1. FCFSScheduler.java

- **Mục đích**: Triển khai thuật toán First-Come First-Serve (FCFS), trong đó các tiến trình được thực thi theo thứ tự chúng đến.
- **Phiên bản cũ**:
    - Triển khai cơ bản các phương thức generateGanttChart và calculateMetrics.
    - Quản lý tiến trình đơn giản và thiếu sắp xếp rõ ràng.
    - Ít hoặc không có bình luận và tài liệu.
- **Phiên bản mới**:
    - **Constructor mới**: Thêm constructor nhận danh sách tiến trình (super(processList)), cung cấp tính linh hoạt để khởi tạo bộ lập lịch với dữ liệu định sẵn.
    - **Sắp xếp rõ ràng**: Danh sách tiến trình được sắp xếp theo thời gian đến trước khi tính toán các chỉ số, đảm bảo tính chính xác và nhất quán.
    - **Tài liệu chi tiết**: Thêm các bình luận Javadoc chi tiết cho các phương thức và lớp, giải thích mục đích, cách sử dụng và logic rõ ràng.
    - **Cải tiến logic**: Cải thiện xử lý cập nhật thời gian khi tiến trình đến sau thời gian hiện tại của mô phỏng.
- **Thay đổi chính**:
    - Tăng tính bảo trì và module.
    - Cải thiện rõ ràng thông qua tài liệu chi tiết.
    - Đảm bảo tính chính xác logic bằng cách sắp xếp trước các tiến trình.

---

2. GanttEntry.java

- **Mục đích**: Đại diện cho một mục trong biểu đồ Gantt, lưu trữ ID tiến trình, thời gian bắt đầu và thời gian kết thúc.
- **Phiên bản cũ**:
    - Các phương thức getter, setter và constructor đơn giản để quản lý các khoảng thời gian thực thi tiến trình.
    - Tài liệu hạn chế hoặc không có.
- **Phiên bản mới**:
    - **Javadoc chi tiết**: Mỗi trường, phương thức và constructor hiện có các bình luận chi tiết giải thích mục đích và tham số.
    - **Đồng nhất trong cách đặt tên**: Các trường và phương thức tuân thủ chuẩn đặt tên thống nhất theo tiêu chuẩn Java.
    - **Không thay đổi chức năng**: Logic và chức năng vẫn giữ nguyên.
- **Thay đổi chính**:
    - Tăng khả năng đọc và sử dụng thông qua tài liệu chi tiết.
    - Không sửa đổi logic cốt lõi.

---

3. Process.java

- **Mục đích**: Đại diện cho một tiến trình trong hệ thống lập lịch, bao gồm các thuộc tính như arrivalTime, burstTime, priority và các chỉ số như waitingTime và turnaroundTime.
- **Phiên bản cũ**:
    - Các phương thức getter và setter cơ bản để truy cập các thuộc tính tiến trình.
    - Tài liệu hạn chế.
- **Phiên bản mới**:

    - **Tài liệu chi tiết**: Thêm các bình luận Javadoc cho tất cả các thuộc tính, phương thức và constructor.
    - **Cải thiện quản lý remainingTime**: Trường remainingTime, quan trọng đối với các thuật toán như Round Robin, được khởi tạo và cập nhật rõ ràng.
    - **Mô tả tham số chi tiết**: Constructor và setter bao gồm các mô tả chi tiết về tham số, giúp dễ hiểu hơn.
- **Thay đổi chính**:
    - Không thay đổi chức năng, đảm bảo tương thích ngược.
    - Cải thiện sự hiểu biết của lập trình viên với các giải thích chi tiết.

