

MySQL Week 4 Exercises

<https://github.com/NckDeSimone/Week-10-Coding-Assignment>

Background

We have been developing a menu-driven application that demonstrates how to perform CRUD (Create, Read, Update, Read) operations on a DIY project database. Thus far, we have learned how to create a connection to a MySQL database and how to insert records into a table. In this section, we will apply our knowledge of querying data to list all projects without project details, and to list a single project with all details.


Objectives

In these exercises, you will expand the menu application to list all projects (name and ID). Then, you will write code to select a project to edit. This will involve returning a selected project along with all project details. This will further our pursuit of implementing CRUD operations on database tables.

In these exercises, you will:

- Hone your SQL query skills by writing SQL statements to fetch a `List of Project` records.
- Learn how to perform multiple queries in a single transaction.
- Write an inner join to fetch `category` rows related to a `project` row.
- Use an `Optional` to either return a `project` record or to throw a custom `Exception`.
- Practice writing Lambda expressions both to list the projects and to throw a custom `Exception` from an `Optional`.

Important

In the exercises below, you will see this icon: . This means to take a screen shot or snip showing the results of the action or the code in the editor.

Also important: you should take the variable names and method names as suggestions. They're good suggestions, but if you want to deviate from them, feel free to do so. However, don't go crazy and change `listProjects()` to `emptyMyBankAccountByBuyingAJeep()`. You should follow Java best practices. Method names should describe what the method does in the interest of self-documentation.

Exercises

Complete these exercises as directed. If you get hopelessly stuck, please see the "Solutions" section below.

In these exercises, you will often be told to call a method prior to creating it. This is a good approach. You set up the return type by assigning it to a variable and setting up the parameters. Then, Eclipse can correctly create the method.

List projects

In this section, you will add code to return and print a list of projects. Several application methods will call this method to let the user select a project from a list. To get the list of projects, you will add the menu option and method in the menu class, then you will add a method in the service class. The DAO class will perform the actual work of fetching the list using JDBC method calls.

Modifications to menu app

In this section you will add another option to the list of available options. Then you will add another `case` to the `switch` method along with the method to call the service to retrieve the list of projects.

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

- Add this line to the list of operations at the top of `ProjectsApp.java`: `"2) List projects"`.
- Add `case 2` to the switch statement in `processUserSelection()`. In the case, call method `listProjects()`. Don't forget to add the `break` statement.
- Have Eclipse create the method `listProjects()`. It should take no parameters and should return nothing. In the method:
 - Create a variable to hold a `List of Projects` named `projects`. Assign the variable the results of a method call to `projectService.fetchAllProjects()`.
 - Print `"\nProjects: "` (without quotes) to the console.
 - For each `Project`, print the ID and name separated by `" : "`. Indent each line with a couple of spaces.
 - At this point, the method should look like this:

```
private void listProjects() {
    List<Project> projects = projectService.fetchAllProject

    System.out.println("\nProjects:");

    projects.forEach(project -> System.out
        .println("    " + project.getProjectId()
            + ": " + project.getProjectName()));
}
```

- Have Eclipse create the method `fetchAllProjects()` in the `ProjectService` class, or create it yourself.
- Save all files. At this point the project should have no errors.

Modifications to project service

You need to fill in the method in the service class to call the DAO class. This method will simply return the results of the method call to the DAO class. The service class in our small application does not do very much. But it allows us to properly separate concerns of input/output, business logic, and database reads and writes. If you always structure your code like this it will be much easier to understand and make changes if needed.

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

- In the method `fetchAllProjects`, call the `fetchAllProjects()` method on the `projectDao` object.
- Have Eclipse create the method `fetchAllProjects()` in `ProjectDao.java` or create it yourself. It takes no parameters and returns a `List of Projects`.

Modifications to project dao

Now you need to write the code to retrieve all the projects from the database. It is structured similarly to the `insertProject()` method, but it will also incorporate a `ResultSet` to retrieve the project row(s).

To implement this method, first you will write the SQL statement that instructs MySQL to return all project rows without any materials, steps, or categories. Then, you will obtain a `Connection` and start a transaction. Next, you will obtain a `PreparedStatement` from the `Connection` object. Then, you will get a `ResultSet` from the `PreparedStatement`. Finally, you will iterate over the `ResultSet` to create a `Project` object for each row returned.

