|  |
| --- |
| HỌC VIỆN KỸ THUẬT MẬT MÃ  **KHOA AN TOÀN THÔNG TiN**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯ |
| **BÀI TẬP MÔN HỌC Kĩ Thuật Lập Trình**  *Đề tài: Socket Programing* |
| Ngành: An toàn thông tin  Sinh viên thực hiện:  Phạm Tiến Dũng - AT160409 Phùng Việt Đức - AT160413 Nguyễn Văn Hai – AT160416  Tống Thế Bảo – AT1604161  PhạmMinhQuang–AT160438  Lớp: A16C404  Giảng viên:  **TS. Nguyễn Mạnh Thắng**  Khoa An Toàn Thông Tin – Học viện Kỹ thuật mật mã  Hà Nội, 2022 |



## NHẬN XÉT VÀ CHO ĐIỂM CỦA GIÁO VIÊN

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

.....................................................................................................................................

## MỤC LỤC

[Lời nói đầu 1](#_bookmark0)

[Chương 1: Những khái niệm cơ bản của thuật toán 5 triết gia ăn tối 2](#_bookmark1)

* 1. [Vấn đề thuật toán 2](#_bookmark2)
  2. [Deadlock 3](#_bookmark3)
     1. [Định nghĩa Deadlock 3](#_bookmark4)
     2. [Điều kiện hình thành tắt nghẽn 3](#_bookmark5)
  3. [Giải quyết tắc nghẽn bằng thuật toán Semaphore 3](#_bookmark6)
     1. [Thuật toán Semaphore là gì? 3](#_bookmark7)
     2. [Giải quyết bằng thuật toán 4](#_bookmark8)

[Chương 2: Thực hiện giải quyết bài toán trên bằng ngôn ngữ Java 5](#_bookmark9)

* 1. [Khởi tạo ban đầu 5](#_bookmark10)
  2. [Chạy chương trình 6](#_bookmark11)
     1. [Ý tưởng thuật toán 6](#_bookmark12)
     2. [Code giải thuật 7](#_bookmark13)
  3. [Kết quả của quá trình: 10](#_bookmark14)

[PHỤ LỤC 11](#_bookmark15)

[KẾT LUẬN 14](#_bookmark16)

[TÀI LIỆU THAM KHẢO 15](#_bookmark17)

## LỜI NÓI ĐẦU

Nếu không có phần mềm, máy tính chỉ là một thiết bị điện tử thông thường. Với sự hỗ trợ của phần mềm, máy tính có thể lưu trữ, xử lý thông tin và người sử dụng có thể gọi lại được thông tin này. Phần mềm máy tính có thể chia thành nhiều loại: chương trình hệ thống, quản lý sự hoạt động của chính máy tính. Chương trình ứng dụng, giải quyết các vấn đề liên quan đến việc sử dụng và khai thác máy tính của người sử dụng. Hệ điều hành thuộc nhóm các chương trình hệ thống và nó là một chương trình hệ thống quan trọng nhất đối với máy tính và cả người sử dụng. Hệ điều hành điều khiển tất cả các tài nguyên của máy tính và cung cấp một môi trường thuận lợi để các chương trình ứng dụng do người sử dụng viết ra có thể chạy được trên máy tính..

Một máy tính hiện đại có thể bao gồm: một hoặc nhiều processor, bộ nhớ chính, clocks, đĩa, giao diện mạng, và các thiết bị vào/ra khác. Tất cả nó tạo thành một hệ thống phức tạp. Để viết các chương trình để theo dõi tất cả các thành phần của máy tính và sử dụng chúng một cách hiệu quả, người lập trình phải biết processor thực hiện chương trình như thế nào, bộ nhớ lưu trữ thông tin như thế nào, các thiết bị đĩa làm việc (ghi/đọc) như thế nào, lỗi nào có thể xảy ra khi đọc một block đĩa,

… đây là những công việc rất khó khăn và quá khó đối với người lập trình. Nhưng rất may cho cả người lập trình ứng dụng và người sử dụng là những công việc trên đã được hệ điều hành hỗ trợ nên họ không cần quan tâm đến nữa. chúng ta cần tiềm hiểu về hệ điều hành để có một cái nhìn tổng quan về những gì liên quan đến việc thiết kế cài đặt cũng như chức năng của hệ điều hành để hệ điều hành đạt được mục tiêu: Giúp người sử dụng khai thác máy tính dễ dàng và chương trình của người sử dụng có thể chạy được trên máy tính.

