



```
again.");
        System.out.println(selection + " is not a valid option. Try
```

```
    }
    } catch (Exception e) {
        System.out.println("Error; " + e.toString() + " Try Again");
    }
}
}
```

```
private void selectProject() {
    listProjects();
    Integer projectId = getIntInput("Select a project id");

    curProject = null;

    curProject = projectServices.fetchProjectById(projectId);
```

```
}
```

```
private void listProjects() {

    List<Project> projects = projectServices.fetchAllProjects();

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

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

```
private void createProject() {
    // TODO Auto-generated method stub
    String name = getStringInput("Enter the project name");
    BigDecimal estimatedHours = getDecimalInput("Enter estimated hours");
    BigDecimal actualHours = getDecimalInput("Enter the actual hours");
    Integer difficulty = getIntInput("Enter the project difficulty 1-5");
    String notes = getStringInput("Enter project notes");
    Project project = new Project();

    project.setProjectName(name);
    project.setEstimatedHours(estimatedHours);
    project.setActualHours(actualHours);
    project.setDifficulty(difficulty);
    project.setNotes(notes);
    Project dbProject = projectServices.addProject(project);

}
```

```
private boolean exitMenu() {

    System.out.println("Exiting the menu");
    return true;
```

```

    }

    private int getUserSelections() {
        // TODO Auto-generated method stub
        PrintOperations();
        Integer input = getIntInput("Enter a menu selesction");
        return Objects.isNull(input) ? -1 : input;
    }

    private Integer getIntInput(String prompt) {
        // TODO Auto-generated method stub
        String input= getStringInput(prompt);

        if (Objects.isNull(input)) {
            return null;
        }
        try {
            return Integer.valueOf(input);
        }
        catch(NumberFormatException e) {
            throw new DbException(input + "This is not a valid option");
        }
    }

}

    private String getStringInput(String prompt) {
        // TODO Auto-generated method stub
        System.out.println(prompt + ": ");
        String line = scanner.nextLine();

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

}

    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 + "This is not a valid option");
        }
    }

}

    private void PrintOperations() {
        // TODO Auto-generated method stub

```

```

        System.out.println();
        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("You are working with project: " + curProject);
        }
    }
}

```

```

package projects.service;

import java.util.List;
import java.util.NoSuchElementException;
import java.util.Optional;

import projects.dao.ProjectDao;
import projects.entity.Project;
import provided.util.DaoBase;

```

```

public class ProjectServices extends DaoBase{

    private ProjectDao projectDao = new ProjectDao();

```

```

    public Project addProject(Project project) {
        // TODO Auto-generated method stub
        return projectDao.insertProject(project);
    }

```

```

    public List<Project> fetchAllProjects() {
        return projectDao.fetchAllProjects();
    }

```

```

    }

    public Project fetchProjectById Integer projectId {

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

package projects.dao;

import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Collection;
import java.util.LinkedList;
import java.util.List;
import java.util.Objects;
import java.util.Optional;

import provided.util.DaoBase;
import projects.entity.Category;
import projects.entity.Material;
import projects.entity.Project;
import projects.entity.Step;
import projects.exceptions.DbException;

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 {
        // TODO Auto-generated method stub
        //@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) {
// TODO Auto-generated catch block
        rollbackTransaction(conn);
        throw new DbException(e);
    }
}
catch (SQLException e) {
    throw new DbException(e);
}

}

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);
    }

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

    }

}

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

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

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

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

                if (Objects.nonNull(project)) {
                    project.getMaterials().addAll(fetchMaterialsForProject(conn, 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 new DbException(e);
        }
    }
}

```

```

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

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;
        }
    }

}

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);

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

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

}

private List<Category> fetchCategoriesForProject(Connection conn,
Integer projectId)
throws SQLException{
    // @formatter:off
    String sql = ""

```



```
(category_id) "
        + "SELECT c.* FROM " + CATEGORY_TABLE + " c "
        + "JOIN " + PROJECT_CATEGORY_TABLE + " pc USING
        + "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;
    }
}
```