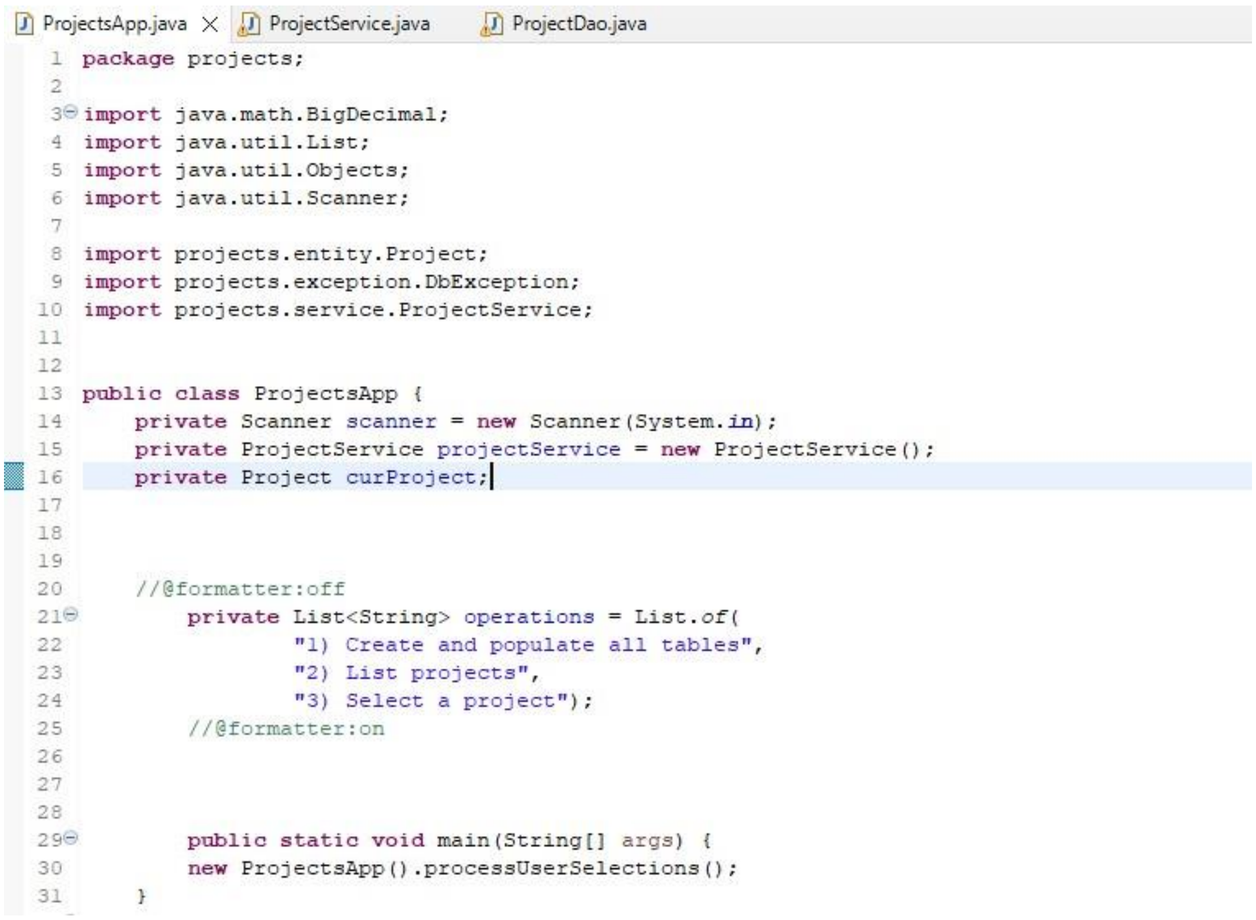
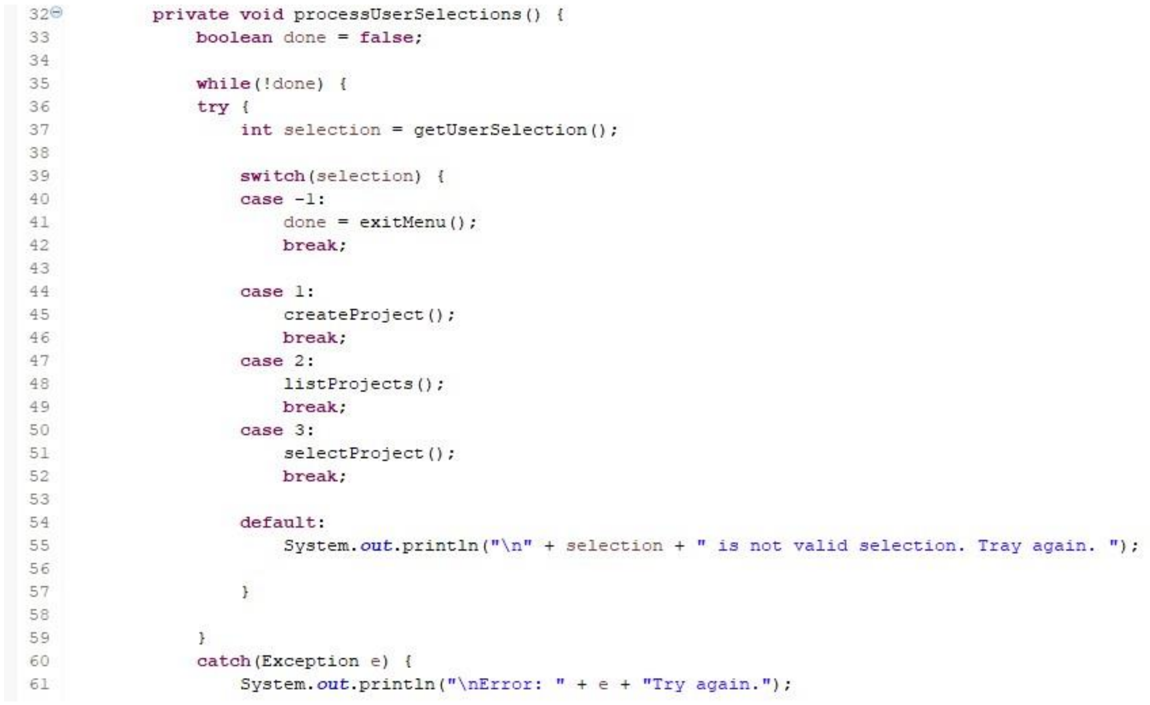
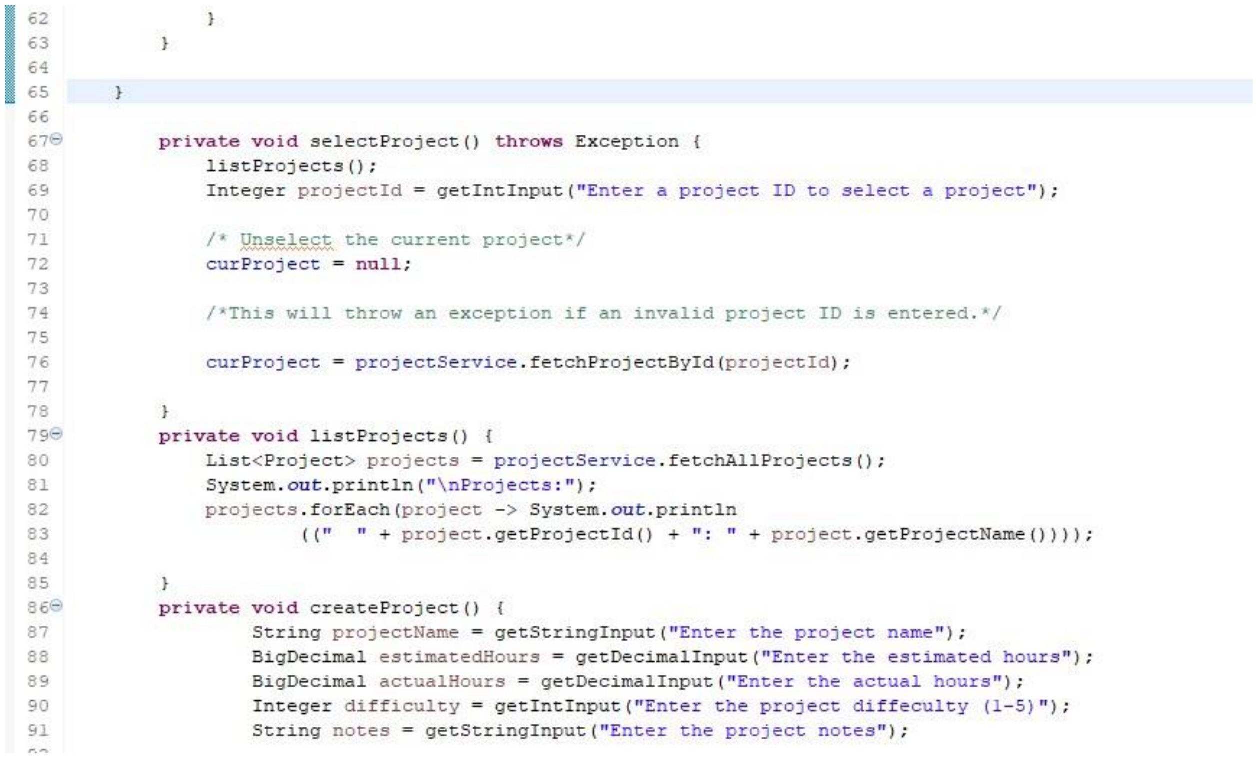


