# JDBC Game Project Description

**Projects are will be collected from UTD BOX on 3/13 @ 12 P.M.**

# 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: This video was built around the Ecommerce version of the JDBC project. The development workflows of the dev (student) and testing projects are identical. Differences are outlined in the following sections of this document.

### Differences with the JDBC Ecomm Project

1. The entity classes are different i.e. EComm vs Game.
2. I have included stubbed out implementation of both the DAO and Service implementation classes. This should make things a bit clearer.
3. TBD as issues are discovered.

# Introduction

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

# Project Goals

The following are the goals of this project:

* To build a ‘games’ schema from the given UML entity classes.
* To deliver an SQL script that will create the ‘games’ schema on the instructor’s MySQL DBMS for grading purposes. One method of accomplishing is to use the MySQL workbench to first create the schema as ER diagrams and then generating the schema’s SQL 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** which has been provided in the testing project materials.
* To implement and deliver a set of four Services that successfully compile against, and can pass the Unit Tests that have been provided in the GameJDBCProjectTesting project provided in the project’s materials.
  + To implement and deliver a Game persistence service based on a provided GameService interface that will be used to CRUD Game entities against the schema. The GameService implementation must pass the Unit Test provided in the testing project materials.
  + To implement and deliver a Player persistence service based on the provided PlayerService interface that will be used to CRUD Player entities against the schema. The PlayerService implementation must pass the Unit Test provided in the testing project materials.
  + To implement and deliver a GamesOwned persistence service based on the provided GamesOwnedService interface that will be used to CRUD GameOwned entities against the schema. The GamesOwnedService implementation must pass the Unit Test provided in the testing project materials.
  + To implement and deliver a GamesPlayed persistence service based on the provided GamesPlayedService interface that will be used to CRUD GamesPlayed entities against the schema. This GamesPlayedService 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 schema. The 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 GameJDBCProjectStudent 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 deliver their project using a protected folder on UTD Box that contains the following material.

* A completed template file: “Game 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 games schema when executed from MySQL workbench on the instructor’s PC.
* A library jar file exported from your GameJDBCProjectStudent project. This jar used by the instructor’s GameJDBCProjectTesting Eclipse project to executes the unit tests.
* The GameJDBCProjectStudent Eclipse project including source code. You deliver the project by copying a ZIP archive of the entire GameJDBCProjectStudent 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 UTD Box can be used to deliver the materials described above. No GitHub repositories. Team MUST share their Team’s Box folder with Prof. Christiansen (mgc013000). Teams will email Prof. Christiansen the URL of their team’s UTD Box folder by the due date & time given at the top of this document. The email subject must include the phrase “4347 JDBC Project” and the email must include the team number and Box folder’s URL. The contents of the Box folder will be collected at the given time. Emails that arrive late or folders that do not contain all the materials described above will be graded late.

