SOURCE CODE OF PHASE END PROJECT CAMERA RENTAL APPLICATION

User.java (class file)-

```
package com.pushpa;

public class User {

    private String username;
    private String password;

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }
}
```

LoginManager.java (class file)-

```
package com.pushpa;
import java.util.ArrayList;
import java.util.List;

public class LoginManager {

    private List<User> userList;

    public LoginManager() {
        userList = new ArrayList<>();
        userList.add(new User("admin", "password")); // Sample user, replace
with your own
    }

    public boolean authenticateUser(String username, String password) {
        for (User user : userList) {
            if (user.getUsername().equals(username) &&
user.getPassword().equals(password)) {
```

```
                return true;
            }
        }
        return false;
    }
}
```

Camera.java (class file)-

```
package com.pushpa;

public class Camera {

    private int id;
    private String brand;
    private String model;
    private double perDayRent;
    private boolean rented;

    public Camera(int id, String brand, String model, double perDayRent) {
        this.id = id;
        this.brand = brand;
        this.model = model;
        this.perDayRent = perDayRent;
        this.rented = false; //Initialize as not rented
    }

    // Getters and setters for the camera

    public int getId() {
        return id;
    }

    public String getBrand() {
        return brand;
    }

    public String getModel() {
        return model;
    }

    public double getPerDayRent() {
        return perDayRent;
    }

    public boolean isRented() {
        return rented;
    }

    public void setRented(boolean rented) {
        this.rented = rented;
    }
}
```

CameraRentalApp.java (class file)-

```
package com.pushpa;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.InputMismatchException;
import java.util.List;
import java.util.Scanner;

public class CameraRentalApp {

    private ArrayList<Camera> cameraList;
    private int nextCameraId;
    private double walletBalance;

    public CameraRentalApp() {
        cameraList = new ArrayList<>();
        nextCameraId = 1;
        walletBalance = 0.0;
    }

    public int getCurrentId() {
        return nextCameraId;
    }

    public double getCurrentWalletBalance() {
        return walletBalance;
    }

    public void addCamera(int id, String brand, String model, double
perDayRent) {
        Camera camera = new Camera(nextCameraId, brand, model, perDayRent);
        cameraList.add(camera);
        nextCameraId++;
    }

    public List<Camera> getCameraList() {
```

```java
        return cameraList;
    }

    public void displayCameraList() {
        if (cameraList.isEmpty()) {
            System.out.println("NO DATA PRESENT AT THIS MOMENT!");
            return;
        } else {
            System.out.println("==============================================
====================");
            System.out.println("CAMERA ID    BRAND       MODEL       PRICE(PER
DAY)        STATUS");
            System.out.println("==============================================
====================");
            for (Camera camera : cameraList) {
                String status = camera.isRented() ? "Rented" : "Available";
                System.out.printf("  %-7d    %-10s    %-10s    %-13.2f    %-
12s  \n",
                        camera.getId(), camera.getBrand(), camera.getModel(),
camera.getPerDayRent(), status);
            }
            System.out.println("==============================================
====================");
        }
    }

    public void removeCameraById() {
        @SuppressWarnings("resource")
        Scanner scanner = new Scanner(System.in);
        try {

            int cameraId = scanner.nextInt();
            scanner.nextLine();
            Camera cameraToRemove = null;
            for (Camera camera : cameraList) {
                if (camera.getId() == 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 MENTIONED ID NOT FOUND IN THE
LIST.");
```

```
            }
        } catch (InputMismatchException e) {
            System.out.println("Invalid input. Please enter a valid integer
camera ID.");
            scanner.nextLine(); // Clear the input buffer
        }
    }


    public void displayAvailableCameras() {
        System.out.println("==================================================
====================");
        System.out.printf("%-10s %-15s %-15s %-15s %-15s%n", "CAMERA ID",
"BRAND", "MODEL", "PRICE(PER DAY)", "STATUS");
        System.out.println("==================================================
====================");

        boolean availableCamerasExist = false;
        for (Camera camera : cameraList) {
            if (!camera.isRented()) {
                availableCamerasExist = true;
                System.out.printf("%-10d %-15s %-15s %.2f %-20s%n",
camera.getId(), camera.getBrand(), camera.getModel(), camera.getPerDayRent(),
"   Available");
            }
        }

        if (!availableCamerasExist) {
            System.out.println("NO AVAILABLE CAMERA AT THIS MOMENT !");
        }
    }

    public void rentCamera(int cameraId, Camera camera) {
        Camera selectedCamera = null;
        for (Camera c : cameraList) {
            if (c.getId() == cameraId) {
                selectedCamera = c;
                break;
            }
        }

        if (selectedCamera == null) {
            System.out.println("CAMERA WITH ID " + cameraId + " NOT FOUND.");
            return;
        }

        if (selectedCamera.isRented()) {
            System.out.println("CAMERA WITH ID " + cameraId + " IS ALREADY
RENTED.");
```

```java
            return;
        }

        if (walletBalance < selectedCamera.getPerDayRent()) {
            System.out.println("ERROR : TRANSACTION FAILED DUE TO INSUFFICIENT
WALLET BALANCE. PLEASE DEPOSIT THE AMOUNT TO YOUR WALLET.");
            return;
        }

        selectedCamera.setRented(true);
        walletBalance -= selectedCamera.getPerDayRent();
        System.out.printf("YOUR TRANSACTION FOR CAMERA - %s %s WITH RENT
INR.%.2f HAS SUCCESSFULLY COMPLETED.\n",
                selectedCamera.getBrand(), selectedCamera.getModel(),
selectedCamera.getPerDayRent());
    }

    public void depositToWallet(double amount) {
        walletBalance += amount;
        System.out.println("YOUR WALLET BALANCE UPDATED SUCCESSFULLY. CURRENT
WALLET BALANCE - INR." + walletBalance);
    }

    public void sortCameraList() {
        Collections.sort(cameraList,
Comparator.comparing(Camera::getBrand).thenComparing(Camera::getModel));
    }
}
```

