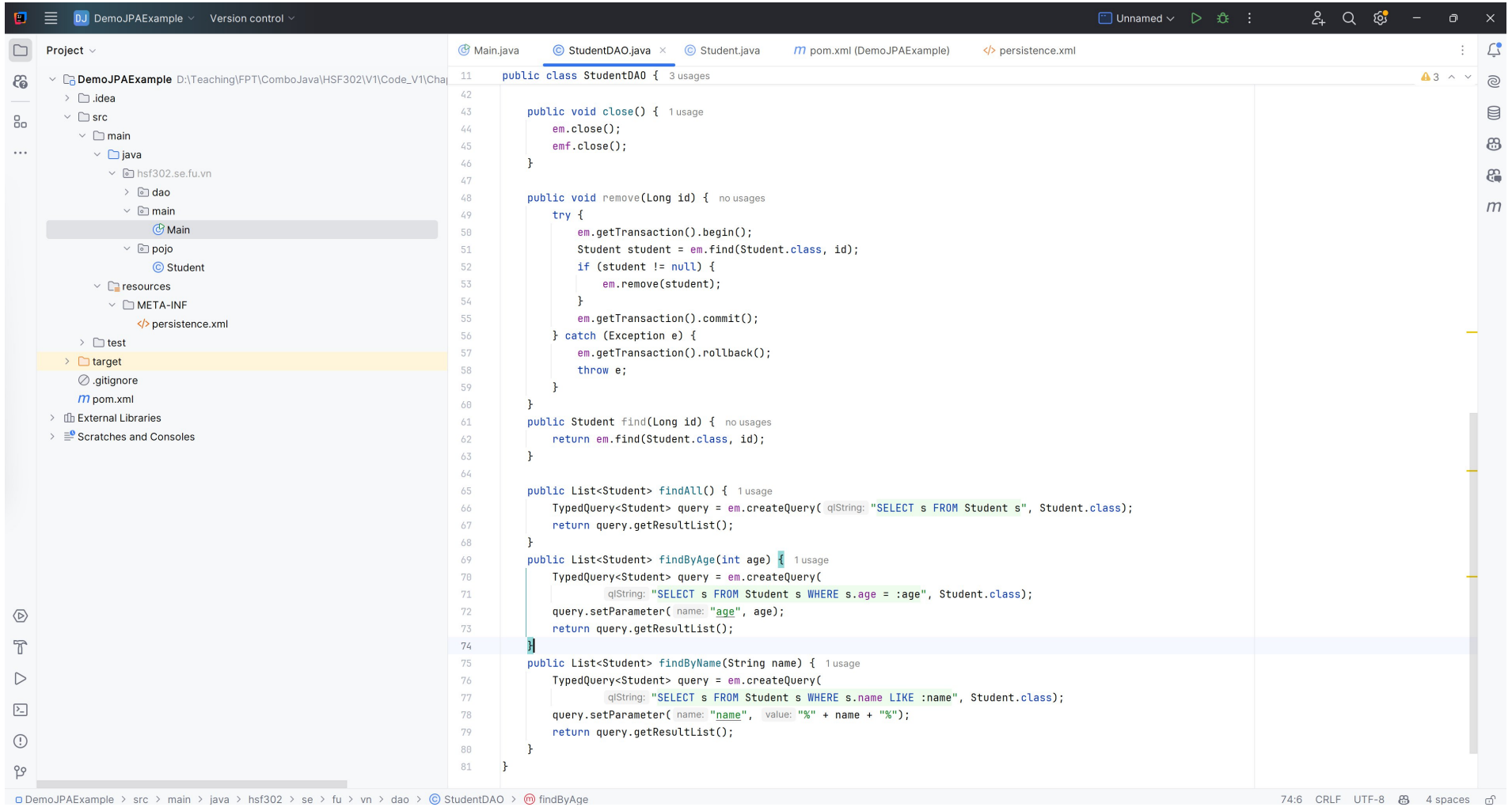
Key Operations of the Entity Manager

- **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.

Example


DemoJPAExample
Version control
Unnamed
Project
DemoJPAExample D:\Teaching\FPT\ComboJava\HSF302\V1\Code_V1\Cha
.idea
src
main
java
hsf302.se.fu.vn
dao
main
Main
pojo
Student
resources
META-INF
persistence.xml
test
target
.gitignore
pom.xml
External Libraries
Scratches and Consoles
Main.java
StudentDAO.java
Student.java
pom.xml (DemoJPAExample)
persistence.xml
import hsf302.se.fu.vn.pojo.Student;
import jakarta.persistence.EntityManager;
import jakarta.persistence.EntityManagerFactory;
import jakarta.persistence.Persistence;
public class StudentDAO { 3 usages
private final EntityManagerFactory emf; 3 usages
private final EntityManager em; 10 usages
public StudentDAO() { 1 usage
emf = Persistence.createEntityManagerFactory(persistenceUnitName: "studentPU");
em = emf.createEntityManager();
}
public void save(Student student) { 1 usage
try {
em.getTransaction().begin();
em.persist(student);
em.getTransaction().commit();
} catch (Exception e) {
em.getTransaction().rollback();
throw e;
}
}
public void update(Student student) { no usages
try {
em.getTransaction().begin();
em.merge(student);
em.getTransaction().commit();
} catch (Exception e) {
em.getTransaction().rollback();
throw e;
}
}
public void close() { no usages
em.close();
emf.close();
}
}
DemoJPAExample > src > main > java > hsf302 > se > fu > vn > dao > StudentDAO > save
17:22 CRLF UTF-8 4 spaces

Example


DemoJPAExample
Version control
Unnamed
Project
DemoJPAExample D:\Teaching\FPT\ComboJava\HSF302\V1\Code_V1\Cha
.idea
src
main
java
hsf302.se.fu.vn
dao
main
Main
pojo
Student
resources
META-INF
persistence.xml
test
target
.gitignore
pom.xml
External Libraries
Scratches and Consoles
Main.java
StudentDAO.java
Student.java
pom.xml (DemoJPAExample)
persistence.xml
public class StudentDAO { 3 usages
public void close() { 1 usage
em.close();
emf.close();
}
public void remove(Long id) { no usages
try {
em.getTransaction().begin();
Student student = em.find(Student.class, id);
if (student != null) {
em.remove(student);
}
em.getTransaction().commit();
} catch (Exception e) {
em.getTransaction().rollback();
throw e;
}
}
public Student find(Long id) { no usages
return em.find(Student.class, id);
}
public List<Student> findAll() { 1 usage
TypedQuery<Student> query = em.createQuery(qlString: "SELECT s FROM Student s", Student.class);
return query.getResultList();
}
public List<Student> findByAge(int age) { 1 usage
TypedQuery<Student> query = em.createQuery(
qlString: "SELECT s FROM Student s WHERE s.age = :age", Student.class);
query.setParameter(name: "age", age);
return query.getResultList();
}
public List<Student> findByName(String name) { 1 usage
TypedQuery<Student> query = em.createQuery(
qlString: "SELECT s FROM Student s WHERE s.name LIKE :name", Student.class);
query.setParameter(name: "name", value: "%" + name + "%");
return query.getResultList();
}
}
DemoJPAExample > src > main > java > hsf302 > se > fu > vn > dao > StudentDAO > findByAge
74:6 CRLF UTF-8 4 spaces

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

FPT Education
FPT UNIVERSITY

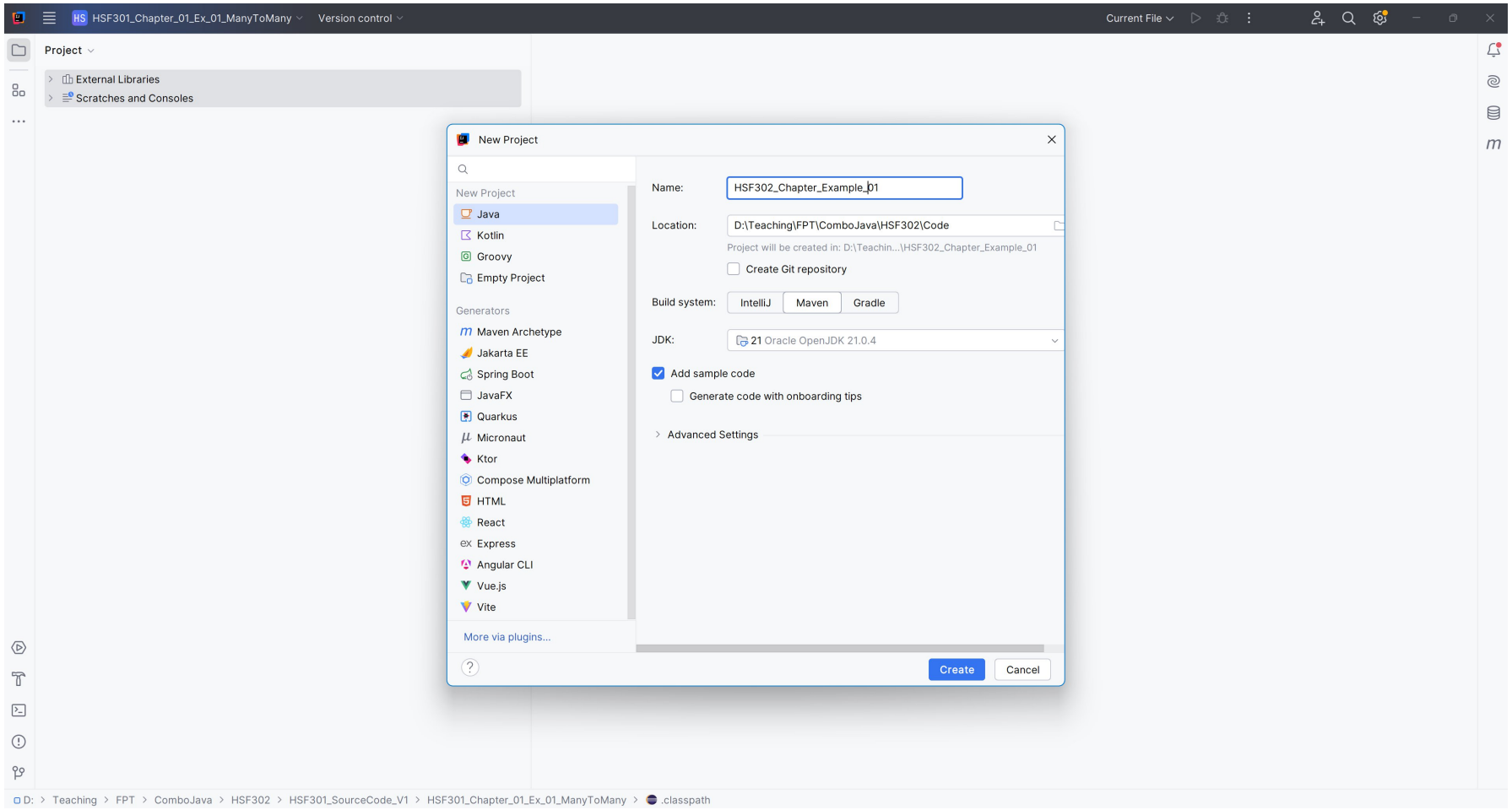
Java

Demo Simple CRUD with JPA

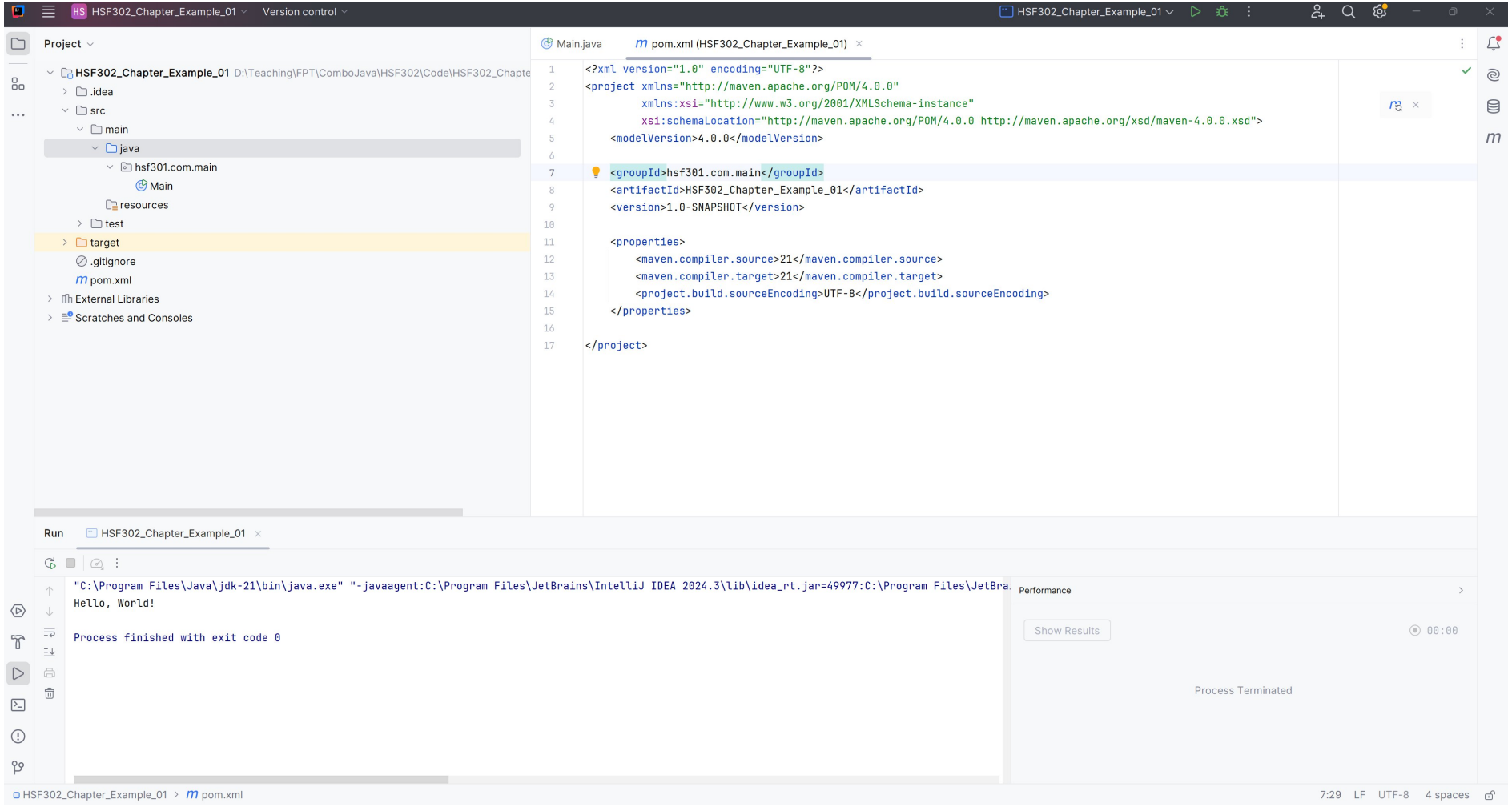
Requirements

- JDK : 21
- JPA : 3.1
- MSSQL : 11.2.3
- Hibernate: 6.5.2

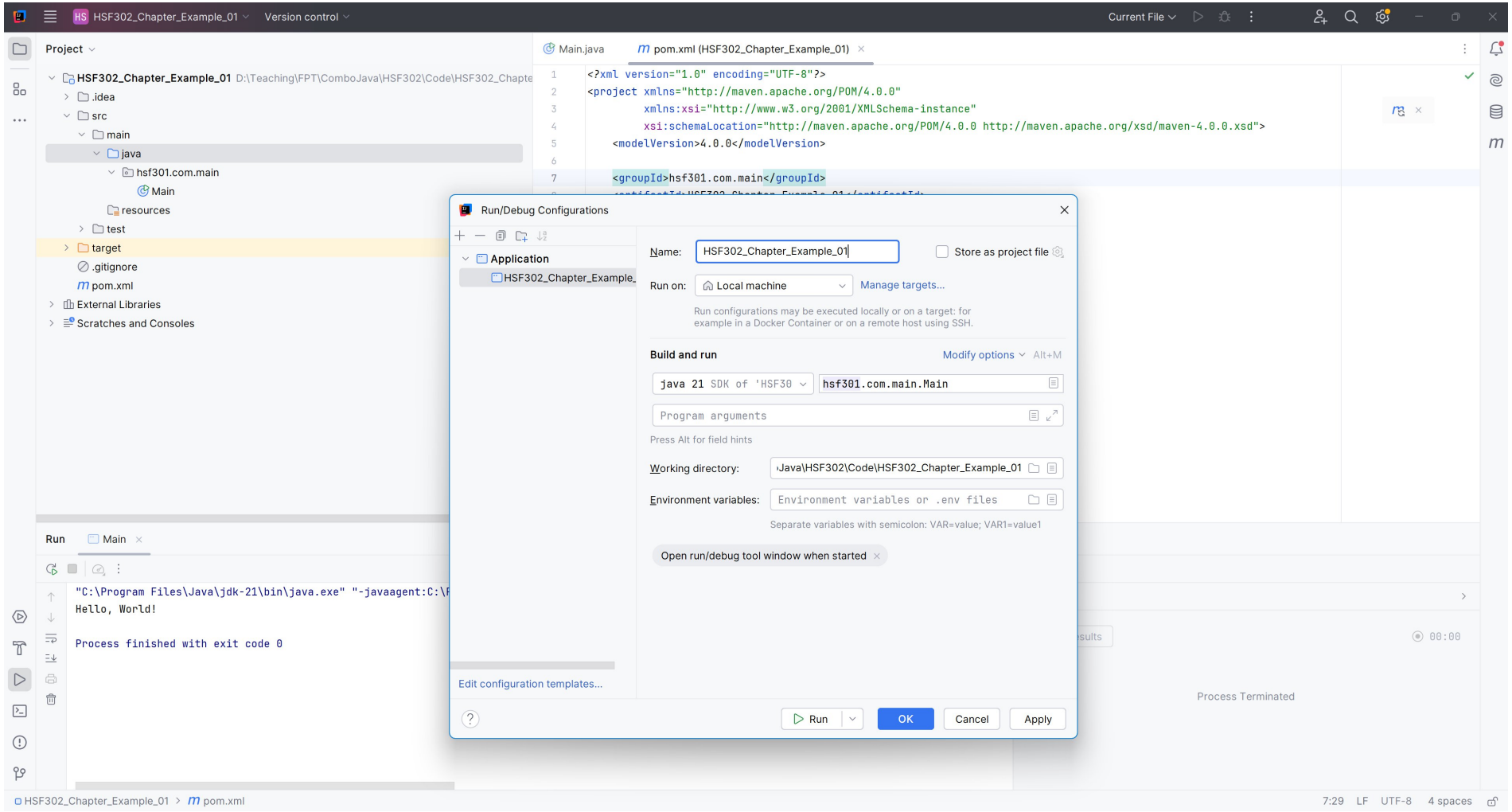
Open IntelliJ, File | New | Maven Project



