BỘ CÔNG THƯƠNG

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

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



**ĐỒ ÁN MÔN HỌC:**

**TÌM HIỂU VỀ THREAD, MULTI-THREAD VÀ DEMO ỨNG DỤNG**

**GVHD:NGUYỄN VĂN TÙNG**

**NHÓM: 02**

**THÀNH VIÊN:**

**Lê Lưu Hoàng Nhân (NT)**

**Nguyễn Văn Hiền**

**Nguyễn Hữu Đức**

**Nguyễn Duy Thịnh**

**Dương Trung Hiếu**

TP.Hồ Chí Minh, tháng 10 năm 2021

BỘ CÔNG THƯƠNG

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

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



**ĐỒ ÁN MÔN HỌC:**

**TÌM HIỂU VỀ THREAD, MULTI-THREAD VÀ DEMO ỨNG DỤNG**

**GVHD: NGUYỄN VĂN TÙNG**

**NHÓM: 02**

**THÀNH VIÊN:**

**Lê Lưu Hoàng Nhân (NT)**

**Nguyễn Văn Hiền**

**Nguyễn Hữu Đức**

**Nguyễn Duy Thịnh**

**Dương Trung Hiếu**

TP.Hồ Chí Minh, tháng 10 năm 2021

**LỜI CAM ĐOAN**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các số liệu, kết quả nêu trong Đồ án là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác. Tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện Đồ án này đã được cảm ơn và các thông tin trích dẫn trong Đồ án đã được chỉ rõ nguồn gốc.

**Sinh viên thực hiện Đồ án**

Lê Lưu Hoàng Nhân

Nguyễn Văn Hiền

Nguyễn Hữu Đức

Nguyễn Duy Thịnh

Dương Trung Hiếu

**LỜI CẢM ƠN**

Chúng em xin chân thành cảm ơn **Khoa Công nghệ thông tin**, trường **đại học Đại học Công nghiệp Thực phẩm Thành Phố Hồ Chí Minh** đã tạo điều kiện thuận lợi cho chúng em học tập và thực hiện đề tài này.  
Chúng em xin bày tỏ lòng biết ơn sâu sắc tới thầy **Nguyễn Văn Tùng** đã tận tình hướng dẫn chỉ bảo chúng em trong quá trình thực hiện đề tài.  
Chúng em xin chân thành cảm ơn quý thầy cô trong khoa Công nghệ Thông tin đã tận tình giảng dạy , trang bị cho em những kiến thức quý báu trong năm vừa qua.  
Chúng con xin chân thành cảm ơn ông bà, cha mẹ đã luôn động viên ủng hộ vật chất lẫn tinh thần trong suốt thời gian qua.  
Chúng em xin cảm ơn sự quan tâm giúp đỡ và ủng hộ của các anh chị bạn bè trong quá trình thực hiện khóa đề tài. Mặc dù đã cố gắng hoàn thành đồ án trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót.

Chúng em rất mong nhận được sự thông cảm, góp ý và tận tình chỉ bảo của quý thầy cô và các bạn.

**MỤC LỤC**

[PHẦN MỞ ĐẦU 5](#_Toc86131641)

[I. Lý do chọn đề tài: 5](#_Toc86131642)

[II. Phạm vi nghiên cứu: 5](#_Toc86131643)

[PHẦN LÝ THUYẾT 5](#_Toc86131644)

[I. Thread là gì? Multi-thread là gì? 5](#_Toc86131645)

[II. Đa nhiệm (multitasking): 6](#_Toc86131646)

[1. Định nghĩa: 6](#_Toc86131647)

[2. Ưu điểm của đa luồng: 7](#_Toc86131648)

[3. Nhược điểm của đa luồng: 7](#_Toc86131649)

[III. Vòng đời (các trạng thái) của một Thread trong java: 8](#_Toc86131650)

[IV. Tạo luồng bằng cách extend từ lớp Thread: 9](#_Toc86131651)

[V. Môt số Phương thức thread thường sử dụng: 11](#_Toc86131652)

[1. Thực nghiệm một số phương thức thường sử dụng: 11](#_Toc86131653)

[2. Nguyên tắc hoạt động của thread: 21](#_Toc86131654)

[VI. Ví dụ minh hoạ về đa luồng: 22](#_Toc86131655)

[PHẦN THỰC HIỆN ĐỒ ÁN 24](#_Toc86131656)

[I. Giới thiệu về ứng dụng quay xổ số: 24](#_Toc86131657)

[1. Giao diện của ứng dụng: 24](#_Toc86131658)

[2. Hướng dẫn sử dụng ứng dụng: 25](#_Toc86131659)

[II. Thực hiện đồ án: 27](#_Toc86131660)

[III. KẾT LUẬN: 30](#_Toc86131661)

[IV. Tài liệu tham khảo: 30](#_Toc86131662)

PHẦN MỞ ĐẦU

1. Lý do chọn đề tài:

Hiện nay, với sự xuất hiện ngày càng nhiều các hệ thống điện tử đã làm cho lượng thông tin trong mọi lĩnh vực phát triển nhanh chóng, có cấu trúc đa dạng và phức tạp. Đặc biệt, trong lĩnh vực xử lý ngôn ngữ tự nhiên, nhận dạng, xử lý ảnh, dự báo thời tiết, v.v. đòi hỏi máy tính phải xử lý một lượng dữ liệu rất lớn, với tốc độ cao.

Nhóm chúng em chọn đề tài thread và multi- thread để thực hiện bởi vì nó có thể xử lý nhanh và hiệu quả trong các lĩnh vực.

1. Phạm vi nghiên cứu:

* Tìm hiểu về lý thuyết của thread và multi-thread
* Tiến hành tạo thread
* Thực nghiệm một số phương pháp thường sử dụng
* Thực hiện ứng dụng

PHẦN LÝ THUYẾT

1. Thread là gì? Multi-thread là gì?

Thread (luồng) về cơ bản là một tiến trình con (sub-process). Một đơn vị xử lý nhỏ nhất của máy tính có thể thực hiện một công việc riêng biệt. Trong Java, các luồng được quản lý bởi máy ảo Java (JVM).

Multi-thread (đa luồng) là một tiến trình thực hiện nhiều luồng đồng thời. Một ứng dụng Java ngoài luồng chính có thể có các luồng khác thực thi đồng thời làm ứng dụng chạy nhanh và hiệu quả hơn.

VD: Trình duyệt web hay các chương trình chơi nhạc là 1 ví dụ điển hình về đa luồng.

+ Khi duyệt 1 trang web, có rất nhiều hình ảnh, CSS, javascript… được tải đồng thời bởi các luồng khác nhau.

+ Khi play nhạc, chúng ta vẫn có thể tương tác được với nút điều khiển như: Play, pause, next, back … vì luồng phát nhạc là luồng riêng biệt với luồng tiếp nhận tương tác của người dùng.

1. Đa nhiệm (multitasking):
2. Định nghĩa:

Multitasking: Là khả năng chạy đồng thời một hoặc nhiều chương trình cùng một lúc trên một hệ điều hành. Hệ điều hành quản lý việc này và sắp xếp lịch phù hợp cho các chương trình đó. Ví dụ, trên hệ điều hành Windows chúng ta có làm việc đồng thời với các chương trình khác nhau như: Microsoft Word, Excel, Media Player, …

Chúng ta sử dụng đa nhiệm để tận dụng tính năng của CPU.

* Đa nhiệm có thể đạt được bằng hai cách:

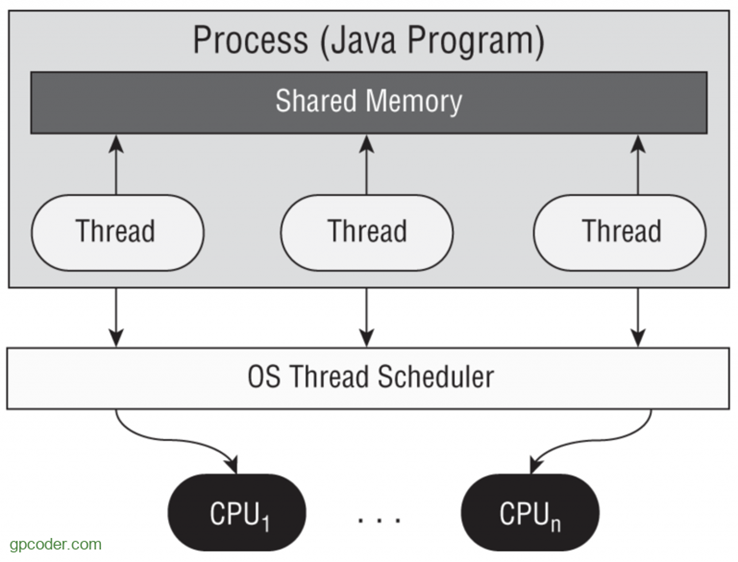
+ Đa nhiệm dựa trên đơn tiến trình (Process) – Đa tiến trình (Multiprocessing).

