|  |  |  |
| --- | --- | --- |
|  | **KONGU ENGINEERING COLLEGE**  (Autonomous)  Perundurai, Erode – 638 060  **DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING** | KEC | Kongu Engineering College |

**SIMPLE JAVA BASED BUS INFORMATION SYSTEM**

**22ITC31 - JAVA PROGRAMMING**

**AN MICRO PROJECT REPORT**

**Submitted by**

**BHARANIDHARAN S**

**(23EIR010)**

**BHARANITHARAN T**

**(23EIR011)**

**BHARATHI M**

**(23EIR012)**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**KONGU ENGINEERING COLLEGE**

**(Autonomous)**

**PERUNDURAI ERODE–638060**

**NOVEMBER 2024**

**BONAFIDE CERTIFICATE**

This is to certify that the Project report entitled **“SIMPLE JAVA BASED BUS INFORMATION SYSTEM”** is the bonafied record of project work done by **BHARANIDHARAN S (23EIR010), BHARANITHARAN T (2EIR011), BHARATHI M (23EIR012)**, in **22ITC31 - JAVA PROGRAMMING** during the academic year 2024-2025.

**COURSE FACULTY                                  HEAD OF THE DEPARTMENT**

**(Signature with seal)**

**Date:**

**ABSTRACT**

* Create a 5 Bus Records with Bus number, Route Name, In time, Out time
* Display the bus records based on out time ascending initially
* Get the Input from the User as route name or bus number
* Update the certain record it can be either bus number, route name, in time or out time
* New bus record added either last or first
* Delete the bus record from the list
* And display updated list
* Display the records either based on in time ascending descending or out time ascending based on user choice

Create variables of private string[] busNumbers, private String[] routes, and methods of public BusManagementSystem(); (constructor), public void addBus(String number, String route), public void updateRoute(String busNumber, String newRoute), public void deleteBus(String number), public void updateRoute(String busNumber, String newRoute), public void updateRoute(String busNumber, String newRoute), public void deleteBus(String number), public void displayAllBuses().

**METHODOLOGIES**

This Java code employs several methodologies to implement a Bus Management System:

**OBJECT-ORIENTED PROGRAMMING (OOP):** The code utilizes encapsulation (Bus class) and abstraction (BusManagementSystem class) to organize data and behavior. **GENERICS:** The List interface and Comparator class leverage generics for type safety. **FUNCTIONAL PROGRAMMING:** Java 8's Stream API and lambda expressions (**buses.stream( )**, **Comparator.comparing( )**) enable concise and expressive data processing. **SEPARATION OF CONCERNS**: The code divides responsibilities between the Bus class (data model) and **BusManagementSystem** class (business logic).

**COMMAND-LINE INTERFACE (CLI):** The main method uses a Scanner to interact with users, demonstrating a simple CLI.

**DATA SORTING**: The **displayBusesByOutTime** and **displayBusesByInTime** methods utilize Java's built-in sorting capabilities.

**DATA FILTERING:** The **deleteBus** method employs Java 8's **removeIf** method for efficient data removal. Overall, this code showcases a well-structured, readable, and maintainable design.

**OBJECTIVES**

**PRIMARY OBJECTIVES:**

1. Design and implement a simple Bus Information System using Java.

2. Store and manage bus records, including bus number, route name, arrival time, and departure time.

3. Provide functionality for adding, updating, and deleting bus records.

**SECONDARY OBJECTIVES:**

1. Implement user authentication and authorization.

2. Display bus records sorted by arrival or departure time.

3. Search for bus records by bus number or route name.

4. Handles error and exceptions.

Problem statement,background and Requirements:

**Background:**

Public transportation systems play a vital role in urban mobility. Efficient management of bus operations is crucial for providing reliable services to commuters. Currently, manual methods are used to manage bus records, leading to inefficiencs and errors.

**Problem statement:**

Design and implement a simple Bus Management System to automate the process of managing bus records, including adding, updating, and deleting bus information. The system should also enable sorting and searching of bus records based on arrival and departure times.

**Requirements**:

problem statement outlines the need for a simple Bus Management System, highlighting the objectives, requirements, expected outcomes, constraints, and assumptions. The provided Java code addresses these requirements, offering a functional solution.