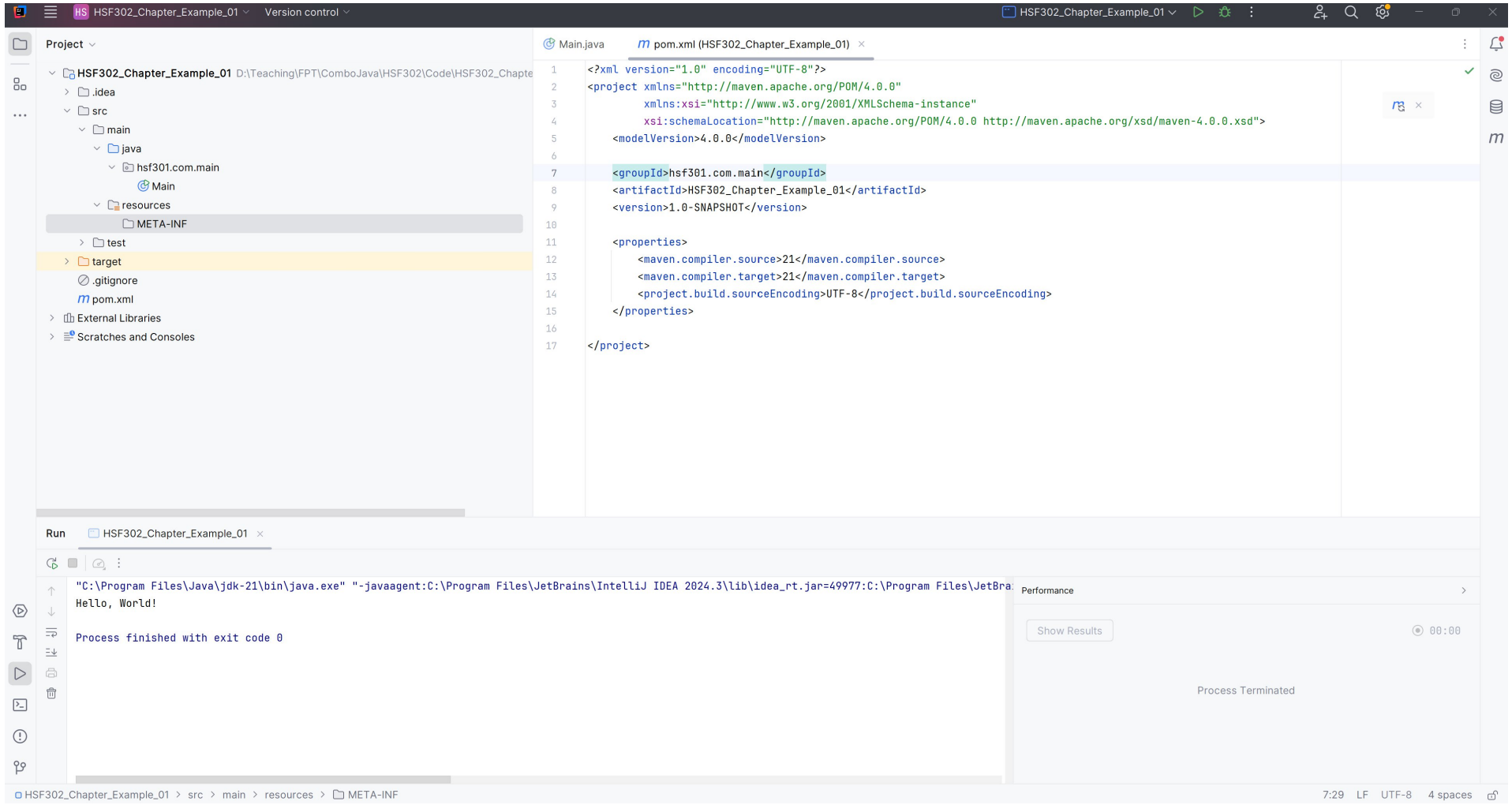
The New Project



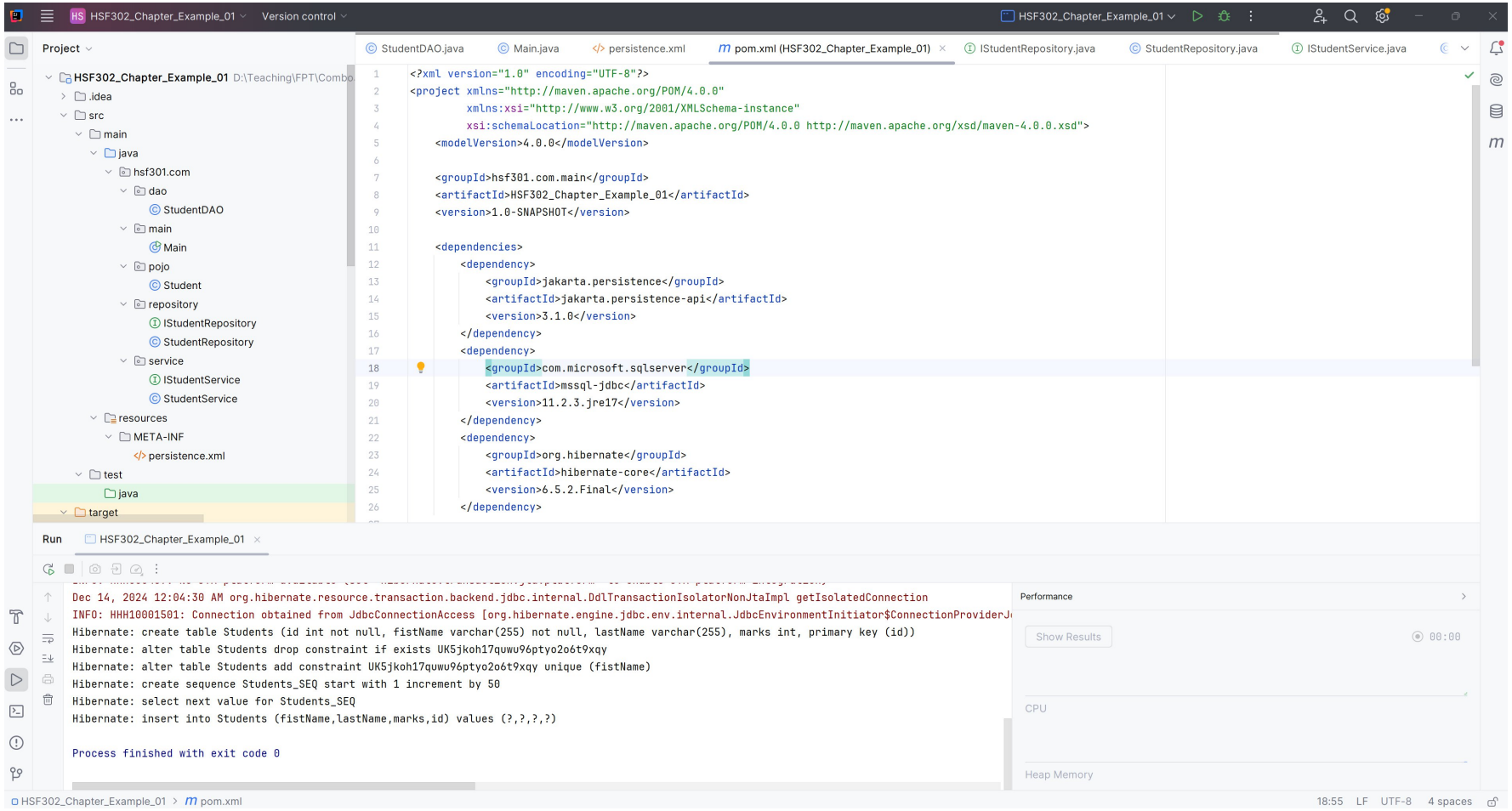
Add Run Configurations



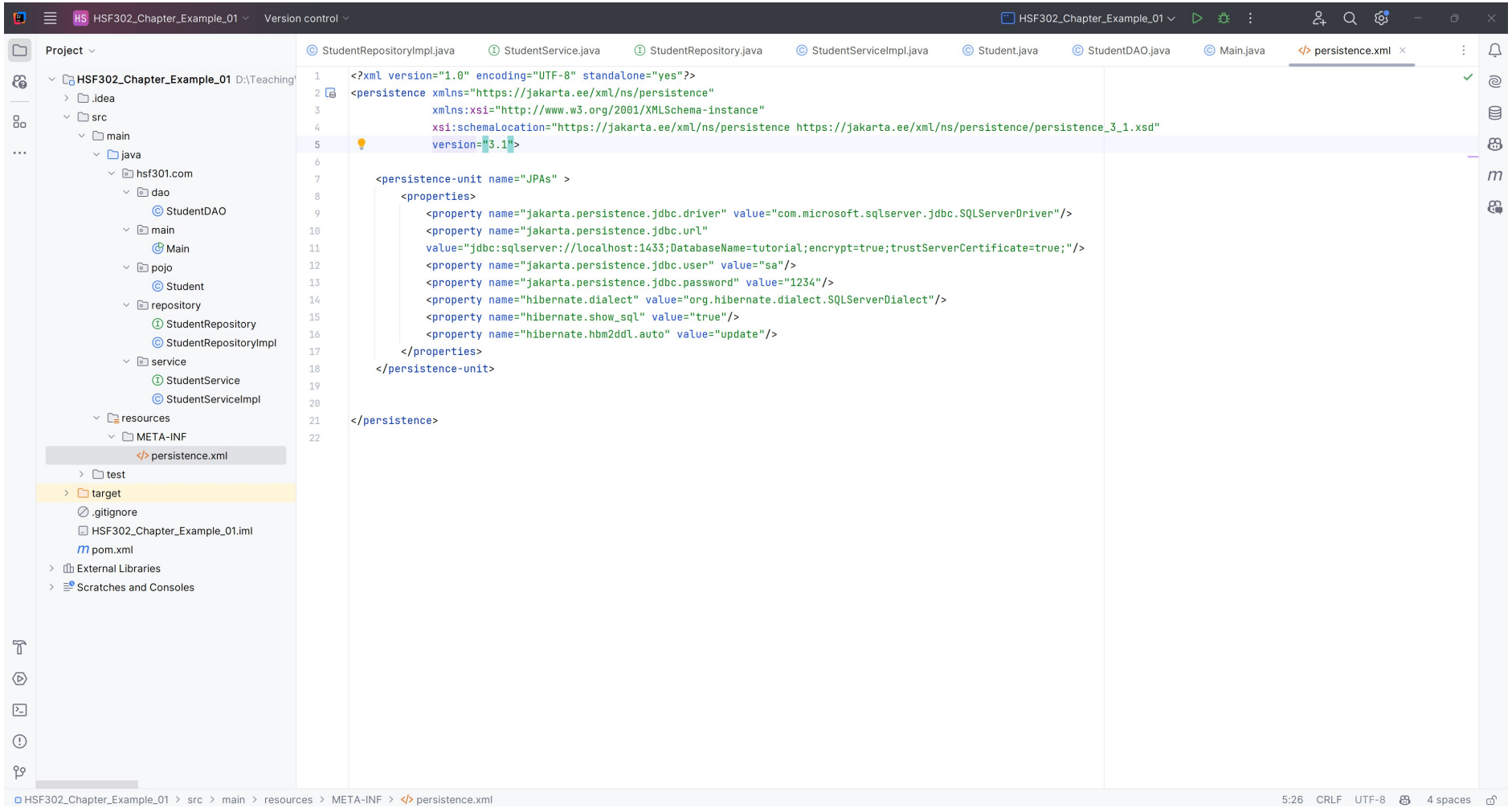
Create folder META-INF in src/main/resources



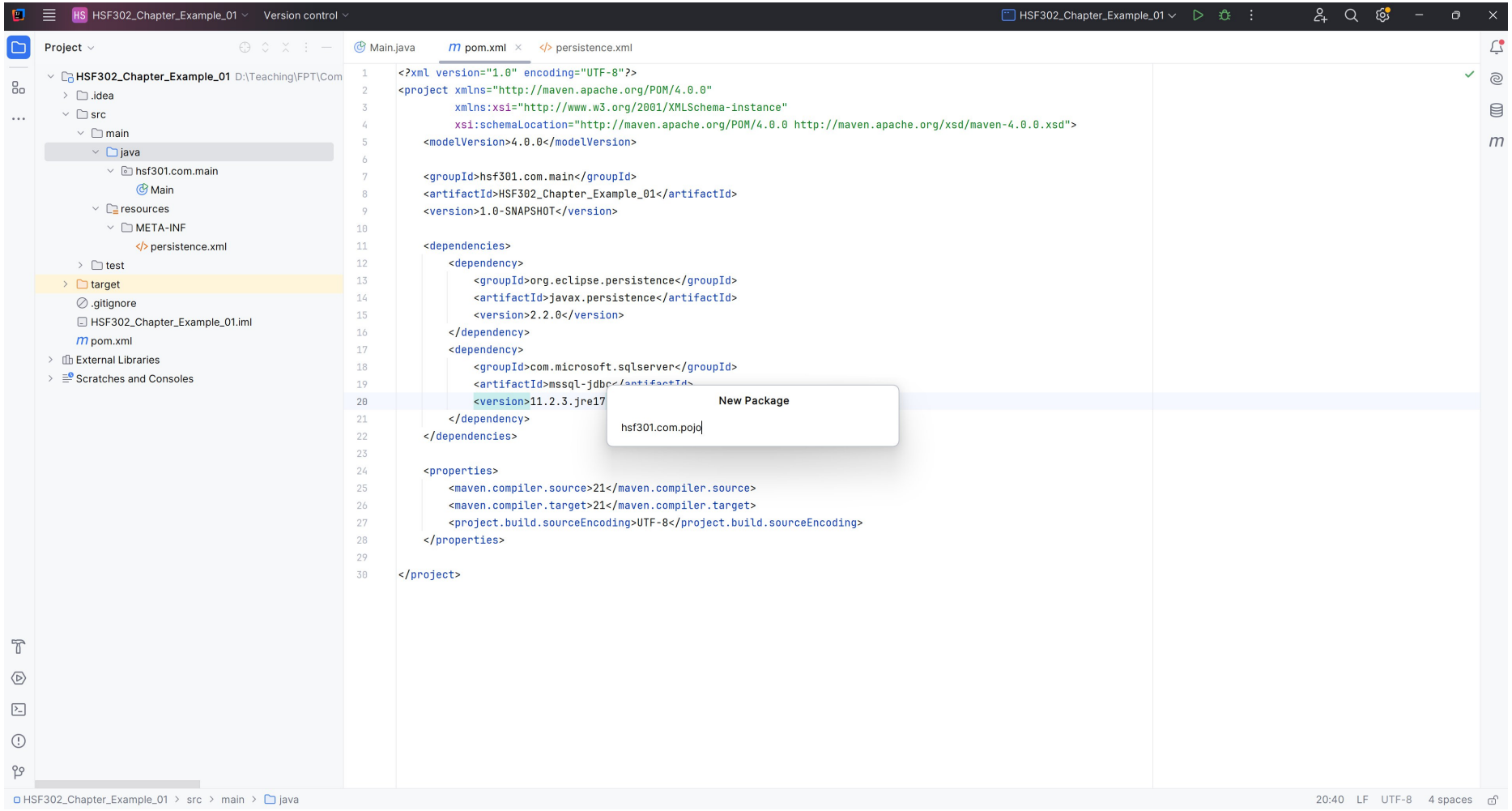
Edit the pom.xml



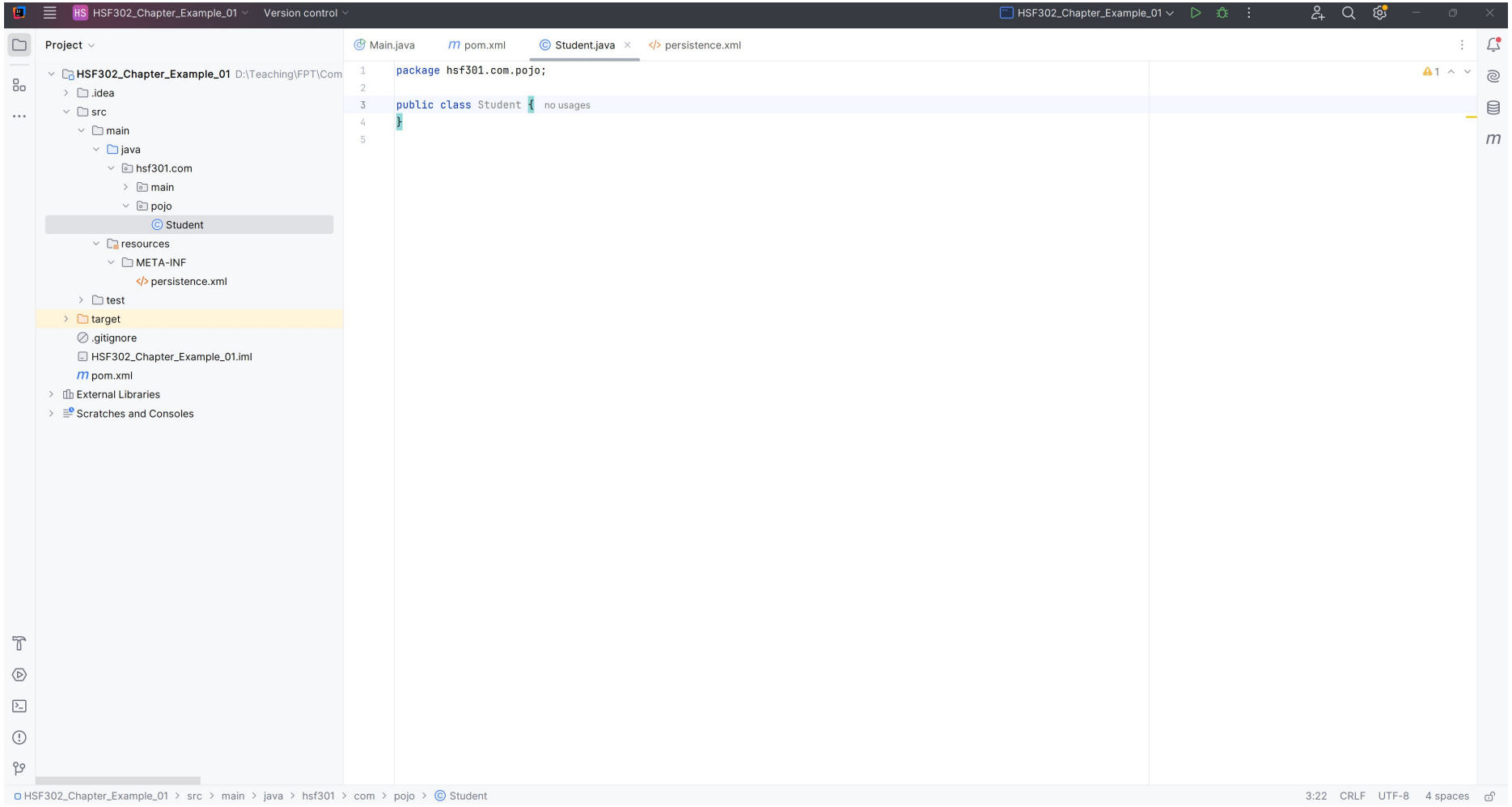
Create the persistence.xml in META-INF folder



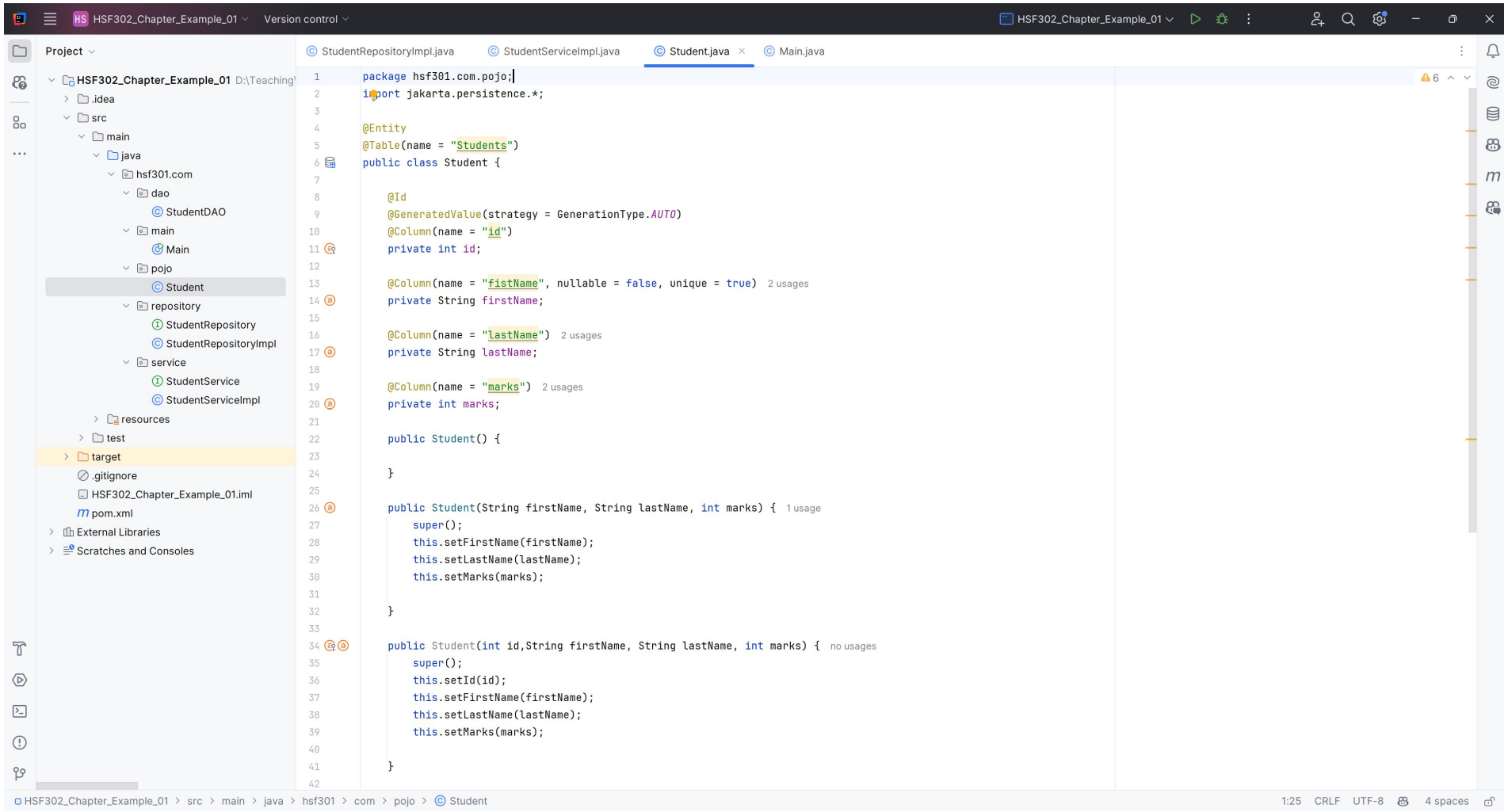
Add hsf301.com.pojo Package in src/main/java



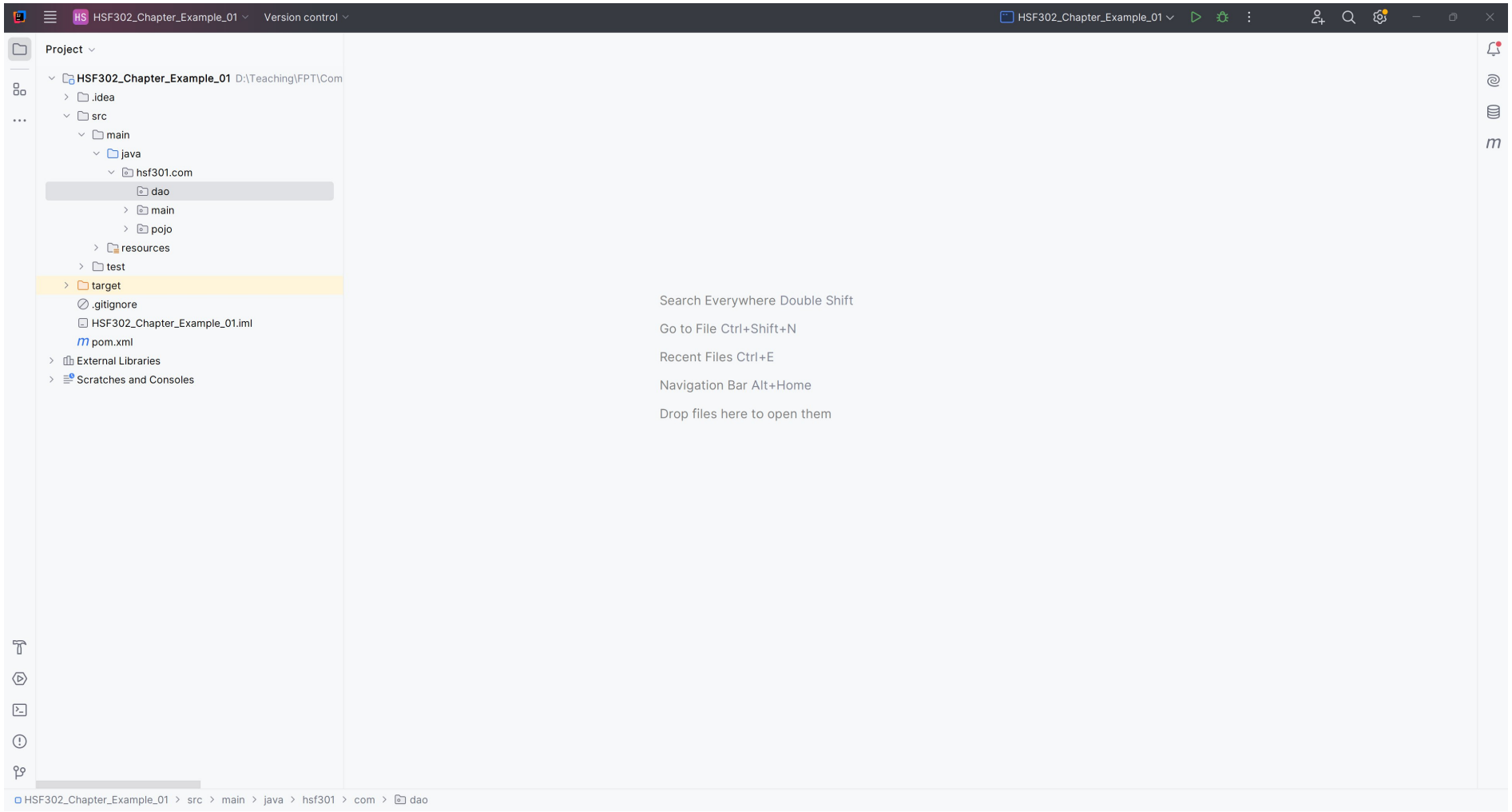
Add Student.java in hsf301.com.pojo package



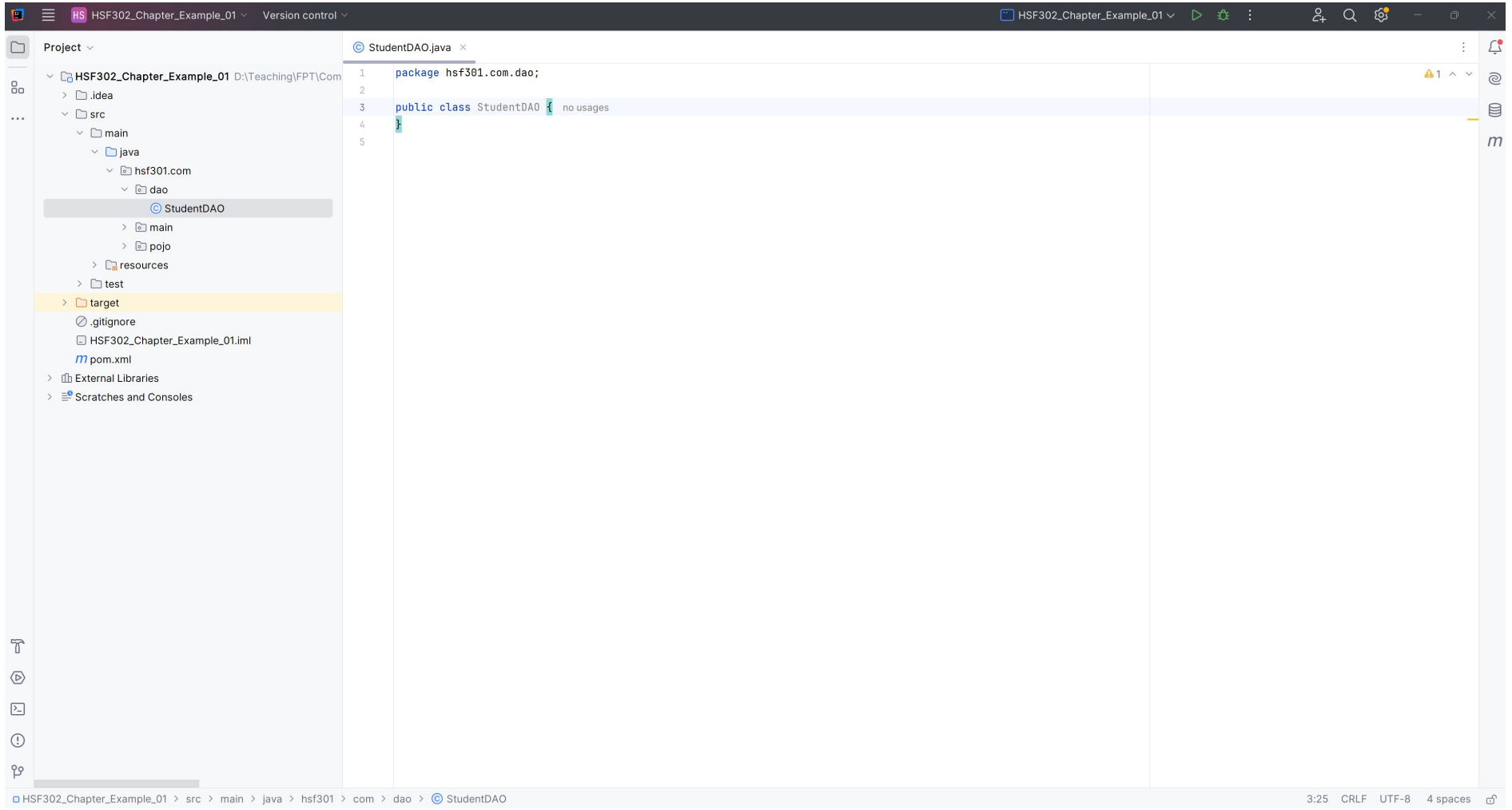
Edit the Student.java



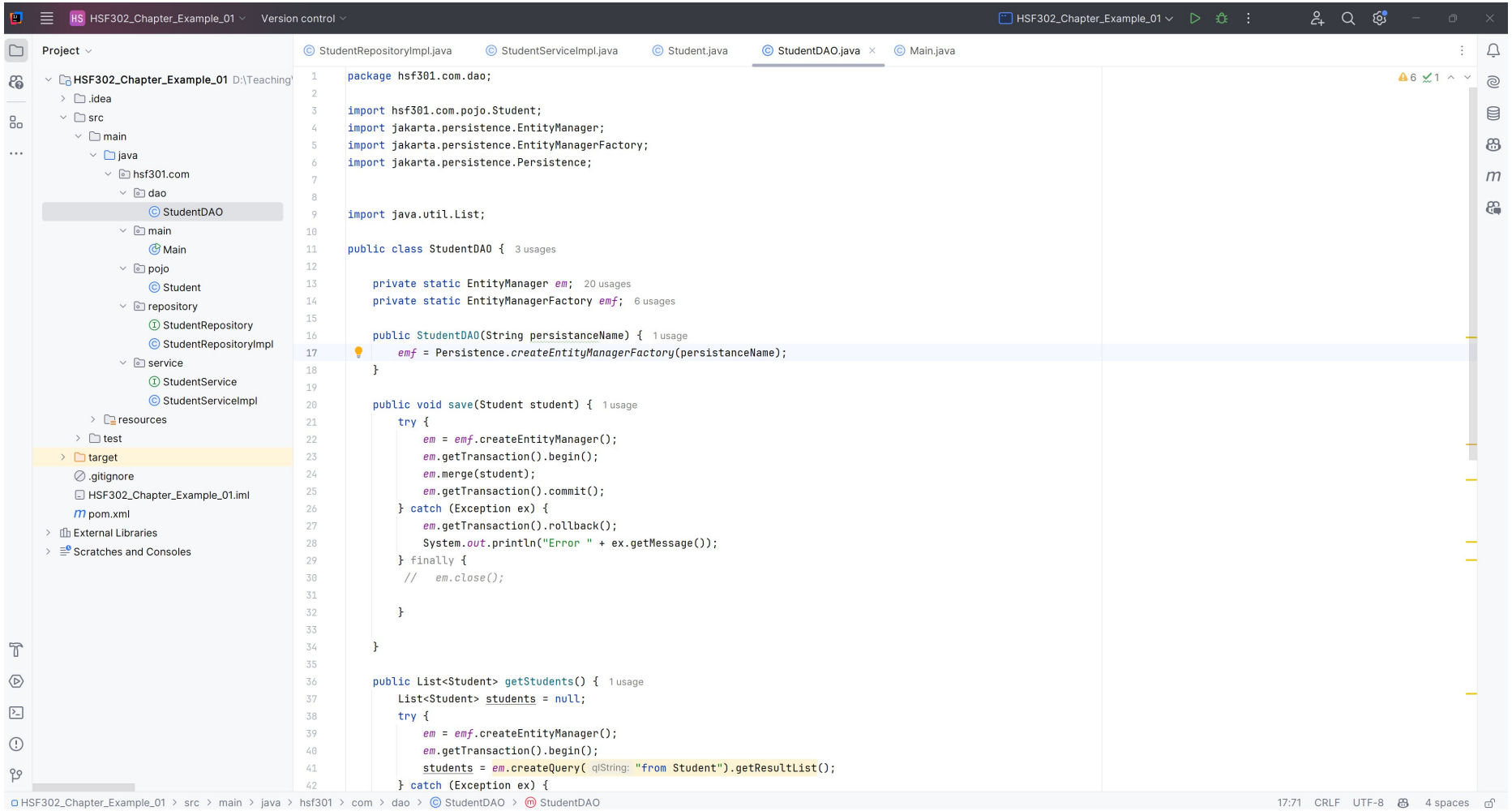
Add hsf301.com.dao Package in src/main/java