* Mỗi tiến trình có địa chỉ riêng trong bộ nhớ, tức là mỗi tiến trình phân bổ vùng nhớ riêng biệt.
* Tiến trình là nặng.
* Sự giao tiếp giữa các tiến trình có chi phí cao.
* Chuyển đổi từ tiến trình này sang tiến trình khác đòi hỏi thời gian để đăng ký việc lưu và tải các bản đồ bộ nhớ, các danh sách cập nhật, …

+ Đa nhiệm dựa trên luồng (Thread) – Đa luồng (MultiThreading).

* Các luồng chia sẻ không gian địa chỉ ô nhớ giống nhau.
* Luồng là nhẹ.
* Sự giao tiếp giữa các luồng có chi phí thấp.

Đa tiến trình (multiprocessing) và đa luồng (multithreading) cả hai được sử dụng để tạo ra hệ thống đa nhiệm (multitasking). Nhưng chúng ta sử dụng đa luồng nhiều hơn đa tiến trình bởi vì các luồng chia sẻ một vùng bộ nhớ chung. Chúng không phân bổ vùng bộ nhớ riêng biệt để tiết kiệm bộ nhớ, và chuyển đổi ngữ cảnh giữa các luồng mất ít thời gian hơn tiến trình



1. Ưu điểm của đa luồng:

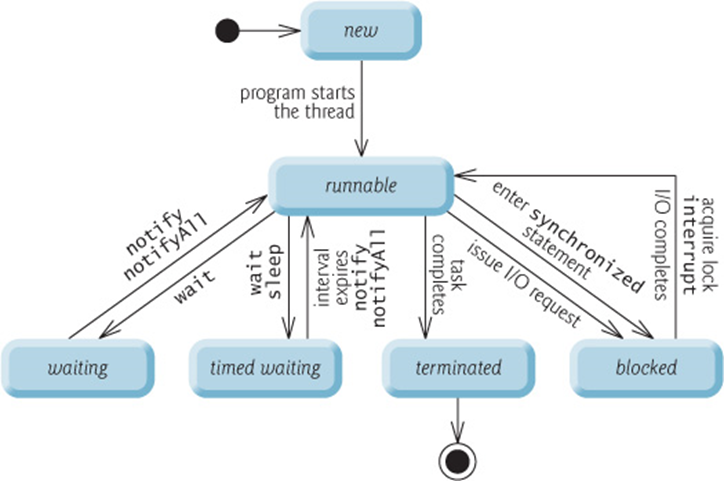
* Nó không chặn người sử dụng vì các luồng là độc lập và bạn có thể thực hiện nhiều công việc cùng một lúc.
* Mỗi luồng có thể dùng chung và chia sẻ nguồn tài nguyên trong quá trình chạy, nhưng có thể thực hiện một cách độc lập.
* Luồng là độc lập vì vậy nó không ảnh hưởng đến luồng khác nếu ngoại lệ xảy ra trong một luồng duy nhất.
* Có thể thực hiện nhiều hoạt động với nhau để tiết kiệm thời gian. Ví dụ một ứng dụng có thể được tách thành : luồng chính chạy giao diện người dùng và các luồng phụ nhiệm gửi kết quả xử lý đến luồng chính.

1. Nhược điểm của đa luồng:

* Càng nhiều luồng thì xử lý càng phức tạp.
* Xử lý vấn đề về tranh chấp bộ nhớ, đồng bộ dữ liệu khá phức tạp.
* Cần phát hiện tránh các luồng chết (dead lock), luồng chạy mà không làm gì trong ứng dụng cả.

Ghi chú: Deadlock (Khoá chết) là gì? Deadlock xảy ra khi 2 tiến trình đợi nhau hoàn thành, trước khi chạy. Kết quả của quá trình là cả 2 tiến trình không bao giờ kết thúc.

1. Vòng đời (các trạng thái) của một Thread trong java:



Vòng đời của thread trong java được kiểm soát bởi JVM. Java định nghĩa các trạng thái của luồng trong các thuộc tính static của lớp Thread.State:

• NEW : Đây là trạng thái khi luồng vừa được khởi tạo bằng phương thức khởi tạo của lớp Thread nhưng chưa được start(). Ở trạng thái này, luồng được tạo ra nhưng chưa được cấp phát tài nguyên và cũng chưa chạy. Nếu luồng đang ở trạng thái này mà ta gọi các phương thức ép buộc stop,resume,suspend … sẽ là nguyên nhân sảy ra ngoại lệ IllegalThreadStateException .

• RUNNABLE : Sau khi gọi phương thức start() thì luồng test đã được cấp phát tài nguyên và các lịch điều phối CPU cho luồng test cũng bắt đầu có hiệu lực. Ở đây, chúng ta dùng trạng thái là Runnable chứ không phải Running, vì luồng không thực sự luôn chạy mà tùy vào hệ thống mà có sự điều phối CPU khác nhau.

• WAITING : Thread chờ không giới hạn cho đến khi một luồng khác đánh thức nó.

• TIMED\_WAITING : Thread chờ trong một thời gian nhất định, hoặc là có một luồng khác đánh thức nó.

• BLOCKED: Đây là 1 dạng của trạng thái “Not Runnable”, là trạng thái khi Thread vẫn còn sống, nhưng hiện tại không được chọn để chạy. Thread chờ một monitor để unlock một đối tượng mà nó cần.

• TERMINATED : Một thread ở trong trạng thái terminated hoặc dead khi phương thức run() của nó bị thoát.

1. Tạo luồng bằng cách extend từ lớp Thread:

* Các cách tạo luồng trong Java:

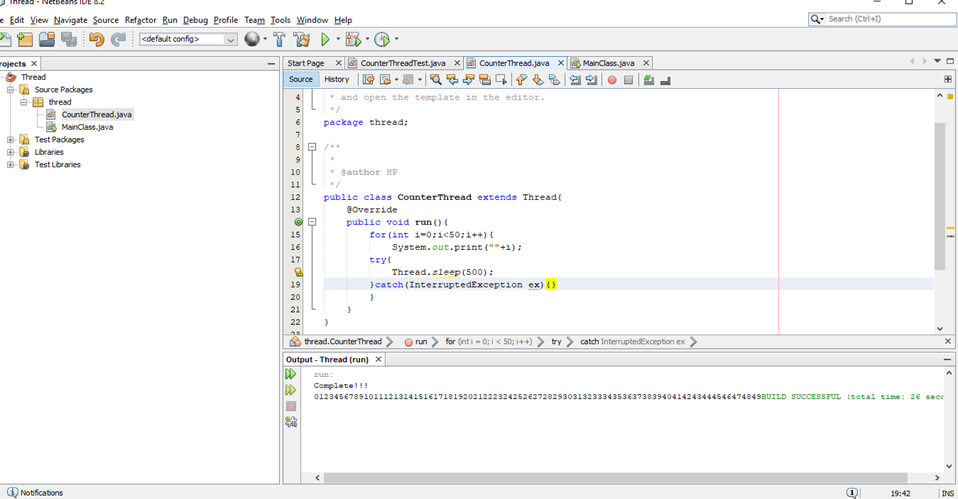
+ Cách 1.Tạo luồng bằng cách extend từ lớp Thread

