Jacqueline Youngblood Week 09 – MySQL Coding Assignment August 26, 2022

Cleanup

In this section, you are working with `ProjectsApp.java` in the `projects` package.

✓1. Delete the debugging line (`DbConnection.getConnection();`) in the main method. The method should now be empty.

✓2. Remove the import statement: `import projects.dao.DbConnection;`

Build the Menu Application

In this section, you are working with `ProjectsApp.java` in the `projects` package.

✓1. In order to display a list of menu options you must store them somewhere. In this step you will write the code that holds the list of operations.

  ✓a. Add a private instance variable named "`operations`". The type is `List<String>`. Initialize it using `List.of` with the following value: "`1) Add a project`".

✓2. Add a `private` instance variable named `scanner`. It is of type `java.util.Scanner`. Initialize it to a `new Scanner` object. Pass `System.in` to the constructor. This will set the scanner so that it accepts user input from the Java console.

✓3. In the `main()` method, create a new `ProjectsApp` object and call the method: `processUserSelections()` method. The method takes zero parameters and returns nothing.

✓4. Now you can create the `processUserSelections()` method as an instance method. Let Eclipse create the method for you by waving your mouse over the compiler error in the `main()` method (over the red squigglies). Eclipse will pop up a menu. Select "`Create method processUserSelections()`". In method `processUserSelections()`:

  ✓a. Add a local variable: `boolean done = false;`

  ✓b. Add a `while` loop below the local variable. Loop until the variable `done` is `true`.

  ✓c. Inside the `while` loop, add a `try/catch` block. The `catch` block should catch `Exception`. Inside the `catch` block print the `Exception` message. Call the `toString()` method on the `Exception` object provided to the `catch` block. This is done by simply concatenating the `Exception` object onto a `String` literal. When you do this Java implicitly calls the `toString()` method behind the scenes.

  ✓d. Inside the `try` block, assign an `int` variable named `selection` to the return value from the method `getUserSelection()`. The method should now look like this:

✓5. Create the method `getUserSelection()`. It takes no parameters and returns an `int`. This method will print the `operations` and then accept user input as an `Integer`. In the `getUserSelection()` method:

  ✓a. Make a method call to `printOperations()`. This method takes no parameters & no returns.

  ✓b. Add a method call to `getIntInput()`. Assign the results of the method call to a variable named `input` of type `Integer`. The method `getIntInput()`, which you haven't written yet. It will return the user's menu selection. The value may be `null`. Pass the `String` literal "`Enter a menu selection`" as a parameter to the method.

✓c. Add a return statement that checks to see if the value in local variable `input` is `null`. If so, return `-1`. (The value `-1` will signal the menu processing method to exit the application.) Otherwise, return the value of `input`.

✓6. Create the method `printOperations()`. It takes no parameters and returns nothing. In the method:

✓a. Print a line to the console: `"\nThese are the available selections. Press the Enter key to quit:");`

✓b. Print all the available menu selections, one on each line. Each line should be indented slightly (2 or 3 spaces). Use any strategy that you choose to print the instructions.

✓7. Create the method `getIntInput`. It takes a single parameter of type `String` named `prompt`. This method accepts input from the user and converts it to an `Integer`, which may be `null`. It is called by `getUserSelection()` and will be called by other data collection methods that require an `Integer`. Inside the method body:

✓a. Assign a local variable named `input` of type `String` to the results of the method call `getStringInput(prompt)`.

✓b. Test the value in the variable `input`. If it is `null`, return `null`. Use `Objects.isNull()` for the `null` check.

✓c. Create a `try/catch` block to test that the value returned by `getStringInput()` can be converted to an `Integer`. The `catch` block should accept a parameter of type `NumberFormatException`.

✓i. In the `try` block, convert the value of input, which is a `String`, to an `Integer` and return it. If the conversion is not possible, a `NumberFormatException` is thrown. The message in `NumberFormatException` is obscure & will get fixed in the `catch` block.

