**Camera Rental Application**

**Phase-End Project Problem Statement**

**Project Agenda:** Create a Camera Rental Application

**Source Code:**

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class Camera {

private String brand;

private String model;

private double perDayPrice;

private boolean isRented;

public Camera(String brand, String model, double perDayPrice) {

this.brand = brand;

this.model = model;

this.perDayPrice = perDayPrice;

this.isRented = false;

}

public String getBrand() {

return brand;

}

public String getModel() {

return model;

}

public double getPerDayPrice() {

return perDayPrice;

}

public boolean isRented() {

return isRented;

}

public void setRented(boolean rented) {

isRented = rented;

}

public String toString() {

return brand + " " + model + " - " + perDayPrice;

}

}

class Wallet {

private double balance;

public Wallet() {

this.balance = 0.0;

}

public double getBalance() {

return balance;

}

public void deposit(double amount) {

balance += amount;

}

public boolean withdraw(double amount) {

if (amount <= balance) {

balance =balance-amount;

return true;

}

return false;

}

}

public class CameraRentalApp {

public static List<Camera> cameraList = new ArrayList<>();

private static Wallet wallet = new Wallet();

public static void main(String[] args) {

boolean exit = false;

Scanner scanner = new Scanner(System.in);

displayWelcomeScreen();

manageMainMenu();

scanner.close();

}

private static void displayWelcomeScreen() {

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

System.out.println("| WELCOME TO CAMERA RENTAL APP |");

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

System.out.println("PLEASE LOGIN TO CONTINUE -");

Scanner sc=new Scanner(System.in);

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

sc.next();

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

sc.next();

System.out.println(" ");

System.out.println("Login Succesful");

System.out.println(" ");

}

private static int getUserChoice(Scanner scanner) {

System.out.print("Enter your choice: ");

return scanner.nextInt();

}

private static void manageMainMenu( ) {

boolean exit = false;

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

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("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Scanner scanner = new Scanner(System.in);

int choice = getUserChoice(scanner);

switch (choice) {

case 1:

manageMyCamera(scanner,false);

manageMainMenu();

break;

case 2:

rentCamera(scanner);

manageMainMenu();

break;

case 3:

viewAllCameras();

manageMainMenu();

break;

case 4:

manageWallet(scanner);

manageMainMenu();

break;

case 5:

exitApplication();

break;

default:

System.out.println("Invalid choice. Please try again.");

}

}

private static void manageMyCamera(Scanner scanner, boolean displayMainMenu) {

System.out.println("\n1. ADD");

System.out.println("2. REMOVE");

System.out.println("3. VIEW MY CAMERAS");

System.out.println("4. GO TO PREVIOUS MENU");

int choice = getUserChoice(scanner);

switch (choice) {

case 1:

addCamera(scanner);

manageMyCamera(scanner, false);

break;

case 2:

removeCamera(scanner);

manageMyCamera(scanner, false);

break;

case 3:

viewMyCameras();

manageMyCamera(scanner, false);

break;

case 4:

manageMainMenu();

break;

default:

System.out.println("Invalid choice. Please try again.Please re-enter the choice below");

manageMainMenu();

}

}

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 perDayPrice = scanner.nextDouble();

Camera camera = new Camera(brand, model, perDayPrice);

cameraList.add(camera);

System.out.println("YOUR CAMERA HAS BEEN SUCCESSFULLY ADDED TO THE LIST.");

}

private static void removeCamera(Scanner scanner) {

viewMyCameras();

System.out.print("Enter the camera ID to remove: ");

int cameraId = scanner.nextInt();

if (cameraId >= 0 && cameraId < cameraList.size()) {

cameraList.remove(cameraId);

System.out.println("CAMERA SUCCESSFULLY REMOVED FROM THE LIST.");

} else {

System.out.println("INVALID CAMERA ID.");

}

}

private static void viewMyCameras() {

if (cameraList.isEmpty()) {

System.out.println("No cameras present at this moment.");

} else {

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

System.out.printf("%-10s %-10s %-10s %-10s %-10s\n",

"CAMERA ID", "BRAND", "MODEL", "PRICE(Per day)", "STATUS");//%-10s it iused for holding the data.