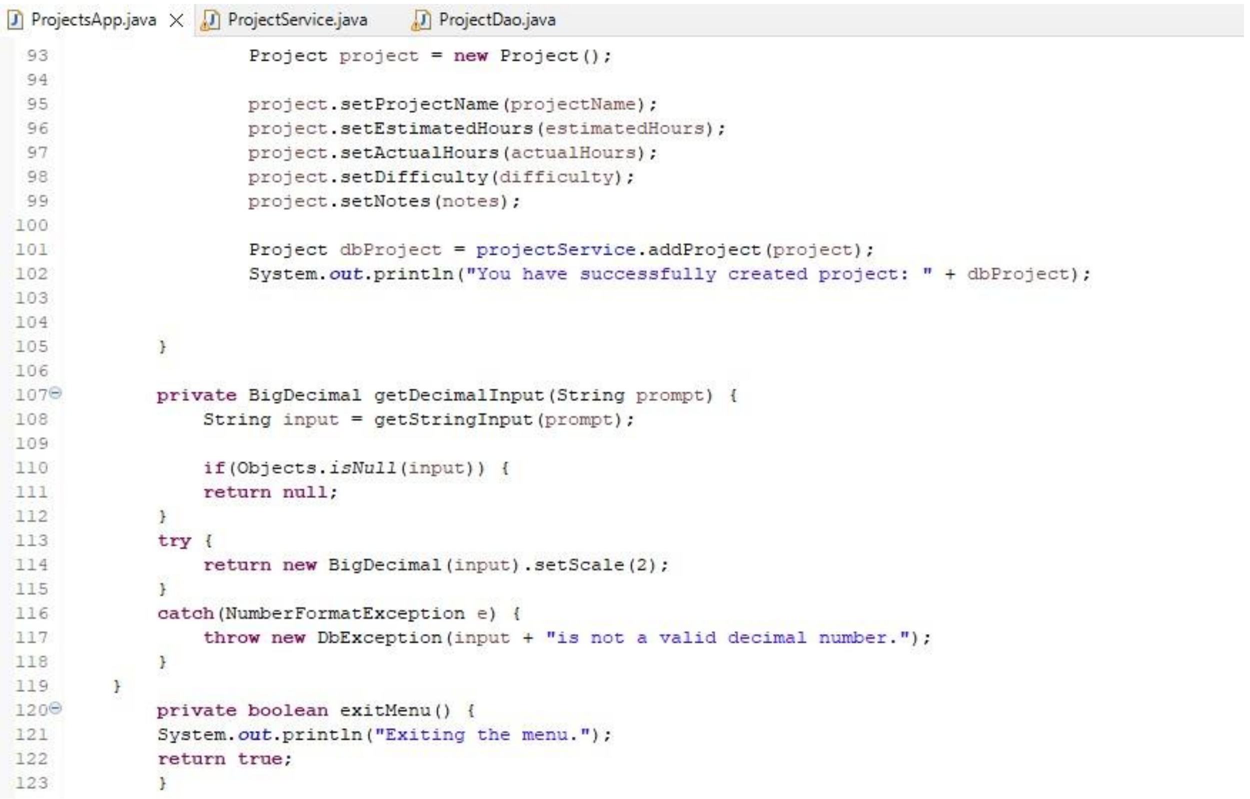
```java
package projects;

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;


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



    //@formatter:off
        private List<String> operations = List.of(
                "1) Create and populate all tables",
                "2) List projects",
                "3) Select a project");
        //@formatter:on



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



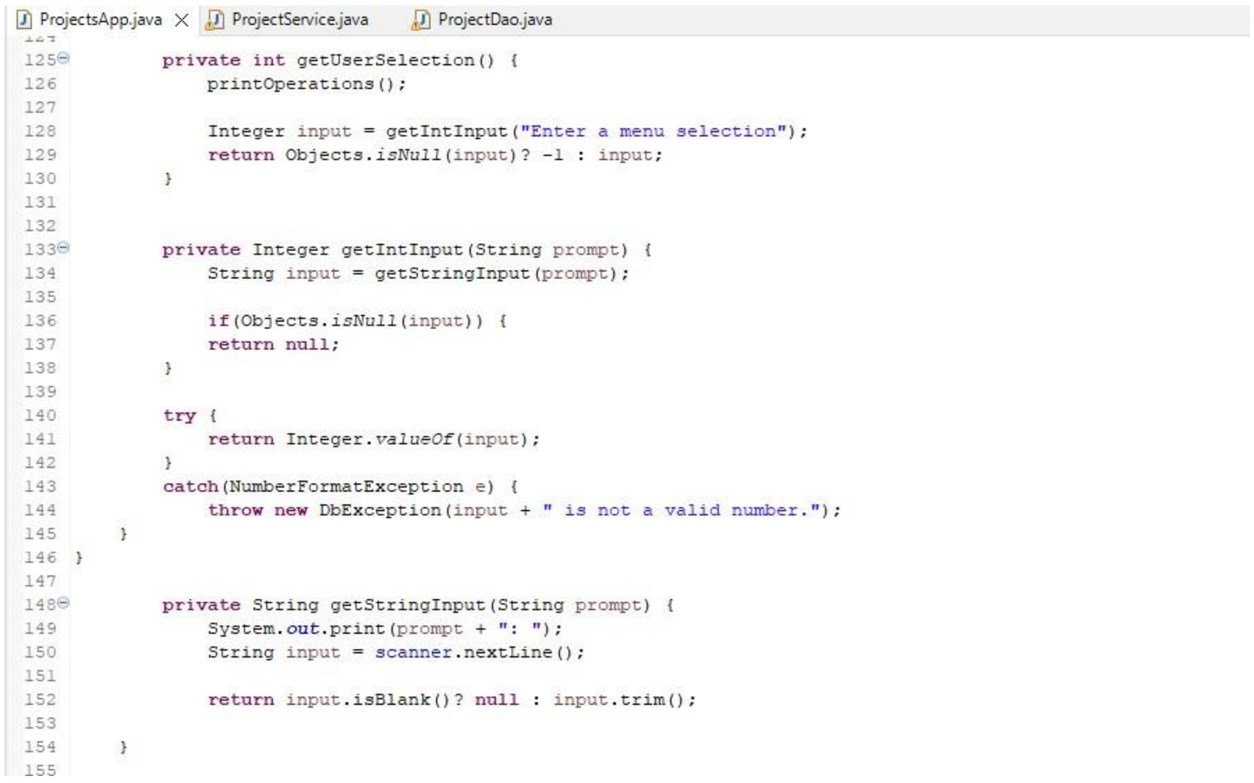
```java
        private void processUserSelections() {
            boolean done = false;

