```
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 projects.dao.exception.DbException;
import projects.entity.Category;
import projects.entity.Material;
import projects.entity.Project;
import projects.entity.Step;
import provided.util.DaoBase;

/**
 * This class uses JDBC to perform CRUD opeartions on the project tables.
 *
 * @author Candace Samuels
 *
 */
@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";



public static 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);
                }
            }

// WEEK 10: START
            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<Category> fetchCategoriesForProject(Connection conn, Integer projectId)
                throws  SQLException {
            // @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);

			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;
        }
    }
}
```

```
}
// WEEK 10: END
```