SEMESTER PROJECT

OBJECT ORIENTED PROGRAMMING

SMART CITY RESOURCE MANAGEMENT SYSTEM

SUBMITTED TO: ***MS. SAJIDA KALSOOM***

SUBMITTED BY:

ALI RAZA (SP24-BCS-013)

ASHEER HADAYAT (SP24-BCS-020)

Smart City Resource Management System

Project Report

1. Introduction

Urbanization and population growth have led to increasing demands on city resources such as transportation, energy, and emergency services. Efficient management of these resources is critical for sustainable development, improved quality of life, and enhanced safety in smart cities. This project presents a comprehensive Smart City Resource Management System designed to monitor, manage, and optimize key urban resources including transport units, power stations, and emergency services.

The system is implemented in Java using an object-oriented approach, integrating real-time alerting, detailed reporting, and maintenance cost tracking. It aims to provide city administrators with actionable insights and automated emergency response capabilities to improve resource utilization and citizen safety.

2. Objectives

- Develop a modular and extensible system to represent and manage various city resources.
- Implement real-time monitoring and alerting mechanisms for emergencies such as power outages.
- Calculate and aggregate maintenance costs and usage metrics for informed decision-making.
- Provide detailed usage reports for transport, power, and emergency services.
- Simulate dynamic scenarios such as traffic changes and emergency dispatch.
- Support data persistence through serialization for long-term resource management.

3. System Architecture and Design

3.1. Overview

The system is designed using an object-oriented paradigm with a hierarchy of classes representing different city resources. Key design principles include modularity, extensibility, and separation of concerns. Interfaces are used to define common behaviors such as alerting and reporting.

3.2. Core Classes and Interfaces

- **CityResource (Abstract Base Class):**
  Serves as the superclass for all city resources. It encapsulates common attributes such as resource ID, location, status, and last update timestamp. It also maintains static fields to track global metrics like total resources, maintenance costs, passengers transported, and energy usage.
- **TransportUnit:**
  Models vehicles such as buses and other transport modes. Attributes include vehicle type, passenger capacity, fuel consumption rate, and current passenger count. It calculates maintenance costs based on vehicle type, fuel consumption, and usage.
- **PowerStation:**
  Represents power generation units with attributes like energy output, power type (e.g., solar), and connected consumers. It implements emergency alerting for outages and calculates maintenance costs based on energy output and consumer load.
- **EmergencyService:**
  Models emergency response units such as fire, medical, and police services. It tracks response times, calls handled, and on-duty status. It can send and respond to emergency alerts, simulating dispatch and response times.
- **Consumer:**
  Represents energy consumers connected to power stations, tracking consumption metrics.
- **ResourceHub and CityZone:**
  Organize resources geographically. ResourceHubs group transport units, while CityZones aggregate hubs, power stations, and emergency services for localized management.
- **Interfaces:**
  - *Alertable* defines emergency alert capabilities.
  - *Reportable* defines methods for generating usage reports and retrieving metrics.

3.3. Data Management

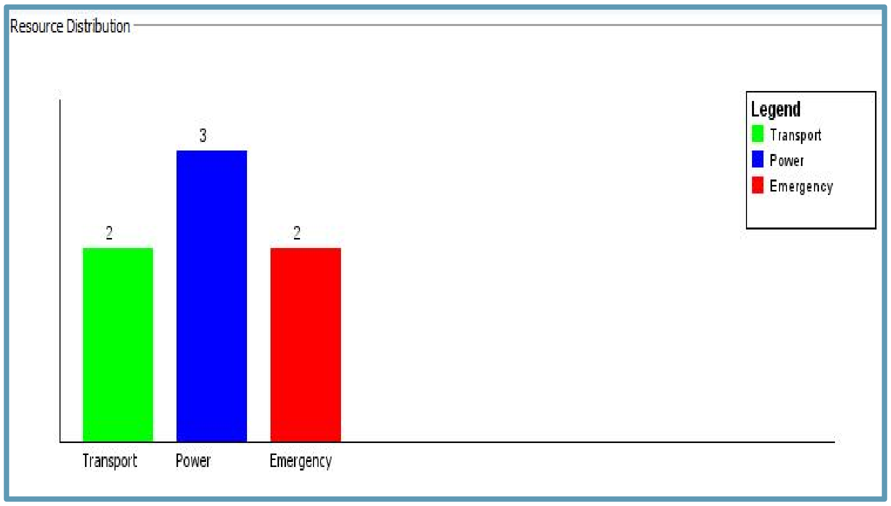
The system uses a repository pattern (`CityRepository`) for storing and retrieving city resources. Serialization is employed to persist resource states, enabling saving and loading of city data for long-term management.