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

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

                case 1:
                    createProject();
                    break;
                case 2:
                    listProjects();
                    break;
                case 3:
                    selectProject();
                    break;

                default:
                    System.out.println("\n" + selection + " is not valid selection. Tray again. ");

                }

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



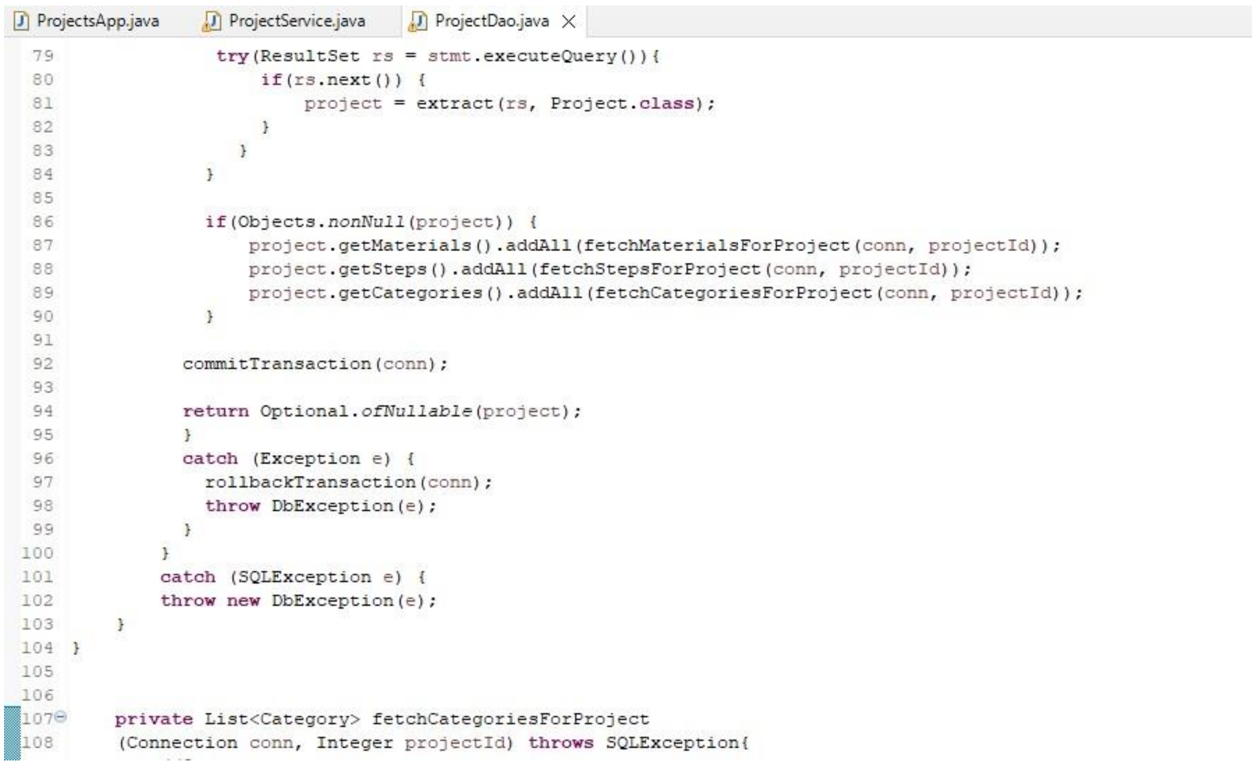
```
            }
        }

    }

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

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

            /*This will throw an exception if an invalid project ID is entered.*/

            curProject = projectService.fetchProjectById(projectId);

        }
        private void listProjects() {
            List<Project> projects = projectService.fetchAllProjects();
            System.out.println("\nProjects:");
            projects.forEach(project -> System.out.println
                    (("   " + project.getProjectId() + ": " + project.getProjectName())));

        }
        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 diffeculty (1-5)");
                String notes = getStringInput("Enter the project notes");
```



