📚 Case Study Report

Title:

"Smart Parking Lot Management System Using Java OOP"

1. Introduction

Urban cities face increasing challenges in managing limited parking spaces. Manual monitoring leads to inefficiencies such as unauthorized parking, time wastage, and revenue loss. This case study presents a Smart Parking Lot Management System, a console-based Java application that allows tracking vehicle entries, calculating parking duration and fees, and managing lot availability.

It demonstrates real-world application of Object-Oriented Programming, control statements, string handling, and other Java programming fundamentals.

2. Objectives

Automate vehicle entry/exit records using Java objects.

Track parking duration using simple logic and operators.

Calculate dynamic parking fees based on duration.

Simulate real-life constraints like parking capacity and vehicle type.

Implement core Java concepts in a realistic problem domain.

3. Concepts and Features Applied

Java Concept Area Real-Life Application in Case Study

OOP Principles Vehicle class, Encapsulation, Object Instantiation

Program Structure main(), classes, methods, constructors

Tokens & Statements Identifiers, literals, keywords, expressions

Data Types & Variables int, String, double, boolean, char

Control Statements if-else, switch, while, for, break

String Handling .equalsIgnoreCase(), .toUpperCase(), .substring()

Input and Output Scanner, System.out.println(), System.out.printf()

Operators Arithmetic, relational, ternary, logical

4. Variables and Data Types

Variable Name Type Purpose

plateNumber String Unique ID for the vehicle

vehicleType String Car, Bike, or EV

entryTime int Simulated time of entry (e.g., 9 for 9 AM)

exitTime int Simulated time of exit

fee double Calculated fee based on time and type

slotAvailable boolean Status of parking slot

vehicleCount static int Number of vehicles currently in the lot

5. Java Program Implementation

import java.util.Scanner;

class Vehicle {

private String plateNumber;

private String vehicleType;

private int entryTime;

private int exitTime;

private double fee;

static int vehicleCount = 0;

static final int MAX\_CAPACITY = 5;

public Vehicle(String plateNumber, String vehicleType, int entryTime) {

this.plateNumber = plateNumber.toUpperCase();

this.vehicleType = vehicleType;

this.entryTime = entryTime;

vehicleCount++;

}

public void exitLot(int exitTime) {

this.exitTime = exitTime;

int duration = exitTime - entryTime;

if (duration <= 0) {

duration = 1; // minimum 1 hour

}

switch (vehicleType.toLowerCase()) {

case "car":

fee = duration \* 20;

break;

case "bike":

fee = duration \* 10;

break;

case "ev":

fee = duration \* 15;

break;

default:

fee = duration \* 25;

}

vehicleCount--;

}

public void displayReceipt() {

System.out.println("\n--- Parking Receipt ---");

System.out.println("Plate Number : " + plateNumber);

System.out.println("Vehicle Type : " + vehicleType);

System.out.println("Entry Time : " + entryTime + ":00");

System.out.println("Exit Time : " + exitTime + ":00");

System.out.printf("Total Fee : ₹%.2f\n", fee);

System.out.println("Thank you for using Smart Parking!");

}

}

public class SmartParkingSystem {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

boolean systemRunning = true;

while (systemRunning) {

System.out.println("\n===== Smart Parking Lot =====");

System.out.println("1. Vehicle Entry");

System.out.println("2. Vehicle Exit");

System.out.println("3. Check Available Slots");

System.out.println("4. Exit System");

System.out.print("Choose an option: ");

int choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1:

if (Vehicle.vehicleCount >= Vehicle.MAX\_CAPACITY) {

System.out.println("Sorry, parking is full.");

break;

}

System.out.print("Enter Vehicle Plate Number: ");

String plate = sc.nextLine();

System.out.print("Enter Vehicle Type (Car/Bike/EV): ");

String type = sc.nextLine();

System.out.print("Enter Entry Time (Hour in 24h): ");

int entry = sc.nextInt();

Vehicle vehicle = new Vehicle(plate, type, entry);

System.out.println("Vehicle Parked Successfully.");

break;

case 2:

System.out.print("Enter Exit Time (Hour in 24h): ");

int exit = sc.nextInt();

// For demo: simulate a single vehicle

Vehicle dummyVehicle = new Vehicle("TS09AB1234", "Car", 9);

dummyVehicle.exitLot(exit);

dummyVehicle.displayReceipt();

break;

case 3:

int available = Vehicle.MAX\_CAPACITY - Vehicle.vehicleCount;

System.out.println("Available Slots: " + available);

break;

case 4:

systemRunning = false;

System.out.println("Exiting system. Goodbye!");

break;

default:

System.out.println("Invalid option.");

}

}

sc.close();

}

}

6. Sample Output

===== Smart Parking Lot =====

1. Vehicle Entry

2. Vehicle Exit

3. Check Available Slots

4. Exit System

Choose an option: 1

Enter Vehicle Plate Number: TS08BB1234

Enter Vehicle Type (Car/Bike/EV): Car

Enter Entry Time (Hour in 24h): 9

Vehicle Parked Successfully.

===== Smart Parking Lot =====

Choose an option: 2

Enter Exit Time (Hour in 24h): 12

--- Parking Receipt ---

Plate Number : TS09AB1234

Vehicle Type : Car

Entry Time : 9:00

Exit Time : 12:00

Total Fee : ₹60.00

Thank you for using Smart Parking!

7. Conclusion

This real-life case study showcases a simplified version of a Smart Parking Lot system, incorporating practical uses of Java programming concepts like object creation, method overloading, variable scope, control statements, and formatted outputs. It provides an excellent foundation for building more complex systems with GUI, database, or IoT integration.