Phase-End Project

Project Agenda: Build a peer-to-peer camera rental application

Scenario:
You have been hired by a company called rentmycam.io as a Full Stack Developer with the aim to create a prototype of a camera rental application

Source Code:

```
package AssesmentPhase1Project;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

class Camera implements Comparable<Camera> {
    private int cameraId;
    private String brand;
    private String model;
    private double pricePerDay;
    private boolean isRented;

    public Camera(int cameraId, String brand, String model, double pricePerDay) {
        this.cameraId = cameraId;
        this.brand = brand;
        this.model = model;
        this.pricePerDay = pricePerDay;
        this.isRented = false;
    }
//getter and setter methods
    public int getCameraId() {
        return cameraId;
    }

    public String getBrand() {
        return brand;
    }

    public String getModel() {
        return model;
    }

    public double getPricePerDay() {
        return pricePerDay;
    }

    public boolean isRented() {
        return isRented;
    }
    public void setRented(boolean rented) {
        isRented = rented;
    }

    @Override
    public String toString() {
        return String.format("Camera ID: %d, Brand: %s, Model: %s, Price per Day:
%.2f, Status: %s",
                cameraId, brand, model, pricePerDay, isRented ? "Rented" :
"Available");
    }

    @Override
   public int compareTo(Camera other) {

      return Integer.compare(this.cameraId, other.cameraId);
    }
}

public class CameraRentalApp {
    private static ArrayList<Camera> cameraList = new ArrayList<>();
    private static double walletBalance = 0.0; // Initial wallet balance

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        boolean isLoggedIn = login(scanner);

        if (!isLoggedIn) {
            System.out.println("Login failed. Exiting the application.");
            return;
        }

        int choice;
        do {
            displayMenu();
            choice = scanner.nextInt();
            try {
            switch (choice) {
                case 1:
                    handleMyCamera(scanner);
                    break;
                case 2:
                    handleRentCamera(scanner);
                    break;
                case 3:
                    handleViewAllCameras();
                    break;
                case 4:
                    handleMyWallet(scanner);
                    break;
                case 5:
                    System.out.println("Exiting the application. Goodbye!");
                    break;
                default:
```

```java
                    System.out.println("Invalid choice. Please try again.");
            }
            }catch (Exception e) {
                System.out.println("An error occurred: " + e.getMessage());
                scanner.nextLine(); // Consume remaining input
            }
        } while (choice != 5);
    }

    private static boolean login(Scanner scanner) {
        System.out.println("+----------------------------------------------------+");
        System.out.println("|        WELCOME TO CAMERA RENTAL APP                |");
        System.out.println("+----------------------------------------------------+");
        System.out.println("PLEASE LOGIN TO CONTINUE");

        System.out.print("USERNAME: ");
        String username = scanner.next();

        System.out.print("PASSWORD: ");
        String password = scanner.next();

        return username.equals("admin") && password.equals("admin123");
    }

    private static void displayMenu() {

        System.out.println("1. MY CAMERA");
        System.out.println("2. RENT A CAMERA");
        System.out.println("3. VIEW ALL CAMERAS");
        System.out.println("4. MY WALLET");
        System.out.println("5. EXIT");
        System.out.println();
        System.out.print("Enter your choice: ");
    }

    private static void handleMyCamera(Scanner scanner) {
        int subChoice;
        do {
            displayMyCameraMenu();
            subChoice = scanner.nextInt();
            switch (subChoice) {
                case 1:
                    addCamera(scanner);
                    break;
                case 2:
                    removeCamera(scanner);
                    break;
                case 3:
                    viewMyCameras();
                    break;
                case 4:
                    System.out.println("Returning to the previous menu.");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (subChoice != 4);
    }

    private static void displayMyCameraMenu() {
        System.out.println("+----------------------------------------------------+");
        System.out.println("|                   MY CAMERA MENU                   |");
        System.out.println("+----------------------------------------------------+");
        System.out.println("1. ADD");
        System.out.println("2. REMOVE");
        System.out.println("3. VIEW MY CAMERAS");
        System.out.println("4. GO TO PREVIOUS MENU");
        System.out.println();
        System.out.print("Enter your choice: ");
    }

    private static void addCamera(Scanner scanner) {
        System.out.print("Enter the camera brand: ");
        String brand = scanner.next();

        System.out.print("Enter the model: ");
        String model = scanner.next();

        System.out.print("Enter the per day price (INR): ");
        double pricePerDay = scanner.nextDouble();

        int cameraId = cameraList.size() + 1;
        Camera camera = new Camera(cameraId, brand, model, pricePerDay);
        cameraList.add(camera);

        System.out.println("Your camera has been added successfully.");
    }
//Linear search  is used to find a specific camera by its ID
//It iterates through the cameraList and compares each camera's ID with the specified
ID until a match is found or the end of the list is reached.