1. Store bus records with bus number, route name, arrival time, and departure time.

2. Provide functionality for adding, updating, and deleting bus records.

3. Enable sorting of bus records by arrival and departure times.

4. Implement searching functionality by bus number or route nam

**Implementation of this program:**

**Bus Management System Implementation:**

Introduction

The Bus Management System is designed to manage bus records efficiently. It allows users to add, update, delete, and display bus information.

System Features

1. Add Bus: Add new bus records.

2. Update Route: Update existing bus routes.

3. Delete Bus: Delete bus records.

4. Display All Buses: Display all bus records.

5. Display Buses by Out Time: Display buses sorted by out time.

6. Display Buses by In Time: Display buses sorted by in time.

Details:

1. Create a Bus class with attributes: busNumber, route, inTime, outTime.

2. Create a BusManagementSystem class with methods: addBus, updateRoute, deleteBus, displayAllBuses, displayBusesByOutTime, displayBusesByInTime.

3. Implement the main method to interact with the user.

Code Snippets

// Bus class

public class Bus {

private String busNumber;

private String route;

private String inTime;

private String outTime;

// Getters and setters

}

// BusManagementSystem class

public class BusManagementSystem {

private List<Bus> buses;

// Methods

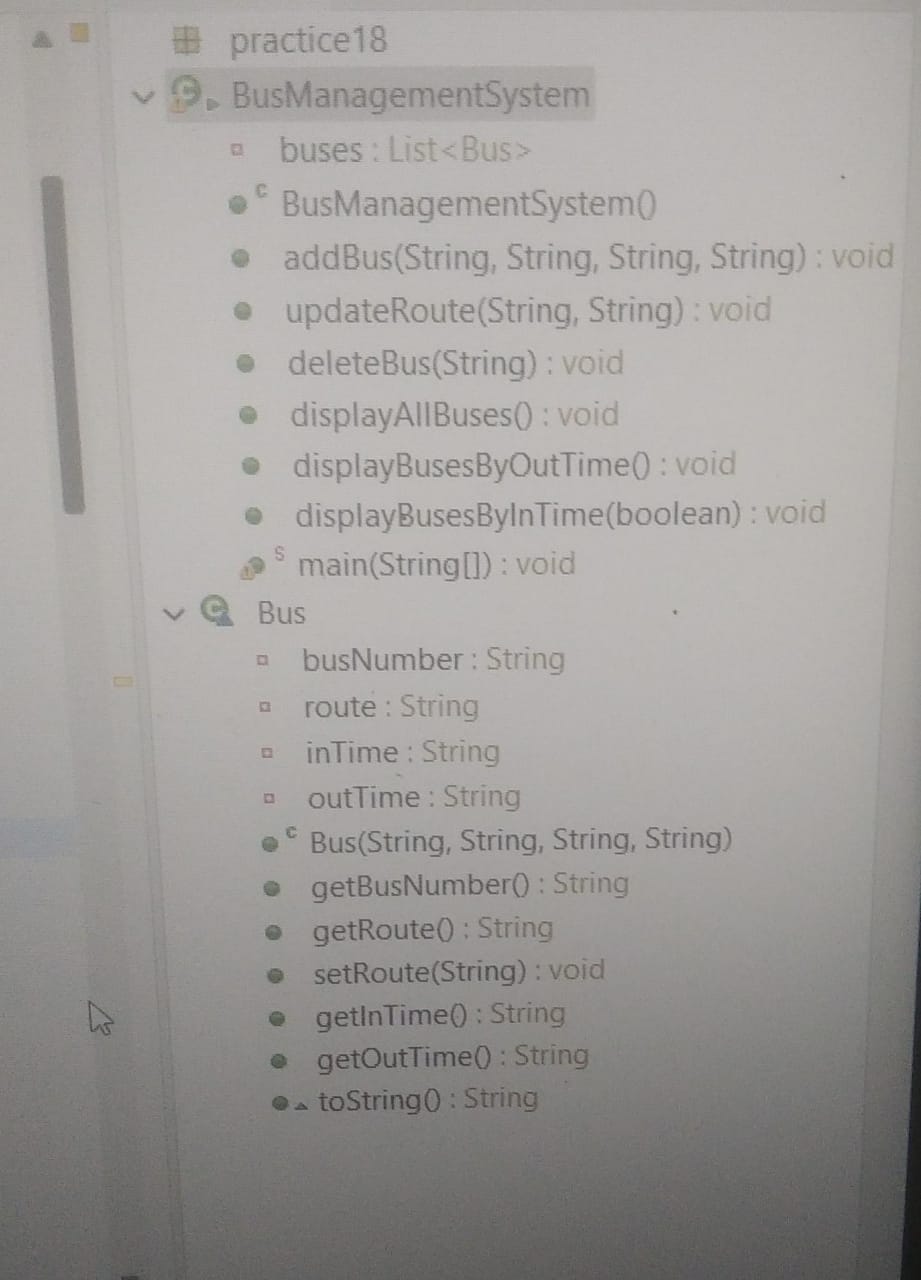
public static void main(String[] args) {