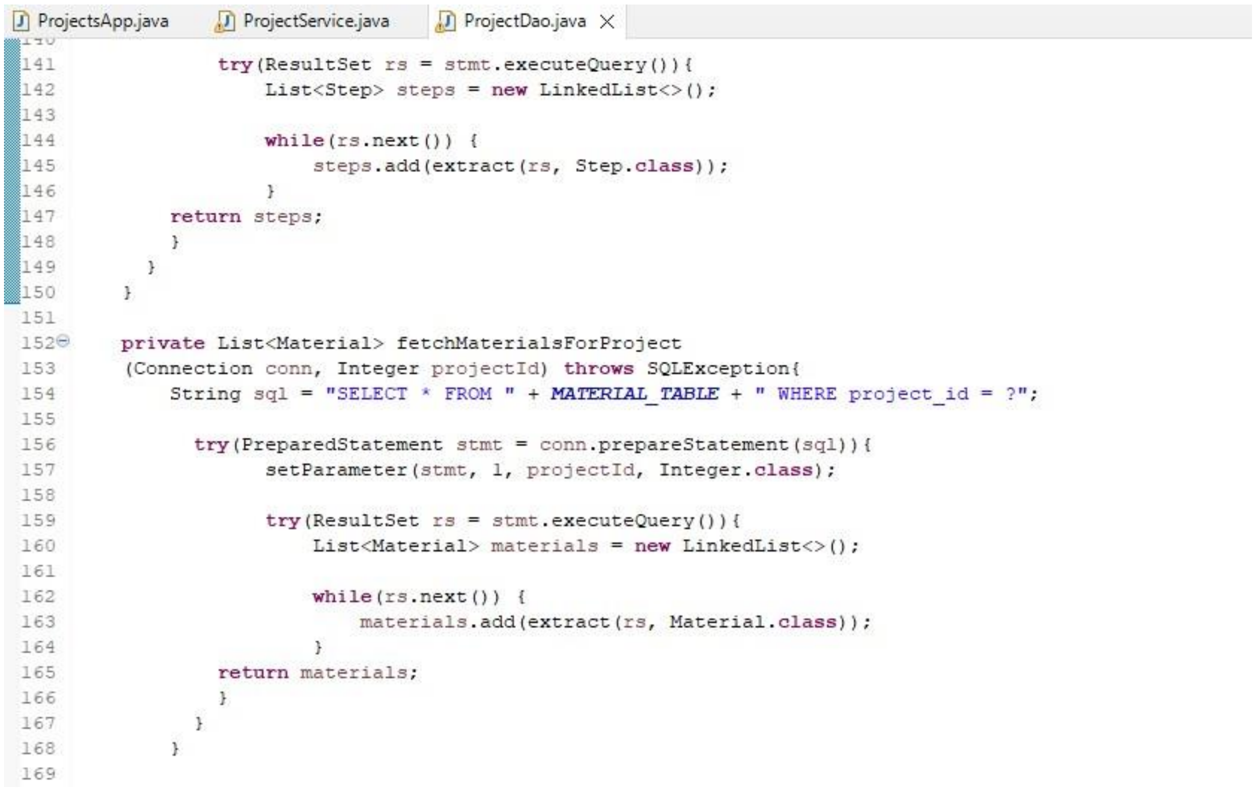
```
                Project project = new Project();

                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);


        }

        private BigDecimal getDecimalInput(String prompt) {
            String input = getStringInput(prompt);

            if(Objects.isNull(input)) {
            return null;
        }
        try {
            return new BigDecimal(input).setScale(2);
        }
        catch(NumberFormatException e) {
            throw new DbException(input + "is not a valid decimal number.");
        }
    }
        private boolean exitMenu() {
        System.out.println("Exiting the menu.");
        return true;
        }
```



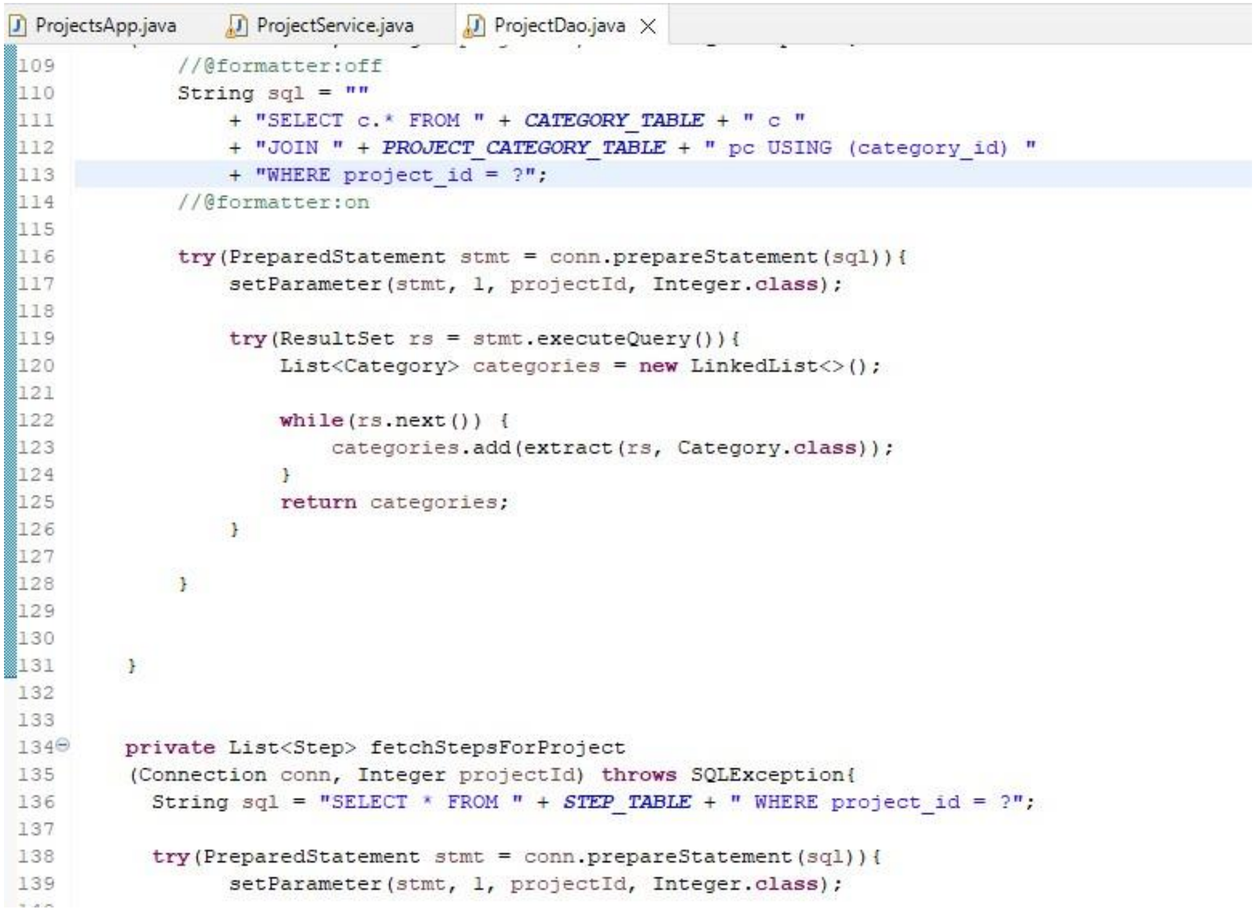
```java
        private int getUserSelection() {
            printOperations();

            Integer input = getIntInput("Enter a menu selection");
            return Objects.isNull(input)? -1 : input;
        }


        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.");
    }
}

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

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

    }