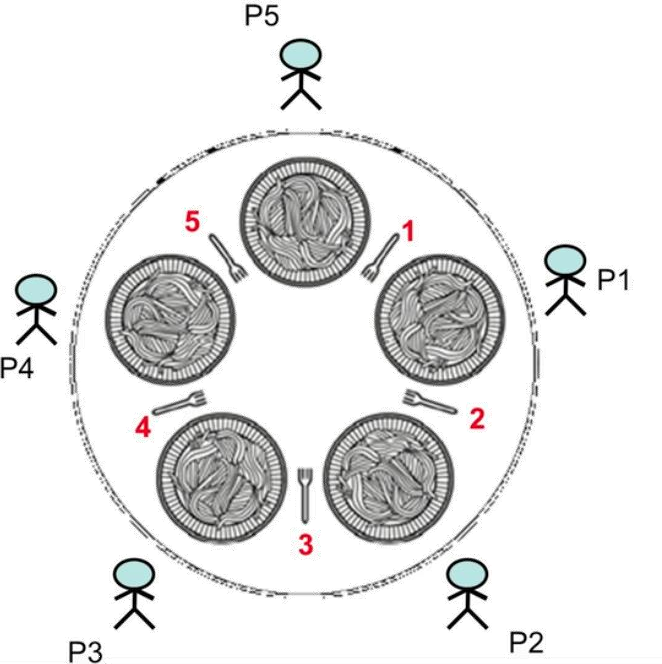
"**Bài toán 5 triết gia ăn tối**" là một bài toán kinh điển về tương tranh và chia sẻ tài nguyên. Việc nghiên cứu bài toán sẽ cho chúng ta hiểu rõ hơn về khía cạnh này của hệ điều hành.

## CHƯƠNG 1: NHỮNG KHÁI NIỆM CƠ BẢN CỦA THUẬT TOÁN 5 TRIẾT GIA ĂN TỐI

## Vấn đề thuật toán

Đây là bài toán cổ điển về hệ điều hành. Bài toán bữa tối của các triết gia được đưa ra bởi nhà toán học E. W. Dijkstra. Bài toán được mô tả như sau: Có năm triết gia cùng ngồi ăn tối quanh một chiếc bàn tròn, trước mặt mỗi người có một đĩa mì Ý, giữa 2 triết gia thì có một chiếc nĩa. Mỗi triết gia dành toàn bộ thời gian để suy nghĩ hoặc ăn khi đói. Mỗi triết gia chỉ có thể ăn khi có được 2 chiếc nĩa bên cạnh mình.

* + - Đói: một triết gia có thể chết đói nếu ông ta không có cách nào để ăn được
    - Tắc nghẽn: các triết gia phải đợi lẫn nhau nên không có ai ăn được.



*Hình vẽ đại diện cho 5 triết gia ngồi quanh bàn tròn cùng với 5 chiếc nĩa*

Phải đặt ra thuật toán sao cho khi một triết gia bị đói thì ông ta sẽ được ăn và đảm bảo không có triết gia nào bị chết đói. Bài toán đặt ra vấn đề “đồng bộ giữa các tiến trình”, giải quyết vấn đề tắc nghẽn có thể xảy ra.