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

- In the method `fetchAllProjects()`:
 - Write the SQL statement to return all projects not including materials, steps, or categories. Order the results by project name.
 - 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.

- Inside the `try` block, start a new transaction.
- 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.
- 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.
- Inside the new innermost `try-with-resource`, create and return a `List` of `Projects`.
- 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:


```
while(rs.next()) {  
    projects.add(extract(rs, Project.class));  
}
```

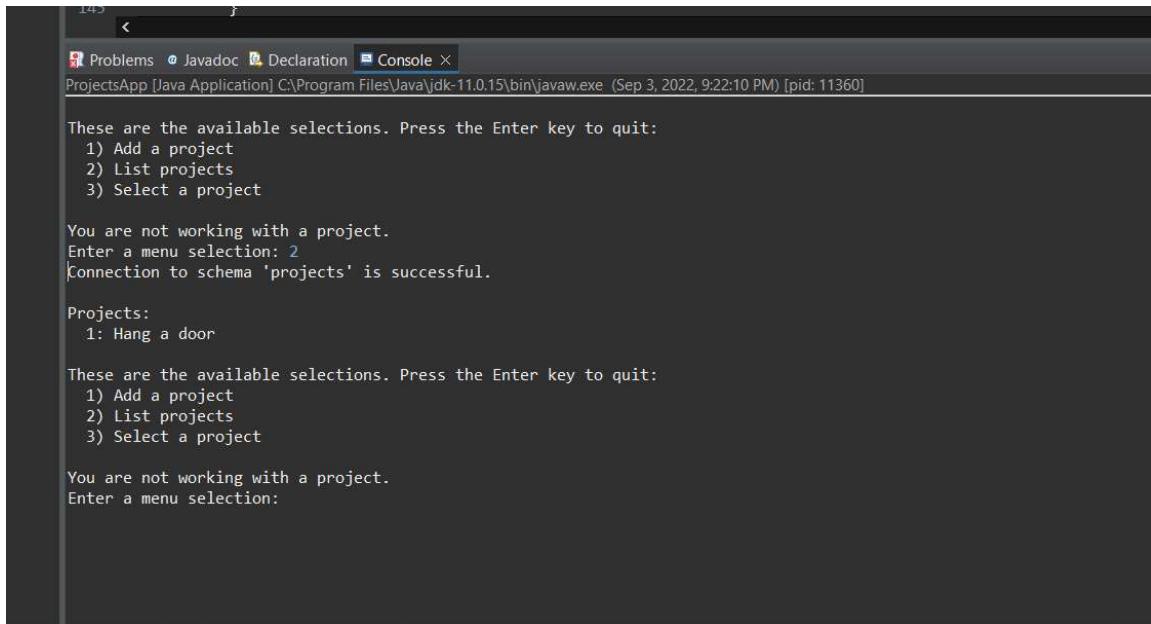
or by doing it manually like this:

```
while(rs.next()) {  
    Project project = new Project();  
  
    project.setActualHours(rs.getBigDecimal("actual_hours");  
    project.setDifficulty(rs.getObject("difficulty", Integer.class));  
    project.setEstimatedHours(rs.getBigDecimal("estimated_hours");  
    project.setNotes(rs.getString("notes"));  
    project.setProjectId(rs.getObject("project_id", Integer.class));  
    project.setProjectName(rs.getString("project_name"));  
  
    projects.add(project);  
}
```

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). 



```
143 }
<
Problems Javadoc Declaration Console x
ProjectsApp [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe (Sep 3, 2022, 9:22:10 PM) [pid: 11360]

These are the available selections. Press the Enter key to quit:
1) Add a project
2) List projects
3) Select a project

You are not working with a project.
Enter a menu selection: 2
Connection to schema 'projects' is successful.

Projects:
1: Hang a door

These are the available selections. Press the Enter key to quit:
1) Add a project
2) List projects
3) Select a project

You are not working with a project.
Enter a menu selection:
```

Select a project

In this section you will write code to select a current project. With a current project selected, you will be able to add materials, steps, and categories in future exercises. This will involve more code than it sounds like on the surface. When you select a current project using the project ID, you will query the project tables to fetch all project details (materials, steps, and categories) within a single transaction.

