CAMERA RENTAL APPLICATION

Algorithm :

 Start the Camera Rental Application.

Display the welcome screen with the application name and developer details.

Prompt the user to select an option to access the features.

If the user selects "List Available Cameras":

a. Retrieve the list of available cameras from the database.

b. Display the list of cameras along with their brand, model, and per-day rental amount.

c. Go back to the main menu.

If the user selects "Rent a Camera":

a. Retrieve the list of available cameras from the database.

b. Display the list of cameras along with their brand, model, and per-day rental amount

c. Prompt the user to select a camera to rent by entering the camera ID.

d. Validate if the camera ID is valid and if the camera is available for rent.

e. If valid, prompt the user to enter the rental duration.

f. Calculate the rental cost based on the per-day rental amount and duration.

g. Deduct the rental amount from the user's wallet balance.

h. Update the status of the rented camera to "Rented".

i. Display a success message confirming the rental transaction.

j. Go back to the main menu.

If the user selects "Manage Wallet":

a. Display the current wallet balance to the user.

b. Prompt the user if they want to deposit more money into the wallet.

c. If yes, prompt the user to enter the deposit amount.

d. Update the wallet balance with the deposited amount.

e. Display a success message confirming the wallet balance update.

f. Go back to the main menu.

If the user selects "Exit":

a. Display a goodbye message.

b. End the Camera Rental Application.

Handle invalid input from the user by displaying an error message and prompting for valid input.

Repeat steps 3-8 until the user chooses to exit the application

Product Overview:

The Camera Rental Application is a Java-based application that allows users to rent cameras, manage their rented cameras, view available cameras, and manage their wallet balance. The application provides a user-friendly interface for users to interact with and perform various operations related to camera rental.

Product Capabilities:

1. User Login:
   * Users can log in using their credentials to authenticate themselves.
2. Camera Listing:
   * The application displays a list of available cameras for rental.
   * Each camera is listed with its brand, model, and per-day rental amount.
   * Users can view all the cameras and their details.
3. Camera Rental:
   * Users can select a camera from the listing to rent.
   * Users can specify the rental duration and complete the rental transaction.
   * The application calculates the rental cost based on the selected camera and duration.
   * The rental transaction is recorded and associated with the user's account.
4. Wallet Management:
   * Users can add or view the amount in their wallet.
   * The application displays the current balance in the user's wallet.
   * Users can deposit additional funds into their wallet if desired.
5. Navigation and Context:
   * Users can navigate between different sections of the application.
   * The main context serves as the starting point and provides access to all the functionalities.
   * Users can return to the main context from any other execution context.
   * Users can close the application when they have finished using it.
6. Search and Sort:
   * Users can search for cameras by their ID, brand, or model.
   * The application provides search functionality to find specific cameras quickly.
   * Users can sort the camera listing based on ID, brand, model, or price.
   * Sorting allows users to view the cameras in a desired order.
7. Enhanced Prototype Tracking:
   * The application code is stored and tracked using Git and a GitHub account.
   * Enhancements and modifications to the prototype are managed through version control.
   * The code repository allows for collaboration and easier management of the application's development.

Appearance and User Interactions:

The application will have a text-based interface where the user interacts with the application through command-line input and output. The user will be prompted with instructions and input prompts, and they can enter their choices or provide information accordingly. The application will provide appropriate feedback and display the requested information or perform the requested operations.

Menu Navigation:

* The main context of the application can present a menu with options for users to choose from.
* The menu can be displayed in a clear and organized manner, making it easy for users to navigate through different sections of the application.

Camera Listing:

* When displaying the list of available cameras, a tabular format can be used to present the camera details.
* Each camera entry can be structured with columns for ID, brand, model, price, and status.
* Clear headers and proper alignment can make the information easy to read and understand.