    private static void removeCamera(Scanner scanner) {
       System.out.println("+----------------------------------------------------+");
        System.out.println("|              AVAILABLE CAMERAS LIST                |");
        System.out.println("+----------------------------------------------------+");

        // Print table header
        System.out.printf("| %-10s | %-15s | %-20s | %-15s | %-8s |\n",
                "Camera ID", "Brand", "Model", "Price per Day", "Status");
        System.out.println("+------------+-----------------+----------------------+--
---------------+----------+");

        for (Camera camera : cameraList) {
            System.out.printf("| %-10d | %-15s | %-20s | %-15.2f | %-8s |\n",
            camera.getCameraId(), camera.getBrand(), camera.getModel(),
            camera.getPricePerDay(), camera.isRented() ? "Rented""Available");
        }
       System.out.println("+------------+-----------------+----------------------+---
--------------+----------+");
        System.out.println();
        System.out.print("Enter the camera ID to remove: ");
        int cameraId = scanner.nextInt();

        Camera cameraToRemove = null;
        for (Camera camera : cameraList) {
            if (camera.getCameraId() == cameraId) {
                cameraToRemove = camera;
                break;
            }
        }

        if (cameraToRemove != null) {
            cameraList.remove(cameraToRemove);
            System.out.println("Camera successfully removed from the list.");
        } else {
            System.out.println("Camera with the specified ID not found.");
        }
    }

    private static void viewMyCameras() {
        System.out.println("+--------------------------------------------------------
--------------------------+");
        System.out.println("|                 MY CAMERAS LIST
|");
        System.out.println("+--------------------------------------------------------
--------------------------+");
        System.out.printf("| %-10s | %-15s | %-20s | %-15s | %-8s |\n",
                "Camera ID", "Brand", "Model", "Price per Day", "Status");
        System.out.println("+------------+-----------------+----------------------+--
---------------+----------+");

        for (Camera camera : cameraList) {

             System.out.printf("| %-10d | %-15s | %-20s | %-15.2f | %-8s |\n",
                    camera.getCameraId(), camera.getBrand(), camera.getModel(),
                    camera.getPricePerDay(), camera.isRented() ? "Rented"Available");
            //System.out.println(camera);
        }
        System.out.println("+------------+-----------------+----------------------+--
---------------+----------+");

        System.out.println();
    }

    private static void handleRentCamera(Scanner scanner) {
        System.out.println("+--------------------------------------------------------
--------------------------+");
        System.out.println("|               AVAILABLE CAMERAS LIST
|");
        System.out.println("+--------------------------------------------------------
--------------------------+");

     // Print table header
        System.out.printf("| %-10s | %-15s | %-20s | %-15s | %-8s |\n",
                "Camera ID", "Brand", "Model", "Price per Day", "Status");
        System.out.println("+------------+----------------+---------------------+--
---------------+----------+");

        ArrayList<Camera> availableCameras = new ArrayList<>();
        for (Camera camera : cameraList) {
            if (!camera.isRented()) {
                availableCameras.add(camera);

      // Print each camera in the table format
          System.out.printf("| %-10d | %-15s | %-20s | %-15.2f | %-8s |\n",
          camera.getCameraId(), camera.getBrand(), camera.getModel(),