## Deadlock

* + 1. *Định nghĩa Deadlock*

Trong môi truờng đa chương, nhiều quá trình có thể cạnh tranh một số giới hạn tài nguyên. Một quá trình yêu cầu tài nguyên, nếu tài nguyên không sẳn dùng tại thời điểm đó, quá trình đi vào trạng thái chờ. Quá trình chờ có thể không bao giờ chuyển trạng thái trở lại vì tài nguyên chúng yêu cầu bị giữ bởi những quá trình đang chờ khác. Trường hợp này được gọi là deadlock (khoá chết).

* + 1. *Điều kiện hình thành tắt nghẽn*

Deadlock chỉ xảy ra khi có đủ 4 điều kiện sau:

* + - * Loại trừ tương hỗ (Mutual Exclusion): Tại một thời điểm, tài nguyên không thể chia sẽ được hệ thống cấp phát cho một tiến trình duy nhất. Tiến trình khác không thể sử dụng cho đến khi tài nguyên được giải phóng.
      * Giữ và chờ (Hold and Wait) : Mỗi tiến trình trong tập hợp tiến trình đang giữ một tài nguyên và chờ đợi để được cấp phát một tài nguyên mới.
      * Không có quyền ưu tiên (No Preemption): Một tiến trình không thể chiếm giữ tài nguyên cho đến khi tài nguyên đó được giải phóng bởi tiến trình đang sử dụng nó.
      * Tồn tại chu kỳ chờ (Circular wait): Các tiến trình giữ một tài nguyên và chờ nhận một tài nguyên khác bởi tiến trình khác. Chúng nối đuôi nhau tạo thành vòng tròn. Chờ vô tận.

## Giải quyết tắc nghẽn bằng thuật toán Semaphore

* + 1. *Thuật toán Semaphore là gì?*

Semaphore là lời giải kinh điển cho bài toán 5 triết gia ăn tối, mặc dù nó không ngăn được hết các Deadlock. Để giải quyết Deadlock, ở vị trí cuối cùng ta sẽ cho triết gia đó được lấy đũa bên phải mình trước sau đó mới lấy đũa bên trái trong khi các triết gia khác sẽ lấy đũa bên trái trước. Điều đó chắc chắn rằng triết gia

vị trí cuối luôn sẵn sàng ăn đầu tiên khi mà các triết gia khác đang cố lấy chiếc đũa bên phải của mình.

* + 1. *Giải quyết bằng thuật toán:*

**while**(true) {

// suy nghĩ

think();

//lấy dĩa pick\_up\_left\_fork(); pick\_up\_right\_fork();

//ăn

eat();

//thả nĩa put\_down\_right\_fork(); put\_down\_left\_fork();

