# JDBC Ecomm Project Description

**Spring 21**

**Projects will be collected on 4/7 from the UTD Box folder found in the ELearning home page.**

# Supporting YouTube Video

A video describing the details of developing this project can be found on YouTube. The URL is: <https://youtu.be/d21ItF0lKJ4>.

Note: Please let Prof. Christiansen know if you find any problems / inaccuracies with these videos.

# Introduction

This project is an introduction to developing services and DAOs (Data Access Objects) that support an application’s persistence requirements. The delivered services will persist a set of entity classes that are provided in the project materials.

# Project Goals

The following are the goals of this project:

* To build a ‘simple\_company’ schema from the given UML entity classes.
* To deliver an SQL script that will create the ‘simple\_customer’ schema on the instructor’s PC for grading purposes. One method of accomplishing is to use the MySQL workbench to create the schema in an ER diagram and then generating the schema’s SQL DDL by exporting the ER diagram. Teams might also build the SQL DDL by hand.
* To populate the schema tables with sample data using the application **PopulateTables.java** that has been provided in the testing project materials.
* To implement and deliver a set of three Services that successfully compile against, and can pass the Unit Tests that have been provided in the JDBCProjectTesting project provided in the project’s materials.
  + To implement and deliver a Customer persistence service based on a provided CustomerPersistenceService interface that will be used to CRUD Customer entities against the delivered schema. This implementation must pass the Unit Test provided in the testing project materials.
  + To implement and deliver a Product persistence service based on a provided ProductPersistenceService interface that will be used to CRUD Purchase entities against the delivered schema. This implementation must pass the Unit Test provided in the testing project materials.
  + To implement and deliver a Purchase persistence service based on a provided PurchasePersistenceService interface that will be used to CRUD Purchase entities against the delivered schema. This implementation must pass the Unit Test provided in the testing project materials.
* To implement and deliver a set of five DAO classes based on provided interfaces that will be used to CRUD information against the delivered schema. These implementations must pass the Unit Tests provided in the project materials.
* **To deliver a library jar file that will be used to grade the project**. This jar will contain the contents of the JDBCProjectForStudents project and will be executed on the instructor’s PC and database to evaluate the quality of the team’s work. See the section “Exporting an Eclipse Project as a Library JAR File” at the end of this document.

# Materials to be Delivered

Each team will package their delivered materials in a ZIP file archive. Include the team number in the zip archive name e.g. ‘JDBC Team 99.zip’. Deliver archive file using the UTD Box folder linked on the course homepage.

* A completed template file: “JDBC Project Evaluation – Team XX.docx” that includes contributing team member names and their Net-IDs, and the module they are responsible for completing. The table columns in red.
* An execution ready SQL script that generates the simple\_company schema when executed from MySQL workbench on the instructor’s PC.
* A library jar file exported from your JDBCProjectForStudents project. This jar used by the instructor’s JDBCProjectTesting Eclipse project to executes the unit tests.
* The JDBCProjectForStudents project including source code. You deliver the project by copying the entire JDBCProjectForStudents directory into the UTD Box folder. Submissions that do not include the entire project will not be graded and will be considered late when resubmitted.

## UTD Box Delivery

Only the UTD Box folder linked in the course home page can be used to deliver the team’s zip file archive.

# Grading Criteria

Grades are awarded based on both team and individual contributions. The items marked in **green** will be evaluated on a team basis. The items marked in **red** describe the development of the three modules and will be evaluated on an individual team member bases.