// Create instance and interact with user

}

}

Details of the variables and methods:



Sample Codes used:

package practice18;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

import java.util.Scanner;

public class BusManagementSystem {

private List<Bus> buses;

public BusManagementSystem() {

buses = new ArrayList<>();

// Sample data

buses.add(new Bus("101", "Route A", "08:00", "10:00"));

buses.add(new Bus("102", "Route B", "09:00", "11:00"));

buses.add(new Bus("103", "Route C", "07:30", "09:30"));

buses.add(new Bus("104", "Route D", "10:00", "12:00"));

buses.add(new Bus("105", "Route E", "06:30", "08:30"));

}

public void addBus(String number, String route, String inTime, String outTime) {

buses.add(new Bus(number, route, inTime, outTime));

}

public void updateRoute(String busNumber, String newRoute) {

for (Bus bus : buses) {

if (bus.getBusNumber().equals(busNumber)) {

bus.setRoute(newRoute);

break;

}

}

}

public void deleteBus(String number) {

buses.removeIf(bus -> bus.getBusNumber().equals(number));

}

public void displayAllBuses() {

buses.forEach(System.out::println);

}

public void displayBusesByOutTime() {

buses.stream()

.sorted(Comparator.comparing(Bus::getOutTime))

.forEach(System.out::println);

}

public void displayBusesByInTime(boolean ascending) {

if (ascending) {

buses.stream()

.sorted(Comparator.comparing(Bus::getInTime))

.forEach(System.out::println);

} else {

buses.stream()

.sorted(Comparator.comparing(Bus::getInTime).reversed())

.forEach(System.out::println);

}

}

public static void main(String[] args) {

BusManagementSystem system = new BusManagementSystem();

Scanner scanner = new Scanner(System.in);

system.displayBusesByOutTime();

System.out.println("Enter route name or bus number to update:");

String input = scanner.nextLine();

System.out.println("Enter new route:");

String newRoute = scanner.nextLine();

system.updateRoute(input, newRoute);

System.out.println("Enter bus number to delete:");

String busNumberToDelete = scanner.nextLine();

system.deleteBus(busNumberToDelete);

system.displayAllBuses();

System.out.println("Display buses by in time ascending (true/false):");

boolean ascending = scanner.nextBoolean();

system.displayBusesByInTime(ascending);

}

}

class Bus {

private String busNumber;

private String route;

private String inTime;

private String outTime;

public Bus(String busNumber, String route, String inTime, String outTime) {

this.busNumber = busNumber;

this.route = route;

this.inTime = inTime;

this.outTime = outTime;

}

public String getBusNumber() {

return busNumber;

}

public String getRoute() {

return route;

}

public void setRoute(String route) {

this.route = route;

}

public String getInTime() {

return inTime;

}

public String getOutTime() {

return outTime;

}

@Override

public String toString() {

return "Bus{" +

"busNumber='" + busNumber + '\'' +

", route='" + route + '\'' +

", inTime='" + inTime + '\'' +

", outTime='" + outTime + '\'' +

'}';

}

}

Output:

Displaying buses by out time:

Bus{busNumber='101', route='Route A', inTime='08:00', outTime='10:00'}

Bus{busNumber='102', route='Route B', inTime='09:00', outTime='11:00'}

...

Enter route name or bus number to update:

Enter new route:

Enter bus number to delete:

Displaying buses by in time ascending (true/false):

Conclusion:

The Bus Management System efficiently manages bus records, providing features for adding, updating, deleting, and displaying bus information. The system is implemented using Java and utilizes object-oriented programming principles.