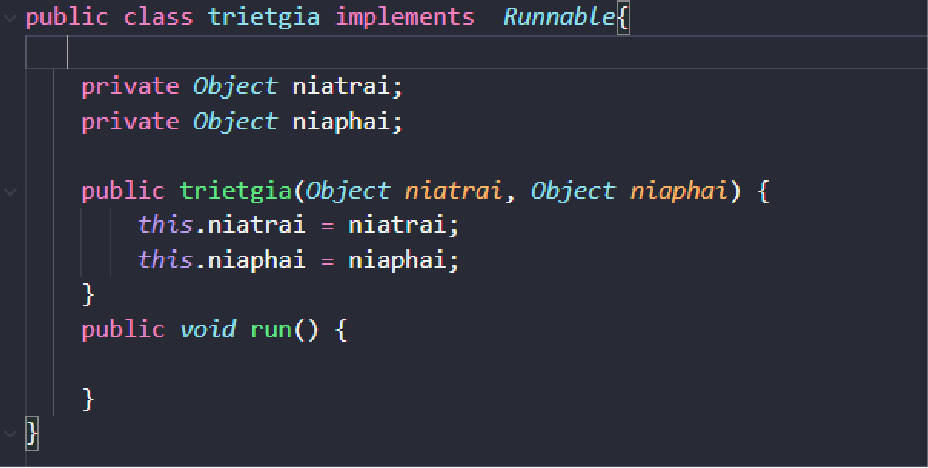
//không còn đói, quay lại suy nghĩ tiếp

}

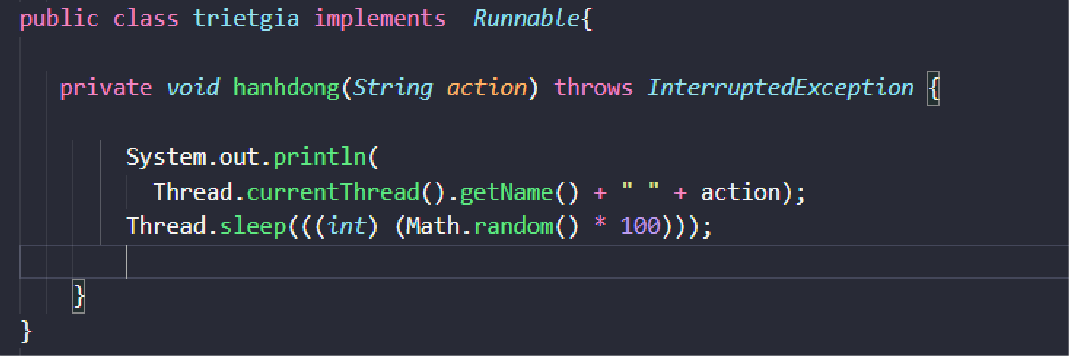
## CHƯƠNG 2: THỰC HIỆN GIẢI QUYẾT BÀI TOÁN TRÊN BẰNG NGÔN NGỮ JAVA

## Khởi tạo ban đầu

Đầu tiên mô hình hóa mỗi triết gia thành các lớp triển khai giao diện Runnable để ta có thể chạy chúng dưới dạng các luồng riêng biệt. Mỗi triết gia có quyền truy cầm vào cả 2 chiếc nĩa ở bên trái và phải



Khởi tạo 1 phương thức giúp triết gia thực hiện các hành động suy nghĩ, ăn, lấy or thả nĩa



Để mô phỏng việc lấy được 1 chiếc nĩa. Ta cần khóa nó để không có 2 chuỗi trietgia nào có được nó cùng 1 lúc. Để đặt được điều này. Ta sử dụng *synchronized* để có được trình giám sát nội bộ đối tượng nĩa và ngăn các luồng khác làm điều tương tự



## Chạy chương trình

* + 1. *Ý tưởng thuật toán*

Thực hiện theo đúng mô tả trước đó: một nhà triết gia suy nghĩ 1 lúc và sau đó quyết định ăn. Sau đó, anh ta lấy nĩa bên trái và bên phải của mình và bắt đầu ăn. Khi hoàn thành, anh ta sẽ thả nĩa xuống.

* + 1. *Code giải thuật*

Full code ( class trietgia):

# public class trietgia implements *Runnable*{

private *Object* niatrai; private *Object* niaphai;

public trietgia(*Object niatrai*, *Object niaphai*) {

# *this*.niatrai = niatrai;

*this*.niaphai = niaphai;

}

private *void* hanhdong(*String action*) throws

*InterruptedException* {

action);

System.out.println( Thread.currentThread().getName() + " " +

Thread.sleep(((*int*) (Math.random() \* 100)));

}

@*Override*

public *void* run() { try {

# while (true) {

hanhdong( ": Suy nghi ");

synchronized (niatrai) { hanhdong( ": Cam nia trai ");

synchronized (niaphai) {

// eating

hanhdong( ": Cam nia phai");

hanhdong(": Bo xuong nia phai");

}

// Back to thinking

hanhdong(": Bo xuong nia trai - Tiep

tuc suy nghi ");