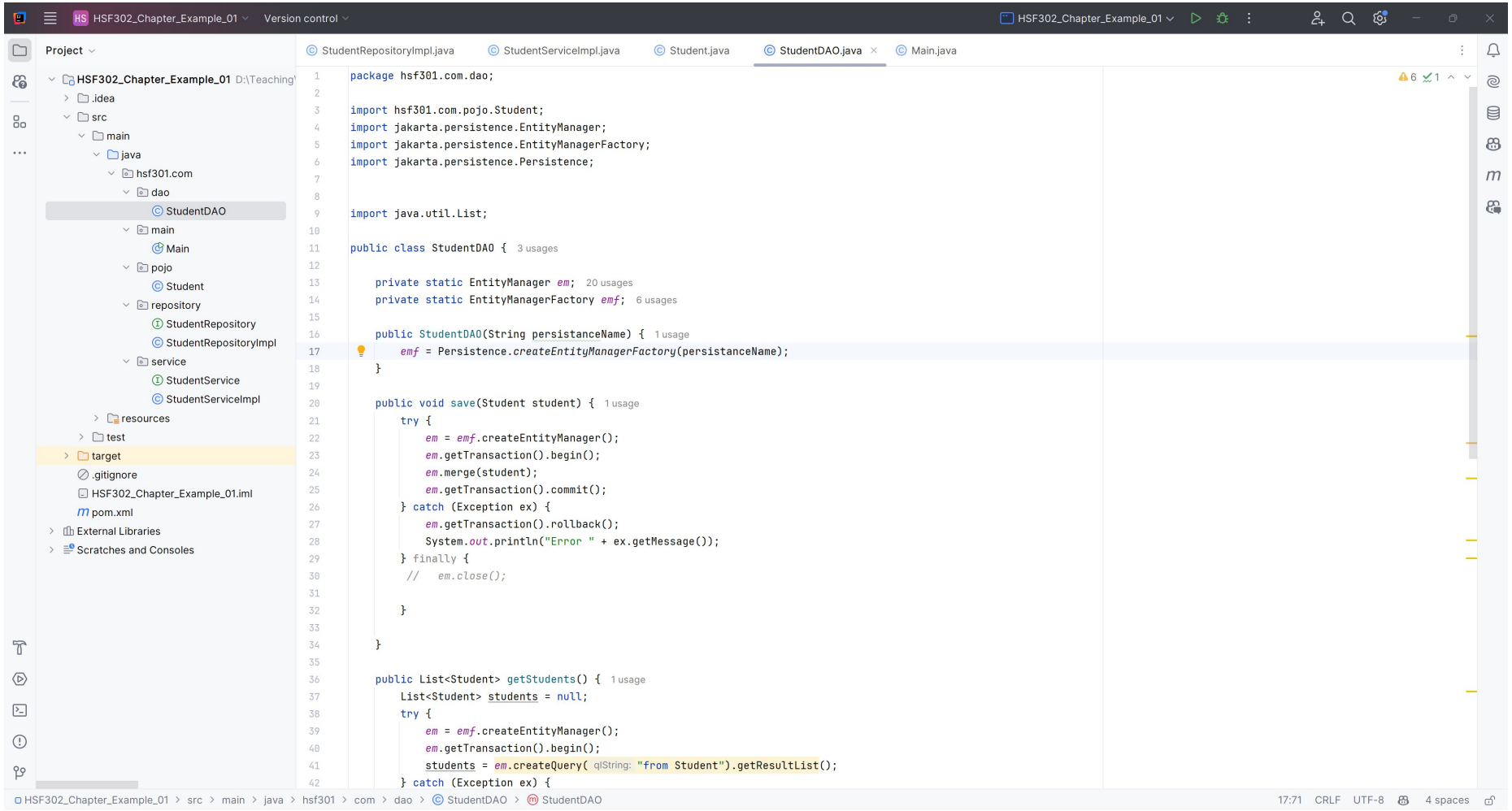
Create StudentDAO in hsf301.com.dao



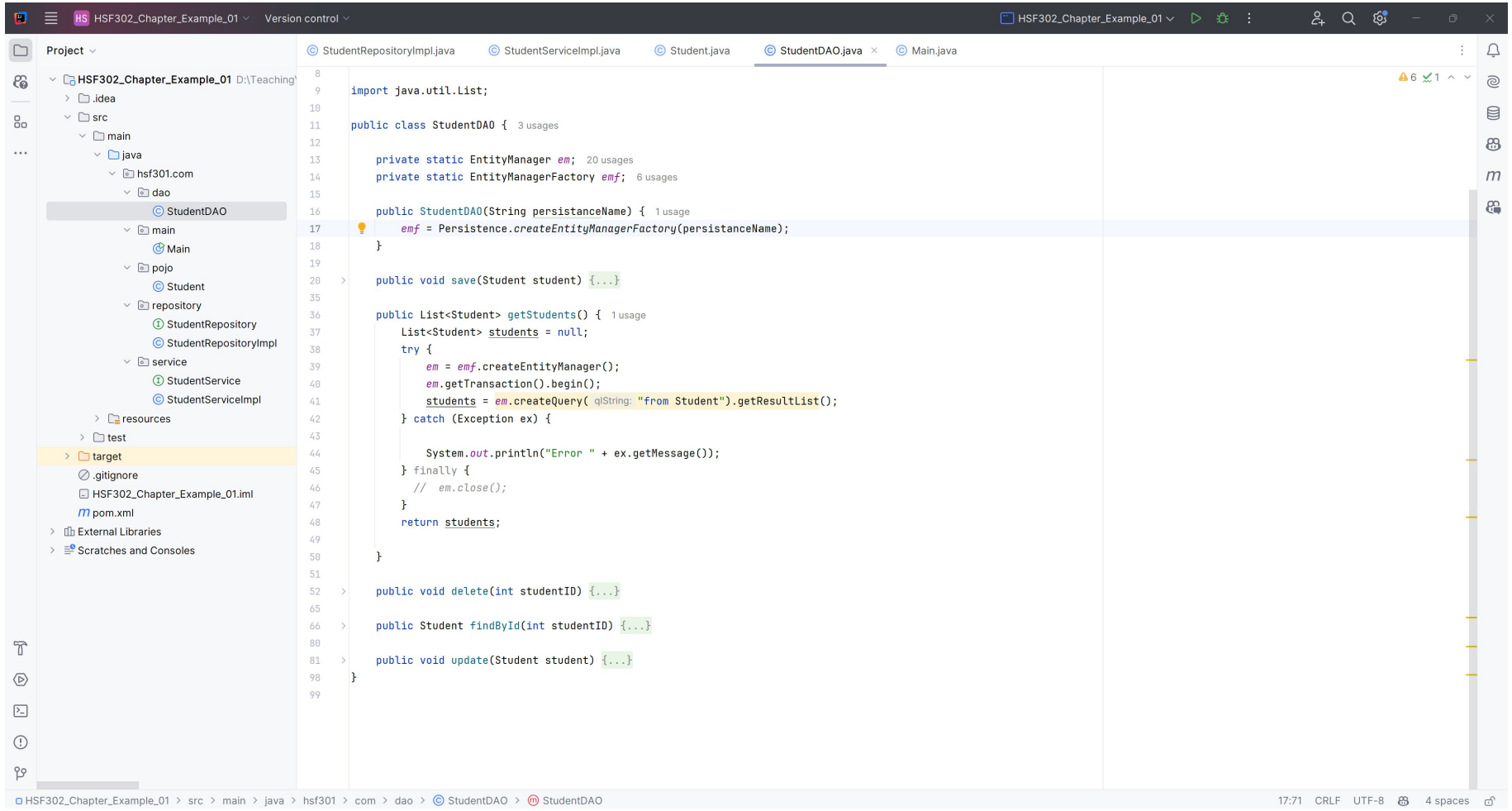
Edit the StudentDAO in hsf301.com.dao



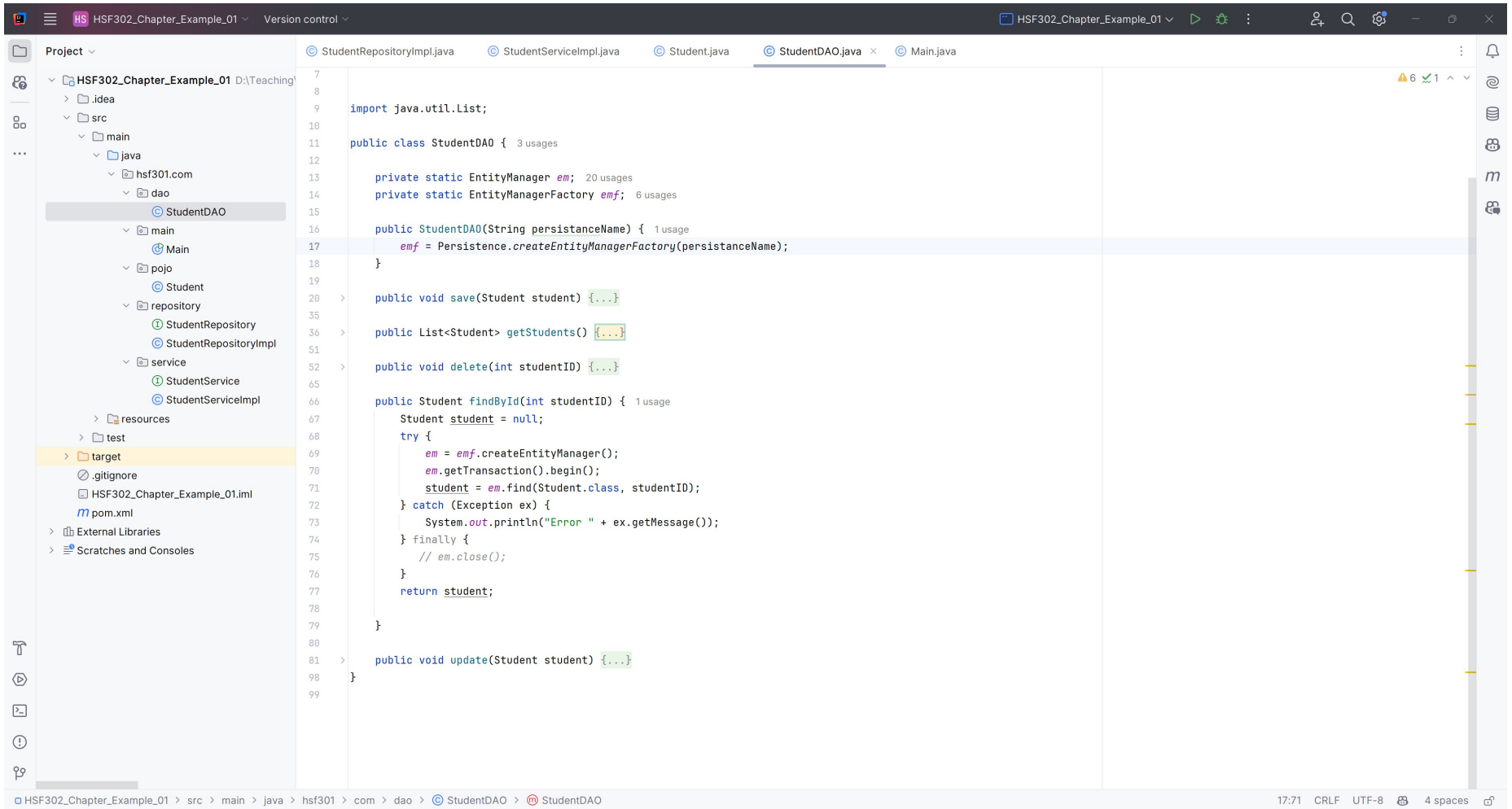
Save Student Method



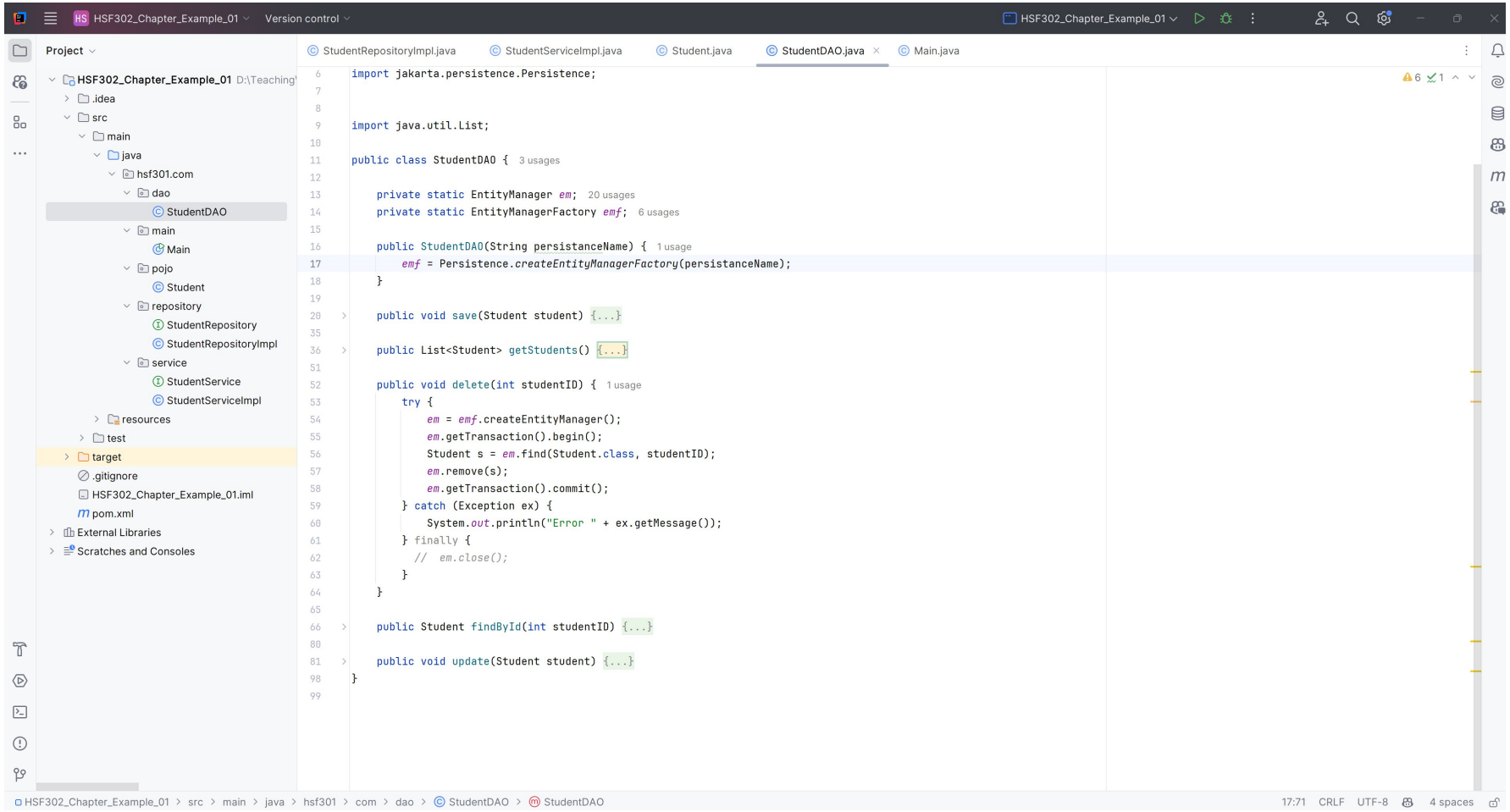
Get All Students Method



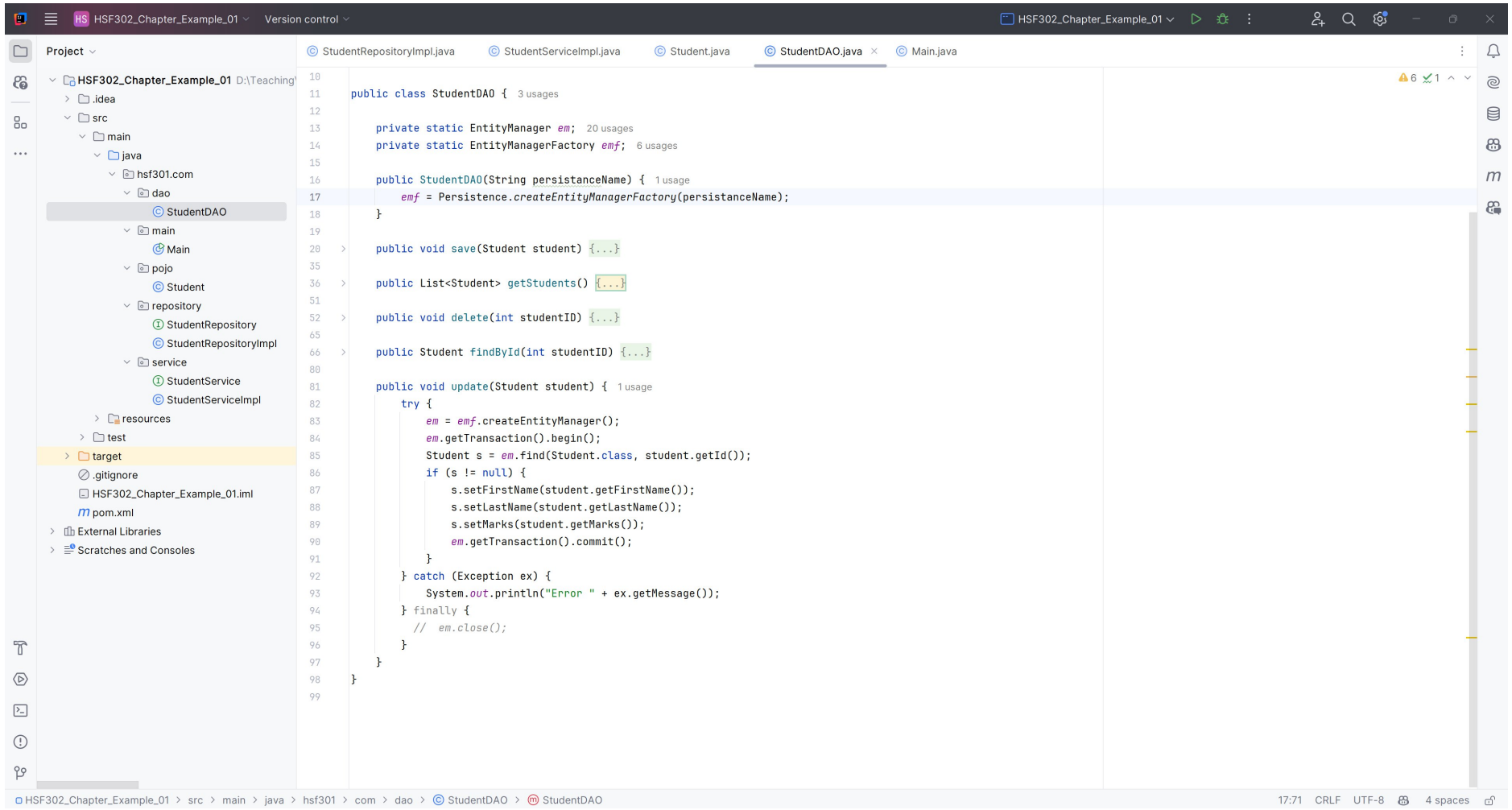
Find a Student Method



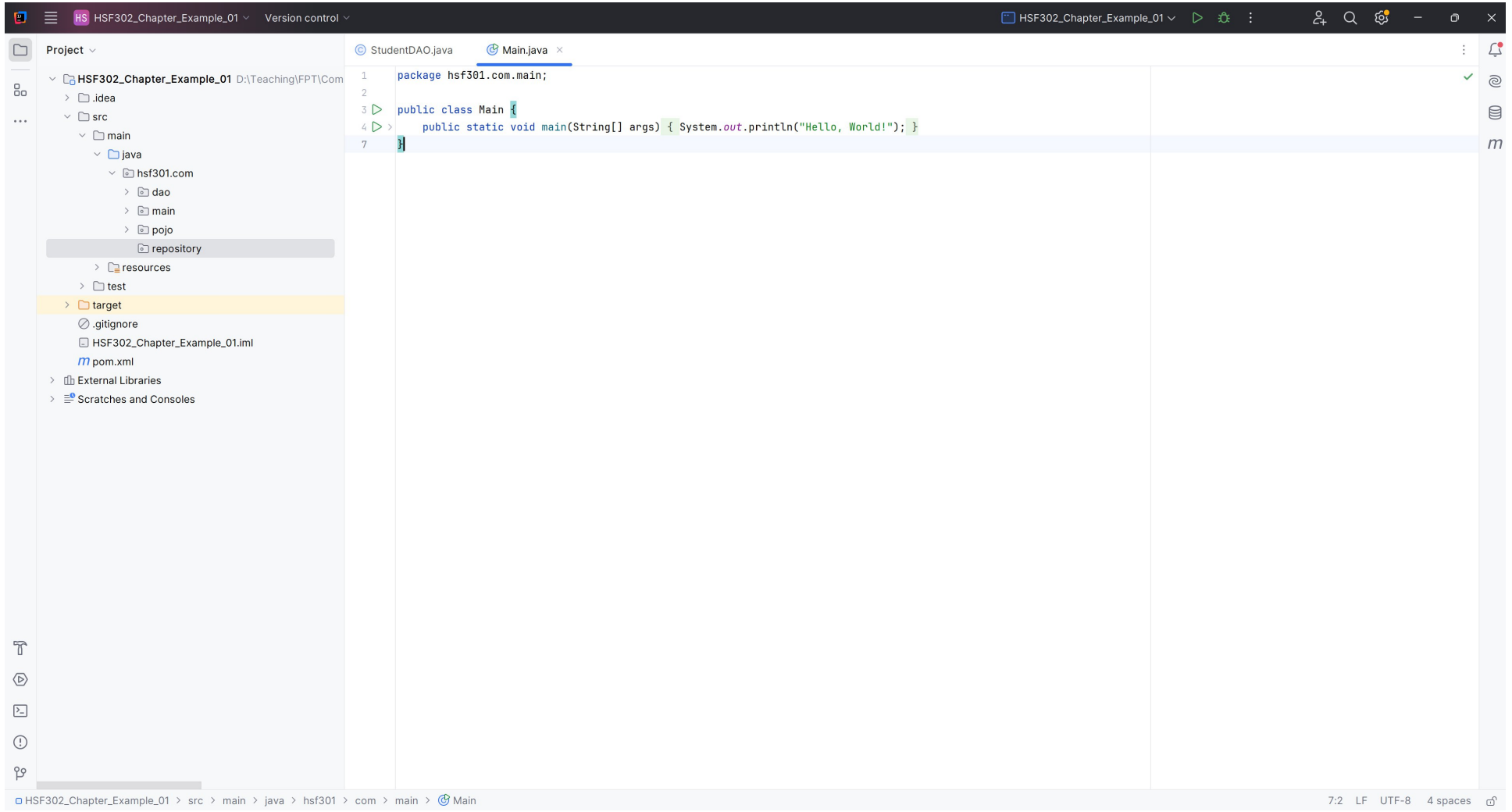
Delete Student Method



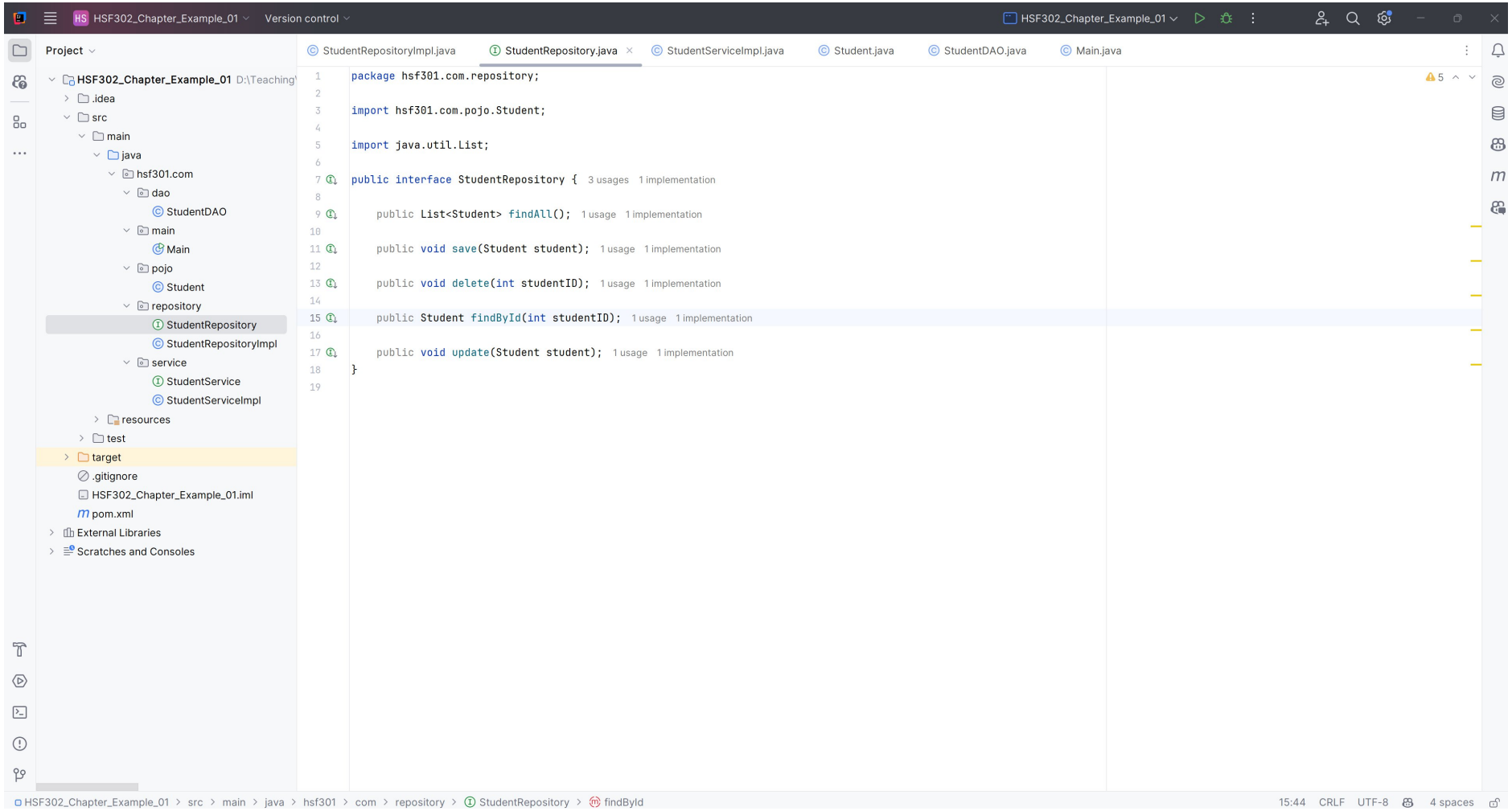
Update Student Method



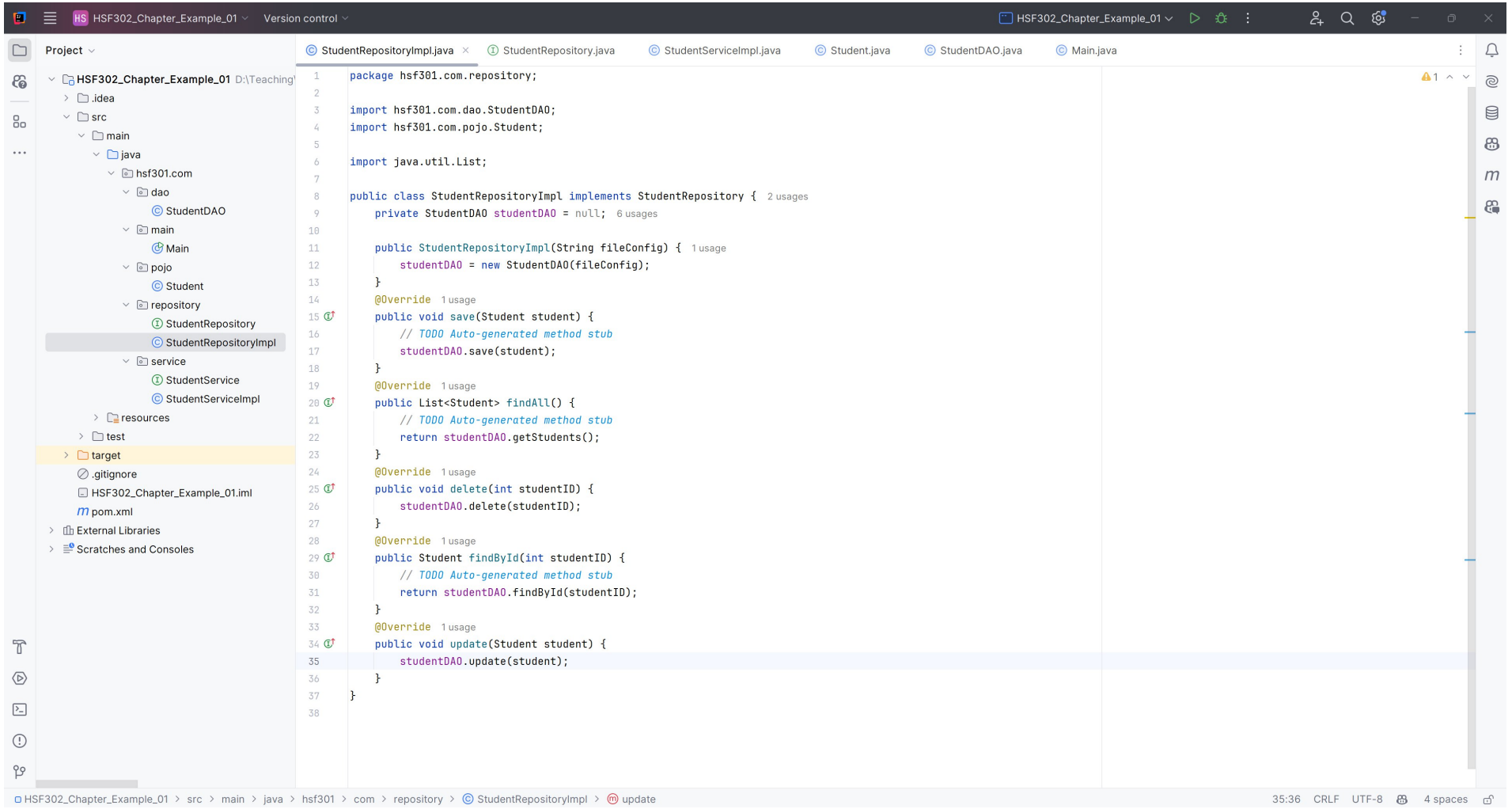
Create hsf301.com.repository Package in src/main/java



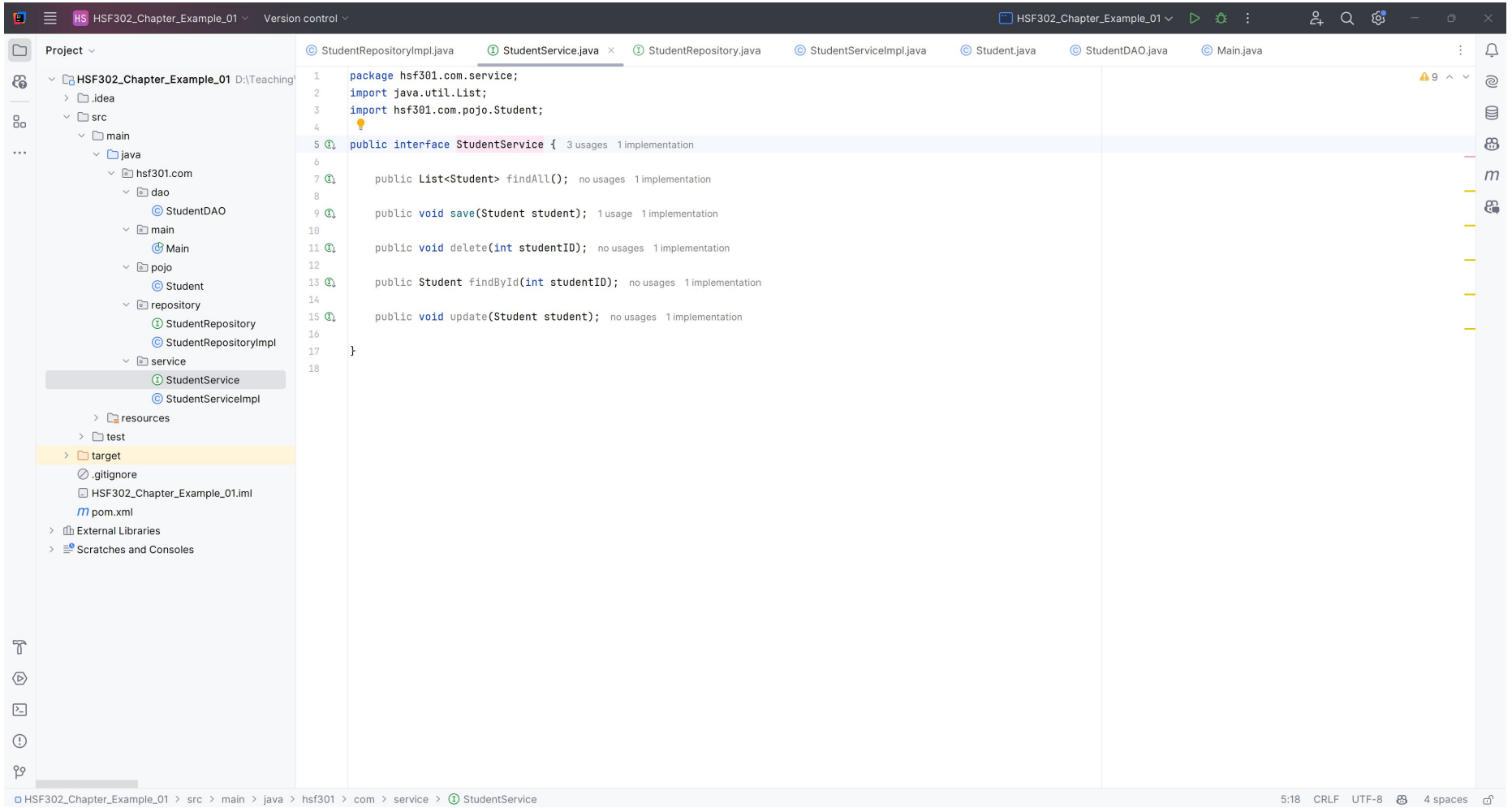
Create StudentRepository



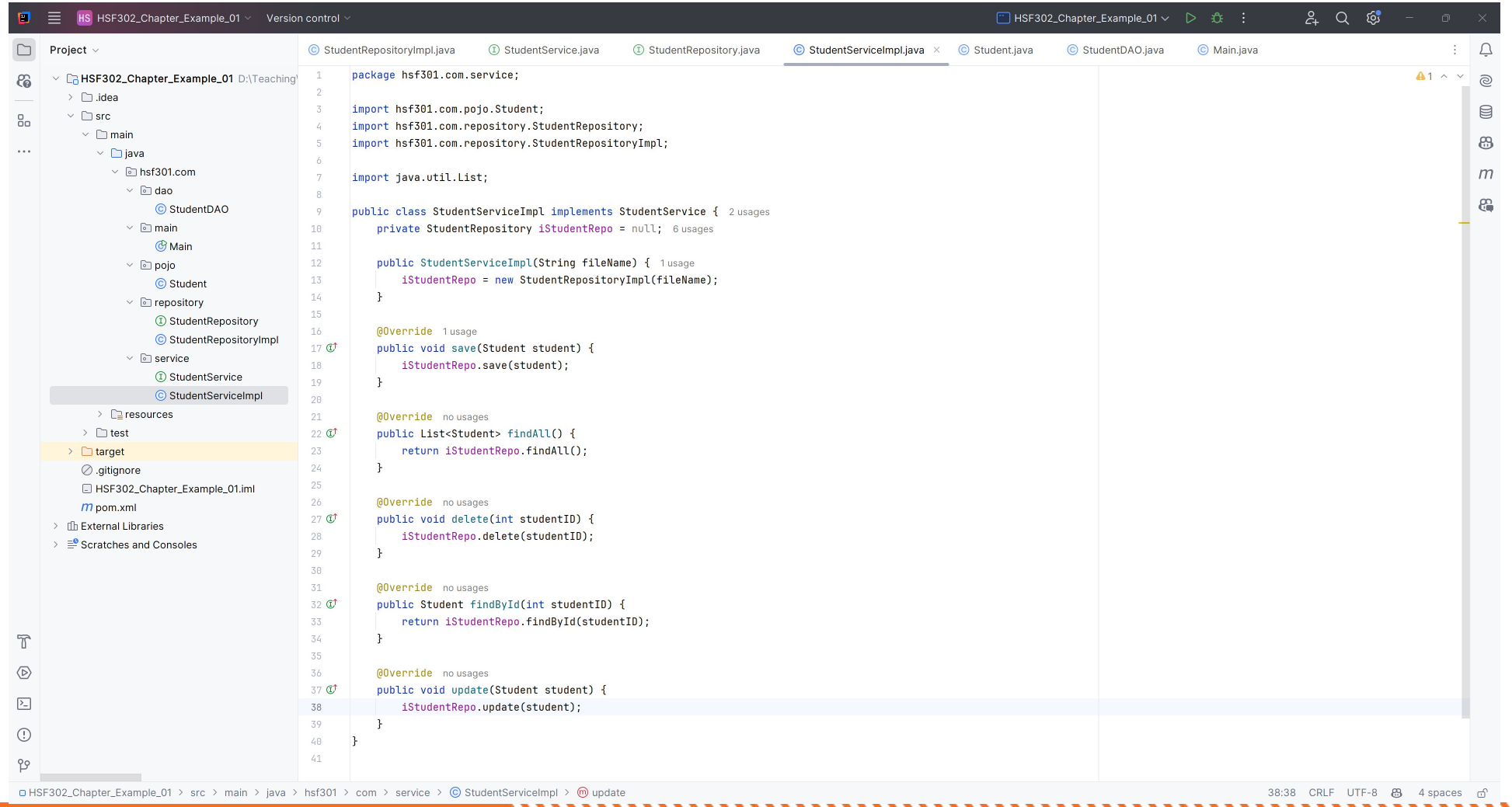
Create StudentRepositoryImpl implement StudentRepository



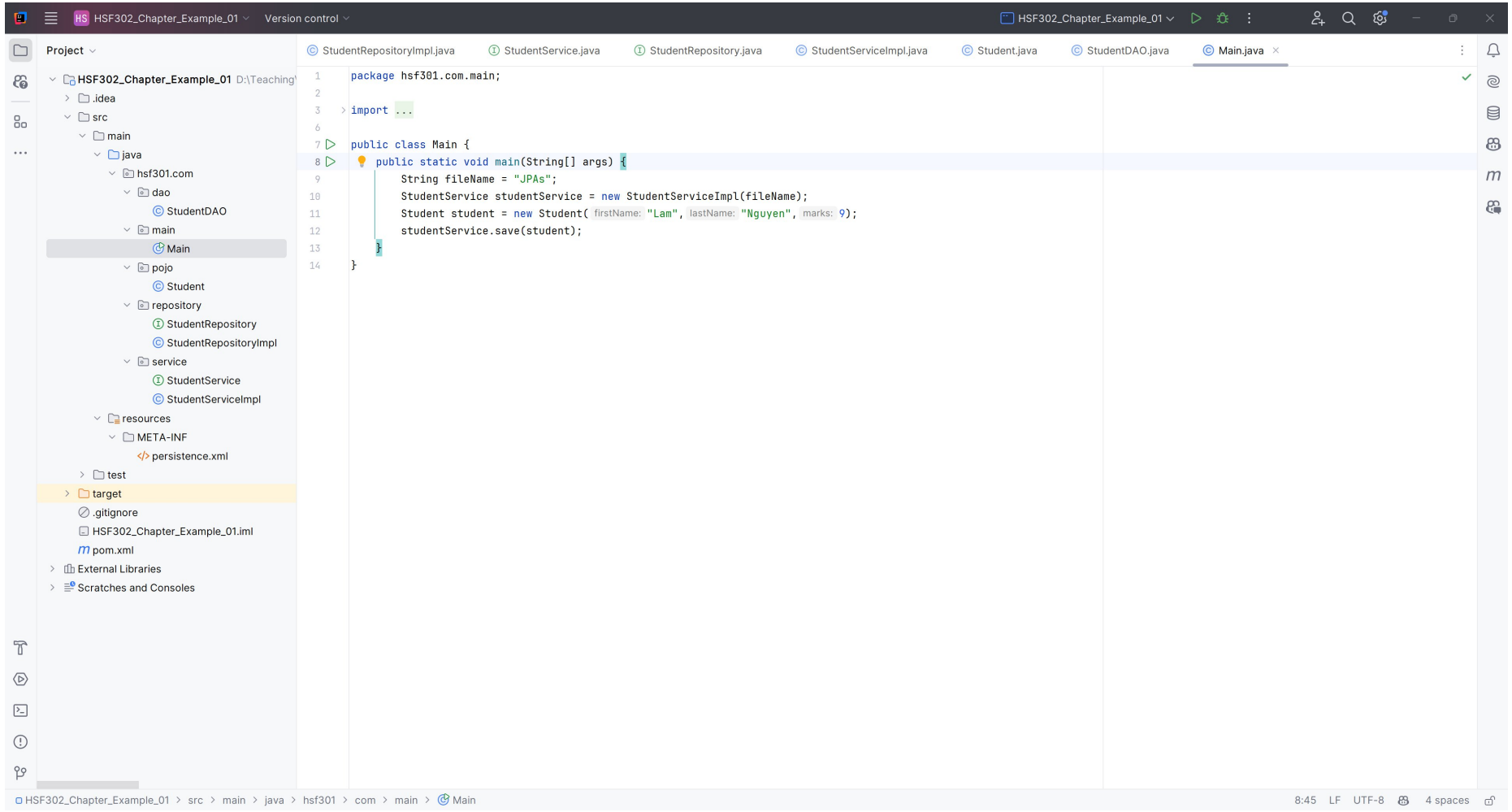
Create StudentService



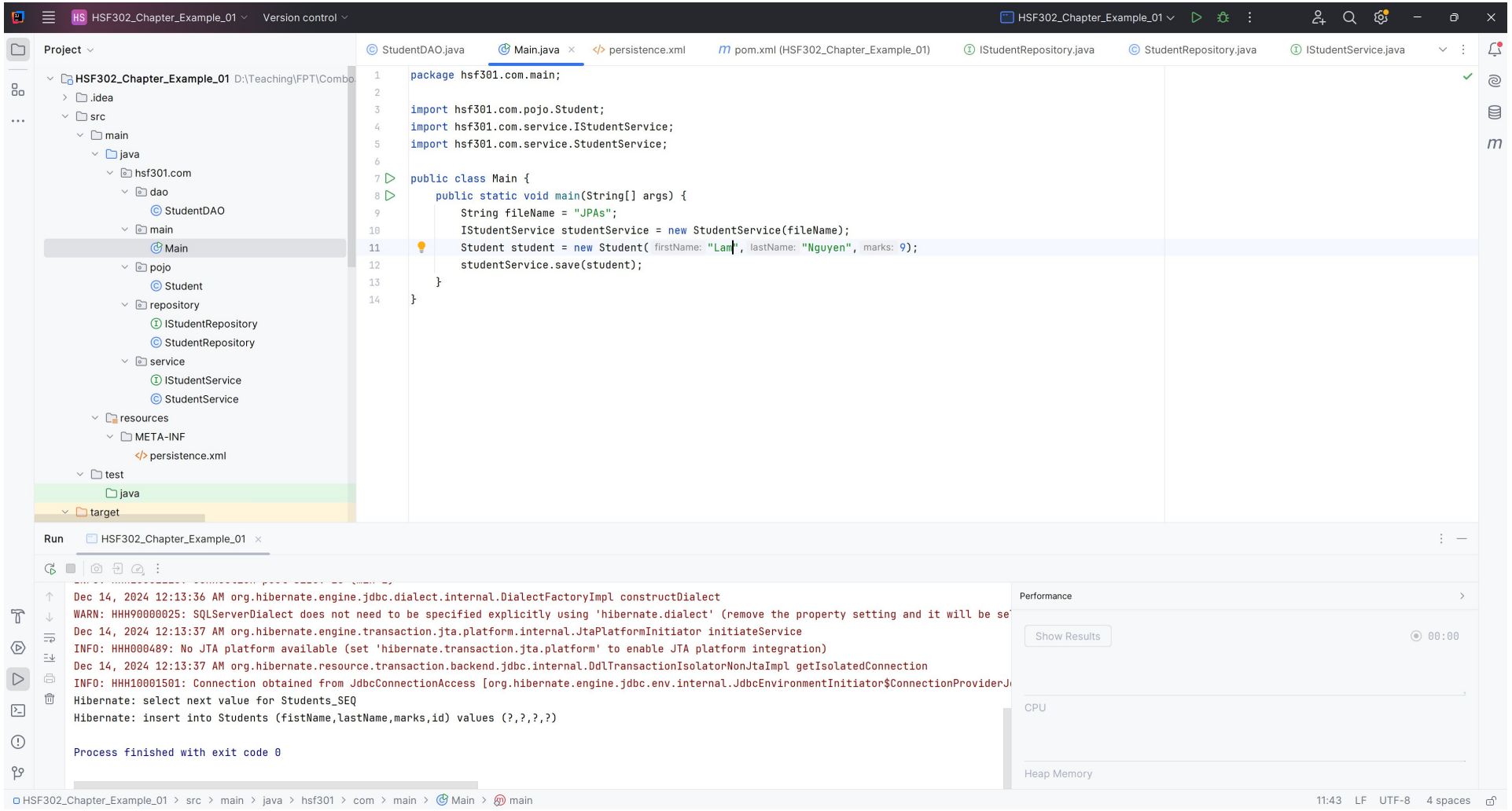
Create StudentServiceImpl implement StudentService



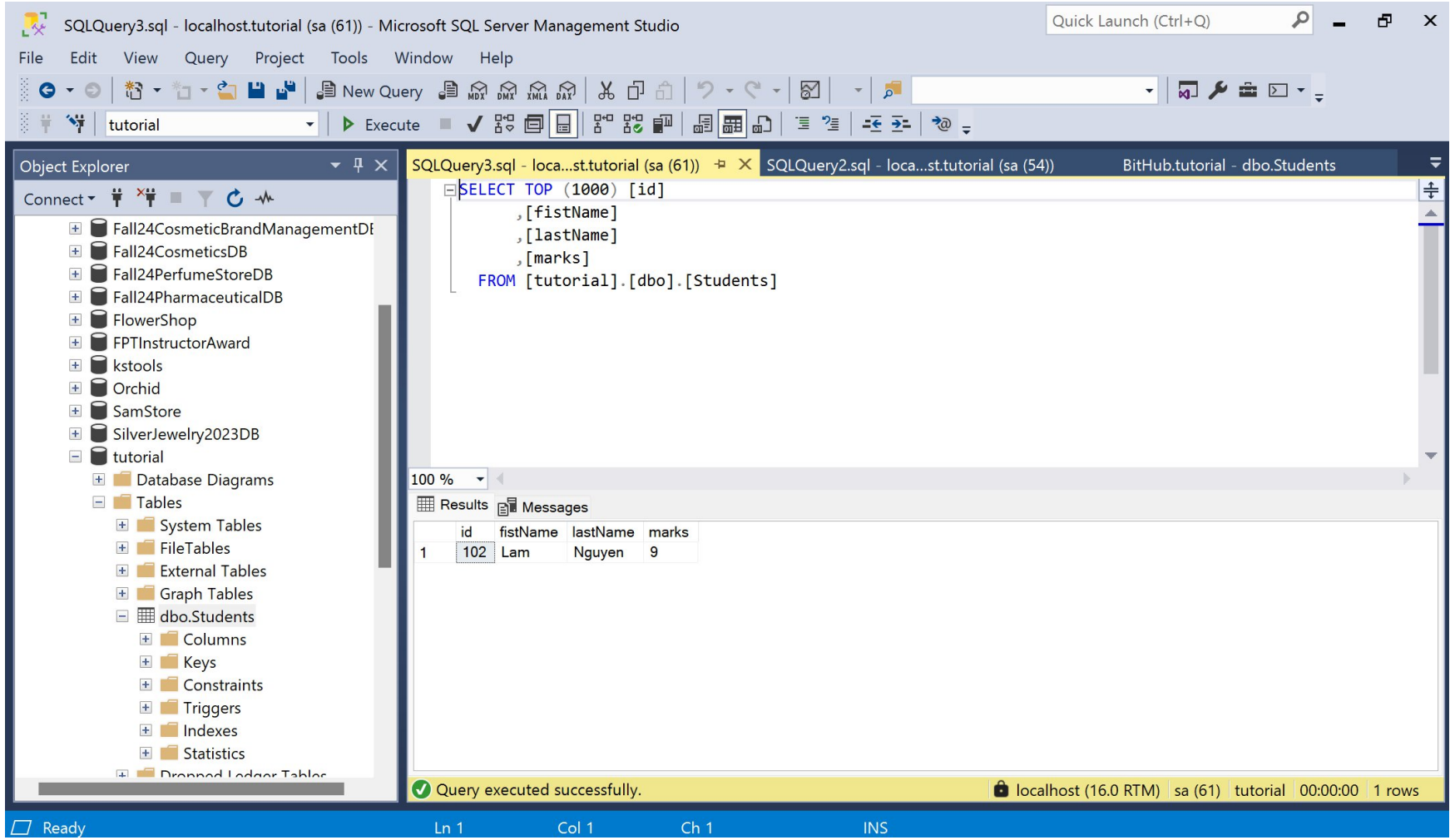
Create Main function



Run Program


HSF302_Chapter_Example_01
Version control
Project
StudentDAO.java
Main.java
persistence.xml
pom.xml (HSF302_Chapter_Example_01)
IStudentRepository.java
StudentRepository.java
IStudentService.java
HSF302_Chapter_Example_01 D:\Teaching\FPT\Combo
.idea
src
main
java
hsf301.com
dao
