Phase :1 **Name:Veeravalli Deepika**

Project : Camera Rental Application

Source Code:

:

```
import java.util.*;
class Camera {
private int cameraId;
private String brand;
private String model;
private double rentalPricePerDay;
private boolean rented;

public Camera(int cameraId, String brand, String model, double rentalPricePerDay) {
this.cameraId = cameraId;
this.brand = brand;
this.model = model;
this.rentalPricePerDay = rentalPricePerDay;
this.rented = false;
}

public int getCameraId() {
return cameraId;
}
public String getBrand() {
return brand;
}
public String getModel() {
return model;
```

```
}
public double getRentalPricePerDay() {
return rentalPricePerDay;
}
public boolean isRented() {
return rented;
}
public void getRented(boolean rented) {
this.rented = rented;
}
public void setRented(boolean rented) {
this.rented = rented;
}
public String getRented() {
return null;
}
}
```

User:

```
public class User {
private String username;
private String password;
private double walletBalance;

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

```
public String getUsername() {
return username;
}
public String getPassword() {
return password;
}
public double getWalletBalance() {
return walletBalance;
}
public void depositToWallet(double amount) {
walletBalance += amount;
System.out.println("Amount deposited successfully. Current wallet balance: " +
walletBalance);
}
}
```

Camera Rental Application:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class CameraRentalApp {
    private static Scanner scanner = new Scanner(System.in);
    private static List<Camera> cameraList = new ArrayList<>();
    private static User currentUser;
    private static List<User> userList;

    public static void main(String[] args) {
        populateCameraList(); // Populate some initial camera data
```

```
boolean running = true;


while (running) {
       System.out.println("+----------------+----------------+");
    System.out.println("|Welcome to the Camera Rental App!!|");
    System.out.println("+----------------+----------------+");
    if (currentUser == null) {
        displayLoginMenu();
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character


        switch (choice) {
            case 1:
                registerUser();
                break;
            case 2:
                loginUser();
                break;
            case 3:
                running = false;
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    } else {
      displayUserMenu();
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character
```

```
            switch (choice) {
                case 1:
                    displayCameras();
                    break;
                case 2:
                    rentCamera();
                    break;
                case 3:
                    listCameras();
                    break;
                case 4:
                    displayWalletBalance();
                    break;
                case 5:
                    logoutUser();
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
                    }
            }
        }

        System.out.println("Thank you for using the Camera Rental App!");
    }

    private static void displayLoginMenu() {

        System.out.println("1. Register");
        System.out.println("2. Login");
```

```
        System.out.println("3. Exit");
        System.out.println();
        System.out.print("Enter your choice: ");
    }

    private static void displayUserMenu() {
        System.out.println("Welcome, " + currentUser.getUsername() + "!");

        System.out.println("1. My Cameras");
        System.out.println("2. Rent a Camera");
        System.out.println("3. View All Cameras");
        System.out.println("4. Manage Wallet");
        System.out.println("5. Exit");
        System.out.println();
        System.out.print("Enter your choice: ");
    }

    private static void registerUser() {

        System.out.print("Enter username: ");
        String username = scanner.nextLine();

        System.out.print("Enter password: ");
        String password = scanner.nextLine();



        currentUser = new User(username, password);
```

```
        System.out.println("Registration successful!!!.");
    }

      private static void loginUser() {
        System.out.print("Enter username: ");
        String username = scanner.nextLine();

        System.out.print("Enter password: ");
        String password = scanner.nextLine();

                if (currentUser != null && currentUser.getUsername().equals(username) &&
currentUser.getPassword().equals(password)) {
            System.out.println("Login successful! Welcome, " + currentUser.getUsername() + "!");
        } else {
            System.out.println("Invalid username or password. Please try again.");
            currentUser = null;
        }
    }


    private static void logoutUser() {
        currentUser = null;
        System.out.println("Logout successful!");
        System.out.println("Thank you!!!");

    }

    private static void populateCameraList() {
        cameraList.add(new Camera(1, "Canon", "EOS R", 500.0));
```

```
    cameraList.add(new Camera(2, "Nikon", "D850", 600.0));
    cameraList.add(new Camera(3, "Sony", "Alpha A7 III", 700.0));
}
private static void displayCameras() {
