Chương 9 THIẾT KẾ USE CASE

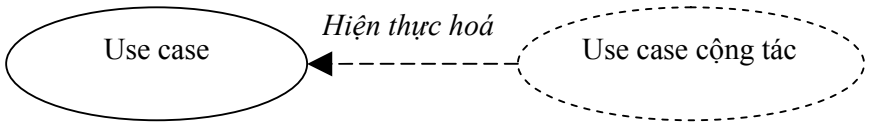
Mục tiêu

Mục tiêu của chương này cung cấp các kiếnthức về:

- Cung cấp cơ bản về kiến trúc ba tầng (tree-layer)
- Xác định các đối tượng ở tầng nghiệp vụ
- Xác định các đối tượng ở tầng truy cập dữ liệu và thiết kế method
- Xác định các đối tượng ở tầng giao diện và thiết kế method

Nội dung

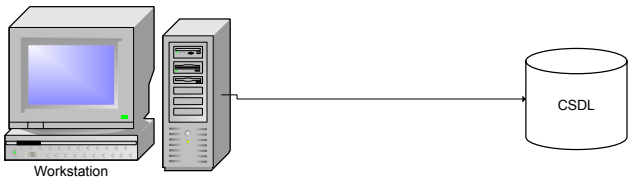
Mô hình use case chúng ta đạt được ở giai đoạn phân tích mô tả tính năng hệ thống từ phía người sử dụng. Các chức năng này được xem như là sự biểu diễn các yêu cầu chức năng mà hệ thống đáp ứng. Trong giai đoạn thiết kế, các chức năng này phải được mô tả ở gốc độ để cài đặt. Như vậy đối với một người thiết kế viên, chúng ta phải đặt câu hỏi là làm sao để biểu diễn sơ đồ use case này đủ chi tiết và trên một ngôn ngữ để có thể cài đặt được. Một cách tiếp cận là tách biệt diễn đạt chức năng thành hai phần: phần mô tả chức năng nhìn từ bên ngoài (từ phía người dùng: hệ thống có những chức năng gì cung cấp cho người sử dụng) và phần mô tả chức chức nhìn từ bên trong (từ phía người phát triển: làm sao để hiện thực hoá các chức năng để cài đặt nó).



Việc hiện thực hoá được đảm nhận bởi một các use case cộng tác và do đó thiết kế một use case chính tập trung vào thiết kế use case cộng tác. Nội dung thiết kế use case cộng tác bao gồm: xác định thêm các đối tượng hệ thống phần mềm cùng cộng tác trong việc thực hiện use case và sự tương tác giữa chúng. Trong tiếp cận kiến trúc ba tầng (tree-layer), các đối tượng hệ thống phần mềm là các đối tượng ở tầng giao diện và tầng truy cập dữ liệu. Sau đó, xác định sự tương tác giữa các đối tượng trong use case nhằm phát hiện các method cho các đối tượng của hai tầng này cũng như hoàn thiện việc tinh chế method cho các lớp. Kết quả của giai đoạn này là một sơ đồ lớp đầy đủ để chuẩn bị cho việc cài đặt hệ thống phần mềm.

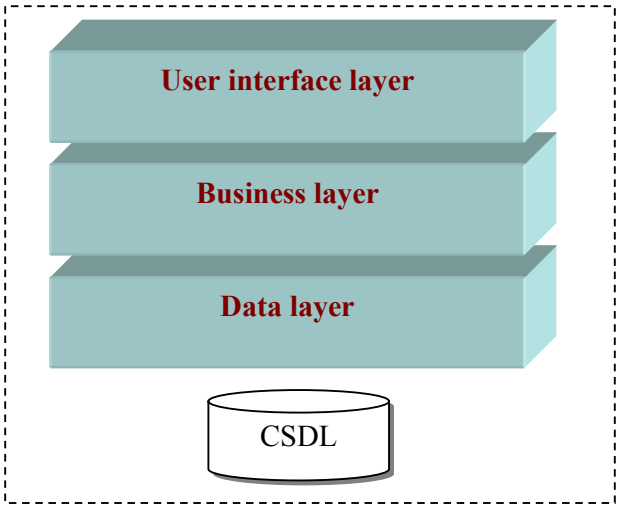
Kiến trúc 3 tầng (three - layer)

