

```
ProjectsApp.java × ProjectService.java ProjectDao.java project_schema.sql
1 package projects;
2
3 import java.math.BigDecimal;
4 import java.util.List;
5 import java.util.Objects;
6 import java.util.Scanner;
7
8 import projects.entity.Project;
9 import projects.exception.DbException;
10 import projects.service.ProjectService;
11
12
13 public class ProjectsApp {
14     private Scanner scanner = new Scanner(System.in);
15     private ProjectService projectService = new ProjectService();
16     private Project curProject;
17
18
19
20     //@formatter:off
21     private List<String> operations = List.of(
22         "1) Create and populate all tables",
23         "2) List projects",
24         "3) Select a project",
25         "4) Update project details",
26         "5) Delete a project");
27
28     //@formatter:on
29
30
31     public static void main(String[] args) {
32         new ProjectsApp().processUserSelections();
33     }
34     private void processUserSelections() {
35         boolean done = false;
36
37         while(!done) {
38             try {
39                 int selection = getUserSelection();
40
41                 switch(selection) {
42                     case -1:
43                         done = exitMenu();
44                         break;
45
46                     case 1:
```

```
ProjectsApp.java × ProjectService.java ProjectDao.java project_schema.sql
47         createProject();
48         break;
49     case 2:
50         listProjects();
51         break;
52     case 3:
53         selectProject();
54         break;
55     case 4:
56         updateProjectDetails();
57         break;
58     case 5:
59         deleteProject();
60         break;
61
62     default:
63         System.out.println("\n" + selection + " is not valid selection. Tray again. ");
64
65     }
66
67 }
68 catch(Exception e) {
69     System.out.println("\nError: " + e + "Try again.");
70 }
71 }
72
73 }
74
75 private void deleteProject() {
76     listProjects();
77
78     Integer projectId = getIntInput("Enter the ID of the Project to delete");
79
80     projectService.deleteProject(projectId);
81     System.out.println("Project" + projectId + " was deleted successfully.");
82
83     if(Objects.nonNull(projectId)) {
84         projectService.deleteProject(projectId);
85     }
86
87     if(Objects.nonNull(curProject) && curProject.getProjectId().equals(projectId)) {
88         curProject = null;
89     }
90 }
91
92 }
```

```

1 *ProjectsApp.java x ProjectService.java ProjectDao.java project_schema.sql
91
92 private void updateProjectDetails() throws Exception {
93     if(Objects.isNull(curProject)) {
94         System.out.println("\nPlease select a project.");
95         return;
96     }
97     String projectName = getStringInput("Enter the project name[" + curProject.getProjectName() + "]\n");
98     BigDecimal estimatedHours = getDecimalInput("Enter the estimated hours [ " + curProject.getEstimatedHours() + "]\n");
99     BigDecimal actualHours = getDecimalInput("Enter the actual hours [ " + curProject.getActualHours() + "]\n");
100     Integer difficulty = getIntInput("Enter the project difficulty (1-5) [" + curProject.getDifficulty() + "]\n");
101     String notes = getStringInput("Enter the project notes[" + curProject.getNotes() + "]\n");
102
103     Project project = new Project();
104
105     project.setProjectId(curProject.getProjectId());
106     project.setProjectName(Objects.isNull(projectName) ? curProject.getProjectName() : projectName);
107
108     project.setEstimatedHours(Objects.isNull(estimatedHours) ? curProject.getEstimatedHours() : estimatedHours);
109     project.setActualHours(Objects.isNull(actualHours) ? curProject.getActualHours() : actualHours);
110     project.setDifficulty(Objects.isNull(difficulty) ? curProject.getDifficulty() : difficulty);
111     project.setNotes(Objects.isNull(notes) ? curProject.getNotes() : notes);
112
113     projectService.modifyProjectDetails(project);
114     curProject = projectService.fetchProjectById(curProject.getProjectId());
115
116
117
118
119
120 private void selectProject() throws Exception {
121     listProjects();
122     Integer projectId = getIntInput("Enter a project ID to select a project");
123
124     /* Unselect the current project*/
125     curProject = null;
126
127     /*This will throw an exception if an invalid project ID is entered.*/
128
129     curProject = projectService.fetchProjectById(projectId);
130
131
132 private void listProjects() {
133     List<Project> projects = projectService.fetchAllProjects();
134     System.out.println("\nProjects:");
135     projects.forEach(project -> System.out.println
136         ((" " + project.getProjectId() + " : " + project.getProjectName())));
137

