Modifications to menu app

In this section, you will be working in `ProjectsApp.java`.

✓1. Add this line to the list of operations at the top of ProjectsApp.java: `"2) List projects"`.

✓2. Add `case 2` to the switch statement in `processUserSelection()`. In the case, call method `listProjects()`. Don't forget to add the `break` statement.

✓3. Have Eclipse create the method `listProjects()`. It should take no parameters and should return nothing. In the method:

   ✓a. Create a variable to hold a `List` of `Project`s named `projects`. Assign the variable the results of a method call to `projectService.fetchAllProjects()`.

   ✓b. Print `"\nProjects:"` (without quotes) to the console.

   ✓c. For each `Project`, print the ID and name separated by `": "`. Indent each line

   ✓d. Have Eclipse create the method `fetchAllProjects()` in the `ProjectService` class, or create it yourself.

   ✓e. Save all files. At this point the project should have no errors.

Modifications to project service

In this section, you will be working in `ProjectService.java`.

✓1. In method `fetchAllProjects`, call the `fetchAllProjects()` method on the `projectDao` object.

✓2. Have Eclipse create the method `fetchAllProjects()` in `ProjectDao.java` or create it yourself. It takes no parameters and returns a `List` of `Project`s.

Modifications to project dao

In this section, you will be working in `ProjectDao.java`.

1. In the method `fetchAllProjects()`:

   ✓a. Write the SQL statement to return all projects not including materials, steps, or categories. Order the results by project name.

   ✓b. Add a `try-with-resource` statement to obtain the `Connection` object. Catch the `SQLException` in a `catch` block and rethrow a new `DbException`, passing in the `SQLException` object.

   ✓c. Inside the `try` block, start a new transaction.

   ✓d. Add an inner `try-with-resource` statement to obtain the `PreparedStatement` from the `Connection` object. In a `catch` block, catch an `Exception` object. Rollback the transaction and throw a new `DbException`, passing in the `Exception` object as the cause.

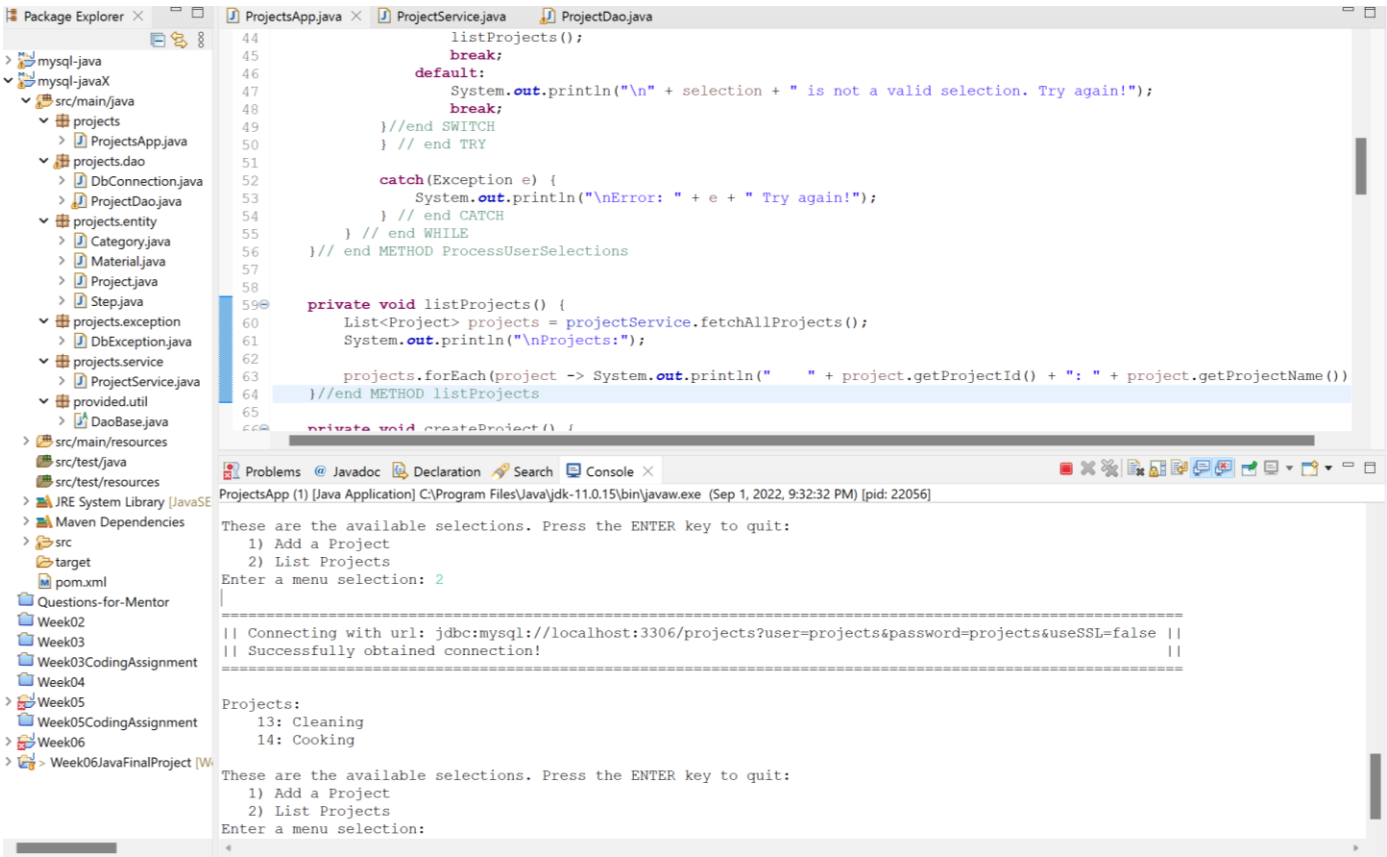
✓e. Inside the (currently) innermost `try-with-resource` statement, add a `try-with-resource` statement to obtain a `ResultSet` from the `PreparedStatement`. Include the import statement for `ResultSet`. It is in the `java.sql` package.

