**About the Final Project**

1. **Main Classes of the program**
2. **Project and Project Task**

As introduced in the Final Project Proposal, this program is a task management system designed for companies that undertake construction projects. I have designed two Classes, Project and ProjectTask to simulate the real-word situation where each construction projects are comprised of different tasks. This “two-layer” design brings a lot of challenges for me, especially for the design of user interface and data storage; but the design that keep track of all information only through one Class, Project, really does not align with the real-word situation, so I chose to implement this “two-layer” design.

In addition to essential information such as projectName, taskName, department, etc., for each Project and ProjectTask, there are two additional fields: isCompleted and isAborted. These two fields record the status of each Project and ProjectTask. My design is as follows:

* Each ProjectTask can be marked as isCompleted or isAborted directly;
* Projects can also be marked as isCompleted or isAborted. But a Project cannot be marked as isCompleted if any of its ProjectTask is not completed and not aborted either. A Project can be marked as isAborted even if there are still pending ProjectTasks; in such case, all its corresponding ProjectTasks will be automatically marked as isAborted.
* (Note: in this program, a Project will not be automatically marked as isCompleted if all of its corresponding ProjectTasks are completed. If the last ProjectTask of a Project is completed, the user needs to mark the ProjectTask as isCompleted, then mark the whole Project as isCompleted.)

ProjectTasks also record projectValue and dueDate.

1. **RiskCodeAssessor**

The RiskCodeAssessor is designed to assess risk codes of overdue ProjectTasks. If a ProjectTask is overdue, its risk level will be assessed based on its projectValue and dueDate.

1. **Staff**

Staff is an abstract Class which is extended by Employee, Manager and GeneralManager. The abstract Class Staff has two methods, displayAlerts() and displayAllOngoingProjects(); the implementation of these methods varies depending on the position of each Staff:

* For Employee, all overdue tasks with YELLOW, ORANGE or RED RiskCode will be displayed as alerts, and Employee only has access to information regarding his or her Projects;
* For Manager, only tasks with ORANGE or RED RiskCode will be displayed as alerts, and Manager has access to information regarding Projects in his or her department;
* For GeneralManager, only tasks with RED RiskCode will be displayed as alerts, and Manager has access to information regarding all Projects in the company.

GengeralManger has two additional methods: displayProjectsByStaffID(int staffID) which displays all projects of a Staff, and displayProjectsByDepartment(Department department) which displays all projects of a Department.

1. **Administrator**

In this program, the Administrator has the authority to create and register Staff, Projects and ProjectTasks.

I have included a user interface for registering Projects and ProjectTasks in the program. Ideally, the program should also contain a user interface for registering Staff; but to avoid over-complicating the project, I have directly coded 8 Staff in the program: four Employees, three Managers and one GeneralManager. The createAndStoreStaff() method creates these 8 Staff instances and stores their information into the database, and this method will be automatically called when the program is started. In other words, we assume that the company has 8 staff and their information will be in the system once the program is started.

1. **StaffAndProjectManagementService**

The StaffAndProjectManagementService Class connects the above-mentioned main parts of the program with data storing service. This approach follows the principle of separation of concerns and help to achieve better code organization, maintainability and testability.

1. **User Interfaces**
2. **UI0LoginPage**

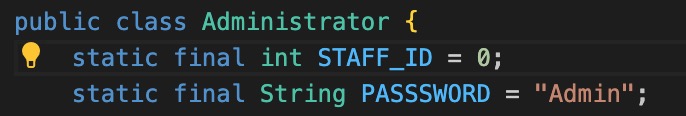
UI0LoginPage is the login page.

To log in the system as Staff, the staffIDs, passwords and the information of the 8 staff created can be found in the createAndStoreStaff() method of Administrator:

A screen shot of a computer screen

Description automatically generated

To log in as the Administrator, please use the following staffID and password:



1. **UI1AdminPage**

If the user logs in the program as the Administrator, he/she will be directed to the AdminPage, where the user can register Projects and related ProjectTasks, look up Projects and ProjectTasks, and change their status.