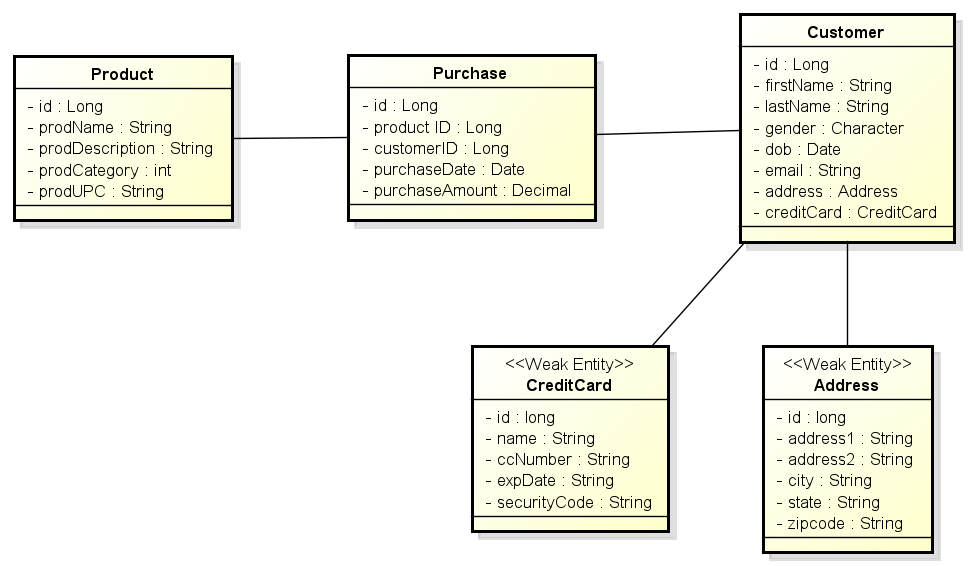
1. **10 Points**. Providing a library JAR file from JDBCProjectForStudents that correctly compiles against the Eclipse project JDBCProjectTesting on the instructor’s personal Eclipse workspace. This will be accomplished by importing the submitted jar into the JDBCProjectTesting project build path.
2. **20 Points**. Providing a SQL script that when executed on MySQL Workbench, creates a schema named ‘simple\_company’ (tables, etc.) on the instructor’s MySQL installation. (The schema name is important because it is reflected in the JDBC URL used to setup the DataSource for testing). The schema must reflect the structure of the entity classes provided.
3. **20 Points**. Executing the application PopulateTable.java and successfully populating the Customer, Address, CreditCard, Product, and Purchase tables created by your team’s schema in step 2.
4. **50 Points**. Executing DAO and Service unit tests provided in the JDBCProjectTesting packages \*.unitTesting.dao and \*.unitTesting.service packages without error. See the section Service Modules.

# Project Requirements

* Use JDK 13.02 (This is more of a suggestion). Download from <https://www.oracle.com/java/technologies/javase-jdk13-downloads.html>
* Teams must not modify the code / contents of the Eclipse project JDBCProjectTesting. Doing so will likely result in the loss of points when the unit tests in the instructor’s own unmodified project no longer work.
* The application DataSource will be configured with the contents of the property file ‘dbconfig.properties’ in the JDBCProjectTesting project. A property file has been provided with attaches to the DBMS running on the local machine. You will need to modify the id (maybe) and password. Do not change the default schema name (simple\_company).
* The CustomerDAO, ProductDAO, and PurchaseDAO create() methods must throw a DAOException if the given Customer, Purchase, and Product has a non-null ID attribute.
* The CustomerDAO, ProductDAO, and PurchaseDAO create() methods must return the given entities with the ID attribute assigned the key value provided by the auto-increment primary key column. An example of how this is accomplished in code has been provided.
* The DAO retrieve\*() methods must return null if retrieving a non-existent ID.
* The implementation of the average
* The implementation of retrievePurchaseSummary() must use SQL aggregation functions to calculate the minimum, maximum, and average purchase made by the given Customer.

# Entity Classes

The following classes represent the entities that will be persisted by the delivered services. Note that Java source files implementing these classes have been provided with the project materials. (JDBCProjectForStudent.zip)



## Please Note These Features…

The class attribute Address.address2 is an optional attribute and should be created NULL’able in the schema.

The Purchase class maintains two references to the Customer and Product objects associated with the Purchase instance. These two Longs represent the values of the Customer and Product primary keys. Normally, these relationships would be references to the actual Customer and Product instances, but are being simplified for the project.

Both CreditCard and Address tables should be implemented with a foreign key back to their owning Customer.

# Service Modules

In the interest of ensuring that each team member contributes to the development of the project, the implementation has been divided into three modules: Customer, Product, and Purchase. One or more team members is to be responsible for one of the three modules. Because the Customer module is significantly larger than the others, two team members can be assigned to developing the Customer module. The correct execution of all three modules will be evaluated on a team basis for 10 points.