✓f. Inside the new innermost `try-with-resource`, create and return a `List` of `Projects`.

✓g. Loop through the result set. Create and assign each result row to a new `Project` object. Add the `Project` object to the `List` of `Projects`. You can do this by calling the extract method.

Test it

✓Test your solution by running `ProjectsApp`. Select "`List projects`". The app should return a list of projects that you have created. Make sure that you have created at least one project. Take a screen shot showing the console with the menu selections, your input, and the listed project(s). ✓



Modifications to menu app

In this section you will be working in `ProjectsApp.java`.

✓1. Add an instance variable of type `Project` named `curProject`.

✓2. Add a new operation: "`3) Select a project`".

✓3. Add a case to the `switch` to handle the operation. Call method `selectProject()`.

✓4. Add a new method named `selectProject()`. It takes no parameters and returns nothing.

   ✓a. Call `listProjects()` to print a `List` of `Project`s.

   ✓b. Collect a project ID from the user and assign it to an Integer variable named `projectId`. Prompt the user with "`Enter a project ID to select a project`".

   ✓c. Set the instance variable `curProject` to `null` to unselect any currently selected project. This is done in case the call to the service results in an exception being thrown. Rather than leave the current project selected in that case, it is unselected first.

   ✓d. Call a new method, `fetchProjectById()` on the `projectService` object. The method should take a single parameter, the project ID input by the user. It should return a `Project` object. Assign the returned `Project` object to the instance variable `curProject`. Note that if an invalid project ID is entered, `projectService.fetchProjectById()` will throw a `NoSuchElementException`, which is handled by the `catch` block in `processUserSelections()`.

   ✓e. At the end of the method, add a check to see if `curProject` is null. If so, print "`Invalid project ID selected.`" on the console.

   f. The method should look like this:

```
private void selectProject() {
  listProjects();
  Integer projectId = getIntInput("Enter a project ID to select a project");

  /* Unselect the current project. */
  curProject = null;

  /* This will throw an exception if an invalid project ID is entered. */
  curProject = projectService.fetchProjectById(projectId);
}
```

✓5. In this step, you will add code to print the current project when the available menu selections are displayed to the user. To do this, find the method `printOperations()`. At the bottom of method `printOperations()`, check if `curProject` is `null`. If `null`, print a message: "`\nYou are not working with a project.`". Otherwise, print the message: "`\nYou are working with project: " + curProject`.

Modifications to project service

In this section you will be working in `ProjectService.java`.

