1. ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

----------------------------------



**BÁO CÁO MINI-PROJECT**

Môn: Lập Trình Hướng Đối Tượng

*Mã học phần: IT3103*

*Mã lớp học: 151965*

*GVHD: Nguyễn Thị Thu Trang*

*Nhóm 10*

|  |  |
| --- | --- |
| Trần Mạnh Hùng | 20226108 |
| Đỗ Quang Huy | 20225854 |
| Đỗ Lường Hương | 20225851 |
| Vương Quốc Huy | 20225637 |
| Lê Hồ Quang Huy | 20225636 |

*Hà Nội, 24/12/2024*

# Thành viên

* Đỗ Quang Huy : Xây dựng mã nguồn cho lớp Round Robin, xây dựng sơ đồ use case(15%)
* Đỗ Lường Hương : Xây dựng mã nguồn cho lớp FCFS(15%)
* Vương Quốc Huy : Xây dựng mã nguồn cho lớp SJN(15%)
* Lê Hồ Quang Huy: Phân chia công việc, xây dựng sơ đồ lớp , xây dựng lớp App xây dựng lớp MainMenuController và SimulationController(40%)
* Trần Mạnh Hùng: xây dựng lớp Process, lớp trừu tượng *Scheduler*(15%)

## 

# Đề tài

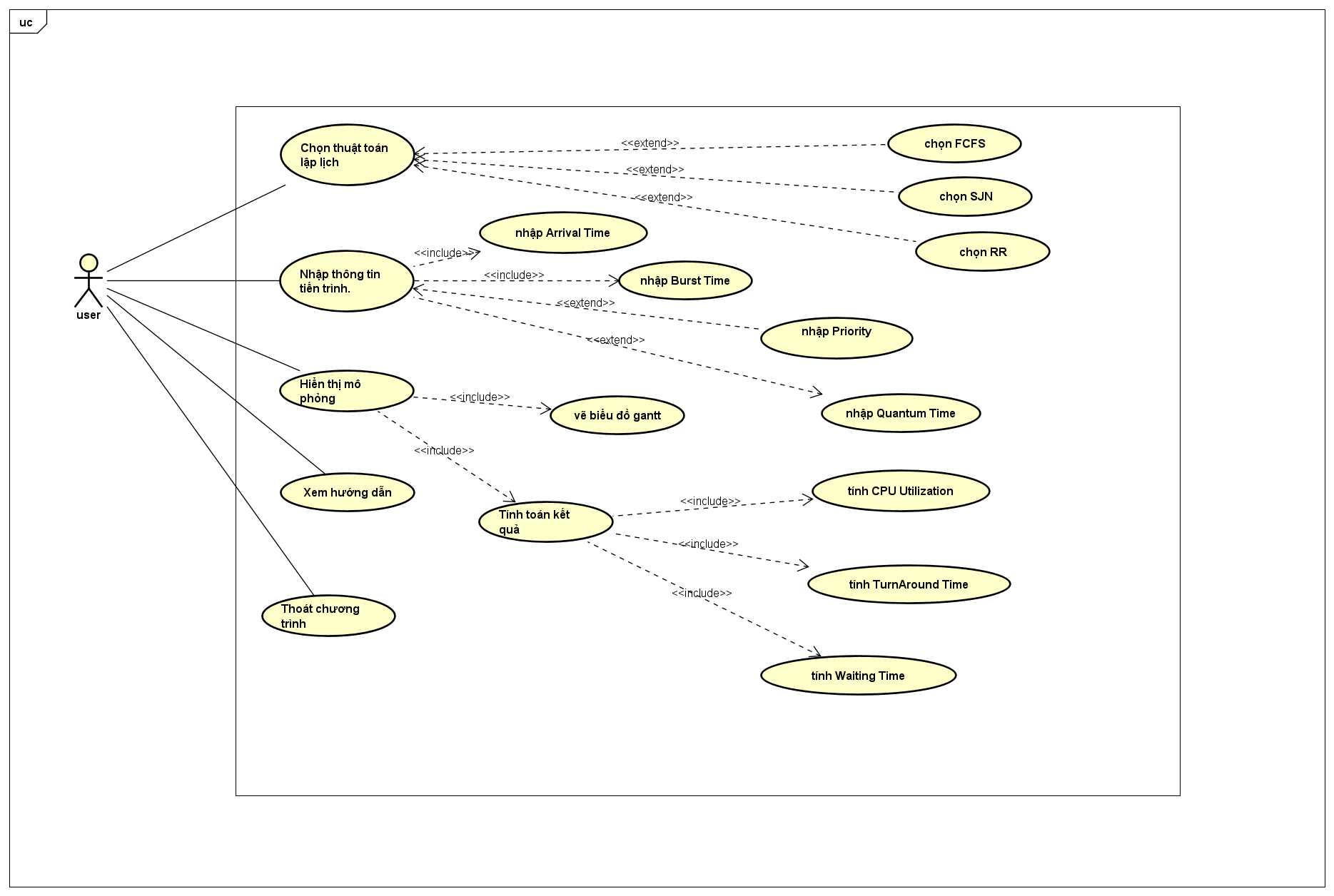
## Tên đề tài

## Simulation of CPU Scheduling Algorithms ( Mô phỏng thuật toán lập lịch CPU)

## Mô tả chi tiết yêu cầu của dự án

Dự án này mô phỏng các thuật toán lập lịch CPU khác nhau, như **FCFS (First-Come-First-Serve)**, **SJN (Shortest Job Next)** và **RR (Round Robin)**. Nó minh họa cách mỗi thuật toán xử lý việc thực thi các tiến trình, ảnh hưởng đến các chỉ số hệ thống như **thời gian chờ** và **sử dụng CPU**.

## Use case diagram



### **1. Actor chính:**

* **User (Người dùng):** Là người sử dụng hệ thống, có thể thực hiện các thao tác chính như chọn thuật toán, nhập thông tin tiến trình, xem mô phỏng, và tính toán kết quả.

