

MySQL Week 9 Exercises

Background

In the homework exercises, you are creating an application that will perform CRUD (Create, Read, Update, and Delete) operations on a MySQL database. This application connects to a DIY Project database and demonstrates many features of SQL and JDBC. Learning these skills will help prepare you for using these skills in the workplace.

As a reminder, in the last two weeks you used JDBC to connect to a MySQL database. Then, you diagrammed the project tables using Draw.io. Lastly, you wrote the CREATE TABLE statements for the five tables and created the tables in DBeaver.

In this week's exercises, you will begin development of the menu-driven application. You will use proper exception handling to gracefully manage any errors. You will write code to add project details to the project tables. This will involve properly creating and managing JDBC resources as well as database transactions.

In future exercises you will write code to read from a single table as well as from joined tables. Finally, you will write code to update and delete table rows.

Objectives

In these exercises, you will:

- Learn to write a menu-driven application with correct exception handling.
- Implement a scanner to gather user input from the console.
- Learn how to work with `BigDecimal` objects.
- Use JDBC to correctly handle resources (`Connections` and `PreparedStatement`s) ensuring that they are closed properly.
- Implement JDBC methods to insert a `Project` object into the project table.

MySQL Week 9 Exercises

Instructions



Points possible: 75

URL to GitHub Repository: <https://github.com/oscarc257/MySQL-Assignments.git>

URL to Public Link of your Video :

<https://www.dropbox.com/s/5ntx0xwr4t6wqxo/Week%209%20Assignment.mp4?dl=0>

Instructions:

1. Follow the [Exercises](#) below to complete this assignment.
 - In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.
 - Create a new repository on GitHub for this week's assignment and push your completed code to this dedicated repo, including your entire Maven Project Directory (e.g., mysql-java) and any .sql files that you create. In addition, screenshot your ERD and push the screenshot to your GitHub repo.
 - Include the screenshots into this Assignment Document indicated by: 
 - Create a video showcasing your work:
 - In this video: record and present your project verbally while showing the results of the working project.
 - Easy way to Create a video: Start a meeting in Zoom, share your screen, open Eclipse with the code and your Console window, start recording & record yourself describing and running the program showing the results.
 - Your video should be a maximum of 5 minutes.
 - Upload your video with a public link.
 - Easy way to Create a Public Video Link: Upload your video recording to YouTube with a public link.
 2. In addition, please include the following in your Coding Assignment Document:
 - The requested screenshots, indicated by: 
 - The URL for this week's GitHub repository.
 - The URL of the public link of your video.
 3. Save the Coding Assignment Document as a .pdf and do the following:
 - Push the .pdf to the GitHub repo for this week.
 - Upload the .pdf to the LMS in your Coding Assignment Submission.
-

MySQL Week 9 Exercises

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.

Exercises

In these exercises you will write code to create a menu-driven application. You will display menu selections to the user and will write the code to add project details to the DIY project tables. This will demonstrate the use of the INSERT statement (the Create part of CRUD).

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 to a variable and set up the parameters. Then, Eclipse can correctly create the method.

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

The exercises in this section build a menu-driven application. This application displays a list of available operations. The user selects which operation to perform. A switch statement then routes the selection to the appropriate method. Along the way you will add in proper exception handling. This is an important step to get right when building any application.

The purpose of the menu application is to perform CRUD operations on a relational database that holds information on DIY projects. Throughout the coming weeks you will add to this application to insert project rows, then materials, steps, and categories. You will fetch projects as a list and fetch an individual project with all the details. You will modify rows and delete an entire project with all associated detail (child) rows.

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". To prevent the Eclipse formatter from reformatting the list, surround the variable declaration with `// @formatter:off` and `// @formatter:on` so that it looks like this:

MySQL Week 9 Exercises

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

This list of operations will be printed on the console so that the user will be reminded which selection to make.

2. In this step you will use a `Scanner` to obtain input from a user from the Java console. A `Scanner` is a Java object that can be used to read from a variety of sources. When you create the `Scanner`, you will set its input source to `System.in`, which is the opposite of `System.out`. You use `System.out` to print to the console. You will use the `Scanner` to read from the console. So, the user types in selections and the `Scanner` reads the input and gives it to the application.

