**Application Flow**

1. The program starts in the `main` method, where the `displayWelcomeScreen` function is called to show a welcome message.

2. The `login` function is called to handle the login process. It prompts the user to enter a username and password and checks if the credentials are valid by calling the `authenticateUser` function. If the user is successfully authenticated, the user is logged in, and the appropriate main menu is shown based on the user's role (admin or user).

3. The `authenticateUser` function checks the provided username and password against predefined values ("admin"/"pass" for admin and "user"/"password" for a regular user) and sets the `loggedInUser` and `loggedInUserRole` variables accordingly. If the authentication is successful, it returns `true`; otherwise, it returns `false`.

4. If the user is logged in as an admin, the `showAdminMainMenu` function is called. It presents the admin menu with options to add a camera, remove a camera, view available cameras, logout, or close the application. Based on the user's choice, the corresponding functions are called.

5. If the user is logged in as a regular user, the `showUserMainMenu` function is called. It presents the user menu with options to view available cameras, rent a camera, manage the wallet balance, logout, or close the application. Based on the user's choice, the corresponding functions are called.

6. The `addCamera` function allows the admin to add a new camera to the `cameraList`. It prompts the admin to enter the camera's brand, model, and rental amount, and then creates a new `Camera` object with the provided information. The new camera is added to the `cameraList`.

7. The `removeCamera` function allows the admin to remove a camera from the `cameraList`. It displays the available cameras and prompts the admin to choose a camera to remove. The selected camera is then removed from the `cameraList`.

8. The `displayAvailableCameras` function displays the available cameras from the `cameraList`.

9. The `rentCamera` function allows the user to rent a camera. It displays the available cameras and prompts the user to choose a camera to rent. If the selected camera is available and the user's wallet balance is sufficient, the camera's availability is updated, and the rental amount is deducted from the user's wallet balance.

10. The `manageWalletBalance` function allows the user to manage their wallet balance. It presents options to add an amount to the wallet or view the current balance. Based on the user's choice, the corresponding functions are called.

11. The `addWalletAmount` function allows the user to add an amount to their wallet balance. It prompts the user to enter the amount and adds it to the current wallet balance.

12. The `viewWalletAmount` function displays the current wallet balance.

13. The `closeApplication` function is called when the user chooses to close the application. It displays a closing message and exits the program.

14. The `logout` function allows the user to log out. It sets the `loggedIn` flag to `false`, clears the `loggedInUser` and `loggedInUserRole` variables, displays a logout message, and calls the `login` function to allow another user to log in.

15. The `getUserChoice` function is a utility function that prompts the user to enter a choice and validates it to ensure it is within the valid range.

16. The `getPositiveDoubleInput` function is a utility function that prompts the user to enter a positive double value and validates the input.

17. The `Camera` class is a simple class that represents a camera object. It has attributes like brand, model, rental amount, and availability. It also has getter and setter methods for these attributes, as well as an overridden `toString` method for displaying camera information.