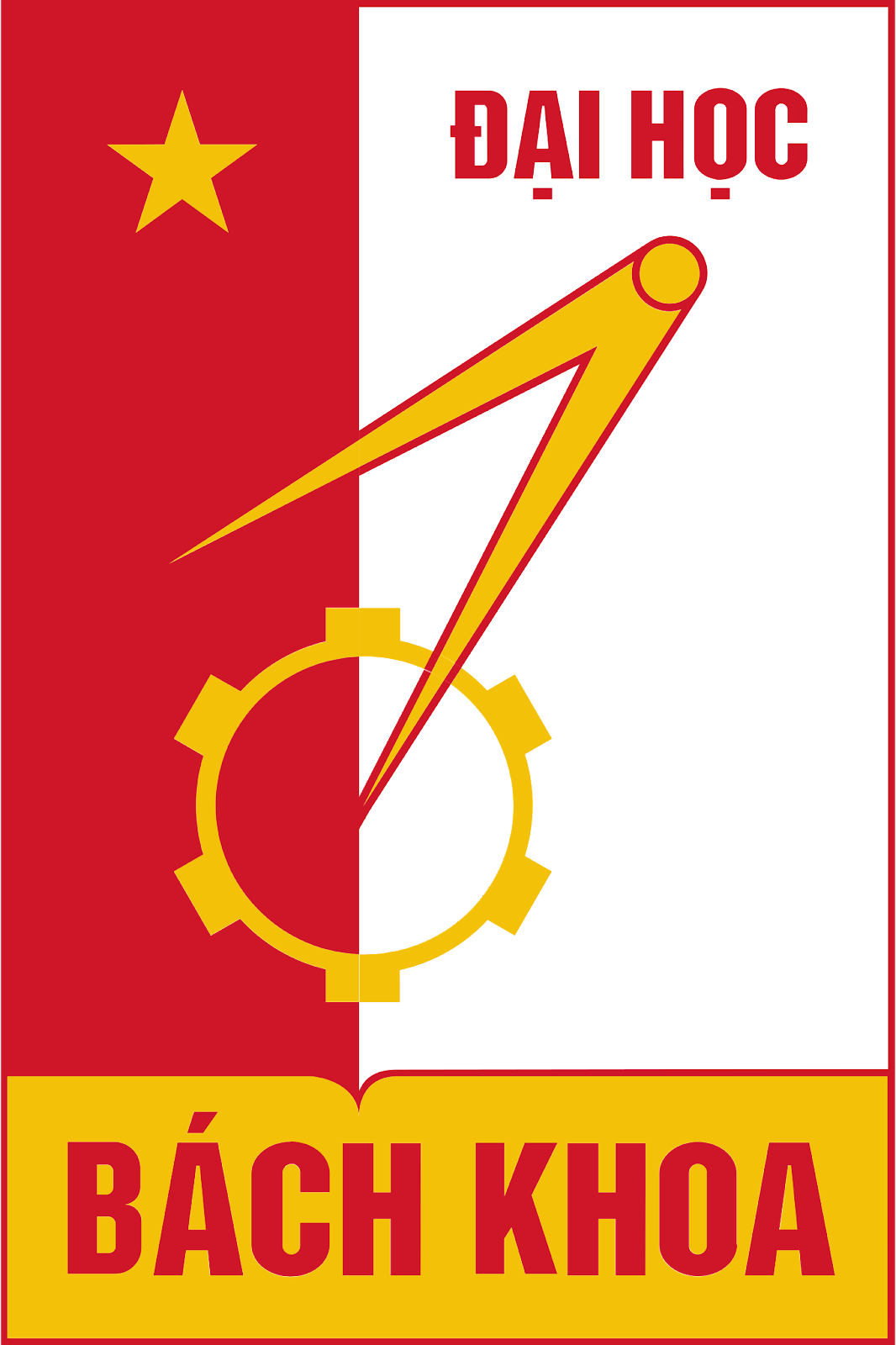
**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**BÁO CÁO BÀI TẬP LỚN**

**Môn: Lập trình hướng đối tượng**

***Đề tài: Trực quan hóa thuật toán định tuyến gói***

|  |  |  |
| --- | --- | --- |
| **Giảng viên:** | **Nguyễn Thị Thu Trang** | |
| **Nhóm:** | **14** |  |

*HÀ NỘI - 2024*

# Phân chia công việc

|  |  |  |
| --- | --- | --- |
| Họ và tên | MSSV | Công việc |
| Nguyễn Thanh Lâm | 20204997 | Viết tài liệu hướng dẫn, chuẩn bị báo cáo và chuẩn bị slide trình bày |
| Nguyễn Ngọc Lan | 20225730 | Viết các lớp cơ bản (Node và các lớp con của Node) và cài đặt thuật toán Dijkstra |
| Vi Dương Khanh | 20225864 | Tạo lớp Packet và cài đặt thuật toán Bellman-Ford |
| Nguyễn Thế Kiên | 20194597 | Thiết kế GUI: giao diện chính, bảng định tuyến và tích hợp với thuật toán |
| Lê Xuân Kỳ | 20226112 | Viết lớp Router, Connection và tạo Abstract Class RoutingAlgorithm |
| Nguyễn Đăng Tùng Lâm | 20225644 | Tạo lớp Graph, Network và cài đặt thuật toán Flooding |

# Giới thiệu bài toán

#### Trong mạng máy tính, thuật toán định tuyến gói tin (packet routing algorithm) là một trong những yếu tố quan trọng để truyền dữ liệu giữa các nút trong mạng. Bài toán yêu cầu xây dựng một chương trình trực quan hóa (visualization) để mô phỏng cách các gói tin được định tuyến qua mạng dựa trên thuật toán định tuyến (ví dụ: Dijkstra, Bellman-Ford, hoặc thuật toán định tuyến tùy chỉnh).