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. It should look like this:

```
private Scanner scanner = new Scanner(System.in);
```

3. In this step you will call the method that processes the menu. In the `main()` method, create a new `ProjectsApp` object and call the method: `processUserSelections()` method. The method takes zero parameters and returns nothing.

```
new ProjectsApp().processUserSelections();
```

4. Now you can create the `processUserSelections()` method as an instance method. This method displays the menu selections, gets a selection from the user, and then acts on the selection. Let Eclipse create the method for you by waving your mouse over the compiler error in the `main()` method (over the red squiggles). 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`.

```
boolean done = false;
```

```
while(!done) {
}
```

- 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

MySQL Week 9 Exercises

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:

```
private void processUserSelections() {
    boolean done = false;

    while(!done) {
        try {
            int selection = getUserSelection();
        }
        catch(Exception e) {
            System.out.println("\nError: " + e + " Try again.");
        }
    }
}
```

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 the method `printOperations()`. This method takes no parameters and returns nothing.
 - 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`. The method should look like this:

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

    Integer input = getIntInput("Enter a menu selection");

    return Objects.isNull(input) ? -1 : input;
}
```

6. Create the method `printOperations()`. It takes no parameters and returns nothing. This method does just what it says, it prints each available selection on a separate line in the console. In the `printOperations()` method:

- a. Print a line to the console:

```
System.out.println("\nThese are the available selections. Press the Enter key to quit:");
```

MySQL Week 9 Exercises

- 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. If you use a Lambda expression as shown in the video, it should look like this:

```
operations.forEach(line -> System.out.println("  " + line));
```

Every List object must implement the `forEach()` method. `forEach()` takes a `Consumer` interface object as a parameter. `Consumer` has a single abstract method, `accept()`. The `accept()` method takes a single parameter and returns nothing. The Lambda expression has a single parameter and `System.out.println` returns nothing. The Lambda expression thus matches the requirements for the `accept()` method.

If you don't want to use a Lambda expression, you can use an enhanced for loop to print the instructions.

7. There will be several user input methods that return different types of objects. Due to the way the `java.util.Scanner` object was implemented, the safest way to get an input line from the user is to input it as a `String` and then convert it to the appropriate type. With this design, all the input methods will ultimately call the `String` input method, which actually prints the prompt and uses the `Scanner` to get the user's input. In this step, you will write a method that returns an `Integer` value.

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 the `NumberFormatException` is totally obscure so it will get fixed in the `catch` block. Here's what the contents of the `try` block should look like:

```
return Integer.valueOf(input);
```
 - ii. In the `catch` block throw a new `DbException` with the message, `input + " is not a valid number. Try again."`
- d. The method should look like this:

MySQL Week 9 Exercises

```
private Integer getIntInput(String prompt) {
    String input = getStringInput(prompt);

    if(Objects.isNull(input)) {
        return null;
    }

    try {
        return Integer.valueOf(input);
    }
    catch(NumberFormatException e) {
        throw new DbException(input + " is not a valid number.");
    }
}
```

8. Now create the method that really prints the prompt and gets the input from the user. Create the method `getStringInput()`. It should have a single parameter of type `String` named `prompt`. This is the lowest level input method. The other input methods call this method and convert the input value to the appropriate type. This will also be called by methods that need to collect `String` data from the user. It should return a `String`. Inside the method:

- Print the prompt using `System.out.print(prompt + ": ")` to keep the cursor on the same line as the prompt. (Note: `print` and not `println`!)
- Assign a `String` variable named `input` to the results of a method call to `scanner.nextLine()`.
- Test the value of `input`. If it is blank return `null`. Otherwise return the trimmed value.
- The method should look like this:

```
private String getStringInput(String prompt) {
    System.out.print(prompt + ": ");
    String input = scanner.nextLine();

    return input.isBlank() ? null : input.trim();
}
```

- At this point the file should have no compile errors.

9. Now we want to add code that will process the user's selection. 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()`:

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

MySQL Week 9 Exercises

- 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. This will test that a non-integer selection prints an error message and gracefully recovers. Run the application. Click in the Eclipse console so that input will go to the scanner. Enter "abc" (without quotes) and press `Enter`. You should get an error message and be prompted again to enter a valid selection. Now press `Enter` with no input. The application should quit. Take a screen shot to show the application output



. It should look something like this:

The screenshot shows the Eclipse IDE with three tabs: `ProjectsApp.java`, `DbConnection.java`, and `mysql-java/pom.xml`. The `ProjectsApp.java` tab is active, showing the following code:

```
64
65     if(Objects.isNull(input)) {
66         return null;
67     }
68
69     try {
70         return Integer.valueOf(input);
71     }
72     catch(NumberFormatException e) {
73         throw new DbException(input + " is not a valid number.");
74     }
75
76 }
77
78 private String getStringInput(String prompt) {
79     System.out.print(prompt + ":");
```

Below the code editor is the `Console` tab, which shows the following output:

```
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.

These are the available selections. Press the Enter key to quit:
1) Add a project
Enter a menu selection:
```


MySQL Week 9 Exercises

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.

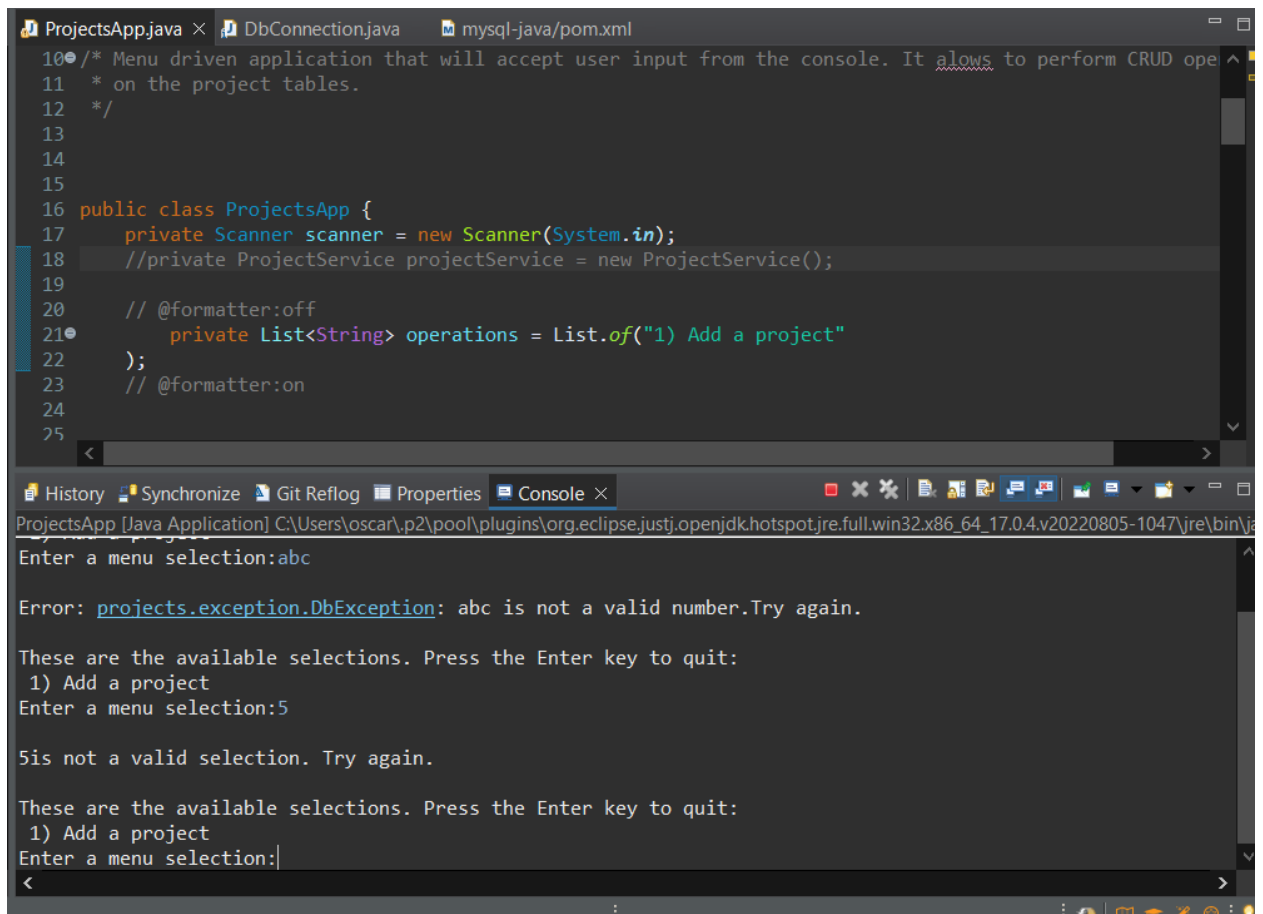
These are the available selections. Press the Enter key to quit:

1) Add a project

Enter a menu selection:

Exiting the menu.

- b. This will test that entering a valid `Integer` without a corresponding case statement will print an error message and recover gracefully. Run the application. Click in the Eclipse console so that input will go to the scanner. Enter "5" (without quotes) and press Enter. You should get an error message and be prompted again to enter a valid selection. Now press Enter with no input. The application should quit. Take a screenshot to show the application output . It should look something like this:



```
ProjectsApp.java x DbConnection.java x mysql-java/pom.xml
10 /* Menu driven application that will accept user input from the console. It allows to perform CRUD operations
11 * on the project tables.
12 */
13
14
15
16 public class ProjectsApp {
17     private Scanner scanner = new Scanner(System.in);
18     //private ProjectService projectService = new ProjectService();
19
20     // @formatter:off
21     private List<String> operations = List.of("1) Add a project"
22 );
23     // @formatter:on
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
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
```

MySQL Week 9 Exercises

```
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.
```

Add project files from student resources

Promineo Tech has provided some resources so that you don't have to write every bit of code. 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. You may need to expand some folders in the package explorer to make the `projects.entity` package visible. When done you should see `Category.java`, `Material.java`, `Project.java`, and `Step.java` in the `projects.entity` package. There should be no errors visible in those files.
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

You will now write the code to collect project information, create the project entities and insert the project row into the project table.

Modifications to the main application file

This section will collect project information from the user and call the service class (not written yet) to store the project row. In this section you will be introduced to the `BigDecimal` class if you haven't seen it before.

The `BigDecimal` class exactly represents decimal numbers (numbers with decimal places). In this, the decimal numbers act like `Integers` with a known number of decimal places. Money is a good example of this. Your bank may perform operations on money that results in fractional pennies, but at the end of the day your account is credited or debited with an exact dollar and penny amount. Fractional pennies are transitive and are not persisted to your account.

The `BigDecimal` object is perfectly suited to handle the SQL `DECIMAL` data type. `DECIMAL` values have a fixed number of digits (precision) and a fixed number of decimal places (scale). In the `CREATE TABLE` statement, a column definition of `DECIMAL (5, 2)` means that the value can range from -999.99 to 999.99. There are a maximum of five digits (precision) with two decimal places (scale).

`BigDecimal` contains immutable values – once created they cannot be changed. Any operation performed on `BigDecimal` results in a new `BigDecimal` object. `BigDecimals` can be created using a

MySQL Week 9 Exercises

constructor, then the scale can be set by calling the `setScale()` method. To set the scale to 2 (two decimal places) do something like this:

```
BigDecimal bd = new BigDecimal("1234.5678").setScale(2);
```

The JDBC driver has methods to natively handle `BigDecimal`. The driver automatically converts from Java `BigDecimal` to SQL `DECIMAL` and vice versa.

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.** (Hint: wave the mouse over `ProjectService`, which should have red squiggles under it. When the menu pops up, click "Create class 'ProjectService'". When the Java Class wizard pops up, change the value in the field "Package" from "projects" to "projects.service".) The editor will switch over to `ProjectService.java`. Switch back to `ProjectsApp.java`.

2. In this step, you will add code in the `switch` statement to handle user selection "1", which will call a method to collect project details and save them in the project table.

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. Now write the method to gather the project details from the user. Once collected, they will be put into a `Project` object. Then, another method will be called to save the project details.

Create the method `createProject()`. It is `private`, takes no parameters, and returns nothing. 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)")`. Note that the instructions don't include code to validate that the input is valid. You can do this if you want.
- e. Add local variable `String notes`. Assign the value to the result of calling `getStringInput("Enter the project notes")`.

MySQL Week 9 Exercises

- f. Create a new variable of type `Project` named `project`. Initialize it to a new `Project` object by calling the zero-argument constructor. Import the `Project` class from the `projects.entity` package. The `Project` class should have been added to the Eclipse project in the section "Add project files from student resources." If Eclipse can't find the import, follow the instructions in that section.
- g. Call the appropriate setters on the `Project` object to set `projectName`, `estimatedHours`, `actualHours`, `difficulty` and `notes`. For example, to add the project name on the `Project` object, call `setProjectName()` and pass it `projectName`.
- h. Call the `addProject()` method on the `projectService` object. Pass it the `Project` object. This method will be created shortly. 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.

The method should look like this. (There will be an error on the line that calls the `projectService` object and the lines that call `getDecimalInput()`).

