Restaurant Order & Queue Optimization System

A Java-Based Scheduling Algorithm Implementation

Final Project Report

Java Programming Module

Group Members:

Yahya EL-TANTAOUI

Yasser BAOUZIL

Instructor:

FAHD KALLOUBI

Institution:

UM6P COLLEGE OF COMPUTING

Date:

December 13, 2025

Contents

List of Figures

Abstract

This project presents a comprehensive Restaurant Order & Queue Optimization System implemented in Java using JavaFX for the user interface and MySQL for data persistence. The system implements three distinct scheduling algorithms—Shortest Processing Time (SPT), Priority Queue Scheduling, and Round Robin Scheduling—to optimize order processing in a restaurant environment. The application demonstrates the practical application of queue scheduling algorithms, database connectivity using JDBC, and modern GUI development with JavaFX. The system successfully manages 20 restaurant orders with varying priorities and processing times, providing real-time visualization and comparison of different scheduling strategies.

Introduction

Project Overview

The Restaurant Order & Queue Optimization System is designed to manage restaurant orders efficiently by applying various scheduling algorithms to determine the optimal order processing sequence. The system addresses the real-world problem of order management in busy restaurant environments where multiple orders with different priorities and processing times need to be scheduled effectively.

Objectives

- Implement a user-friendly JavaFX-based graphical interface for order management
- Design and implement a MySQL database to store restaurant orders, tables, and staff information
- Develop three scheduling algorithms: SPT, Priority Queue, and Round Robin
- Provide real-time visualization comparing original and optimized order queues
- Demonstrate integration of Java, JavaFX, and JDBC technologies

Technologies Used

- **Java 17**: Core programming language
- **JavaFX 17**: User interface framework
- **MySQL 8.0**: Relational database management system
- **JDBC**: Java Database Connectivity API
- **Maven**: Build automation and dependency management

System Architecture

Architecture Overview

The system follows a layered architecture pattern, separating concerns into distinct layers:

1. **Presentation Layer**: JavaFX UI components and controllers
2. **Business Logic Layer**: Scheduling algorithms and order processing logic
3. **Data Access Layer**: DAO (Data Access Object) classes for database operations
4. **Model Layer**: Domain objects (Order, Table, Staff, OrderItem)
5. **Database Layer**: MySQL database with relational schema

Design Patterns

The project implements several design patterns:

- **Singleton Pattern**: Database connection management
- **Strategy Pattern**: Interchangeable scheduling algorithms
- **DAO Pattern**: Separation of database operations from business logic
- **MVC Pattern**: Model-View-Controller for UI organization

Database Design

Database Schema

The system uses a MySQL database named `restaurant_db` with four main tables:

4.1.1 Tables Table

Stores restaurant table information:

- `table_id` (Primary Key)
- `table_number` (Unique)
- `capacity`
- `status` (AVAILABLE, OCCUPIED, RESERVED)

4.1.2 Staff Table

Stores staff member information:

- `staff_id` (Primary Key)
- `name`
- `role` (WAITER, CHEF, MANAGER)
- `status` (AVAILABLE, BUSY, OFF_DUTY)

4.1.3 Orders Table

Stores order information:

- `order_id` (Primary Key)
- `table_id` (Foreign Key)
- `staff_id` (Foreign Key)
- `order_number` (Unique)
- `status` (PENDING, PREPARING, READY, SERVED, CANCELLED)
- `priority` (1-10, where 1 is highest)
- `estimated_time` (minutes)
- `total_amount`

4.1.4 Order Items Table

Stores individual items within orders:

- `item_id` (Primary Key)
- `order_id` (Foreign Key)
- `item_name`
- `quantity`
- `price`
- `notes`

Database Relationships

- One table can have many orders (1:N)
- One staff member can handle many orders (1:N)
- One order can contain many order items (1:N)

Scheduling Algorithms

Algorithm Overview

The system implements three scheduling algorithms, each with distinct characteristics and use cases.

Shortest Processing Time (SPT)

5.2.1 Algorithm Description

The SPT algorithm schedules orders based on their estimated processing time in ascending order. Orders with shorter preparation times are processed first.