### **2. Use Case chính:**

#### **2.1. Chọn thuật toán lập lịch**

* Người dùng có thể chọn một trong ba thuật toán:
  + **FCFS (First-Come, First-Serve):** Tiến trình được xử lý theo thứ tự đến.
  + **SJN (Shortest Job Next):** Ưu tiên tiến trình có thời gian xử lý ngắn nhất.
  + **RR (Round Robin):** Phân chia thời gian xử lý theo quantum (vòng tròn).

#### **2.2. Nhập thông tin tiến trình**

* **Bao gồm:**
  + **Arrival Time:** Thời gian tiến trình đến hệ thống.
  + **Burst Time:** Thời gian cần để hoàn thành tiến trình.
  + **Priority:** Độ ưu tiên của tiến trình (nếu sử dụng thuật toán ưu tiên).
  + **Quantum Time:** Thời gian quantum áp dụng trong Round Robin.

#### **2.3. Hiển thị mô phỏng**

* Mô phỏng trực quan cách hệ thống thực thi các tiến trình:
  + **Biểu đồ Gantt:** Hiển thị thứ tự và khoảng thời gian mỗi tiến trình được xử lý.
  + **Trạng thái tiến trình:** Cập nhật thời gian thực khi các tiến trình được xử lý.

#### **2.4. Tính toán kết quả**

* Kết quả được tính toán và hiển thị:
  + **CPU Utilization:** Hiệu suất sử dụng CPU.
  + **Waiting Time:** Tổng thời gian chờ của từng tiến trình.
  + **Turnaround Time:** Tổng thời gian từ khi tiến trình đến cho đến khi hoàn thành.

#### **2.5. Xem hướng dẫn**

* Người dùng có thể xem các giải thích chi tiết về:
  + Cách hoạt động của từng thuật toán.
  + Ý nghĩa các chỉ số như Waiting Time, Turnaround Time, và CPU Utilization.

#### **2.6. Thoát chương trình**

* Người dùng có thể chọn thoát ứng dụng. Hệ thống sẽ yêu cầu xác nhận trước khi đóng.