Hầu hết các hệ thống được phát triển sử dụng các công cụ CASE ngày nay hoặc trong các môi trường phát triển ứng dụng client – server có xu hướng xây dựng một kiến trúc hai tầng (two – layer): giao diện (interface) và dữ liệu (data). Trong một hệ thống hai tầng, các màn hình giao diện người dùng liên kết để truy cập dữ liệu thông qua các đoạn chương trình được cài trực tiếp trên các giao diện. Ví dụ, trong một chương trình viết trên Visual Basic trong một form giao diện, một thủ tục xử lý biến cố trong button "Update" của form này có tên bUpdate_Click() có thể thực hiện luôn việc truy cập và cập nhật CSDL trực tiếp, như vậy thủ tục cài đặt luôn các ngữ nghĩa về tác nghiệp (business). Việc thiết kế theo mô hình này tạo ra một sự phụ thuộc rất lớn giữa giao diện và CSDL và do đó, rất khó để cải tiến, bảo trì và tái sử dụng.



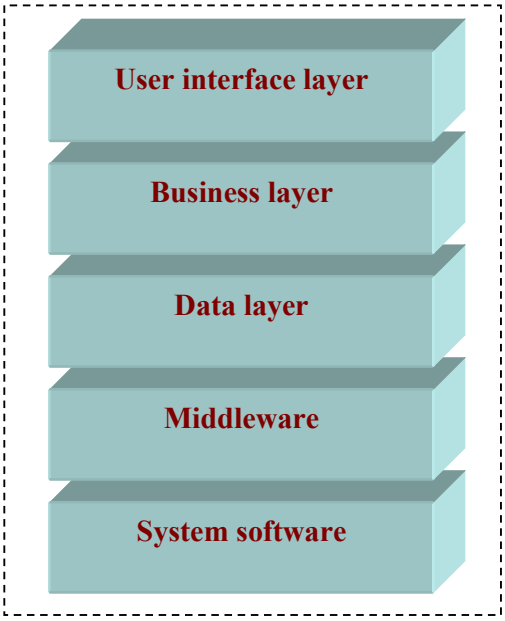
Hình 5. Kiến trúc hai lớp: giao diện và dữ liệu

Một cách tiếp cận khác về kiến trúc tốt hơn chính là tạo ra sự độc lập giữa giao diện và người sử dụng bằng cách cô lập các chức năng của giao diện với các chức năng tác nghiệp (business), và cô lập các chức năng tác nghiệp với các chi tiết về truy cập CSDL đó là cách tiếp cận ba tầng (three-layer). Từ cách tiếp cận này cho phép chúng ta tạo ra được các đối tượng đại diện các đối tượng hữu hình trong thực tế nhưng hoàn toàn độc lập với cách thức mà các đối tượng này trình bày tới người dùng hoặc là với cách mà dữ liệu của nó được lưu trữ vật lý trong CSDL. Do đó, ba tầng trong cách tiếp cận này là: tầng giao diện người dùng (user interface layer), tầng tác nghiệp (business layer), và tầng truy cập dữ liệu (data layer).



Hình 6. Sơ đồ biểu diễn tiếp cận ba tầng

Một tiếp cận khác đầy đủ hơn của một kiến trúc hệ thống có thể được trình bày như sơ đồ sau:



Hình 7. Một sơ đồ khác của cách tiếp cận ba tầng (nhiều tầng)

Trong đó,

Tầng Middleware: chứa các thành phần xây dựng giao diện (ví dụ: thành phần dạng ActiveX), thành phần giao diện tới các hệ quản trị CSDL (ví dụ: ODBC, JDBC driver), các dịch vụ hệ điều hành độc lập với platform, các thành phần nhúng OLE (ví dụ: các công cụ soạn thảo sơ đồ nhúng, các bảng tính nhúng,…).