* Bài tập lớn này tìm cách giải quyết vấn đề này bằng cách tạo ra bằng cách tạo ra một hệ thống phần mềm mô phỏng hoạt động của các thuật toán định tuyến gói tin trong mạng máy tính. Hệ thống sẽ minh họa rõ ràng cách các gói tin được truyền qua mạng, đồng thời hỗ trợ người dùng hiểu các thuật toán định tuyến và phân tích hiệu quả của chúng. Hệ thống bao gồm các chức năng chính:

##### Mô hình hoá mạng máy tính (Biểu diễn dưới dạng đồ thị)

###### Nút (Node): Đại diện cho các thiết bị mạng

###### Network: Đại diện cho mạng chứa các node kết nối với nhau

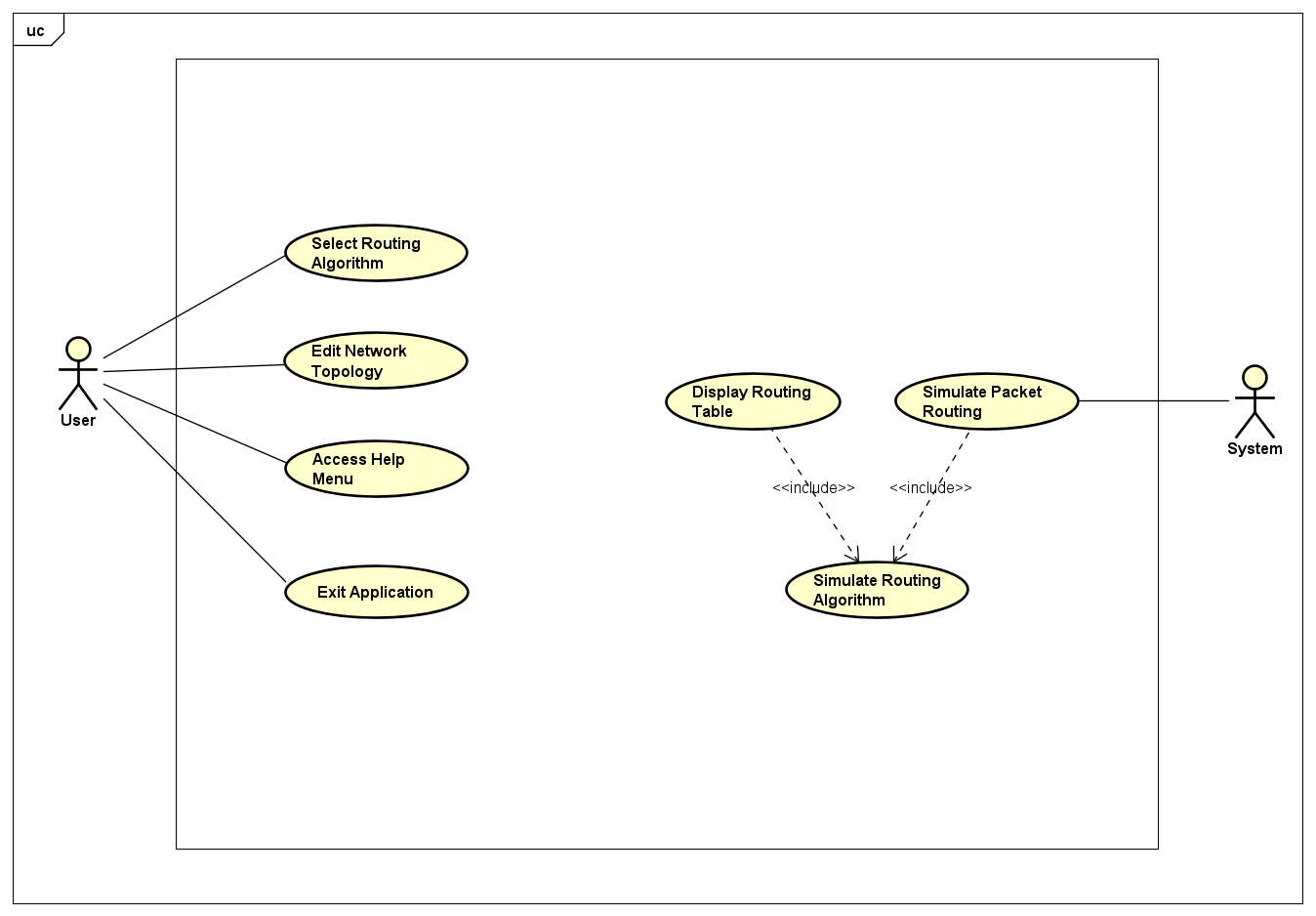
###### Connection: Đại diện kết nối giữa 2 node

* + Mô phỏng gói tin: Cho phép người dùng tạo ra các gói tin (Packet):
    - Nguồn và đích gói tin
    - Hành trình di chuyển qua các node
  + Thuật toán định tuyến: Triển khai các thuật toán định tuyến để xác định đường đi của gói ( Dijstra, BellmanFord,..)
  + Trực quan hoá quá trình
  + Tương tác với người dùng
    - Nhập cấu trúc mạng
    - Chọn thuật toán
    - Xem trực quan hoá

