|  |
| --- |
| ĐẠI HỌC QUỐC GIA HÀ NỘI  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**    **Nhóm 7:** Lê Văn Hùng  Vương Thị Mỵ  Phan Văn Nam  Bùi Thị Mai  Nguyễn Thị Thùy Dương  **TÌM HIỂU HỆ ĐA TÁC TỬ VỚI NỀN TẢNG JADE**  **Hà Nội - 2016** |

**MỤC LỤC**

[ĐẶT VẤN ĐỀ 1](#_Toc444109344)

[CHƯƠNG I: CƠ SỞ LÍ THUYẾT 2](#_Toc444109345)

[I.1. Tác tử và hệ đa tác tử 2](#_Toc444109346)

[I.1.1. Khái niệm tác tử 2](#_Toc444109347)

[I.1.2. Các đặc điểm tính chất của tác tử 3](#_Toc444109348)

[I.1.3. Phân biệt tác tử và đối tượng 3](#_Toc444109349)

[I.1.4. Các loại tác tử 3](#_Toc444109350)

[I.2. Hệ đa tác tử 3](#_Toc444109351)

[I.2.1. Khái niệm 3](#_Toc444109352)

[I.2.2. Sự phối hợp của các tác tử 4](#_Toc444109353)

[I.2.3. Ưu điểm và nhược điểm 4](#_Toc444109354)

[I.2.4. Ứng dụng 4](#_Toc444109355)

[I.3. JADE framework 5](#_Toc444109356)

[I.3.1. Giới thiệu 5](#_Toc444109357)

[I.3.2. Kiến trúc JADE 5](#_Toc444109358)

[I.3.2. Các thành phần JADE 6](#_Toc444109359)

[I.3.2. Tác tử trong JADE 8](#_Toc444109360)

[I.3.3. Các hành vi của agent 8](#_Toc444109361)

[CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ PHẦN MỀM 12](#_Toc444109362)

[CHƯƠNG III: CÀI ĐẶT VÀ TRIỂN KHAI 13](#_Toc444109363)

[III.1 Cài đặt JADE 13](#_Toc444109364)

[CHƯƠNG 4: KẾT LUẬN 15](#_Toc444109365)

**DANH MỤC HÌNH ẢNH**

[Hình 1: Tác tử và môi trường của nó 2](#_Toc444109480)

[Hình 2: Khung chứa và các nền tảng 7](#_Toc444109481)

[Hình 3: Ví dụ về kiểu hành vi thực hiện một lần 9](#_Toc444109482)

[Hình 4: Ví dụ kiểu hành vi thực hiện mãi mãi 9](#_Toc444109483)

[Hình 5: Ví dụ về kiểu hành vi đánh thức 10](#_Toc444109484)

[Hình 6: Ví dụ về kiểu hành vi thực hiện có chu kì 11](#_Toc444109485)

[Hình 7: Mô tả cách thêm các thư viện JADE để cấu hình một project 13](#_Toc444109486)

[Hình 8: Mô tả cách cấu hình các tham số trong JADE để chạy một chương trình 14](#_Toc444109487)

**DANH SÁCH CÁC TỪ VIẾT TẮT**

JADE **J**ava **a**gent **de**velopment Framework

FIPA **F**oundation for **I**ntelligent **P**hysical **A**gents

# ĐẶT VẤN ĐỀ

# CHƯƠNG I: CƠ SỞ LÍ THUYẾT

## I.1. Tác tử và hệ đa tác tử

Chương này trước hết giới thiệu các khái niệm về tác tử, tổng quan các công nghệ tác tử, kiến trúc tác tử, các ngôn ngữ lập trình và các công cụ phát triển. Tiếp theo sẽ mô tả các đặc tả của FIPA- tập các tiêu chuẩn phổ biến nhất và được chấp nhận rộng rãi cho phát triển các nền tảng và ứng dụng đa tác tử.

JADE là một nền tảng tuân theo các đặc tả FIPA và hơn nữa nó còn mở rộng mô hình FIPA trong một số lĩnh vực như tác tử cho thiết bị di động, tác tử cho dịch vụ web.

### I.1.1. Khái niệm tác tử

Thuật ngữ tác tử (Agent) hay tác tử phần mềm đã được sử dụng rộng rãi và xuất hiện trong nhiều lĩnh vực nghiên cứu như trí tuệ nhân tạo, cơ sở dữ liệu, các tài liệu về hệ điều hành và mạng máy tính.