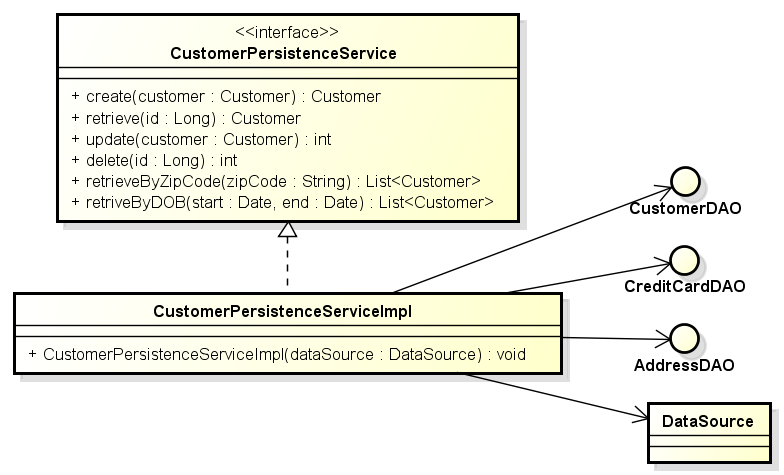
The following illustrates the Service Module and associated DAO interfaces that will be delivered with the project. The services are provided a DataSource in their constructors. Services use the DataSource to build JDBC Connections that are passed to the DAO methods to CRUD and query for instances of the associated entities.

The delivered classes must implement these interfaces for the grading / testing to execute correctly on an individual basis. Note that Java interfaces for these services (and DAOs) have been provided with the project materials.

## Customer Persistence Module

The CustomerPersistenceService is responsible for managing the persistence of Customer, Address, and CreditCard entities.

This service is to be implemented so that every operation on a Customer also manipulates its associated Address and CreditCard instances. For example, retrieving a Customer though its ID, also returns the Customer’s Address and CreditCard. Deleting a Customer also removes its Address and CreditCard. Etc.

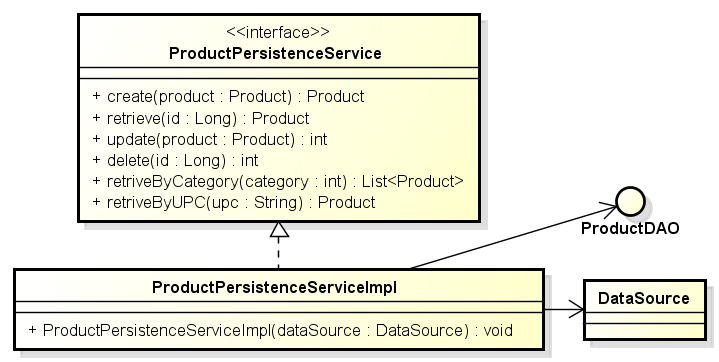


The CustomerPersistenceService provides the standard CRUD operation that allows the persistence and management of these instances. The service also provides operations that query for collections of Customer (Address & CreditCard) based on search criteria zip-code and date of birth ranges.

Notice that Address and CreditCard have not been provided persistence services in this design. This is because these are weak entities and attached to their Customer strong entity objects and all operations on a Customer are applied to their dependent Address & CreditCard objects.

## Product Persistence Module

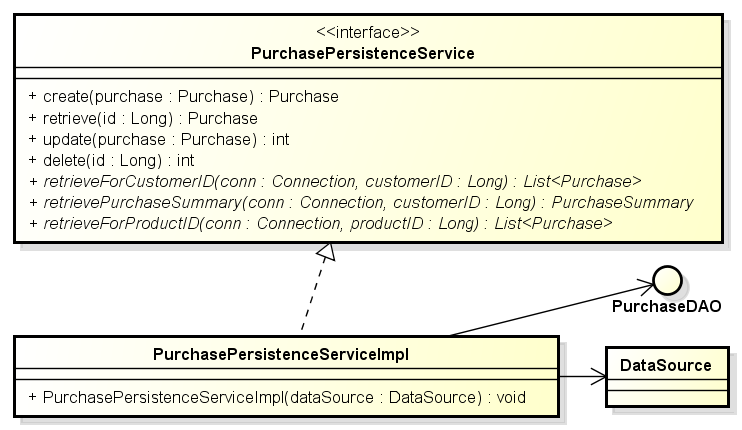
The ProductPersistenceService provides the standard CRUD methods for Products maintained in the ecommerce sites inventory. The service also provides operations that allow for the query for Products in certain product categories and retrieving Products by their UPC (Universal Product Code).



## Purchase Persistence Module

The PurchasePersistenceService provides the standard CRUD methods for Purchases made by Customers for Products. The services also allows for the retrieval of Purchases made by specific Customer, and for the retrieval of Purchases for specific Products.