Modifications to menu app

In this section, you will add a new option to the selections available to the user. Then you will add a method to select the current project. After selecting the current project, you will modify the `printOperations()` method to display the currently selected project, if any. It will print all project details including materials, steps, and categories.

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

- Add an instance variable of type `Project` named `curProject`.
- Add a new operation: "3) Select a project".
- Add a case to the `switch` to handle the operation. Call method `selectProject()`.
- In this step you will create the method, `selectProject()`. This method will list the project IDs and names so that the user can select a project ID. Once the ID is entered, the service is called to return the project details. If successful, the current project is set to the returned project. Follow these instructions to write the method.

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

- Call `listProjects()` to print a List of Projects.
- 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".
- 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.
- 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()`.
- 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.
- The method should look like this:

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

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

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

- 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`.

```
if(Objects.isNull(curProject)) {
    System.out.println("\nYou are not working with a project.");
}
else {
    System.out.println("\nYou are working with project: " + curP
}
```

Modifications to project service

In this section you will create a method in the project service that will call the DAO to retrieve a single Project object with all details, including materials, steps, and categories. This method will throw an exception if the project with the given ID does not exist.

Note that you will temporarily assign the results of a method call to the DAO to an `Optional<Project>` object. This will cause Eclipse to create the return type on the DAO method as `Optional<Project>`. Once the method has been created, you can delete the assignment and return the `Project`, if successful. If not successful, the method will throw a `NoSuchElementException`.

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

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

- Let Eclipse create the method for you in the `ProjectDao` class. The editor will display `ProjectDao.java`. Return to `ProjectService.java`. Save all files.
- Replace the variable and assignment with a `return` statement. This will cause a compilation error, which you will correct next.
- 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 custom message, "Project with project ID=" + `projectId` + " does not exist.". The method should look like this:

```
public Project fetchProjectById(Integer projectId) {  
    return projectDao.fetchProjectById(projectId).orElseTh  
        () -> new NoSuchElementException(  
            "Project with project ID=" + projectId  
            + " does not exist.");  
}
```

Save all files. At this point there should be no compilation errors.

Modifications to project dao

In this section you will write the code that will retrieve a project row and all associated child

rows: materials, steps, and categories. The method will start with the usual `try-with-resource` statement to obtain the `Connection`. Then you will add a `try/catch` before obtaining the `PreparedStatement`. This is done so that after obtaining the project details, the materials, steps, and categories can be retrieved within the same transaction.

If you get stuck and don't understand the instructions, please refer to the "Solutions" section at the end of this assignment.

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

- In the method `fetchProjectById()`:
 - 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.
 - 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.
 - Start a transaction inside the `try-with-resource` statement.
 - 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.
 - 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. This is OK. Here is the method at this point.


```

public Optional<Project> fetchProjectById(Integer projectId) {
    String sql = "SELECT * FROM " + PROJECT_TABLE + " WHERE ID = " + projectId;

    try(Connection conn = DbConnection.getConnection()) {
        startTransaction(conn);

        try {
            Project project = null;

            return Optional.ofNullable(project);
        }
        catch(Exception e) {
            rollbackTransaction(conn);
            throw new DbException(e);
        }
    }
    catch(SQLException e) {
        throw new DbException(e);
    }
}

```

- 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.
- 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.
- 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));

```

- Commit the transaction. Here's what the method should look like now:

```

public Optional<Project> fetchProjectById(Integer projectId) {
    String sql = "SELECT * FROM " + PROJECT_TABLE + " WHERE ID=" + projectId;

    try(Connection conn = DbConnection.getConnection()) {
        startTransaction(conn);

        try {
            Project project = null;

            try(PreparedStatement stmt = conn.prepareStatement(sql)) {
                setParameter(stmt, 1, projectId, Integer.class);

                try(ResultSet rs = stmt.executeQuery()) {
                    if(rs.next()) {
                        project = extract(rs, Project.class);
                    }
                }
            }

            if(Objects.nonNull(project)) {
                project.getMaterials().addAll(fetchMaterialsForProject(project));
                project.getSteps().addAll(fetchStepsForProject(project));
                project.getCategories().addAll(fetchCategoriesForProject(project));
            }

            commitTransaction(conn);

            return Optional.ofNullable(project);
        }
        catch(Exception e) {
            rollbackTransaction(conn);
            throw new DbException(e);
        }
    }
    catch(SQLException e) {
        throw new DbException(e);
    }
}

```

- In this step you will write the methods that will return materials, steps, and categories as `Lists`. Each method is structured similarly. Since the `Connection` object is passed into each method, you won't have to obtain the `Connection` from `DbConnection.getConnection()`.

Also, you won't need to add `catch` blocks to the `try-with-resource` statements because the caller makes the method calls within a `try` block. It won't hurt to catch the

`SQLException` and turn it into an unchecked exception as you have been doing. But it won't hurt to simply declare the exception in the method signature either. It's your choice. So, this:

```
private List<Material> fetchMaterialsForProject(Connection conn,
        Integer projectId) throws SQLException {
```

versus this:

```
try(PreparedStatement stmt = conn.prepareStatement(sql)) {
}
catch(SQLException e) {
    throw new DbException(e);
}
```

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.

- Each method should take the `Connection` and the project ID as parameters.
- Each method should return a `List` of the appropriate type (i.e., `List<Material>`).
- 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.
- 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.
- Here is a sample method (all three methods should have the identical structure). However, when you fetch the categories, you will need to join with the `project_category` join table as shown below.

```

private List<Category> fetchCategoriesForProject(Connection
    Integer projectId) throws SQLException {
    // @formatter:off
    String sql = ""
        + "SELECT c.* FROM " + CATEGORY_TABLE + " c "
        + "JOIN " + PROJECT_CATEGORY_TABLE + " pc USING (c"
        + "WHERE project_id = ?";
    // @formatter:on

    try(PreparedStatement stmt = conn.prepareStatement(sql)
        setParameter(stmt, 1, projectId, Integer.class);

        try(ResultSet rs = stmt.executeQuery()) {
            List<Category> categories = new LinkedList<>();

            while(rs.next()) {
                categories.add(extract(rs, Category.class));
            }

            return categories;
        }
    }
}

```

Test it

Now it's time to test that the code works. Since you haven't written the code to add materials, steps, and categories yet, you will have to add some rows manually.

Instructions for DBeaver

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

Instructions for MySQL CLI

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

The test

- 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');
```

- Make sure you have added one or more projects. In the editor type this to find a valid

project_id:

```
SELECT * FROM project;
```

- 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);
```

- 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);
```

- 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);
```

- 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.  It should look something like this:

```
dao/ProjectDao.java - Eclipse IDE
Project Run Window Help
DbException.java DbConnection.java ProjectsApp.java projects_schema.sql ProjectService.java ProjectDao.java X
132 String sql = "SELECT * FROM " + MATERIAL_TABLE + " WHERE project_id = ?";
133
134 try(PreparedStatement stmt = conn.prepareStatement(sql)){
135     setParameter(stmt, 1, projectId, Integer.class);
136
137     try(ResultSet rs = stmt.executeQuery()){
138         List<Material> materials = new LinkedList<>();
139
140         while(rs.next()) {
141             materials.add(extract(rs, Material.class));
142         }
143
144         return materials;
145     }
146 }
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
275
```


These are the available selections. Press the Enter key to quit:

- 1) Add a project
- 2) List projects
- 3) Select a project

You are not working with a project.

Enter a menu selection: 3

Connection to schema 'projects' is successful.

Projects:

- 1: Hang a door

Enter a project ID to select a project: 1

Connection to schema 'projects' is successful.

These are the available selections. Press the Enter key to quit:

- 1) Add a project
- 2) List projects
- 3) Select a project

You are working with project:

ID=1

name=Hang a door

estimatedHours=4.00

actualHours=3.00

difficulty=3

notes=Use the door hangers from Home Depot

Materials:

ID=1, materialName=Door in frame, numRequired=1, cost=null

ID=2, materialName=Package of door hangers from Home Depot, numR

ID=3, materialName=2-inch screws, numRequired=20, cost=null

Steps:

ID=1, stepText=Align hangers on opening side of door vertically

ID=2, stepText=Screw hangers into frame

Categories:

ID=1, categoryName=Doors and Windows

ID=2, categoryName=Repairs

Enter a menu selection:

You should see project details, a list of materials, a list of steps, and a list of categories.

If you do not get a result like that shown above, check the console for errors. If you can't figure it out there are two things you can try:

- In the catch block in method `processUserSelection()`, print the entire stack trace of the exception like this:

```
catch(Exception e) {
```

```
    System.out.println("\nError: " + e + " Try again.");
```


```
    e.printStackTrace();
```

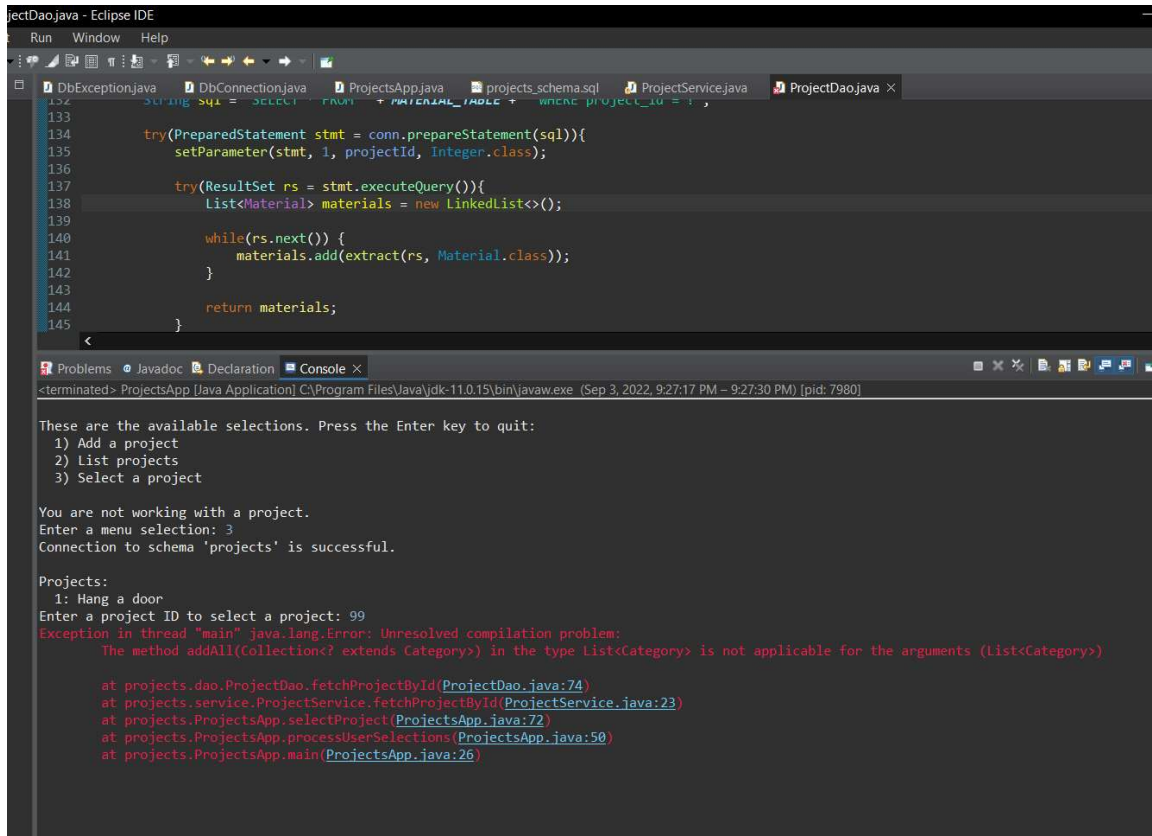
}

- Start the application in debug mode. Load `ProjectsApp.java` into the editor. Right-click in editor and select "Debug As" / "Java Application". Set breakpoints as appropriate. Work through the application until you find the error.
- This article on debugging is a little dated but still applicable. You don't need to worry about the section on remote debugging.

https://www.eclipse.org/community/eclipse_newsletter/2017/june/article1.php

- 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.  It should look something like the screen shot below.



```
jectDao.java - Eclipse IDE
Run Window Help
DbException.java DbConnection.java ProjectsApp.java projects_schema.sql ProjectService.java ProjectDao.java
132
133
134 try(PreparedStatement stmt = conn.prepareStatement(sql)){
135     setParameter(stmt, 1, projectId, Integer.class);
136
137     try(ResultSet rs = stmt.executeQuery()){
138         List<Material> materials = new LinkedList<>();
139
140         while(rs.next()) {
141             materials.add(extract(rs, Material.class));
142         }
143
144         return materials;
145     }
}

Problems Javadoc Declaration Console X
<terminated> ProjectsApp [Java Application] C:\Program Files\Java\jdk-11.0.15\bin\javaw.exe (Sep 3, 2022, 9:27:17 PM - 9:27:30 PM) [pid: 7980]

These are the available selections. Press the Enter key to quit:
1) Add a project
2) List projects
3) Select a project

You are not working with a project.
Enter a menu selection: 3
Connection to schema 'projects' is successful.