StudentDAO
main
Main
pojo
Student
repository
IStudentRepository
StudentRepository
service
IStudentService
StudentService
resources
META-INF
persistence.xml
test
java
target
package hsf301.com.main;
import hsf301.com.pojo.Student;
import hsf301.com.service.IStudentService;
import hsf301.com.service.StudentService;
public class Main {
public static void main(String[] args) {
String fileName = "JPAs";
IStudentService studentService = new StudentService(fileName);
Student student = new Student(firstName: "Lam", lastName: "Nguyen", marks: 9);
studentService.save(student);
}
}
Run
HSF302_Chapter_Example_01
Dec 14, 2024 12:13:36 AM org.hibernate.engine.jdbc.dialect.internal.DialectFactoryImpl constructDialect
WARN: HHH90000025: SQLServerDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it will be se
Dec 14, 2024 12:13:37 AM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
Dec 14, 2024 12:13:37 AM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJ
Hibernate: select next value for Students_SEQ
Hibernate: insert into Students (fistName,lastName,marks,id) values (?,?,?,?)
Process finished with exit code 0
Performance
Show Results
00:00
CPU
Heap Memory
HSF302_Chapter_Example_01 › src › main › java › hsf301 › com › main › Main › main
11:43
LF
UTF-8
4 spaces