### **3. Quan hệ giữa các Use Case**

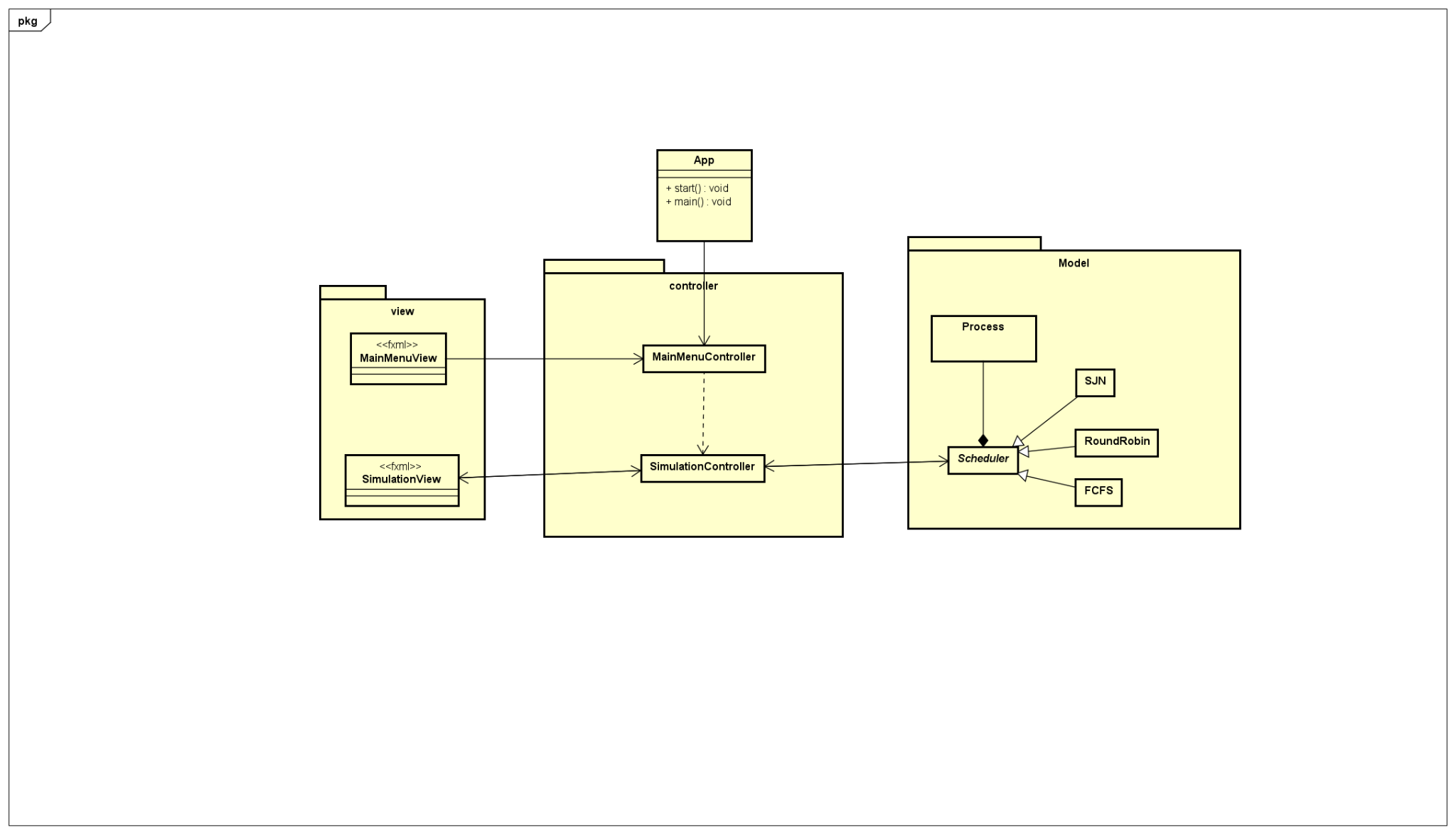
* **<<include>>:**
  + Một số chức năng được tái sử dụng trong nhiều Use Case. Ví dụ:
    - Khi **nhập thông tin tiến trình**, có thể yêu cầu nhập Arrival Time, Burst Time, hoặc Priority tùy thuộc vào thuật toán.
    - Tính toán kết quả bao gồm các bước con: tính CPU Utilization, Waiting Time, và Turnaround Time.
* **<<extend>>:**
  + Một số chức năng mở rộng dựa trên lựa chọn của người dùng. Ví dụ:
    - **Chọn thuật toán lập lịch** sẽ mở rộng thành ba lựa chọn: FCFS, SJN, và RR.

### **4. Mô tả tương tác tổng quát**

1. Người dùng mở ứng dụng và chọn thuật toán.
2. Nhập thông tin tiến trình cần xử lý.
3. Quan sát trực quan biểu đồ mô phỏng và kết quả.
4. Nếu cần, xem phần hướng dẫn để hiểu thuật toán và cách tính toán.
5. Sau khi hoàn tất, thoát chương trình.