# Grading Criteria

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

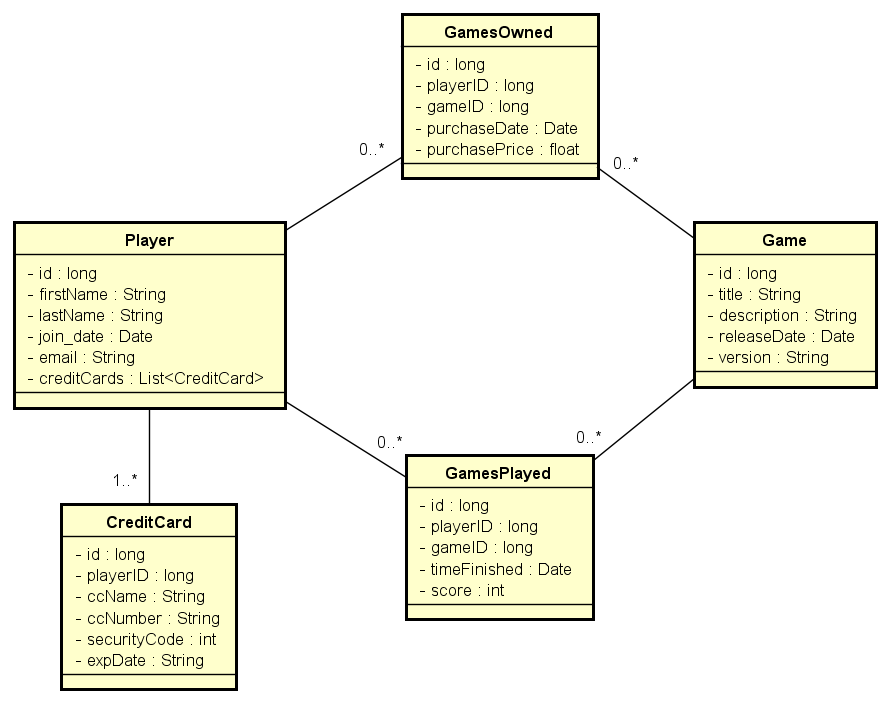
1. **10 Points**. Providing a library JAR file from GameJDBCProjectStudent that correctly compiles against the Eclipse project GameJDBCProjectTesting on the instructor’s personal Eclipse workspace.
2. **20 Points**. Providing a SQL script that when executed on MySQL Workbench, creates a schema named ‘games’ 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. **10 Points**. Executing the application PopulateTable.java provided in the testing project and successfully populating the tables created by your team’s schema in step 2.
4. **10 Points** is awarded on a team basis for the correct implementation of all four modules.
5. **50 Points**. Executing DAO and Service unit tests provided in the GameJDBCProjectTesting packages testSrc.dao and testSrc.service packages without error. See the section Service Modules.

# Project Requirements

* Teams must not modify the code / contents of the Eclipse project GameJDBCProjectTesting. 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 GameJDBCProjectTesting 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 (games).
* The CreditCardDAO, GameDAO, GamesOwnedDAO, GamesPlayedDAO and PlayerDAO create() methods must throw a DAOException if the given CreditCard, Game, GamesOwned, GamesPlayed, and Player has a non-null ID attribute.
* The CreditCardDAO, GameDAO, GamesOwnedDAO, GamesPlayedDAO and PlayerDAO 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.

# Entity Classes

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



## Please Note These Features…

The Player maintains a reference to 1 or more CreditCard. Note that the associated CreditCard are maintained in a List instance variable on Player.

The classes GamesOwned and GamesPlayed maintain their references to Player and Game using the playerID and gameID variables. There are no reference variables for Games or Player on the GamesOwned and GamesPlayed objects.

CreditCard is a “Weak Entity” in the design. That is, CreditCard existence is linked to the owning Player. Player is a Strong Entity in that it maintains a unique identity, and that makes it selectable, etc. The CreditCard table should be implemented with a foreign key back to its owning Player.

# Service Modules

In the interest of ensuring that each team member contributes to the development of the project, the implementation has been divided into four modules: Player, Game, and GamesPlayed and GamesOwned. One or more team members is to be responsible for each of these modules. Because the Player module is significantly more difficult than the others, more weight will be given to its development.

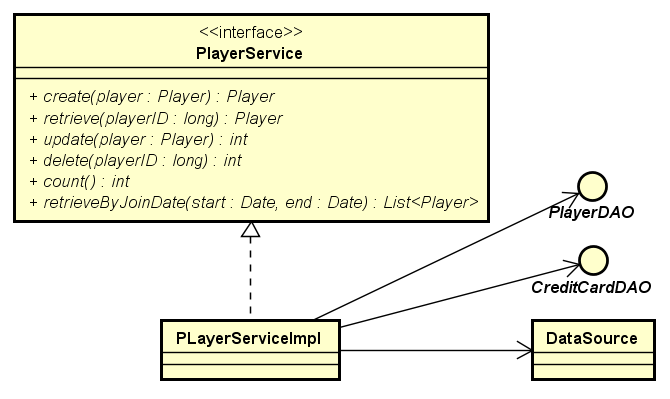
The following illustrates the Service Modules and associated DAO interfaces that will be delivered with the project. The service implementations are provided a DataSource in their constructors. Service implementations use their DataSource to build JDBC Connections that are passed to the DAO methods to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

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

## Player Service Module

The interface PlayerService is responsible for managing the persistence of Player and CreditCard entities.