Tác tử là một hệ thống máy tính được đóng gói nằm trong một số môi trường có khả năng hành động độc lập và linh hoạt trong môi trường đó nhằm đáp ứng mục tiêu thiết kế của nó. [1]

**MÔI TRƯỜNG**

**TÁC TỬ**

Sensor thu nhận tin hiệu

Phản ứng lại

Hình 1: Tác tử và môi trường của nó

Hình 1 mang đến một cái nhìn trừu tượng về tác tử. Trong hình này, chúng ta thấy hành động phản ứng lại môi trường được tạo ra bởi chính tác tử để ảnh hưởng lại đối với môi trường của chính nó. Trong hầu hết các lĩnh vực phức tạp hợp lý, một tác tử sẽ không hoàn toàn kiểm soát được môi trường của nó mà tác tử chỉ có thể kiểm soát tốt nhất một phần môi trường ảnh hưởng đến nó. Từ góc nhìn của các tác tử, điều này có nghĩa là cùng một hành động được thi hành trong hai hoàn cảnh giống hệ nhau có thể cho kết quả hoàn toàn khác nhau và trong trường hợp cụ thể kết quả đó có thể là kết quả không mong muốn. Tuy nhiên, thông thường một tác tử sẽ có sẽ một tập các hành động sẵn có, tập các hành động này mô tả khả năng có thể có của tác tử, tuy nhiên các hành động trong tập này đều được thi hành trong mọi tình huống. Ví dụ: Hành động nâng một cái bàn chỉ có thể thực hiện được nếu cân nặng của cái bàn đủ nhỏ để tác tử có thể nâng được nó hoặc một ví dụ khác là hành động mua một chiếc xe Ferrari, kết quả của hành động này có thể thất bại như tác tử không đủ tiền để mua chiếc xe đó.

### I.1.2. Các đặc điểm tính chất của tác tử

Tác tử mang một số tính chất sau [5]:

* Tính tự chủ: Các tác tử đóng gói một số trạng thái (không thể truy cập từ các tác tử khác) và ra quyết định làm cái gì đều dựa trên những trạng thái này mà không có sự can thiệp trực tiếp của con người hay những thứ khác.
* Tính phản ứng: Tác tử được đặt trong một môi trường (có thể là thế giới vật chất, hoặc một người dùng thông qua một giao diện người dùng hoặc một tập hợp các tác tử khác, môi trường Internet hoặc môi trường được kết hợp từ những môi trường này) và có thể phản hồi một cách kịp thời với những thay đổi xảy ra trong môi trường đó.
* Tính chủ động: Các tác tử không đơn giản là chỉ hành động để phản ứng lại môi trường mà còn có thể thực hiện các hành vi có mục đích một cách chủ động.
* Khả năng xã hội: Các tác tử tương tác với các tác tử khác thông qua một số loại ngôn ngữ giao tiếp tác tử (agent communication language – ACL) và thường có khả năng tham gia vào các hoạt động xã hội (như giải quyết vấn đề hợp tác và đàm phán) để đạt được mục tiêu của chúng.

### I.1.3. Phân biệt tác tử và đối tượng

### I.1.4. Các loại tác tử

Có bốn loại tác tử:

Tác tử phản xạ đơn giản: hành động theo một quy tác có điều kiện phù hợp với trạng thái hiện thời.

Tác tử phản xạ dựa trên mô hình: sử dụng một mô hình nội bộ để giám sát trạng thái hi thái hiện tại của môi trường.

Tác tử dựa trên mục tiêu:

Tác tử dựa trên lợi ích:

## I.2. Hệ đa tác tử

### I.2.1. Khái niệm

Hệ đa tác tử là hệ thống bao gồm nhiều tác tử có khả năng phối hợp với nhau để giải quyết được những vấn đề phức tạp mà đơn tác tử không thể giải quyết được.

Một hệ đa tác tử là một tập hợp các tác tử có mục đích riêng, miền tri thức riêng nhưng có thể tương tác với nhau để hoàn thành mục tiêu chung tổng thể của hệ thống.

### I.2.2. Sự phối hợp của các tác tử

Phối hợp là tổ chức và quản lý quan hệ phụ thuộc trong hành động của các tác tử sao cho toàn hệ thống hoạt động một cách thống nhất.

Cần có sự phối hợp giữa các tác tử vì:

Hành động của từng tác tử phụ thuộc vào hành động của tác tử khác. Hành động của tác tử phụ thuộc vào nhau trong hai trường hợp: Quyết định của tác tử này ảnh hưởng đến tác tử khác, chẳng hạn khi đá bóng việc tiền đạo chạy lên phía trước sẽ ảnh hưởng tới quyết định của tiền vệ chuyền bóng lên thay vì chyền ngang. Trường hợp thứ hai, hành động của tác tử có thể mâu thuẫn với nhau.

Phối hợp cho phép tránh được tình trạng hỗn loạn. Trong hệ thống bao gồm nhiều tác tử, mỗi tác tử chỉ có thể hình dung cục bộ về môi trường và hành động của mình, hành động của nhiều tác tử có thể mâu thuẫn với nhau, tình trạng hỗn loạn là rất tự nhiên và không thể tránh khỏi nếu không có cơ chế phối hợp.

Phối hợp cho phép đạt được những ràng buộc tổng thể. Ràng buộc tổng thể là ràng buộc mà nhóm tác tử cần thoả mãn trong quá trình hoạt động. Nếu mỗi tác tử làm việc riêng rẽ và đều cố gắng tối ưu hàm mục tiêu riêng của mình thì các ràng buộc này sẽ bị phá vỡ.Không cá thể nào có khả năng thực hiện công việc một mình do hạn chế về tài nguyên, khả năng hoặc thông tin. Nhiều công việc không thể hoàn thành bởi những tác tử hoạt động riêng rẽ do không đủ tài nguyên hoặc thông tin.