Result


SQLQuery3.sql - localhost.tutorial (sa (61)) - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
Execute
tutorial
Object Explorer
Connect
Fall24CosmeticBrandManagementDB
Fall24CosmeticsDB
Fall24PerfumeStoreDB
Fall24PharmaceuticalDB
FlowerShop
FPTInstructorAward
kstools
Orchid
SamStore
SilverJewelry2023DB
tutorial
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Students
Columns
Keys
Constraints
Triggers
Indexes
Statistics
SQLQuery3.sql - loca...st.tutorial (sa (61))
SQLQuery2.sql - loca...st.tutorial (sa (54))
BitHub.tutorial - dbo.Students
SELECT TOP (1000) [id]
,[fistName]
,[lastName]
,[marks]
FROM [tutorial].[dbo].[Students]
100 %
Results
Messages
id fistName lastName marks
1 102 Lam Nguyen 9
Query executed successfully.
localhost (16.0 RTM) sa (61) tutorial 00:00:00 1 rows
Ready
Ln 1
Col 1
Ch 1
INS

FPT Education
FPT UNIVERSITY

Java

Mapping in JPA

Relationships Annotations in JPA

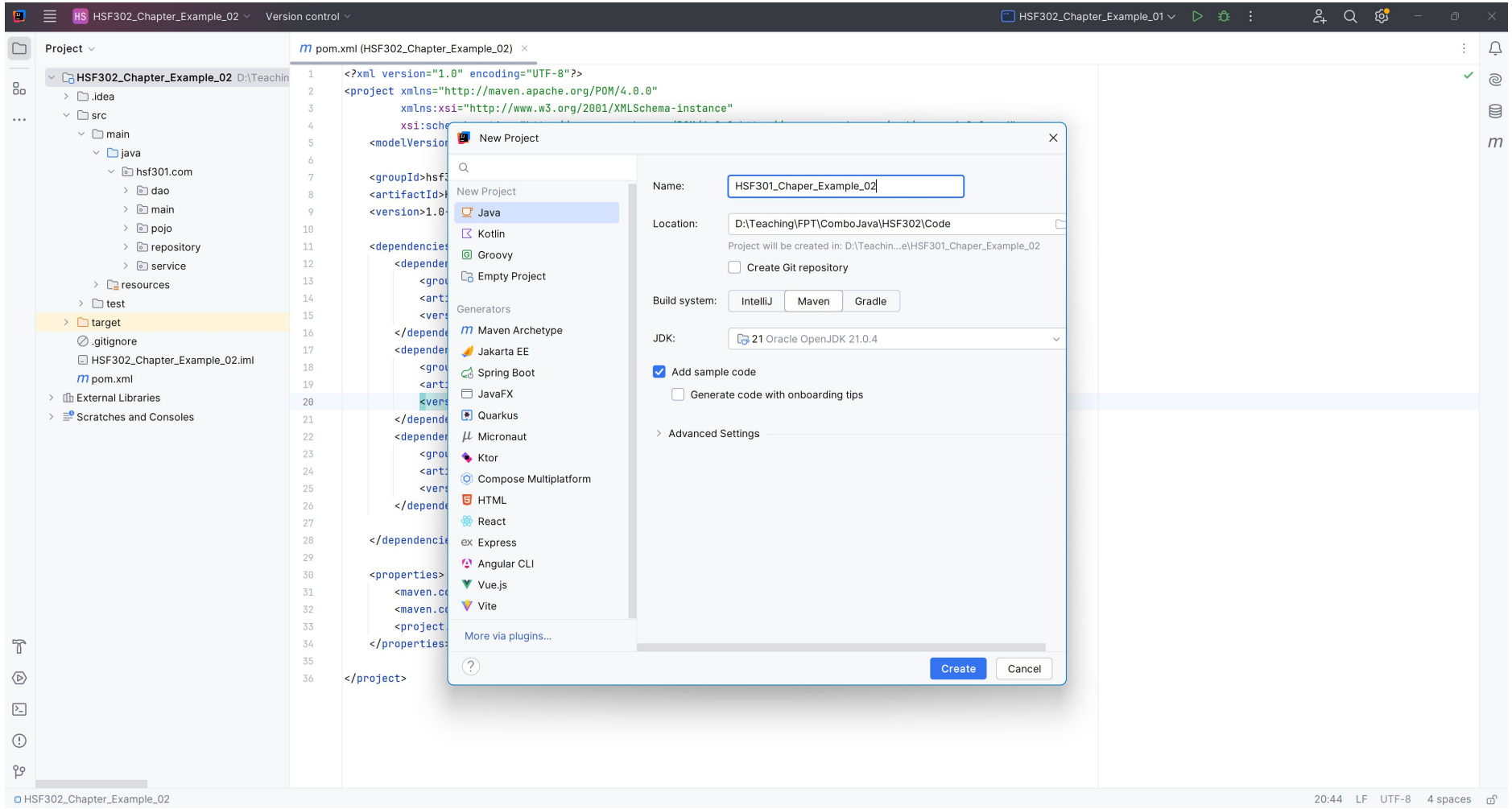
- **@ManyToOne:** This annotation defines a many-to-one relationship between two entities.
- **@OneToMany:** This annotation defines a one-to-many relationship between two entities.
- **@OneToOne:** This annotation defines a one-to-one relationship between two entities.
- **@ManyToMany:** This annotation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

FPT Education
FPT UNIVERSITY

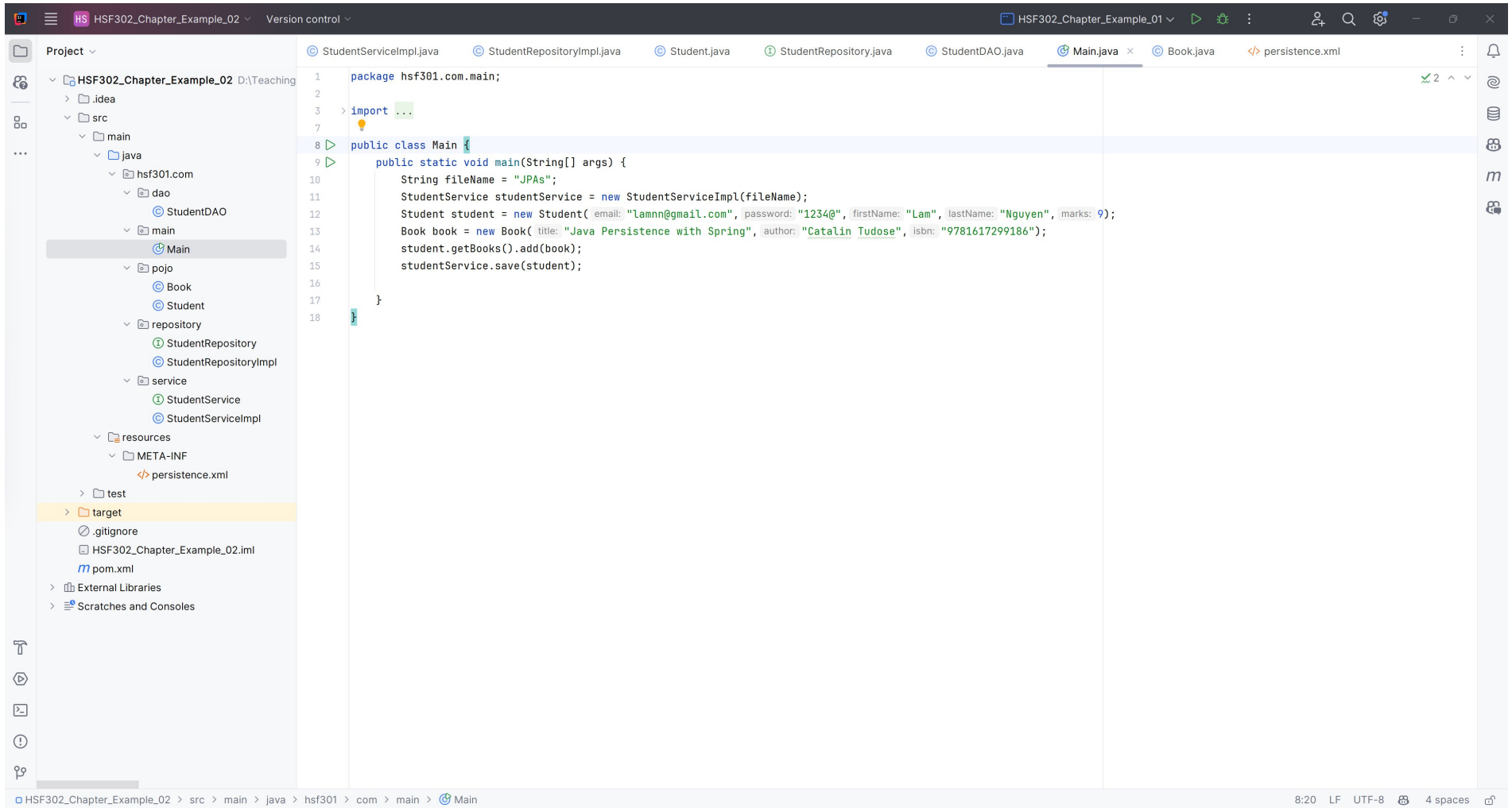
Java

Demo JPA
(One To Many)

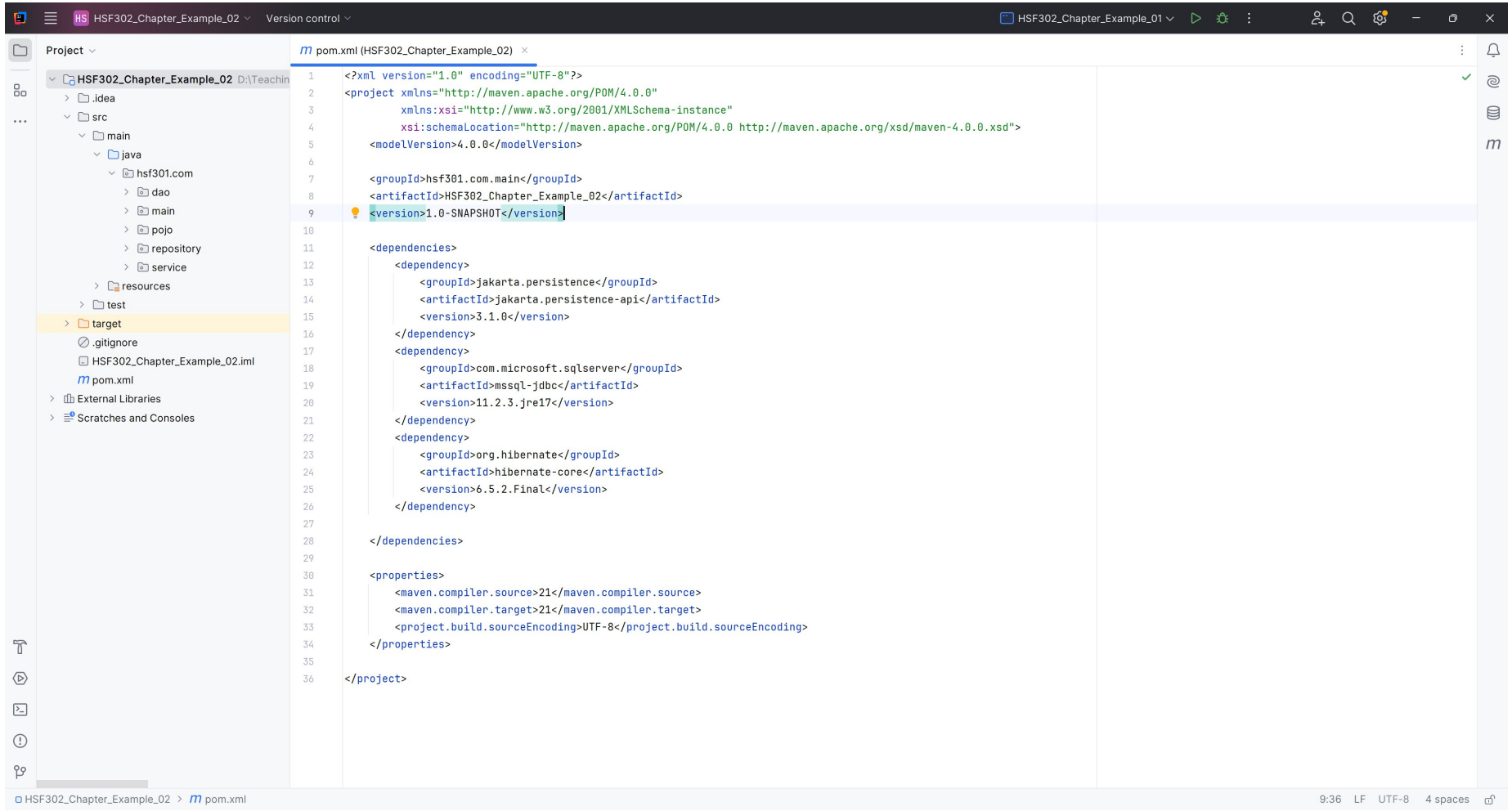
Open IntelliJ, File | New | Maven Project



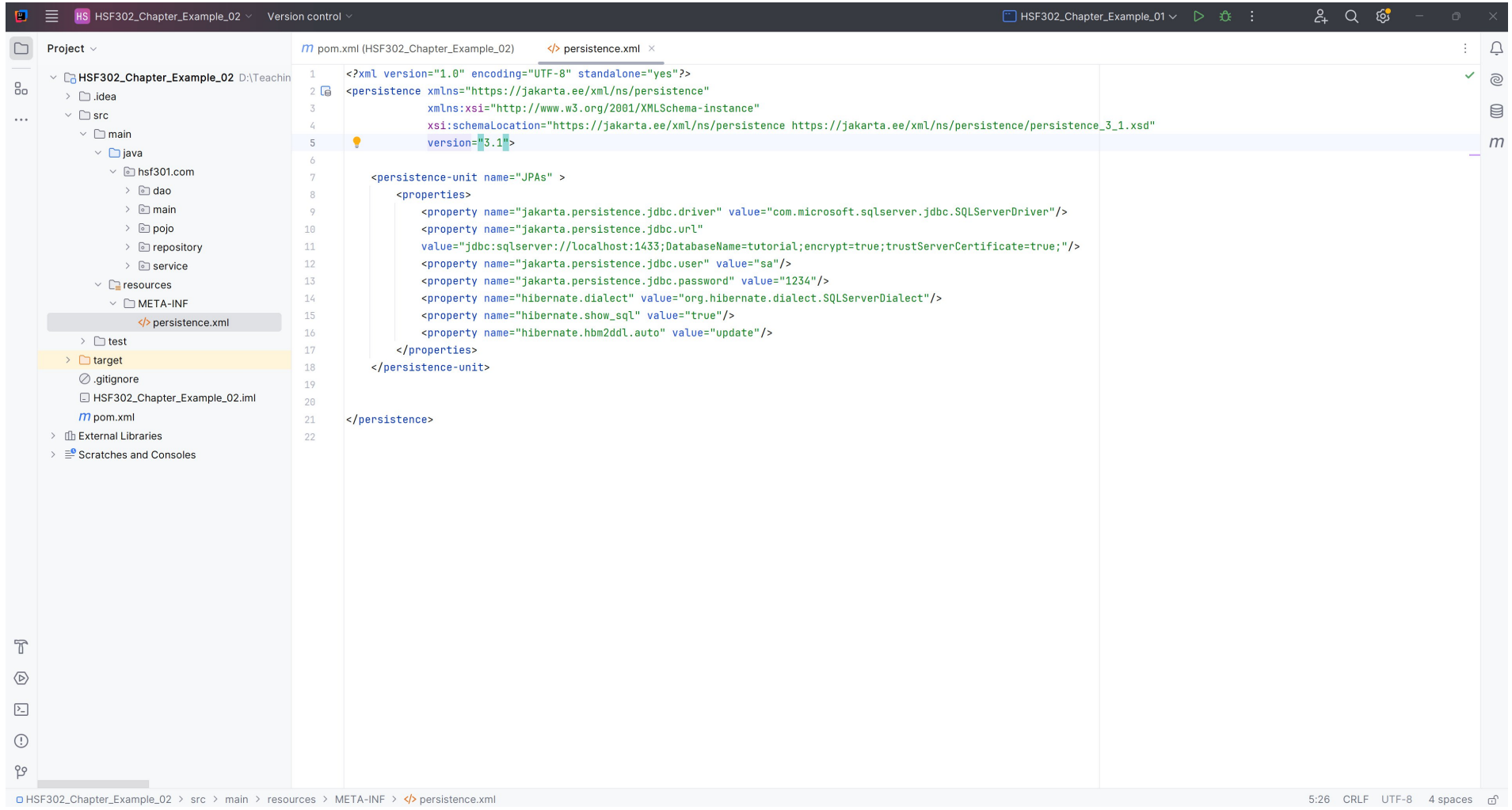
Create a structure of Maven Project



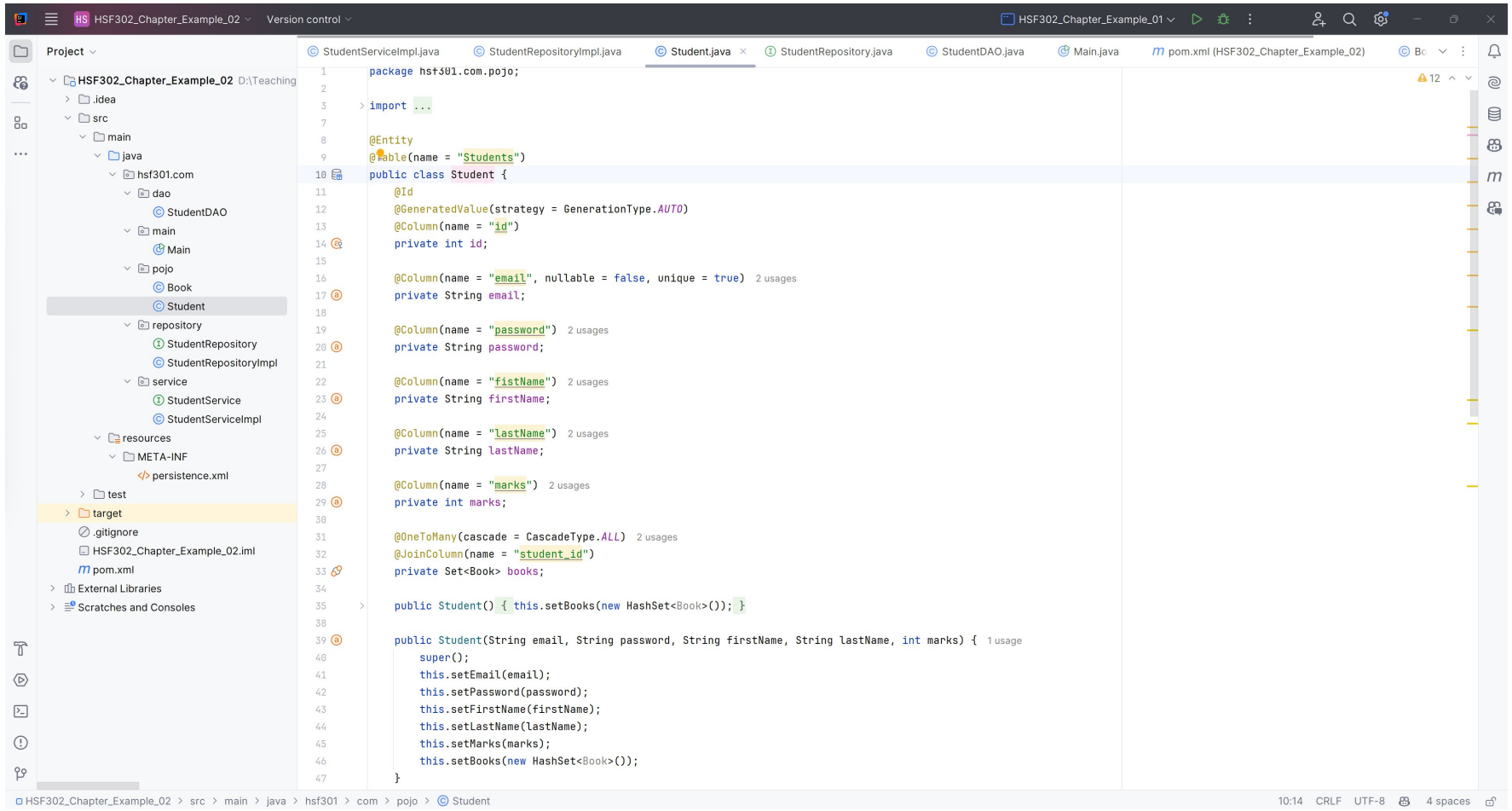
Edit pom.xml


HSF302_Chapter_Example_02
Version control
HSF302_Chapter_Example_01
Project
pom.xml (HSF302_Chapter_Example_02)
HSF302_Chapter_Example_02 D:\Teachin
.idea
src
main
java
hsf301.com
dao
main
pojo
repository
service
resources
test
target
.gitignore
HSF302_Chapter_Example_02.iml
pom.xml
External Libraries
Scratches and Consoles
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>hsf301.com.main</groupId>
<artifactId>HSF302_Chapter_Example_02</artifactId>
<version>1.0-SNAPSHOT</version>
<dependencies>
<dependency>
<groupId>jakarta.persistence</groupId>
<artifactId>jakarta.persistence-api</artifactId>
<version>3.1.0</version>
</dependency>
<dependency>
<groupId>com.microsoft.sqlserver</groupId>
<artifactId>mssql-jdbc</artifactId>
<version>11.2.3.jre17</version>
</dependency>
<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-core</artifactId>
<version>6.5.2.Final</version>
</dependency>
</dependencies>
<properties>
<maven.compiler.source>21</maven.compiler.source>
<maven.compiler.target>21</maven.compiler.target>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
</project>
HSF302_Chapter_Example_02 > pom.xml
9:36 LF UTF-8 4 spaces

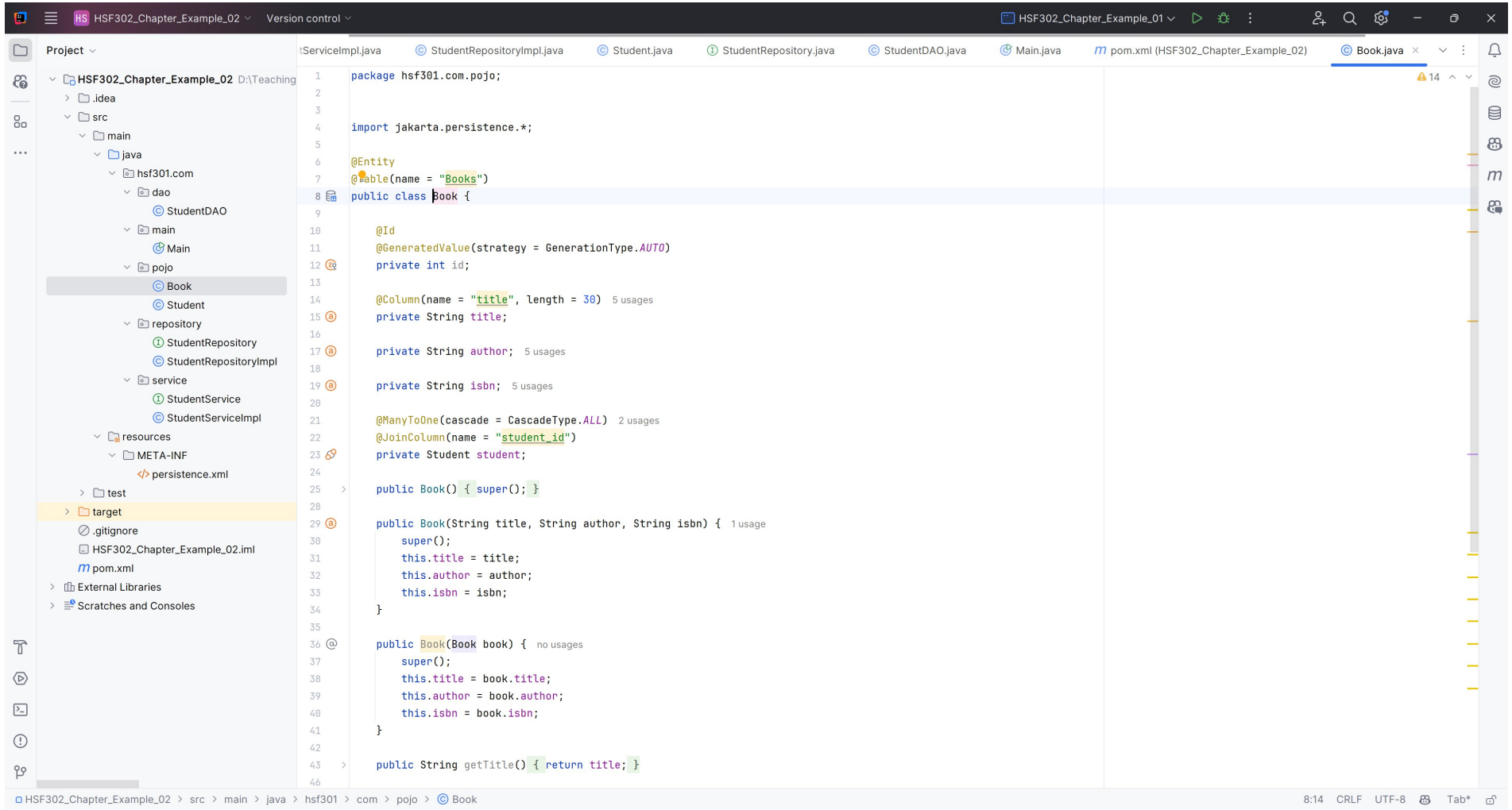
Create persistence.xml in META-INF folder