The method retrievePurchaseSummary() must use SQL aggregation functions to calculate the minimum, maximum, and average purchase made by the given Customer. These three values are returned to the service’s client in an instance of PurchaseSummary.

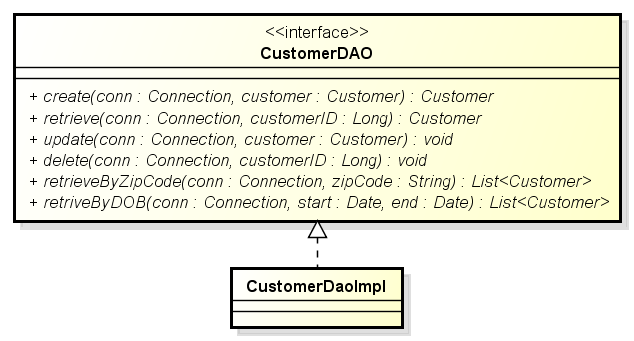


# DAO Interfaces

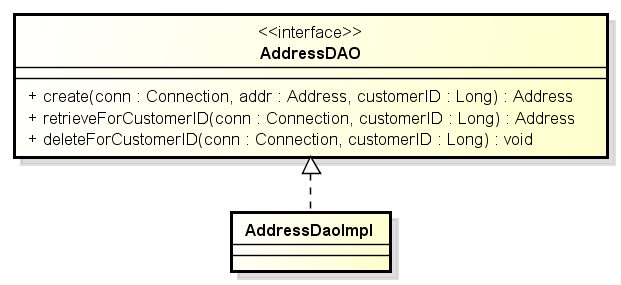
The following illustrates the DAO interfaces that will be implemented and delivered with the project. The delivered classes must implement these interfaces for the grading / testing to execute correctly on an individual basis. These Java interfaces have been provided with the project materials.

Notice that each of the DAO methods requires an open JDBC Connection instance that the operation uses to interact with the DBMS. The reason for passing the Connection to each method is that it allows the service classes that utilize the DAOs to manage the connection and to create transactions that span several DAO / JDBC / SQL operations.

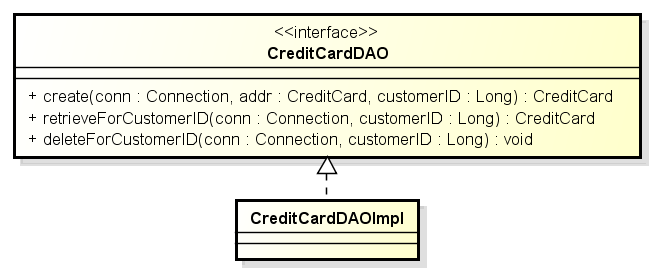
## CustomerDAO



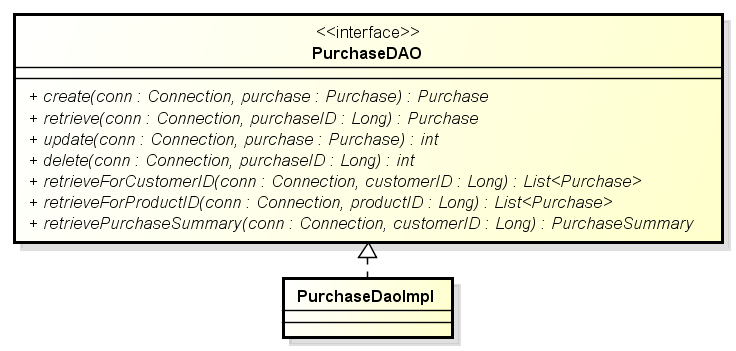
## AddressDAO



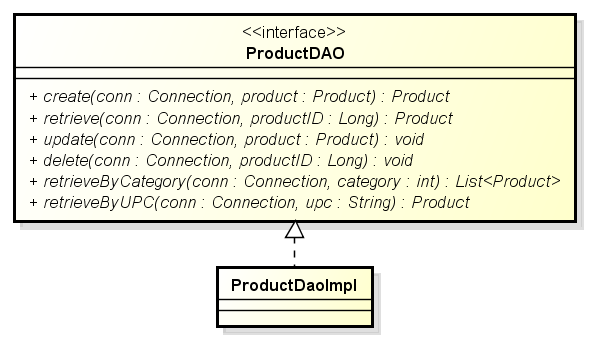
## CreditCardDAO



## PurchaseDAO



## ProductDAO



# Provided Materials

You have been provided two archived Eclipse projects (zip files) that will be imported into your Eclipse workspace. The last section of this document provides instruction on how the import is accomplished.