}

}

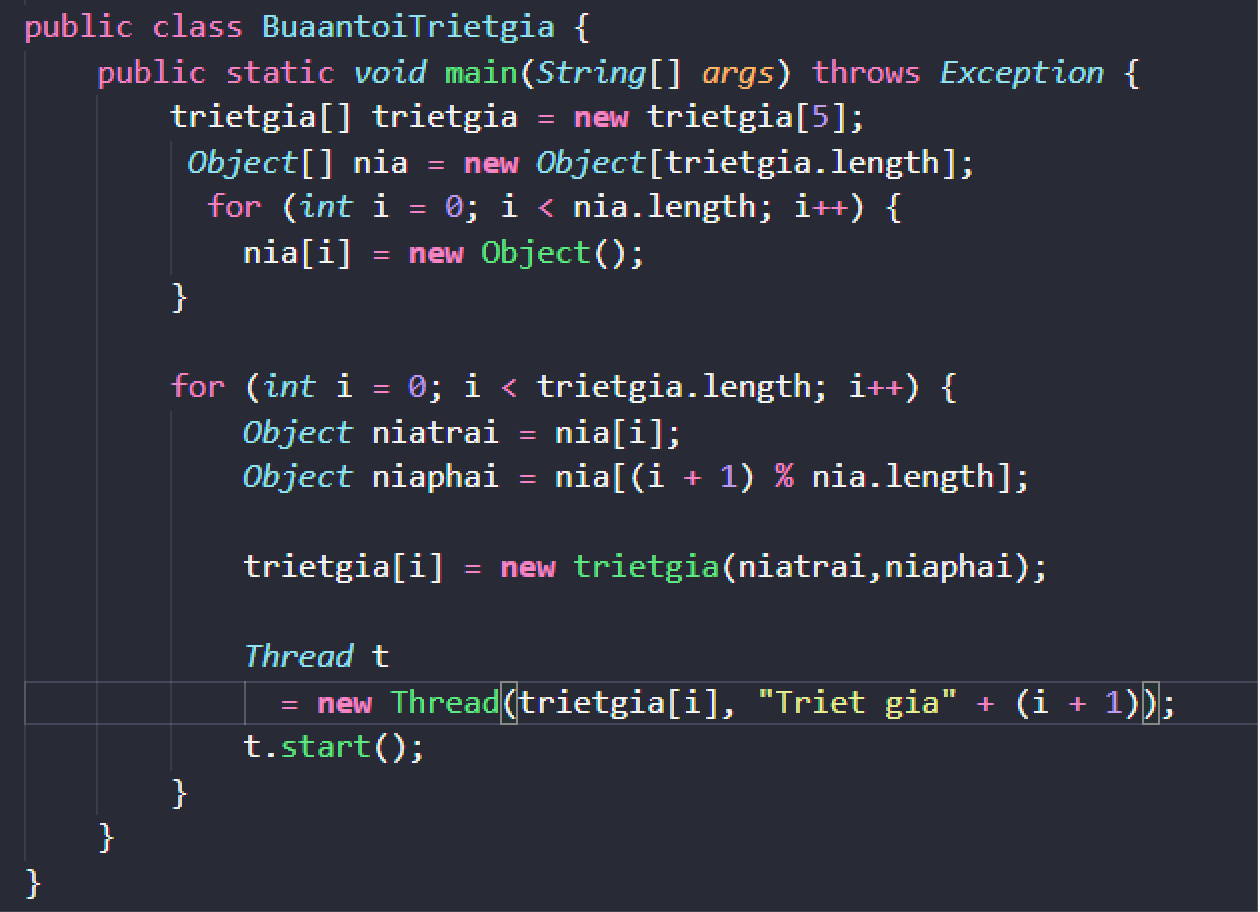
} catch (*InterruptedException e*) { Thread.currentThread().interrupt(); return;