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

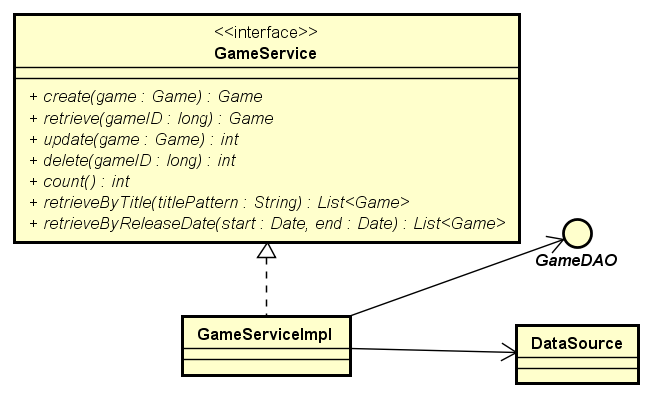


The PlayerService provides the standard CRUD operation that allows the persistence and management of Player. The service also provides an operation that query for collections of Player based on search join date ranges.

Notice that CreditCard has not been provided a persistence service in this design. This is because CreditCard is a weak entity and attached to its Player strong entity object. All CRUD operations applied to a Player instance are applied to their dependent CreditCard instances.

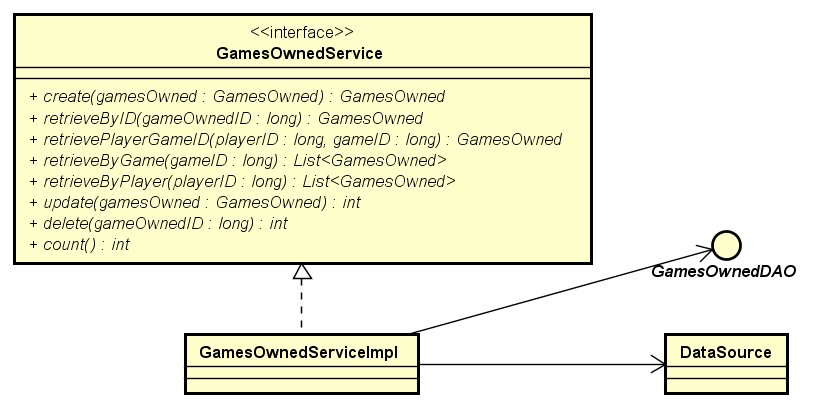
## Game Service Module

The GameService provides the standard CRUD methods for Game instances. The service also provides operations that allow for the query for Game by title and release date range.



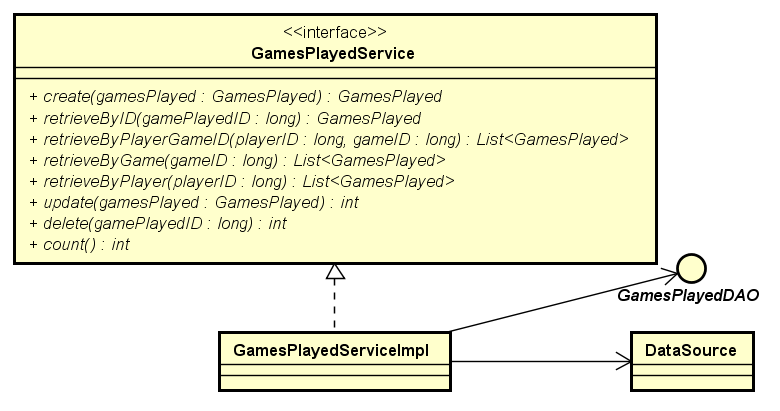
## GamesOwned Service Module

The GamesOwned provides the standard CRUD methods for GamesOwned instances. The services allows for the retrieval of GamesOwned by specific Player using the Player ID and Game by the Game ID.



## GamesPlayed Service Module

The GamesPlayed provides the standard CRUD methods for GamesPlayed instances. The services allows for the retrieval of GamesPlayed by specific Player using the Player ID and Game by the Game ID.