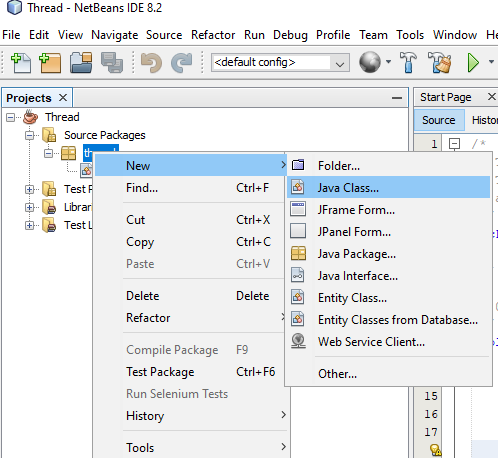
+ Cách 2.Tạo luồng bằng cách implement từ Interface Runnable

* Do đề tài đồ án của chúng em làm về Thread nên nhóm em sẽ thực hiện Cách 1.
* Để tạo luồng bằng cách tạo lớp kế thừa từ lớp Thread, ta phải làm các công việc sau :

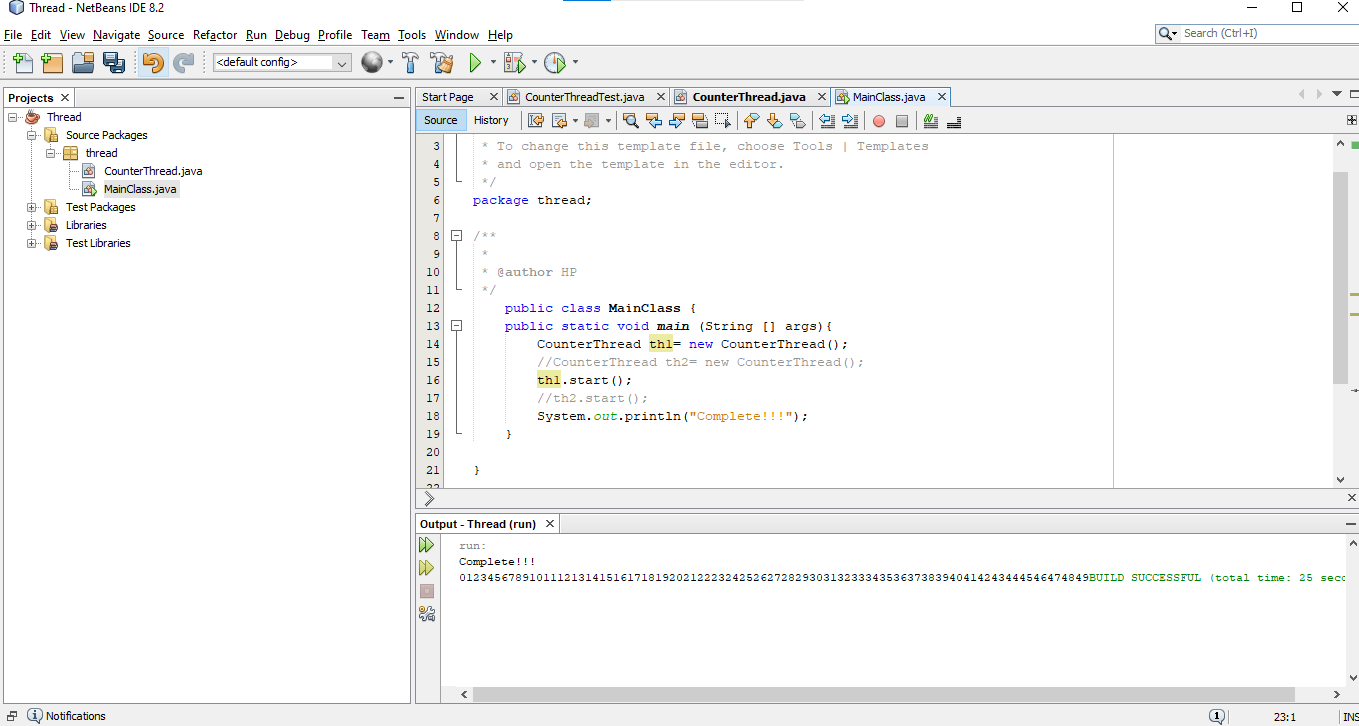
Bước 1: Khai báo 1 lớp mới kế thừa từ lớp Thread.

Bước 2: Override lại phương thức run ở lớp này, những gì trong phương thức run sẽ được thực thi khi luồng bắt đầu chạy. Sau khi luồng chạy xong tất cả các câu lệnh trong phương thức run thì luồng cũng tự hủy.

Bước 3 : Tạo 1 thể hiện (hay 1 đối tượng) của lớp ta vừa khai báo.



Bước 4: Sau đó gọi phương thức start() của đối tượng này để bắt đầu thực thi luồng.



**Lưu ý :**

• Tuy ta khai báo những công việc cần làm của luồng trong phương thức run() nhưng khi thực thi luồng ta phải gọi phương thức start(). Vì đây là phương thức đặc biệt mà java xây dựng sẵn trong lớp Thread, phương thức này sẽ cấp phát tài nguyên cho luồng mới rồi chạy phương thức run() ở luồng này. Vì vậy, nếu ta gọi phương thức run() mà không gọi start() thì cũng tương đương với việc gọi 1 phương thức của 1 đối tượng bình thường và phương thức vẫn chạy trên luồng mà gọi phương thức chứ không chạy ở luồng mới tạo ra, nên vẫn chỉ có 1 luồng chính làm việc chứ ứng dụng vẫn không phải là đa luồng.

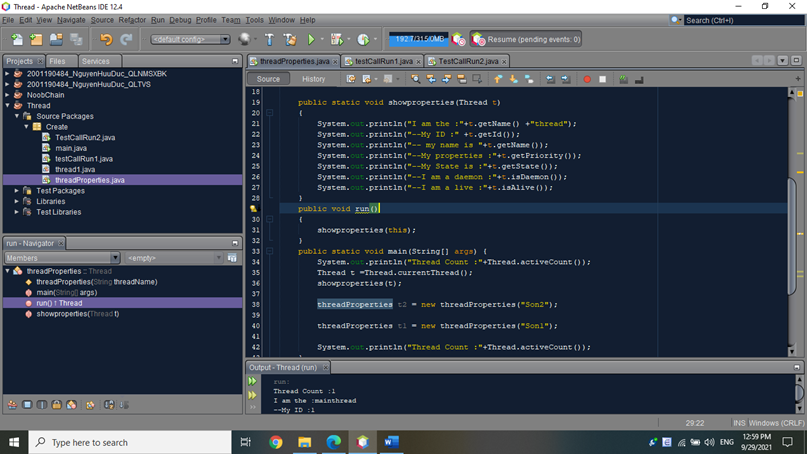
• Sau khi start một thread, nó không bao giờ có thể được start lại. Nếu bạn làm như vậy ngoại lệ IllegalThreadStateException sẽ xảy ra.

1. Môt số Phương thức thread thường sử dụng:

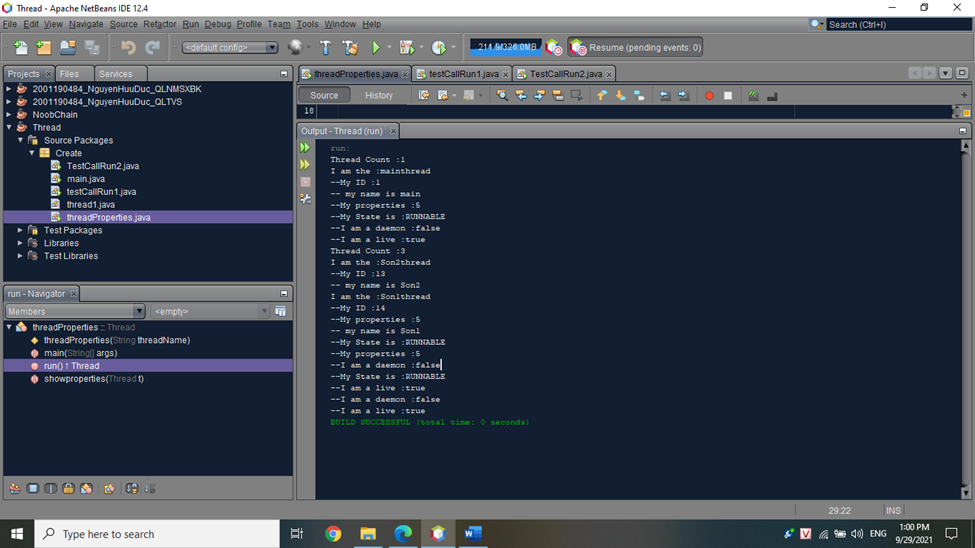


1. Thực nghiệm một số phương thức thường sử dụng:
2. Chạy một thread:

* Thực hiện:



* Kết quả:



* Giải thích:

GetName : Lấy tên của thread

getID : Lấy mã ID của thread

getpriority : Lấy thứ tự ưu tiên của thread

getStates : Lấy trạng thái của thread(đang chạy ,đang chờ ,...vv)

isAlive : Kiểm tra thread có đang sống không nếu không trả về false

isDaemon : Kiểm tra luồng hiện tại có phải là luồng hiểm hay không

giải thích về Luồng hiểm :

Luồng hiểm trong java (thread daemon) là một luồng cung cấp dịch vụ cho luồng người dùng. Thời gian sống của nó phụ thuộc vào các luồng người dùng. Tức là khi tất cả các luồng người dùng chết, JVM sẽ tự động chấm dứt luồng này.