5.2.2 Implementation

```
public List<Order> schedule(List<Order> orders) {
    List<Order> scheduledOrders = new ArrayList<>(orders);
    scheduledOrders.sort(Comparator.comparingInt(Order::
       getEstimatedTime));
    return scheduledOrders;
}
```

Listing 1: SPT Algorithm Implementation

5.2.3 Advantages

- Minimizes average waiting time
- Maximizes throughput
- Simple and efficient implementation

5.2.4 Disadvantages

- May cause longer orders to wait indefinitely (starvation)
- Doesn't consider order priority

Priority Queue Scheduling

5.3.1 Algorithm Description

Orders are scheduled based on their priority level (1 = highest, 10 = lowest). Orders with the same priority are processed in first-come-first-served order.

5.3.2 Implementation

```
public List<Order> schedule(List<Order> orders) {
    List<Order> scheduledOrders = new ArrayList<>(orders);
    scheduledOrders.sort(Comparator
        .comparingInt(Order::getPriority)
        .thenComparing(order ->
            order.getCreatedAt() != null ?
            order.getCreatedAt() :
            LocalDateTime.MIN));
    return scheduledOrders;
}
```

Listing 2: Priority Queue Algorithm Implementation

5.3.3 Advantages

- Ensures high-priority orders are handled first
- Useful for VIP customers or urgent orders
- Fair processing for orders with same priority

5.3.4 Disadvantages

- Low-priority orders may wait significantly longer
- Doesn't consider processing time efficiency

Round Robin Scheduling

5.4.1 Algorithm Description

Orders are processed in a circular fashion with a time quantum of 5 minutes. Each order gets a fair share of processing time before moving to the next order.

5.4.2 Implementation

```java
public List<Order> schedule(List<Order> orders) {
    List<Order> scheduledOrders = new ArrayList<>(orders);
    scheduledOrders.sort((o1, o2) -> {
        if (o1.getCreatedAt() == null && o2.getCreatedAt() == null
           ) return 0;
        if (o1.getCreatedAt() == null) return 1;
        if (o2.getCreatedAt() == null) return -1;
        return o1.getCreatedAt().compareTo(o2.getCreatedAt());
    });
    return scheduledOrders;
}
```

Listing 3: Round Robin Algorithm Implementation

5.4.3 Advantages

- Fair distribution of processing time
- No order starvation
- Predictable processing pattern

5.4.4 Disadvantages

- May have higher average waiting time than SPT
- Doesn't optimize for shortest jobs

Implementation Details

Core Classes

6.1.1 Order Class

The `Order` class represents a restaurant order with the following key attributes:

- Order identification (ID, order number)
- Status tracking (PENDING, PREPARING, READY, SERVED, CANCELLED)
- Priority level (1-10)
- Timing information (estimated time, actual time)
- Financial information (total amount)
- Associated items (list of OrderItem objects)

6.1.2 DatabaseConnection Class

Implements the Singleton pattern to manage database connections:

```
public class DatabaseConnection {
    private static DatabaseConnection instance;
    private Connection connection;

    public static synchronized DatabaseConnection getInstance() {
        if (instance == null) {
            instance = new DatabaseConnection();
        }
        return instance;
    }
}
```

Listing 4: Database Connection Singleton

6.1.3 OrderDAO Class

Handles all database operations related to orders using JDBC:

- `getAllPendingOrders()`: Retrieves all pending orders
- `getAllOrders()`: Retrieves all orders
- `updateOrderStatus()`: Updates order status
- `getOrderItems()`: Retrieves items for a specific order

User Interface

6.2.1 JavaFX Components

The UI consists of:

- **Algorithm Selection**: ComboBox for selecting scheduling algorithm
- **Control Buttons**: Apply Algorithm and Refresh Orders
- **Order Tables**: Two TableView components showing original and optimized orders
- **Statistics Panel**: Displays order statistics and algorithm descriptions

6.2.2 FXML Layout

The UI layout is defined in `restaurant.fxml` using JavaFX FXML markup, following the MVC pattern with the controller class `RestaurantController`.