          camera.getPricePerDay(), camera.isRented() ? "Rented" : "Available");
            }
        }

        System.out.println("+------------+----------------+---------------------+--
---------------+----------+");
        System.out.println();

        System.out.print("Enter the camera ID you want to rent: ");
        int cameraId = scanner.nextInt();

        Camera selectedCamera = null;
        for (Camera camera : availableCameras) {
            if (camera.getCameraId() == cameraId) {
                selectedCamera = camera;
                break;
            }
        }

        if (selectedCamera != null) {
            double rentAmount = selectedCamera.getPricePerDay();
            if (walletBalance >= rentAmount) {
                selectedCamera.setRented(true);
                walletBalance -= rentAmount;
                System.out.printf("YOUR TRANSACTION FOR CAMERA - %s %s with rent INR
%.2f HAS SUCCESSFULLY COMPLETED%n",
                        selectedCamera.getBrand(), selectedCamera.getModel(),
rentAmount);
            } else {
                System.out.println("ERROR: TRANSACTION FAILED DUE TO INSUFFICIENT
WALLET BALANCE. PLEASE DEPOSIT THE AMOUNT TO YOUR WALLET");
            }
        } else {
            System.out.println("Invalid camera ID. Please try again.");
        }
    }




//Quicksort is used to sort the cameralist based on the cameraID
    private static void quickSort(ArrayList<Camera> list, int low, int high) {
        if (low < high) {
            int pivotIndex = partition(list, low, high);
            quickSort(list, low, pivotIndex - 1);
            quickSort(list, pivotIndex + 1, high);
        }
    }

    private static int partition(ArrayList<Camera> list, int low, int high) {
        Camera pivot = list.get(high);
        int i = low - 1;

        for (int j = low; j < high; j++) {
            if (list.get(j).getCameraId() < pivot.getCameraId()) {
                i++;
                Collections.swap(list, i, j);
            }
        }

        Collections.swap(list, i + 1, high);
        return i + 1;
    }

    private static void handleViewAllCameras() {
        System.out.println("+--------------------------------------------------------
---------------------------+");
        System.out.println("|               ALL CAMERAS LIST
|");
        System.out.println("+--------------------------------------------------------
---------------------------+");

        quickSort(cameraList, 0, cameraList.size() - 1);

        System.out.printf("| %-10s | %-15s | %-20s | %-15s | %-8s |\n",
                "Camera ID", "Brand", "Model", "Price per Day", "Status");
        System.out.println("+------------+-----------------+----------------------+--
---------------+----------+");

        for (Camera camera : cameraList) {
            System.out.printf("| %-10d | %-15s | %-20s | %-15.2f | %-8s |\n",
                    camera.getCameraId(), camera.getBrand(), camera.getModel(),
                    camera.getPricePerDay(), camera.isRented() ? "Rented"Available");
        }

        System.out.println("+------------+-----------------+----------------------+--
---------------+----------+\n");
    }




     private static void handleMyWallet(Scanner scanner) {
     System.out.println("YOUR CURRENT WALLET BALANCE IS - INR " + walletBalance);
     System.out.print("DO YOU WANT TO DEPOSIT MORE AMOUNT TO YOUR WALLET? (1.YES
2.NO): ");
     int depositChoice = scanner.nextInt();
    if (depositChoice == 1) {
        System.out.print("ENTER THE AMOUNT(INR): ");
        double amount = scanner.nextDouble();

        walletBalance += amount;
        System.out.println("YOUR WALLET BALANCE UPDATED SUCCESSFULLY. CURRENT WALLET
BALANCE: INR " + walletBalance);
    } else {
        System.out.println("Going back to the previous menu...");
    }
}
}
```