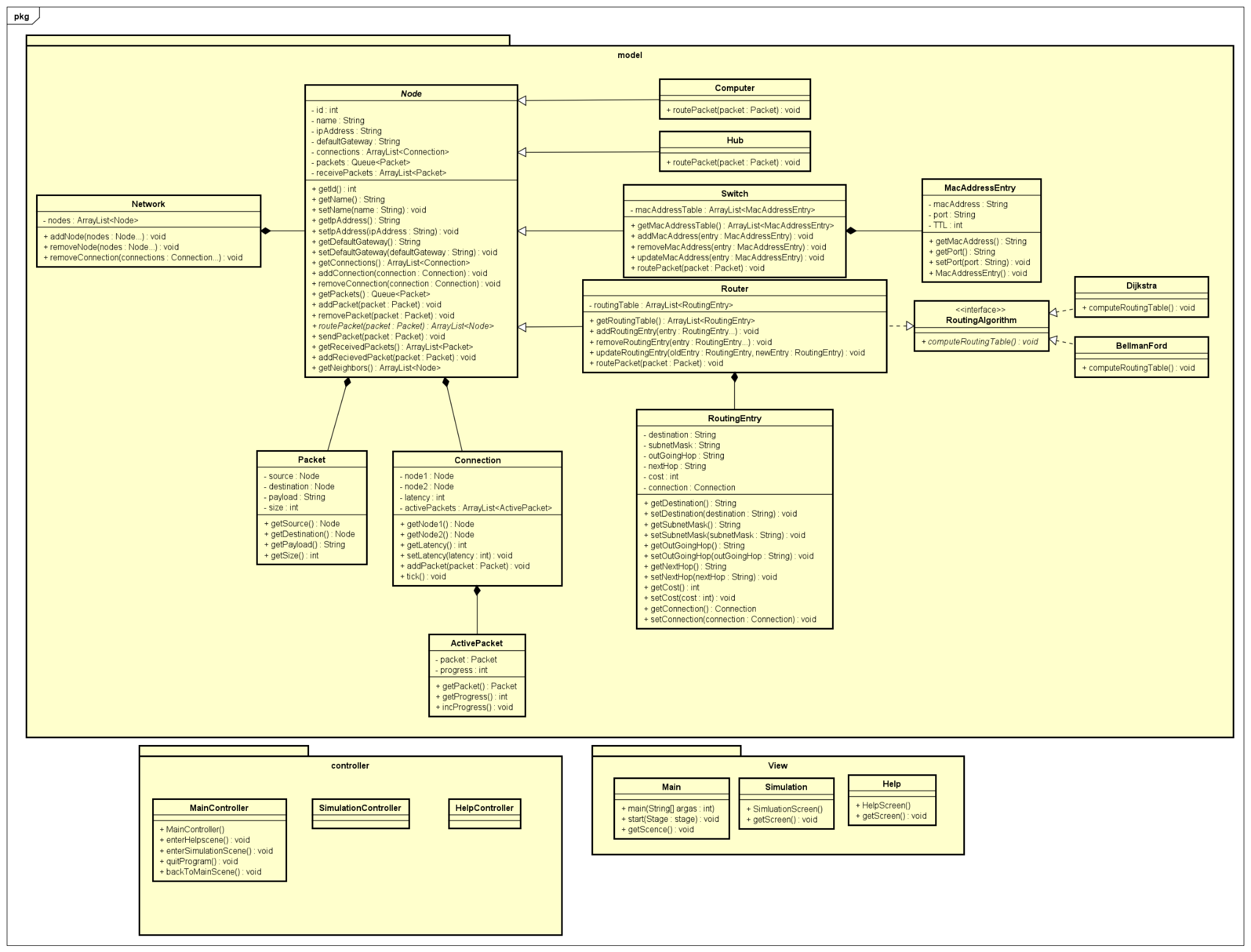
# Mô hình hóa dữ liệu

## 1. Biểu đồ UML

### 1.1. Use Case Diagram



### 1.2. Class Diagram



## 2. Giải thích thiết kế

#### Lớp Network

##### Vai trò: Đại diện cho mạng chứa các nút (nodes) kết nối với nhau.

##### Thuộc tính:

##### Nodes: Danh sách các Node trong mạng.

##### Phương thức:

##### addNode(Node...): Thêm một hoặc nhiều nút vào mạng.

##### removeNode(Node...): Loại bỏ một hoặc nhiều nút khỏi mạng.

##### removeConnection(Connection…): Xóa 1 hoặc nhiều kết nối.

#### Lớp Node

#### Vai trò: Đại diện cho các thành phần cơ bản trong mạng, chẳng hạn như máy tính, hub, switch hoặc router.

#### Thuộc tính:

##### id, name: Định danh và tên của nút.

##### ipAddress, defaultGateway: Địa chỉ IP và cổng mặc định của nút.

##### connections: Danh sách các kết nối với các nút khác.

##### packets: Hàng đợi gói tin sẽ gửi đi.

##### receivePackets: Danh sách các gói tin nhận được.

* + Phương thức:
    - Getter và setter (Id chỉ có getter)
    - addConnection(), removeConnection(): Thêm/Xóa 1 kết nối trong danh sách kết nối
    - addPacket(): Thêm 1 gói tin vào danh sách gói tin sẽ gửi đi
    - removePacket(): Xóa gói tin khỏi danh sách gói tin sẽ gửi (Khi gói tin đã đc gửi đi)
    - *routePacket()*: Định tuyến cho gói tin. Tìm đường đi kế tiếp cho gói tin phụ thuộc vào kiểu Node là gì (Computer, Hub, Switch, Router).
    - sendPacket(): Gửi gói tin đi khi đã tìm được đường đi tiếp theo cho gói tin thông qua phương thức routePacket().
    - getNeighbors(): Lấy danh sách các Node “hàng xóm” của Node hiện tại

#### Các lớp con của Node

#### Computer

#### Vai trò: Đại diện cho máy tính trong mạng.

#### Phương thức: routePacket(Packet): Xác định xem đích (destination) của gói tin có nằm trong mạng cục bộ hay không. Nếu có, gửi trực tiếp; nếu không, gửi gói tin đến default gateway.

#### Hub

#### Phương thức: routePacket(Packet): Chuyển tiếp gói tin đến tất cả các nút kết nối với nó.

#### Switch

#### Vai trò: Đại diện cho thiết bị chuyển mạch (Switch).

#### Thuộc tính: macAddressTable: lưu bảng chuyển tiếp dựa trên địa chỉ MAC

#### Phương thức:

* + - * Getter
      * addMacAddress(): Thêm 1 dòng vào bảng chuyển tiếp
      * removeMacAddress(): Xóa 1 dòng khỏi bảng chuyển tiếp
      * updateMacAddress(): Sửa 1 dòng trong bảng chuyển tiếp
      * routePacket(Packet): Chuyển tiếp gói tin dựa trên bảng MAC
* Lớp MacAddressEntry
  + Vai trò: Đại diện cho một dòng trong bảng MAC
  + Thuộc tính:
    - macAddress: Địa chỉ MAC của host
    - port: Cổng kết nối với host
    - TTL: Thời gian giữ lại thông tin trong bảng (Thường là 300 giây)
  + Phương thức: Getter và setPort()

#### Lớp Router

#### Vai trò: Thiết bị định tuyến trong mạng.

#### Thuộc tính: routingTable: Bảng lưu thông tin định tuyến của từng Router

#### Phương thức đặc biệt:

* + - Getter
    - addRoutingEntry (): Thêm 1 dòng vào bảng định tuyến
    - removeRoutingEntry (): Xóa 1 dòng khỏi bảng định tuyến
    - updateRoutingEntry (): Sửa 1 dòng trong bảng định tuyến
    - routePacket(Packet): Chuyển tiếp gói tin dựa trên bảng định tuyến
* Lớp RoutingEntry
  + Vai trò: Đại diện cho một dòng trong bảng định tuyến.
  + Thuộc tính:
    - destination: Đích đến
    - subnetMask: Mặt nạ mạng của node đích
    - outgoingHop: Cổng ra của gói tin
    - nextHop: Điểm đến kế tiếp của gói tin
    - cost: Chi phí định tuyến
  + Phương thức: Getter và setter

#### Lớp Packet

#### Vai trò: Đại diện cho gói tin truyền trong mạng.

#### Thuộc tính:

* + - source, destination: Nút nguồn và đích.
    - payload: Nội dung của gói tin.
    - size: Kích thước của gói tin.
  + Phương thức: Getter