✓1. Create method `fetchProjectById()`. It returns a `Project` object and takes an `Integer projectId` as a parameter. Inside the method:

   ✓a. Temporarily assign a variable of type `Optional<Project>` to the results of calling `projectDao.fetchProjectById()`. Pass the project ID to the method.

   ```
   Optional<Project> op = projectDao.fetchProjectById(projectId);
   ```

   This temporary assignment will cause Eclipse to create the correct return value (`Optional<Project>`) in `ProjectService.java`.

   ✓b. Let Eclipse create the method for you in the `ProjectDao` class. The editor will display `ProjectDao.java`. Return to `ProjectService.java`. Save all files.

   ✓c. Replace the variable and assignment with a `return` statement. This will cause a compilation error, which you will correct next.

   ✓d. Add a method call to `.orElseThrow()` just inside the semicolon at the end of the method call to `projectDao.fetchProjectById()`. Use a zero-argument Lambda expression inside the call to `.orElseThrow()` to create and return a new `NoSuchElementException` with the message, `"Project with project ID=" + projectId + " does not exist."`.

Modifications to project dao

In this section you will be working in `ProjectDao.java`.

1. In the method `fetchProjectById()`:

   ✓a. Write the SQL statement to return all columns from the project table in the row that matches the given `projectId`. Make sure to use the parameter placeholder "`?`" in the SQL statement.

   ✓b. Obtain a `Connection` object in a `try-with-resource` statement. Add the `catch` block to handle the `SQLException`. In the `catch` block throw a new `DbException` passing the `SQLException` object as a parameter.

   ✓c. Start a transaction inside the `try-with-resource` statement.

   ✓d. Below the method call to `startTransaction()`, add an inner `try/catch`. The `catch` block should handle `Exception`. Inside the `catch` block, rollback the transaction and throw a new `DbException` that takes the `Exception` object as a parameter.

   ✓e. Inside the `try` block, create a variable of type `Project` and set it to `null`. Return the `Project` object as an `Optional` object using `Optional.ofNullable()`. Save the file. You should have no compilation errors at this point but you may see some warnings.

   ✓f. Inside the inner `try` block, obtain a `PreparedStatement` from the `Connection` object in a `try-with-resource` statement. Pass the SQL statement in the method call to `prepareStatement()`. Add the `projectId` method parameter as a parameter to the `PreparedStatement`.

g. Obtain a `ResultSet` in a `try-with-resource` statement. If the `ResultSet` has a row in it (`rs.next()`) set the `Project` variable to a new `Project` object and set all fields from values in the `ResultSet`. You can call the `extract()` method for this.

h. Below the `try-with-resource` statement that obtains the `PreparedStatement` but inside the try block that manages the rollback, add three method calls to obtain the list of materials, steps, and categories. Since each method returns a `List` of the appropriate type, you can call `addAll()` to add the entire `List` to the `List` in the Project object:

```
project.getMaterials().addAll(fetchMaterialsForProject(conn, projectId));
```

i. Commit the transaction.

2. Follow these instructions to write the three methods to return materials, steps, and categories. Each method should return a `List` of the appropriate type. At this point there should be no compilation errors.

   a. Each method should take the `Connection` and the project ID as parameters.

   b. Each method should return a `List` of the appropriate type (i.e., `List<Material>`).

   c. Each method is written in the same way as the other query methods with the exception that the `Connection` is passed as a parameter, so you don't need to call `DbConnection.getConnection()` to obtain it.

   d. Each method can add throws `SQLException` to the method declaration. This is because the method call to each method is within a `try/catch` block.

Instructions for DBeaver

1. Create a connection to the projects schema in DBeaver if you haven't already.
2. Right-click on the connection name and select "`SQL Editor`" / "`Recent SQL Script`".