```

*ProjectsApp.java × ProjectService.java ProjectDao.java project_schema.sql

```
137
138
139 private void createProject() {
140     String projectName = getStringInput("Enter the project name");
141     BigDecimal estimatedHours = getDecimalInput("Enter the estimated hours");
142     BigDecimal actualHours = getDecimalInput("Enter the actual hours");
143     Integer difficulty = getIntInput("Enter the project difficulty (1-5)");
144     String notes = getStringInput("Enter the project notes");
145
146     Project project = new Project();
147
148     project.setProjectName(projectName);
149     project.setEstimatedHours(estimatedHours);
150     project.setActualHours(actualHours);
151     project.setDifficulty(difficulty);
152     project.setNotes(notes);
153
154     Project dbProject = projectService.addProject(project);
155     System.out.println("You have successfully created project: " + dbProject);
156
157 }
158
159 private BigDecimal getDecimalInput(String prompt) {
160     String input = getStringInput(prompt);
161
162     if(Objects.isNull(input)) {
163         return null;
164     }
165     try {
166         return new BigDecimal(input).setScale(2);
167     }
168     catch(NumberFormatException e) {
169         throw new DbException(input + "is not a valid decimal number.");
170     }
171 }
172
173 private boolean exitMenu() {
174     System.out.println("Exiting the menu.");
175     return true;
176 }
177
178 private int getUserSelection() {
179     printOperations();
180
181     Integer input = getIntInput("Enter a menu selection");
182     return Objects.isNull(input)? -1 : input;
183 }
```

```

182         return Objects.isNull(input)? -1 : input;
183     }
184
185
186     private Integer getIntInput(String prompt) {
187         String input = getStringInput(prompt);
188
189         if(Objects.isNull(input)) {
190             return null;
191         }
192
193         try {
194             return Integer.valueOf(input);
195         }
196         catch(NumberFormatException e) {
197             throw new DbException(input + " is not a valid number.");
198         }
199     }
200
201     private String getStringInput(String prompt) {
202         System.out.print(prompt + ": ");
203         String input = scanner.nextLine();
204
205         return input.isBlank()? null : input.trim();
206     }
207
208
209     private void printOperations() {
210         System.out.println("\nThese are the available selections. press the Enter key to quit:");
211
212         operations.forEach(line -> System.out.println(" " + line));
213
214         if(Objects.isNull(curProject)) {
215             System.out.println("\nYou are not working with a project.");
216         }
217         else {
218             System.out.println("\nYou are working with project: " + curProject);
219         }
220     }
221 }
222

```

ProjectsApp.java ProjectService.java X ProjectDao.java project_schema.sql

```
1 package projects.service;
2
3 import java.util.List;
4
5
6
7
8
9
10
11 public class ProjectService {
12     private static final String SCHEMA_FILE = "PROJECT_SCHEMA.SQL";
13     private static final String DATA_FILE = "project_data.sql";
14     private ProjectDao projectDao = new ProjectDao();
15
16     public Project fetchProjectById(Integer projectId) throws Exception {
17         Optional<Project> op = projectDao.fetchProjectById(projectId);
18
19         return projectDao.fetchProjectById(projectId).orElseThrow( ()
20             -> new NoSuchElementException
21                 ("Project with Project ID=" + projectId
22                  + " does not exist.));
23     }
24
25
26
27     public Project addProject(Project project) {
28         return projectDao.insertProject(project);
29     }
30
31
32
33     public List<Project> fetchAllProjects() {
34
35         return projectDao.fetchAllProjects();
36     }
37
38     public void modifyProjectDetails(Project project) {
39
40         if(!projectDao.modifyProjectDetails(project)) {
41             throw new DbException("Project with ID=" + project.getProjectId() + " does not exist.");
42         }
43     }
44
45
46
47
48
49     public void deleteProject(Integer projectId) {
50         if(!projectDao.deleteProject(projectId)) {
51             throw new DbException(" Project with ID=" + projectId + " does not exist.");
52         }
53     }
```



```

1 package projects.dao;
2 import java.math.BigDecimal;
19
20
21 public class ProjectDao extends DaoBase{
22     private static final String CATEGORY_TABLE = "category";
23     private static final String MATERIAL_TABLE = "material";
24     private static final String PROJECT_TABLE = "project";
25     private static final String PROJECT_CATEGORY_TABLE = "project_category";
26     private static final String STEP_TABLE = "step";
27
28
29 public Project insertProject(Project project) {
30     //@formatter: off
31     String sql = ""
32         + "INSERT INTO " + PROJECT_TABLE + " "
33         + "(project_name, estimated_hours, actual_hours, difficulty, notes)"
34         + "VALUES "
35         + "(?, ?, ?, ?, ?)";
36     //@formatter: on
37
38     try(Connection conn = DbConnection.getConnection()){
39         startTransaction(conn);
40
41         try(PreparedStatement stmt = conn.prepareStatement(sql)){
42             setParameter(stmt, 1, project.getProjectName(), String.class);
43             setParameter(stmt, 2, project.getEstimatedHours(), BigDecimal.class);
44             setParameter(stmt, 3, project.getActualHours(), BigDecimal.class);
45             setParameter(stmt, 4, project.getDifficulty(), Integer.class);
46             setParameter(stmt, 5, project.getNotes(), String.class);
47
48             stmt.executeUpdate();
49
50             Integer projectId = getLastInsertId(conn, PROJECT_TABLE);
51             commitTransaction(conn);
52
53             project.setProjectId(projectId);
54             return project;
55         }
56         catch(Exception e) {
57             rollbackTransaction(conn);
58             throw new DbException(e);
59         }
60     }
61     catch(SQLException e) {
62         throw new DbException(e);