Có nhiều luồng hiểm java chạy tự động, ví dụ: Gc, finalizer, v.v.

Các điểm cần nhớ về luồng hiểm trong java

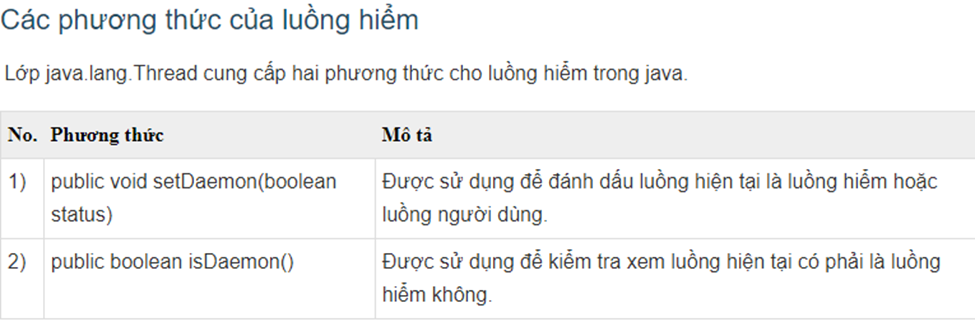
• Nó cung cấp dịch vụ cho các luồng người dùng.

• Thời gian sống của nó phụ thuộc vào các luồng người dùng.

• Nó là một luồng có mức ưu tiên thấp.

Tại sao JVM kết thúc luồng hiểm nếu không có luồng người dùng?

Mục đích duy nhất của luồng hiểm là nó cung cấp dịch vụ cho luồng người dùng cho các tác vụ nền background supporting task (background supporting task). Nếu không có luồng người dùng, tại sao JVM vấn tiếp tục chạy luồng này? Đó là lý do tại sao JVM kết thúc luồng hiểm nếu không có thread người dùng nào đang được thực thi.



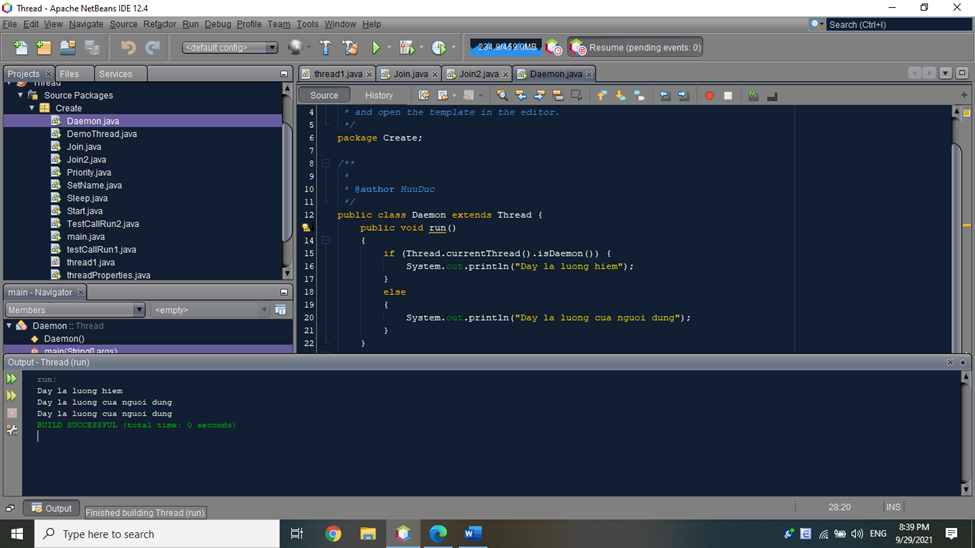
1. Demo từng phương thức :Chủ yếu các method quan trọng:

* Method Daemon:

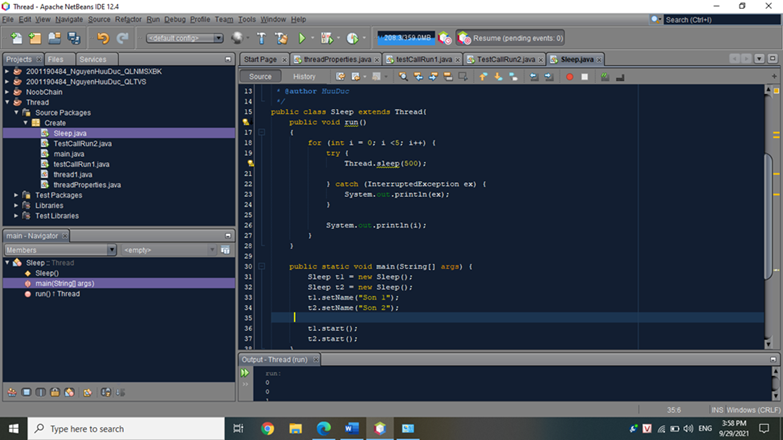
A screenshot of a computer

Description automatically generated

* Kết quả:



* Sleep: Cho ngủ 500ms = ½ giây:
* Kết quả:

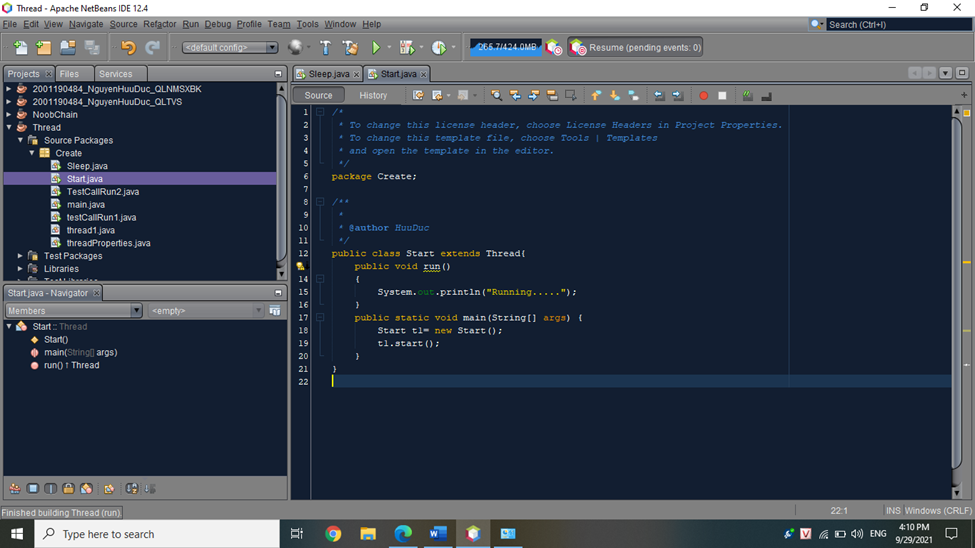


* 2 thread xen kẽ nhau thực hiện khoảng cách ½ giây .Xen kẽ theo cơ chế Time Slicing mỗi thread thực hiện được JVM phân chia tuần tự