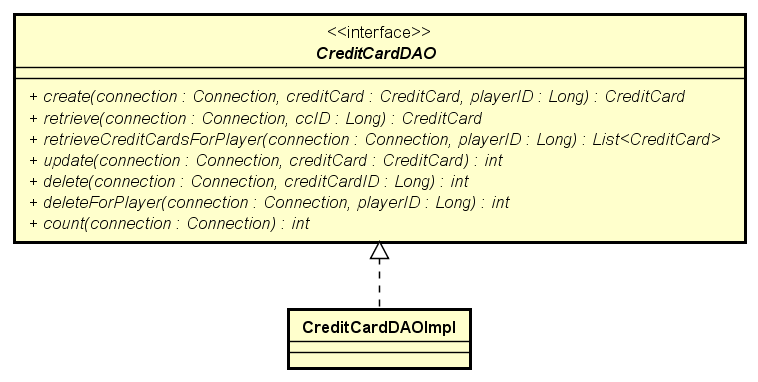


# DAO Interfaces

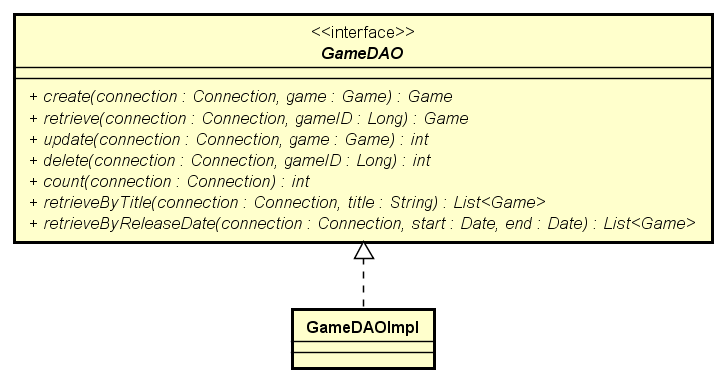
The following illustrates the DAO interfaces that will be implemented and delivered with the project. The delivered classes must implement these interfaces for grading and testing to execute correctly on an individual class basis. These Java interfaces have been provided with the project source code.

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.