### I.2.3. Ưu điểm và nhược điểm

Mỗi tác tử sẽ hành động như một chương trình thông minh, quản lý các tài nguyên trong phạm vi của mình, kết hợp với các tác tử khác để chia sẻ tài nguyên. Với cách tiếp cận này, hệ thống sẽ dễ dàng phát triển và mở rộng vì tác tử hoạt động và phát triển độc lập.

### I.2.4. Ứng dụng

Trong lĩnh vực công nghiệp: kiểm soát tiến trình (Jenning, 1994), chẩn đoán hệ thống (Albert, 2003), sản xuất (Parunak, 1987), dịch vụ vận tải (Neagu, 2006), và quản lý mạng (Greenwood, 2006).

Trong lĩnh vực quản lý thông tin: sử dụng để tìm kiếm và lọc thông tin.

Trong lĩnh vực giao thông vận tải: Hệ thống OASIS là một hệ thống điều khiển không lưu phức tạp dự trên mô hình tác tử BDI, được triển khai và được sử dụng thành công tại sân bay Sydney ở Australia.

## I.3. JADE framework

### I.3.1. Giới thiệu

Telecom Italia (viết tắt là CSELT) đã phát triển những phần mềm đầu tiên, cuối cùng trở thành nền tảng JADE vào cuối năm 1998. Với quan điểm là để cung cấp các dịch vụ cho người phát triển ứng dụng và để dễ dàng sử dụng được và truy cập được cho cả những người phát triển lâu năm và người mới có ít hoặc không có chút kiến thức nào về những đặc tả của FIPA, JADE đặc biệt nhấn mạnh vào sự đơn giản và tiện dụng của các phần mềm API.[3]

JADE đã trở thành mã nguồn mở từ năm 2000 và được phân phối bởi Telecom Italia, đảm bảo tất cả các quyền cơ bản để tạo thuận lợi cho việc sử dụng phần mềm có trong các sản phầm thương mại: quyền làm bản sao của phần mềm và phân phối các bản sao, quyền được truy cập mã nguồn, và quyền được thay đổi mã và thực hiện các cải tiến của nó.

JADE có một website, http://jade.tilab.com, từ đó các phần mềm, tài liệu, mã nguồn ví dụ, và rất nhiều thông tin về cách sử dụng của JADE đều có sẵn. Dự án hoan nghênh sự tham gia của cộng đồng mã nguồn mở với nhiều cách thức để tham gia và đóng góp cho dự án, chúng đều được chi tiết hóa trên trang web.

Khi JADE lần đầu tiên được công bố bởi Telecom Italia, nó đã được sử dụng hầu như chỉ bởi cộng đồng FIPA nhưng khi tích hợp các chức năng lại vượt xa các chi tiết kỹ thuật FIPA. Do đó nó đã được sử dụng bởi một cộng đồng các nhà phát triển được phân phối trên toàn cầu.

Một trong những phần mở rộng của lõi JADE được cung cấp bởi LEAP, một dự án tài trợ một phần bởi Ủy ban châu Âu đã góp phần đáng kể từ năm 2000 và 2002 nhằm hướng JADE tới Java Micro Edition và môi trường mạng không dây. Công việc này được dẫn dắt bởi Giovanni Caire. Ngày nay, nó được dùng như một JADE run-time cho các nền tảng J2ME-CLDC và J2ME-CDC, và nó được sử dụng để giải quyết các vấn đề và thách thức đặt ra trong viễn thông di động, đây được coi là một trong những tính năng hàng đầu của JADE.

### I.3.2. Kiến trúc JADE



Hình 2: Mối quan hệ giữa các thành phần kiến trúc chính trong JADE

Hình 2 mô tả các thành phần kiến trúc chính trong nền tảng JADE. Một nền tảng JADE bao gồm các vùng chứa tác tử (agents container) được phân tán qua mạng. Các tác tử sống trong các vùng chứa, các vùng chứa là các tiến trình Java để chạy JADE và tất cả các dịch vụ cần cho việc lưu trữ và thực thi tác tử. Trong số các vùng chứa thì có một vùng chứa đặc biệt là vùng chứa chính (main container) đây là nơi khởi động của nền tảng JADE. Đây là khung chứa đầu tiên được chạy, tất cả các khung chứa khác phải đăng kí với khung chứa chính. Trong khung chứa chính có một thành phần là bảng khung chứa (Container table – CT) chính là nơi mà đăng kí các đối tượng giao chiếu và địa chỉ truyền thông của tất cả các nút khung chứa trong nền tảng.

Bảng mô tả tác tử toàn cụ (Global agent descriptor table - GADT) là nơi đăng kí của tất các tác tử trong khung chứa, các thông tin đăng kí bao gồm trạng thái hiện tại và vị trí của tác tử.

ASM và DF là hai tác tử đặc biệt cung cấp việc quản lí tác tử, các dịch vụ trang vàng, trang trắng.