Tầng System software: chứa các thành phần về hệ điều hành, CSDL, giao diện tới các phần cứng (ví dụ: các driver phần cứng cụ thể), v.v…

Xác định lớp ở tầng dịch vụ tác nghiệp (business layer)

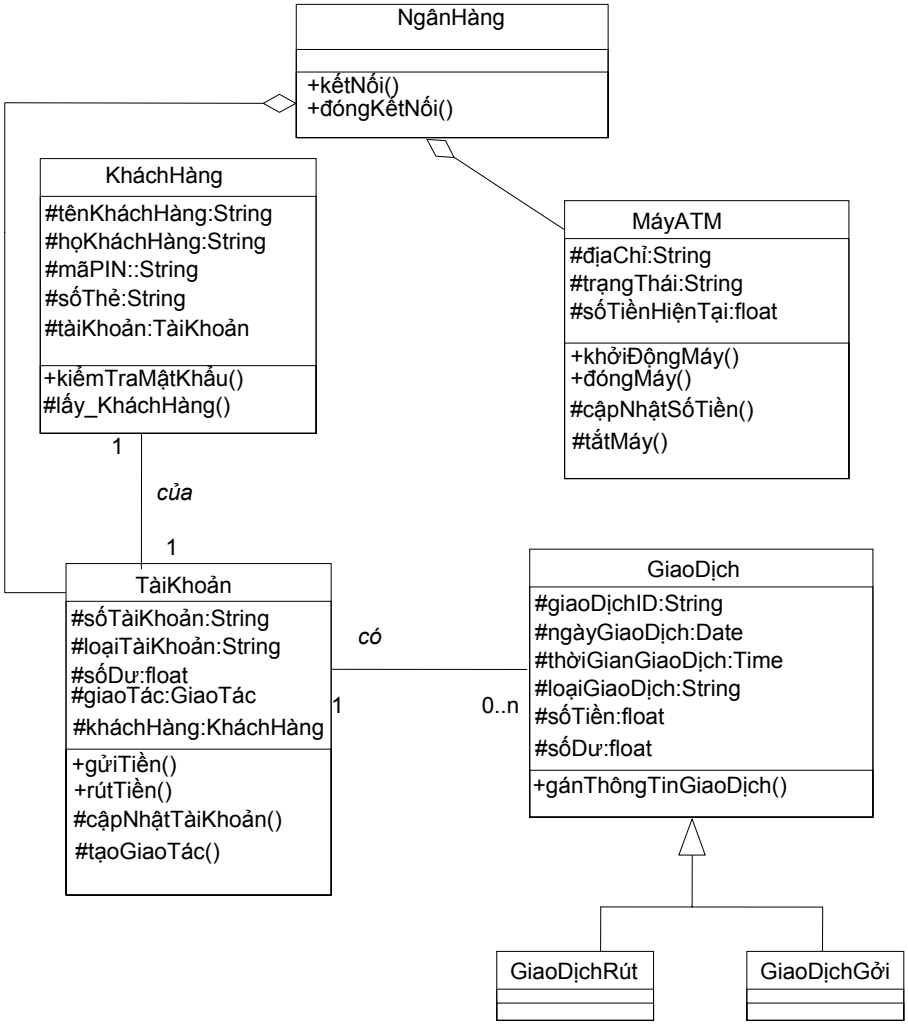
Tầng này chứa đựng tất cả đối tượng mô tả tác thành phần nghiệp hệ thống (bao gồm cả dữ liệu và hành vi). Nó diễn đạt các đối tượng tồn tại trong thực tế vào trong hệ thống cần quản lý. Ví dụ, đơn đặt hàng, khách hàng, hoá đơn, nhà cung cấp,… hầu hết các phương pháp luận phân tích thiết kế đều đưa ra phương pháp xác định đối tượng này trong giai đoạn phân tích. Tuy nhiên, khi xác định các đối tượng ở lớp này chúng ta phải luôn nhớ hai điều sau:

- Các đối tượng ở tầng tác nghiệp không nên quan tâm đến cách thức nó được hiển thị và bởi ai. Các đối tượng này được thiết kế để độc lập với bất kỳ một giao diện cụ thể, và vì vậy cách thức chi tiết để hiển thị một đối tượng nên tồn tại trong tầng giao diện thay vì trong tầng tác nghiệp.
- Các đối tượng ở tầng tác nghiệp cũng không nên quan tâm đến nguồn gốc của nó hình thành. Có nghĩa là các đối tượng này sẽ độc lập về dữ liệu của nó được lấy từ truy cập CSDL hay là từ truy xuất tập tin.