# III. Design

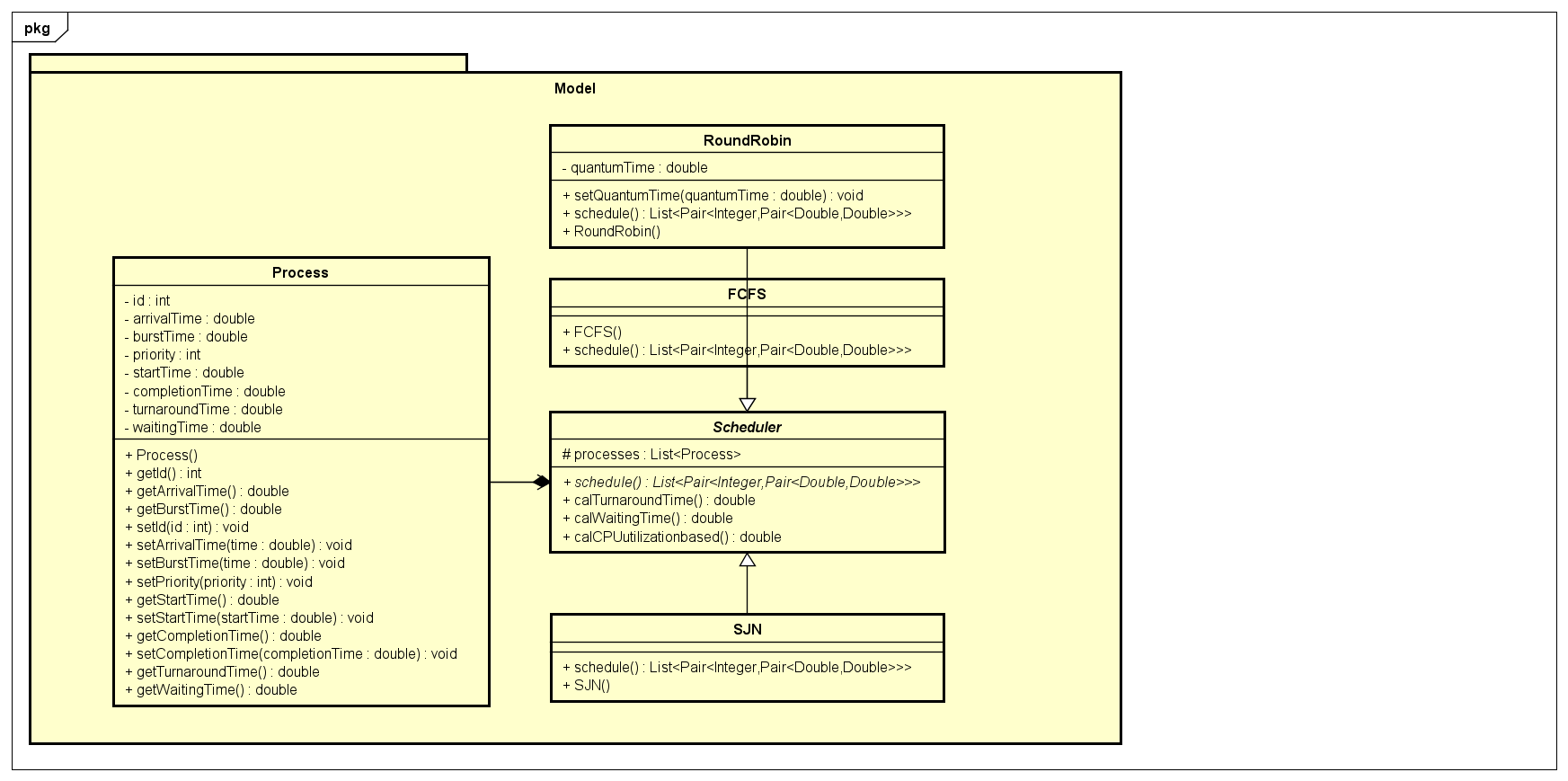
## Sơ đồ lớp chung (General class diagram)



* Gói (package) controller đóng vai trò trung tâm trong việc điều khiển luồng của chương trình. Các lớp trong gói này sẽ nhận các yêu cầu từ người dùng (thông qua giao diện), xử lý các yêu cầu đó và cập nhật giao diện tương ứng.
* Gói (package) view được thiết kế để tạo giao diện chính của chương trình mô phỏng các thuật toán lập lịch CPU. Nó đóng vai trò là **menu chính**, nơi người dùng có thể chọn thuật toán lập lịch cần mô phỏng hoặc tìm hiểu thêm thông tin.
* Gói (package) App khởi động chính của ứng dụng JavaFX. Nó đóng vai trò làm điểm vào (entry point) để chạy chương trình.
* Gói(package) Model thực hiện các chức năng sau:
* Mô hình hóa các đối tượng và tiến trình trong hệ thống (lớp Process).
* Chứa các thuật toán lập lịch khác nhau như FCFS, Round Robin, và SJN để xử lý các tiến trình.
* Tính toán các chỉ số hiệu suất như Turnaround Time, Waiting Time và CPU Utilization.