#### Lớp Connection

#### Vai trò: Đại diện cho kết nối giữa hai nút.

#### Thuộc tính:

* + - node1, node2: Hai nút kết nối.
    - latency: Độ trễ của kết nối.
    - activePackets: Lưu danh sách các gói tin đang được truyền trên kết nối hiện tại
  + Phương thức:
    - Getter và setLantency()
    - addPacket(): Thêm 1 gói tin vào activePackets
    - tick():

#### Lớp ActivePacket

#### Vai trò: Theo dõi trạng thái của gói tin trong quá trình truyền

#### Thuộc tính:

* + - packet
    - progress: Trạng thái của gói tin hiện tại (Đã truyền được hết kết nối hay chưa?). Nếu progress = latency thì gói tin đã đi qua hết kết nối à đến đích à thêm vào receivePacket của node đích nếu đó là đích của gói tin. Hoặc thêm vào packet để gói tin tiếp tục được truyền đến đích.
  + Phương thức:
    - Getter
    - incProgess(): progress tăng lên sau mỗi giây cho đến khi = latency.

### Interface RoutingAlgorithm

* + Vai trò: Tính toán bảng định tuyến theo các thuật toán định tuyến Dijkstra hoặc Bellman Ford

### Các lớp Dijkstra và BellmanFord

* + Vai trò: Triển khai thuật toán định tuyến khác nhau để tính bảng định tuyến cho Router

# Kỹ thuật lập trình hướng đối tượng

Trong dự án này, các kỹ thuật lập trình hướng đối tượng được áp dụng xuyên suốt để mô hình hóa và quản lý các thực thể trong mạng, triển khai các thuật toán định tuyến, cũng như xây dựng giao diện trực quan cho người dùng. Các khái niệm OOP quan trọng bao gồm kế thừa, trừu tượng hóa, và đóng gói.

## 1. Kế thừa

Kế thừa cho phép dự án xây dựng một hệ thống phân cấp các lớp, trong đó các lớp con có thể chia sẻ và mở rộng các thuộc tính lẫn phương thức của lớp cha. Một số ví dụ nổi bật về kế thừa trong dự án:

* Lớp Router kế thừa từ Node
  + Lớp Node chứa các thông tin chung: id, name, ipAddress.
  + Lớp Router kế thừa tất cả những thuộc tính chung này và bổ sung thêm bảng định tuyến (routingTable), cùng với các phương thức như addRoutingEntry(), removeRoutingEntry(), v.v.
  + Điều này cho phép Router mở rộng hành vi của Node nhằm thực hiện chức năng chuyên biệt của một router trong mạng.
* Lớp Connection kế thừa từ Edge
  + Lớp Edge quản lý hai nút (node1, node2) và trọng số (weight).
  + Lớp Connection mở rộng bằng cách thêm thông tin về băng thông (bandwidth) và độ trễ (latency).
  + Qua đó, Connection vẫn duy trì bản chất của một Edge, đồng thời cung cấp thêm chi tiết quan trọng phục vụ mô phỏng mạng.
* Lớp Network kế thừa từ Graph
  + Lớp Graph quản lý danh sách nodes và connections, cùng với các phương thức thêm/xóa node, connection, truy xuất neighbors, v.v.
  + Lớp Network kế thừa toàn bộ chức năng này và thêm các thuộc tính, phương thức riêng như packets, sendPacket(), routePackets(), simulateTick() để mô phỏng hoạt động mạng.
* Lớp RoutingAlgorithm và các lớp con Dijkstra, BellmanFord, Flooding
  + RoutingAlgorithm là lớp cha cung cấp các phương thức chung như initializeRoutingTable() và một phương thức trừu tượng computeShortestPath().
  + Dijkstra, BellmanFord đều kế thừa từ lớp này, ghi đè (override) computeShortestPath() để triển khai thuật toán riêng.
  + Flooding có thể được xem như một lớp riêng hoặc cũng có thể kế thừa và tùy biến nếu muốn.

Việc tổ chức kế thừa hợp lý giúp các lớp có cấu trúc rõ ràng, dễ mở rộng, tái sử dụng mã hiệu quả và tiết kiệm thời gian cho nhóm phát triển.

## 2. Trừu tượng hóa

Trừu tượng hóa được áp dụng nhằm đơn giản hóa cách biểu diễn các thực thể mạng, ẩn giấu các chi tiết triển khai phức tạp, chỉ cung cấp những giao diện (interface) cần thiết để tương tác. Một số ví dụ về trừu tượng hóa:

* Phân tách lớp RoutingAlgorithm với các lớp con (Dijkstra, BellmanFord):
  + Lớp RoutingAlgorithm chứa các phương thức khởi tạo bảng định tuyến (initializeRoutingTable()) và khai báo một phương thức trừu tượng computeShortestPath().
  + Những chi tiết như cách tính đường đi ngắn nhất theo Dijkstra hay kiểm tra trọng số âm trong Bellman-Ford đều được giấu trong các lớp con.
  + Giúp lớp cao hơn (như lớp GUI) chỉ cần gọi computeShortestPath() mà không cần biết cụ thể thuật toán bên trong.
* Lớp Graph và lớp Network:
  + Giao diện (interface) hoặc các phương thức công khai như addNode(), removeNode(), addConnection(), removeConnection(), getNeighbors() được cung cấp để tương tác với các đối tượng trong mạng.
  + Người dùng hay các module GUI không cần biết chi tiết cấu trúc dữ liệu (List, Map, v.v.) bên trong; họ chỉ gọi phương thức để thêm/xóa node, connection.
* Giao diện người dùng (GUI):
  + GUI chỉ hiển thị các nút (routers, switches, computers), các kết nối (connections) và gói tin (packets) di chuyển, không lộ chi tiết logic xử lý định tuyến.
  + Khi người dùng chọn “Run Dijkstra” hoặc “Run Bellman-Ford”, chỉ cần một lệnh gọi cấp cao đến lớp quản lý thuật toán.

Nhờ trừu tượng hóa, hệ thống trở nên dễ dùng và dễ phát triển, cho phép thay thế hoặc nâng cấp thuật toán, cấu trúc bên trong mà không ảnh hưởng đến phần còn lại của ứng dụng.