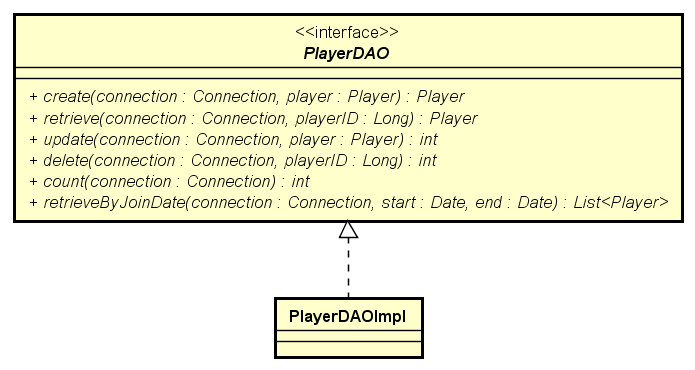
## CreditCardDAO



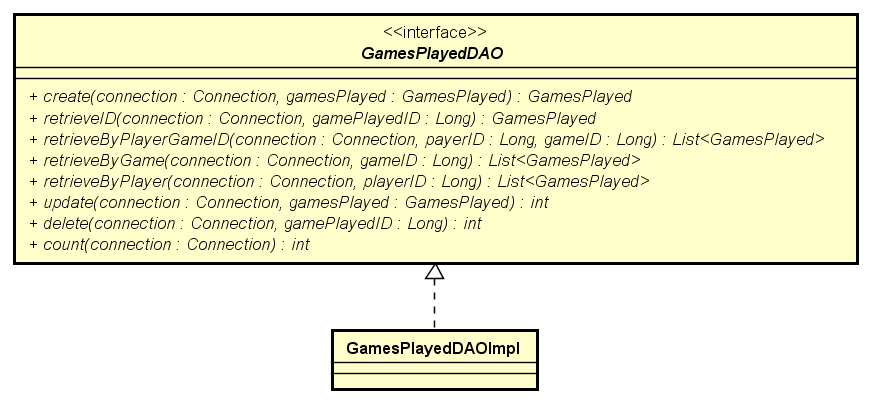
## GameDAO



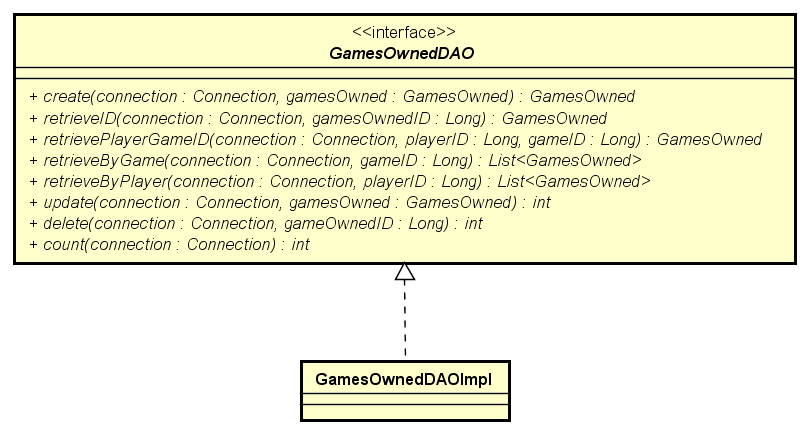
## PlayerDAO



## GamesPlayedDAO



## GamesOwnedDAO



# Provided Materials

## Game JDBC Project Description.docx

This document containing project description and instructions.

## Game JDBC Project Evaluation - Team XX.docx

The team evaluation form to be completed by each team and included in the UTD Box folder when submitting the project deliverables.

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:

## GameJDBCProjectForStudent.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.**

## GameJDBCProjectTesting.zip

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

**Teams should exercise caution when modifying code / contents of this project**. Some of the unit tests in the testing project will require entering active primary key (ID) values to match the auto-generated key values generated by MySQL. However, changing the underlying logic of unit tests 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 GameJDBCProjectTesting project includes:

1. The application PopulateTables.java uses built-in JDBC to populate the database generated by your team’s schema with sample data. This includes three CSV files containing the data used to create Player, Game, and CreditCard instances.
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 GameService, PlayerService, GamesOwnedService and GamesPlayedService 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 (games).

# 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.
2. Be sure that the Testing project’s Eclipse build path includes the Student project. It should be default.
3. Set up a MySQL DBMS on either AWS or your local PC.
4. 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().
5. 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 ‘games’.
6. 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).
7. 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 create\* methods in the DAO Implementation classes in the student development project.
8. Implement the remaining DAO interfaces against your schema. That is, complete the DAO implementation classes with queries that reflect your schema.
9. Use the DAO JUnit tests (see classes in the package srcTest.dao) to validate the ‘correctness’ of your DAO implementations.
10. Implement the PlayerService interface against your DAOs.
11. Implement the GameService interface against your DAOs.
12. Implement the GamesOwnedService interface against your DAOs.
13. Implement the GamesPlayedService interface against your DAOs.
14. Use the provided service unit test (see the package srcTest.service) to validate the correctness of your service implementations.
15. When all tests are working correctly package ONLY the contents of the GameJDBCProjectStudent 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.
16. Test that the library jar file works correctly by including the library jar in the GameJDBCProjectTesting project’s build path. NOTE: You will need to remove the project GameJDBCProjectStudent 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

A testing utility class “TestingUtil.java” will be found in the Student Development project. This utility class is used to automate the identification of active primary key values (IDs) in the player and games tables. In some cases this utility class will be unable to identify actual ID values and the values provided in the Test Cases must be changed to reflect the values found in these tables.

# 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 low bandwidth IOPS (I/O operations 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 ‘GAME but used in a query ‘game’, 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

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 (…/games \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