Hình 3: Mô hình UML mô tả mối quan hệ giữa các thành phần kiến trúc chính

### I.3.2. Các thành phần JADE

JADE gồm các gói sau:

* *jade.core* là các lõi của hệ thống. Nó bao gồm lớp *Agent*, đây là lớp cơ bản mà tất các lớp phải kế thừa khi lập trình ứng dụng. Bên cạnh đó, lớp *Behaviour* được chứa trong các gói con *jade.core.behaviours*. Lớp *Behaviour* thực hiện nhiệm vụ, hoặc ý định của một agent. Những nhiệm vụ hoặc ý định là các đơn vị hoạt động một lgic có thể được tạo theo những cách khác nhau để thực hiện các nhiệm vụ phức tạp và có thể được thực hiện đồng thời. Người lập trình ứng dụng cần xác định hành vi của các tác tử bằng cách lập trình các hành vi và sự thực tương tác của các hành vi này giữa các tác tử.
* Gói con *jade.lang.acl* được cung cấp để xử lý ngôn ngữ giao tiếp tác tử (Agent Communication Language) dựa theo chuẩn FIPA.
* Gói *jade.content* chứa một tập các lớp để hỗ trợ người dùng định nghĩa các bản thể (ontologies) và nội dung ngôn ngữ. Một hướng dẫn riêng biệt mô tả làm thế nào để sử dụng sự JADE để hỗ nội dung thông tiệp. Trong *jade.content.lang.sl* các SL codec, gồm bộ phân tích cú pháp và các bộ mã hóa.
* Gói *jade.domain* chứa tất cả những lớp Java mà đại diện cho các tác tử quản lý được định nghĩa bởi chuẩn FIPA, cụ thể là các tác tử AMS và DF. Tác tử ASM và DF cung cấp chu kỳ sống, các dịch vụ trang trắng và trang vàng. Các gói con *jade.domain.FIPAAgentManagement* chứa FIPA-Agent-Management Ontology và tất cả các lớp đại diện cho các khái niệm của nó. Các gói con *jade.domain.JADEAgentManagement* chứa các phần mở rộng của JADE cho việc quản lí tác tử (Agent – Management) (ví dụ như đánh hơi tin nhắn, kiểm soát vòng đời của các agent ...), bao gồm cả bản thể và tất cả các lớp đại diện cho các khái niệm của nó. Các gói con *jade.domain.introspection* chứa các khái niệm được sử dụng cho việc đàm thoại giữa các công cụ JADE (ví dụ như Sniffer và Introspector) và hạt nhân JADE. Các gói *jade.domain.mobility* phụ có chứa tất cả các khái niệm sử dụng để giao tiếp về tính di động (mobility).
* Các gói *jade.gui* chứa một bộ các lớp tổng quát hữu ích để tạo GUI để hiển thị và sửa đổi định danh tác tử (Agent-Identifiers), các mô tả tác tử (Agent Descriptions), các thông điệp ngôn ngữ giao tiếp giữa các tác tử (ACLMessages), …

Các gói *jade.mtp* chứa một giao diện (interface) Java mà mỗi giao thức chuyển thông điệp nên thực hiện để có thể được dễ dàng tích hợp với nền tảng JADE, và thực hiện một tập hợp các giao thức này.

Gói *jade.proto* là gói chứa các lớp để mô hình các giao thức tương tác chuẩn (Ví dụ: Các giao thức theo chuẩn FIPA như fipa-request, fipa-query, fipa-contract-net, fipa-subscribe, ...) và các lớp để giúp người lập trình ứng dụng tạo ra các giao thức của riêng mình.

Các gói FIPA chứa các module IDL xác định bởi FIPA cho IIOP dựa trên việc truyền tin nhắn.

Cuối cùng, gói *jade.wrapper* cung cấp bao đóng cho các chức năng mức độ cao của JADE cho phép người sử dụng JADE như một thư viện, các ứng dụng Java bên ngoài khởi động các tác tử JADE và các vùng chứa tác tử (container agents)

**AMS**: đại diện quyền lực trong một flatform, là agent duy nhất có thể thực hiện hoạt động quản lý bắt đầu, tạm đừng hoặc kết thúc toàn bộ flatform, (agent bình thường có thể yêu cầu hoạt động như vậy tới AMS).

**DF**: cung cấp dịch vụ Page Yellow, nơi các agent có thể tạo ra các dịch vụ nó cung cấp và tìm các agent khác cung cấp các dịch vụ nó cần.

### I.3.2. Sự giao tiếp giữa các tác tử trong JADE

Sự giao của tác tử có lẽ là tính năng cơ bản nhất của JADE và được cài đặt tuân theo các thông số kĩ thuật FIPA. Mô hình truyền thông dựa trên việc truyền thông điệp không đồng bộ (Asynchronous Message Passing). Do đó, mỗi tác tử có một hộp thư hay gọi chính xác hơn là hàng đợi tin nhắn tác tử, nơi JADE trong khi chạy đã đẩy thông điệp được gửi bởi các tác tử khác vào. Bất cứ khi nào một thông điệp được gửi vào trong hộp thư, các tác tử nhận sẽ được thông báo. Tuy nhiên, khi nào hoặc nếu, các tác tử lấy các thông điệp từ hàng đợi để xử lí thì còn phụ thuộc sự lựa chọn thiết kế của người lập trình. Quá trình này được mô tả trong hình 4.