These projects are:

## JDBCProjectForStudents.zip

This project provides the basis for your DAO and Services implementations. This includes the Entity classes, DAO and Service interfaces as Java source. Your implementation will be written within this project, using these classes / definitions. The library jar file you create and submit for grading will be built from the work performed in this project.

**Do not change either the provided interfaces or the packaging structure. If changes are made, the unit tests will not compile (much less execute) and your team receives no credit for the work.**

## JDBCProjectTesting.zip

This project contains programs and unit tests that you will use to configure and test your DAO and service implementations.

**Teams must not modify the code / contents of this project (With one exception. See section “Notes on Unit Test”)**. Changing the contents of the testing project will likely result in the unit tests in the instructor’s own unmodified testing project no longer working (not passing) and the loss of project points.

The JDBCProjectTesting project includes:

1. The application PopulateTables.java uses built-in JDBC to populate the database generated by your team’s schema. This includes a number of CSV files containing the data used to populate the CUSTOMER, ADDRESS, CREDIT\_CARD, etc. tables.
2. A number of Junit unit test classes that will test the correctness of your DAO implementations.
3. A number of Junit test classes that will test the correctness of your CustomerPersistenceService, PurchasePersistenceService, and ProductPersistenceService implementations.
4. The class DataSourceManager.java that is used to initialize (configure) and provide access to a singleton instance of a DataSource used by the unit test classes. **Note**: The text file dbconfiguration.properties contains the DataSource configuration and must be updated with the parameters needed to access your MySQL installation. You must not modify the URL’s schema (simple\_company).
5. A sample Java class SampeCreateMethod that is intended to provide an example of extracting the auto-increment primary key assigned to entity classes that are created (inserted).

# Suggested Development Process

The following steps detail a suggested process for executing this project.

1. Import both projects (Student and Testing) into your Eclipse workspace. **Note** that initially the project JDBCProjectForStudent will contain compile errors as the DAO and service implementations have not been completed (that is for you to do).
2. Be sure that the Testing project’s Eclipse build path includes the Student project.
3. You can use the main() application embedded in the DataSourceManager to verify that your application dbconfig.properties is correctly configured to connect to your running MySQL server. NOTE: You must have a schema and table installed on your DBMS and update the select in the DataSourceManager main().
4. Define a schema that meets the needs of the given entity classes. It is suggested that you use MySQL Workbench and that you export the CREATE SCHEMA SQL you will need to submit from workbench. The schema’s name must be ‘simple\_company’.
5. Implement some of the DAO interfaces against your schema. That is, implement the DAO with queries that are being utilized by the PopulateTable.java application. You need those services in place to populate the tables with test data in the next step (6).
6. Use the program PopulateTables.java (provided in the Test project) to populate the database from the data contained in the CSV files. Note that executing this application requires completing the insert\* methods in the DAO Implementation classes in the student development project.
7. Implement the remaining DAO interfaces against your schema. That is, complete the DAO implementation classes with queries that reflect your schema.
8. Use the provided DAO unit tests (see the package \*.unitTesting.dao) to validate the ‘correctness’ of your DAO implementations.
9. Implement the CustomerPersistenceService interface against your DAOs.
10. Implement the ProductPersistenceService interface against your DAOs.
11. Implement the PurchasePersistenceService interface against your DAOs.
12. Use the provided service unit test (see the package \*.unitTesting.service) to validate the correctness of your service implementations.
13. When all the tests work correctly, or you run out of time, package ONLY the contents of the JDBCProjectForStudents project into a library jar file that will be submitted for grading. See the section “Exporting an Eclipse Project as a Library JAR File” at the end of this document.
14. Test that the library jar file works correctly by including the library jar in the JDBCProjectTesting project’s build path. NOTE: You will need to remove the project JDBCProjectForStudents from the testing project’s build path for the validation to work.

**I would appreciate any feedback concerning problems students encountered with this document or video provided for this project**

# Notes on Unit Tests

Some of the unit tests rely on there being rows with specific IDs. For example “AddressDAOTest.java” has the constant customerID that defaults to 7. This test will fail a Customer in with this ID is not found in the database.

This customer will be created when the schema is re-installed and the populate application is run. This is because the auto-increment will start over from zero.

