

```
ProjectsApp.java X ProjectService.java ProjectDao.java
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     //@formatter:on
26
27
28
29     public static void main(String[] args) {
30         new ProjectsApp().processUserSelections();
31     }
```

```
32 private void processUserSelections() {
33     boolean done = false;
34
35     while(!done) {
36         try {
37             int selection = getUserSelection();
38
39             switch(selection) {
40                 case -1:
41                     done = exitMenu();
42                     break;
43
44                 case 1:
45                     createProject();
46                     break;
47                 case 2:
48                     listProjects();
49                     break;
50                 case 3:
51                     selectProject();
52                     break;
53
54                 default:
55                     System.out.println("\n" + selection + " is not valid selection. Try again. ");
56
57             }
58         }
59     }
60     catch(Exception e) {
61         System.out.println("\nError: " + e + "Try again.");
62     }
```

```

62     }
63 }
64
65 }
66
67 private void selectProject() throws Exception {
68     listProjects();
69     Integer projectId = getIntInput("Enter a project ID to select a project");
70
71     /* Unselect the current project */
72     curProject = null;
73
74     /* This will throw an exception if an invalid project ID is entered. */
75
76     curProject = projectService.fetchProjectById(projectId);
77
78 }
79 private void listProjects() {
80     List<Project> projects = projectService.fetchAllProjects();
81     System.out.println("\nProjects:");
82     projects.forEach(project -> System.out.println
83         ((" " + project.getProjectId() + ": " + project.getProjectName())));
84
85 }
86 private void createProject() {
87     String projectName = getStringInput("Enter the project name");
88     BigDecimal estimatedHours = getDecimalInput("Enter the estimated hours");
89     BigDecimal actualHours = getDecimalInput("Enter the actual hours");
90     Integer difficulty = getIntInput("Enter the project difficulty (1-5)");
91     String notes = getStringInput("Enter the project notes");
92 }

```

ProjectsApp.java × ProjectService.java ProjectDao.java

```

93     Project project = new Project();
94
95     project.setProjectName(projectName);
96     project.setEstimatedHours(estimatedHours);
97     project.setActualHours(actualHours);
98     project.setDifficulty(difficulty);
99     project.setNotes(notes);
100
101     Project dbProject = projectService.addProject(project);
102     System.out.println("You have successfully created project: " + dbProject);
103
104
105 }
106
107 private BigDecimal getDecimalInput(String prompt) {
108     String input = getStringInput(prompt);
109
110     if (Objects.isNull(input)) {
111         return null;
112     }
113     try {
114         return new BigDecimal(input).setScale(2);
115     }
116     catch (NumberFormatException e) {
117         throw new DbException(input + "is not a valid decimal number.");
118     }
119 }
120 private boolean exitMenu() {
121     System.out.println("Exiting the menu.");
122     return true;
123 }

```

```
ProjectsApp.java × ProjectService.java ProjectDao.java
125 private int getUserSelection() {
126     printOperations();
127
128     Integer input = getIntInput("Enter a menu selection");
129     return Objects.isNull(input)? -1 : input;
130 }
131
132
133 private Integer getIntInput(String prompt) {
134     String input = getStringInput(prompt);
135
136     if(Objects.isNull(input)) {
137         return null;
138     }
139
140     try {
141         return Integer.valueOf(input);
142     }
143     catch(NumberFormatException e) {
144         throw new DbException(input + " is not a valid number.");
145     }
146 }
147
148 private String getStringInput(String prompt) {
149     System.out.print(prompt + ": ");
150     String input = scanner.nextLine();
151
152     return input.isBlank()? null : input.trim();
153 }
154 }
155
```

```
ProjectsApp.java ProjectService.java ProjectDao.java ×
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(conn, projectId));
88     project.getSteps().addAll(fetchStepsForProject(conn, projectId));
89     project.getCategories().addAll(fetchCategoriesForProject(conn, projectId));
90 }
91
92 commitTransaction(conn);
93
94 return Optional.ofNullable(project);
95 }
96 catch (Exception e) {
97     rollbackTransaction(conn);
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 conn, Integer projectId) throws SQLException{
109     ...
110 }
```

```

140
141         try(ResultSet rs = stmt.executeQuery()){
142             List<Step> steps = new LinkedList<>();
143
144             while(rs.next()) {
145                 steps.add(extract(rs, Step.class));
146             }
147             return steps;
148         }
149     }
150 }
151
152 private List<Material> fetchMaterialsForProject
153 (Connection conn, Integer projectId) throws SQLException{
154     String sql = "SELECT * FROM " + MATERIAL_TABLE + " WHERE project_id = ?";
155
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