## 3. Đóng gói

Đóng gói được áp dụng để bảo vệ dữ liệu và đảm bảo tính toàn vẹn của thông tin, đồng thời kiểm soát truy cập qua các phương thức getter/setter hoặc các hàm xử lý an toàn. Một số ví dụ về đóng gói:

* Thuộc tính trong các lớp Node, Edge, Router, Connection
  + Các thuộc tính như id, name, ipAddress, weight, bandwidth, latency thường được khai báo là private.
  + Việc đọc hoặc ghi giá trị chỉ được phép thông qua getter và setter (ví dụ getWeight(), setWeight()). Điều này giúp kiểm soát khi thay đổi trọng số, băng thông, độ trễ, tránh việc set các giá trị không hợp lệ.
* Bảng định tuyến (routingTable) trong lớp Router
  + routingTable chứa Map<Node, RoutingEntry>, nơi RoutingEntry có thể lưu chi tiết như nextHop, distance, v.v.
  + Tương tự, truy xuất hoặc chỉnh sửa bảng định tuyến đều thông qua các phương thức addRoutingEntry(), removeRoutingEntry(), updateRoutingTable(), giúp kiểm soát cập nhật và ngăn chặn lỗi logic khi thao tác trực tiếp.
* Bên trong lớp Network
  + Danh sách packets hay networkState (trạng thái các nút) cũng được đóng gói. Chỉ một số phương thức nhất định như sendPacket(Packet p), routePackets(), hay simulateTick() được phép thao tác và cập nhật, giữ cho dữ liệu nhất quán trong toàn hệ thống.

Nhờ đóng gói, dự án duy trì tính toàn vẹn và giảm thiểu lỗi do truy cập và sửa đổi dữ liệu bất cẩn.

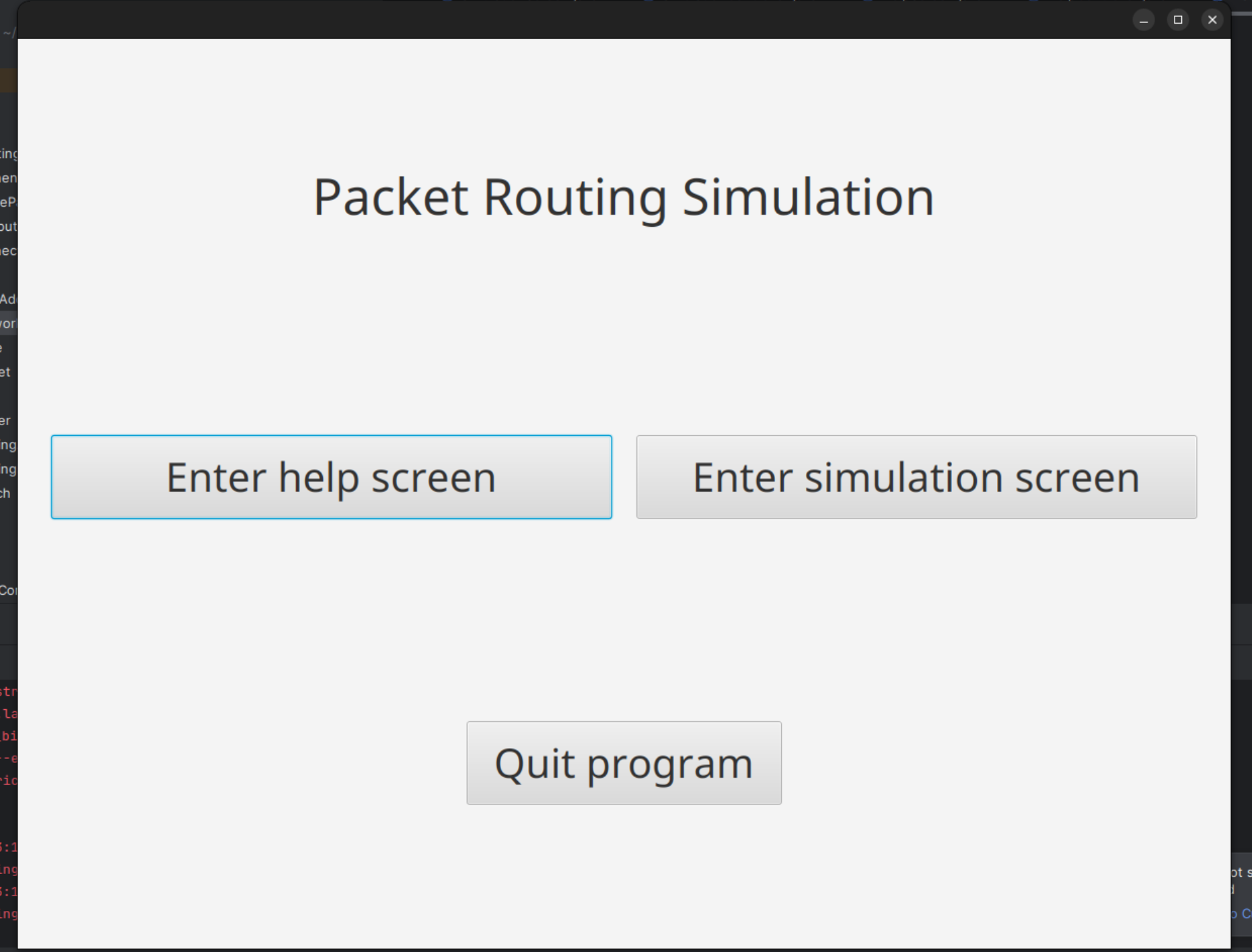
## 4. Lợi ích và Tầm quan trọng của OOP trong dự án

* Việc sử dụng các kỹ thuật OOP mang lại nhiều lợi ích:
* Tái sử dụng mã:
  + Kế thừa giúp tận dụng các lớp cơ sở như Note, Edge, Graph để mở rộng thành Router, Connection, Network mà không viết lại logic.
* Đóng gói dữ liệu:
  + Bảo đảm tính toàn vẹn cho các thuộc tính trọng yếu (trọng số, băng thông, địa chỉ IP, v.v.).
  + Tránh sửa đổi bất cẩn hay truy cập trái phép.
* Dễ mở rộng và bảo trì:
  + Trừu tượng hóa thuật toán định tuyến giúp thêm hoặc thay đổi thuật toán (ví dụ: thêm OSPF, RIP) mà không phá vỡ kiến trúc tổng thể.
  + GUI chỉ cần gọi các phương thức công khai, không phụ thuộc vào cấu trúc nội bộ.
* Tính nhất quán:
  + Các lớp con tuân thủ một khuôn mẫu từ lớp cha, giúp dễ nắm bắt luồng xử lý và bảo trì lâu dài.
* Hiệu quả phát triển:
  + Tách biệt rõ ràng giữa mô hình dữ liệu, logic thuật toán, và giao diện.
  + Dễ phân công công việc trong nhóm, nâng cao tính cộng tác.