---

4. RRScheduler.java

- **Mục đích**: Triển khai thuật toán Round Robin sử dụng quantum để phân bổ các khoảng thời gian cho các tiến trình.
- **Phiên bản cũ**:
    - Logic nội tuyến cho tạo biểu đồ Gantt và tính toán chỉ số.
    - Thiếu tính module, với các tính toán dư thừa được nhúng trong nhiều phương thức.
- **Phiên bản mới**:
    - **Phương thức runSimulation**: Logic lập lịch được tách thành một phương thức riêng biệt, tập trung và đơn giản hóa logic cốt lõi.
    - **Cờ isSimulated**: Được giới thiệu để đảm bảo mô phỏng chỉ chạy một lần, tránh các tính toán không cần thiết.
    - **Xử lý chi tiết biểu đồ Gantt**: Cải thiện logic để tạo các mục Gantt, đặc biệt cho các tiến trình bị ngắt quãng bởi quantum.
    - **Tài liệu chi tiết**: Thêm các giải thích chi tiết cho các phương thức và các bước chính của thuật toán.

- **Thay đổi chính**:
    - Tăng hiệu suất thông qua tính module và giảm tính toán dư thừa.
    - Cải thiện bảo trì với sự tách biệt rõ ràng về logic.
    - Dễ dàng gỡ lỗi và mở rộng với logic mô phỏng tập trung.

---

5. Scheduler.java

- **Mục đích**: Lớp trừu tượng định nghĩa các thuộc tính và phương thức chung cho tất cả các thuật toán lập lịch.
- **Phiên bản cũ**:
    - Các phương thức tối thiểu để thêm tiến trình và định nghĩa trừu tượng cho tính toán chỉ số và tạo biểu đồ Gantt.
- **Phiên bản mới**:
    - **Tạo danh sách tiến trình mặc định**: Thêm phương thức createDefaultProcessList để tạo các tiến trình mẫu phục vụ kiểm thử và trình diễn.
    - **Tài liệu chi tiết**: Thêm các bình luận chi tiết cho các phương thức trừu tượng và các hàm tiện ích chung.
    - **Cải thiện ví dụ sử dụng**: Tạo tiến trình mặc định giúp nhanh chóng trình diễn và kiểm thử các lớp bộ lập lịch kế thừa.
- **Thay đổi chính**:
    - Hỗ trợ kiểm thử và trình diễn dễ dàng với danh sách tiến trình mặc định.
    - Cải thiện sự hiểu biết về mục đích và cách sử dụng của lớp.

---

6. SJNScheduler.java

- **Mục đích**: Triển khai thuật toán Shortest Job Next (SJN), ưu tiên các tiến trình có thời gian burst ngắn nhất.
- **Phiên bản cũ**:
    - Logic tạo biểu đồ Gantt và tính toán chỉ số được nhúng bên trong.
    - Xử lý lỗi hạn chế đối với các kịch bản không có tiến trình nào sẵn sàng.
- **Phiên bản mới**:
    - **Phương thức runSimulation**: Tách riêng logic lập lịch chính vào một phương thức dành riêng để tăng tính module.
    - **Xử lý biểu đồ Gantt**: Thêm thuộc tính ganttChart để lưu trữ kết quả, tránh tính toán dư thừa.
    - **Xử lý lỗi**: Cải thiện xử lý các tình huống không có tiến trình sẵn sàng bằng cách tăng thời gian mô phỏng một cách phù hợp.
    - **Tài liệu chi tiết**: Thêm các bình luận giải thích chi tiết logic và mục đích của từng bước.

- **Thay đổi chính**:
    - Tăng cường tính chắc chắn và hiệu suất.
    - Module hóa logic lập lịch để cải thiện tính bảo trì.

Tóm tắt Cập nhật

1. **Cải thiện tài liệu**:
    - Thêm các bình luận Javadoc trên toàn bộ các tệp, nâng cao khả năng đọc hiểu và sự hiểu biết của lập trình viên.
2. **Module hóa**:

- Tách logic mô phỏng vào các phương thức riêng trong RRScheduler và SJNScheduler, cải thiện khả năng bảo trì.

3. **Tính linh hoạt**:
   - Thêm các constructor và phương thức tiện ích như createDefaultProcessList để đơn giản hóa việc kiểm thử và tái sử dụng.
4. **Hiệu suất**:
   - Tránh các hoạt động dư thừa bằng cách giới thiệu các cờ (ví dụ: isSimulated) và tập trung hóa logic.