I.1.1. Xác định các class tầng tác nghiệp

Các class ở tầng tác nghiệp đã được xác định trong giai đoạn phân tích (xem chương 6). Trong phần này chúng tả mô tả lại theo từng use case để cho phép chúng ta có một cách nhìn về những gì mà các đối tượng ở tầng tác nghiệp kế hợp với nhau trong hoạt động đáp ứng yêu cầu sử dụng được mô tả thông qua use case. Chú ý rằng một lớp trong tầng này đều có thể tham gia xử lý trong nhiều use case khác nhau.

Các kết quả thiết kế lớp trong chương 7 đã cho chúng ta một sơ đồ thiết kế về tầng nghiệp vụ này.



Sơ đồ lớp của hệ thống ATM tầng nghiệp vụ

Xác định lớp ở tầng truy cập dữ liệu (data layer)

Tầng này chứa các đối tượng nhằm mục đích cung cấp các dịch vụ về dữ liệu cho tầng tác nghiệp. Nói chung, tất cả các nhu cầu truy cập CSDL hoặc tập tin để thao tác trên dữ liệu được lưu trữ của hệ thống đều phải thông qua tầng này. Do đó, các đối tượng tầng này phải truy cập vật lý CSDL ở các vị trí (CSDL quan hệ, file, internet,…) và xử lý nó. Hai nhiệm vụ chính của tầng này là:

- Chuyển dịch các yêu cầu: chuyển dịch tất cả các yêu cầu liên quan đến dữ liệu từ tầng tác nghiệp đến một phương thức truy cập dữ liệu thích hợp (dạng SQL, truy xuất file,…)
- Chuyển dịch kết quả: chuyển dịch tất cả dữ liệu truy cập được tới các đối tượng tác nghiệp thích hợp.

Xác định các đối tượng lưu trữ và persistence

Một chương trình sẽ tạo ra một số lượng dữ liệu trong quá trình thực thi. Mỗi dữ liệu sẽ có một thời gian sống (lifetime) khác nhau. Dựa vào thời gian sống này chúng ta có thể phân thành những loại dữ liệu sau:

- Là kết quả tạm thời để đánh giá một biểu thức
- Các biến trong quá trình thực thi một thủ tục (các tham số và biến trong phạm vi cục bộ)
- Các biến toàn cục và các biết cấp phát một cách tự động
- Dữ liệu tồn tại giữa các lần thực thi một chương trình
- Dữ liệu tồn tại giữa các phiên bản của một chương trình
- Dữ liệu tồn tại lâu hơn chương trình

Ba loại dữ liệu đầu tiên đều gọi là dữ liệu tạm thời (transient data). Là dữ liệu có thời gian sống phụ thuộc vào thời gian sống của tiến trình sử dụng nó. Khi tiến trình kết thúc thì dữ liệu này bị giải phóng. Ngược lại, ba loại dữ liệu cuối gọi là dữ liệu persistent (persistent data). Các dữ liệu này tồn tại lâu hơn tiến trình sử dụng nó và có thể độc lập với tiến trình này. Các ngôn ngữ lập trình cung cấp sự hỗ trợ hoàn hảo cho các loại dữ liệu tạm thời. Các loại dữ liệu persistent sẽ được quản lý bởi hệ quản trị cơ sở dữ liệu hoặc hệ thống file lưu trữ.

Đối với các đối tượng cũng được áp dụng một cách tương tự. Các đối tượng cũng có một thời gian sống. Chúng được tạo ra một cách tường minh và có thể tồn tại trong một khoảng thời gian ngắn (thời gian của một ứng dụng, của một phiên,…). Tuy nhiên, cũng có đối tượng tồn tài lâu hơn thời gian của một ứng dụng, để làm điều này thì đối tượng phải được lưu trữ ra tập tin hoặc cơ sở dữ liệu. Việc lưu trữ này sẽ cung cấp cho đối tượng thời gian sống lâu hơn quá trình của tiến trình. Đặc tính này người ta gọi là *persistent*.