You can also change the value of the constant to an ID that is found in the corresponding table. This is the only change to the unit test source code that is allowed.

Other unit tests rely on a foreign key value that is not found in the database. For example AddressDaoTest.testForeignKey() expects an SQLException that will be caused by an invalid CustomerID value.

# Issues You May Encounter

## Wow! AWS MySQL is Slow.

It is, but I think there are a few things going on here.

1. We are using the free tier MySQL server hosted on a VM with a low bandwidth IOPS (I/Os Per Second)
2. We are sending / receiving queries from our PCs across the internet. An actual AWS deployment would have the application server and DBMS running on the same AWS subnet.

## AWS is case sensitive on table names (and columns?).

If a table is created with the name ‘CUSTOMER’ but used in a query ‘customer’, the MySQL server will respond to the query with a table not found error. Be sure that your table names and query names match case.

See <https://stackoverflow.com/questions/11165944/how-to-change-mysql-table-names-in-linux-server-to-be-case-insensitive>

This error is due to the fact that file names are case insensitive on Windows and case sensitive on Linux. Nothing to do with AWS specifically.

## Invalid default value for timestamp column (purchase date)

An error message generated when executing the DDL generated by MySQL workbench after completing the schema and exporting from workbench.

See <https://stackoverflow.com/questions/9192027/invalid-default-value-for-create-date-timestamp-field>

Need to remove “TRADITIONAL” from the @OLD\_SQL\_MODE generated by workbench.

## Unable to execute DataSourceManager.main()

Unable to execute the test application which verifies the correct configuration of the DataSource i.e. URL / ID / PASSWORD.

I found that if you leave a space at the end of the URL line in the properties configuration file (…/simple\_company \n) the MySQL driver will not recognize the schema on the server. Remove the space. IDK if this is AWS specific.

## MySQLTransactionRollbackException

While testing on my laptop I’ve experienced a transaction hang for ~30 seconds and then fail with the exception:

com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException

This error has only occurred on one of several machines that I’ve tested on and only occurs when I run all the unit test in a selected package i.e. in rapid succession. When I run the test individually, the error does not occur.

I’ve looked into this error and can find nothing wrong with my implementation. My advice is that if your team experiences this error under the above conditions, you can ignore it.

# Using Eclipse

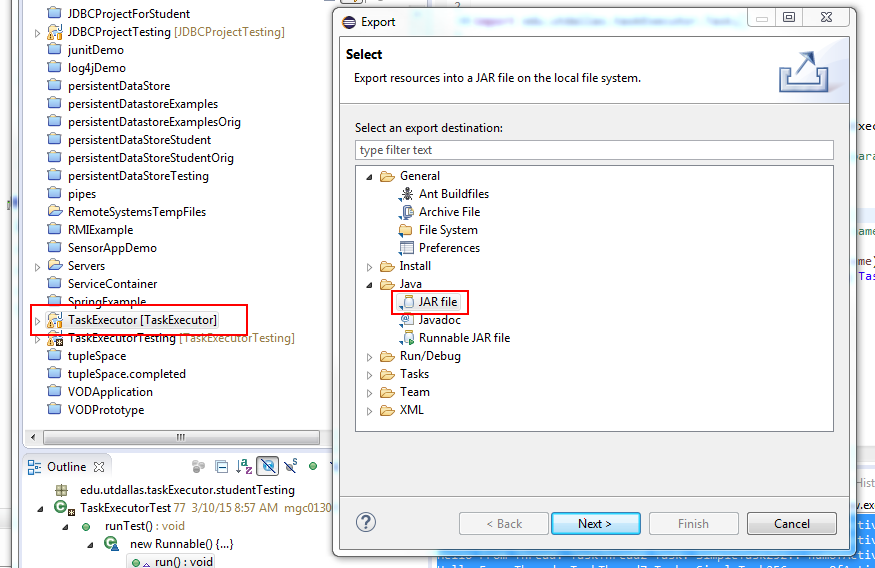
## Executing Unit Tests in Eclipse

1. You may need to install the JUnit library on the testing project.
2. You can select individual JUnit test cases (Test Classes), right click, and select “Run As JUnit Test”. This can also be done on a package to run all the tests in the package or on the project to run all test cases in the project.