```



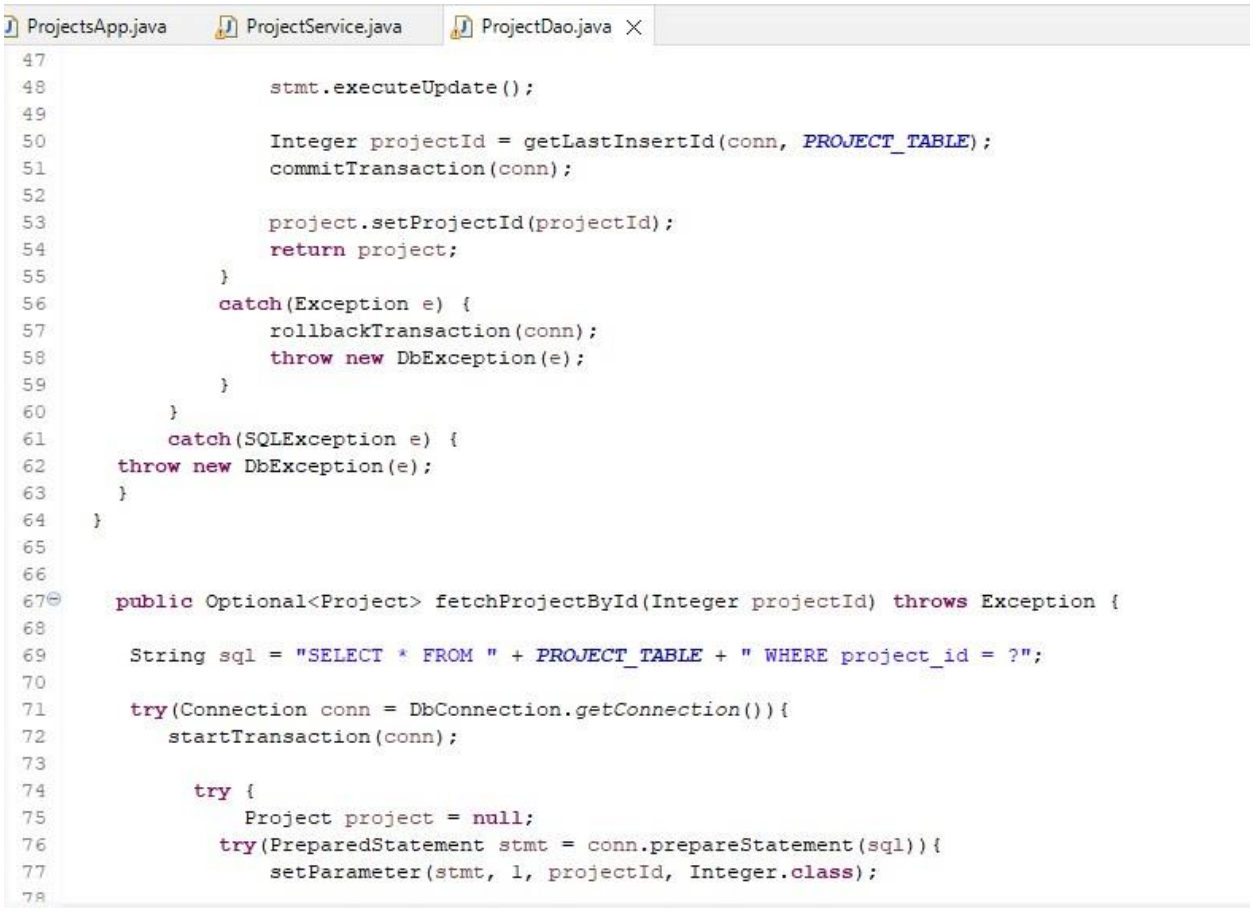
```java
              try(ResultSet rs = stmt.executeQuery()){
                  if(rs.next()) {
                      project = extract(rs, Project.class);
                  }
                }
             }

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

           commitTransaction(conn);

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


    private List<Category> fetchCategoriesForProject
    (Connection conn, Integer projectId) throws SQLException{
```



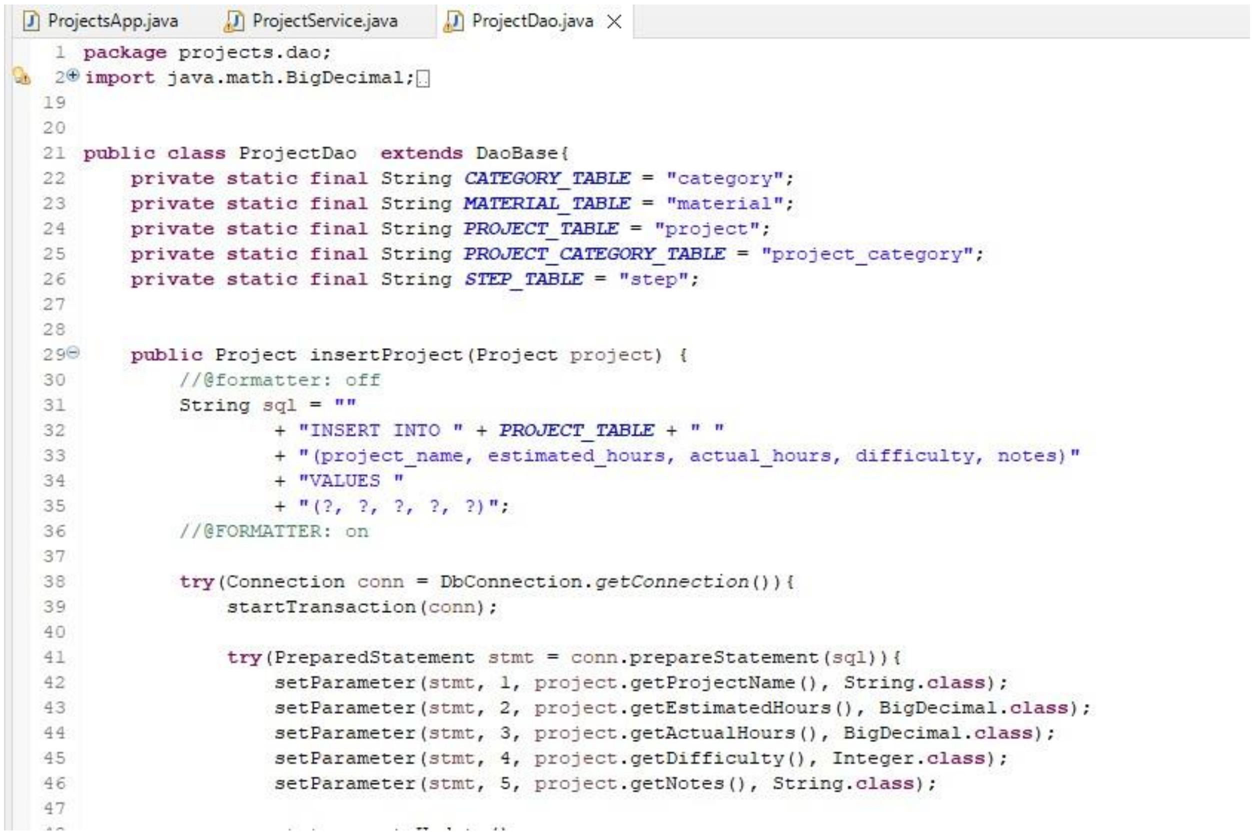
```
        try(ResultSet rs = stmt.executeQuery()){
            List<Step> steps = new LinkedList<>();

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

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

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

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

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