```
private void createProject() {
    String projectName = getStringInput("Enter the project name");
    BigDecimal estimatedHours = getDecimalInput("Enter the estimated hours");
    BigDecimal actualHours = getDecimalInput("Enter the actual hours");
    Integer difficulty = getIntInput("Enter the project difficulty (1-5)");
    String notes = getStringInput("Enter the project notes");

    Project project = new Project();

    project.setProjectName(projectName);
    project.setEstimatedHours(estimatedHours);
    project.setActualHours(actualHours);
    project.setDifficulty(difficulty);
    project.setNotes(notes);

    Project dbProject = projectService.addProject(project);
    System.out.println("You have successfully created project: " + dbProject);
}
```

4. To get rid of the compilation errors, you will need to create two methods. In this step you will create the method `getDecimalInput()`.

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 lines:

- a. The line in the `try` block. Change it to:

```
return new BigDecimal(input).setScale(2);
```

MySQL Week 9 Exercises

This will create a new `BigDecimal` object and set the number of decimal places (the scale) to 2.

- b. The message in `DbException`. Change it to:

```
input + " is not a valid decimal number."
```

- Now create the second method that will fix the compilation errors. Wave the mouse over `"projectService.addProject()"`. When the menu pops up, select "Add method 'addProject(project)' in in type 'ProjectService'".
- Save all files. All compiler errors should now be gone.

Modifications to project service

The service layer in this small application is implemented by a single file, `ProjectService.java`.

Mostly this file acts as a pass-through between the main application file that runs the menu (`ProjectsApp.java`) and the DAO file in the data layer (`ProjectDao.java`).

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

- In this step, you will create the DAO class and initialize a variable of that type. 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. If possible, let Eclipse create the class for you. In any event, create a `ProjectDao` class in the `projects.dao` package. Make sure that `ProjectDao` extends `DaoBase` from the `provided.util` package. Save all files. You should have no compile errors. The editor will probably change to the `ProjectDao` class. Change back to the `ProjectService` class.
- 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. The `addProject()` method should look like this:

```
public Project addProject(Project project) {  
    return projectDao.insertProject(project);  
}
```

- Wave the mouse over `insertProject()` (with the red squiggles) and select "Create method 'insertProject(Project)' in type 'ProjectDao'". Save all files. You should have no compile errors.

Modifications to project DAO

Now you want to create the class that will read and write to the MySQL database. In this section you will write the values that were collected from the user and that are contained in a `Project` object to the project table using JDBC method calls.

In this section you will be working in file `ProjectDao.java` in the `projects.dao` package. If you followed the steps above, `ProjectDao` should extend `DaoBase`. If it doesn't, do that now or you will run into problems later.

MySQL Week 9 Exercises

Add constants

First, you should add some constants with the table names. It's a good idea to add constants for values that are used over and over again in a class. The table names are used by all the methods that write to or read from the tables.

In this section, you will be adding constants into the `ProjectDao` class. These are placed at the top of the class just inside the class body. Java does not have a "constant" keyword. Instead, a constant is specified using `static final`. Constants can either be `public` or `private`. In this file all the constants should be `private`.

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".

The constants should look like this:

```
private static final String CATEGORY_TABLE = "category";
private static final String MATERIAL_TABLE = "material";
private static final String PROJECT_TABLE = "project";
private static final String PROJECT_CATEGORY_TABLE = "project_category";
private static final String STEP_TABLE = "step";
```

Save the project details

There are several steps that must be taken to save the project details. First, you must create the SQL statement. Then you will obtain a `Connection` and start a transaction. Next you will obtain a `PreparedStatement` and set the parameter values from the `Project` object. Finally, you will save the data and commit the transaction. Follow the steps below to 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`. Make sure to add the correct blank spaces between words or it won't work. It should look like this:

MySQL Week 9 Exercises

```
// @formatter:off
String sql = ""
    + "INSERT INTO " + PROJECT_TABLE + " "
    + "(project_name, estimated_hours, actual_hours, difficulty, notes) "
    + "VALUES "
    + "(?, ?, ?, ?, ?)";
// @formatter:on
```

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.

```
try(Connection conn = DbConnection.getConnection()) {
}
catch(SQLException e) {
    throw new DbException(e);
}
```

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 the `Exception` object passed into the catch block. This will ensure that the transaction is rolled back when an exception is thrown.
 - c. The method should look like this at this point:

MySQL Week 9 Exercises

```
public Project insertProject(Project project) {
    // @formatter:off
    String sql = ""
        + "INSERT INTO " + PROJECT_TABLE + " "
        + "(project_name, estimated_hours, actual_hours, difficulty, notes) "
        + "VALUES "
        + "(?, ?, ?, ?, ?)";
    // @formatter:on

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

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

Transaction started

Transaction rolled back

5. In this step you will set the project details as parameters in the `PreparedStatement` object. Inside the inner `try` block, set the parameters on the `Statement`. Use the convenience method in `DaoBase` `setParameter()`. This method handles `null` values correctly. (See the JavaDoc comments on that method for details.) Add these parameters: `projectName`, `estimatedHours`, `actualHours`, `difficulty`, and `notes`. When done it should look like this:

```
setParameter(stmt, 1, project.getProjectName(), String.class);
setParameter(stmt, 2, project.getEstimatedHours(), BigDecimal.class);
setParameter(stmt, 3, project.getActualHours(), BigDecimal.class);
setParameter(stmt, 4, project.getDifficulty(), Integer.class);
setParameter(stmt, 5, project.getNotes(), String.class);
```

6. Now you can save the project details. Perform the insert by calling `executeUpdate()` on the `PreparedStatement` object. Do not pass any parameters to `executeUpdate()` or it will reset all the parameters leading to an obscure error.
7. Obtain the project ID (primary key) by calling the convenience method in `DaoBase`, `getLastInsertId()`. (See the JavaDoc documentation on that method for details.) 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. At this point there should be no compile errors. The method should now look like this:

MySQL Week 9 Exercises

```
public Project insertProject(Project project) {
    // @formatter:off
    String sql = ""
        + "INSERT INTO " + PROJECT_TABLE + " "
        + "(project_name, estimated_hours, actual_hours, difficulty, notes) "
        + "VALUES "
        + "(?, ?, ?, ?, ?)";
    // @formatter:on

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

        try(PreparedStatement stmt = conn.prepareStatement(sql)) {
            setParameter(stmt, 1, project.getProjectName(), String.class);
            setParameter(stmt, 2, project.getEstimatedHours(), BigDecimal.class);
            setParameter(stmt, 3, project.getActualHours(), BigDecimal.class);
            setParameter(stmt, 4, project.getDifficulty(), Integer.class);
            setParameter(stmt, 5, project.getNotes(), String.class);


            stmt.executeUpdate();

            Integer projectId = getLastInsertId(conn, PROJECT_TABLE);
            commitTransaction(conn);

            project.setProjectId(projectId);
            return project;
        }
        catch(Exception e) {
            rollbackTransaction(conn);
            throw new DbException(e);
        }
    }
    catch(SQLException e) {
        throw new DbException(e);
    }
}
```

Test it

After all that coding, it's a good idea to test that it actually works. You need to ensure that you can add a project row to the project table with no errors.