}

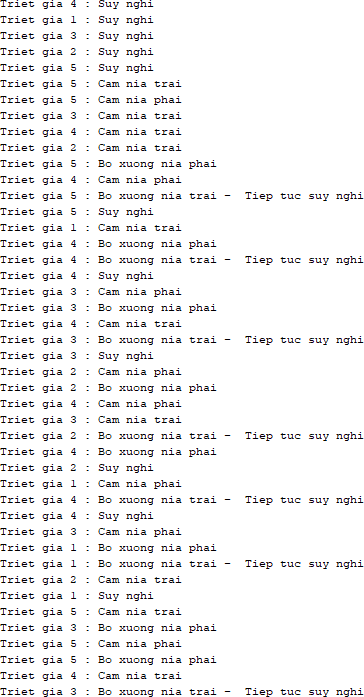
}

}

Để bắt đầu toàn bộ quá trình. Ta tạo 1 class BuaantoiTrietgia



## Kết quả của quá trình:



## PHỤ LỤC

FULL CODE bài làm:

## Class BuaantoiTrietgia:

public class BuaantoiTrietgia {

public static *void* main(*String*[] *args*) throws

*Exception* {

trietgia[] trietgia = **new** trietgia[5]; *Object*[] nia = **new** *Object*[trietgia.length]; for (*int* i = 0; i < nia.length; i++) {

nia[i] = **new** Object();

}

for (*int* i = 0; i < trietgia.length; i++) {

*Object* niatrai = nia[i];

*Object* niaphai = nia[(i + 1) % nia.length];

if (i == trietgia.length - 1) {

trietgia[i] = **new** trietgia(niaphai,

niatrai);

} else {

trietgia[i] = **new** trietgia(niatrai,

niaphai);

}

*Thread* t = **new** Thread(trietgia[i], "trietgia " + (i + 1));

t.start();

}

}

}

**Class trietgia:**

public class trietgia implements *Runnable*{

private *Object* niatrai; private *Object* niaphai;

public trietgia(*Object niatrai*, *Object niaphai*) {

*this*.niatrai = niatrai;

*this*.niaphai = niaphai;

}

private *void* hanhdong(*String action*) throws

*InterruptedException* {

System.out.println( Thread.currentThread().getName() + " " +

action);

Thread.sleep(((*int*) (Math.random() \* 100)));

# }

@*Override*

public *void* run() { try {

# while (true) {

hanhdong( ": Suy nghi ");

synchronized (niatrai) { hanhdong( ": Cam nia trai ");

synchronized (niaphai) { hanhdong( ": Cam nia phai");

hanhdong(": Bo xuong nia phai");

}

hanhdong(": Bo xuong nia trai - Tiep

tuc suy nghi ");

}

}

} catch (*InterruptedException e*) { Thread.currentThread().interrupt(); return;

# }

}

}

## KẾT LUẬN

1. Những kết quả đạt được của bài tập môn học
   * Giúp sinh viên nắm được các kiến thức cơ bản về hệ điều hành cũng như thuật toán 5 triết gia ăn tối.
   * Nắm được các kiến thức liên quan đến môn học và ứng dụng của nó.
   * Nâng cao khả năng làm việc nhóm.
   * Biết được cách triển khai thuật toán bằng ngôn ngữ lập trình Java.
2. Hướng phát triển tiếp theo và đề xuất
   * Triển khai các thuật toán vào thực hành thực tế, và ứng dụng thuật toán vào các môn học sau.

## TÀI LIỆU THAM KHẢO

Những tài liệu tham khảo để thực hiện được đề tài.

[1]. https://viblo.asia/p/tim-hieu-ve-dong-bo-luong-trong-java-voi-bai-toan-bua-toi- cua-cac-triet-gia-OeVKBdMrlkW

[2].https://[www.geeksforgeeks.org/dining-philosopher-problem-using-](http://www.geeksforgeeks.org/dining-philosopher-problem-using-) semaphores/#:~:text=The%20Dining%20Philosopher%20Problem%20%E2%80% 93%20The,two%20chopsticks%20adjacent%20to%20him.

[3]. Slide bài giảng.