```



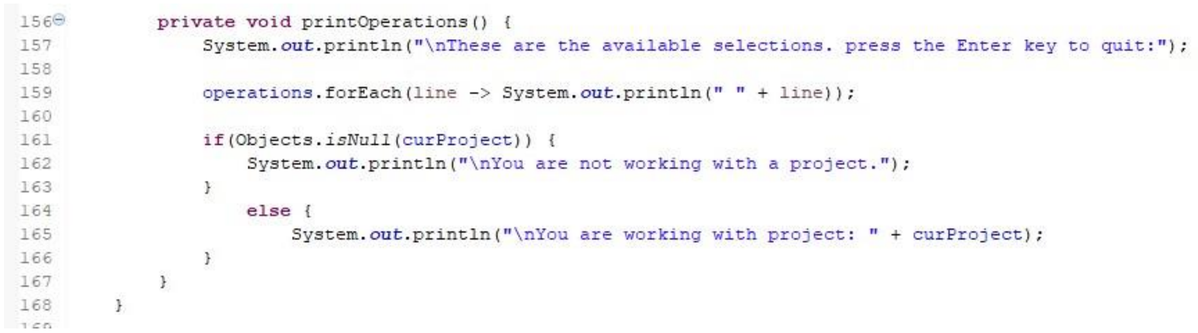
```
    //@formatter:off
    String sql = ""
        + "SELECT c.* FROM " + CATEGORY_TABLE + " c "
        + "JOIN " + PROJECT_CATEGORY_TABLE + " pc USING (category_id) "
        + "WHERE project_id = ?";
    //@formatter:on

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

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

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

    }


}


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

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


ProjectsApp.java ProjectService.java ProjectDao.java

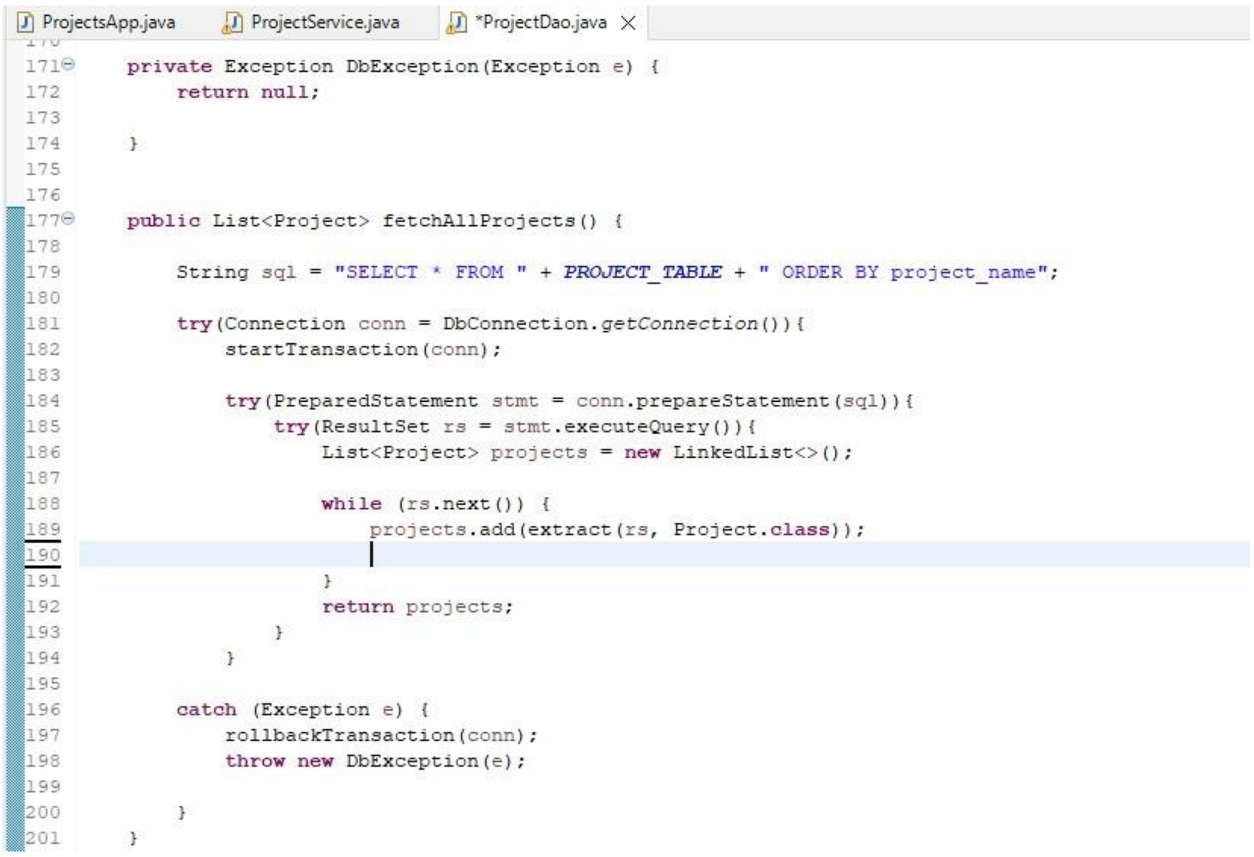
```
                stmt.executeUpdate();

                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);
    }
  }


    public Optional<Project> fetchProjectById(Integer projectId) throws Exception {

     String sql = "SELECT * FROM " + PROJECT_TABLE + " WHERE project_id = ?";

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

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



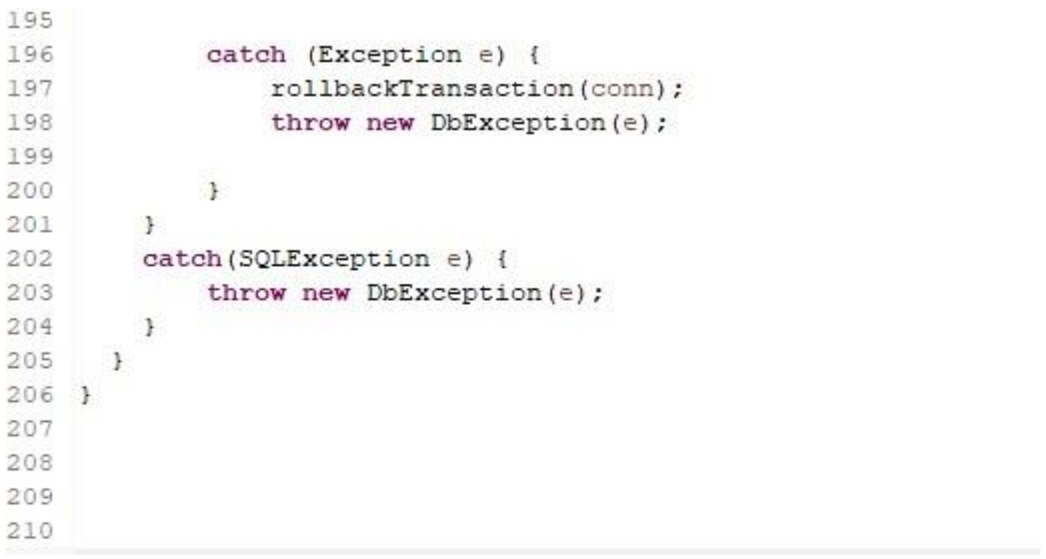
```java
package projects.dao;
import java.math.BigDecimal;


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";


    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);

