TASK

Capstone Project III - OOP

Visit our website

Introduction

WELCOME TO THE THIRD CAPSTONE PROJECT!

Welcome to your third Capstone Project! Well done on getting this far! This Capstone is a milestone in your learning so far. In this project, you will be using object-oriented programming to create a solution for a real-world problem. Remember, it is worth putting some extra time and effort into this project. It will eventually become part of your developer portfolio.

Get in touch

Connect for support

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

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

The best way to get help is to login to Discord at **https://discord.com/invite/hyperdev** to start a chat with a code reviewer, as well as share your thoughts and problems with peers. You can also schedule a call or get support via email.

DEVELOPER PORTFOLIO

Developers who have the edge are those who find ways to apply their newfound skills from the get-go. A developer portfolio (a collection of online creations that you have made) allows you to demonstrate your skills rather than just telling people about them. It's a way of bringing your CV to life and introducing yourself to the world. As you learn more skills and put these into practice, each project that you complete will become more efficient and eye-catching. Object-oriented programming is one of the most important programming paradigms today! Prospective employers will want evidence that a Web Developer is comfortable using object-oriented programming. This application series offers you the means to develop an object-oriented program to add to your developer portfolio.

Compulsory Task 1

In this task, we're going to be simulating an email message. Some of the logic has been filled out for you in the **email.js** file.

- Open the file called **email.js**.
- Create a class definition for an `Email` which has four variables: `hasBeenRead`, `emailContents`, `isSpam` and `fromAddress`.
- The constructor should initialise the sender's email address.
- The constructor should also initialise `hasBeenRead` and `isSpam` to false.
- Create a function in this class called `markAsRead` which should change `hasBeenRead` to true.
- Create a function in this class called `markAsSpam` which should change `isSpam` to true.
- Create a list called `inbox` to store all emails (note that you can have a list of objects).
- Then create the following methods:
    - `addEmail` - which takes in the contents and email address from the received email to make a new `Email` object.
    - `getCount` - returns the number of messages in the store.
    - `getEmail` - returns the contents of an email in the list. For this, allow the user to input an index i.e. `getEmail(i) method` returns the email stored at position `i` in the list. Once this has been done, `hasBeenRead` should now be true.
    - `getUnreadEmails` - should return a list of all the emails that haven't been read.
    - `getSpamEmails` - should return a list of all the emails that have been marked as spam.
    - `delete` - deletes an email in the inbox.

Now that you have these set up, let's get everything working!

Fill in the rest of the logic for what should happen when the user inputs send/read/quit. Some of it has been done for you.

Rate us

Share your thoughts

HyperionDev 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.