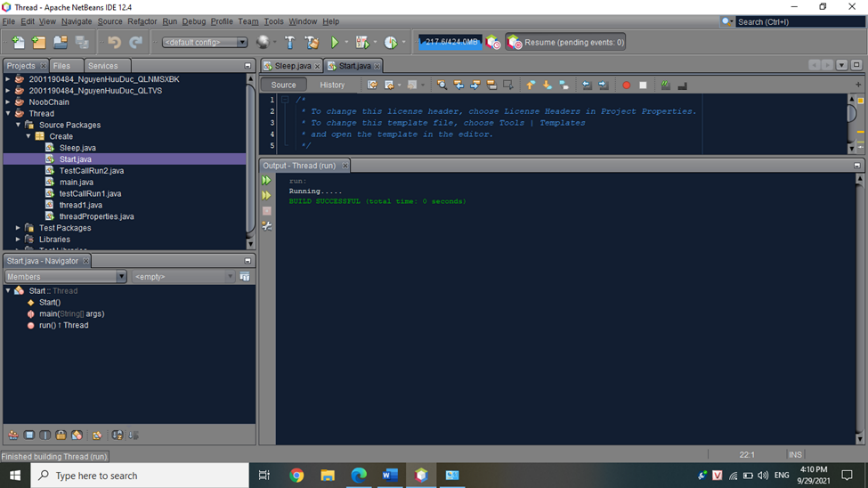
A screenshot of a computer

Description automatically generated with medium confidence

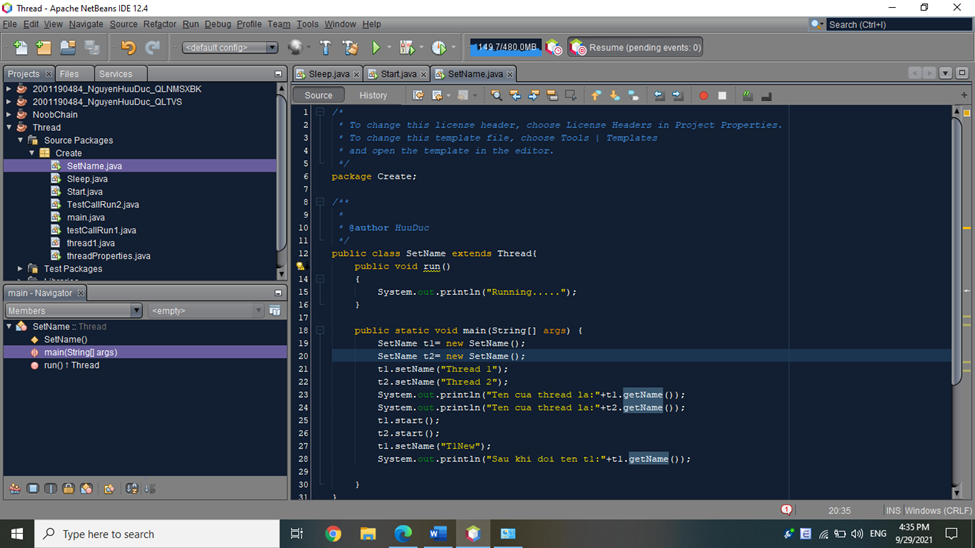
* Start



* Kết quả :



* Lưu ý : Không thể Start cùng 1 thread 2 lần Sẽ xảy ra lỗi sau
* Exception in thread "main" java.lang.IllegalThreadStateException SetName cho thread:



* Kết quả :

A screenshot of a computer

Description automatically generated

* Priority
* Trường hợp chưa set priority:

A screenshot of a computer

Description automatically generated

* Khởi chạy

A screenshot of a computer

Description automatically generated

* Trường hợp Setpriority

A screenshot of a computer

Description automatically generated

* Khởi chạy

A screenshot of a computer

Description automatically generated

* Method tiếp theo : Run:

\*Lưu ý cần phân biệt giữa thread.run và thread.start()

Đối với thread.run khi thực thi chỉ chạy duy nhất phần hàm run một cách độc lập không có luồng mới nào được tạo

Ngược lại đối với start một luồng new được tạo và mã bên trong run() được thực thi trong new thread

* Demo Run:

A screenshot of a computer

Description automatically generated

* Kết quả :

A screenshot of a computer

Description automatically generated

* Method Join : Phương thức join() chờ một thread chết. Nói cách khác, nó làm cho các thread đang chạy ngừng hoạt động cho đến khi luồng mà nó tham gia hoàn thành nhiệm vụ của nó.
* Demo

A screenshot of a computer

Description automatically generated

* Result

A screenshot of a computer

Description automatically generated

Khi t1 chạy xong mới thực thi t2 và t3

* Khi dùng join(miliSeconds)

A screenshot of a computer

Description automatically generated

* Kết quả :

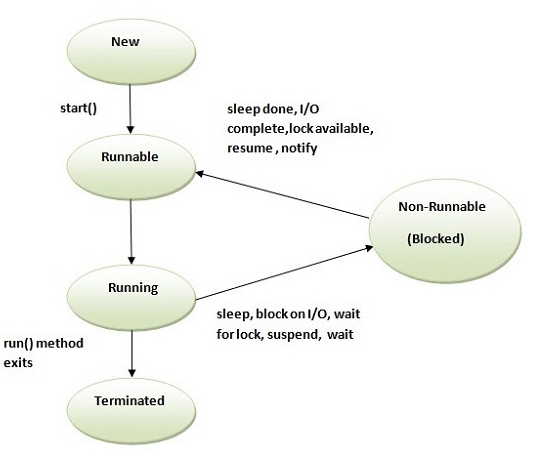
A screenshot of a computer

Description automatically generated

Kết luận join khi có thời gian sẽ tiếp tục được vào khoảng thời gian đã cài đặt trước xác định

Method Wait, Notify, NotifyAll không thuộc class Thread Sẽ không được đề cập trong này.

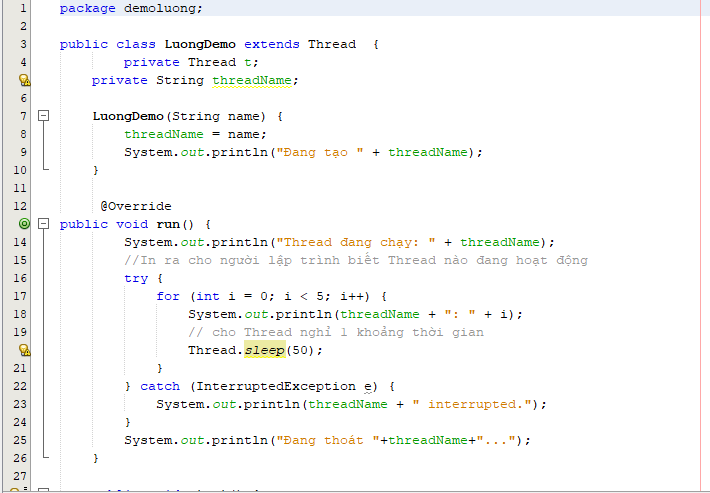
1. Nguyên tắc hoạt động của thread:

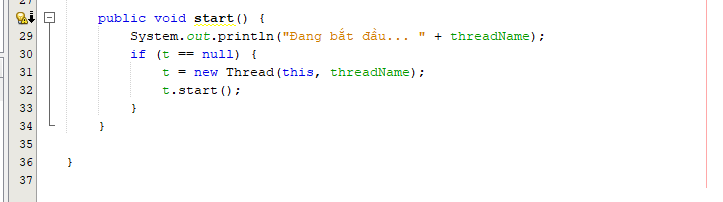


* Giải thích:
* New: Thread ở trạng thái new nếu bạn tạo một thể hiện của lớp Thread nhưng trước khi gọi phương thức start().
* Runnable: Thread ở trạng thái runnable sau khi gọi phương thức start(), nhưng trình lên lịch (scheduler) của thread đã không chọn nó là thread đang chạy.
* Running**:** Thread ở trạng thái running nếu trình lên lịch của thread đã chọn nó.
* Non-Runnable (Blocked)**:** Đây là trạng thái khi thread vẫn còn sống, nhưng hiện tại không được chọn để chạy.
* Terminated**:** Một thread ở trong trạng thái terminated hoặc dead khi phương thức run() của nó bị thoát.