```



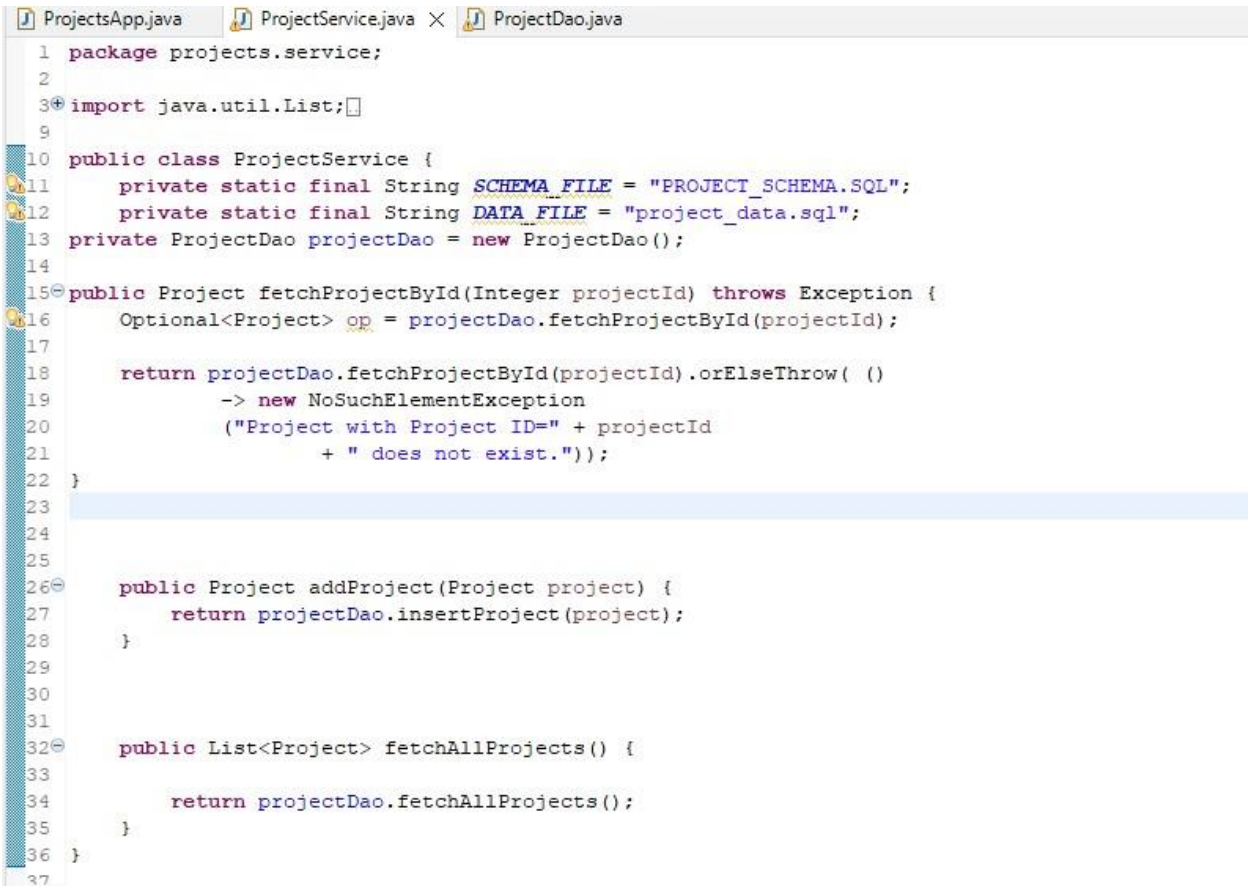
```java
        private void printOperations() {
            System.out.println("\nThese are the available selections. press the Enter key to quit:");

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

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



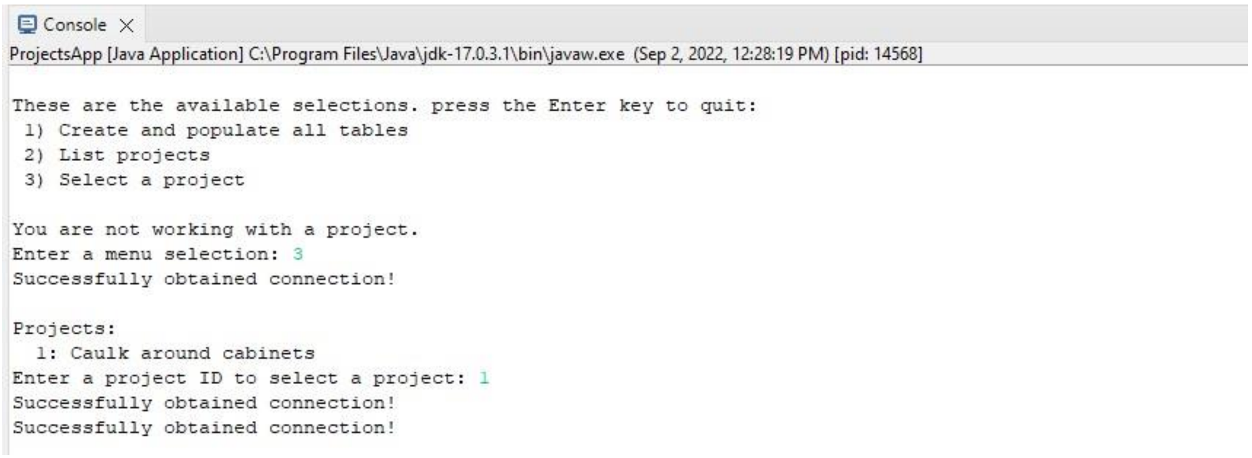
```java
    private Exception DbException(Exception e) {
        return null;

    }


    public List<Project> fetchAllProjects() {

        String sql = "SELECT * FROM " + PROJECT_TABLE + " ORDER BY project_name";

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

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

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

                    }
                    return projects;
                }
            }

        catch (Exception e) {
            rollbackTransaction(conn);
            throw new DbException(e);

        }
    }
```



```java

        catch (Exception e) {
            rollbackTransaction(conn);
            throw new DbException(e);

