Hyperiondev

TASK

Capstone Project I — Databases

Visit our website

Introduction

WELCOME TO THE FIRST CAPSTONE PROJECT!

In this task, you will be modifying your previous Capstone Project so that the program now uses a database, instead of text files, to store the data needed for the program. You will be tasked with designing and implementing a database. You will also write the code needed to interact with the database.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

DEVELOPER PORTFOLIO

By the end of this project, you will be adding this program to your developer portfolio. Today most applications rely heavily on data! A prospective employer will want to see that you can persist and manipulate data. This project should make it clear to a potential employer that you can follow a development process to create a data-driven program that is debugged, tested, refactored and documented and that meets a client's specifications! Put extra time and effort into this project to make sure that it really showcases the skills you have acquired!

THE TASK AT HAND

You have been asked to create a project management system for a small structural engineering firm called "Poised". Poised does the engineering needed to ensure the structural integrity of various buildings. They want you to create a Java program that they can use to keep track of the many projects on which they work.

Poised stores the following information for each project that they work on:

- Project number.
- Project name.
- What type of building is being designed? E.g. House, apartment block or store, etc.
- The physical address for the project.
- ERF number.
- The total fee being charged for the project.
- The total amount paid to date.
- Deadline for the project.
- The name, telephone number, email address and physical address of the architect for the project.
- The name, telephone number, email address and physical address of the contractor for the project.
- The name, telephone number, email address and physical address of the customer for the project.

Poised wants to be able to use your program to do the following:

- Capture information about new projects. If a project name is not provided when the information is captured, name the project using the surname of the customer. For example, a house being built by Mike Tyson would be called "House Tyson" and an apartment block owned by Jared Goldman would be called "Apartment Goldman".
- Update information about existing projects. Information may need to be adjusted at different stages throughout the lifecycle of a project. For example, the deadline might change after a meeting with various stakeholders.

- Finalise existing projects. When a project is finalised, the following should happen:
    - An invoice should be generated for the client. This invoice should contain the customer's contact details and the total amount that the customer must still pay. This amount is calculated by subtracting the total amount paid to date from the total fee for the project. If the customer has already paid the full fee, an invoice should not be generated.
    - The project should be marked as "finalised" and the completion date should be added. All the information about the project should be added to a text file called "Completed project".
- See a list of projects that still need to be completed.
- See a list of projects that are past the due date.
- Find and select a project by entering either the project number or project name.

Before you begin

A key focus of this project will be ensuring that your code is correct, well-formatted and readable. In this regard, make sure that you do the following before submitting your work:

1. Make sure that your code is readable. To ensure this, add comments to your code, use descriptive variable names and make good use of whitespace and indentation. See **this style guide** to see how classes and methods should be named and how your program should be formatted.

2. Make sure that your code is as efficient as possible. How you choose to write code to create the solution to the specified problem is up to you. However, make sure that you write your code as efficiently as possible.

3. Make sure that all output that your program provides to the user is easy to read and understand. Labelling all data that you output (whether in text files or to the screen) is essential to make the data your program produces more user-friendly.

Compulsory Task 1

Follow these steps:

- Design and create a database called PoisePMS. Assume that each project can only be assigned to one Structural Engineer. Each project will also only have one Project Manager, one Architect and one customer.

  Submit the following:

  - Dependency diagrams for each table in the database.
  - An ERD that shows the relationships between the tables in your database.
  - Screenshots of your console that show how each table was created.

- Add at least two rows of data to each table in the database. Submit screenshots of your console that show how data is added to the tables.

Compulsory Task 2

Follow these steps:

- Copy and paste the code that you wrote for the last Capstone Project in the previous level of this Bootcamp into the Dropbox folder for this Capstone Project.

- Modify your code so that it:
  - Reads and writes data about projects and people associated with projects from your database instead of text files. Your program should not use any text files.
  - Capture information about new projects and add these to the database.
  - Update information about existing projects.
  - Finalise existing projects. When a project is finalised the following should happen:
    - An invoice should be generated for the client. This invoice should contain the customer's contact details and the total amount that the customer must still pay. This amount is calculated by subtracting the total amount paid to date from the total fee for the project. If the customer has already paid the full fee, an invoice should not be generated.
    - The project should be marked as "finalised" and the completion date should be added.

    - Find all projects that still need to be completed from the database.
    - Find all projects that are past the due date from the database.
    - Find and select a project by entering either the project number or project name.

- Besides meeting the above criteria, you should also do the following:
    - Include exception handling. Use try-catch blocks wherever appropriate.
    - Remove all errors from your code. Take extra care to detect and remove all logical and runtime errors.
    - Adequately refactor your code.
    - Document your code. Adhere to the style guide found **here**.
    - Use Javadoc to generate API documentation from documentation comments for your program.
    - Follow the guidelines **here** to create a Readme file for this project.

- After receiving feedback from your mentor and improving your code based on this feedback, add your program to Github.

Rate us

Share your thoughts

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.