Camera Rental Application

Writeups

Project Specifications

1. The Camera Rental Application allows users to rent cameras and manage their rental transactions.

2. Users can log in to the application using a username and password.

3. The application supports the following features:

    - **Rent a camera:** Users can view a list of available cameras and rent a camera by providing the camera ID. The rental amount is deducted from the user's wallet balance.
    - **View rented cameras:** Users can view a list of cameras they have rented.
    - **Add a camera:** Users can add a new camera to the available camera list.
    - **Remove a camera:** Users can remove a camera from the available camera list.
    - **View all cameras:** Users can view a list of all cameras, including both available and rented cameras.
    - **Wallet balance:** Users can check their wallet balance, which stores the amount of money they have available for renting cameras. They can deposit amount when they need.

4. The application uses an ArrayList to store and access camera data.

5. Each **camera** has the following attributes:

    - **ID:** A unique identifier for the camera.
    - **Brand:** The brand of the camera.
    - **Model:** The model of the camera.
    - **Price per day:** The rental price per day for the camera.
    - **Availability status:** Indicates whether the camera is available for rent or already rented.

6. The application ensures data integrity and consistency by validating user inputs, such as camera IDs, wallet balance, and login credentials.

7. The application handles various possible kinds of exceptions using try-catch blocks.

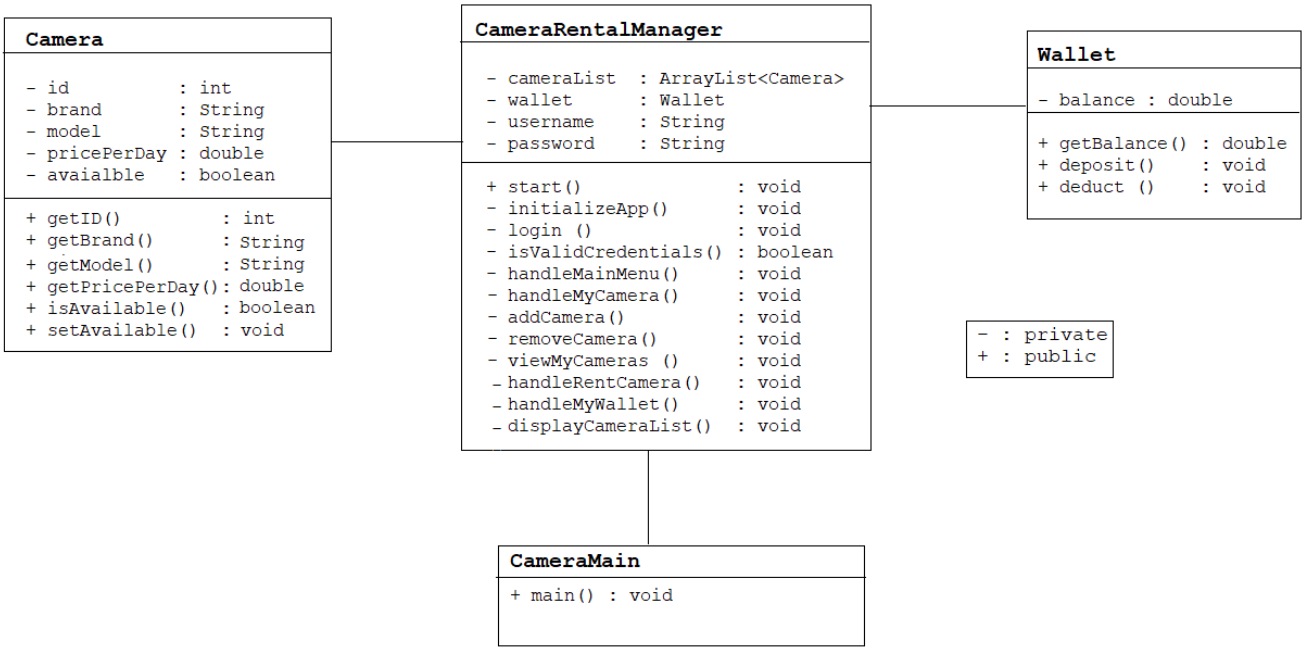
8. The application provides a user-friendly command-line interface for interacting with the features.

(Set username = admin, password = password)

Exceptions Handled

1. When the user enters an invalid, non – integer ID for camera
2. When the user enters an invalid, non – integer or integers other than provided choices as input for choice
3. When the user enters an invalid amount.