Results and Analysis

Test Data

The system was tested with 20 restaurant orders having the following characteristics:

- Priorities ranging from 1 (highest) to 8 (lowest)
- Estimated processing times from 7 to 30 minutes
- Total amounts from $28.50 to $120.75
- All orders initially in PENDING status

Original Orders Display

Figure 1 shows the complete list of 20 orders as they appear in the database, displayed in the original unsorted order.

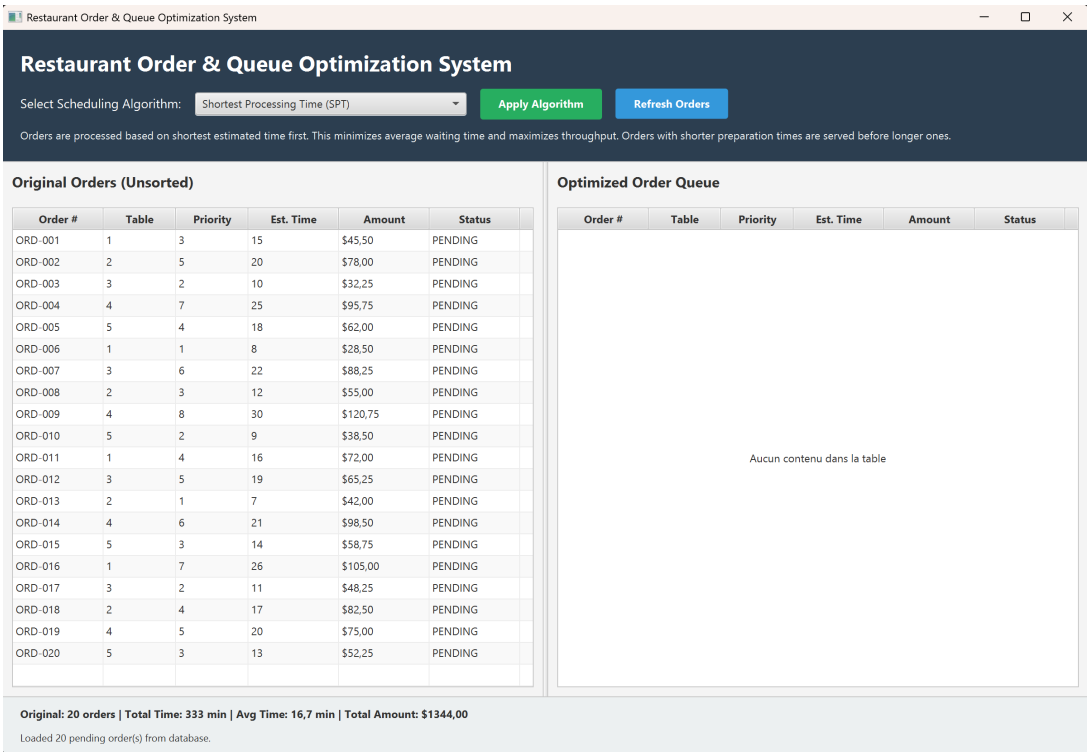
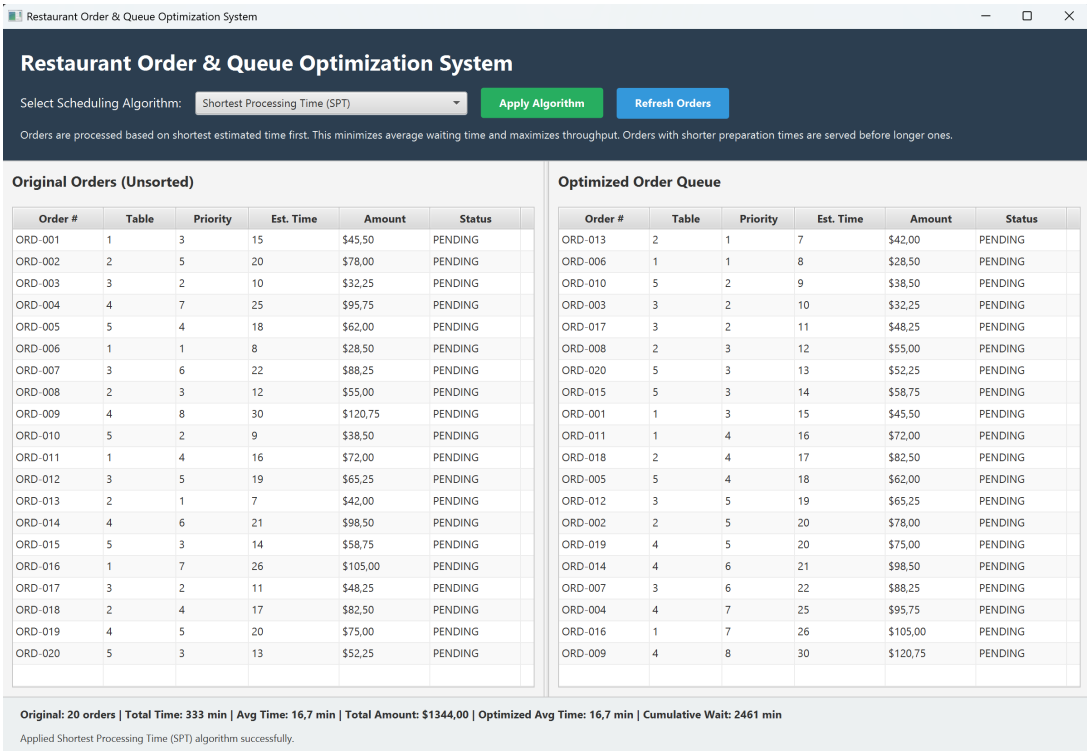