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

int id = 0;

for (Camera camera : cameraList) {

System.out.printf("%-10s %-10s %-10s %-10.2f %-10s\n",

id++, camera.getBrand(), camera.getModel(),

camera.getPerDayPrice(), camera.isRented() ? "Rented" : "Available”);

}

}

}

private static void rentCamera(Scanner scanner) {

viewAllCameras();

if (cameraList.isEmpty()) {

System.out.println("No cameras available for rent at this moment.");

return;

}

System.out.print("Enter the camera ID you want to rent: ");

int cameraId = scanner.nextInt();

if (cameraId >= 0 && cameraId < cameraList.size()) {

Camera camera = cameraList.get(cameraId);

if (camera.isRented()) {

System.out.println("Camera is already rented.");

} else {

if (wallet.getBalance() >= camera.getPerDayPrice()) {

wallet.withdraw(camera.getPerDayPrice());

camera.setRented(true);

String output = "YOUR TRANSACTION FOR CAMERA " + camera.getBrand() + " " + camera.getModel() +

" WITH RENT INR." + camera.getPerDayPrice() + " HAS SUCCESSFULLY COMPLETED";

System.out.println(output);

} else {

System.out.println("Insufficient wallet balance. Please deposit the amount to your wallet.");

}

}}

else {

System.out.println("Invalid camera ID.");

}

}

private static void viewAllCameras() {

System.out.println("\nFOLLOWING IS THE LIST OF AVAILABLE CAMERA(S)\n");

if (cameraList.isEmpty()) {

System.out.println("No cameras available at this moment.");

} else {

System.out.printf("%-10s %-10s %-10s %-10s %-10s\n",

"CAMERA ID", "BRAND", "MODEL", "PRICE", "STATUS");

int id = 1;

for (Camera camera : cameraList) {

System.out.printf("%-10s %-10s %-10s %-10.2f %-10s\n",

id++, camera.getBrand(), camera.getModel(),

camera.getPerDayPrice(), camera.isRented() ? "Rented" : "Available");

}

}

}

private static void manageWallet(Scanner scanner) {

Scanner sc=new Scanner(System.in);

System.out.println("\nMY WALLET\n");

System.out.printf("Your current wallet balance is INR %.2f\n", wallet.getBalance());

System.out.println("Do you want to deposit more amount to your wallet?");

System.out.println("1. Yes");

System.out.println("2. No");

int choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.print("Enter the amount (INR): ");

double amount = scanner.nextDouble();

wallet.deposit(amount);

System.out.printf("Your wallet balance updated successfully. Current wallet balance: INR %.2f\n", wallet.getBalance());

break;

case 2:

break;

default:

System.out.println("Invalid choice. Please try again.");

}

}

private static void exitApplication() {

System.out.println("Exiting the application... Goodbye!");

System.exit(0);

}

}

**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.

**Description:**

* It defines a camera rental application with classes for Camera and Wallet.
* The main class, CameraRentalApp, implements functionality for managing cameras, renting them, viewing camera details, and managing wallet balance.
* The program has a text-based interface with menus, login functionality, and feedback on transactions.
* Users can add/remove cameras, view their cameras, rent available cameras, view all available cameras, and manage their wallet balance.
* It providesa way to interact with the Camera Rental Application System.

**Sprint Planning:**

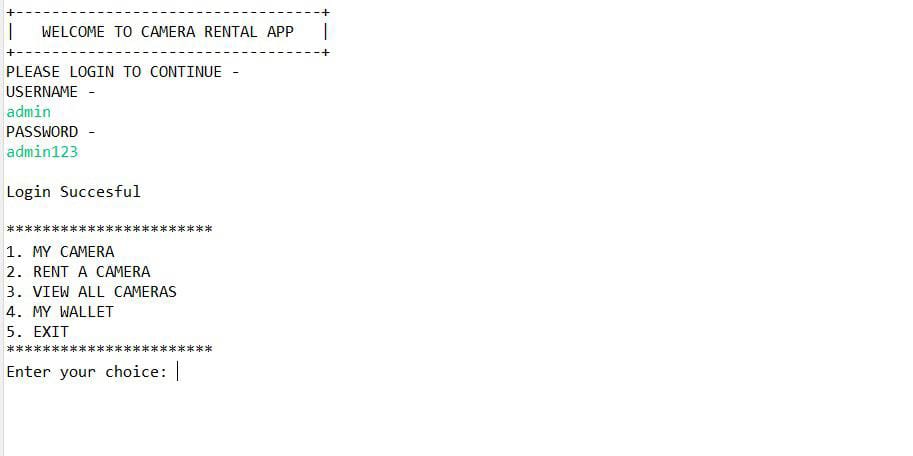
* It took almost 6 hours to complete the task
* For first 3 hours , the first option with 3 sub options i.e; ADD, REMOVE, VIEW CAMERA’S
* Next 3 hours for setting remaining options.