1. Ví dụ minh hoạ về đa luồng:
2. Ta cho 2 Thread thực hiện cùng lúc việc đếm số nguyên từ 0 đến 5 bằng cách sử dụng vòng lặp For từ đó có thể hiểu thêm được cách vận hành của Thread

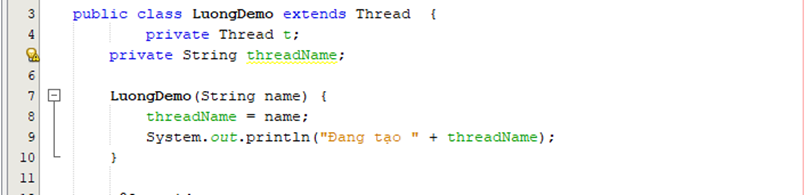
Bước 1: Thực hiện bằng cách tạo 1 class LuongDemo mới extend từ lớp Thread





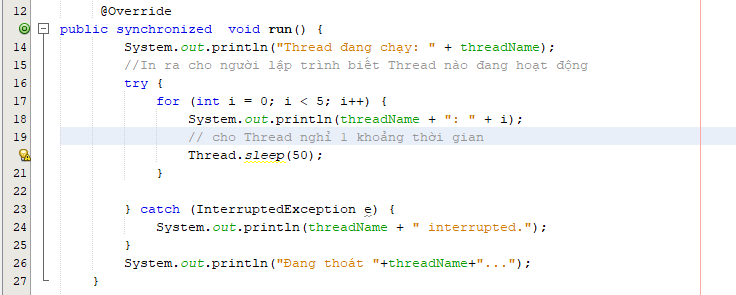
Bước 2: Ta khai báo 1 lớp LuongDemo kế thừa từ lớp Thread

Sau đó ta viết các câu lệnh khai báo và các lệnh thực hiện việc tạo các Thread mới

  
Bước 3: Sử dụng một trong hai phương thức

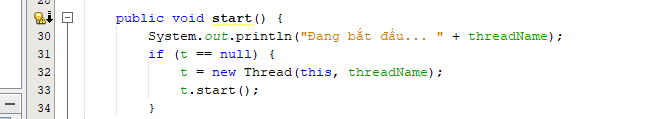
* Phương thức run()

T override lại phương thức run ở lớp này, những gì trong phương thức run() sẽ được thực thi khi luồng bắt đầu chạy. Sau khi luồng chạy xong tất cả các câu lệnh trong phương thức run thì luồng cũng tự hủy.



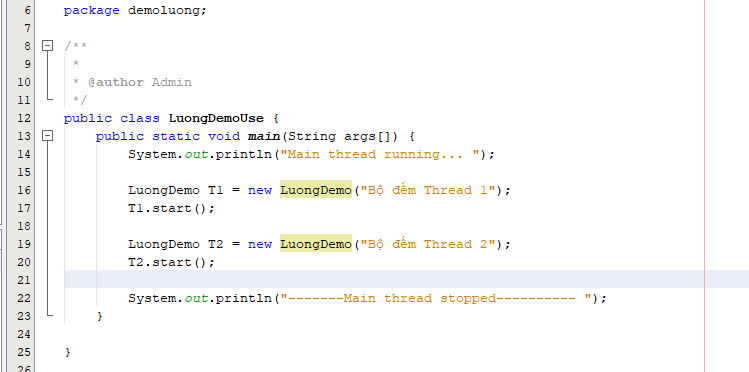
* Phương thức start()

Sẽ được gọi chạy ở 1 class có chứa hàm Main:

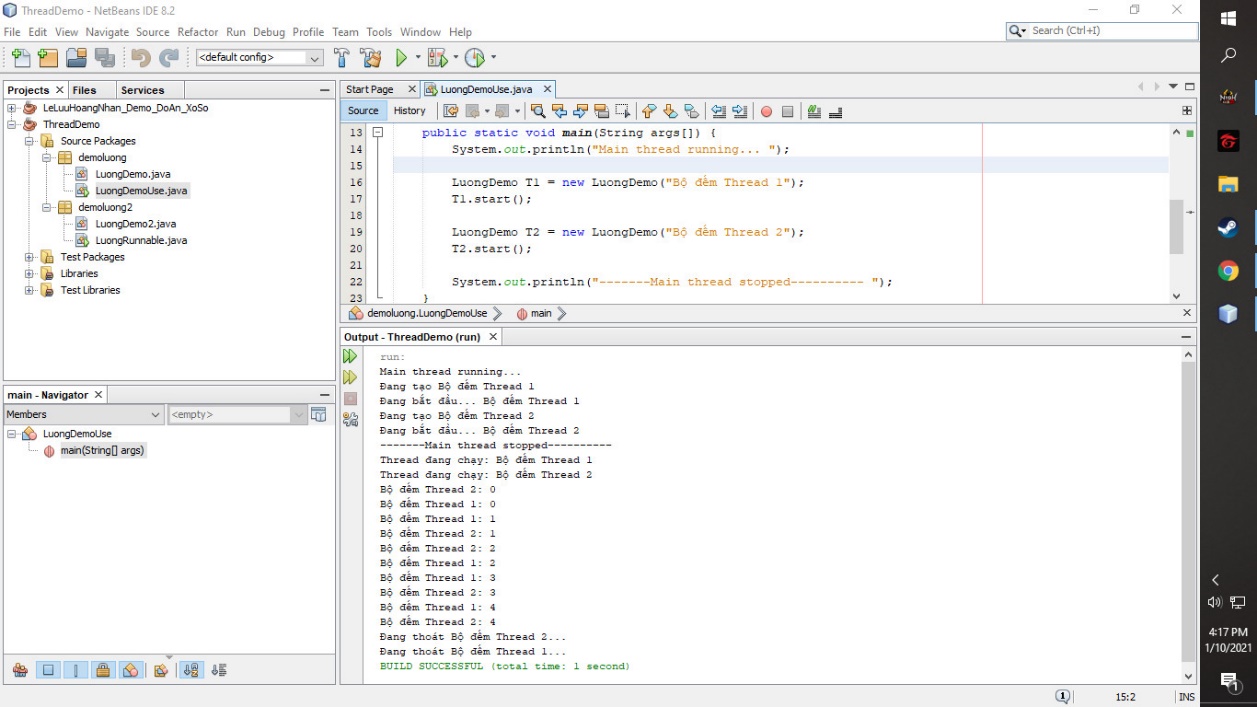


Bước 4: Tạo class LuongDemoUse

Tạo thêm 1 lớp có chứa hàm Main để gọi chạy phương thức start() có trong class LuongDemo:

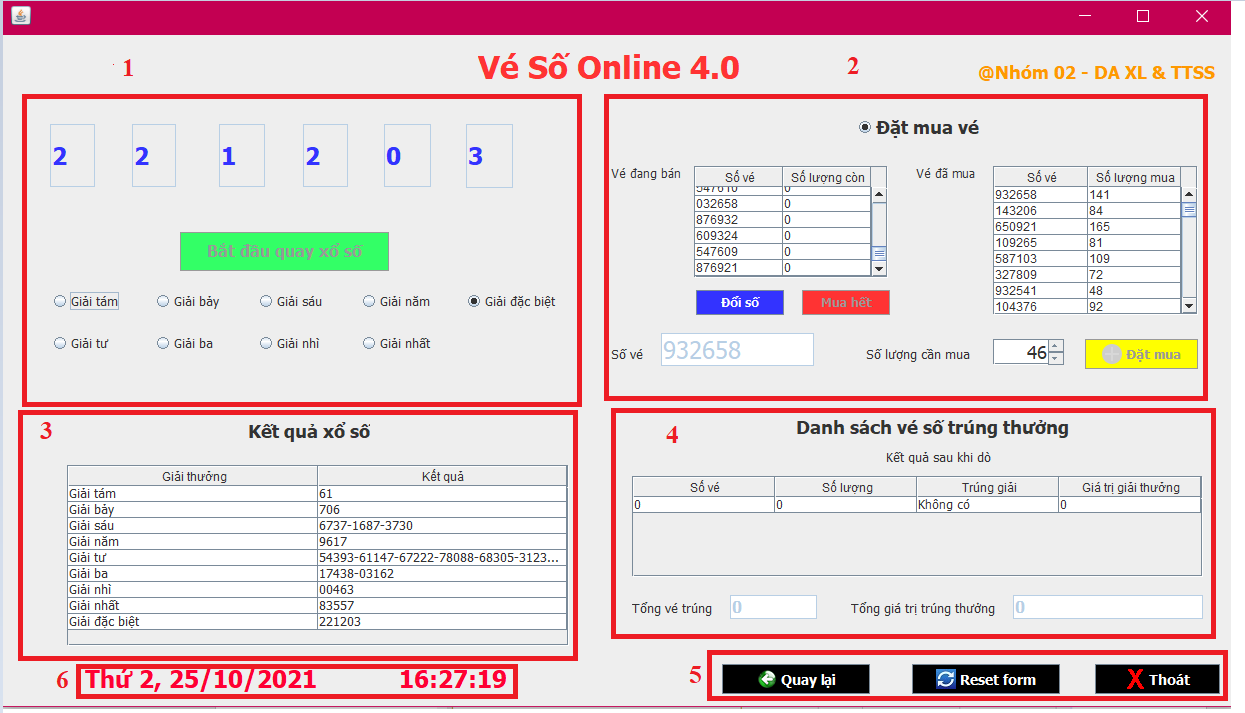


1. Kết quả của ví dụ:



PHẦN THỰC HIỆN ĐỒ ÁN

1. Giới thiệu về ứng dụng quay xổ số:
2. Giao diện của ứng dụng:



Gồm:

1:Khu vực quay vé

2:Khu vực đặt vé, mua vé

3:Kết quả xổ số

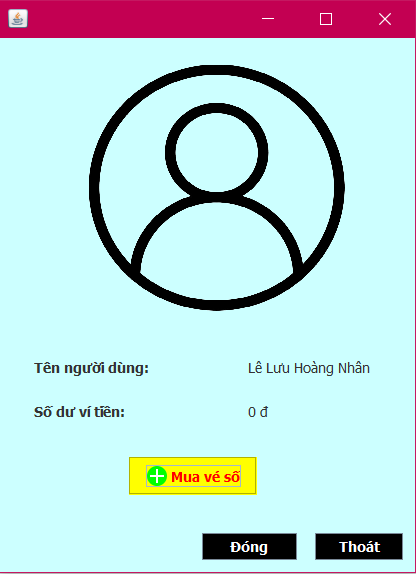
4:Danh sách trúng thưởng

5:Các nút thao tác

6:Thời gian

1. Hướng dẫn sử dụng ứng dụng:

Bước 1:Click vào “Mua vé số” để bắt đầu vào form vé số.



Bước 2: Ta chọn tick tròn của ”Đặt mua vé” để mở khóa khu vực mua vé; sau đó chọn vé cần mua và điều chỉnh số lượng cần mua. Click vào “Đặt mua” để mua vé.

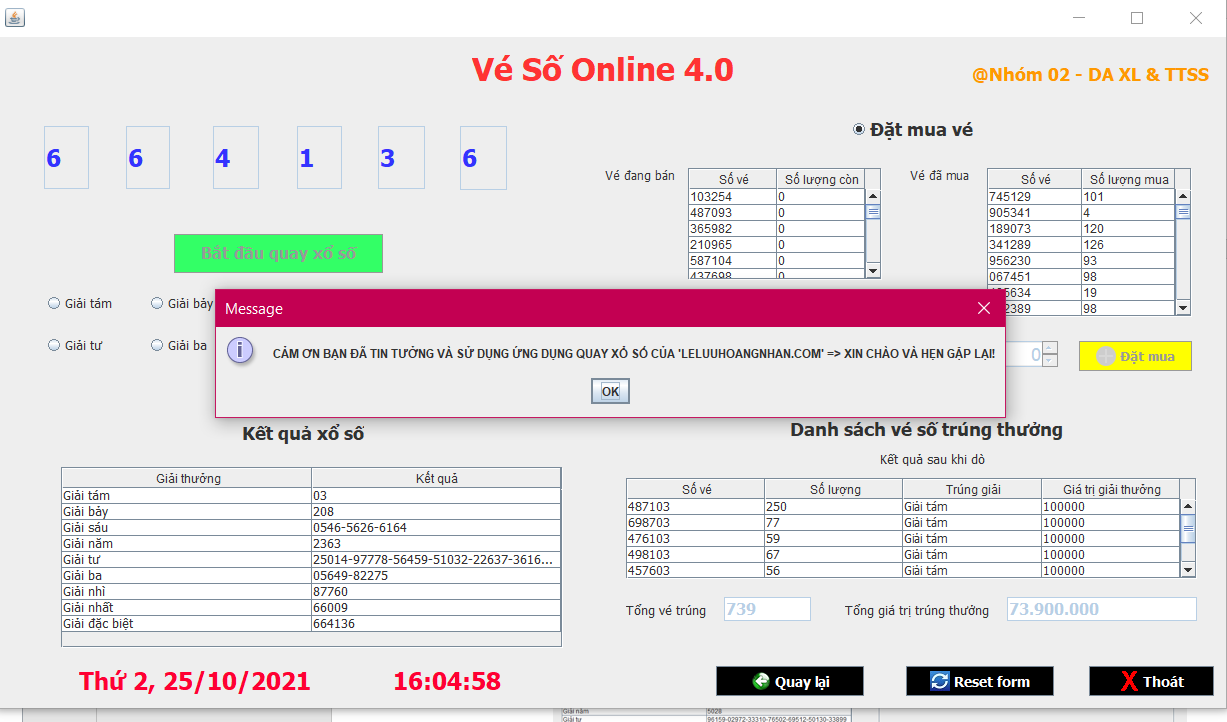
Lưu ý: “Đổi số” là đổi các số mới trong danh sách đang bán.

“Mua hết” là mua hết số vế đang bán.

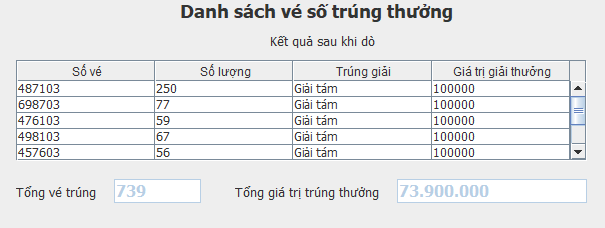


Bước 3: Sau khi đặt mua hoàn tất,ta bấm “ Bắt đầu quay xổ số”để quay số hoặc chờ đến 16 giờ thì bắt đầu quay số. Sau đó chờ đợi đến khi quay xong số.

Thông báo sau khi đã tiến hành quay số xong



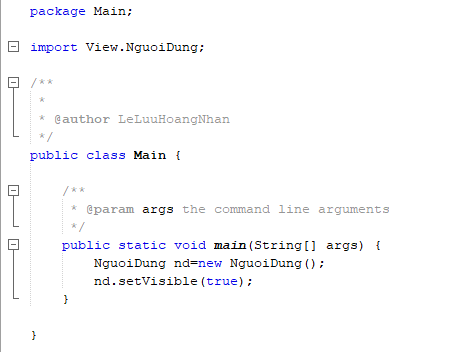
Danh sách trúng thưởng sau khi quay



1. Thực hiện đồ án:

Ứng đụng được thiết kế bằng 3 class và 2 form.

