ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN NGUYÊN LÝ HỆ ĐIỀU HÀNH

Đề tài:

Đa luồng giải quyết vấn đề đồng bộ hóa

GVHD : Mai Văn Hà
SVTH : Trần Hoàng Minh
Lớp : 11T1

Nội dung báo cáo

I. Mở đầu
II. Cơ sở lý thuyết
III. Phân tích thiết kế hệ thống
IV. Triển khai và đánh giá kết quả
V. Kết luận và hướng phát triển

I. MỞ ĐẦU

- Trong khuôn khổ Đồ Án môn học Nguyên Lý hệ điều hành, dưới sự hướng dẫn của thầy Mai Văn Hà.

- Đề Tài em được phân công là: **Đa luồng giải quyết vấn đề đồng bộ hóa.**

II. CƠ SỞ LÝ THUYẾT

- Luồng (Thread)
- Tạo quản lý luồng trong java
- Các trạng thái và phương thức trong luồng
- Đa luồng và mức ưu tiên
- Đồng bộ hóa
- Deadlock

II. CƠ SỞ LÝ THUYẾT

Luồng (thread)

- Tiến trình : là một chương trình chạy trên hệ điều hành và được quản lý thông qua các thẻ
- Một tiến trình có thể bao gồm nhiều luồng.
- Các luồng của một tiến trình có thể chia sẻ với nhau về không gian địa chỉ chương trình, các đoạn dữ liệu và môi trường xử lý, đồng thời cũng có vùng dữ liệu riêng để thao tác.

II. CƠ SỞ LÝ THUYẾT

Tạo quản lý luồng trong java

- Khi chương trình Java thực thi hàm main() tức là luồng main được thực thi. Tuyến này được tạo ra một cách tự động, tại đây :

  - Các luồng con sẽ được tạo ra từ đó

  - Nó là luồng cuối cùng kết thúc việc thực thi. Ngay khi luồng main() ngừng thực thi, chương trình bị chấm dứt .

II. CƠ SỞ LÝ THUYẾT

Tạo quản lý luồng trong java

- Phân chia thời gian giữa các luồng

  - CPU thực thi chỉ một luồng tại một thời điểm nhất định.

  - Các luồng có độ ưu tiên bằng nhau thì được phân chia thời gian sử dụng bộ vi xử lý.

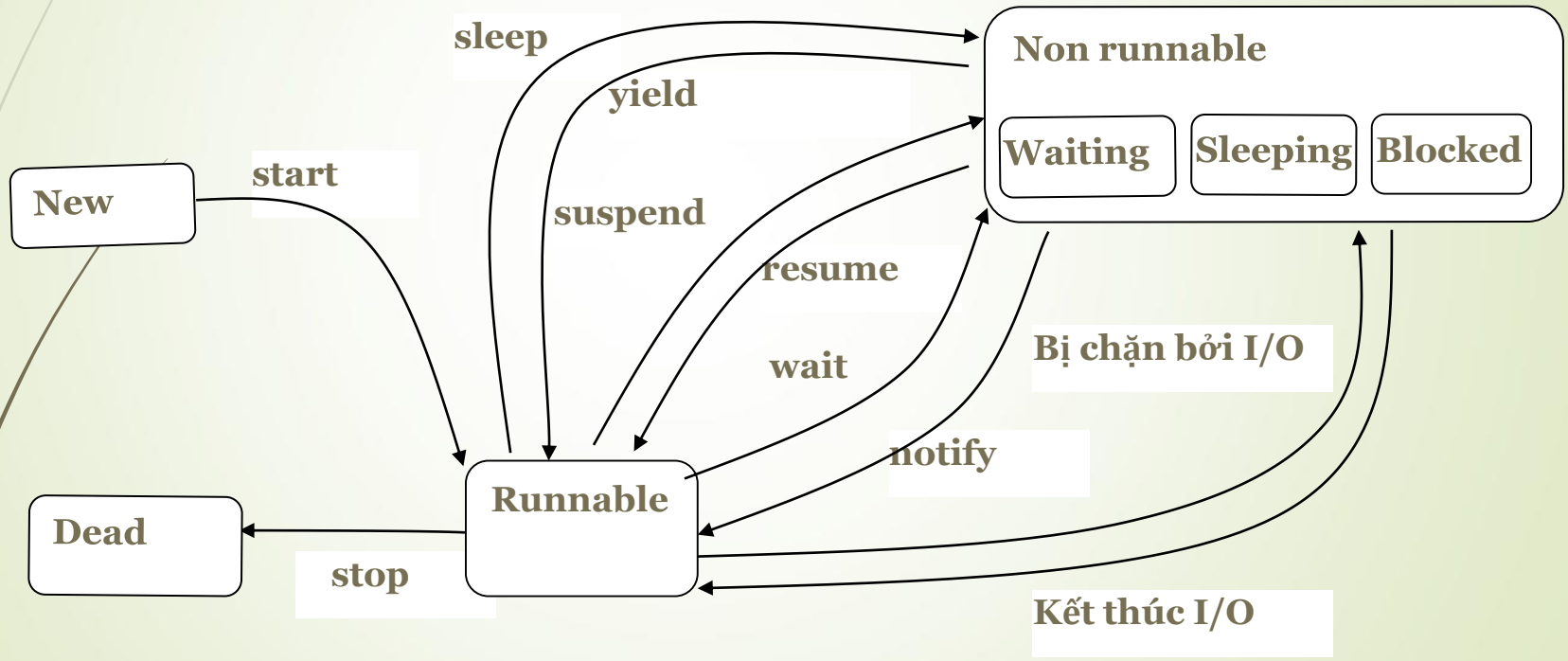
- Java cung cấp hai giải pháp tạo luồng:

  - Thiết lập lớp con của Thread

  - Cài đặt lớp xử lý luồng từ giao diện Runnable

II. CƠ SỞ LÝ THUYẾT

Các trạng thái và phương thức trong luồng



II. CƠ SỞ LÝ THUYẾT

Các trạng thái và phương thức trong luồng

New: Khi một luồng mới được tạo ra với toán tử new() và sẵn sàng hoạt động.

Runnable: Trạng thái mà luồng đang chiếm CPU để thực hiện, khi bắt đầu thì nó gọi hàm start(). Bộ lập lịch phân luồng của hệ điều hành sẽ quyết định luồng nào sẽ được chuyển về trạng thái Runnable và hoạt động. Cũng cần lưu ý rằng ở một thời điểm, một luồng ở trạng thái Runnable có thể hoặc không thể thực hiện.

Non runnable (blocked): Từ trạng thái runnable chuyển sang trạng thái ngừng thực hiện ("bị chặn") khi gọi một trong các hàm: sleep(), suspend(), wait(), *hay bị chặn lại ở* Input/output.

II. CƠ SỞ LÝ THUYẾT

Các trạng thái và phương thức trong luồng

Waiting: khi ở trạng thái Runnable, một luồng thực hiện hàm wait() thì nó sẽ chuyển sang trạng thái chờ đợi (Waiting).

Sleeping: khi ở trạng thái Runnable, một luồng thực hiện hàm sleep() thì nó sẽ chuyển sang trạng thái ngủ (Sleeping).

Blocked: khi ở trạng thái Runnable, một luồng bị chặn lại bởi những yêu cầu về tài nguyên, như yêu cầu vào/ra (*I/O*), thì nó sẽ chuyển sang trạng bị chặn (Blocked)

Dead: Luồng chuyển sang trạng thái "chết" khi nó kết thúc hoạt động bình thường, có thể gọi hàm stop() để kết thúc ("giết chết") một luồng.

II. CƠ SỞ LÝ THUYẾT

Đa luồng và mức ưu tiên

- Đa luồng (Multithread)
  - Một ứng dụng có thể bao hàm nhiều luồng. Mỗi luồng được đăng kí một công việc riêng biệt, mà chúng được thực thi đồng thời với các luồng khác.
  - Đa luồng giữ thời gian nhàn rỗi của hệ thống thành nhỏ nhất. Điều này cho phép bạn viết chương trình có hiệu quả cao với sự tận dụng CPU là tối đa.