Main.java (class file)-

```java
package com.soumya;
import java.util.InputMismatchException;
import java.util.Scanner;

//Driver code
public class Main {
    public static void main(String[] args) {
        //Initialising scanner class
        Scanner scanner = new Scanner(System.in);
        //Initialising objects by using their class
        LoginManager loginManager = new LoginManager();
        CameraRentalApp app = new CameraRentalApp();
        System.out.println("+-----------------------------------------------+");
        System.out.println("|  WELCOME  TO  CAMERA  RENTAL  APP  |");
        System.out.println("+-----------------------------------------------+");
        boolean authenticated = false;
        while (!authenticated) {
            System.out.println("\nPLEASE LOGIN TO CONTINUE - ");
            System.out.print("USERNAME - ");
            String username = scanner.next();
            System.out.print("PASSWORD - ");
            String password = scanner.next();

            if (loginManager.authenticateUser(username, password)) {
                authenticated = true;
            } else {
                System.out.println("INVALID USERNAME OR PASSWORD. PLEASE TRY
AGIN ! ");
            }
        }
        boolean exit = false;
        while (!exit) {
            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");
            try {
                int choice = scanner.nextInt();

                switch (choice) {

                case 1:
                    boolean cameraExit = false;
                    do {
                        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");
                        int cameraChoice = scanner.nextInt();

                        switch (cameraChoice) {

                        case 1:
                            System.out.print("ENTER THE CAMERA BRAND - ");
                            String brand = scanner.next();
                            System.out.print("EMTER THE MODEL - ");
                            String model = scanner.next();
                            System.out.print("ENTER THE PER DAY PRICE (INR) -
");

                            double perDayRent = scanner.nextDouble();
                            app.addCamera(choice, brand, model, perDayRent);
                            System.out.println("YOUR CAMERA HAS BEEN
SUCCESSFULLY ADDED TO THE LIST.");
                            break;
                        case 2:
                            app.displayCameraList();
                            System.out.println("ENTER THE CAMERA ID TO REMOVE
- ");

                            app.removeCameraById();
                            break;
                        case 3:
                            System.out.println("3. VIEW MY CAMERAS");
                            app.displayCameraList();
                            break;
                        case 4:
                            cameraExit = true;
                            break;
                        default:
                            System.out.println("INVALID CHOICE!");
                        }
                    } while (!cameraExit);
                    break;

                case 2:
                    System.out.println("FOLLOWING IS THE LIST OF AVAILABLE
CAMERA(S) - ");
                    app.displayAvailableCameras(); // Call a method to display
the available cameras
                    System.out.println("ENTER THE CAMERA ID YOU WANT TO RENT -
");
                    int cameraId = scanner.nextInt();
                    scanner.nextLine();

                    Camera selectedCamera = null;
                    for (Camera camera : app.getCameraList()) {
                        if (camera.getId() == cameraId) {

                            selectedCamera = camera;
                            break;
                        }
                    }

                    if (selectedCamera != null) {
                        app.rentCamera(cameraId, selectedCamera);
                    } else {
                        System.out.println("INVALID CAMERA ID.");
                    }
                    break;
```

```java
                case 3:
                    app.displayCameraList();
                    break;

                case 4:
                    double walletBalance = app.getCurrentWalletBalance();
                    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)\n");
                    int walletChoice = scanner.nextInt();

                    switch (walletChoice) {

                    case 1:
                        System.out.print("ENTER THE AMOUNT (INR) - ");
                        double amount = scanner.nextDouble();
                        app.depositToWallet(amount);
                        break;
                    case 2:
                        System.out.println("NO UPDATES HAS BEEN DONE BY THE
USER. THUS CURRENT BALANCE - INR." +walletBalance);
                        break;
                    default:
                        System.out.println("INVALID CHOICE.");
                    }
                    break;

                case 5:
                    exit = true;
                    System.out.println("EXITING THE APPLICATION... THANK
YOU!");

                    break;
                default:
                    System.out.println("INVALID CHOICE!");
                    break;
                }
            } catch (InputMismatchException e) {
                // TODO: handle exception
                System.out.println("INVALID INPUT. PLEASE ENTER A VALID
INTEGER CHOICE.");
                scanner.nextLine();
            }
        }
        scanner.close();
    }
}
```