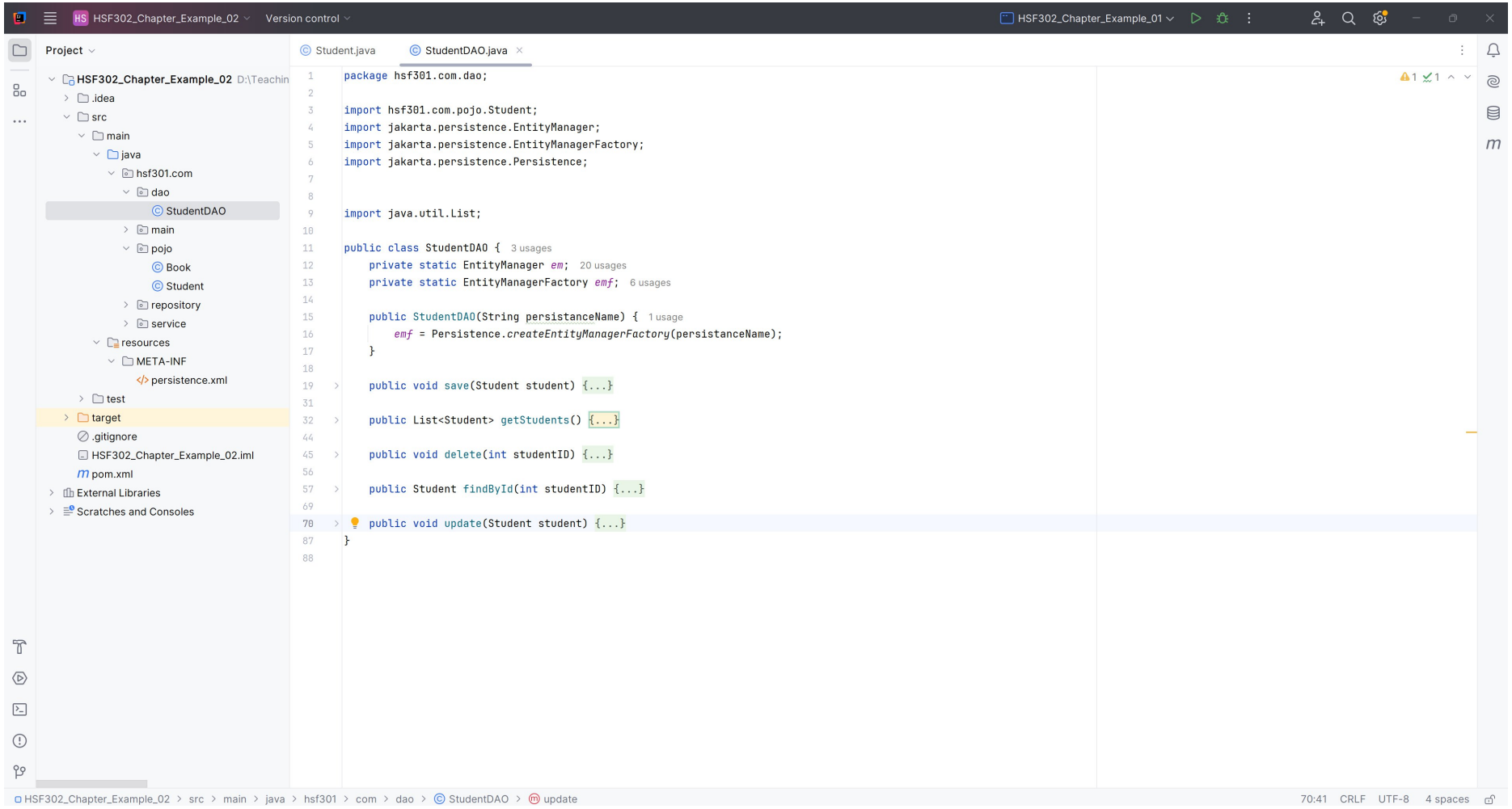
Create Student.java in Pojo



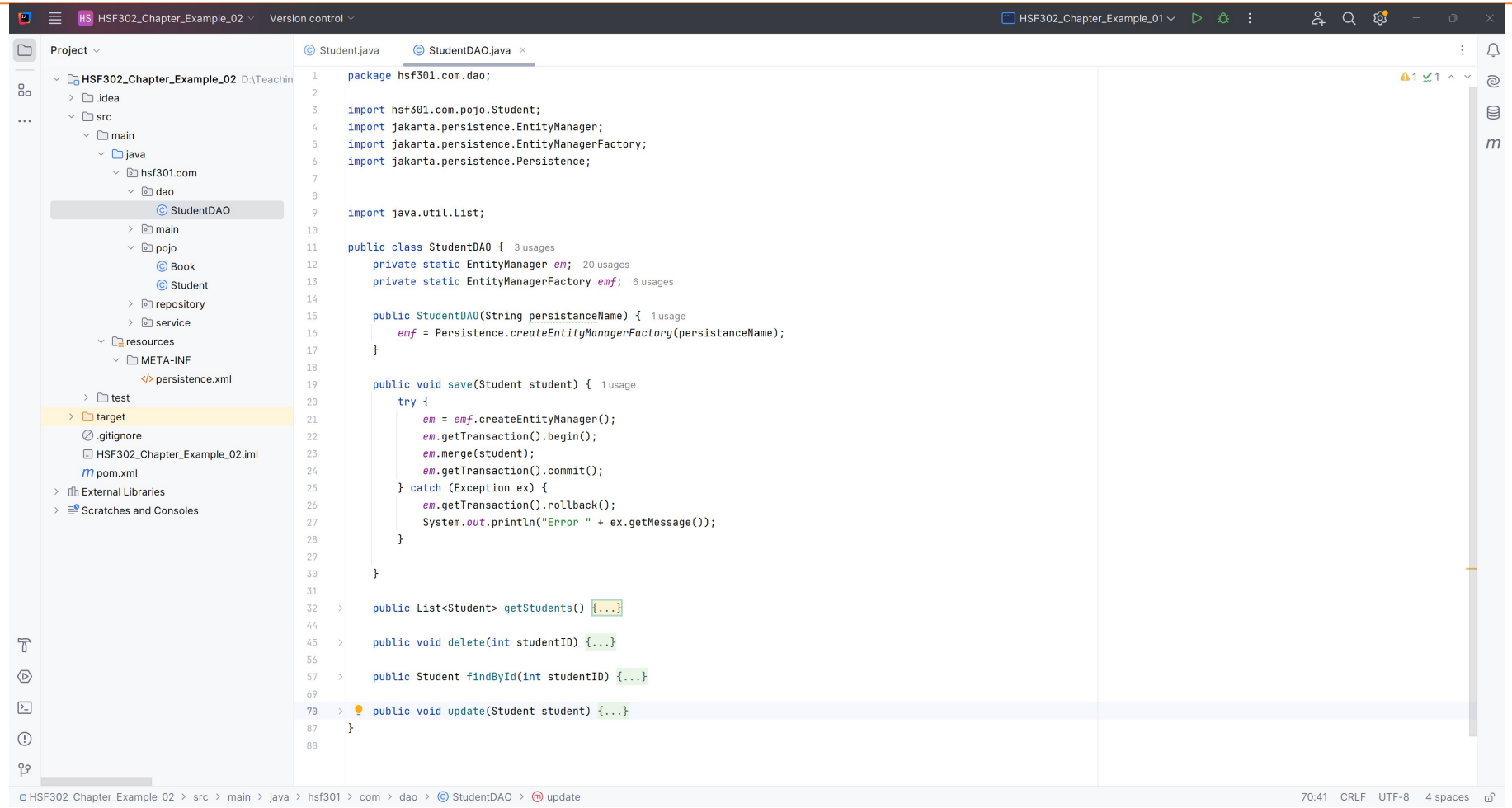
Create Book.java in hsf301.com.pojo



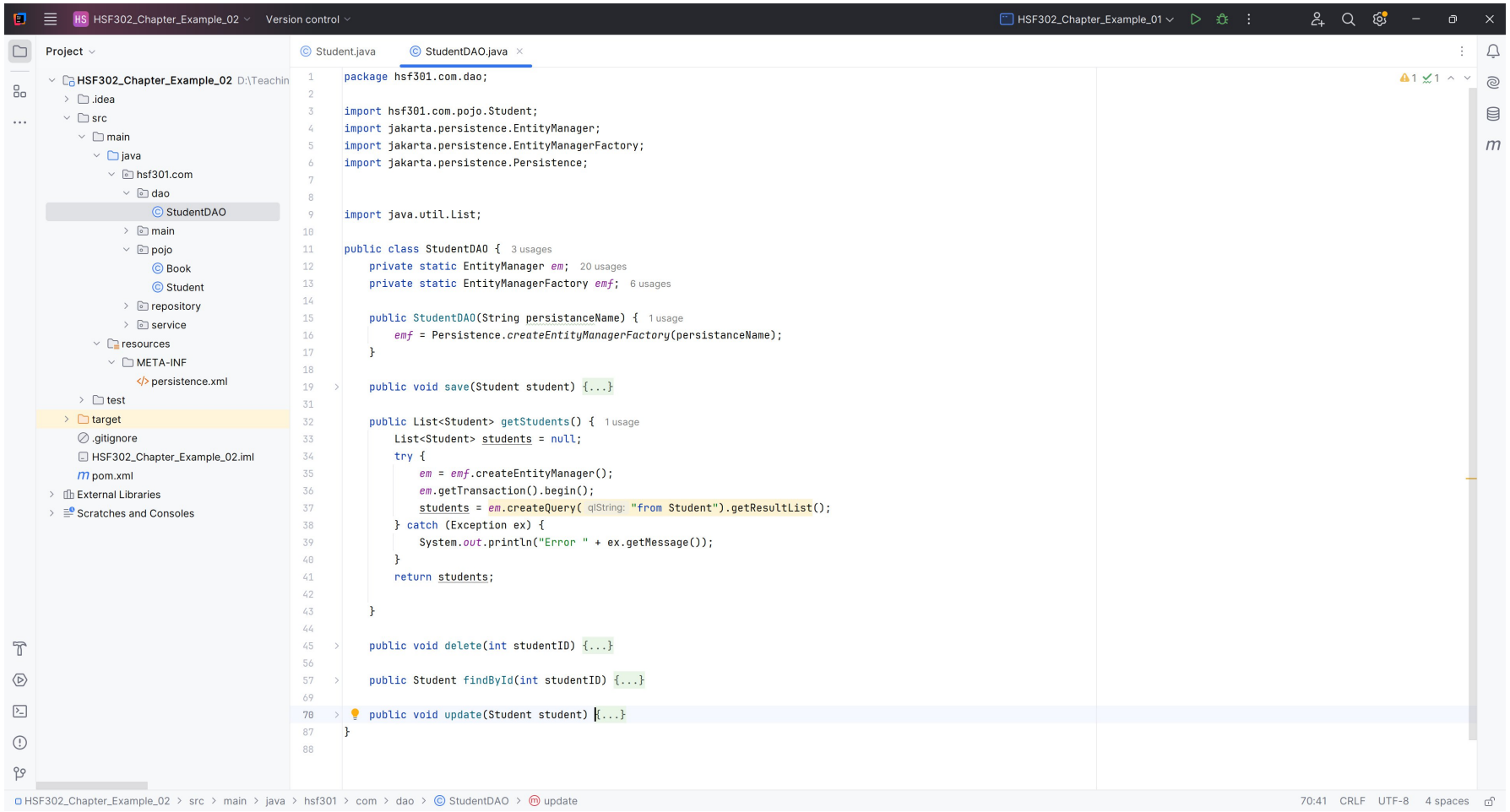
Create StudentDAO



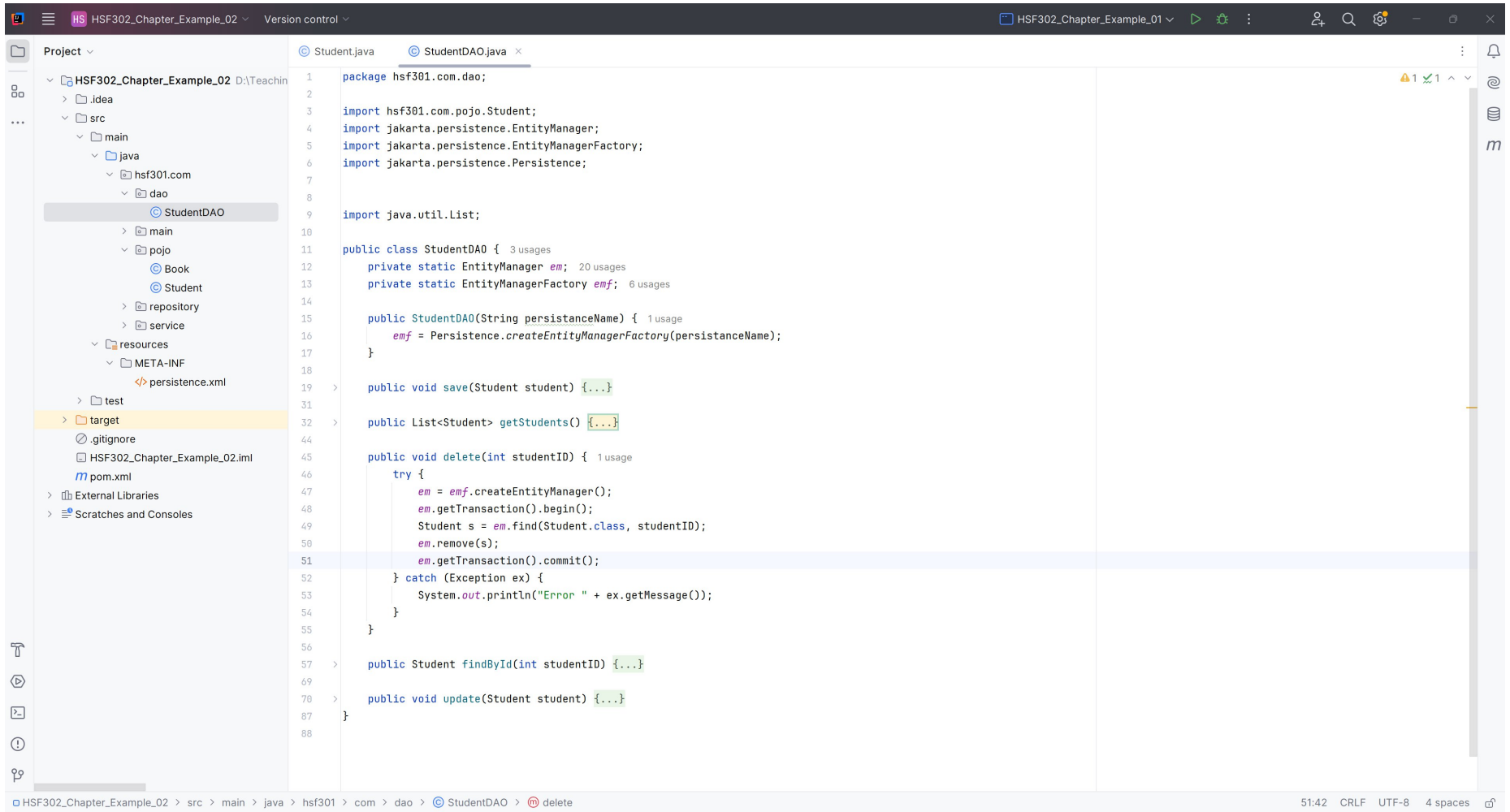
Save Student in StudentDAO



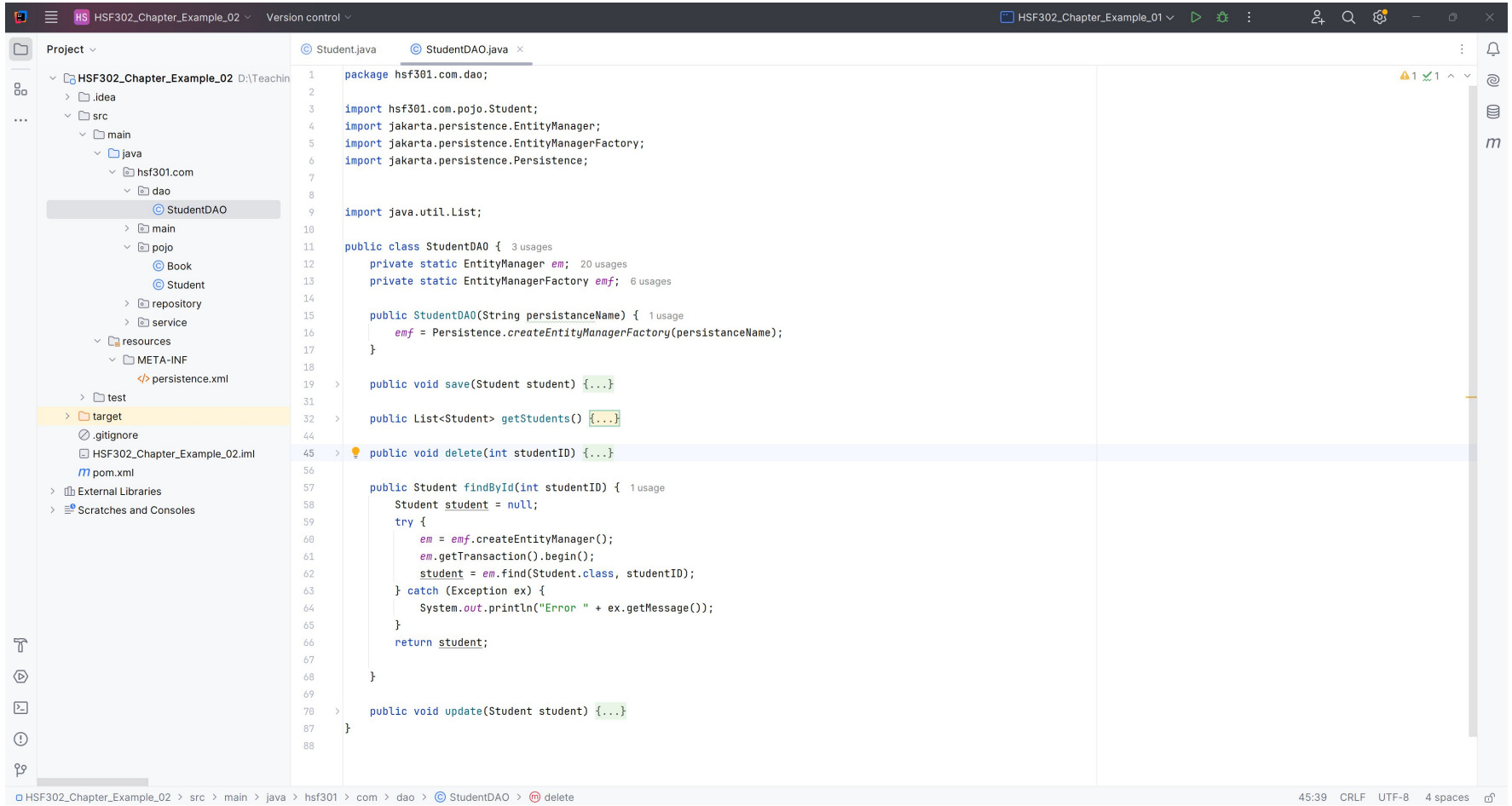
Get All Students in StudentDAO



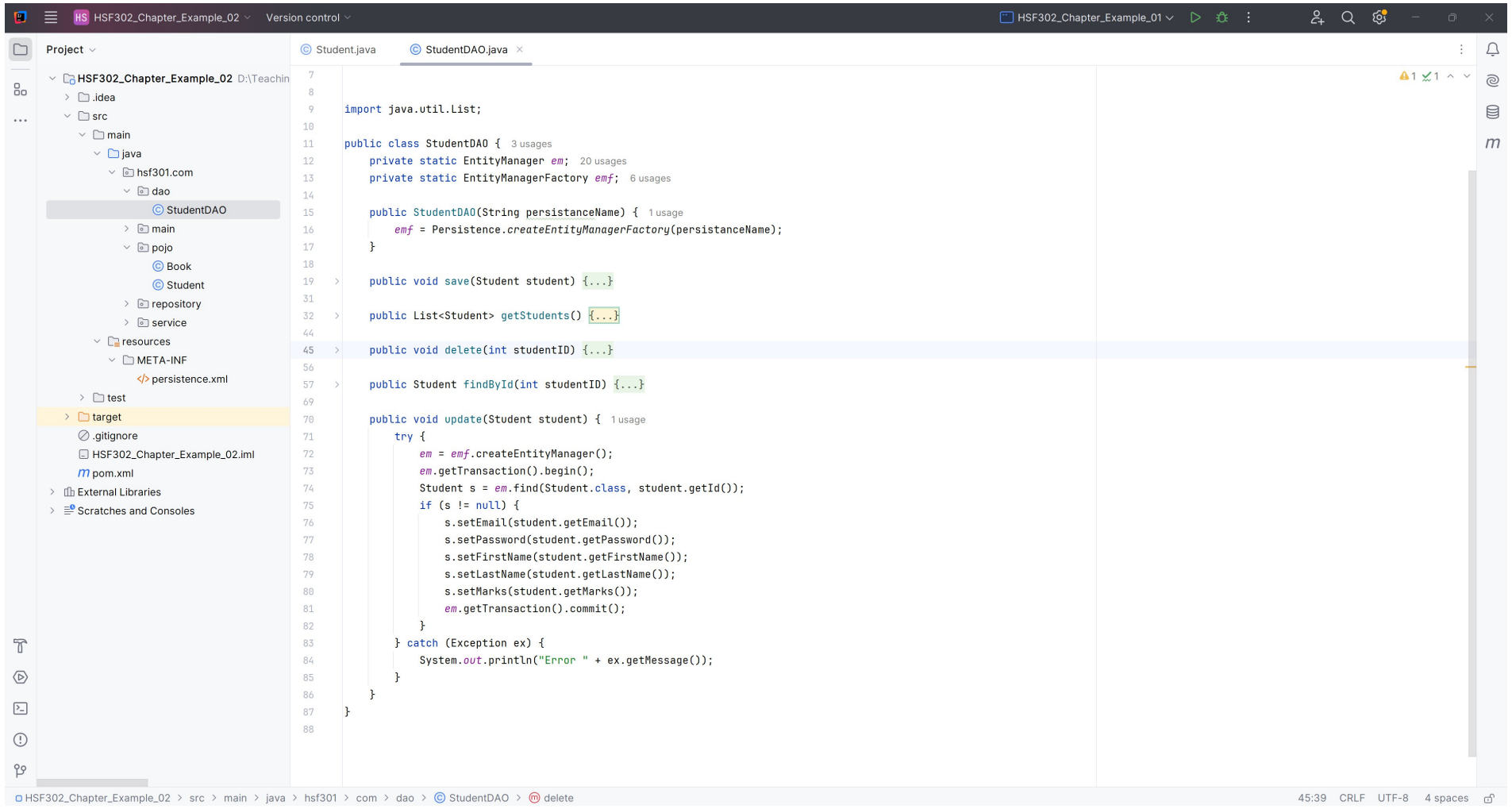
Delete Student in StudentDAO



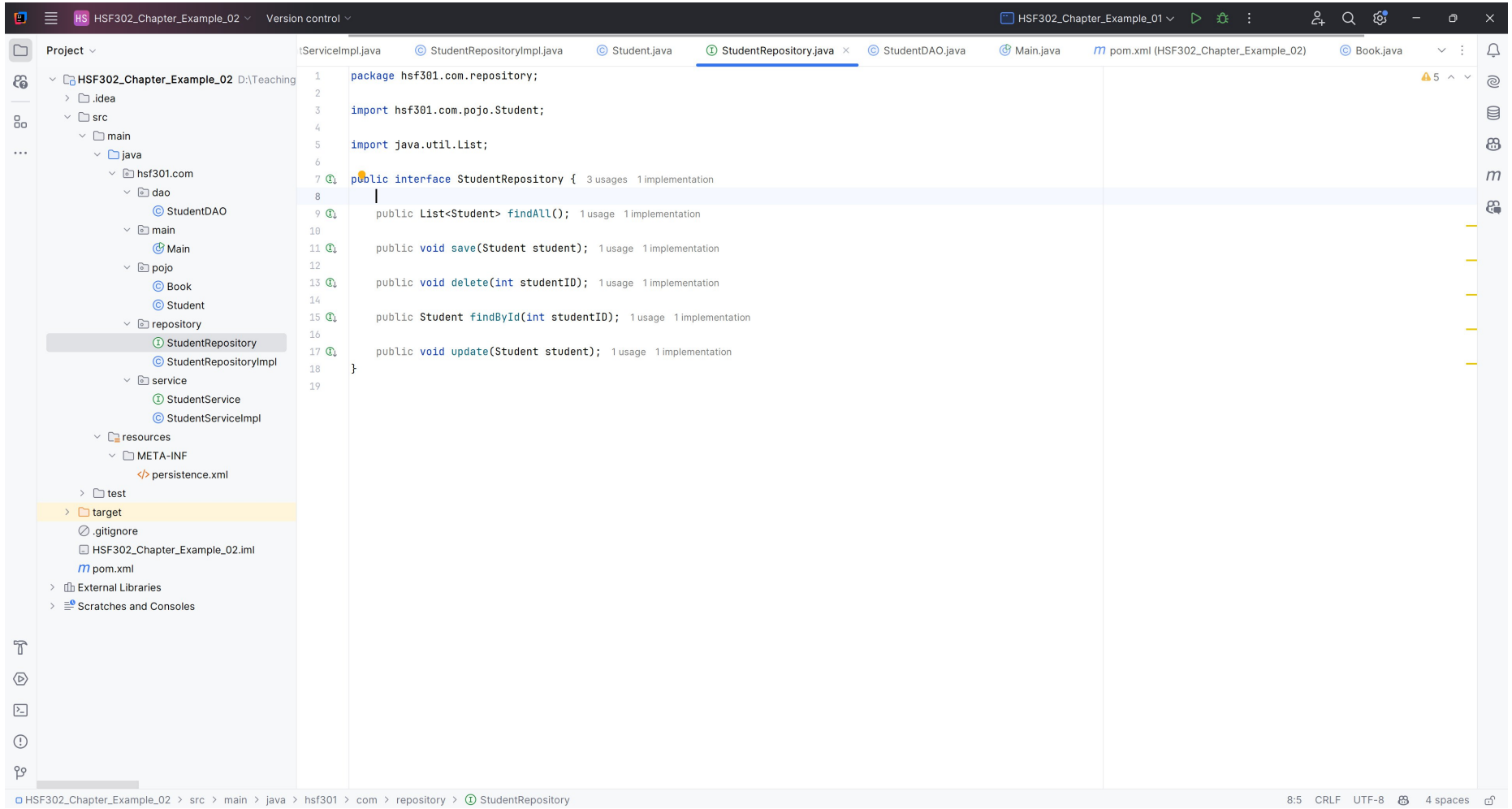
Get a Student in StudentDAO



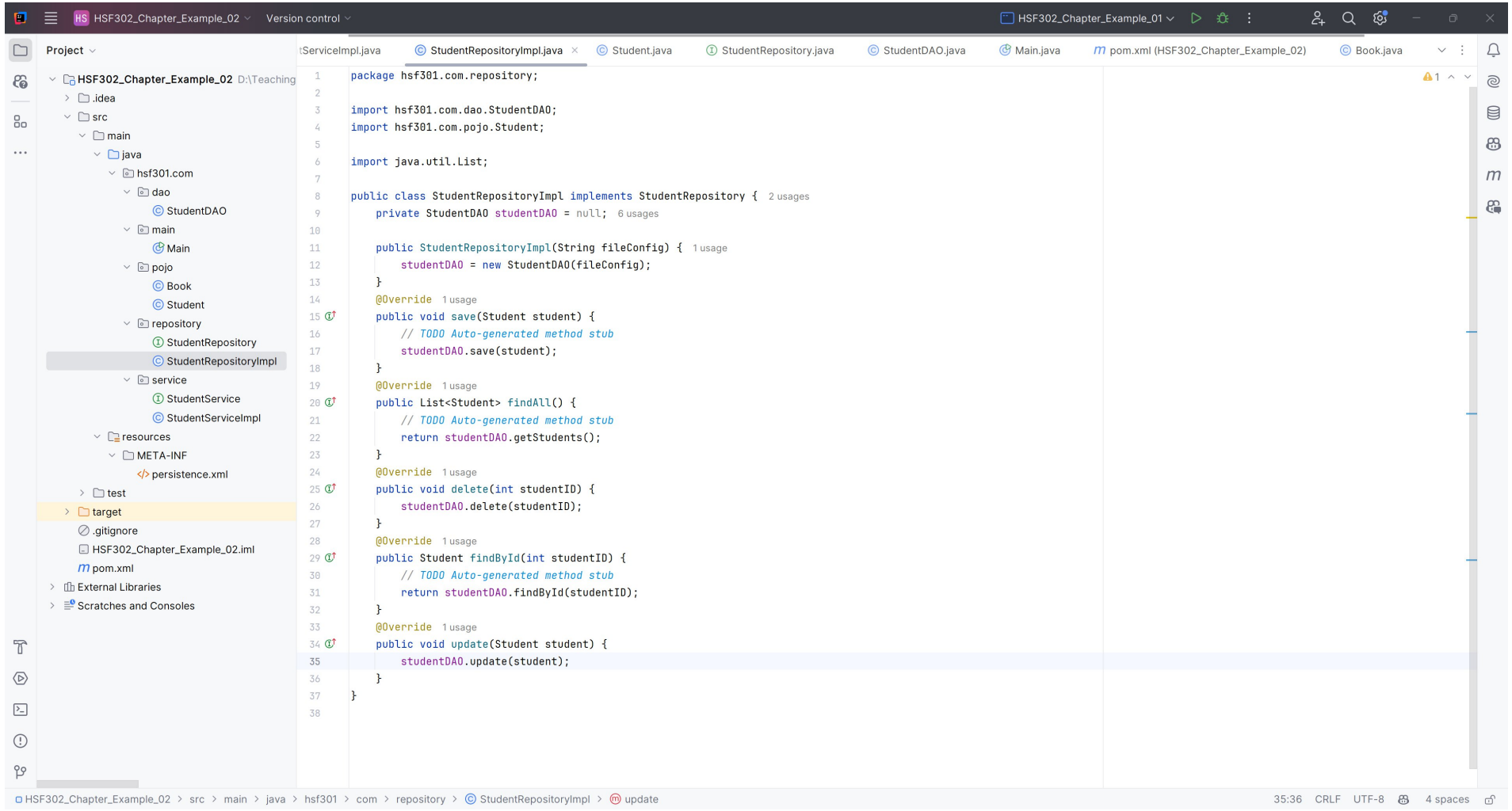
UpdateStudent in StudentDAO



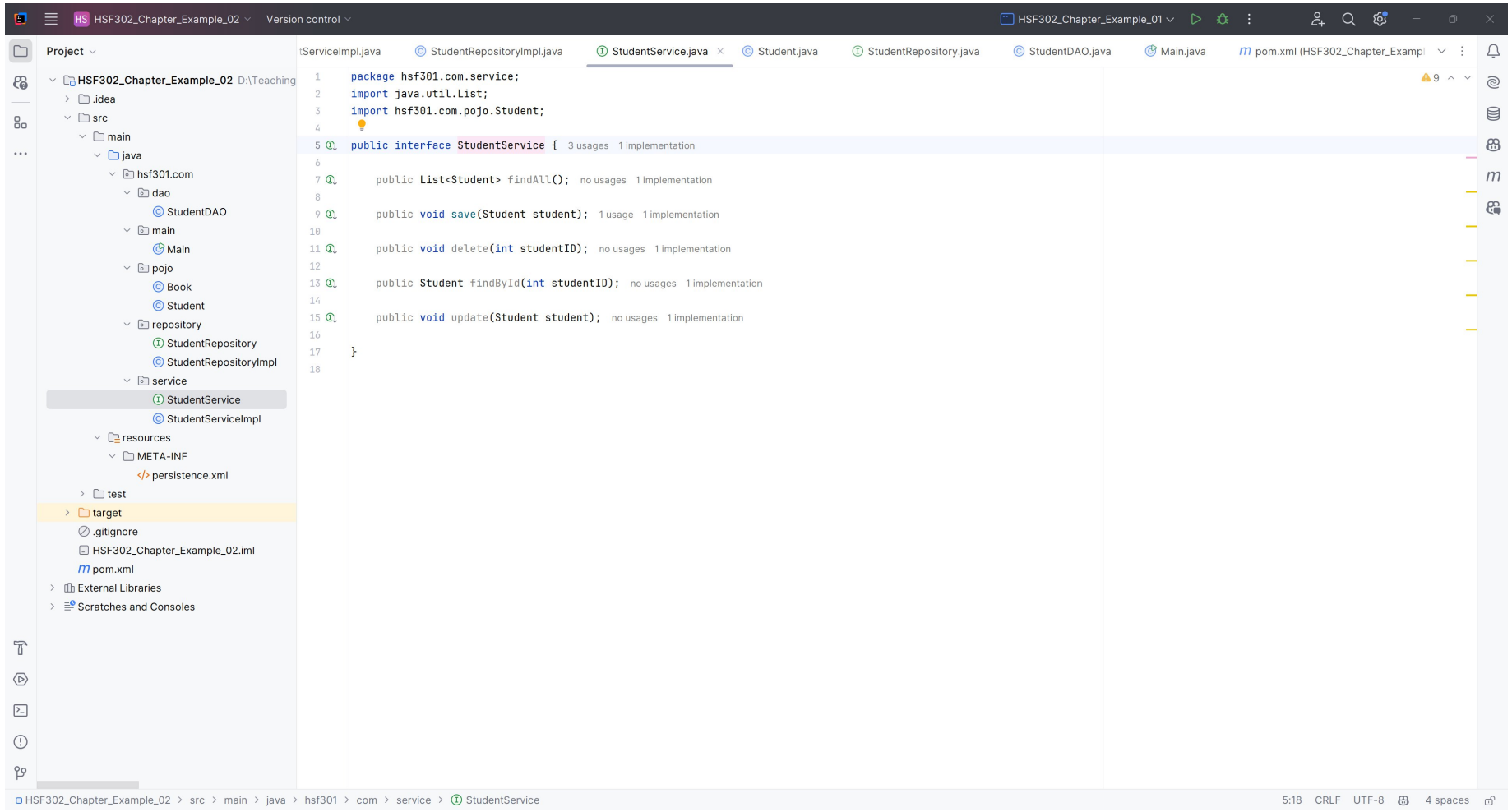
Create StudentRepository



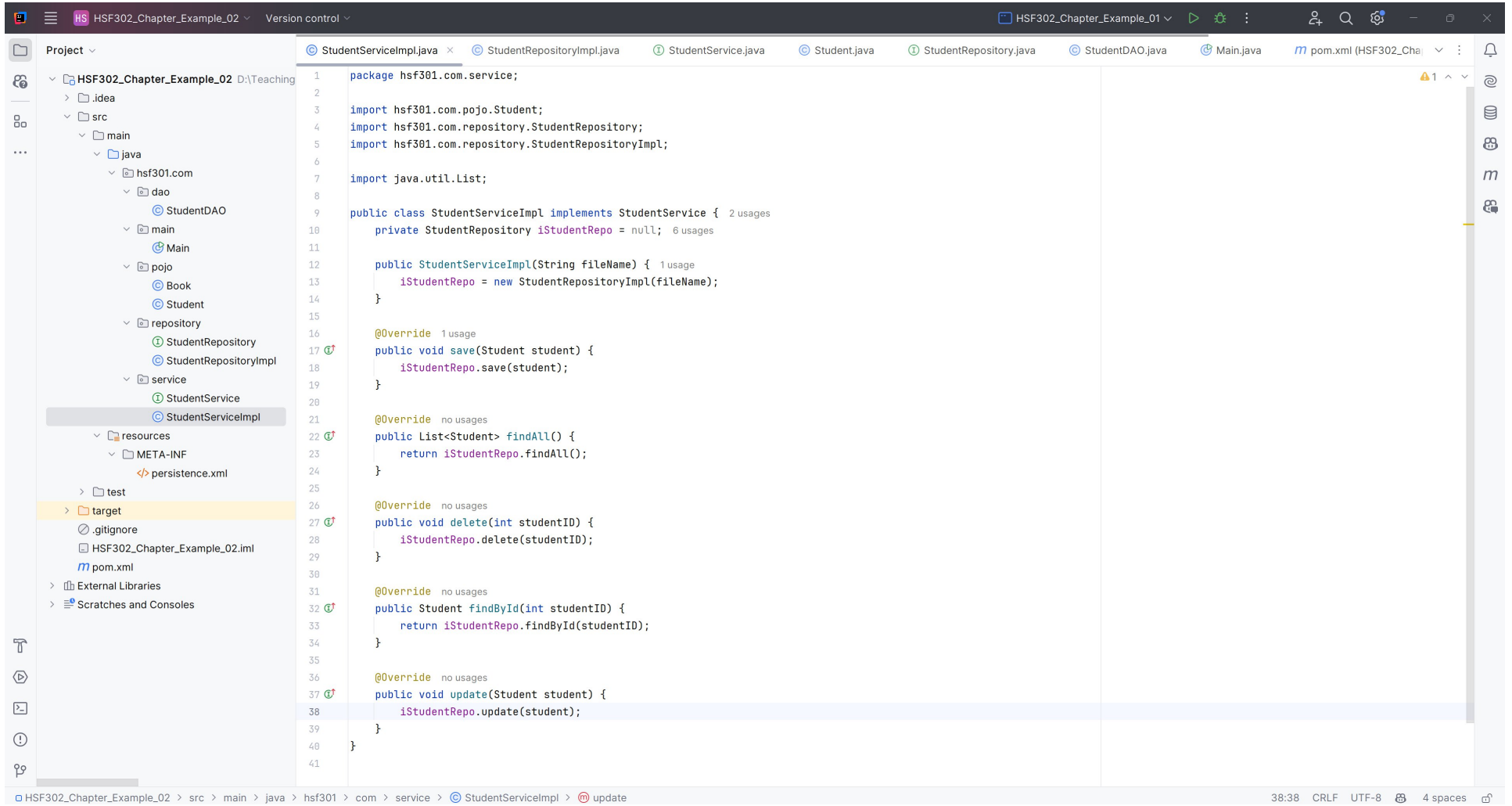
Create StudentRepositoryImpl



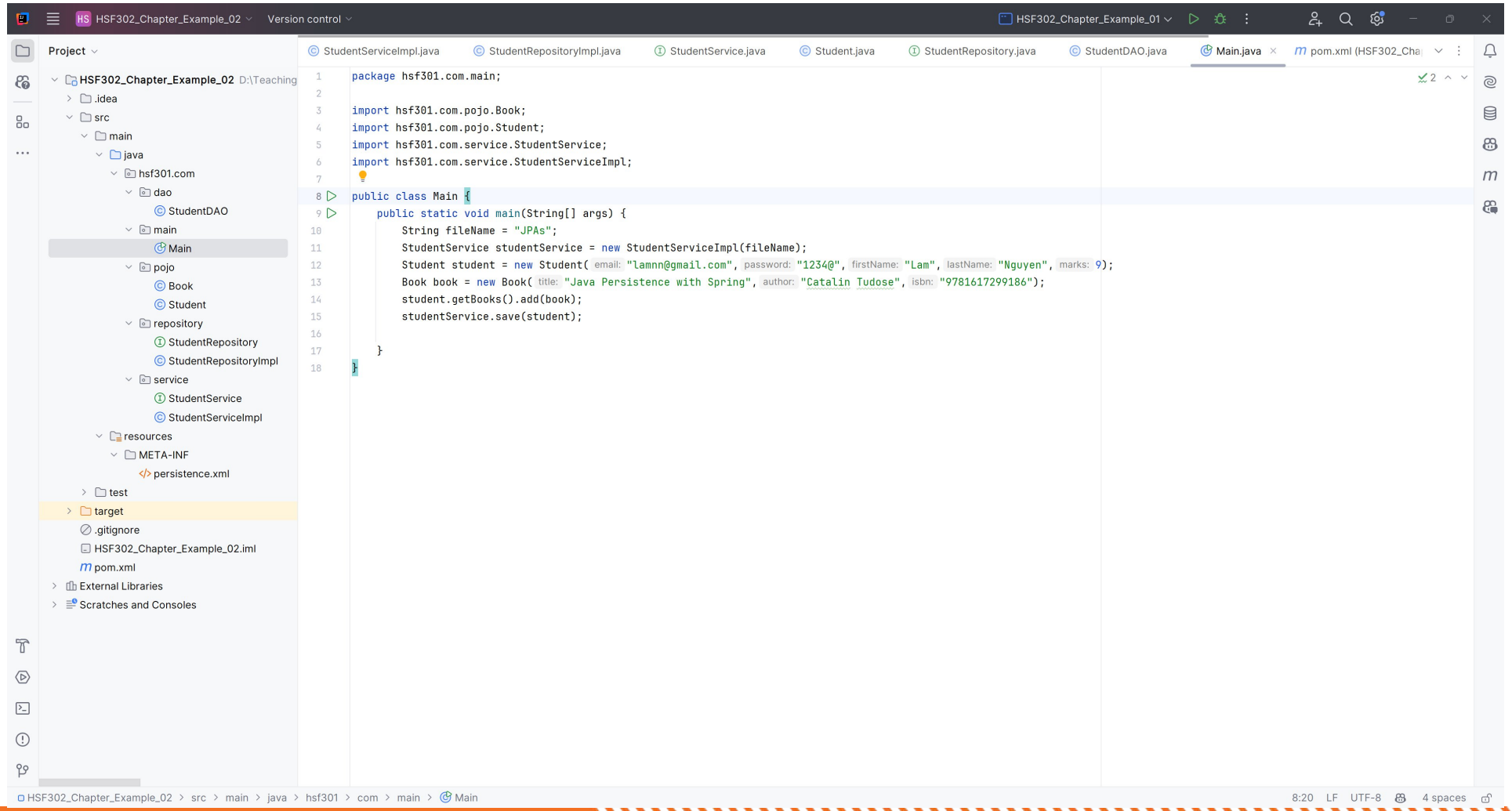
Create StudentService