## Mô tả chi tiết

### Model



* Lớp **Process**
* Thuộc tính
  + **id**: Id của tiến trình
  + **arrivalTime**: Thời gian đến của tiến trình
  + **burstTime**: Thời gian để tiến trình hoàn thành xong
  + **priority**: Độ ưu tiên của tiến trình
  + **startTime**: Thời gian vào của tiến trình
  + **completionTime**: Thời gian tiến trình hoàn thành
  + **turnaroundTime**: Khoảng thời gian để tiến trình hoàn thành công việc
  + **waitingTime**: Tổng thời gian chờ của tiến trình
* Phương thức: Sử dụng các phương thức **getter** và **setter** của các thuộc tính để có thể lấy ra thuộc tính và thay đổi
* Chức năng:
* **Mô hình hóa các tiến trình** trong hệ thống, lưu trữ thông tin quan trọng về thời gian đến, thời gian thực thi, và thời gian bắt đầu và kết thúc.
* **Quản lý các thời gian liên quan** đến tiến trình trong suốt quá trình lập lịch, bao gồm thời gian bắt đầu và thời gian hoàn thành.
* **Cung cấp các phương thức để lấy và cập nhật thông tin** của tiến trình, giúp hỗ trợ các thuật toán lập lịch CPU trong việc theo dõi và xử lý các tiến trình.
* Lớp **Scheduler**
* Thuộc tính:
  + **processes**: List tiến trình, đầu vào của thuật toán lập lịch
* Phương thức:
  + **schedule()**: Phương thức abstract, trả về list tiến trình: **ID**, **thời gian vào**, **thời gian ra**.
  + **calTurnaroundTime()**: Tính turnaround time trung bình.
  + **calWaitingTime()**: Tính thời gian chờ trung bình.
  + **calCPUUtilization()**: Tính hiệu suất của CPU trong quá trình lập lịch.
* Lớp **FCFS**
* Kế thừa từ lớp cha Scheduler
* Thuộc tính:
  + **processes**: List tiến trình, đầu vào của thuật toán lập lịch
* Phương thức:
  + **schedule():**

Tạo một danh sách ganttChart để lưu trữ kết quả của lịch trình.

Đặt thời gian bắt đầu và thời gian hoàn thành của tất cả các tiến trình trong danh sách processes về 0.0.

Sắp xếp các tiến trình theo thời gian đến (arrivalTime).

Khởi tạo biến currentTime bằng 0.0.

Duyệt qua từng tiến trình trong danh sách processes:

Nếu currentTime nhỏ hơn arrivalTime của tiến trình, cập nhật currentTime bằng arrivalTime của tiến trình.

Tính toán startTime và completionTime của tiến trình.