Class Diagrams:



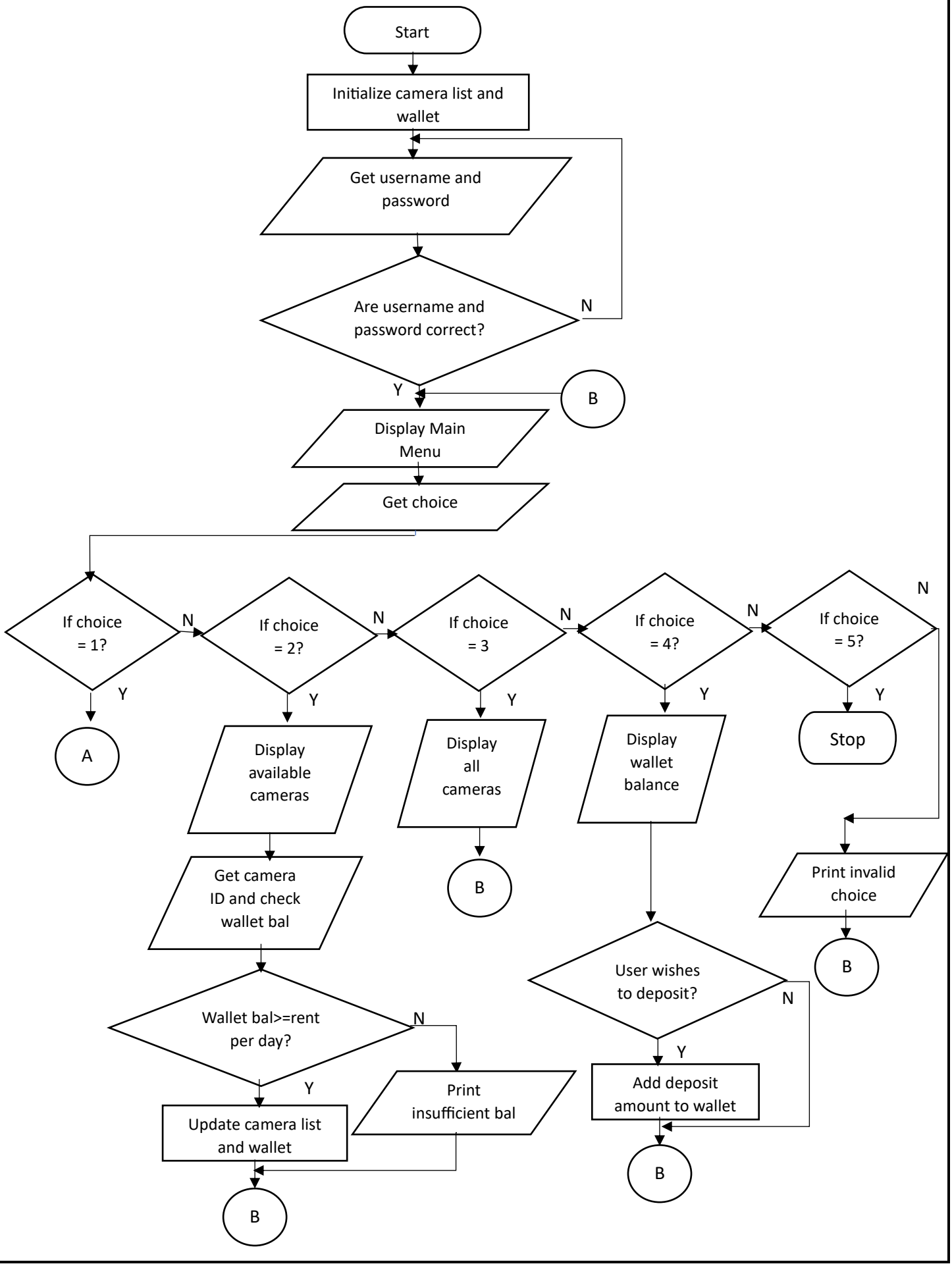
Algorithm:

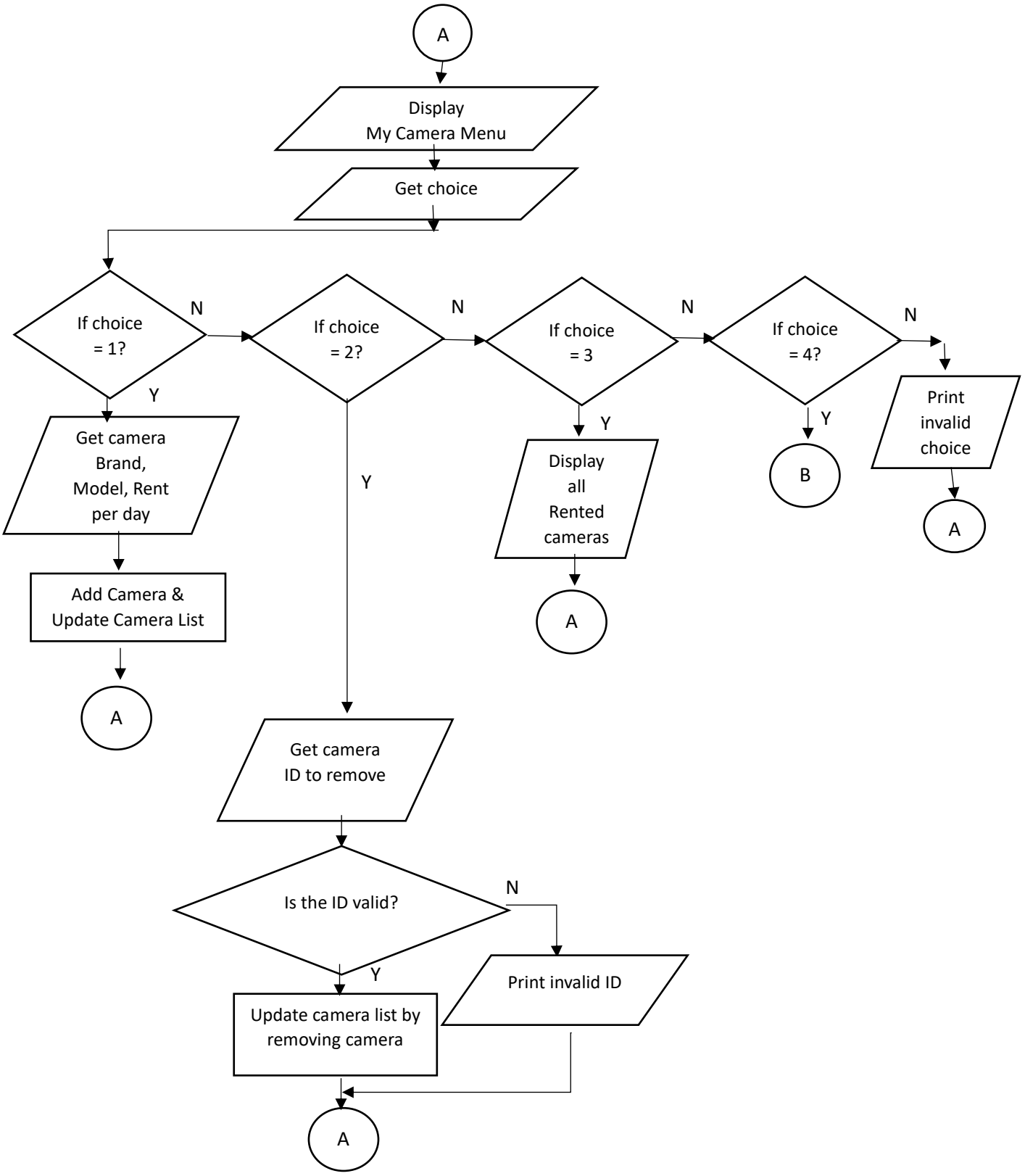
1. Start the application.
2. Initialize the camera list and wallet.
3. Prompt the user to log in with a username and password.
4. Validate the username and password. If invalid, go back to step 3.
5. If the login is successful, display the main menu with options.

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.
7. 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.
8. 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.
9. Handle the "VIEW ALL CAMERAS" functionality:
    - Display the list of all cameras (rented and available).
    - Go back to step 6.
10. 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.

11. 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.
12. 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.
13. 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.

Flowchart:


Start
Initialize camera list and wallet
Get username and password
Are username and password correct?
N
Y
B
Display Main Menu
Get choice
If choice = 1?
N
Y
A
If choice = 2?
N
Y
Display available cameras
Get camera ID and check wallet bal
Wallet bal>=rent per day?
N
Y
Update camera list and wallet
B
Print insufficient bal
B
If choice = 3
N
Y
Display all cameras
B
If choice = 4?
N
Y
Display wallet balance
User wishes to deposit?
N
Y
Add deposit amount to wallet
B
If choice = 5?
N
Y
Stop
Print invalid choice
B


A
Display
My Camera Menu
Get choice
If choice
= 1?
N
If choice
= 2?
N
If choice
= 3
N
If choice
= 4?
N
Print
invalid
choice
A
Y
Get camera
Brand,
Model, Rent
per day
Add Camera &
Update Camera List
A
Y
Get camera
ID to remove
Is the ID valid?
N
Print invalid ID
Y
Update camera list by
removing camera
A
Y
Display
all
Rented
cameras
A
Y
B