✓ii. In the `catch` block throw a new `DbException` with the message, `input + " is not a valid number. Try again."`

✓8. Create the method `getStringInput()`. It should have a single parameter of type `String` named `prompt`. It should return a `String`. Inside the method:

✓a. Print the prompt using `System.out.print(prompt + ": ")`

✓b. Assign a `String` variable named `input` to the results of a method call to `scanner.nextLine()`.

✓c. Test the value of `input`. If it is blank return `null`. Otherwise return the trimmed value.

✓d. At this point the file should have no compile errors.

✓9. Since the user enters an `Integer` value (the menu selection number) you can use a `switch` statement to process the selection. Back in the method `processUserSelections()`:

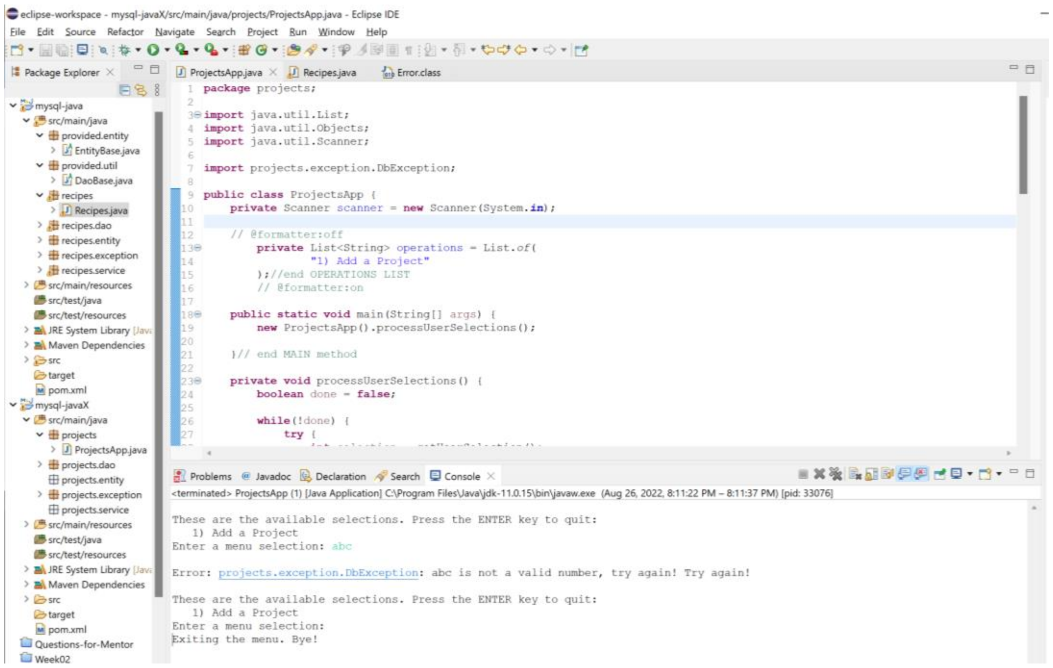
✓a. Add a `switch` statement below the method call to `getUserSelection()`. Create a `switch` statement to switch on the value in the local variable `selection`.

✓b. Add the first case of `-1`. Inside this case, call `exitMenu()` and assign the result of the method call to the local variable `done`. Make sure to add the `break` statement.

✓c. Add the default case. Print a message: `"\n" + selection + " is not a valid selection. Try again."`.