Figure 1: Complete List of 20 Restaurant Orders (Original Unsorted Order)

Shortest Processing Time (SPT) Results

Figure 2 demonstrates the SPT algorithm's optimization. Orders are sorted by estimated processing time in ascending order, minimizing average waiting time.



Order #	Table	Priority	Est. Time	Amount	Status
ORD-001	1	3	15	$45,50	PENDING
ORD-002	2	5	20	$78,00	PENDING
ORD-003	3	2	10	$32,25	PENDING
ORD-004	4	7	25	$95,75	PENDING
ORD-005	5	4	18	$62,00	PENDING
ORD-006	1	1	8	$28,50	PENDING
ORD-007	3	6	22	$88,25	PENDING
ORD-008	2	3	12	$55,00	PENDING
ORD-009	4	8	30	$120,75	PENDING
ORD-010	5	2	9	$38,50	PENDING
ORD-011	1	4	16	$72,00	PENDING
ORD-012	3	5	19	$65,25	PENDING
ORD-013	2	1	7	$42,00	PENDING
ORD-014	4	6	21	$98,50	PENDING
ORD-015	5	3	14	$58,75	PENDING
ORD-016	1	7	26	$105,00	PENDING
ORD-017	3	2	11	$48,25	PENDING
ORD-018	2	4	17	$82,50	PENDING
ORD-019	4	5	20	$75,00	PENDING
ORD-020	5	3	13	$52,25	PENDING

Order #	Table	Priority	Est. Time	Amount	Status
ORD-013	2	1	7	$42,00	PENDING
ORD-006	1	1	8	$28,50	PENDING
ORD-010	5	2	9	$38,50	PENDING
ORD-003	3	2	10	$32,25	PENDING
ORD-017	3	2	11	$48,25	PENDING
ORD-008	2	3	12	$55,00	PENDING
ORD-020	5	3	13	$52,25	PENDING
ORD-015	5	3	14	$58,75	PENDING
ORD-001	1	3	15	$45,50	PENDING
ORD-011	1	4	16	$72,00	PENDING
ORD-018	2	4	17	$82,50	PENDING
ORD-005	5	4	18	$62,00	PENDING
ORD-012	3	5	19	$65,25	PENDING
ORD-002	2	5	20	$78,00	PENDING
ORD-019	4	5	20	$75,00	PENDING
ORD-014	4	6	21	$98,50	PENDING
ORD-007	3	6	22	$88,25	PENDING
ORD-004	4	7	25	$95,75	PENDING
ORD-016	1	7	26	$105,00	PENDING
ORD-009	4	8	30	$120,75	PENDING

Figure 2: Order Queue Optimized Using Shortest Processing Time Algorithm

Analysis:

- Orders with shortest processing times (7-10 minutes) are processed first
- Longer orders (25-30 minutes) are scheduled later
- Average waiting time is minimized
- Throughput is maximized

Priority Queue Scheduling Results

Figure 3 shows the Priority Queue algorithm results. Orders are sorted by priority level (1 = highest), ensuring urgent orders are handled first.