II. CƠ SỞ LÝ THUYẾT

Đa luồng và mức ưu tiên

- Mức ưu tiên giữa các luồng

- Trong Java, mỗi luồng có một mức ưu tiên thực hiện nhất định.
- Khi chương trình chính thực hiện sẽ tạo ra luồng chính, luồng cha. Luồng này sẽ tạo ra các luồng con, và cứ thế tiếp tục
- Mức ưu tiên của các luồng có thể đặt lại trong khoảng từ MIN_PRIORITY và MAX_PRIORITY (mặc định là 1 – 10), hoặc NORM_PRIORITY (mặc định là 5)

II. CƠ SỞ LÝ THUYẾT

Đồng bộ hóa

- Các luồng chia sẻ với nhau cùng một không gian bộ nhớ, nghĩa là chúng có thể chia sẻ với nhau các tài nguyên.
- Để cho các luồng chia sẻ với nhau được các tài nguyên và hoạt động hiệu quả, luôn đảm bảo nhất quán dữ liệu thì phải có cơ chế đồng bộ chúng.
- Mấu chốt của sự đồng bộ là khái niệm "monitor" (giám sát) hay còn gọi là "semaphore" (cờ hiệu)
- Hàm của một lớp chỉ cho phép một luồng được thực hiện ở một thời điểm thì nó phải khai báo synchronized, được gọi là *hàm đồng bộ*.

II. CƠ SỞ LÝ THUYẾT

Deadlock

- Cơ chế đồng bộ trong Java là rất tiện lợi, khá mạnh, nhưng không giải quyết được mọi vấn đề nảy sinh trong quá trình xử lý đa luồng
- Một "deadlock" xảy ra khi hai tuyến có một phụ thuộc vòng quanh trên một cặp đối tượng đồng bộ

III. PHÂN TÍCH HỆ THỐNG

- **Phân tích yêu cầu**
- **Phân tích chức năng**

III. PHÂN TÍCH HỆ THỐNG

Phân tích yêu cầu

✓ Bài toán đặt ra

Bài toán hệ thống giao dịch tiền mặt ATM (Automated teller machine). Xử lý trường hợp hai máy ATM rút tiền cùng lúc cùng một tài khoản.

✓ Yêu cầu

- Xây dựng một tài nguyên để các luồng sử dụng chung, thiết lập cơ sở dữ liệu.

- Xử lý việc đồng bộ để đa luồng – nhiều máy ATM giao dịch cùng môt lúc (rút tiền, chuyển tiền, đổi mật khẩu).

III. PHÂN TÍCH HỆ THỐNG

Phân tích chức năng

- ✓ Chức năng rút tiền
- ✓ Chức năng chuyển tiền
- ✓ Chức năng đổi mật khẩu

III. PHÂN TÍCH HỆ THỐNG

Phân tích chức năng

✓ Các hàm chính trong chương trình:

STT	Tên khai báo	Chức năng
1	show(String mapin)	Hàm trả về đố tượng giao dịch
2	update(String mapin, int tien)	Hàm cập nhật tài nguyên
3	update(String mapin, String pass)	Hàm cập nhật tài nguyên
4	check(String mapin)	Hàm kiểm tra giá trị trong tài nguyên
5	check(String mapin, String pass)	Chứa luồng đang thực hiện
6	ruttien(ATM atm)	Hàm tác động tới tài nguyên
7	chuyentien(ATM atm)	Hàm tác động tới tài nguyên
8	doimk(ATM atm)	Hàm tác động tới tài nguyên
9	Xuly(String congviec, ATM atm)	Hàm được đồng bộ hóa
10	KetQua()	Hàm trả về kết quả giao dịch