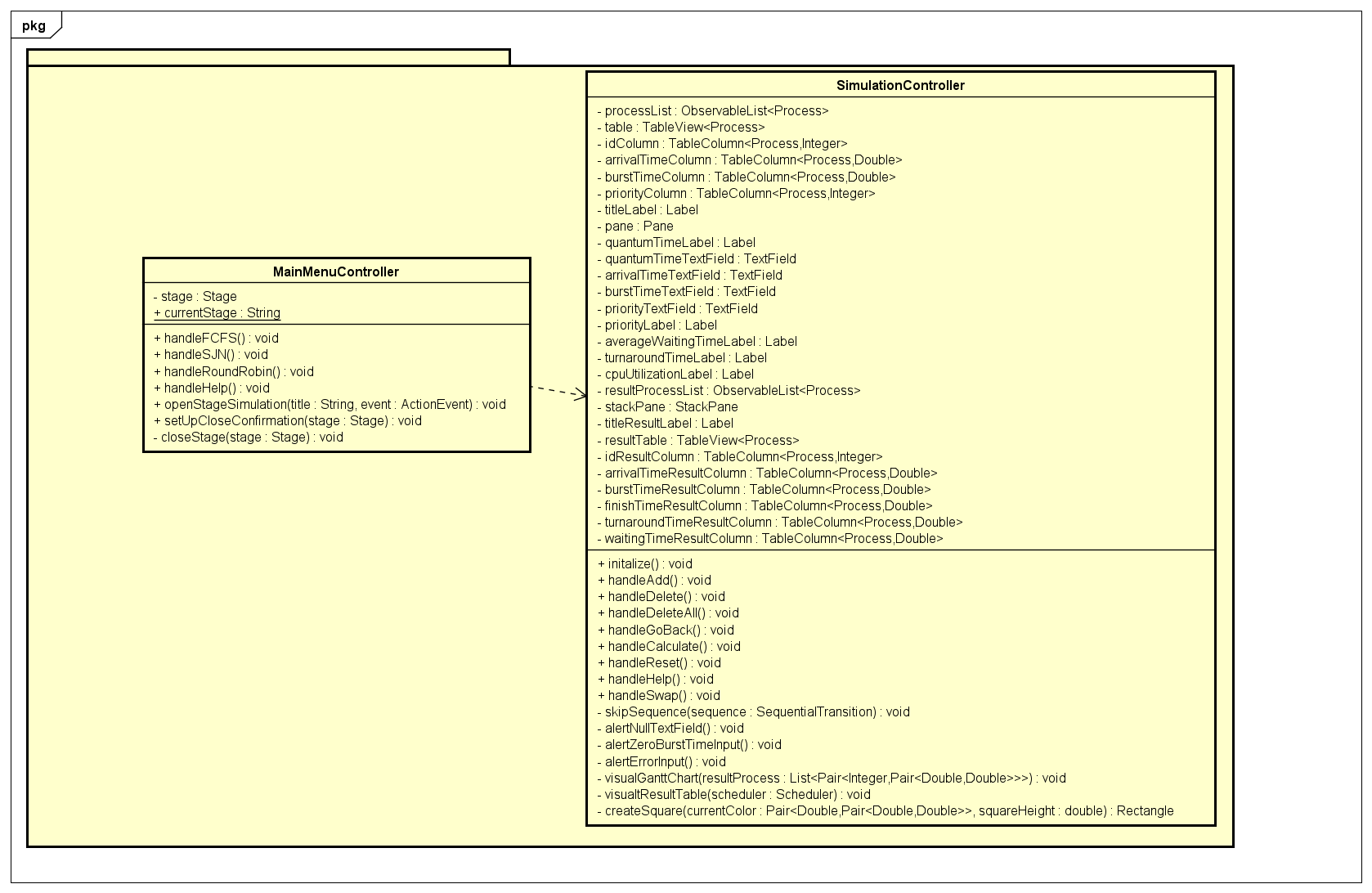
Cập nhật startTime và completionTime cho tiến trình.

Thêm thông tin của tiến trình vào ganttChart.

Phương thức này trả về danh sách ganttChart, chứa các cặp giá trị gồm ID của tiến trình và cặp giá trị thời gian bắt đầu và thời gian hoàn thành của tiến trình

* Lớp **SJN**
* Kế thừa từ lớp cha Scheduler
* Thuộc tính:
* Phương thức:
  + **schedule():**
* Lớp **RoundRobin**
* Kế thừa từ lớp cha Scheduler
* Thuộc tính:
  + Lớp RoundRobin có thêm giá trị **quantum time**, là thời gian lượng tử phục vụ cho quá trình lập lịch
* Phương thức:
  + **schedule():**