JADE chạy phân tán

Chuẩn bị thông điệp gửi đến tác tử 2

Lấy thông điệp từ hàng đợi thông điệp và xử lí nó

Đẩy thông điệp vào hàng đợi thông điệp của tác tử 2

Hình 4: Mô hình truyền thông điệp không đồng bộ JADE

Các định dạng đặc biệt của thông điệp trong JADE phải phù hợp với định nghĩa của cấu trúc tin nhắn FIPA – ACL. Mỗi thông điệp bao gồm các trường sau:

* Tác tử gửi thông điệp
* Danh sách các tác tử nhận.
* Các cách thức giao tiếp (hay còn gọi là biểu hiện - performative) cho thấy ý định của phía gửi bằng cách gửi tin nhắn. Ví dụ, nếu các biểu hiện là yêu cầu (REQUEST), phía gửi muốn người nhận để thực hiện một hành động, nếu nó là thông báo (INFORM) người gửi muốn người nhận phải nhận thức được một thực tế, nếu nó là một đề xuất (PROPOSE) hay một gọi đề xuất (Call for proposal – CFP) thì phía gửi muốn tham gia vào một cuộc đàm phán.
* Các nội dung có chứa các thông tin thực tế được trao đổi bằng thông điẹp (Ví dụ: hành động được thực hiện trong một thông điệp REQUEST, hoặc sự việc mà người gửi muốn tiết lộ trong một thông điệp INFORM, …)
* Ngôn ngữ thể hiện nội dung cho thấy cú pháp sử dụng để thể hiện nội dung. Cả người gửi và người nhận phải có khả năng mã hóa và phân tích cách diễn tả phù hợp với cú pháp này để các giao tiếp có hiệu quả.
* Bản thể (ontology) chỉ ra ý nghĩa của những kí hiệu được sử dụng trong nội dung. Cả người gửi và người nhận phải mô tả cùng một ý nghĩa với các kí hiệu để các giao tiếp có hiệu quả.
* Một số trường bổ sung được sử dụng để kiểm soát một số cuộc hội thoại song song và để xác định thời gian chờ để nhận được một câu trả lời như là định danh cuộc hội thoại (conversation-id), hồi đáp – với (reply-with), hồi đáp cho và hồi đáp bởi (in-reply-to and reply-by).

Một tin nhắn trong JADE được thực hiện như một đối tượng của lớp jade.lang.acl.ACLMessage cung cấp và thiết lập các phương pháp để truy cập tất cả các trường theo quy định của các định dạng ACL. Tất cả biểu hiện định nghĩa trong đặc tả FIPA được ánh xạ như hằng số trong các lớp ACLMessage.

#### I.3.2.1. Gửi thông điệp

Việc gửi thư tới tác tử khác cũng đơn giản như việc điền vào các trường của một đối tượng ACLMessage và sau đó gọi phương thức send() của lớp Agent. Đoạn mã dưới đây tạo ra một thông điệp dạng INFORM để báo cho một tác tử có biệt danh là Peter với nôi dung “Today it’s raining”:

ACLMessage msg = new ACLMessage(ACLMessage.INFORM);

msg.addReceiver(new AID("Peter", AID.ISLOCALNAME));

msg.setLanguage("English");

msg.setOntology("Weather-forecast-ontology");

msg.setContent("Today it’s raining");

send(msg);

#### I.3.2.2. Nhận thông điệp

JADE trong khi chạy tự động đưa các thông điệp vào hàng đợi tin nhắn của tác tử nhận ngay khi thông điệp đến. Một tác tử có thể lấy thông điệp từ hàng đợi thông điệp của mình bằng phương thức receive(). Phương pháp này trả về thông điệp đầu tiên trong hàng đợi thông điệp, hoặc không có gì nếu hàng đợi thông điệp trống rỗng một cách lập tức.

ACLMessage msg = receive();

if (msg != null) {

// Xử lí thông điệp

}

#### I.3.2.3. Lưa chọn thông điệp từ hàng đợi thông điệp

Khi thông điệp được gửi đến hàng đợi thông điệp tác tử có thể không nhất thiết lựa chọn thông điệp đầu tiên trong hàng đợi mà tác tử lựa chọn dạng thông điệp mà nó muốn nhận. Muốn vậy, phía gửi gói thông điệp mà nó muốn gửi đến phía nhận trong một khuôn (template) thông qua phương thức:

MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.ACCEPT\_PROPOSAL);