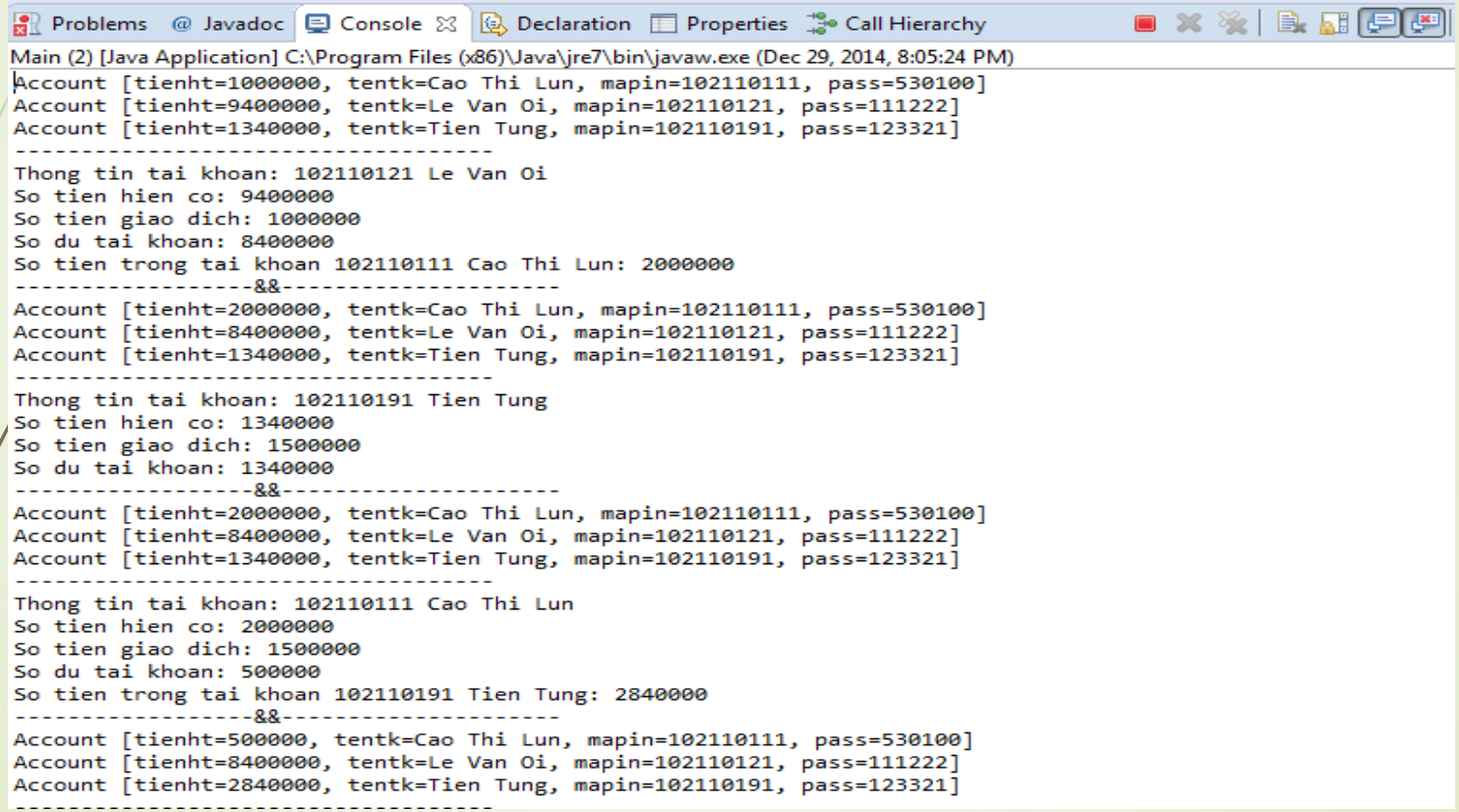
III. PHÂN TÍCH HỆ THỐNG

Phân tích chức năng

- ✓ Hàm xử lý đồng bộ:

```
public synchronized void Xuly(String congviec, ATM atm){
        if (ruttien) ruttien();
       if (chuyentien) chuyentien();
       if (doimk) doimk();
}
```

IV. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ



IV. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

Đánh giá và nhận xét

- Chương trình cung cấp đầy đủ các thao tác cũng như chức năng mà đề tài đã yêu cầu, mô phỏng đúng chức năng giao dịch của máy ATM.
- Bên cạnh đó thể hiện được vấn đề đa luồng đồng bộ.
- Vì chỉ là chương trình mô phỏng nên còn nhiều thiếu sót.

V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

- **Những kết quả đạt được**

Chương trình mô phỏng hoạt động ổn định, đáp ứng được chức năng mô phỏng lại máy ATM thể hiện đồng bộ hóa đa luồng một cách trực quan và dễ hiểu.

- **Những vấn đề tồn tại**

Giao diện chương trình chưa thân thiện, các chức năng đơn giản và không có nhiều sự giao tiếp với người dung.

- **Hướng phát triển**

Xây dựng thêm nhiều chức năng khác đề giao tiếp trực quan với người dùng nhằm đem đến khả năng mô phỏng trực quan hơn.

KẾT THÚC

CHÂN THÀNH CẢM ƠN QUÝ THẦY CÔ