        }
    }
    catch(SQLException e) {
        throw new DbException(e);
    }
  }
}




```

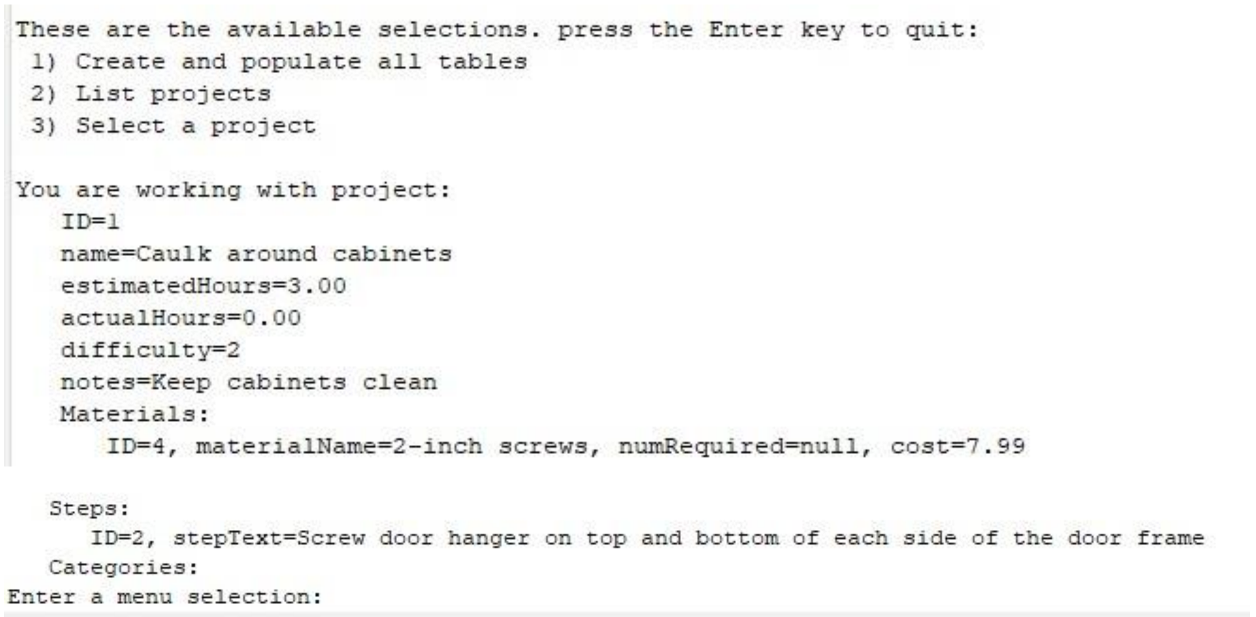


```java
package projects.service;

import java.util.List;

public class ProjectService {
    private static final String SCHEMA_FILE = "PROJECT_SCHEMA.SQL";
    private static final String DATA_FILE = "project_data.sql";
private ProjectDao projectDao = new ProjectDao();

public Project fetchProjectById(Integer projectId) throws Exception {
    Optional<Project> op = projectDao.fetchProjectById(projectId);

    return projectDao.fetchProjectById(projectId).orElseThrow( ()
            -> new NoSuchElementException
            ("Project with Project ID=" + projectId
                    + " does not exist."));
}



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



    public List<Project> fetchAllProjects() {

        return projectDao.fetchAllProjects();
    }
}
```

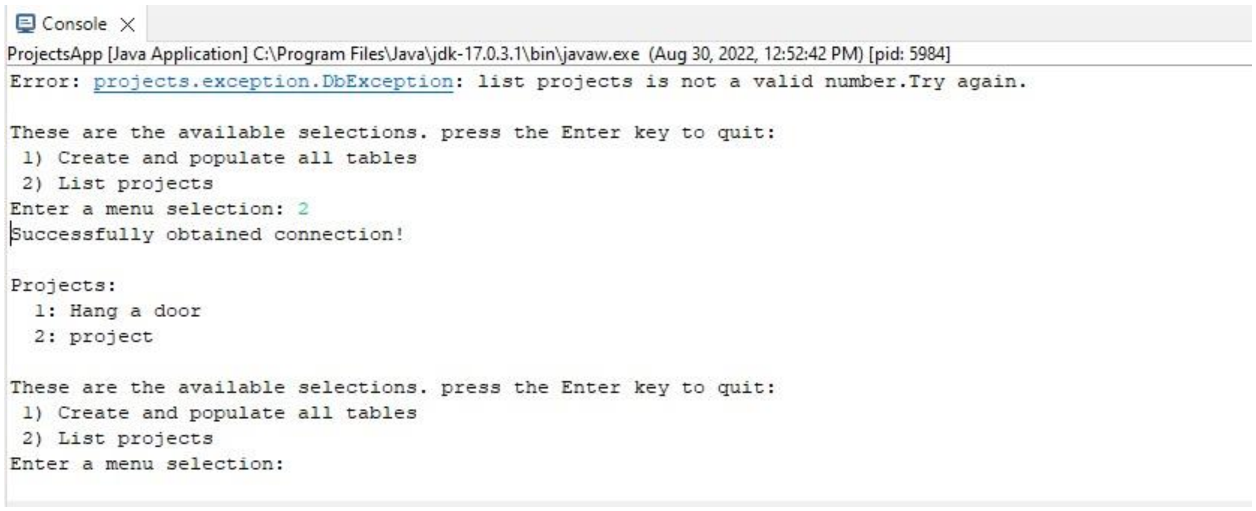


ProjectsApp [Java Application] C:\Program Files\Java\jdk-17.0.3.1\bin\javaw.exe (Sep 2, 2022, 12:28:19 PM) [pid: 14568]

```
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!
```



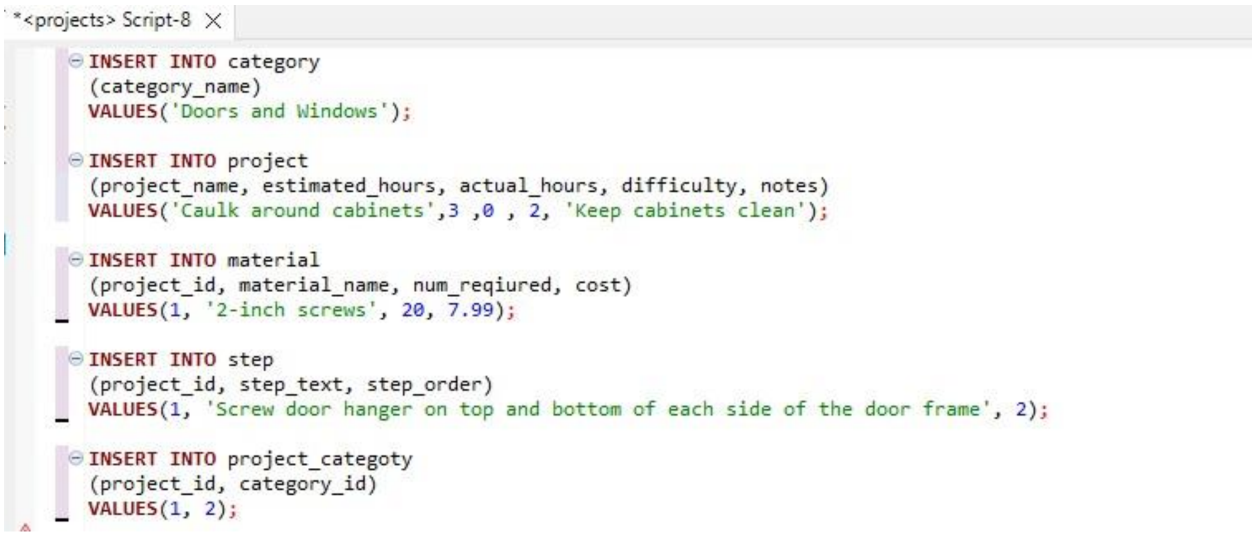
```
These are the available selections. press the Enter key to quit:
 1) Create and populate all tables
 2) List projects
 3) Select a project

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 ✕

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 ✕

```sql
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_reqiured, 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_categoty
(project_id, category_id)
VALUES(1, 2);
```

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