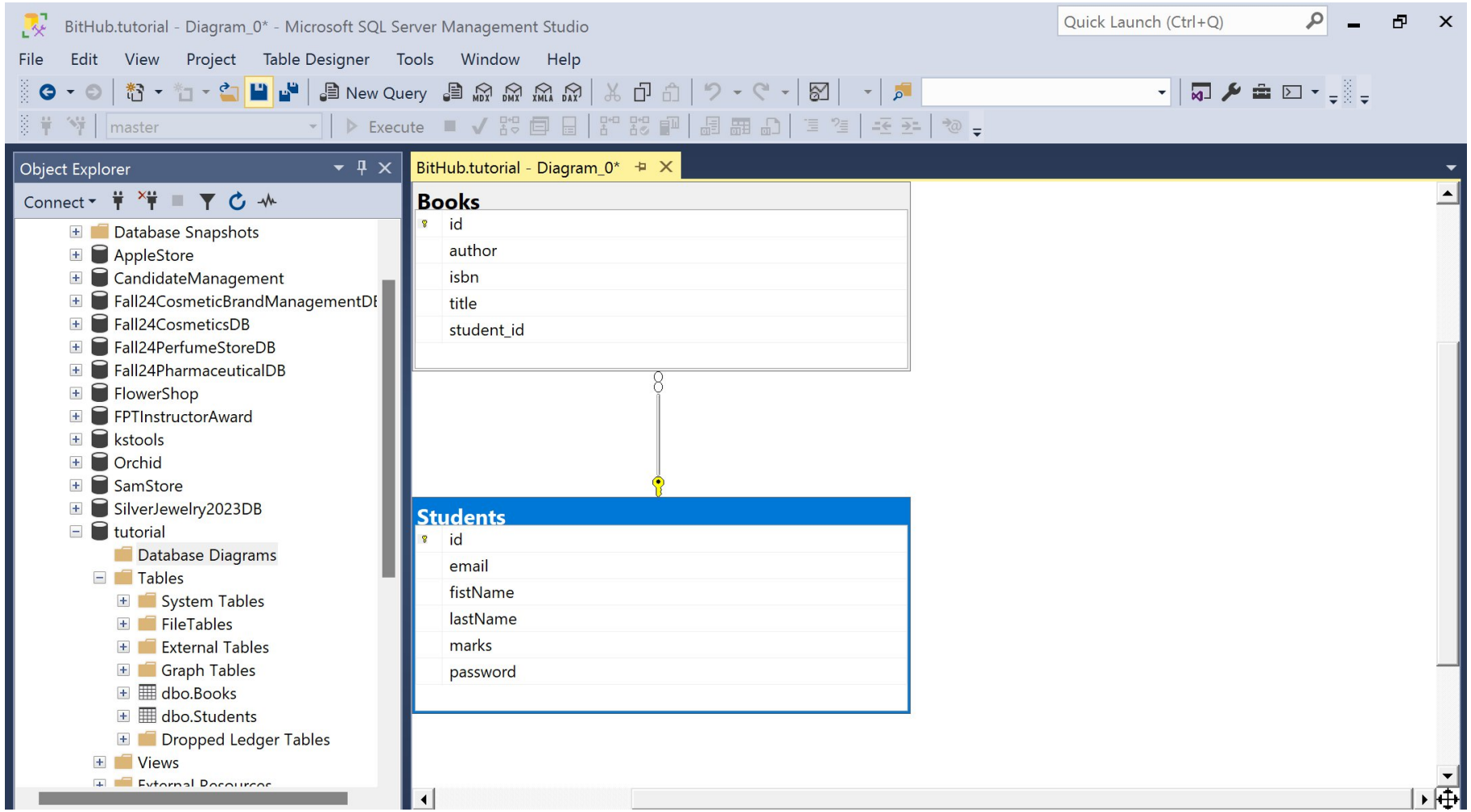
Create StudentServiceImpl



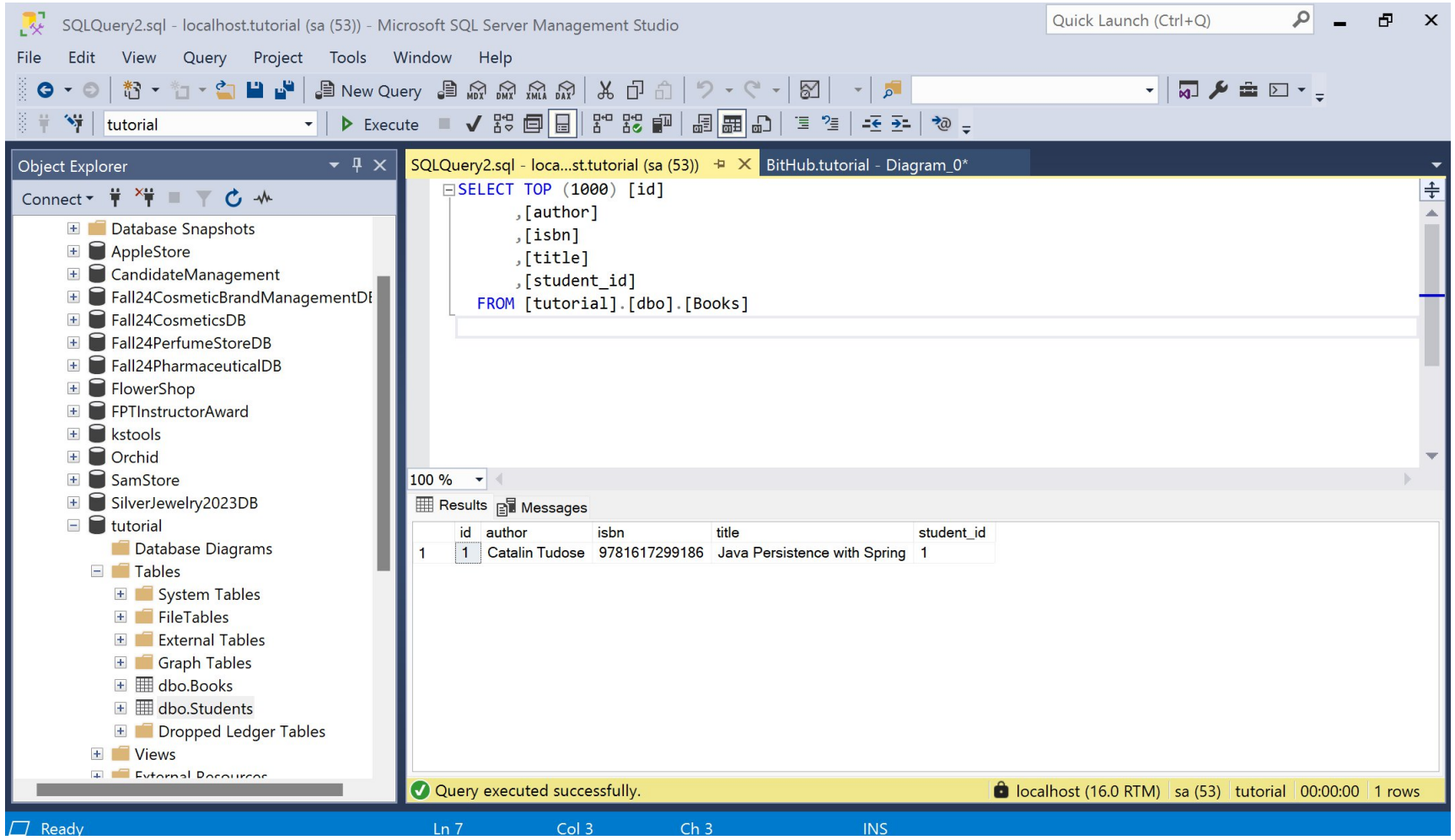
Create Main function



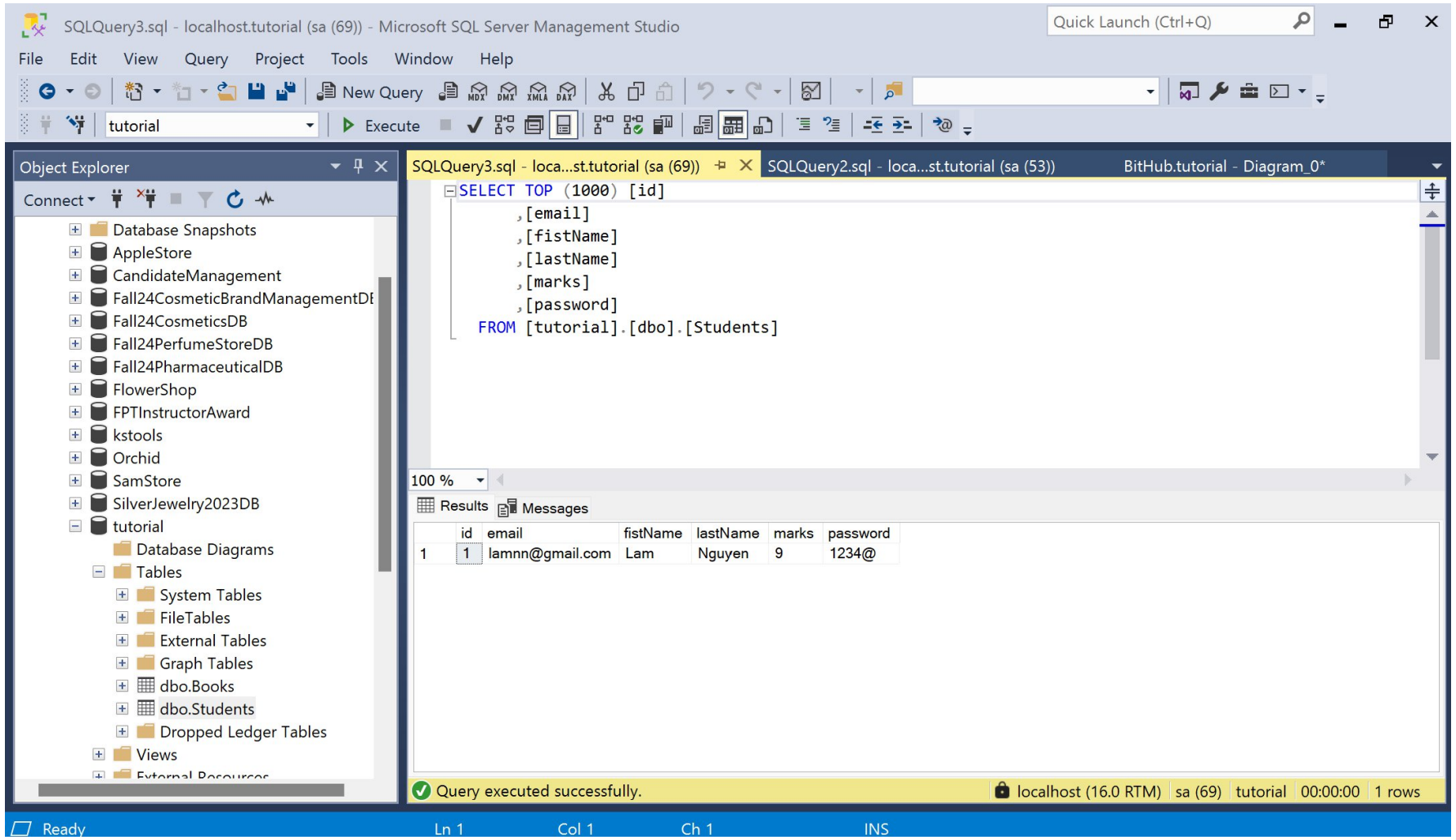
Result


BitHub.tutorial - Diagram_0* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Project Table Designer Tools Window Help
New Query
Execute
master
Object Explorer
Connect
Database Snapshots
AppleStore
CandidateManagement
Fall24CosmeticBrandManagementDB
Fall24CosmeticsDB
Fall24PerfumeStoreDB
Fall24PharmaceuticalDB
FlowerShop
FPTInstructorAward
kstools
Orchid
SamStore
SilverJewelry2023DB
tutorial
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Books
dbo.Students
Dropped Ledger Tables
Views
BitHub.tutorial - Diagram_0*
Books
id
author
isbn
title
student_id
Students
id
email
fistName
lastName
marks
password