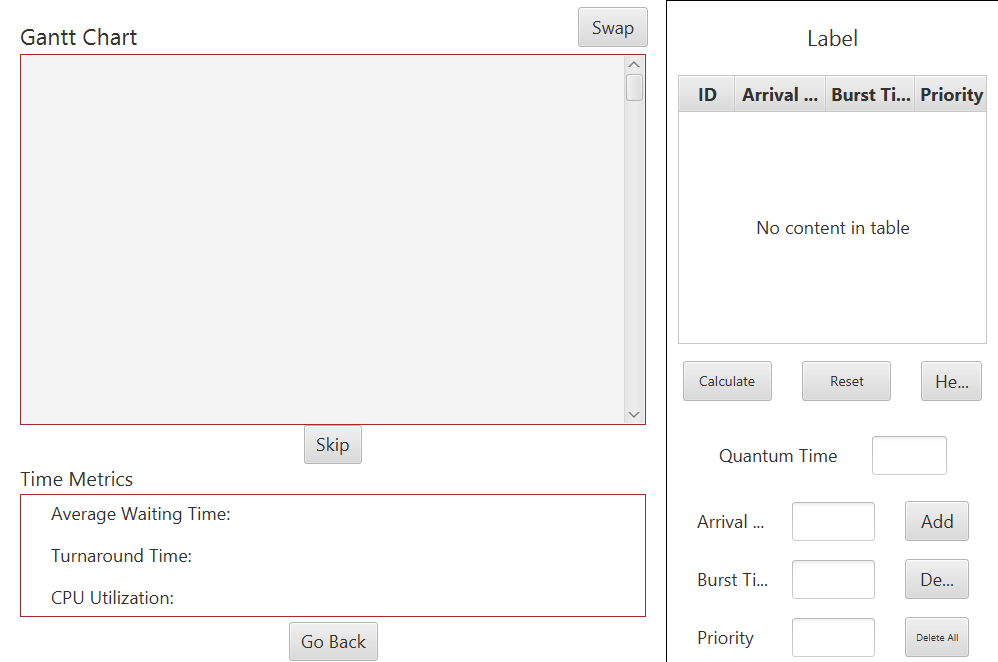
### Controller



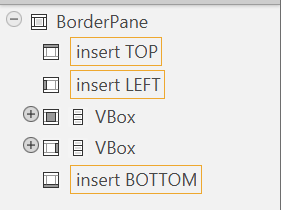
* Lớp **SimulationController**
* Thuộc tính:
  + **table**: bảng tiến trình đầu vào, cùng các cột **idColumn**, **arrivalTimeColumn**, **burstTimeColumn**, **priorityColumn**.
  + **processList**: list tiến trình đầu vào, truy xuất từ **table**.
  + **titleLabel**: tiêu đề của trình mô phỏng.
  + **quantumTimeLabel, quantumTimeTextField**: nơi nhập quantum time cho bộ lập lịch.
  + **arrivalTimeTextField**: nơi nhập arrival time.
  + **burstTimeTextField**: nơi nhập burst time.
  + **priorityLabel, priorityTextField**: nơi nhập priority.
  + **averageWaitingTimeLabel, turnaroundTimeLabel, cpuUtilizationLabel**: hiển thị time metrics của quá trình lập lịch.
  + **pane**: hiển thị gantt chart.
  + **resultTable**: bảng tiến trình đầu ra, cùng các cột **idResultColumn, arrivalTimeResultColumn, burstTimeResultColumn, finishTimeResultColumn, turnaroundTimeResultColumn, waitingTimeResultColumn.**
  + **stackPane**: chứa **pane**(gantt chart), **result table**.
  + **swapButton**: swap giữa 2 trạng thái **gantt chart, result table**.
  + **titleResultLabel**: tiêu đề của bảng đầu ra, hiện thị giá trị là **Gantt Chart** hoặc **Result Table.**
* Phương thức:
  + **initialize():** Khởi tạo trạng thái ban đầu của stage, tùy vào thuật toán đang lựa chọn để ẩn/hiện các thuộc tính:
    - Với **FCFS**, **quantumTimeLabel, quantumTimeTextField, priorityLabel, priorityTextField, priorityColumn** bị ẩn đi.
    - Với **SJN**, **quantumTimeLabel, quantumTimeTextField** bị ẩn đi.
    - Với **RR**, **priorityLabel, priorityTextField, priorityColumn** bị ẩn đi.
  + **visualGanttChart()**: Hiển thị kết quả lập lịch của tiến trình, các tiến trình xuất hiện lần lượt theo thứ tự xử lý bằng 1 dãy *pause*, có thể bỏ qua hiệu ứng bằng nút skip.
  + **createSquare()**: Tạo Rectangle để đại diện cho tiến trình trong Gantt Chart, với màu sắc ngẫu nhiên.
  + **handleAdd()**: Thêm tiến trình vào bảng đầu vào.
  + **handleDelete()**: Xóa 1 tiến trình khỏi bảng đầu vào.
  + **handleGoBack()**: Quay trở lại main menu.
  + **handleDeleteAll()**: Xóa tất cả các tiến trình trong bảng đầu vào.
  + **handleCalculate()**: Thực hiện lập lịch, hiển thị gantt chart, result table và time metrics.
  + **handleReset()**: Reset gantt chart, result table và time metrics.
  + **handleHelp()**: Giải thích thuật toán lập lịch hiện tại.
  + **handleSwap()**: Swap giữa gantt chart và result table.
  + **skipSequence()**: Bỏ qua hiệu ứng động gantt chart.
  + **alertNullTextField()**: Hiển thị cảnh báo khi có ô nhập liệu trống.
  + **alertZeroBurstTimeInput()**: Hiển thị cảnh báo khi burst time = 0.
  + **alertErrorInput()**: Hiển thị cảnh báo khi giá trị nhập vào < 0.
  + **visualResultTable()**: Hiện thị bảng kết quả lập lịch.
* Chức năng: Điều khiển lớp **SimulationView**.
* Lớp **MainMenuController**
* Các Thuộc Tính:
  + **stage:** Biến truyền vào phục vụ cho quá trình đóng chương trình.
  + **currentStage:** Biến tĩnh biểu thị cho thuật toán lập lịch đã lựa chọn.
* Các Phương Thức:
  + **handleFCFS():** Khi ấn vào nút FCFS, đồng nghĩa với việc chọn thuật toán lập lịch FCFS, gán giá trị **currentStage** = “FCFS”, gọi đến phương thức **openStageSimulation().**
  + **handleSJN():**Khi ấn vào nút SJN, đồng nghĩa với việc chọn thuật toán lập lịch SJN, gán giá trị **currentStage** = “SJN”, gọi đến phương thức **openStageSimulation().**
  + **handleRoundRobin():**Khi ấn vào nút RoundRobin, đồng nghĩa với việc chọn thuật toán lập lịch RoundRobin, gán giá trị **currentStage** = “RoundRobin”, gọi đến phương thức **openStageSimulation().**
  + **handleHelp():** hiển thị 1 dialog giải thích các thuật toán.
  + **openStageSimulation():** đóng stage main, mở stage simulation.
  + **setupCloseConfirmation():** set up quá trình đóng chương trình.
  + **closeStage():** truyền method setupCloseConfirmation(dùng để đóng chương trình) cho main menu bằng cách khai báo 1 controller trong lớp application(entry point của chương trình).
* Chức năng: điều khiển lớp **MainMenuView.**

