Customer Support Ticketing System

Project Overview

The **Customer Support Ticketing System** is a Java-based mini project that simulates a real-world helpdesk application used in customer service environments. The system allows support agents or staff to manage customer issues effectively by using a **Priority Queue**, ensuring that high-priority issues are addressed before others.

Each customer issue is treated as a **support ticket**, which includes information such as:

- Ticket ID (unique identifier)
- Customer Name
- Issue Description
- Priority Level (1 = High, 2 = Medium, 3 = Low)

The system supports **CRUD operations**:

- **Create**: Add new tickets.
- **Read**: View all open tickets.
- **Update**: Resolve or update ticket status.
- **Delete**: Remove served tickets.

Efficient data structures like PriorityQueue and HashMap are used to manage ticket processing and lookup operations with optimal time complexity. This project demonstrates how algorithms and data structures are applied to solve real-world problems in customer service domains.

Key Features

- Create new support tickets with details and priority
- Automatically assigns a **unique Ticket ID**
- High-priority tickets are served before others using **PriorityQueue**
- Fast lookup and management of tickets using **HashMap**
- Mark tickets as resolved (Remove after being served)
- View all current tickets in order of priority

Technologies Used

- **Java (JDK 8 or later)**
- **Core Java OOPs**
- **Data Structures**:
    - PriorityQueue (for managing tickets based on priority)
    - HashMap (for storing and accessing ticket details)
- **Scanner Class** (for input handling)

Key Skills Developed

- Object-Oriented Programming (OOP)
- Problem-solving with Priority Queues
- CRUD application development
- Data structure implementation in real-world scenarios
- Efficient ticket ID generation and tracking
- Java collection framework mastery
- Console UI and input validation

Detailed Explanation :

- **Ticket Class**: Represents a support ticket with ID, customer name, issue description, and priority.
- **TicketManager Class**: Manages ticket creation, storage, resolution, and display using a PriorityQueue and a HashMap.
- **Main Class**: Provides a menu-based interface to interact with the system (e.g., create, view, resolve tickets).

Tickets are prioritized based on the priorityLevel field (1 = High, 2 = Medium, 3 = Low). The **PriorityQueue** ensures that the ticket with the highest priority is always at the front.

Conclusion :

The **Customer Support Ticketing System** is a practical and efficient implementation of a real-world problem using Java. It emphasizes the importance of data structures (like PriorityQueue and HashMap) and object-oriented principles in solving business problems.

This project gives learners a solid foundation in:

- Managing real-time data
- Structuring code for maintainability

- Understanding Java Collections and priority-based processing

The project can be expanded in the future with features like:

- GUI (Swing/JavaFX)
- Ticket categorization
- User login system
- Database integration (e.g., MySQL, SQLite)

Coding :

```
import java.util.*;

class Ticket implements Comparable<Ticket> {
    int id;
    String customerName;
    String issue;
    int priority; // 1 = High, 2 = Medium, 3 = Low

    public Ticket(int id, String customerName, String issue, int priority) {
        this.id = id;
        this.customerName = customerName;
        this.issue = issue;
        this.priority = priority;
    }

    @Override
    public int compareTo(Ticket other) {
        return Integer.compare(this.priority, other.priority); // lower number = higher priority
    }

    @Override
```

```
    public String toString() {
        return "Ticket ID: " + id + ", Name: " + customerName +
                ", Issue: " + issue + ", Priority: " + priority;
    }
}

public class SupportTicketSystem {
    private static int ticketCounter = 1001;
    private static PriorityQueue<Ticket> ticketQueue = new PriorityQueue<>();
    private static Map<Integer, Ticket> ticketMap = new HashMap<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void createTicket() {
        System.out.print("Enter customer name: ");
        String name = scanner.nextLine();
        System.out.print("Enter issue description: ");
        String issue = scanner.nextLine();
        System.out.print("Enter priority (1-High, 2-Medium, 3-Low): ");
        int priority = scanner.nextInt();
        scanner.nextLine(); // consume newline

        Ticket ticket = new Ticket(ticketCounter++, name, issue, priority);
        ticketQueue.offer(ticket);
        ticketMap.put(ticket.id, ticket);
        System.out.println("Ticket created successfully! Ticket ID: " + ticket.id + "\n");
    }

    public static void viewTickets() {
        if (ticketQueue.isEmpty()) {
```

```
            System.out.println("No open tickets.\n");
            return;
        }
        System.out.println("Open Tickets (in order of priority):");
        for (Ticket ticket : ticketQueue) {
            System.out.println(ticket);
        }
        System.out.println();
    }

    public static void serveTicket() {
        if (ticketQueue.isEmpty()) {
            System.out.println("No tickets to serve.\n");
            return;
        }
        Ticket served = ticketQueue.poll();
        ticketMap.remove(served.id);
        System.out.println("Serving Ticket:\n" + served + "\n");
    }

    public static void main(String[] args) {
        int choice;
        do {
            System.out.println("==== Support Ticket System ====");
            System.out.println("1. Create Ticket");
            System.out.println("2. View All Tickets");
            System.out.println("3. Serve Highest Priority Ticket");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
```

```
            choice = scanner.nextInt(); scanner.nextLine();

            switch (choice) {
                case 1: createTicket(); break;
                case 2: viewTickets(); break;
                case 3: serveTicket(); break;
                case 4: System.out.println("Exiting..."); break;
                default: System.out.println("Invalid choice. Try again.\n");
            }
        } while (choice != 4);
    }
}
```

Output :

```
==== Support Ticket System ====
1. Create Ticket
2. View All Tickets
3. Serve Highest Priority Ticket
4. Exit
Enter your choice: 1
Enter customer name: Alice
Enter issue description: Cannot login to portal
Enter priority (1-High, 2-Medium, 3-Low): 1
Ticket created successfully! Ticket ID: 1001

==== Support Ticket System ====
1. Create Ticket
2. View All Tickets
3. Serve Highest Priority Ticket
```

4. Exit

Enter your choice: 2

Open Tickets (in order of priority):

Ticket ID: 1001, Name: Alice, Issue: Cannot login to portal, Priority: 1

==== Support Ticket System ====

Enter your choice: 3

Serving Ticket:

Ticket ID: 1001, Name: Alice, Issue: Cannot login to portal, Priority: 1