```

```

64     }
65
66
67     public Optional<Project> fetchProjectById(Integer projectId) throws Exception {
68
69         String sql = "SELECT * FROM " + PROJECT_TABLE + " WHERE project_id = ?";
70
71         try(Connection connn = DbConnection.getConnection()){
72             startTransaction(connn);
73
74             try {
75                 Project project = null;
76                 try(PreparedStatement stmt = connn.prepareStatement(sql)){
77                     setParameter(stmt, 1, projectId, Integer.class);
78
79                     try(ResultSet rs = stmt.executeQuery()){
80                         if(rs.next()) {
81                             project = extract(rs, Project.class);
82                         }
83                     }
84                 }
85
86                 if(Objects.nonNull(project)) {
87                     project.getMaterials().addAll(fetchMaterialsForProject(connn, projectId));
88                     project.getSteps().addAll(fetchStepsForProject(connn, projectId));
89                     project.getCategories().addAll(fetchCategoriesForProject(connn, projectId));
90                 }
91
92                 commitTransaction(connn);
93
94                 return Optional.ofNullable(project);
95             }
96             catch (Exception e) {
97                 rollbackTransaction(connn);
98                 throw DbException(e);
99             }
100         }
101         catch (SQLException e) {
102             throw new DbException(e);
103         }
104     }
105
106
107     private List<Category> fetchCategoriesForProject
108     (Connection connn, Integer projectId) throws SQLException{

```


*ProjectsApp.java ProjectService.java ProjectDao.java X project_schema.sql

```
1109 // @formatter:off
1110 String sql = ""
1111     + "SELECT c.* FROM " + CATEGORY_TABLE + " c "
1112     + "JOIN " + PROJECT_CATEGORY_TABLE + " pc USING (category_id) "
1113     + "WHERE project_id = ?";
1114 // @formatter:on
1115
1116 try(PreparedStatement stmt = conn.prepareStatement(sql)){
1117     setParameter(stmt, 1, projectId, Integer.class);
1118
1119     try(ResultSet rs = stmt.executeQuery()){
1120         List<Category> categories = new LinkedList<>();
1121
1122         while(rs.next()) {
1123             categories.add(extract(rs, Category.class));
1124         }
1125         return categories;
1126     }
1127 }
1128
1129
1130
1131 }
1132
1133
1134 private List<Step> fetchStepsForProject
1135 (Connection conn, Integer projectId) throws SQLException{
1136     String sql = "SELECT * FROM " + STEP_TABLE + " WHERE project_id = ?";
1137
1138     try(PreparedStatement stmt = conn.prepareStatement(sql)){
1139         setParameter(stmt, 1, projectId, Integer.class);
1140
1141         try(ResultSet rs = stmt.executeQuery()){
1142             List<Step> steps = new LinkedList<>();
1143
1144             while(rs.next()) {
1145                 steps.add(extract(rs, Step.class));
1146             }
1147             return steps;
1148         }
1149     }
1150 }
1151
1152 private List<Material> fetchMaterialsForProject
1153 (Connection conn, Integer projectId) throws SQLException{
1154     String sql = "SELECT * FROM " + MATERIAL_TABLE + " WHERE project_id = ?";
1155 }
```

```

156         try(PreparedStatement stmt = conn.prepareStatement(sql)){
157             setParameter(stmt, 1, projectId, Integer.class);
158
159             try(ResultSet rs = stmt.executeQuery()){
160                 List<Material> materials = new LinkedList<>();
161
162                 while(rs.next()) {
163                     materials.add(extract(rs, Material.class));
164                 }
165                 return materials;
166             }
167         }
168     }
169
170
171     private Exception DbException(Exception e) {
172         return null;
173     }
174
175
176
177     public List<Project> fetchAllProjects() {
178
179         String sql = "SELECT * FROM " + PROJECT_TABLE + " ORDER BY project_name";
180
181         try(Connection conn = DbConnection.getConnection()){
182             startTransaction(conn);
183
184             try(PreparedStatement stmt = conn.prepareStatement(sql)){
185                 try(ResultSet rs = stmt.executeQuery()){
186                     List<Project> projects = new LinkedList<>();
187
188                     while (rs.next()) {
189                         projects.add(extract(rs, Project.class));
190                         /* Alternative approach*/
191                         // Project project = new Project();
192
193                         //project.setActualHours(rs.getBigDecimal("actual_hours"));
194                         //project.setDifficulty(rs.getObject("difficulty", Integer.class));
195                         //project.setEstimatedHours(rs.getBigDecimal("estimated_hours"));
196                         //project.setNotes(rs.getString("notes"));
197                         //project.setProjectId(rs.getObject("project_id", Integer.class));
198                         //project.setProjectName(rs.getString("project_name"));
199
200                         //projects.add(project);