```

```

109     //@formatter:off
110     String sql = "
111         + "SELECT c.* FROM " + CATEGORY_TABLE + " c "
112         + "JOIN " + PROJECT_CATEGORY_TABLE + " pc USING (category_id) "
113         + "WHERE project_id = ?";
114     //@formatter:on
115
116     try(PreparedStatement stmt = conn.prepareStatement(sql)){
117         setParameter(stmt, 1, projectId, Integer.class);
118
119         try(ResultSet rs = stmt.executeQuery()){
120             List<Category> categories = new LinkedList<>();
121
122             while(rs.next()) {
123                 categories.add(extract(rs, Category.class));
124             }
125             return categories;
126         }
127     }
128 }
129
130 }
131
132
133
134 private List<Step> fetchStepsForProject
135 (Connection conn, Integer projectId) throws SQLException{
136     String sql = "SELECT * FROM " + STEP_TABLE + " WHERE project_id = ?";
137
138     try(PreparedStatement stmt = conn.prepareStatement(sql)){
139         setParameter(stmt, 1, projectId, Integer.class);
140

```

```
ProjectsApp.java  ProjectService.java  ProjectDao.java X
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);
63 }
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 conn = DbConnection.getConnection()){
72         startTransaction(conn);
73
74         try {
75             Project project = null;
76             try(PreparedStatement stmt = conn.prepareStatement(sql)){
77                 setParameter(stmt, 1, projectId, Integer.class);
78             }
```



```

ProjectsApp.java  ProjectService.java  ProjectDao.java X
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
156 private void printOperations() {
157     System.out.println("\nThese are the available selections. press the Enter key to quit:");
158
159     operations.forEach(line -> System.out.println(" " + line));
160
161     if(Objects.isNull(curProject)) {
162         System.out.println("\nYou are not working with a project.");
163     }
164     else {
165         System.out.println("\nYou are working with project: " + curProject);
166     }
167 }
168 }

```

```

ProjectsApp.java  ProjectService.java  *ProjectDao.java X
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                 }
191                 return projects;
192             }
193         }
194     }
195
196     catch (Exception e) {
197         rollbackTransaction(conn);
198         throw new DbException(e);
199     }
200 }
201
202
203
204
205
206 }
207
208
209
210

```

```

195
196     catch (Exception e) {
197         rollbackTransaction(conn);
198         throw new DbException(e);
199     }
200 }
201
202 catch(SQLException e) {
203     throw new DbException(e);
204 }
205 }
206 }
207
208
209
210

```

```
ProjectsApp.java | ProjectService.java | ProjectDao.java
1 package projects.service;
2
3 import java.util.List;
4
5
6
7
8
9
10 public class ProjectService {
11     private static final String SCHEMA_FILE = "PROJECT_SCHEMA.SQL";
12     private static final String DATA_FILE = "project_data.sql";
13     private ProjectDao projectDao = new ProjectDao();
14
15     public Project fetchProjectById(Integer projectId) throws Exception {
16         Optional<Project> op = projectDao.fetchProjectById(projectId);
17
18         return projectDao.fetchProjectById(projectId).orElseThrow( ()
19             -> new NoSuchElementException
20                 ("Project with Project ID=" + projectId
21                 + " does not exist."));
22     }
23
24
25
26     public Project addProject(Project project) {
27         return projectDao.insertProject(project);
28     }
29
30
31
32     public List<Project> fetchAllProjects() {
33
34         return projectDao.fetchAllProjects();
35     }
36 }
37
```

```
Console
ProjectsApp [Java Application] C:\Program Files\Java\jdk-17.0.3.1\bin\javaw.exe (Sep 2, 2022, 12:28:19 PM) [pid: 14568]

These are the available selections. press the Enter key to quit:
1) Create and populate all tables
2) List projects
3) Select a project

You are not working with a project.
Enter a menu selection: 3
Successfully obtained connection!

Projects:
1: Caulk around cabinets
Enter a project ID to select a project: 1
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

You are working with project:

ID=1
name=Caulk around cabinets
estimatedHours=3.00
actualHours=0.00
difficulty=2
notes=Keep cabinets clean
Materials:
ID=4, materialName=2-inch screws, numRequired=null, cost=7.99

Steps:

ID=2, stepText=Screw door hanger on top and bottom of each side of the door frame

Categories:

Enter a menu selection:

Console X

ProjectsApp [Java Application] C:\Program Files\Java\jdk-17.0.3.1\bin\javaw.exe (Aug 30, 2022, 12:52:42 PM) [pid: 5984]

Error: [projects.exception.DbException](#): list projects is not a valid number.Try again.

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

- 1) Create and populate all tables
- 2) List projects

Enter a menu selection: 2

Successfully obtained connection!

Projects:

- 1: Hang a door
- 2: project

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

- 1) Create and populate all tables
- 2) List projects

Enter a menu selection:

*<projects> Script-8 X

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

INSERT INTO project
(project_name, estimated_hours, actual_hours, difficulty, notes)
VALUES('Caulk around cabinets',3,0,2,'Keep cabinets clean');

INSERT INTO material
(project_id, material_name, num_required, cost)
VALUES(1,'2-inch screws',20,7.99);

INSERT INTO step
(project_id, step_text, step_order)
VALUES(1,'Screw door hanger on top and bottom of each side of the door frame',2);

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

<https://github.com/bmason1969/Week-10.git>