---

4. Implementation Details

4.1. Transport Unit Management

Transport units are instantiated with specific vehicle types (e.g., Bus), passenger capacities, and fuel consumption rates. The system tracks the current number of passengers and adjusts this dynamically based on traffic conditions. For example, the `adjustRouteBasedOnTraffic (int change)` method modifies passenger counts and logs traffic updates in the GUI.



Maintenance cost calculation for transport units considers a base cost dependent on vehicle type, fuel consumption multiplied by a fuel rate, and usage cost proportional to passenger capacity. This cost contributes to the global maintenance cost metric.

4.2. Power Station Operations

Power stations maintain energy output levels and track connected consumers. They simulate power outages by changing status to "Outage" and sending emergency alerts. Upon outage detection, nearby emergency services are notified to respond.

Maintenance costs are calculated based on the power type with different base rates (e.g., solar vs. conventional) and scaled by the number of connected consumers. The system aggregates total energy consumption across all power stations.

4.3. Emergency Services

Emergency services handle dispatching and response to alerts such as power outages. Each unit has a response time and tracks the number of calls handled. When an emergency alert is received, the service status changes to "Responding" and after a timer simulating response duration, reverts to "Available."

Maintenance costs for emergency services include base costs varying by service type (fire, medical, police), equipment costs, and usage-based costs proportional to calls handled. The system aggregates total emergency responses and calculates average response times.

4.4. Reporting and Alerts

Each resource implements the `Reportable` interface, providing detailed usage reports including metrics like passenger counts, fuel consumption, energy output, response times, and maintenance costs. Reports are formed strings suitable for display in the GUI.

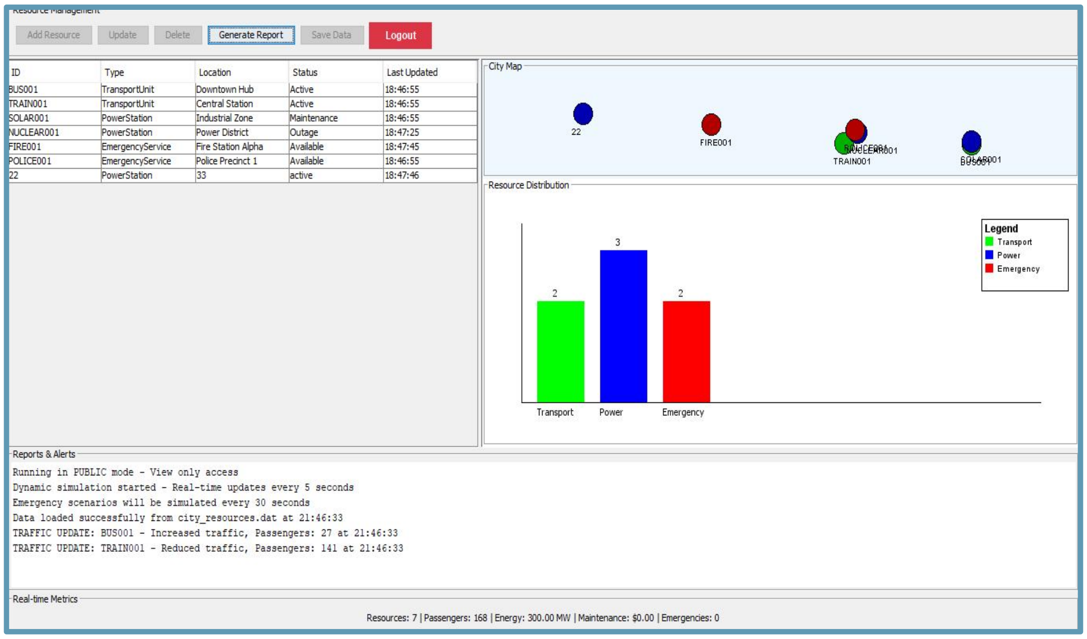
The `Alertable` interface enables resources to send emergency alerts. Alerts are timestamped and logged in a dedicated report area in the GUI, providing real-time monitoring of city resource status and emergencies.