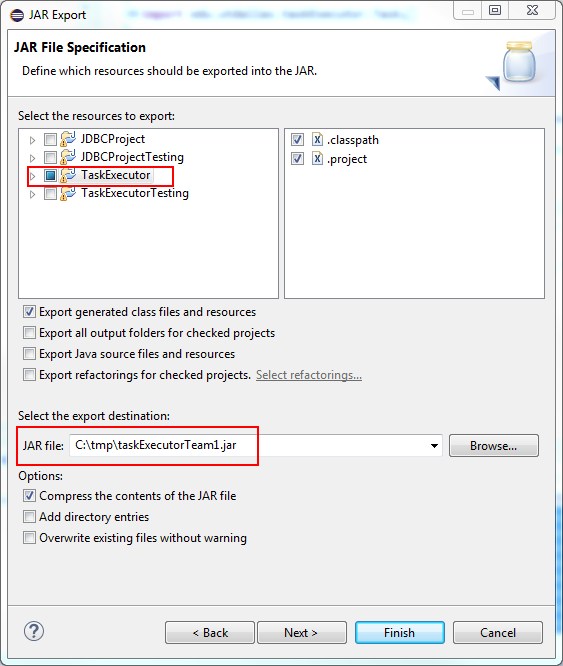
## Exporting an Eclipse Project as a Library JAR File

This section provides a procedure describing of how to export the project containing your project implementation as a library .jar file for submission.

1. Select the project that you wish to export.
2. Use the right mouse button, or the file menu, to select the Export feature.
3. Select Java >>JAR File as shown below, and then Next.



1. On the JAR Export panel, make sure that the desired project is selected and enter the path and file name for the exported library jar file.
2. Select Finish and the export operation will be completed.

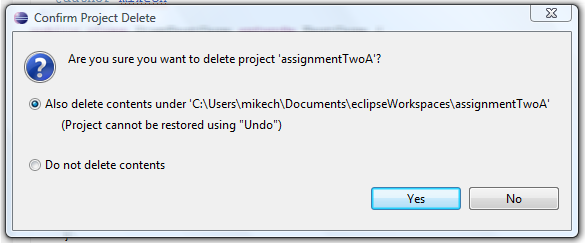


## Importing a Project into Eclipse from a Project Zip File

This programming assignment provides two exported projects that must be imported into your Eclipse workspace. These projects are provided as zip file archives that will be one of the files that can be downloaded from the WebCT assignment. The zip archive may contain sample code or a project template that can be used as a starting point for your efforts. You will be importing the project zip archive into your workspace.

**Optional: Removing existing projects with the same name from the workspace**

You cannot import a project with the same name into the workspace. This means that if you import and try to re-import the project template you must first delete the old project from the workspace. This is accomplished by selecting the existing project from the package explorer and selecting the “Edit > Delete” menu item. This will bring up the dialog shown in the following graphic. Notice that the option “Also Delete Contents Under C:\...” is selected**. It is very important that this option is selected** so that the project files are removed from you workspace

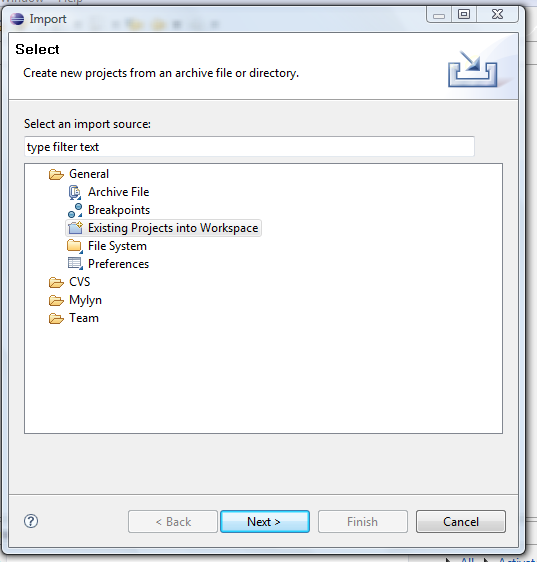


A this point the Old project will be have been removed from your workspace and you may begin importing the project template

## Importing the Project

The process for importing the template project is a follows.

Open the import wizard using the “File > Import” menu item. This brings up the import dialog shown in the following graphic. Make sure to select the “Existing Projects into Workspace” option (under General) and press Next.



This brings up the following import dialog. There are a few import things to note:

1. You need to select the “Select archive file” option and then press browse to select the project template archive (zip) file.
2. When the file opens, you need to select the project.
3. Press Finish and the project will be imported into your workspace.