    while (true) {
        System.out.println("---------------------------");
        System.out.println("1. Add a Camera");
        System.out.println("2. Remove a Camera");
        System.out.println("3. Go back to Previous Menu");
        System.out.println("---------------------------");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character

        switch (choice) {
            case 1:
                addCamera();
                break;
            case 2:
                removeCamera();
                break;

            case 3:
                return; // Go back to the previous menu
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
}
```

```
    private static void listCameras() {
        if (cameraList.isEmpty()) {
            System.out.println("No Data Present at This Moment.");
        } else {
            System.out.println("======Available cameras:======");
           // for (Camera camera : cameraList) {

System.out.println("========================================================================"
);
            System.out.println("cameraId       Brand       Model       Rent per Day      status");

System.out.println("========================================================================
===");
           for (Camera camera : cameraList) {
               System.out.printf("%-5s      %-7s      %-13s    $%-12.2f %-10s \n",
                       camera.getCameraId(),camera.getBrand(), camera.getModel(),
camera.getRentalPricePerDay(),(camera.isRented() ? "Rented" : "Available"));
            }

            System.out.println("============================================================");
        }

            }



    private static void addCamera() {
        System.out.print("Enter camera ID: ");
```

```
    int cameraId = scanner.nextInt();
    scanner.nextLine(); // Consume the newline character

    System.out.print("Enter camera brand: ");
    String brand = scanner.nextLine();

    System.out.print("Enter camera model: ");
    String model = scanner.nextLine();

    System.out.print("Enter rental price per day: ");
    double rentalPricePerDay = scanner.nextDouble();
    scanner.nextLine(); // Consume the newline character

    Camera camera = new Camera(cameraId, brand, model, rentalPricePerDay);
    cameraList.add(camera);
    System.out.println("Camera added to the main display.");
}

private static void removeCamera() {
    System.out.print("Enter camera ID to remove: ");
    int cameraId = scanner.nextInt();
    scanner.nextLine(); // Consume the newline character

    boolean found = false;
    for (Camera camera : cameraList) {
        if (camera.getCameraId() == cameraId) {
            cameraList.remove(camera);
            found = true;
            System.out.println("Camera removed successfully.");
```

```
            break;
        }
    }

    if (!found) {
        System.out.println("Camera not found.");
    }
}



private static void rentCamera() {
    System.out.print("Enter camera ID to rent: ");
    int cameraId = scanner.nextInt();
    scanner.nextLine(); // Consume the newline character

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

    if (selectedCamera != null) {
        if (selectedCamera.isRented()) {
            System.out.println("Camera is already rented.");
        } else {
            if (currentUser.getWalletBalance() >= selectedCamera.getRentalPricePerDay()) {
                currentUser.depositToWallet(-selectedCamera.getRentalPricePerDay());
```

```
                selectedCamera.setRented(true);
                System.out.println("Camera rented successfully!");
            } else {
                System.out.println("Insufficient balance in your wallet to rent this camera add money to your wallet .");
            }
        }
    } else {
        System.out.println("Camera not found.");
    }
}
private static void displayWalletBalance() {
    while (true) {
        System.out.println("----------------------------");
        System.out.println("1. Deposit Money");
        System.out.println("2. Show Available Balance");
        System.out.println("3. Go back to Previous Menu");
        System.out.println("----------------------------");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character

        switch (choice) {
            case 1:
                depositToWallet();
                break;
            case 2:
                showAvailableBalance();
                break;
```

```
            case 3:
                return; // Go back to the previous menu
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
}

private static void depositToWallet() {
    System.out.print("Enter the amount to deposit: ");
    double amount = scanner.nextDouble();
    scanner.nextLine();

   currentUser.depositToWallet(amount);
    System.out.println("Amount deposited successfully!");



      }

private static void showAvailableBalance() {
    double balance = currentUser.getWalletBalance();
    System.out.println("Available Balance: $" + balance);
    System.out.print("Do you want to deposit more? (y/n): ");
    String choice = scanner.nextLine().toLowerCase();


 // Ask if the user wants to deposit more money
    if (choice.equals("y")) {
        depositToWallet(); // Recursively call the depositToWallet() method
```

```
        }

      }



}
```