Result


SQLQuery2.sql - localhost.tutorial (sa (53)) - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
tutorial
Execute
Object Explorer
Connect
Database Snapshots
AppleStore
CandidateManagement
Fall24CosmeticBrandManagementDB
Fall24CosmeticsDB
Fall24PerfumeStoreDB
Fall24PharmaceuticalDB
FlowerShop
FPTInstructorAward
kstools
Orchid
SamStore
SilverJewelry2023DB
tutorial
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Books
dbo.Students
Dropped Ledger Tables
Views
SQLQuery2.sql - loca...st.tutorial (sa (53))
BitHub.tutorial - Diagram_0*
SELECT TOP (1000) [id]
,[author]
,[isbn]
,[title]
,[student_id]
FROM [tutorial].[dbo].[Books]
100 %
Results
Messages
id author isbn title student_id
1 1 Catalin Tudose 9781617299186 Java Persistence with Spring 1
Query executed successfully.
localhost (16.0 RTM) sa (53) tutorial 00:00:00 1 rows
Ready
Ln 7 Col 3 Ch 3 INS

Result

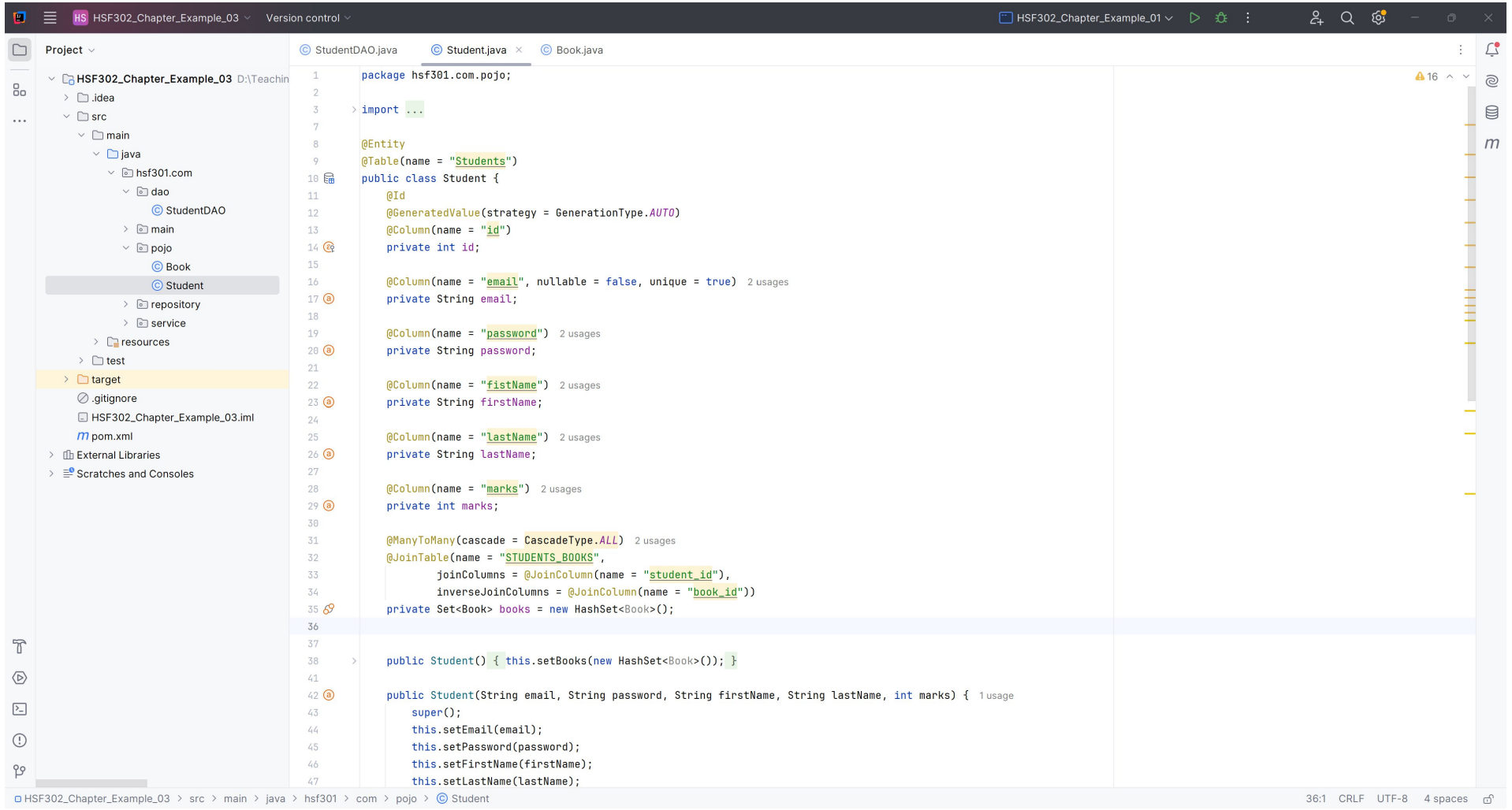

SQLQuery3.sql - localhost.tutorial (sa (69)) - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
Execute
tutorial
Object Explorer
Connect
Database Snapshots
AppleStore
CandidateManagement
Fall24CosmeticBrandManagementDB
Fall24CosmeticsDB
Fall24PerfumeStoreDB
Fall24PharmaceuticalDB
FlowerShop
FPTInstructorAward
kstools
Orchid
SamStore
SilverJewelry2023DB
tutorial
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Books
dbo.Students
Dropped Ledger Tables
Views
SQLQuery3.sql - loca...st.tutorial (sa (69))
SQLQuery2.sql - loca...st.tutorial (sa (53))
BitHub.tutorial - Diagram_0*
SELECT TOP (1000) [id]
,[email]
,[fistName]
,[lastName]
,[marks]
,[password]
FROM [tutorial].[dbo].[Students]
100 %
Results
Messages
id email fistName lastName marks password
1 1 lamnn@gmail.com Lam Nguyen 9 1234@
Query executed successfully.
localhost (16.0 RTM) sa (69) tutorial 00:00:00 1 rows
Ready
Ln 1 Col 1 Ch 1 INS

FPT Education
FPT UNIVERSITY

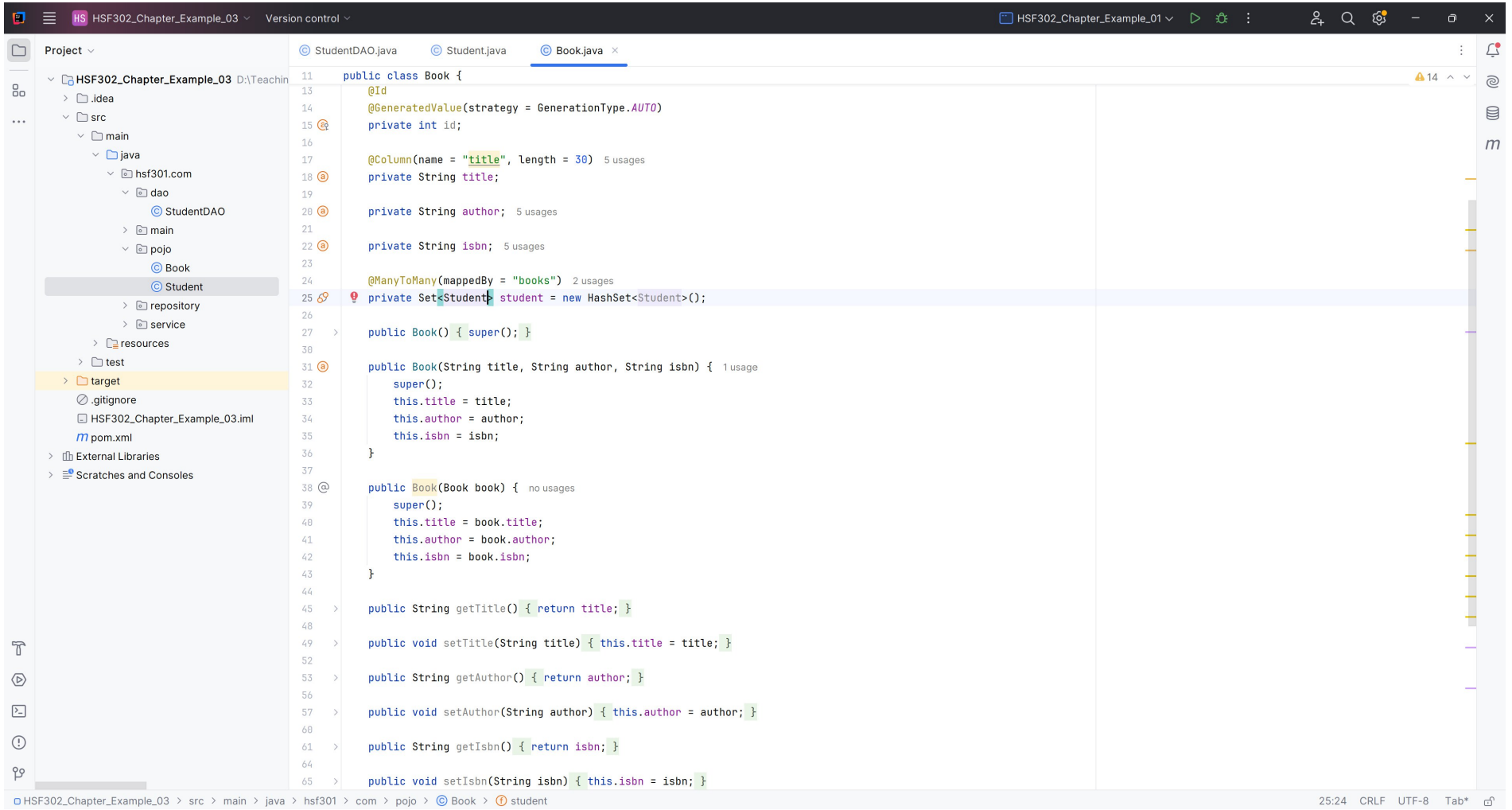
Java

Demo JPA
(Many To Many)

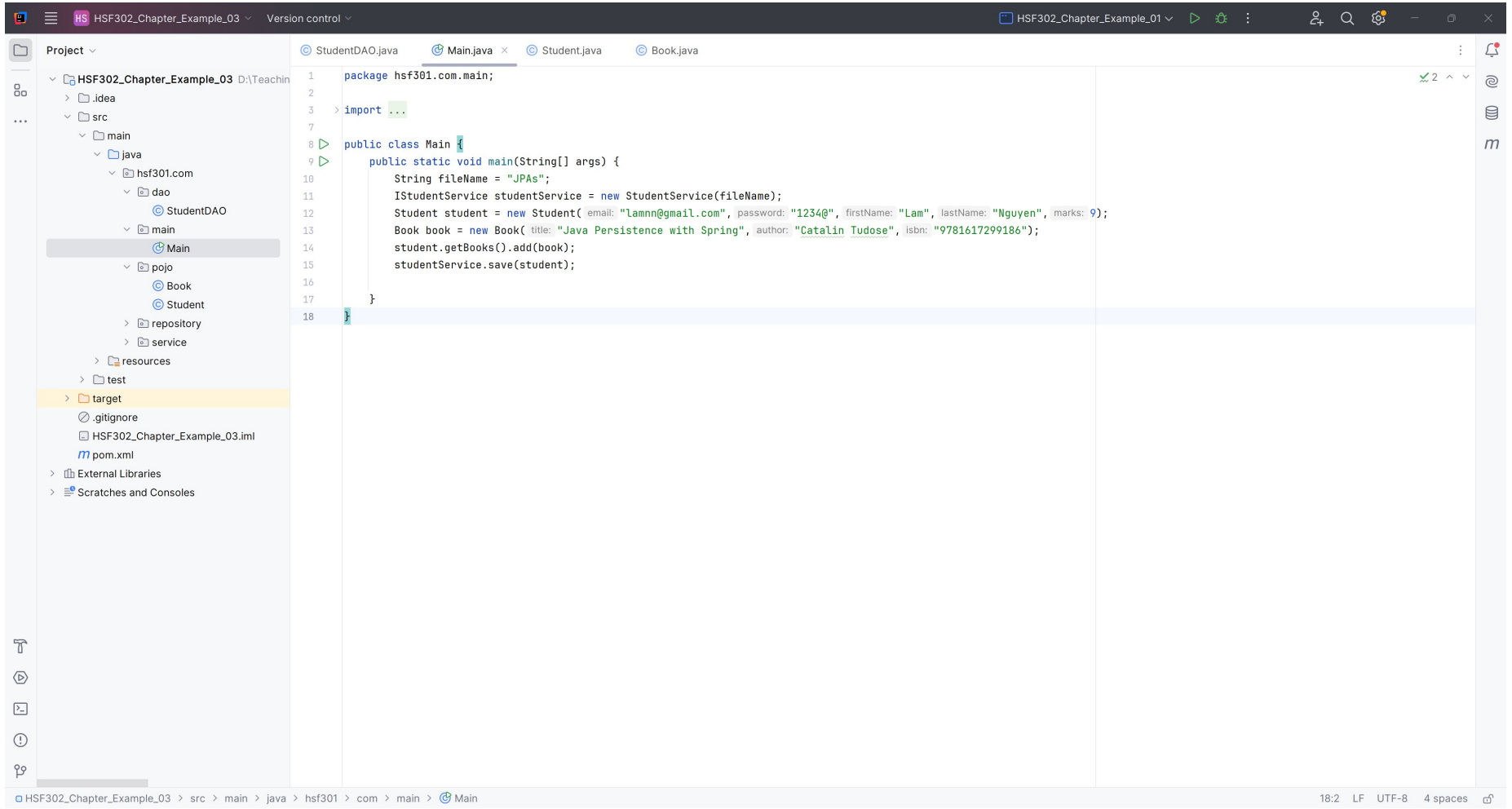
2. Create Student.java in Pojo's Package



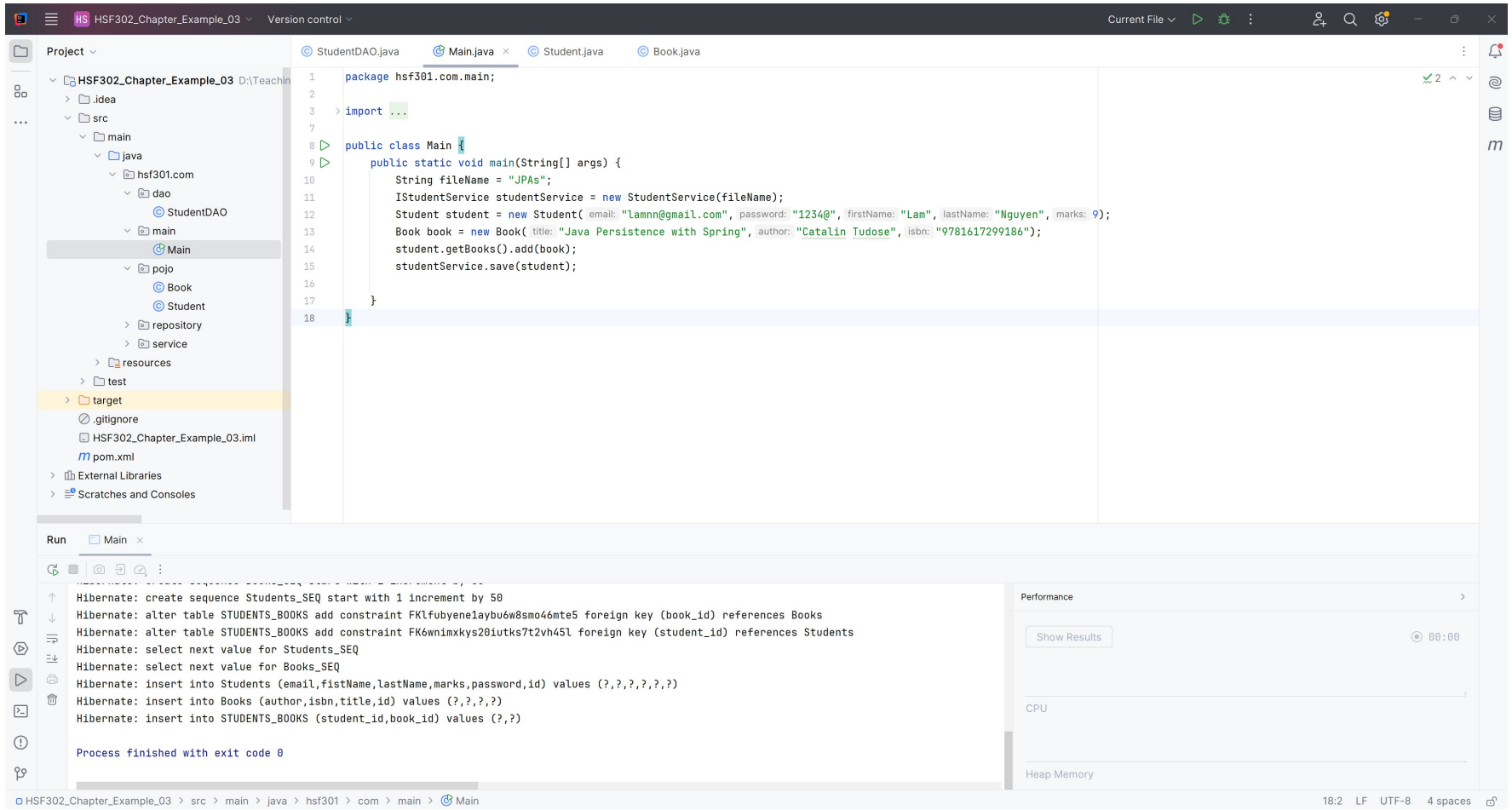
1. Create Book.java in Pojo's Package



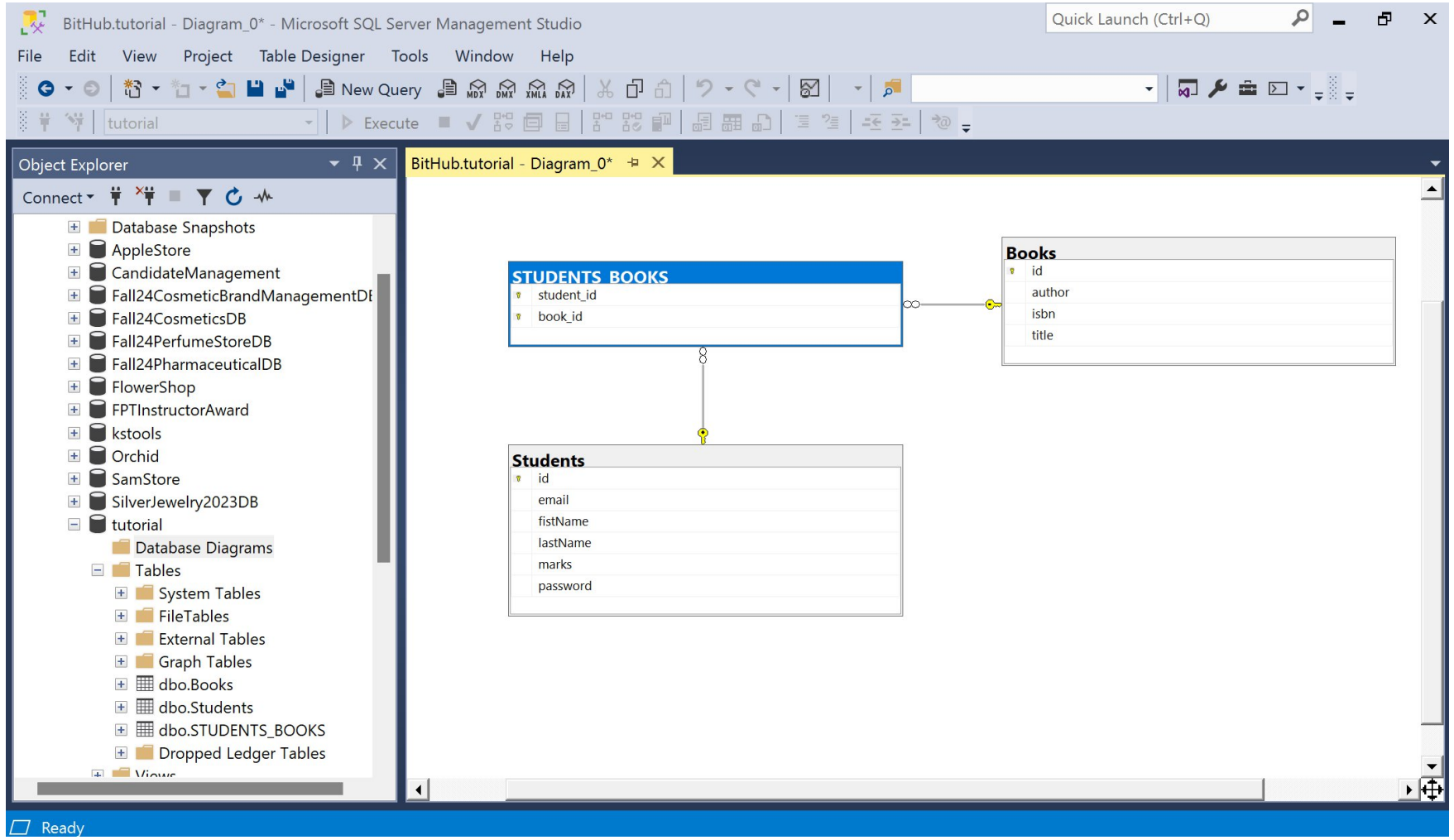
3. Run Program



4. Result


HSF302_Chapter_Example_03
Version control
Current File
Project
StudentDAO.java
Main.java
Student.java
Book.java
HSF302_Chapter_Example_03 D:\Teachin
.idea
src
main
java
hsf301.com
dao
StudentDAO
main
Main
pojo
Book
Student
repository
service
resources
test
target
.gitignore
HSF302_Chapter_Example_03.iml
pom.xml
External Libraries
Scratches and Consoles
package hsf301.com.main;
import ...
public class Main {
public static void main(String[] args) {
String fileName = "JPAs";
IStudentService studentService = new StudentService(fileName);
Student student = new Student(email: "lamnn@gmail.com", password: "1234@", firstName: "Lam", lastName: "Nguyen", marks: 9);
Book book = new Book(title: "Java Persistence with Spring", author: "Catalin Tudose", isbn: "9781617299186");
student.getBooks().add(book);
studentService.save(student);
}
}
Run
Main
Hibernate: create sequence Students_SEQ start with 1 increment by 50
Hibernate: alter table STUDENTS_BOOKS add constraint FKlfubyene1aybu6w8smo46mte5 foreign key (book_id) references Books
Hibernate: alter table STUDENTS_BOOKS add constraint FK6wnimxkys20iutks7t2vh45l foreign key (student_id) references Students
Hibernate: select next value for Students_SEQ
Hibernate: select next value for Books_SEQ
Hibernate: insert into Students (email,fistName,lastName,marks,password,id) values (?,?,?,?,?,?)
Hibernate: insert into Books (author,isbn,title,id) values (?,?,?,?)
Hibernate: insert into STUDENTS_BOOKS (student_id,book_id) values (?,?)
Process finished with exit code 0
Performance
Show Results
00:00
CPU
Heap Memory
HSF302_Chapter_Example_03 > src > main > java > hsf301 > com > main > Main
18:2 LF UTF-8 4 spaces

5. Result


BitHub.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
Database Snapshots
AppleStore
CandidateManagement
Fall24CosmeticBrandManagementDB
Fall24CosmeticsDB
Fall24PerfumeStoreDB
Fall24PharmaceuticalDB
FlowerShop
FPTInstructorAward
kstools
Orchid
SamStore
SilverJewelry2023DB
tutorial
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Books
dbo.Students
dbo.STUDENTS_BOOKS
Dropped Ledger Tables
BitHub.tutorial - Diagram_0*
STUDENTS_BOOKS
student_id
book_id
Books
id
author
isbn
title
Students
id
email
fistName
lastName
marks
password
Ready

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