---

5. User Interface

The system includes a graphical user interface (`SmartCityGUI`) built with Java Swing. The GUI features:

- A report area displaying real-time alerts, status updates, and usage reports.
- Controls for simulating traffic changes, power outages, and emergency dispatch.
- Visual feedback on resource statuses, enabling city administrators to monitor and react promptly.



ID	Type	Location	Status	Last Updated
BUS001	TransportUnit	Downtown Hub	Active	18:46:55
TRAIN001	TransportUnit	Central Station	Active	18:46:55
SOLAR001	PowerStation	Industrial Zone	Maintenance	18:46:55
NUCLEAR001	PowerStation	Power District	Outage	18:47:25
FIRE001	EmergencyService	Fire Station Alpha	Available	18:47:45
POLICE001	EmergencyService	Police Precinct 1	Available	18:46:55
22	PowerStation	33	active	18:47:46

```
Running in PUBLIC mode - View only access
Dynamic simulation started - Real-time updates every 5 seconds
Emergency scenarios will be simulated every 30 seconds
Data loaded successfully from city_resources.dat at 21:46:33
TRAFFIC UPDATE: BUS001 - Increased traffic, Passengers: 27 at 21:46:33
TRAFFIC UPDATE: TRAIN001 - Reduced traffic, Passengers: 141 at 21:46:33
```

Resources: 7 | Passengers: 168 | Energy: 300.00 MW | Maintenance: $0.00 | Emergencies: 0

The GUI enhances usability by providing an interactive platform for managing city resources and responding to emergencies.

---

6. Testing and Simulation

The system supports various test scenarios to validate functionality:

- **Traffic Simulation:** Adjust passenger counts on transport units to simulate increased or decreased traffic flow.
- **Power Outage Simulation:** Trigger outages in power stations to test emergency alerting and dispatch of emergency services.
- **Maintenance Cost Verification:** Calculate and verify maintenance costs for different resource types under various usage conditions.
- **Emergency Response:** Simulate emergency calls and track response times and service availability.

These simulations help ensure the system behaves as expected under realistic urban scenarios.

---

7. Results and Analysis

The system provides aggregated metrics and detailed reports that offer insights into city resource utilization:

- Total maintenance costs across all resources help budget planning.
- Passenger transport statistics inform transit capacity planning.
- Energy consumption data supports sustainable energy management.
- Emergency response metrics highlight service efficiency and readiness.

The alerting mechanism ensures timely responses to critical incidents, minimizing downtime and enhancing public safety.

---

8. Discussion

This project demonstrates the feasibility of integrating diverse city resources into a unified management system. Key strengths include:

- Modular design allows easy extension to new resource types.
- Real-time alerting and reporting improving situational awareness.
- Detailed cost and usage tracking aiding operational decisions.

Limitations include simplified location proximity checks and static response time simulation, which could be enhanced with GIS integration and real-time data feeds.

Future enhancements could be incorporated:

- AI-driven predictive maintenance based on historical data.
- IoT sensor integration for live resource monitoring.
- Advanced analytics dashboards for city planners.

---

9. Conclusion

The Smart City Resource Management System successfully models and manages multiple urban resources with detailed tracking, reporting, and emergency response capabilities. It provides a valuable tool for city administrators to optimize resource usage, reduce costs, and enhance citizen safety.

The system's extensible architecture and comprehensive feature set make it a strong foundation for future smart city initiatives.

---

10. References

- Java SE Documentation for serialization and Swing GUI components.
- Urban resource management literature on smart cities.
- Object-oriented design principles and design patterns.

---

11. Appendices

- Source code listings for core classes.
- Sample usage reports generated by the system.
- User manual for operating the GUI and simulating scenarios.