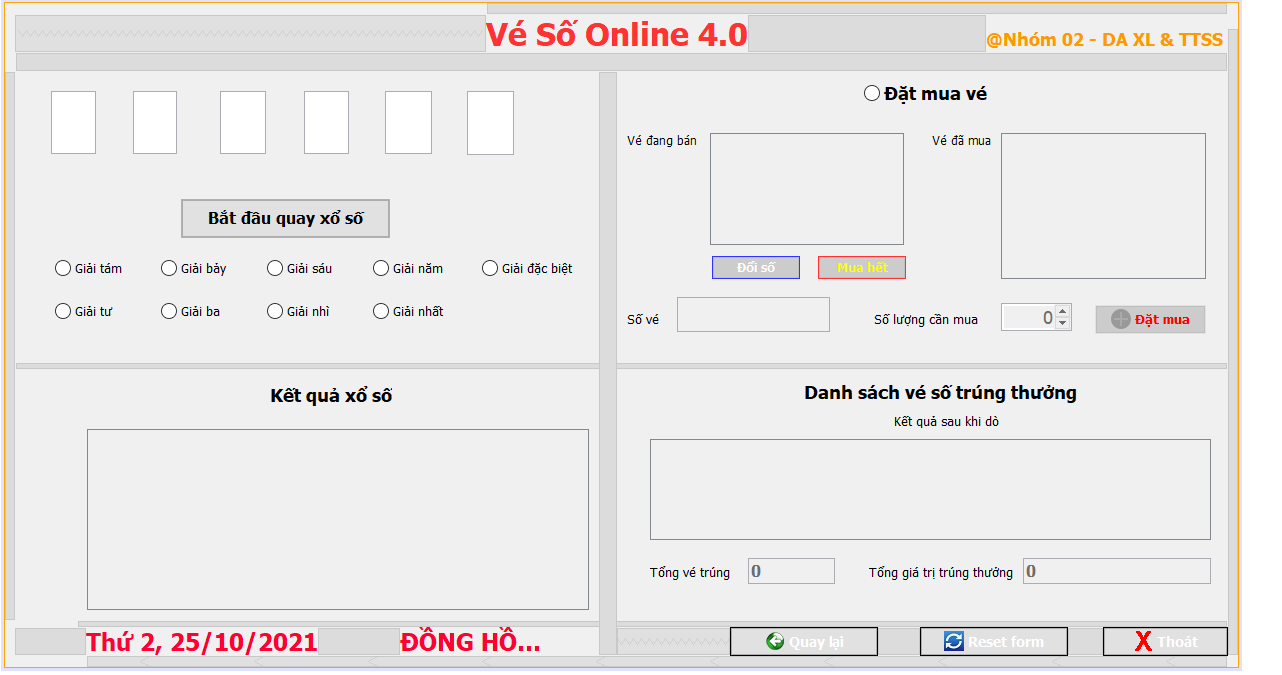
* Trong đó class Main có chứa hàm main sẽ thực hiện nhiệm vụ gọi chạy Form đã thiết kế



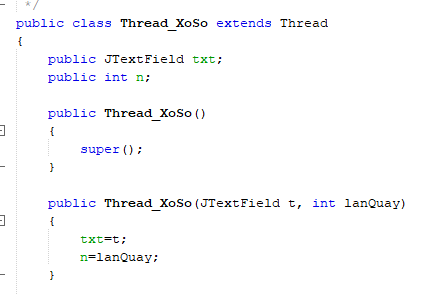
* Form người dung được thiết kế



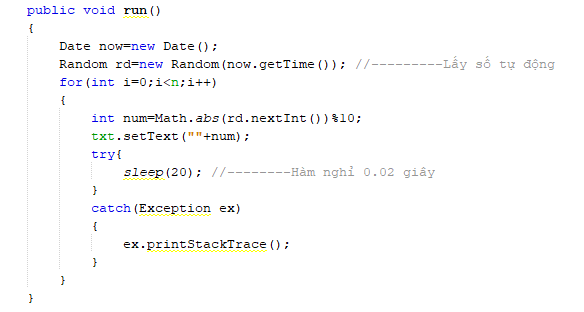
* Form xổ số được thiết kế



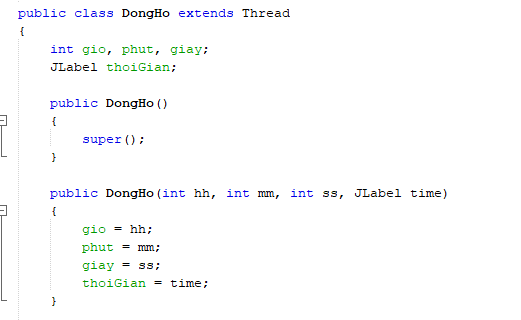
* Class Thread\_XoSo được kế thừa từ lớp Thread



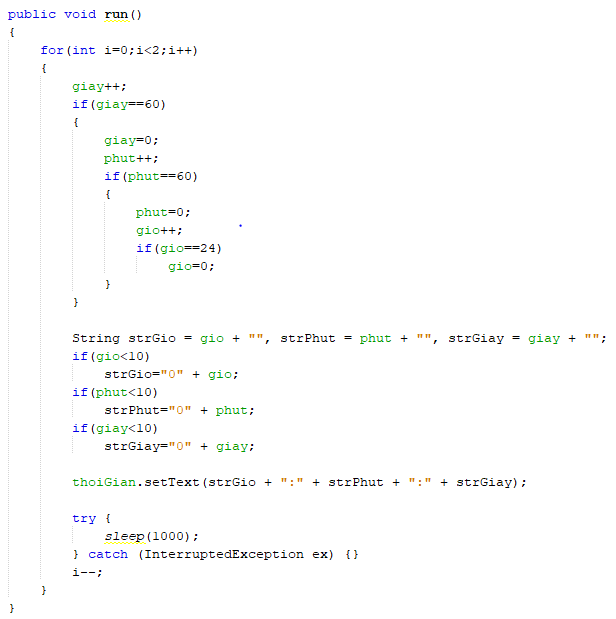
* Phương thức run() (Thread\_XoSo)



* Class DongHo được kế thừa từ lớp Thread



* Phương thức run() (DongHo)



1. KẾT LUẬN:

Đánh giá kết quả xử lý tuần tự và song song:

- Xử lý tuần tự: Tốc độ xử lý sẽ chậm hơn, tốn nhiều tgian chờ đợi.

- Xử lý rsong song: Tốc độ xử lý sẽ nhanh hơn, đỡ mất nhiều tgian chờ đợi.

Demo ứng dụng vé số online 4.0: Nếu như áp dụng xử lý tuần tự thì phải chờ đợi vòng quay số 1 quay xong rồi mới tới vòng quay thứ 2 và chờ đợi lặp lại tiếp tục cho tới vòng quay thứ 6. Như vậy sẽ tốn khá nhiều thời gian.

- Nếu cũng bài toán đó, nhưng áp dụng xử lý song song thì các vòng quay sẽ được thực hiện đồng thời. Giúp tốc độ bài toán sẽ nhanh hơn gấp n lần và không mất quá nhiều tgian để chờ đợi.

1. Tài liệu tham khảo:

https://gpcoder.com/3484-lap-trinh-da-luong-trong-java-java-multi-threading/

https://www.eginnovations.com/blog/java-threads/

https://www.w3schools.com/java/java\_threads.asp

https://www.geeksforgeeks.org/java-program-to-create-a-thread/

https://www.javatpoint.com/how-to-create-a-thread-in-java

https://viblo.asia/p/multithreading-cac-cach-khoi-tao-va-su-dung-java-thread-5y8Rr7n0Mob3

https://en.wikipedia.org/wiki/Multithreading\_(computer\_architecture)

**PHỤ LỤC**

1. **Bảng phân công:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **HỌ VÀ TÊN** | **CÔNG VIỆC** | **CHỨC VỤ** |
| 1 | 2001190186 | Lê Lưu Hoàng Nhân | Thiết kế ứng dụng vé số online 4.0 + Báo cáo demo ứng dụng. | Nhóm trưởng |
| 2 | 2001190484 | Nguyễn Hữu Đức | -Nguyên tắc hoạt động Thread.  -Một số phương thức thường sử dụng của Thread. | Thành viên |
| 3 | 2001181105 | Nguyễn Văn Hiền | -Định nghĩa Thread, Multi-thread.  -Thiết kế Word và Powerpoint. | Thành viên |
| 4 | 2001190529 | Dương Trung Hiếu | -Các cách tạo ra luồng trong java.  -Tạo luồng bằng cách extend từ lớp Thread. | Thành viên |
| 5 | 2001190832 | Nguyễn Duy Thịnh | Ví dụ minh họa về cách dùng Thread và Multi-thread + Báo cáo phần slide. | Thành viên |

1. **Đánh giá của nhóm trưởng:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nhân (NT) | Hiền | Đức | Hiếu | Thịnh |
| Tốt  (Các thành viên đánh giá) | Tốt | Khá | Tốt | Tốt |