10. Now that the menu code has been written you will need to test it to see if it works. Test the application two ways:

a. Run the application. Enter "`abc`" (without quotes) and press `Enter`. Now press `Enter` with no input. Take a screen shot to show the application output .


eclipse-workspace - mysql-javaX/src/main/java/projects/ProjectsApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
ProjectsApp.java
Recipes.java
Error.class
Problems Javadoc Declaration Search Console

```java
package projects;

import java.util.List;
import java.util.Objects;
import java.util.Scanner;

import projects.exception.DbException;

public class ProjectsApp {
    private Scanner scanner = new Scanner(System.in);

    // @formatter:off
        private List<String> operations = List.of(
            "1) Add a Project"
        );//end OPERATIONS LIST
        // @formatter:on

    public static void main(String[] args) {
        new ProjectsApp().processUserSelections();

    }// end MAIN method

    private void processUserSelections() {
        boolean done = false;

        while(!done) {
            try {
```

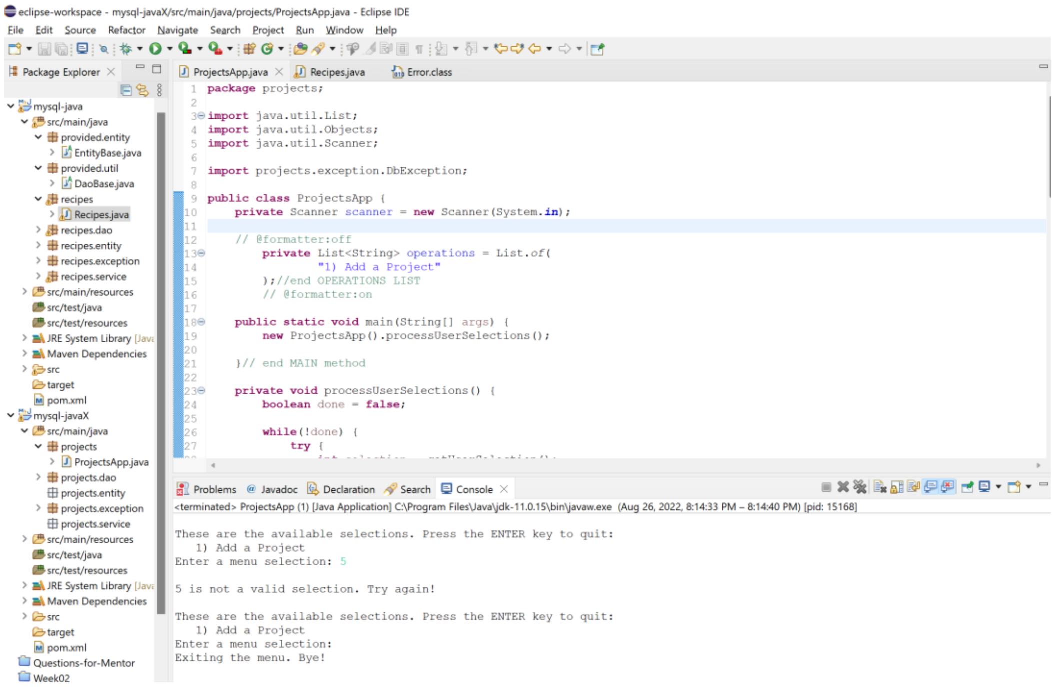
```
<terminated> ProjectsApp (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe (Aug 26, 2022, 8:11:22 PM – 8:11:37 PM) [pid: 33076]

These are the available selections. Press the ENTER key to quit:
   1) Add a Project
Enter a menu selection: abc

Error: projects.exception.DbException: abc is not a valid number, try again! Try again!

These are the available selections. Press the ENTER key to quit:
   1) Add a Project
Enter a menu selection:
Exiting the menu. Bye!
```

b. Run the application. Enter "`5`" (without quotes) and press `Enter`. Now press `Enter` with no input. Take a screen shot to show the application output .