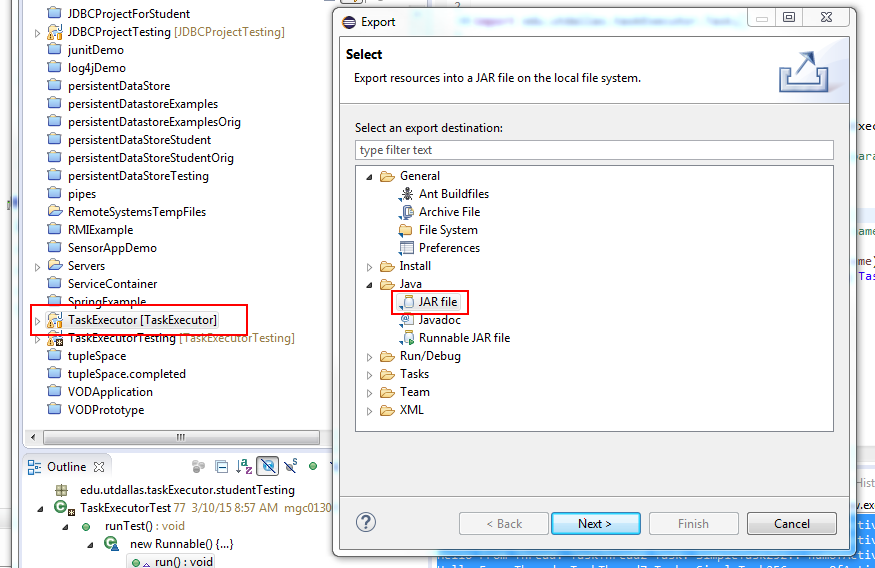
## Executing Unit Tests in Eclipse

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.

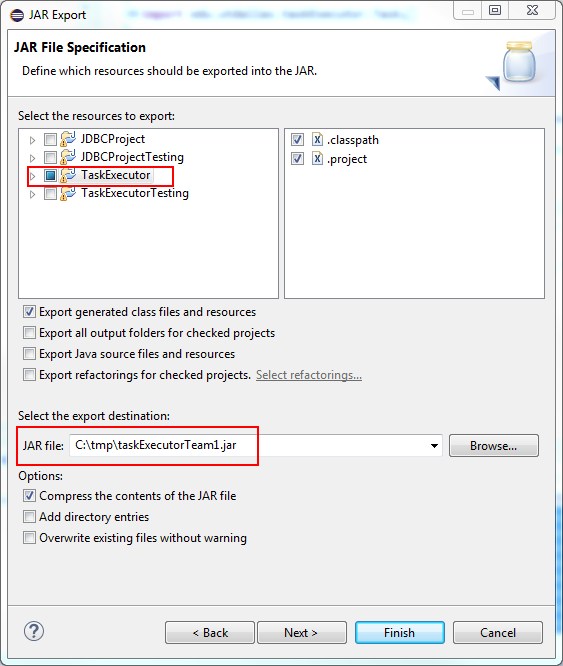
## Exporting an Eclipse Project as a Library JAR File

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.

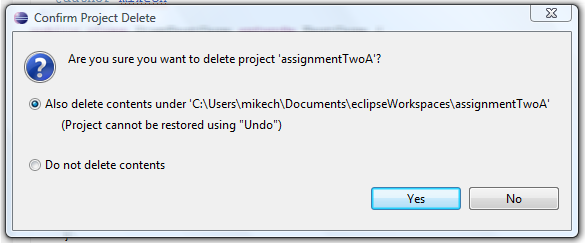


## Importing a Project into Eclipse from a Project Zip File

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

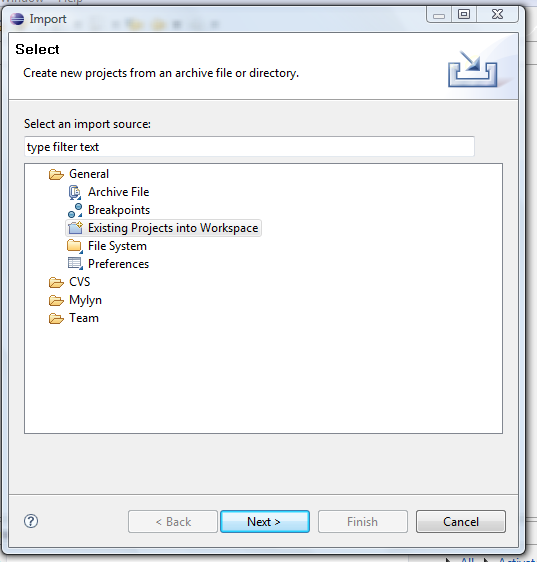


A this point the Old project will be have been removed from your workspace and you may begin importing the project template

## 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.