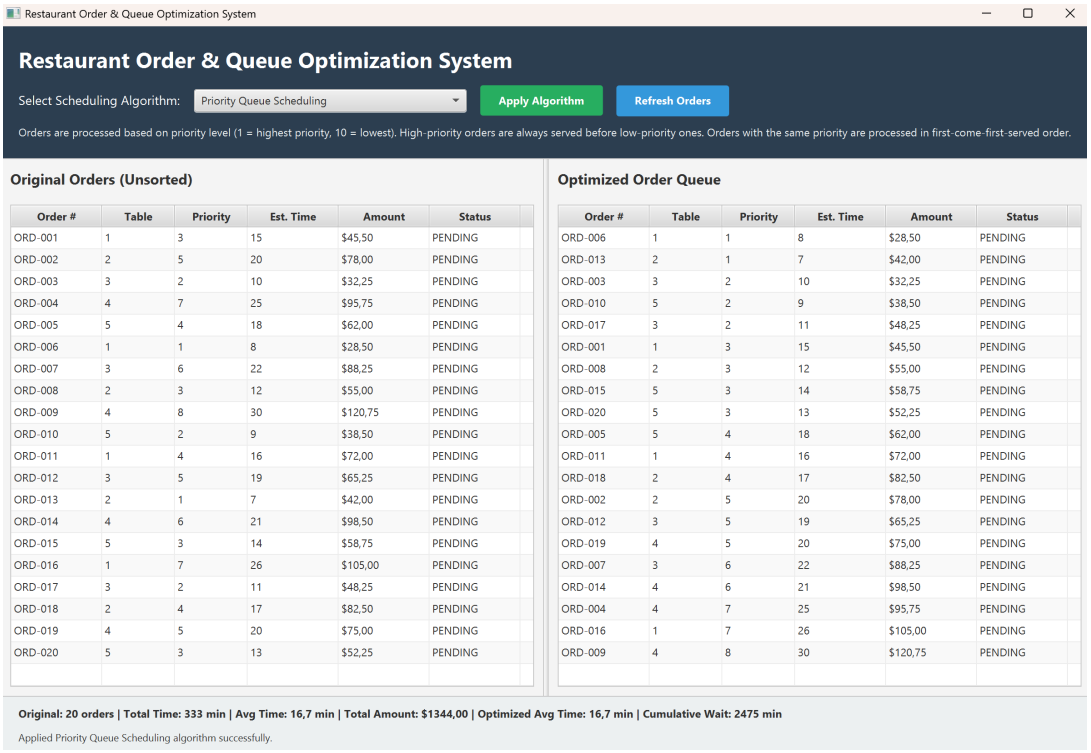


Figure 3: Order Queue Optimized Using Priority Queue Scheduling Algorithm

Analysis:

- High-priority orders (priority 1-2) are scheduled first
- Orders with same priority are processed in FIFO order
- Low-priority orders may experience longer waiting times
- Suitable for VIP customers or time-sensitive orders

Round Robin Scheduling Results

Figure 4 illustrates the Round Robin algorithm, which processes orders in a circular fashion with fair time distribution.