```

```

200         // projects.add(project);
201     }
202     return projects;
203 }
204 }
205
206 catch (Exception e) {
207     rollbackTransaction(conn);
208     throw new DbException(e);
209 }
210 }
211 }
212 catch (SQLException e) {
213     throw new DbException(e);
214 }
215 }
216
217
218 public boolean modifyProjectDetails(Project project) {
219     // @formatter : off
220     String sql = ""
221         + "UPDATE " + PROJECT_TABLE + " SET "
222         + "project_name = ?, "
223         + "actual_hours = ?, "
224         + "difficulty = ? , "
225         + "notes = ? "
226         + "WHERE project_id = ?";
227     // @formatter : on
228
229     try (Connection conn = DbConnection.getConnection()) {
230         startTransaction(conn);
231
232         try (PreparedStatement stmt = conn.prepareStatement(sql)) {
233             setParameter(stmt, 1, project.getProjectName(), String.class);
234             setParameter(stmt, 2, project.getEstimatedHours(), BigDecimal.class);
235             setParameter(stmt, 3, project.getActualHours(), BigDecimal.class);
236             setParameter(stmt, 4, project.getDifficulty(), Integer.class);
237             setParameter(stmt, 5, project.getNotes(), String.class);
238             setParameter(stmt, 6, project.getProjectId(), Integer.class);
239
240             boolean modified = stmt.executeUpdate() == 1;
241             commitTransaction(conn);
242
243             return modified;
244         }
245         catch (Exception e) {
246             rollbackTransaction(conn);

```

```

247         throw new DbException(e);
248     }
249 }
250 catch(SQLException e) {
251     throw new DbException(e);
252 }
253 }
254 }
255 }
256 }
257 }
258 public boolean deleteProject(Integer projectId) {
259     String sql = "DELETE FROM " + PROJECT_TABLE + " WHERE project_id = ?";
260
261     try(Connection conn = DbConnection.getConnection()){
262         startTransaction(conn);
263
264         try(PreparedStatement stmt = conn.prepareStatement(sql)){
265             setParameter(stmt, 1, projectId, Integer.class);
266
267             boolean deleted = stmt.executeUpdate() == 1;
268
269             commitTransaction(conn);
270             return deleted;
271         }
272         catch(Exception e) {
273             rollbackTransaction(conn);
274             throw new DbException(e);
275         }
276     }
277     catch(SQLException e) {
278         throw new DbException(e);
279     }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }

```

Console X

ProjectsApp [Java Application] C:\Program Files\Java\jdk-17.0.3.1\bin\javaw.exe (Sep 5, 2022, 9:27:16 PM) [pid: 7640]

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

- 1) Create and populate all tables
- 2) List projects
- 3) Select a project
- 4) Update project details
- 5) Delete a project

You are not working with a project.

Enter a menu selection: 4

Please select a project.

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

- 1) Create and populate all tables
- 2) List projects
- 3) Select a project
- 4) Update project details
- 5) Delete a project

You are not working with a project.

Enter a menu selection:

Console X

ProjectsApp [Java Application] C:\Program Files\Java\jdk-17.0.3.1\bin\javaw.exe (Sep 5, 2022, 9:31:42 PM) [pid: 8548]

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

- 1) Create and populate all tables
- 2) List projects
- 3) Select a project
- 4) Update project details
- 5) Delete a project

You are not working with a project.

Enter a menu selection: 3

Successfully obtained connection!

Projects:

- 1: Caulk around cabinets
- 2: Caulk around cabinets
- 3: Caulk around cabinets
- 4: Caulk around cabinets
- 5: Hang a door
- 6: Hang a door
- 7: Hang a door

Enter a project ID to select a project: 5

Successfully obtained connection!

Successfully obtained connection!

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

- 1) Create and populate all tables
- 2) List projects
- 3) Select a project
- 4) Update project details
- 5) Delete a project

You are working with project:

ID=5
name=Hang a door
estimatedHours=4.00
actualHours=3.00
difficulty=2
notes=Keep cabinets clean
Materials:
Steps:
Categories:

Enter a menu selection: 4

Enter the project name[Hang a door]:

<https://github.com/bmason1969/Week-11.git>