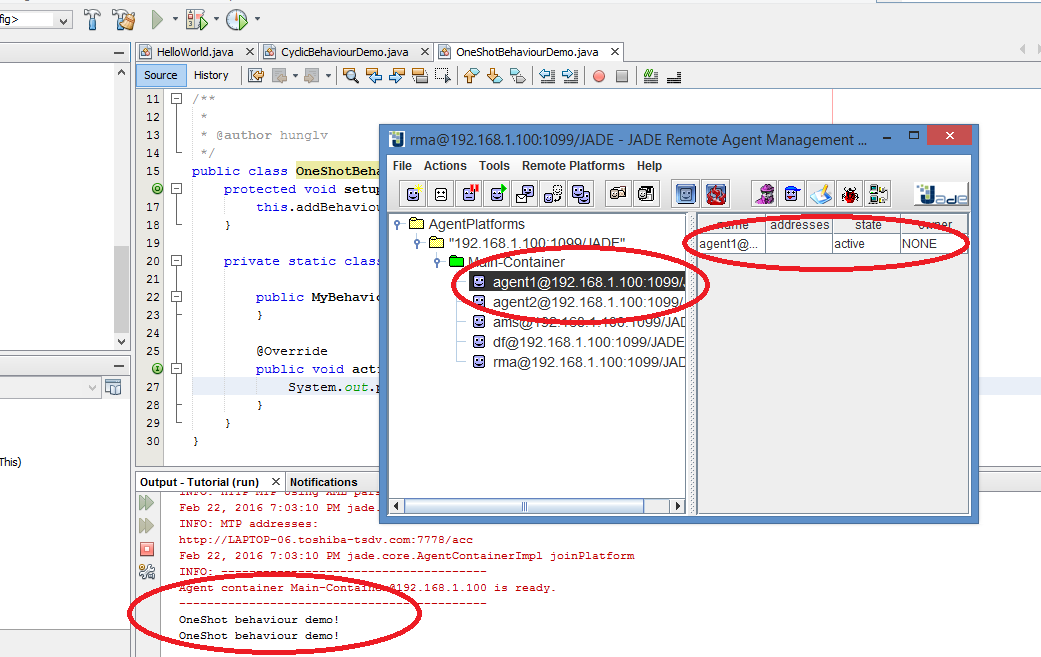
ACLMessage.ACCEPT\_PROPOSAL là việc quy định dạng thông điệp muốn gửi đi, phương thức MessageTemplate.MatchPerformative() đưa thông điệp muốn gửi đi vào một cái khuôn. Khi thông điệp đến hàng đợi của phía gửi, phía gửi sẽ lựa chọn từ hàng đợi thông điệp cái thông điệp đầu tiên có dạng ACLMessage.ACCEPT\_PROPOSAL.

### I**.3.3. Các hành vi của tác tử**

Hành vi là một công việc mà tác tử có thể thực thi. Các tác tử có rất nhiều loại hành vi, các hành vi của mỗi tác tử đều có đặc điểm chung là đều thực hiện hai phương thức chính là phương thức action() và phương thức done(), các hành vi của tác tử đều được cài đặt như một đối tượng của lớp kế thừa từ lớp *jade.core.behaviours.Behaviour*. Tuy nhiên, trong nội dung của môn học em xin trình bày bốn loại hành vi chính hay được sử dụng là: Hành vi thực hiện một lần (OneShotBehaviour), hành vi thực hiện mãi mãi (CyclicBehavior), hành vi đánh thức (WakerBehaviour), hành vi thực hiện có chu kì (TickerBehavior).

Hành vi thực hiện một lần (OneShotBehaviour): Đây là kiểu hành vi chỉ thực hiện một lần duy nhất nên phương thức action() chỉ thực hiện duy nhất 1 lần. Vì thế giá trị trả về của phương thức done() mặc định là true. Kiểu hành vi này thuộc lớp jade.core.behaviours.OneShotBehaviour.

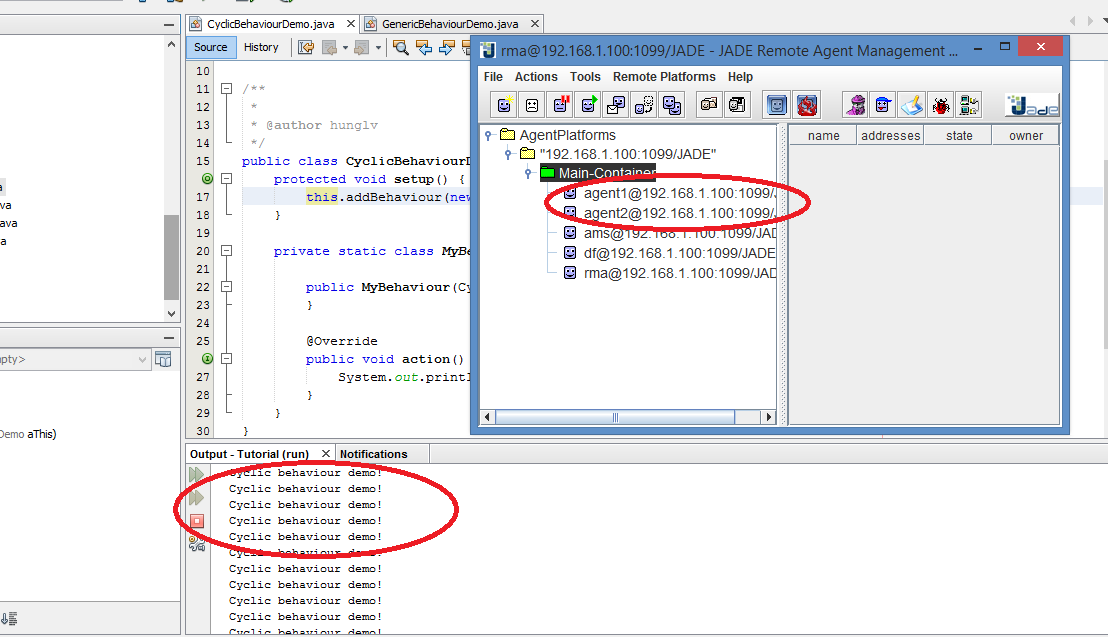
Ví dụ về kiểu hành vi thực hiện một lần được trình bày ở hình 2:



Hình 5: Ví dụ về kiểu hành vi thực hiện một lần

Hành vi thực hiện mãi mãi (CyclicBehaviour): Đây là kiểu hành vi thực hiện mãi mãi, vì thế phương thức action() thực thi cùng một hành vi một cách liên tục. Vì thế kết quả trả về của phương thức done() luôn luôn là false. Kiểu hành vi này thuộc lớp jade.core.behaviours.CyclicBehaviour.

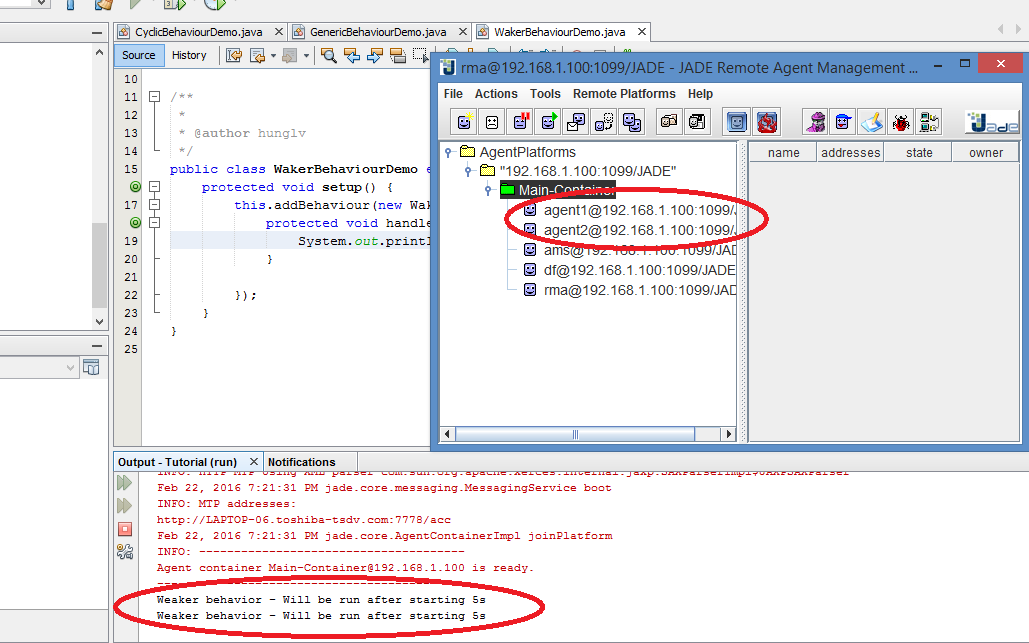
Ví dụ về kiểu hành vi thực hiện mãi mãi được mô tả như ở hình 3.



Hình 6: Ví dụ kiểu hành vi thực hiện mãi mãi

Hành vi đánh thức (WakeBehaviour): Đây là hành vi được thực thi duy nhất một lần sau một khoảng thời gian xác đinh vì thế phương thức action() và phương thức done() đã thực sự được cài đặt vì thế phương thức handleElapsedTimeout() được cài đặt như một lớp con và được thực thi sau một khoảng thời gian xác định (dưới dạng ms). Kiểu hành vi này thuộc lớp jade.core.behaviours.WakerBehaviour.

