CPU Scheduling Algorithms Report

Executive Summary

This report analyzes a JavaFX-based CPU scheduling algorithms simulator developed as part of an operating systems coursework. The application implements five major CPU scheduling algorithms: First Come First Serve (FCFS), Shortest Job First (SJF), Shortest Remaining Time (SRT), Round Robin (RR), and Multilevel Feedback Queue (MLFQ). The system provides both tabular results and visual Gantt charts to demonstrate algorithm behavior.

Project Architecture

1. Application Structure

The project follows a Model-View-Controller (MVC) architecture pattern:

- **App.java**: Main application entry point that initializes the JavaFX environment
- **Controller.java**: Handles user interface logic and algorithm execution
- **Process.java**: Data model representing individual processes
- **GanttChart.java**: Utility class for generating visual Gantt charts
- **CPU.fxml**: FXML layout file defining the user interface structure

2. Technology Stack

- **JavaFX**: For the graphical user interface
- **FXML**: For declarative UI layout
- **Java Collections**: For data management (ObservableList, LinkedList, Queue)
- **Maven**: Build and dependency management (inferred from structure)

Algorithm Implementations

1. First Come First Serve (FCFS)

Implementation Location: `Controller.calculateFCFS()`

Algorithm Logic:

- Processes execute in order of arrival
- Non-preemptive scheduling
- Simple sequential execution with completion time tracking

Key Features:

- Handles arrival time gaps by advancing current time
- Calculates turnaround time, waiting time, and response time
- Generates fixed-width Gantt chart visualization

2. Shortest Job First (SJF) - Non-Preemptive

Implementation Location: `Controller.calculateSJF()`

Algorithm Logic:

- Selects process with shortest burst time among arrived processes
- Uses boolean array to track completed processes
- Handles cases where no processes have arrived

Strengths:

- Correctly implements SJF selection criteria
- Proper handling of process arrival times
- Accurate time calculations

3. Shortest Remaining Time (SRT) - Preemptive

Implementation Location: `Controller.calculateSRT()`

Algorithm Logic:

- Preemptive version of SJF
- Continuously selects process with minimum remaining time
- Tracks first response time for each process
- Executes processes in 1-time-unit increments

Technical Implementation:

- Uses remaining time array for preemption tracking
- Boolean array tracks first execution for response time calculation
- Time-unit-by-time-unit execution simulation

4. Round Robin (RR)

Implementation Location: `Controller.calculateRR()`

Algorithm Logic:

- Queue-based implementation with configurable time quantum
- Handles process arrivals during execution
- Manages process re-queuing after time quantum expiration

Notable Features:

- Dynamic queue management for newly arrived processes
- Proper handling of partial time quantum execution
- Time quantum validation with user feedback

5. Multilevel Feedback Queue (MLFQ)

Implementation Location: `Controller.calculateMLFQ()`

Algorithm Logic:

- Three-queue system with increasing time quantums
- Queue 1 and Queue 2 use Round Robin with different time quantums
- Queue 3 uses FCFS (configurable via dropdown)
- Process demotion between queues

Configuration:

- User-defined time quantums for Queue 1 and Queue 2
- Validation ensures Queue 2 time quantum > Queue 1 time quantum
- Dropdown selection for Queue 3 algorithm (currently only FCFS)

User Interface Design

1. Layout Structure

The interface uses a dual-pane design:

- **Left Pane**: Input controls and algorithm selection
- **Right Pane**: Results display with table and Gantt chart

2. Dynamic UI Management

The controller implements sophisticated UI state management:

- `Disable()`: Hides all input controls
- `EnableFCFS_SJF_SRT()`: Shows basic arrival/burst time inputs
- `EnableRR()`: Adds time quantum input field
- `EnableMLFQ()`: Displays three-queue configuration options

3. Input Validation

Multiple validation layers:

- Space-separated input parsing with comma rejection
- Numeric validation for time values
- MLFQ-specific validation for queue parameters
- Equal-length validation for arrival and burst time arrays

Gantt Chart Visualization

1. Design Philosophy

All Gantt charts use fixed-width blocks (50x50 pixels) rather than proportional scaling:

- Consistent visual representation across algorithms
- Simplified layout management
- Clear process identification

2. Algorithm-Specific Features

- **FCFS/SJF**: Sequential blocks with process completion times

- **SRT**: Single time-unit blocks showing preemptive switching
- **RR**: Blocks representing time quantum executions
- **MLFQ**: Color-coded blocks with idle time representation

3. Visual Elements

- Process labels centered within blocks
- Time markers below each block
- Color coding: Light blue (FCFS), Light green (SJF/SRT), Sky blue (RR/MLFQ), Gray (Idle)

Conclusion

This CPU scheduling simulator successfully demonstrates the behavior of five fundamental scheduling algorithms through both numerical results and visual representations. The implementation shows solid understanding of operating systems concepts and JavaFX development. While there are opportunities for enhancement in areas such as proportional visualization and additional algorithm support, the current implementation provides a valuable educational tool for understanding CPU scheduling behavior.

The project demonstrates competent software engineering practices with its clear architecture, comprehensive error handling, and user-friendly interface design. The dynamic UI adaptation and robust input validation contribute to a professional-quality application suitable for educational use.

Team member

```
Chin Hongnyheng
Virak Rith
```