Application Flow:

1. User Login:
   * The user is prompted to enter their username and password to log into the camera rental application.
2. Main Menu:
   * After successful login, the user is presented with a main menu displaying different options to choose from.
   * The user can select one of the following options:
     + My Camera: Allows the user to manage their rented cameras.
     + Rent a Camera: Allows the user to browse and rent available cameras.
     + View All Cameras: Displays a list of all available cameras.
     + My Wallet: Allows the user to manage their wallet balance.
     + Exit: Terminates the application.
3. My Camera:
   * If the user selects "My Camera" from the main menu, they are taken to the My Camera submenu.
   * In this submenu, the user can perform the following actions:
     + Add Camera: Allows the user to add a new camera to their collection.
     + Remove Camera: Lets the user remove a camera from their collection.
     + View My Cameras: Displays a list of cameras the user currently owns.
     + Go to Previous Menu: Takes the user back to the main menu.
4. Rent a Camera:
   * If the user chooses "Rent a Camera" from the main menu, they are taken to the Rent a Camera section.
   * Here, the user can browse through the available cameras and select one to rent.
   * After selecting a camera, the user is prompted to confirm the rental and complete the transaction.
5. View All Cameras:
   * When the user selects "View All Cameras" from the main menu, a list of all available cameras is displayed.
   * The list includes details such as camera ID, brand, model, price, and status.
   * The user can browse through the list to view the available cameras.
6. My Wallet:
   * If the user chooses "My Wallet" from the main menu, they are taken to the My Wallet section.
   * Here, the user can view their current wallet balance and choose to deposit more money if desired.
7. Exit:
   * If the user selects "Exit" from the main menu, the application terminates, and the user session ends.

Number and Duration of Sprints:

Sprint 1 (1 day):

* User login functionality
* Main menu implementation
* Basic camera listing and display
* My Camera functionality (add, remove, view)
* Basic wallet functionality (view balance)
* Complete camera listing and display
* Rent a Camera functionality
* View All Cameras functionality
* Enhanced wallet functionality (deposit, view)

Sprint 2 (10 hours):

* Refinement of existing features
* Bug fixes and improvements

Sprint 2 (3 hours):

* Documentation and final touches
* Testing and bug fuxing

Set Up Git and GitHub account to store and track enhancements of the prototypes:

Implementation of Java Concepts in Camera Rental Application:

1. Data Structures:
   * List: A list data structure is used to store the collection of camera objects. It allows dynamic resizing and provides methods to add, remove, and access elements efficiently.
   * Map: A map data structure can be used to store user login credentials, where the username is the key and the password is the value. It provides efficient retrieval of values based on the keys.
   * Sorting: Sorting algorithms such as bubble sort, or merge sort can be implemented to sort the camera list based on specific criteria such as ID, brand, or price.
   * Searching: Searching algorithms like binary search can be used to find a specific camera based on its ID or other attributes.
2. Object-Oriented Programming:
   * Encapsulation: The Camera class encapsulates the properties (ID, brand, model, price, status) and provides getter and setter methods to access and modify the values.
   * Inheritance: If there are different types of cameras, you can create subclasses (e.g., DSLR, Mirrorless) that inherit from the Camera class, allowing code reuse and specialization.
   * Polymorphism: Polymorphism can be utilized if there are specific operations or behaviors associated with different types of cameras. For example, you can define abstract methods in the Camera class and provide different implementations in its subclasses.
3. User Input and Output:
   * The Scanner class can be used to handle user input from the console, allowing users to enter login credentials, menu choices, camera details, etc.
   * System.out.println statements can be used to provide feedback and display relevant information to the user.

By incorporating these Java concepts and data structures, the Camera Rental Application can be developed to effectively manage and manipulate camera data, handle user interactions, and provide a seamless user experience.

Camera Rental Application Operations:

1. List a Camera:
   * Display the list of available cameras along with their brand, model, and per-day rental amount.
   * Provide the user with an option to view all cameras.
2. Select a Camera to Rent:
   * Present the user with a list of available cameras.
   * Allow the user to select a camera by entering the camera ID or choosing from a menu.
   * Check if the camera is available for rent.
   * If available, prompt the user for the rental duration.
   * Calculate the rental cost based on the per-day rental amount and duration.
   * Deduct the rental amount from the user's wallet balance.
   * Update the status of the rented camera to "Rented."
3. Add or View Wallet Amount:
   * Provide the user with an option to view their current wallet balance.
   * Give the user the ability to add funds to their wallet by entering the deposit amount.
4. Navigate to Main Context:
   * Allow the user to return to the main menu or previous context from any operation.
   * Provide an option to go back to the main menu when viewing cameras, renting a camera, or managing the wallet.
5. Close the Application:
   * Offer an option to exit the application gracefully.
   * Handle any necessary cleanup tasks before terminating the program.

These operations provide the core functionalities of the Camera Rental Application, allowing users to list available cameras, rent cameras, manage their wallet balance, navigate between different contexts, and close the application when done.