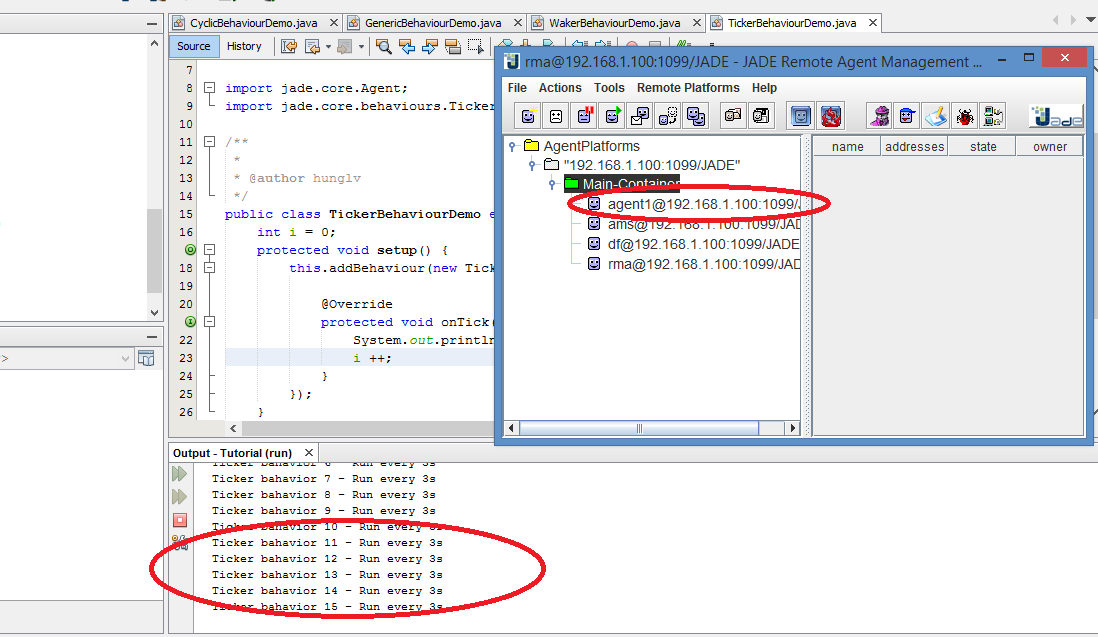
Ví dụ về kiểu hành đánh thức được mô tả ở hình 3



Hình 7: Ví dụ về kiểu hành vi đánh thức

Hành vi thực hiện có chu kì (TickerBehaviour): Đây là loại hành vi thực hiện mãi mãi trừ phi phương thức done() được thực thi. Phương thức action() và done() đã được cài đặt sẵn vì vậy phương thức onTick() được thực hiện định kì với một khoảng thời gian nhất định (đặc tả dưới dạng ms). Kiểu hành vi này thuộc lớp jade.core.behaviours.TickerBehaviour.

Ví dụ về kiểu hành vi thực hiện có chu kì được mô tả ở hình 5.



Hình 8: Ví dụ về kiểu hành vi thực hiện có chu kì

# CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ PHẦN MỀM

# CHƯƠNG III: CÀI ĐẶT VÀ TRIỂN KHAI

## III.1 Cài đặt JADE

Trước tiên, chúng ta tải framework JADE tại trang chủ sau: <http://jade.tilab.com/>. Khởi động Netbean và tạo một project mới, thêm các thư viện sau vào phần “Set Project Configuration” trong của project vừa tạo: http.jar, iiop.jar, jade.jar, jadeTools.jar, jadeExamples.jar, commons-codec-1.3.jar.



Hình 9: Mô tả cách thêm các thư viện JADE để cấu hình một project

Khi tạo một lớp tác tử mới phải kế thừa từ lớp Agent của lớp *jade.core.Agent,* phần mã chương trình sẽ được viết trong phương thức setup()và được cấu hình với các tham số:

* Main class: jade.Boot
* Arguments: “-gui ‘Tên tác tử’:’Lớp thực thi tác tử’.

Ví dụ: “-gui agent1:behaviour.TickerBehaviourDemo”



Hình 10: Mô tả cách cấu hình các tham số trong JADE để chạy một chương trình

Jf dung để khai báo yello page

Các phương thức của Aggent:

getName(): xxxxxx@IP:port/JADE (VD: [Hello@192.168.1.101:1099/JADE](mailto:Hello@192.168.1.101:1099/JADE))

getLocalName(): Trả lại tên: xxxxx (VD: Hello)

Agent này sẽ sống mãi vì thế, khi show trạng thái của nó luôn là active. Để agent tự động xóa đi khi chạy xong thì ta sử dụng phương thức doDelete(), sau khi agent được chạy xong nó sẽ tự động xóa đi.

Truyền tham số cho các agent:

Vào customize -> Edit Arguments bằng cách thêm các tham số theo mẫu (Hung\_Le,Van\_Anh)

Gửi thông điệp giữa các Agents

Tạo 2 class Sender và Receiver, 2 class này đều extend lớp Agent.

# CHƯƠNG 4: KẾT LUẬN

Kiến thức thu được, kinh nghiệm, hướng phát triển?

**TÀI LIỆU THAM KHẢO**

[1]Nicholas R. Jennings and Michael Wooldridge, “Agent-Oriented Software Engineering”, Department of Electronic Engineering Queen Mary & Westfield College University of London London E1 4NS, United Kingdom, pp 5.

[2] Michael Wooldridge, “Agent-Based Software Engineering”, Mitsubishi Electric Digital Library Group, 18th Floor, Centre Point, 103 New Oxford Street London WC1A 1EB, United Kingdom. September 19, 1997.

[3] John Wiley&Sons, “Developing Multi-Agent Systems with JADE”, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England. Copyright 2007.

[1] Giovanni Caire , 2009. Jade programing for beginers.pdf. TILAB, formerly CSELT

[2] Loạt bài hướng dẫn lập trình hướng agent - <https://www.youtube.com/watch?v=VHt9A9Jn1Jk&list=PL6_53ptjaqs4zs-xSK7SwabqC-CjubpIV>

[3]