Projects:
1: Hang a door
Enter a project ID to select a project: 99
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The method addAll(Collection<? extends Category>) in the type List<Category> is not applicable for the arguments (List<Category>)

at projects.dao.ProjectDao.fetchProjectById(ProjectDao.java:74)
at projects.service.ProjectService.fetchProjectById(ProjectService.java:23)
at projects.ProjectsApp.selectProject(ProjectsApp.java:72)
at projects.ProjectsApp.processUserSelections(ProjectsApp.java:50)
at projects.ProjectsApp.main(ProjectsApp.java:26)
```


These are the available selections. Press the Enter key to quit:

- 1) Add a project
- 2) List projects
- 3) Select a project

You are not working with a project.

Enter a menu selection: 3

Connection to schema 'projects' is successful.

Projects:

- 1: Hang a door

Enter a project ID to select a project: 99

Connection to schema 'projects' is successful.

Error: [projects.exception.DbException](#): Project with project ID=99

These are the available selections. Press the Enter key to quit:

- 1) Add a project
- 2) List projects
- 3) Select a project

You are not working with a project.

Enter a menu selection:

Exiting the menu.

Solutions

These solutions are provided as a reference. Please work through the exercises on your own as best you can.

ProjectsApp.java

These screen shots do not contain the entire Java file. Only the parts changed since the prior exercises are shown.

```
public class ProjectsApp {
    private Scanner scanner = new Scanner(System.in);
    private ProjectService projectService = new ProjectService();
    private Project curProject;

    // @formatter:off
    private List<String> operations = List.of(
        "1) Add a project",
        "2) List projects",
        "3) Select a project"
    );
    // @formatter:on
}
```

```

private void processUserSelections() {
    boolean done = false;

    while(!done) {
        try {
            int selection = getUserSelection();

            switch(selection) {
                case -1:
                    done = exitMenu();
                    break;

                case 1:
                    createProject();
                    break;

                case 2:
                    listProjects();
                    break;

                case 3:
                    selectProject();
                    break;

                default:
                    System.out.println("\n" + selection + " is not a valid sele
                    break;
            }
        }
        catch(Exception e) {
            System.out.println("\nError: " + e + " Try again.");
        }
    }
}

```

```

/**
 *
 */
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);
}

/**
 *
 */
private void listProjects() {
    List<Project> projects = projectService.fetchAllProjects();

    System.out.println("\nProjects:");

    projects.forEach(project -> System.out
        .println("    " + project.getProjectId() + ": " + project.getProjectName()));
}

/**
 * Print the menu selections, one per line.
 */
private void printOperations() {
    System.out.println("\nThese are the available selections. Press the E

    /* With Lambda expression */
    operations.forEach(line -> System.out.println("    " + line));

    /* With enhanced for loop */
    // for(String line : operations) {
    // System.out.println("    " + line);
    // }

    if(Objects.isNull(curProject)) {
        System.out.println("\nYou are not working with a project.");
    }
    else {
        System.out.println("\nYou are working with project: " + curProject);
    }
}

```

ProjectService.java

These screen shots do not contain the entire Java file. Only the parts changed since the prior exercises are shown.

```
/**
 * This method calls the project DAO to retrieve all project rows without accompany:
 * (materials, steps and categories).
 *
 * @return A list of project records.
 */
public List<Project> fetchAllProjects() {
    return projectDao.fetchAllProjects();
}

/**
 * This method calls the project DAO to get all project details, including material:
 * categories. If the project ID is invalid, it throws an exception.
 *
 * @param projectId The project ID.
 * @return A Project object if successful.
 * @throws NoSuchElementException Thrown if the project with the given ID does not exist.
 */
public Project fetchProjectById(Integer projectId) {
    return projectDao.fetchProjectById(projectId).orElseThrow(() -> new NoSuchElementException(
        "Project with project ID=" + projectId + " does not exist."));
}
```

ProjectDao.java

These screen shots do not contain the entire Java file. Only the parts changed since the prior exercises are shown.