eclipse-workspace - mysql-javaX/src/main/java/projects/ProjectsApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
ProjectsApp.java
Recipes.java
Error.class
Problems Javadoc Declaration Search Console

```java
package projects;

import java.util.List;
import java.util.Objects;
import java.util.Scanner;

import projects.exception.DbException;

public class ProjectsApp {
    private Scanner scanner = new Scanner(System.in);

    // @formatter:off
        private List<String> operations = List.of(
            "1) Add a Project"
        );//end OPERATIONS LIST
        // @formatter:on

    public static void main(String[] args) {
        new ProjectsApp().processUserSelections();

    }// end MAIN method

    private void processUserSelections() {
        boolean done = false;

        while(!done) {
            try {
```

```
<terminated> ProjectsApp (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe (Aug 26, 2022, 8:14:33 PM – 8:14:40 PM) [pid: 15168]

These are the available selections. Press the ENTER key to quit:
   1) Add a Project
Enter a menu selection: 5

5 is not a valid selection. Try again!

These are the available selections. Press the ENTER key to quit:
   1) Add a Project
Enter a menu selection:
Exiting the menu. Bye!
```

Add project files from student resources

In this section, you will add files into the `mysql-java` project from the student resources.

✓1. Drag the four files from the student resources `/Homework/entity` folder and drop them onto the `projects.entity` package in the Eclipse project. When done you should see `Category.java`, `Material.java`, `Project.java`, and `Step.java` in the `projects.entity` package.

✓2. Drag the directory named "`provided`" from the student resources `/Homework` folder and drop it onto `src/main/java` in the package explorer. When done, there should be a new package named `provided.util` with a single file in it named `DaoBase.java`.

Add a new project to the project table

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

✓1. At the top of the class, add a private instance variable of type `ProjectService` named `projectService` and call the zero-argument constructor to initialize it. Let Eclipse create the `ProjectService` class. Make sure the class is created in the `projects.service` package

✓2. In the method `processUserSelections()`, add `case 1` to the switch statement. Inside the case, call the method `createProject()`. This method takes no parameters and returns nothing. Remember to add the `break` statement.

✓3. Create the method `createProject()`. It is `private`, no parameters, & no returns. In this method:

   ✓a. Add local variable `String projectName`. Assign the value to the result of calling `getStringInput("Enter the project name")`.

   ✓b. Add local variable `BigDecimal estimatedHours`. Assign the value to the result of calling `getDecimalInput("Enter the estimated hours")`. You may need to add the import statement for `BigDecimal`. It is in the `java.math` package.

   ✓c. Add local variable `BigDecimal actualHours`. Assign the value to the result of calling `getDecimalInput("Enter the actual hours")`.

   ✓d. Add local variable `Integer difficulty`. Assign the value to the result of calling `getIntInput("Enter the project difficulty (1-5)")`.

   ✓e. Add local variable `String notes`. Assign the value to the result of calling `getStringInput("Enter the project notes")`.

   ✓f. Create a new variable of type `Project` named `project`. Initialize it to a `new Project` object by calling the zero-argument constructor. Import `Project` class from `projects.entity` package.

   ✓g. Call the appropriate setters on the `Project` object to set `projectName`, `estimatedHours`, `actualHours`, `difficulty` and `notes`.

   ✓h. Call the `addProject()` method on the `projectService` object. Pass it the `Project` object. This method should return an object of type `Project`. Assign it to variable `dbProject`.

   ✓i. Print a success message to the console `"You have successfully created project: " + dbProject`. The value returned from `projectService.addProject()` is different from the `Project` object passed to the method. It contains the project ID that was added by MySQL.

✓4. Create the method `getDecimalInput()`. The easiest way to do this is to create the method body, then copy the method contents from `getIntInput()` and paste it into the method body. Fix the following:

  ✓a. The line in the `try` block. Change it to: `return new BigDecimal(input).setScale(2);`

  ✓b. The message in `DbException`. Change it to: `input + " is not a valid decimal number."`

✓5. Wave the mouse over "`projectService.addProject()`". When the menu pops up, select "`Add method 'addProject(project)' in type 'ProjectService'`".

✓6. Save all files. All compiler errors should now be gone.