**OUTPUT:**

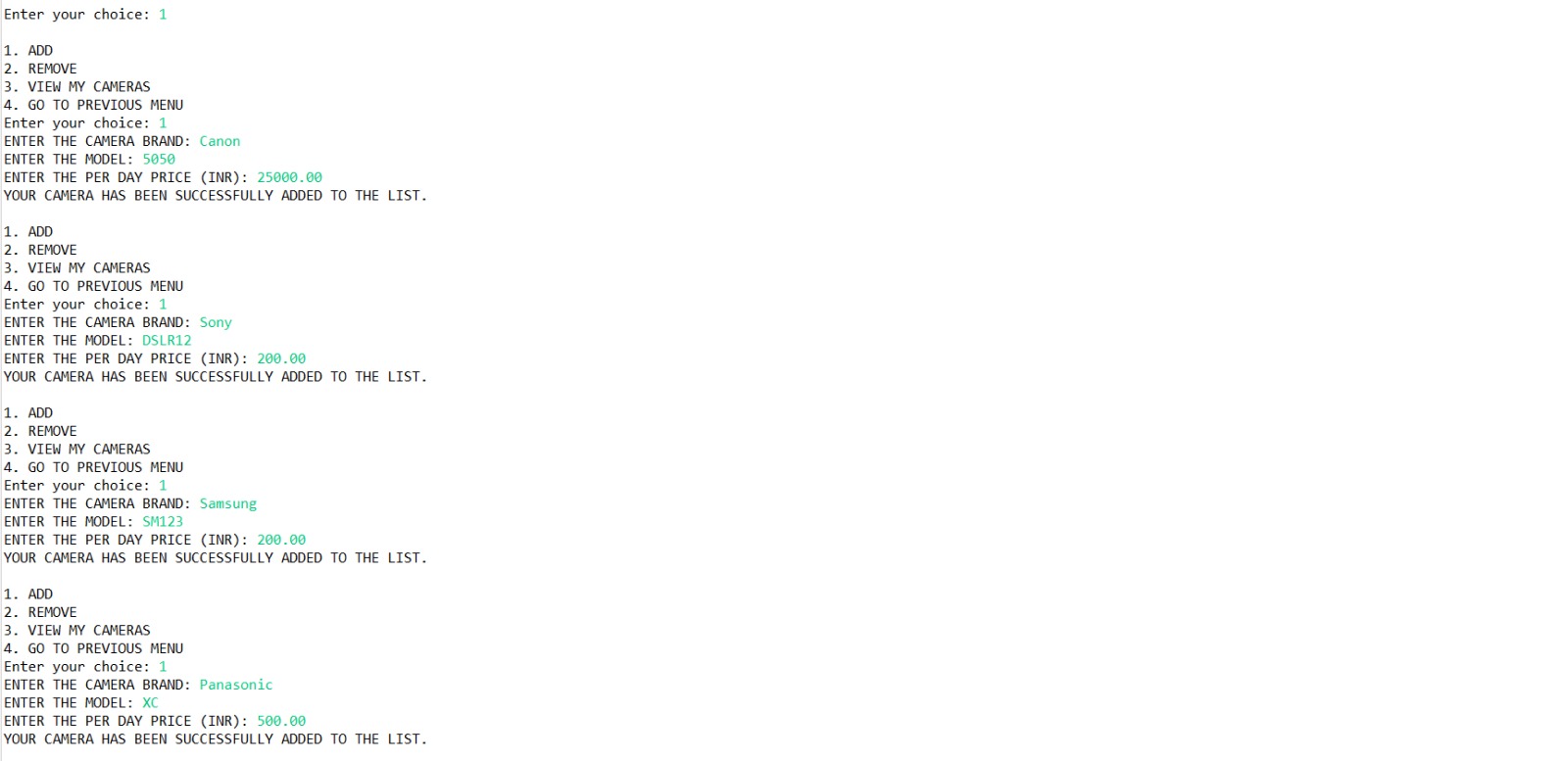
**1.Login :**



**2.Main Menu :**



**3.Add Camera :**

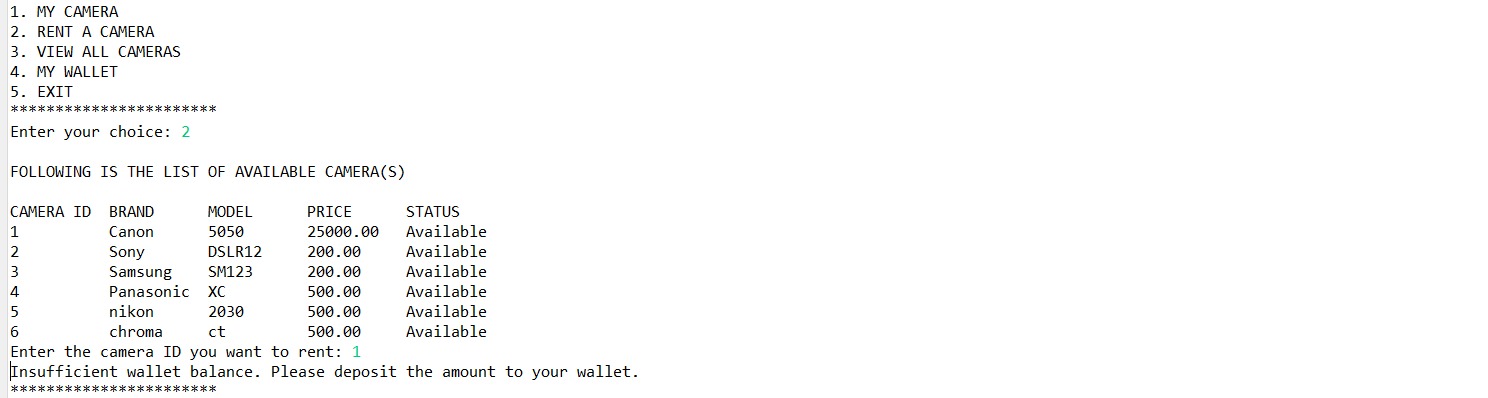


****

**4.View My Camera :**



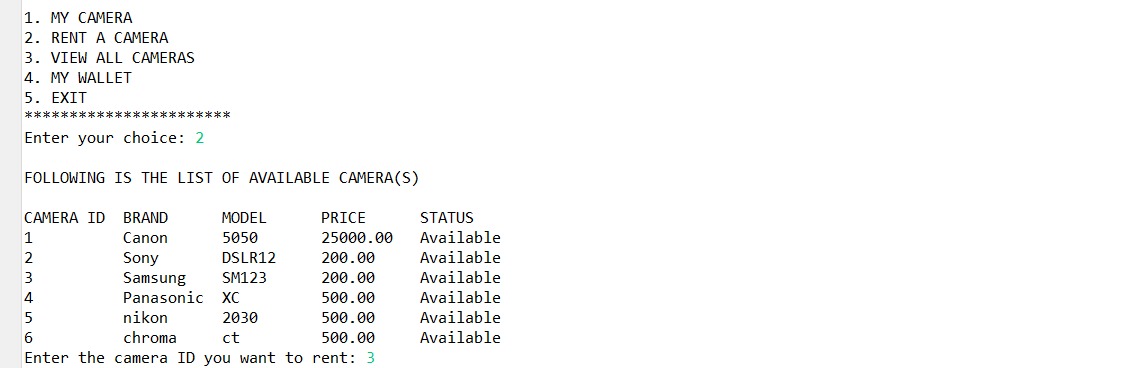
**5.Rent Camera But Insufficient Balance :**

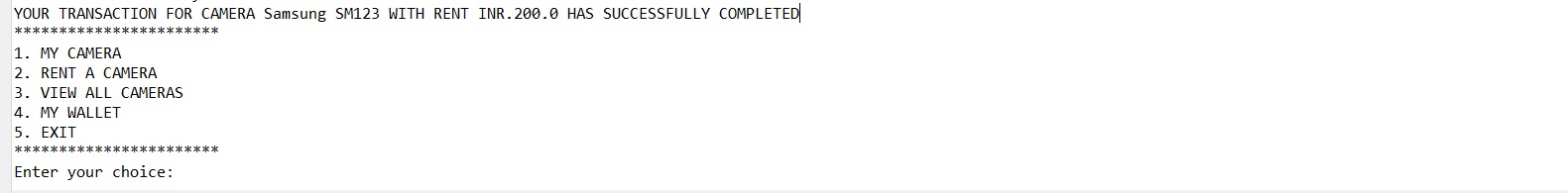


**6.Add Amount to Wallet :**



**7.Rent Camera Successfully :**





**8.Check Balance :**



**9.View my Camera :**



**10.Exiting :**