Instructions for MySQL CLI

1. Start up MySQL CLI. Enter the root password.
2. Type "`use projects;`" (without the quotes).

The test

1. Add one or more categories. You don't have to enter the category ID, MySQL will manage that for you.

```
INSERT INTO category (category_name) VALUES ('Doors and Windows');
```

2. Make sure you have added one or more projects. In the editor type this to find a valid `project_id`:

```
SELECT * FROM project;
```

3. Add one or more `material` records. If your `project_id` is `1`, enter something like this:

```
INSERT INTO material (project_id, material_name, num_required)VALUES
(1, '2-inch screws', 20);
```

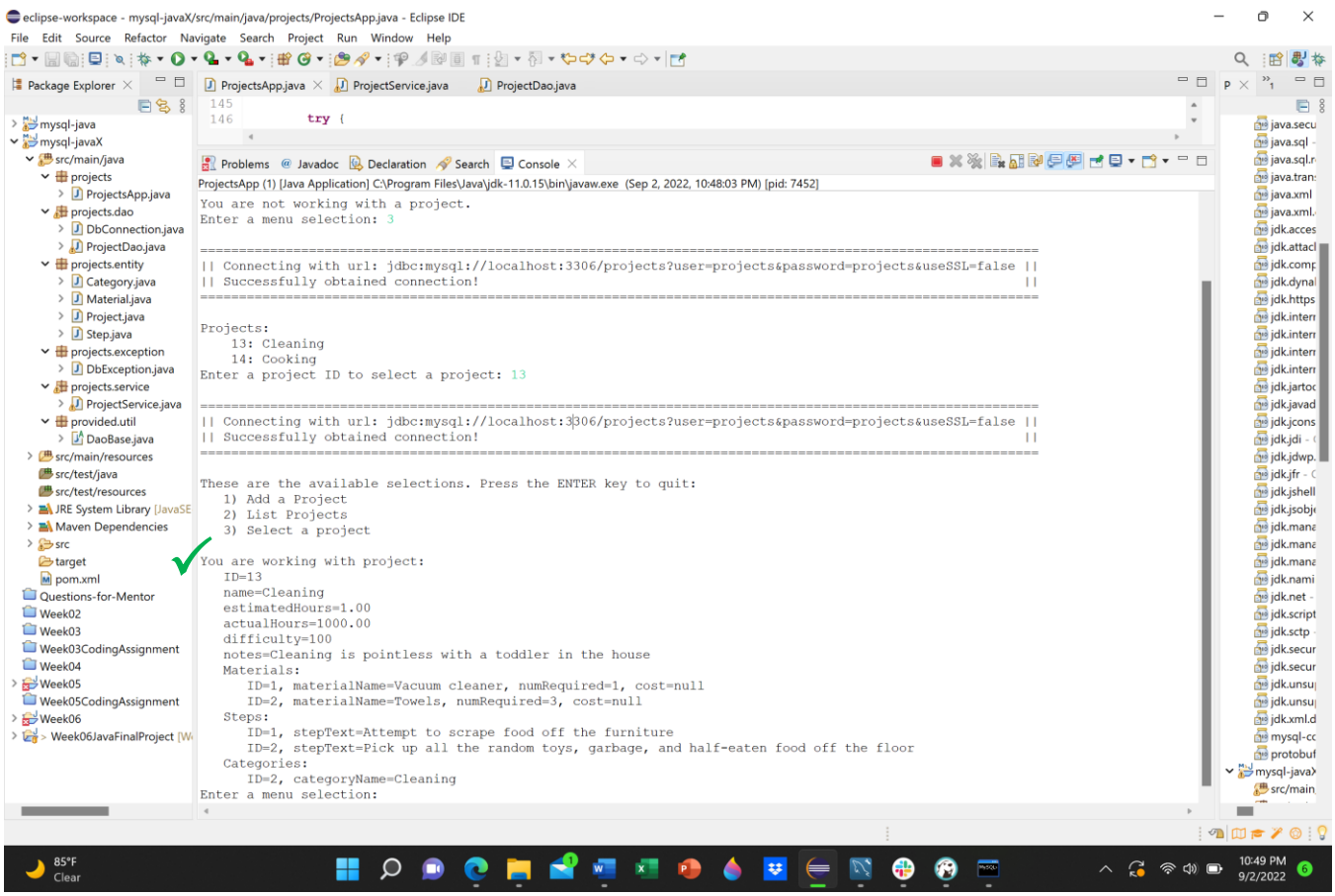
4. Add one or more `step` records. If your `project_id` is `1`, enter something like the following:

```
INSERT INTO step (project_id, step_text, step_order) VALUES
(1, 'Screw door hangers on the top and bottom of each side of the door
frame', 1);
```