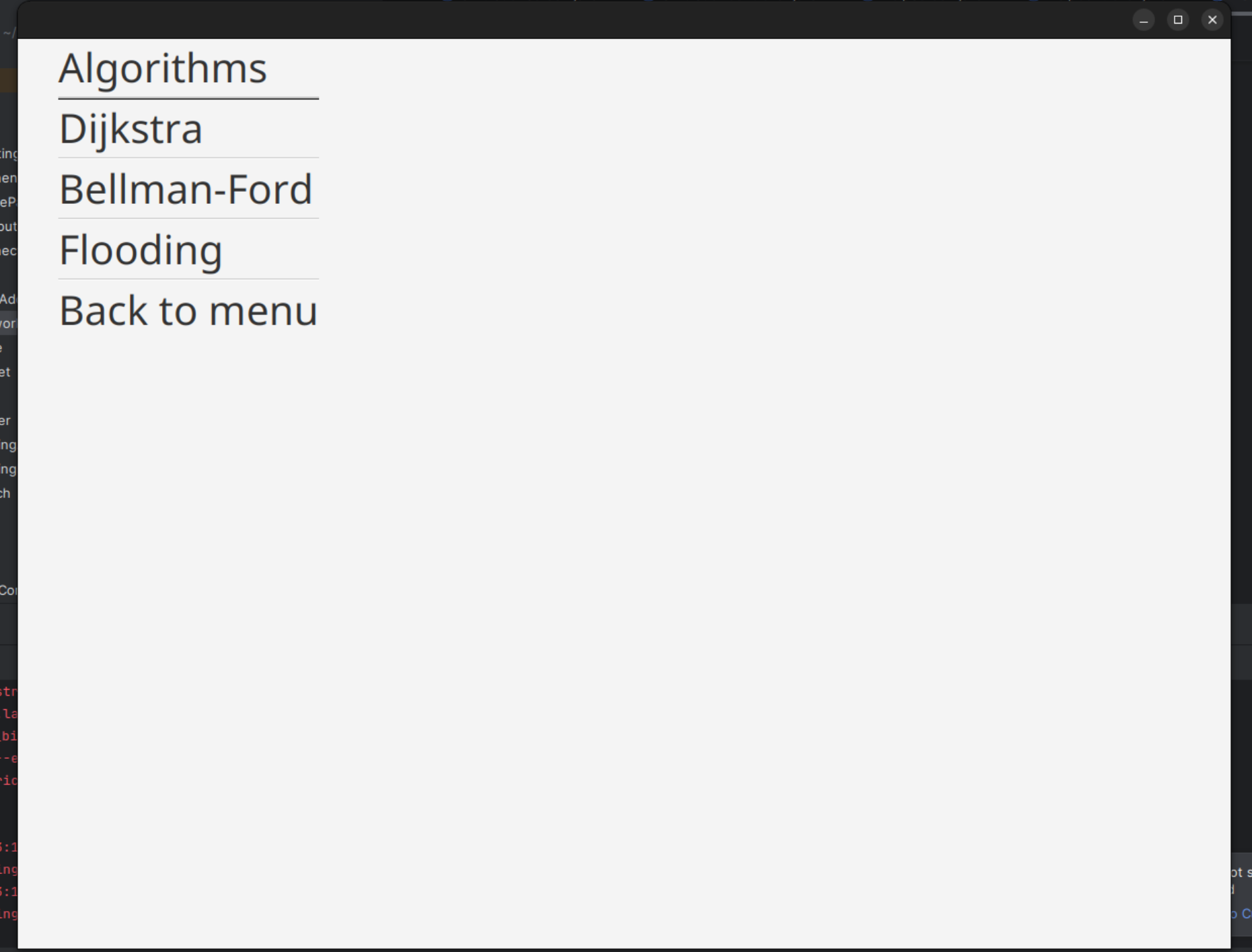
Bằng cách áp dụng những nguyên lý OOP – đặc biệt là kế thừa, trừu tượng hóa và đóng gói – dự án Visualization of Packet Routing Algorithms đạt được cấu trúc logic, rõ ràng, dễ bảo trì, và linh hoạt khi mở rộng. Sự tách biệt giữa lớp mô hình (Model), thuật toán (Controller) và giao diện (View) cho phép người dùng quan sát trực quan cách các thuật toán định tuyến (Dijkstra, Bellman-Ford, Flooding) hoạt động, tạo nền tảng vững chắc cho những phát triển mô phỏng mạng cao cấp hơn trong tương lai.