### View:

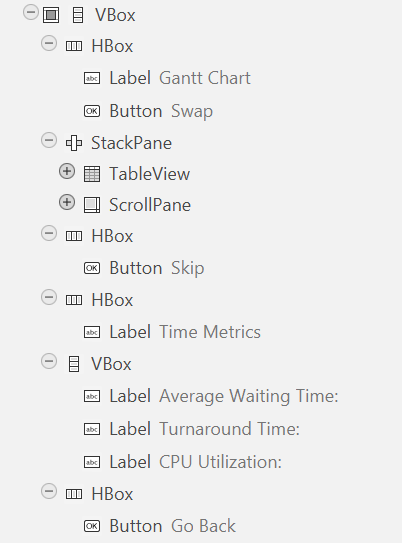
* Lớp **SimulationView**
  + Tổng quan:

****

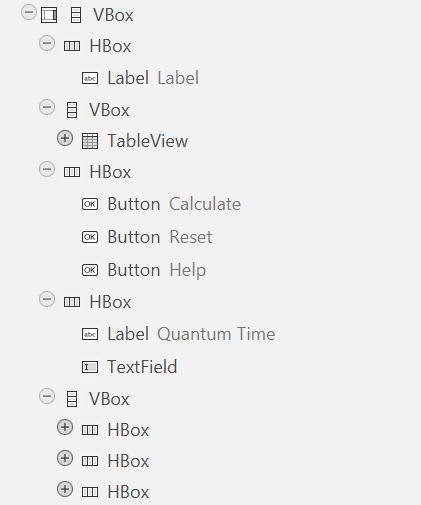
* BorderPane:



* Center:



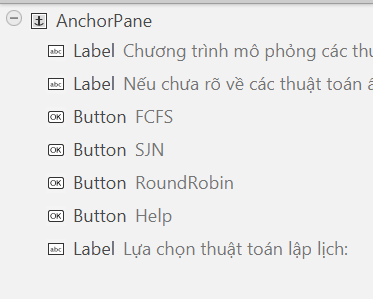
* Right:



* Lớp **MainMenuView**
  + Tổng quan:



* AnchorPane:



# IV. Tham khảo

Các phương thức mô phỏng gantt chart, hiện thị hộp thoại được tham khảo từ ChatGPT.