Modifications to project service

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

✓1. At the top of the class, add a private instance variable of type `ProjectDao` named `projectDao`. Assign the variable to a new `ProjectDao` object by calling the constructor with no parameters. Create a `ProjectDao` class in the `projects.dao` package. Make sure that `ProjectDao` extends `DaoBase` from the `provided.util` package. Change back to the `ProjectService` class.

✓2. In method `addProject()`, call the method `insertProject()` on the `projectDao` object. The method should take a single parameter. Pass it the `Project` parameter and return the value from the method.

✓3. Wave the mouse over `insertProject()` (with the red squigglies) and select "`Create method 'insertProject(Project)' in type 'ProjectDao'`".

Modifications to project DAO

✓1. Add the constant for the category table named `CATEGORY_TABLE`. Set the value to "`category`".

✓2. Add the constant for the material table named `MATERIAL_TABLE`. Set the value to "`material`".

✓3. Add the constant for the project table named `PROJECT_TABLE`. Set the value to "`project`".

✓4. Add the constant for the project-category table named `PROJECT_CATEGORY_TABLE`. Set the value to "`project_category`".

✓5. Add the constant for the step table named `STEP_TABLE`. Set the value to "`step`".

Save the project details

In this section, you will be working exclusively in the method `insertProject()` in `ProjectDao.java`.

✓1. Write the SQL statement that will insert the values from the `Project` object passed to the `insertProject()` method. Remember to use question marks as placeholder values for the parameters passed to the `PreparedStatement`. Add the fields `project_name`, `estimated_hours`, `actual_hours`, `difficulty`, and `notes`.

✓2. Obtain a connection from `DbConnection.getConnection()`. Assign it a variable of type `Connection` named `conn` in a try-with-resource statement. Catch the `SQLException` in a `catch` block added to the try-with-resource. From within the `catch` block, throw a new `DbException`. The `DbException` constructor should take the `SQLException` object passed into the `catch` block.

✓3. Start a transaction. Inside the `try` block, start a transaction by calling `startTransaction()` and passing in the `Connection` object. `startTransaction()` is a method in the base class, `DaoBase`.

✓4. Obtain a `PreparedStatement` object from the `Connection` object. Inside the `try` block and below `startTransaction()`, add another `try-with-resource` statement to obtain a `PreparedStatement` from the `Connection` object.

✓a. Pass the SQL statement as a parameter to `conn.prepareStatement()`.

✓b. Add a `catch` block to the inner `try` block that catches `Exception`. In the `catch` block, roll back the transaction and throw a `DbException` initialized with `Exception` passed into the `catch`.

✓5. Inside the inner `try` block, set the parameters on the `Statement`. Use the convenience method in DaoBase `setParameter()`. This method handles `null` correctly. Add these parameters: `projectName`, `estimatedHours`, `actualHours`, `difficulty`, and `notes`.

✓6. Save the project details by calling `executeUpdate()` on the `PreparedStatement` object.

✓7. Obtain the project ID (primary key) by calling the convenience method in `DaoBase`, `getLastInsertId()`. Pass the `Connection` object and the constant `PROJECT_TABLE` to `getLastInsertId()`. Assign the return value to an `Integer` variable named `projectId`.

✓8. Commit the transaction by calling the convenience method in `DaoBase`, `commitTransaction()`. Pass the `Connection` object to `commitTransaction()` as a parameter.

✓9. Set the `projectId` on the `Project` object that was passed into `insertProject` and return it.