# Hướng dẫn sử dụng

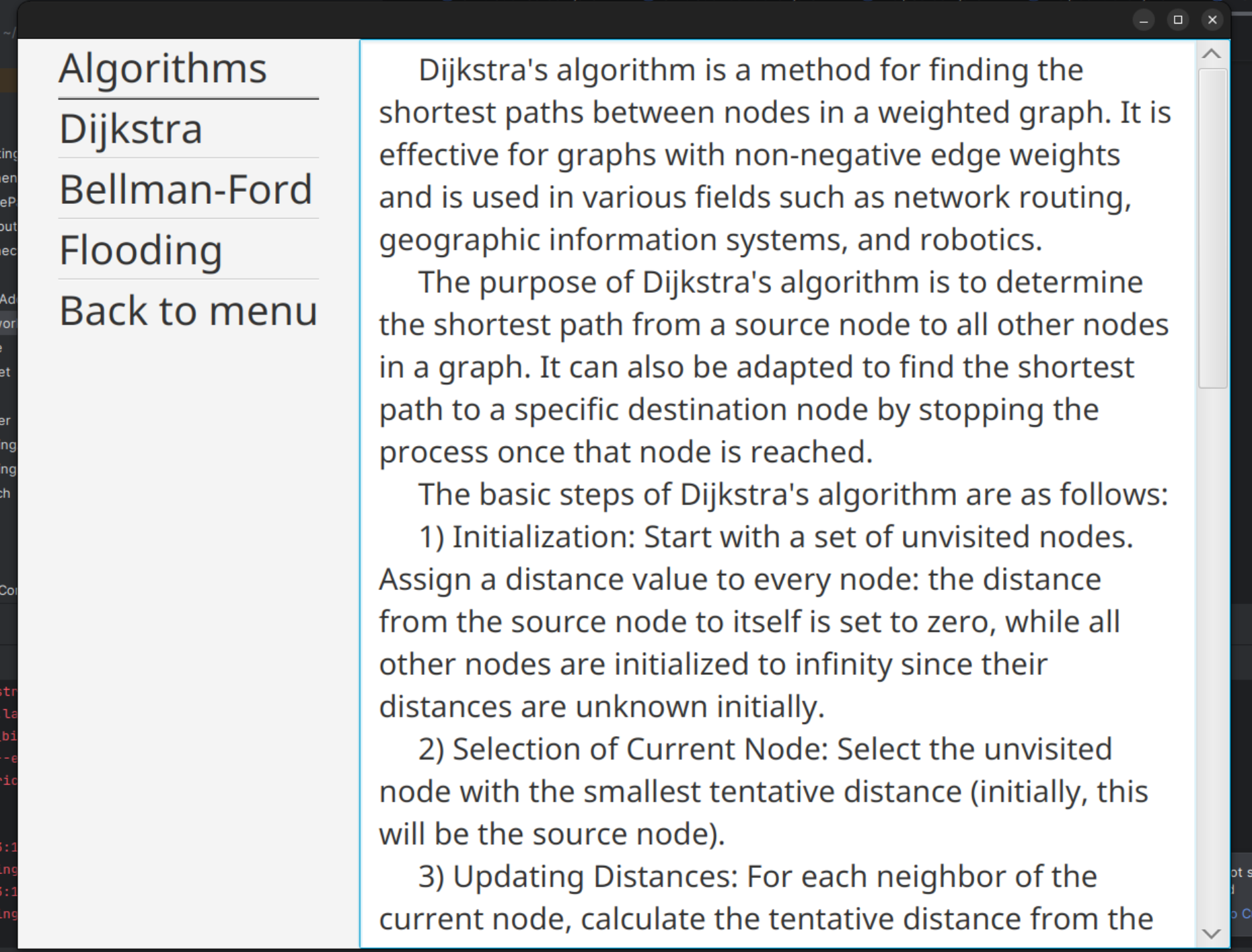
* Màn hình lúc mới chạy chương trình:



* Enter help screen: Nhấn để chuyển sang màn hình giải thích các thuật toán
* Enter simulation screen: Nhấn để chuyển sang màn hình chạy mô phỏng các thuật toán
* Quit program: Thoát chương trình
* Enter help screen:



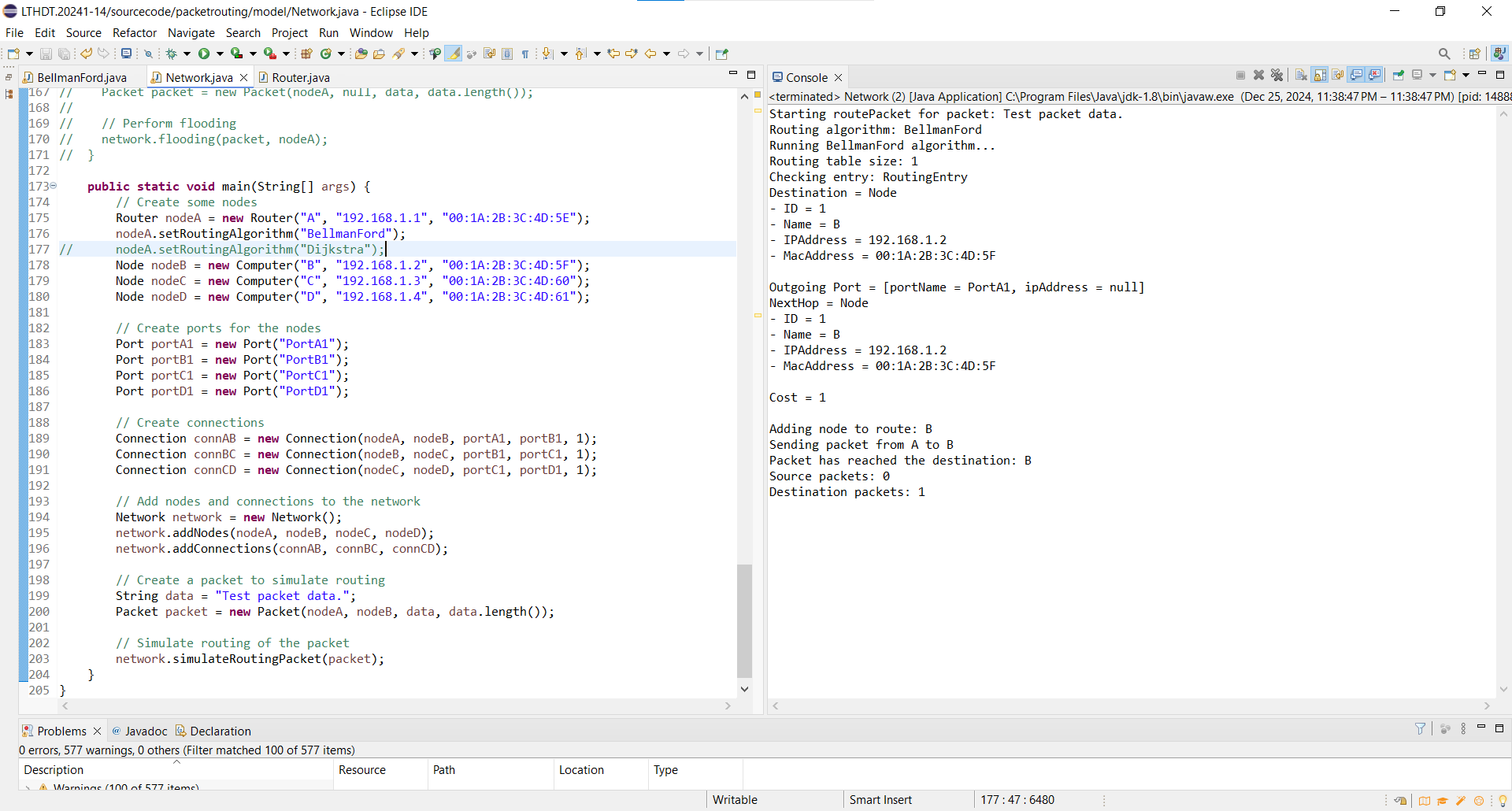
* Có 4 mục bên tai trái, 3 thuật toán và chọn về màn hình chính, nếu chọn cái cuối sẽ quay về màn hình trước, khi chọn 1 trong 3 thuật toán, phần giải thích sẽ hiện ra như sau.