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. It should look something like this:

MySQL Week 9 Exercises

```
30
31     try(Connection conn = DbConnection.getConnection()) {
32         startTransaction(conn);
33         try(PreparedStatement stmt = conn.prepareStatement(sql)) {

History Synchronize Git Reflog Properties Console X
ProjectsApp [Java Application] C:\Users\oscar.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v202208

These are the available selections. Press the Enter key to quit:
1) Add a project
Enter a menu selection:1
Enter the project name:scroll
Enter the estimate hours:5
Enter the actual hours:4
Enter the project difficulty (1-5):3
Enter the projectnotes:Use nails to hang scroll
Connecting with uri=jdbc:mysql://localhost:3306/projects?user=projects&password=projects
Connection to schema 'projects' is successful.

Error: projects.exception.DbException: java.lang.ClassCastException: class java.math.BigDecimal

These are the available selections. Press the Enter key to quit:
1) Add a project
Enter a menu selection:
Error: projects.exception.DbException: nails is not a valid number.Try again.

These are the available selections. Press the Enter key to quit:
1) Add a project
Enter a menu selection:
Error: projects.exception.DbException: to is not a valid number.Try again.

These are the available selections. Press the Enter key to quit:
1) Add a project
Enter a menu selection:
Error: projects.exception.DbException: hang is not a valid number.Try again.

These are the available selections. Press the Enter key to quit:
1) Add a project
Enter a menu selection:
Error: projects.exception.DbException: scroll is not a valid number.Try again.

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

MySQL Week 9 Exercises

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

1) Add a project

Enter a menu selection: 1

Enter the project name: Hang a door

Enter the estimated hours: 4

Enter the actual hours: 3

Enter the project difficulty (1-5): 3

Enter the project notes: Use the door hangers from Home Depot

Connection to schema 'projects' is successful.

You have successfully created project:

ID=1

name=Hang a door

estimatedHours=4.00

actualHours=3.00

difficulty=3

notes=Use the door hangers from Home Depot

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.

MySQL Week 9 Exercises

Solutions

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

ProjectsApp.java

```
package projects;

import java.math.BigDecimal;
import java.util.List;
import java.util.Objects;
import java.util.Scanner;
import projects.entity.Project;
import projects.exception.DbException;
import projects.service.ProjectService;

/**
 * This class is a menu-driven application that accepts user input from the console. It then
 * performs CRUD operations on the project tables.
 *
 * @author Promineo
 */
public class ProjectsApp {
    private Scanner scanner = new Scanner(System.in);
    private ProjectService projectService = new ProjectService();

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

    /**
     * Entry point for Java application.
     *
     * @param args Unused.
     */
    public static void main(String[] args) {
        new ProjectsApp().processUserSelections();
    }
}
```

MySQL Week 9 Exercises

```
/**
 * This method prints the operations, gets a user menu selection, and performs the requested
 * operation. It repeats until the user requests that the application terminate.
 */
private void processUserSelections() {
    boolean done = false;

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

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

                case 1:
                    createProject();
                    break;

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

/**
 * Gather user input for a project row then call the project service to create the row.
 */
private void createProject() {
    String projectName = getStringInput("Enter the project name");
    BigDecimal estimatedHours = getDecimalInput("Enter the estimated hours");
    BigDecimal actualHours = getDecimalInput("Enter the actual hours");
    Integer difficulty = getIntInput("Enter the project difficulty (1-5)");
    String notes = getStringInput("Enter the project notes");

    Project project = new Project();

    project.setProjectName(projectName);
    project.setEstimatedHours(estimatedHours);
    project.setActualHours(actualHours);
    project.setDifficulty(difficulty);
    project.setNotes(notes);

    Project dbProject = projectService.addProject(project);
    System.out.println("You have successfully created project: " + dbProject);
}
```

MySQL Week 9 Exercises

```
/**
 * Gets the user's input from the console and converts it to a BigDecimal.
 *
 * @param prompt The prompt to display on the console.
 * @return A BigDecimal value if successful.
 * @throws DbException Thrown if an error occurs converting the number to a BigDecimal.
 */
private BigDecimal getDecimalInput(String prompt) {
    String input = getStringInput(prompt);

    if(Objects.isNull(input)) {
        return null;
    }

    try {
        /* Create the BigDecimal object and set it to two decimal places (the scale). */
        return new BigDecimal(input).setScale(2);
    }
    catch(NumberFormatException e) {
        throw new DbException(input + " is not a valid decimal number.");
    }
}

/**
 * Called when the user wants to exit the application. It prints a message and returns
 * {@code true} to terminate the app.
 *
 * @return {@code true}
 */
private boolean exitMenu() {
    System.out.println("Exiting the menu.");
    return true;
}