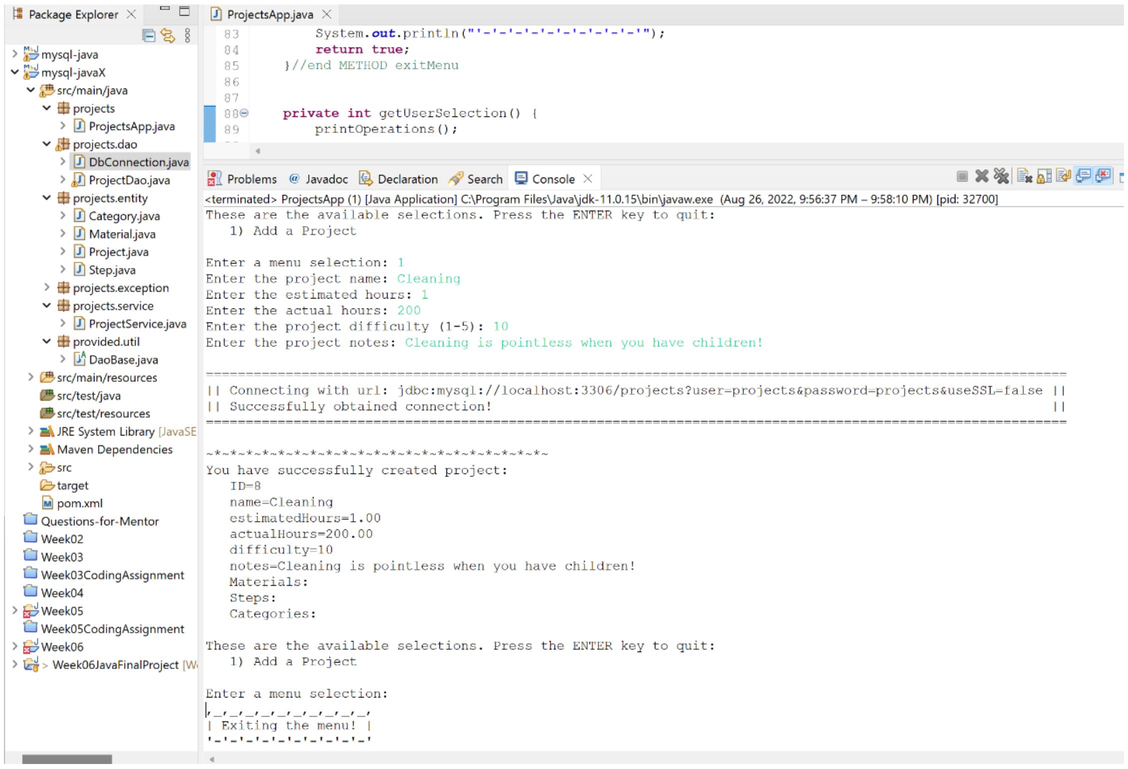
Test it

✓1. Run the application.

✓2. Enter the menu selection "`1`".

✓3. Enter project name, estimated hours, actual hours, difficulty and notes.

✓4. Take a screen shot 📷 of the console output showing the data entry and the printed Project object.



```
            System.out.println("'-'-'-'-'-'-'-'-'-'");
            return true;
        }//end METHOD exitMenu


        private int getUserSelection() {
            printOperations();
```

```
<terminated> ProjectsApp (1) [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe (Aug 26, 2022, 9:56:37 PM – 9:58:10 PM) [pid: 32700]
These are the available selections. Press the ENTER key to quit:
   1) Add a Project

Enter a menu selection: 1
Enter the project name: Cleaning
Enter the estimated hours: 1
Enter the actual hours: 200
Enter the project difficulty (1-5): 10
Enter the project notes: Cleaning is pointless when you have children!

=====================================================================================================
|| Connecting with url: jdbc:mysql://localhost:3306/projects?user=projects&password=projects&useSSL=false ||
|| Successfully obtained connection!                                                                    ||
=====================================================================================================

~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~
You have successfully created project:
   ID=8
   name=Cleaning
   estimatedHours=1.00
   actualHours=200.00
   difficulty=10
   notes=Cleaning is pointless when you have children!
   Materials:
   Steps:
   Categories:

These are the available selections. Press the ENTER key to quit:
   1) Add a Project

Enter a menu selection:
,_,_,_,_,_,_,_,_,_,
| Exiting the menu! |
'-'-'-'-'-'-'-'-'-'
```

GitHub Repo Address

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