**Documentation**

GITHUB REPOSITORY LINK = <https://github.com/HARISH40356/phase1.git>

**Core java concepts used:**

* Opps concepts like class, object, polymorphism, abstraction, encapsulation.
* Control statements if, if else, switch, for, while, do while
* Validation using equals().
* Use of different access modifiers.
* Scanner class, assignment operators.
* List, ArrayList to store camera list.
* toString() method.

**Description:**

**Package:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Main package** | **App name** | **Sub package** | **classes** | **Description** |
| com | camerarental | bean | Camera | Camera details |
| service | CameraService | Camera service methods add, remove, view all, my wallet and rent camera. |
| cemerarental | main | App | Camera rental application starting point it contains main method. |
| utility | CameraRentalUtility | Camera rental application utility methods login, main menu option, sub menu option, close app. |

**Class:**

|  |  |  |
| --- | --- | --- |
| **class** | **method** | **use** |
| App | main | main class print welcome message and call login method |
| CameraRentalUtility | login, mainOption, subMenuOption, closeApp | This class contains switch case and all method calls  and closeApp method implementation. |
| Camera | getter and setter, default and parameter constructor, toString | Camera details will set and get from this class by using setter, getter, constructors and toString method. |
| CameraService | addCamera, removeCamera, viewAllCamera, myWallet, rentCamera | AppCamera main functionalities are implemented in this class. |

SPRINT CYCLE

➢ Planning

➢ Execution

➢ Review

➢ Retrospective

* Planning

**Algorithm:**

1. Start the application.
2. Initialize the camera list and wallet.
3. Prompt the user to log in with a username and password.
4. Validate the username and password. If invalid, go back to step 3.
5. If the login is successful, display the main menu with options.
6. Based on the user's input, perform the corresponding action:

• If the user selects "MY CAMERA", go to step 7.

• If the user selects "RENT A CAMERA", go to step 8.

• If the user selects "VIEW ALL CAMERAS", go to step 9.

• If the user selects "MY WALLET", go to step 10.

• If the user selects "EXIT", end the application.

1. Handle the "MY CAMERA" functionality:

• Display options: Add, Remove, View My Cameras, Go to Previous Menu.

• Based on the user's input, perform the corresponding action:

• If the user selects "ADD", go to step 11.

• If the user selects "REMOVE", go to step 12.

• If the user selects "VIEW MY CAMERAS", go to step 13.

• If the user selects "GO TO PREVIOUS MENU", go back to step 6.

1. Handle the "RENT A CAMERA" functionality:

• Display the list of available cameras.

• Prompt the user to enter the camera ID to rent.

• If the camera ID is valid, check if the camera is available and the user has sufficient balance in the wallet.

• If both conditions are met, deduct the rental amount from the wallet, mark the camera as rented, and display a success message.

• If any condition fails, display an error message.

• Go back to step 6.

1. Handle the "VIEW ALL CAMERAS" functionality:

• Display the list of all cameras (rented and available).

• Go back to step 6.

1. Handle the "MY WALLET" functionality:

• Display the current wallet balance.

• Prompt the user if they want to deposit more money.

• If yes, prompt for the amount and deposit it into the wallet.

• Go back to step 6.

1. Handle the "ADD" functionality under "MY CAMERA":

• Prompt the user to enter the camera brand, model, and price per day.

• Create a new Camera object with an auto-incremented ID and the provided details.

• Add the camera to the camera list.

• Go back to step 7.

1. Handle the "REMOVE" functionality under "MY CAMERA":

• Display the list of cameras.

• Prompt the user to enter the camera ID to remove.

• Remove the camera with the corresponding ID from the camera list.

• Handle cases where an undesirable value for ID, i.e., other the integer is entered.

• Go back to step 7.

1. Handle the "VIEW MY CAMERAS" functionality under "MY CAMERA":

• Display the list of cameras that are added by the user.

• Go back to step 7.

* Execution

➢ Performing Tasks

➢ Implementing modules

Performing Tasks:

To perform a task that helps the user to

➢ Add

➢ Remove

➢ View

➢ Rent a camera

➢ Deposit money to their wallet

Implementing modules

➢ Camera management

➢ User management

➢ Rental management

➢ Reporting module

Camera management:

▪ This module would handle functionalities related to adding, removing, and modifying camera details like brand, model, price, and availability.

▪ It would also handle searching for specific camera

User management :

▪ This module manage the user profile

Rental management:

▪ This module would handle the process of renting and returning cameras.

Reporting module:

▪ This module would provide reports on camera is available status.

* Review:

Aim:

To create a camerarental application.

Demonstration check:

The application is running as expected

* Retrospective:

If needed we can update modules in the future

**Flowchart:**

View all cameras

Rent camera

Exit

My camera

Print welcome message

Print wrong credentials

Read main option

Read username and password

Username, password correct

Read and add amount

My wallet

Print all cameras

Add

Read details

Insufficient balance

Read id

Id not found

Remove

Id found

Read id

Add camera

Read main option

Rent

Id found

Print all cameras

Main menu

Remove camera

View all cameras