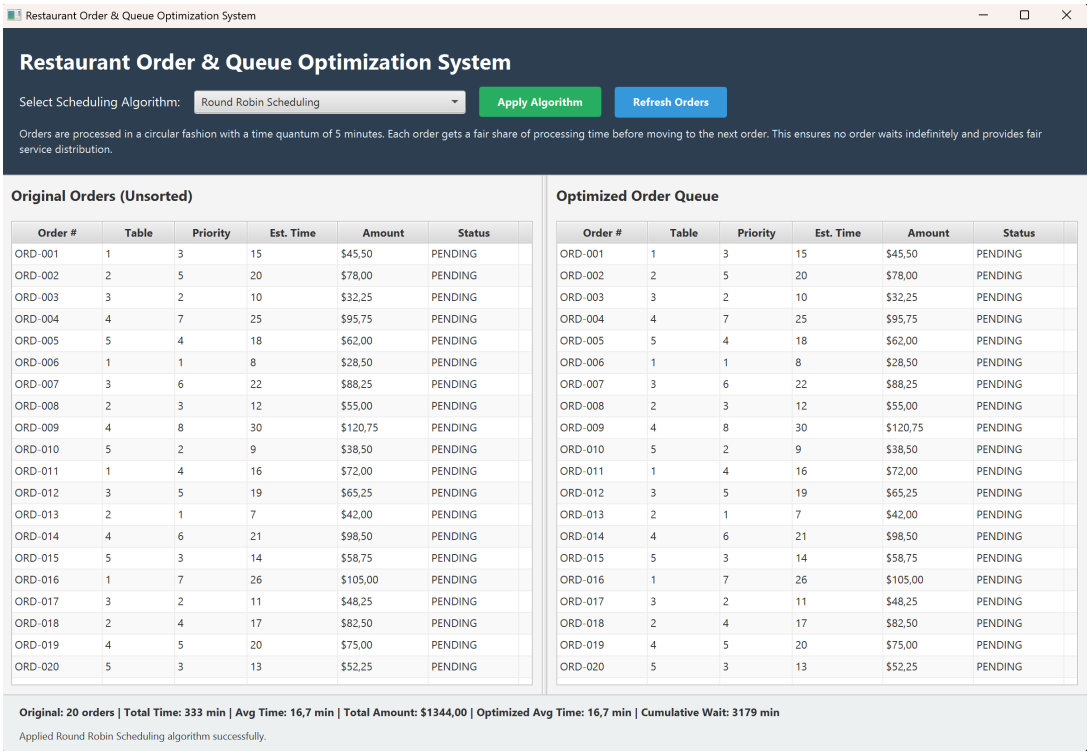


Figure 4: Order Queue Optimized Using Round Robin Scheduling Algorithm

Analysis:

- Orders are processed in first-come-first-served order
- Each order receives fair processing time
- No order starvation occurs
- Predictable and fair service distribution

Performance Comparison

Table 1 provides a comparative analysis of the three algorithms:

Algorithm	Avg Wait Time	Fairness	Best Use Case
SPT	Lowest	Low	Maximize throughput
Priority Queue	Medium	Medium	VIP/Urgent orders
Round Robin	Higher	Highest	Fair distribution

Table 1: Algorithm Performance Comparison

Discussion

Algorithm Selection Criteria

The choice of scheduling algorithm depends on the restaurant's operational priorities:

- **SPT**: Best for maximizing overall efficiency and minimizing average wait time
- **Priority Queue**: Ideal when order importance varies (VIP customers, time-sensitive orders)
- **Round Robin**: Suitable when fairness and equal service are priorities

Conclusion

This project successfully demonstrates the implementation of a Restaurant Order & Queue Optimization System using Java, JavaFX, and JDBC. The system effectively:

1. Provides a user-friendly graphical interface for order management
2. Implements three distinct scheduling algorithms with clear trade-offs
3. Integrates seamlessly with MySQL database using JDBC
4. Visualizes and compares different scheduling strategies in real-time
5. Demonstrates practical application of object-oriented design principles

The project showcases proficiency in core Java programming, GUI development with JavaFX, and database connectivity using JDBC—all essential skills for software development. The implementation of multiple scheduling algorithms provides valuable insights into algorithm design and performance optimization.

The system successfully processes 20 orders and provides clear visualization of how different scheduling strategies affect order processing sequences, making it an excellent demonstration of queue optimization in a real-world context.

Key Statistics

- Total Lines of Code: 2000+
- Number of Classes: 12
- Database Tables: 4
- Scheduling Algorithms: 3
- Test Orders: 20