1. **UI2UserPage**

If the user logs in the program as Staff, he/she will be directed to the corresponding UserPage.

The UserPage contains a text area that will automatically invoke the displayAlerts() method, and displays alerts about delayed ProjectTasks whenever the Staff logs in.

The UserPage of all Staff contains the option to display all ongoing projects, which corresponds to the displayAllOngoingProjects() method. The UserPage of GeneralManager contains two more options to display projects by Departments and display projects by staffID which correspond to the displayProjectsByStaffID(int staffID) and displayProjectsByDepartment(Department department) methods respectively.

1. **Data Storing**

The **DataStoringService** Class is used for storing, updating and retrieving information from the database.

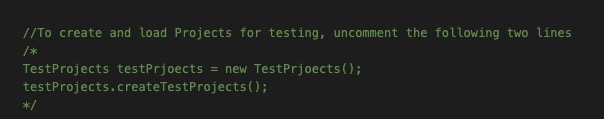
I used SQLite to store information, because due to the design where each Project is comprised of ProjectTasks, it is easier to use projectID as a foreign key in the table that stores ProjectTasks to connect Projects and ProjectTasks.

To utilize the built-in SQLite of Java, the JDBC “sqlite-jdbc-3.42.0.0.jar” needs to be added to the classpath. I am using VSCode, so in my case, I added the jar file into the libraries folder.

1. **Test Projects**

Outside the folder of project files, there is a separate file named “TestProjects.java”. This file is not part of the program, but includes test Projects and ProjectTasks that I used to test the program. I also included this file in the submission for your reference, so that you don’t need to be bothered to come up with various tests for the program. I also included the Excel spreadsheet named “Information of Test Projects and Tasks”, which might be helpful in understanding the test Projects and ProjectTasks.

In the “Main.java” of the program, I have left some lines commented:



If the last few lines are uncommented and the TestProjects.java file is included in the “scr” folder, the test Projects and ProjectTasks will be created and stored after these two lines are executed.

(Note: if the TestProjects.java file is used for testing, please execute the TestProjects.createTestProjects() before using the AdminPage to register Projects, because the program automatically sets up the counters for Projects and ProjectTasks, thus the projectIDs and taskIDs are determined by the order in which the projects and tasks are created.)

1. **Potential Improvements**

Considering the scope of the project, I have simplified some real-world scenarios. The project currently has some limitations and potential areas for improvement include:

* Enhanced encapsulation can be achieved by setting some relevant fields of different Classes as private and adding getters and setters.
* I simplified the relationships between various Classes. For example, now I assume that the company is a small company and each department only has one Manager. Therefore, the responsible person for each Project is an Employee. If the General Manager wants to look up the projects of Employees, he/she can look up the projects by staffID; if the General Manager wants to look up the projects of Managers, he/she can look up the projects by Department. In reality, the situation might be more complicated; for example, ideally there should be an option to look up projects by the staffID of Managers as well.
* I have implemented necessary error handling logic to address errors such as incorrect number format and invalid user ID etc. But more “checks” can help prevent other errors. In theory, it is still possible to input erroneous information. For example, Staff No. 1 is from Department.ROAD\_PROJECT. Currently, it is still possible to register a project for Staff No. 1 but with another Department. In the future, I will try to implement more “checks” to prevent this kind of errors.
* Ideally, there should be more types of ProjectTasks, and the RiskCodeAssessor should take into account more factors in addition to dueDate and projectValue.
* Ideally, there should be more options available for users. For example, currently, the Employees and Managers can check ongoing projects; and the General Manager has the options to view all projects and tasks of an Employee and that of one Department. But in reality, all users should have the option to choose between viewing only ongoing projects or all projects, and have the flexibility to define customized selection criteria with greater freedom.

Nonetheless, I have tried to make the project scalable; it should be easy to implement the above-mentioned improvements. I will try to improve the project progressively in the future.

Thank you for reading the introduction and reviewing my project!