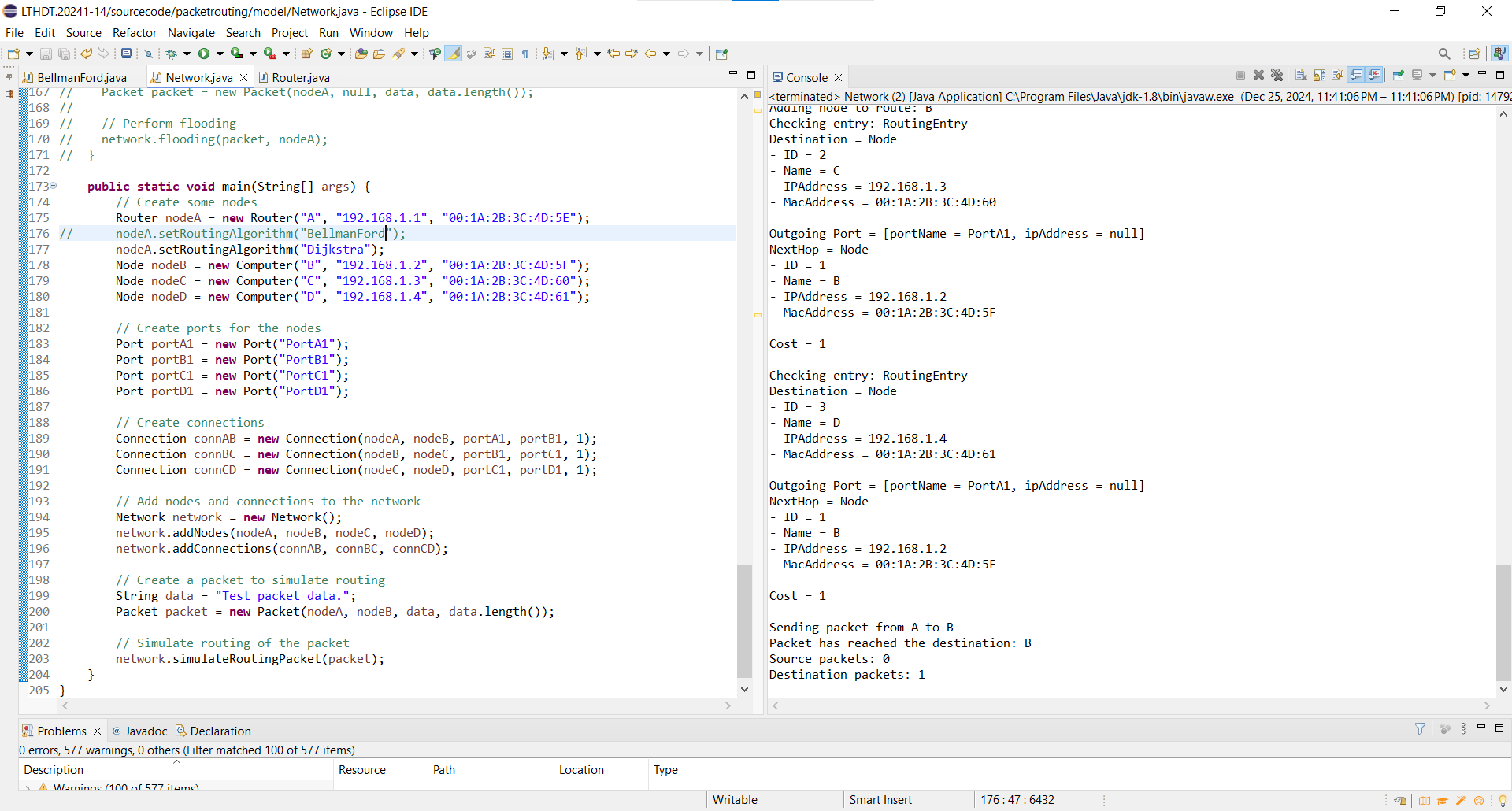
* Có thể lăn chuột ở phía tay phải để lướt lên xuống.
* Có thể chọn lại 1 thuật toán khác để đọc phần giải thích của thuật toán đó.

# Demo

**Bellman Ford**



**Dijkstra**



# Kết luận

Dự án trực quan hóa thuật toán định tuyến gói đã hoàn thành thành công nhờ sử dụng các kỹ thuật lập trình hướng đối tượng và quy trình thiết kế mô phỏng mạng. Trong quá trình phát triển, dữ liệu về các node, kết nối, gói tin, cũng như thuật toán định tuyến (Dijkstra, Bellman-Ford, Flooding) được mô hình hóa và quản lý bằng những lớp OOP chuyên biệt. Việc áp dụng Java và các thư viện hỗ trợ đã giúp việc triển khai, mô phỏng và hiển thị đồ thị mạng trở nên hiệu quả, trực quan và dễ theo dõi.

Dự án đã xây dựng một mô hình mạng tương đối toàn diện, bao gồm các lớp Node, Router, Connection, Packet, Graph, Network… để biểu diễn đầy đủ các yếu tố của một hệ thống mạng thực tế. Giao diện người dùng (GUI) cũng được thiết kế để cho phép người dùng tương tác (thêm, xóa node, kết nối, chạy thuật toán, theo dõi bảng định tuyến) và quan sát trực quan cách các gói tin di chuyển cũng như kết quả tính toán đường đi ngắn nhất. Việc sử dụng kế thừa, trừu tượng hóa và đóng gói giúp đảm bảo mã nguồn có tính tái sử dụng, bảo vệ dữ liệu và cải thiện tính toàn vẹn của toàn bộ quá trình mô phỏng.

Nhìn chung, dự án này đã chứng minh tầm quan trọng của việc mô phỏng và trực quan hóa các thuật toán định tuyến, cũng như lợi ích của các kỹ thuật lập trình hướng đối tượng trong việc tạo ra một hệ thống linh hoạt, có tổ chức và dễ dàng mở rộng. Công cụ mô phỏng được tạo ra trong dự án này sẽ là nguồn tài nguyên quý giá cho bất kỳ ai quan tâm đến lĩnh vực mạng máy tính, giúp người học và nhà nghiên cứu hiểu rõ hơn về nguyên lý hoạt động của các thuật toán định tuyến và hỗ trợ cho việc phân tích, phát triển các giải pháp mạng trong tương lai.