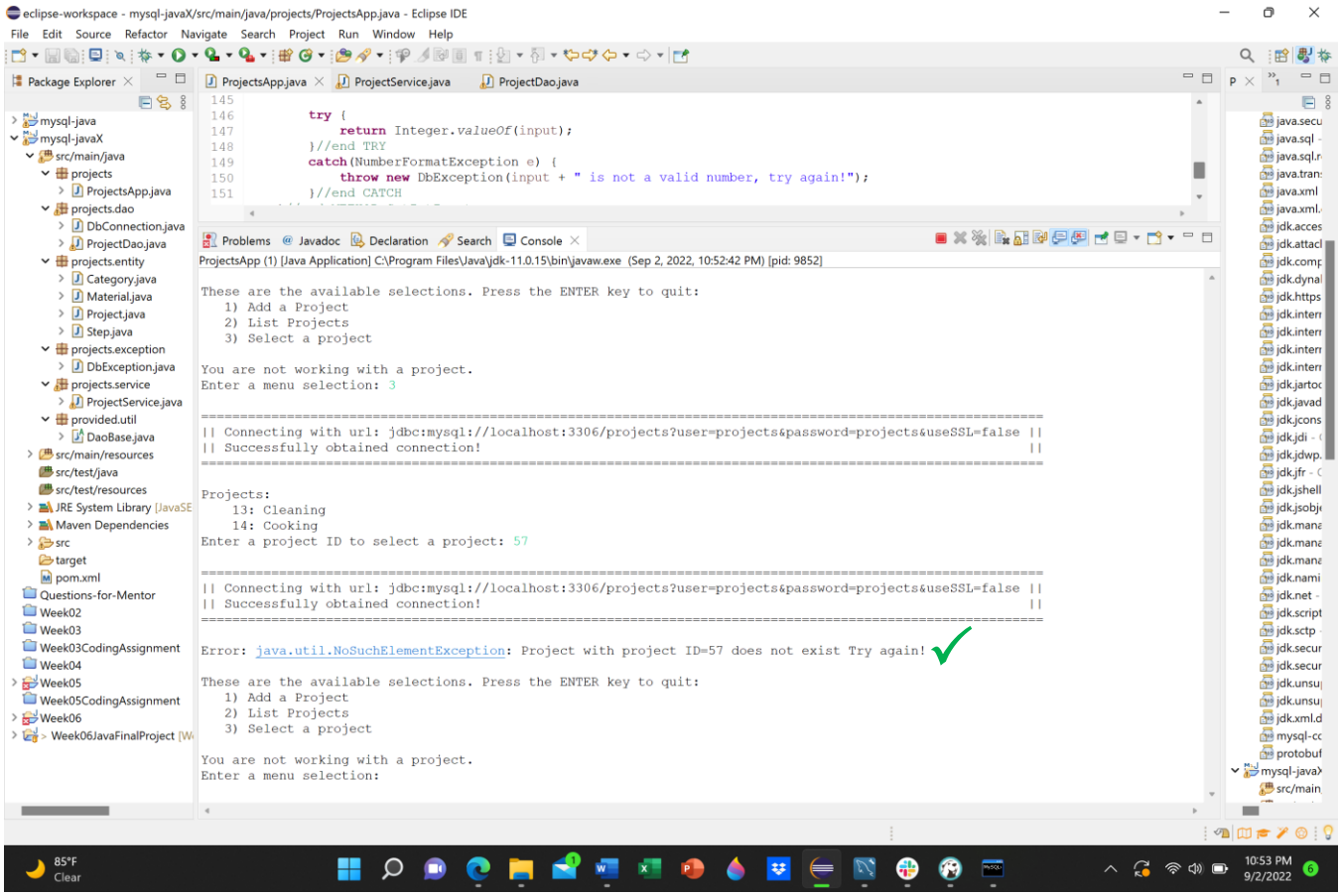
5. Add one or more `project_category` records. This is a join table that contains two foreign keys. One foreign key points to a project row and the other points to a category row. So, if your project ID is `1` and the category ID for 'Doors and Windows' is `2`, enter the join row like this:

```
INSERT INTO project_category (project_id, category_id) VALUES(1, 2);
```

6. Run `ProjectsApp` as a Java application. Enter "`3`" to select a project. Enter a project ID. Take a screen shot showing that the project is selected.



7. Now test with an invalid project ID. Run the application. Enter "3" to select a project. Enter an invalid number. Take a screen shot of the console.



GitHub Link

https://github.com/JaxYoungblood/MySQLCodingProject.git