```
public List<Project> fetchAllProjects() {
    String sql = "SELECT * FROM " + PROJECT_TABLE + " ORDER BY project

    try(Connection conn = DbConnection.getConnection()) {
        startTransaction(conn);

        try(PreparedStatement stmt = conn.prepareStatement(sql)) {
            try(ResultSet rs = stmt.executeQuery()) {
                List<Project> projects = new LinkedList<>();

                while(rs.next()) {
                    projects.add(extract(rs, Project.class));

                    /* Alternative approach */
                    // Project project = new Project();
                    //
                    // project.setActualHours(rs.getBigDecimal("actual_hours"))
                    // project.setDifficulty(rs.getObject("difficulty", Integer
                    // project.setEstimatedHours(rs.getBigDecimal("estimated_h
                    // project.setNotes(rs.getString("notes")));
                    // project.setProjectId(rs.getObject("project_id", Integer
                    // project.setProjectName(rs.getString("project_name")));
                    //
                    // projects.add(project);
                }

                return projects;
            }
        }
    } catch(Exception e) {
        rollbackTransaction(conn);
        throw new DbException(e);
    }
} catch(SQLException e) {
    throw new DbException(e);
}
```



```

public Optional<Project> fetchProjectById(Integer projectId) {
    String sql = "SELECT * FROM " + PROJECT_TABLE + " WHERE project_id = ?";

    try(Connection conn = DbConnection.getConnection()) {
        startTransaction(conn);

        try {
            Project project = null;

            try(PreparedStatement stmt = conn.prepareStatement(sql)) {
                setParameter(stmt, 1, projectId, Integer.class);

                try(ResultSet rs = stmt.executeQuery()) {
                    if(rs.next()) {
                        project = extract(rs, Project.class);
                    }
                }
            }

            if(Objects.nonNull(project)) {
                project.getMaterials().addAll(fetchMaterialsForProject(conn, project));
                project.getSteps().addAll(fetchStepsForProject(conn, projectId));
                project.getCategories().addAll(fetchCategoriesForProject(conn, project));
            }

            commitTransaction(conn);
            return Optional.ofNullable(project);
        }
        catch(Exception e) {
            rollbackTransaction(conn);
            throw new DbException(e);
        }
    }
    catch(SQLException e) {
        throw new DbException(e);
    }
}

```

```

private List<Category> fetchCategoriesForProject(Connection conn,
    Integer projectId) throws SQLException {
    // @formatter:off
    String sql = ""
        + "SELECT c.* FROM " + CATEGORY_TABLE + " c "
        + "JOIN " + PROJECT_CATEGORY_TABLE + " pc USING (category_id) "
        + "WHERE project_id = ?";
    // @formatter:on

    try(PreparedStatement stmt = conn.prepareStatement(sql)) {
        setParameter(stmt, 1, projectId, Integer.class);

        try(ResultSet rs = stmt.executeQuery()) {
            List<Category> categories = new LinkedList<>();

            while(rs.next()) {
                categories.add(extract(rs, Category.class));
            }

            return categories;
        }
    }
}

private List<Step> fetchStepsForProject(Connection conn, Integer projectId) throws SQ
    String sql = "SELECT * FROM " + STEP_TABLE + " WHERE project_id = ?";

    try(PreparedStatement stmt = conn.prepareStatement(sql)) {
        setParameter(stmt, 1, projectId, Integer.class);

        try(ResultSet rs = stmt.executeQuery()) {
            List<Step> steps = new LinkedList<>();

            while(rs.next()) {
                steps.add(extract(rs, Step.class));
            }

            return steps;
        }
    }
}

```

```

private List<Material> fetchMaterialsForProject(Connection conn, Integer pr
    throws SQLException {
    String sql = "SELECT * FROM " + MATERIAL_TABLE + " WHERE project_id = ?";

    try(PreparedStatement stmt = conn.prepareStatement(sql)) {
        setParameter(stmt, 1, projectId, Integer.class);

        try(ResultSet rs = stmt.executeQuery()) {
            List<Material> materials = new LinkedList<>();

            while(rs.next()) {
                materials.add(extract(rs, Material.class));
            }

            return materials;
        }
    }
}

```