/**
 * This method prints the available menu selections. It then gets the user's menu selection from
 * the console and converts it to an int.
 *
 * @return The menu selection as an int or -1 if nothing is selected.
 */
private int getUserSelection() {
    printOperations();

    Integer input = getIntInput("Enter a menu selection");

    return Objects.isNull(input) ? -1 : input;
}
```

MySQL Week 9 Exercises

```
/**
 * Prints a prompt on the console and then gets the user's input from the console. It then
 * converts the input to an Integer.
 *
 * @param prompt The prompt to print.
 * @return If the user enters nothing, {@code null} is returned. Otherwise, the input is converted
 *         to an Integer.
 * @throws DbException Thrown if the input is not a valid Integer.
 */
private Integer getIntInput(String prompt) {
    String input = getStringInput(prompt);

    if(Objects.isNull(input)) {
        return null;
    }

    try {
        return Integer.valueOf(input);
    }
    catch(NumberFormatException e) {
        throw new DbException(input + " is not a valid number.");
    }
}

/**
 * Prints a prompt on the console and then gets the user's input from the console. If the user
 * enters nothing, {@code null} is returned. Otherwise, the trimmed input is returned.
 *
 * @param prompt The prompt to print.
 * @return The user's input or {@code null}.
 */
private String getStringInput(String prompt) {
    System.out.print(prompt + ": ");
    String input = scanner.nextLine();

    return input.isBlank() ? null : input.trim();
}

/**
 * Print the menu selections, one per line.
 */
private void printOperations() {
    System.out.println("\nThese are the available selections. Press the Enter key to quit:");

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

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

MySQL Week 9 Exercises

ProjectService.java

```
package projects.service;

import projects.dao.ProjectDao;
import projects.entity.Project;

/**
 * @author Promineo
 */
public class ProjectService {
    private ProjectDao projectDao = new ProjectDao();

    /**
     * This method simply calls the DAO class to insert a project row.
     *
     * @param project The {@link Project} object.
     * @return The Project object with the newly generated primary key value.
     */
    public Project addProject(Project project) {
        return projectDao.insertProject(project);
    }
}
```

ProjectDao.java

```
package projects.dao;

import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import projects.entity.Project;
import projects.exception.DbException;
import provided.util.DaoBase;

/**
 * This class uses JDBC to perform CRUD operations on the project tables.
 *
 * @author Promineo
 */
@SuppressWarnings("unused")
public class ProjectDao extends DaoBase {
    private static final String CATEGORY_TABLE = "category";
    private static final String MATERIAL_TABLE = "material";
    private static final String PROJECT_TABLE = "project";
    private static final String PROJECT_CATEGORY_TABLE = "project_category";
    private static final String STEP_TABLE = "step";
}
```


MySQL Week 9 Exercises

```
/**
 * Insert a project row into the project table.
 *
 * @param project The project object to insert.
 * @return The Project object with the primary key.
 * @throws DbException Thrown if an error occurs inserting the row.
 */
public Project insertProject(Project project) {
    // @formatter:off
    String sql = ""
        + "INSERT INTO " + PROJECT_TABLE + " "
        + "(project_name, estimated_hours, actual_hours, difficulty, notes) "
        + "VALUES "
        + "(?, ?, ?, ?, ?)";
    // @formatter:on

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

        try(PreparedStatement stmt = conn.prepareStatement(sql)) {
            setParameter(stmt, 1, project.getProjectName(), String.class);
            setParameter(stmt, 2, project.getEstimatedHours(), BigDecimal.class);
            setParameter(stmt, 3, project.getActualHours(), BigDecimal.class);
            setParameter(stmt, 4, project.getDifficulty(), Integer.class);
            setParameter(stmt, 5, project.getNotes(), String.class);

            stmt.executeUpdate();

            Integer projectId = getLastInsertId(conn, PROJECT_TABLE);
            commitTransaction(conn);

            project.setProjectId(projectId);
            return project;
        }
        catch(Exception e) {
            rollbackTransaction(conn);
            throw new DbException(e);
        }
    }
    catch(SQLException e) {
        throw new DbException(e);
    }
}
```