Trong hệ thống ATM, các lớp persistent gồm: KháchHàng, TàiKhoản, GiaoTác, GiaoTácGửi, GiaoTácRút. Các lớp như MáyATM, NgânHàng được xác định không phải là lớp persistent bởi vì chúng ta không có nhu cầu lưu trữ thông tin của các đối tượng các lớp này.

Chuyển đổi đối tượng sang mô hình quan hệ

Trong cơ sở dữ liệu quan hệ, một lược đồ được hình thành bởi các bảng (table) gồm các cột và dòng. Trong mô hình đối tượng, tương ứng tới một bảng là một lớp (hoặc nhiều lớp). Các thành phần tương ứng như sau:

- Một cột ứng với một thuộc tính (persistent) lớp
- Một dòng của bảng ứng với một đối tượng (thể hiện của một lớp)
- Một stored procedure có thể tương ứng với một method.

Như vậy, việc chuyển đổi sơ đồ lớp sang lược đồ quan hệ gồm có những công việc sau:

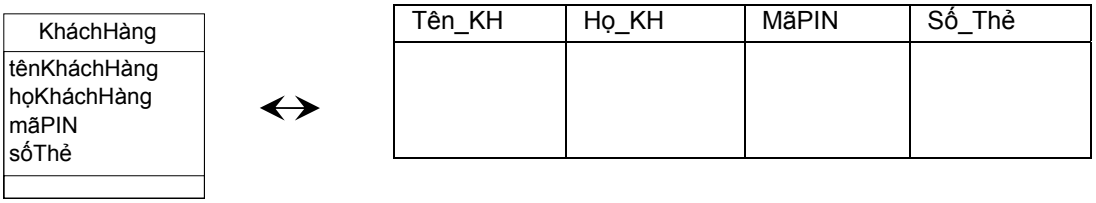
- Chuyển đổi lớp – bảng
- Chuyển đổi mối liên kết
    - Chuyển đổi liên kết kết hợp
    - Chuyển đổi liên kết kế thừa

Chuyển đổi lớp – bảng

Đa số các trường hợp thì việc chuyển đổi này là một – một. Một lớp sẽ chuyển thành một bảng cụ thể như sau:

- Một lớp → một bảng

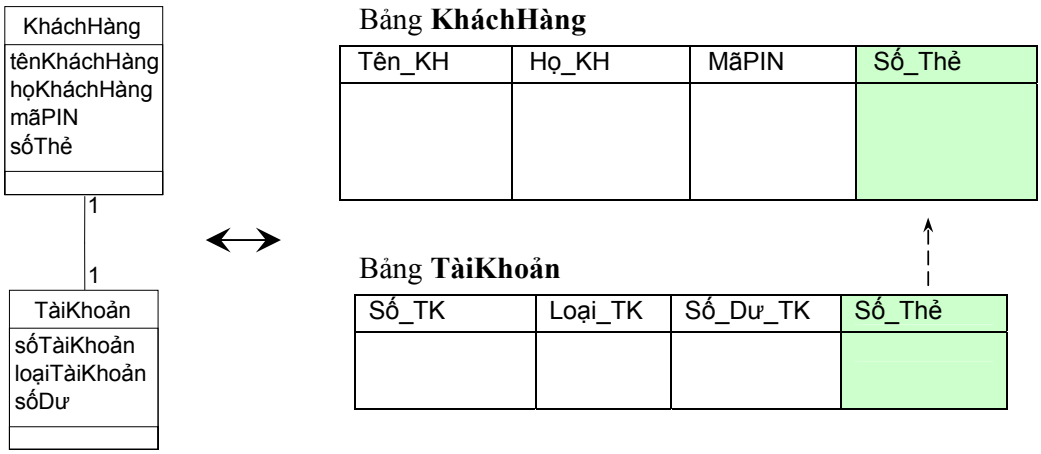
- Một thuộc tính (persistent) → một cột: chỉ có các thuộc tính có nhu cầu lưu trử và được đòi hỏi bởi ứng dụng sẽ được chuyển thành cột của bảng.
- Một đối tượng (thể hiện) → một dòng



Chuyển đổi mối liên kết

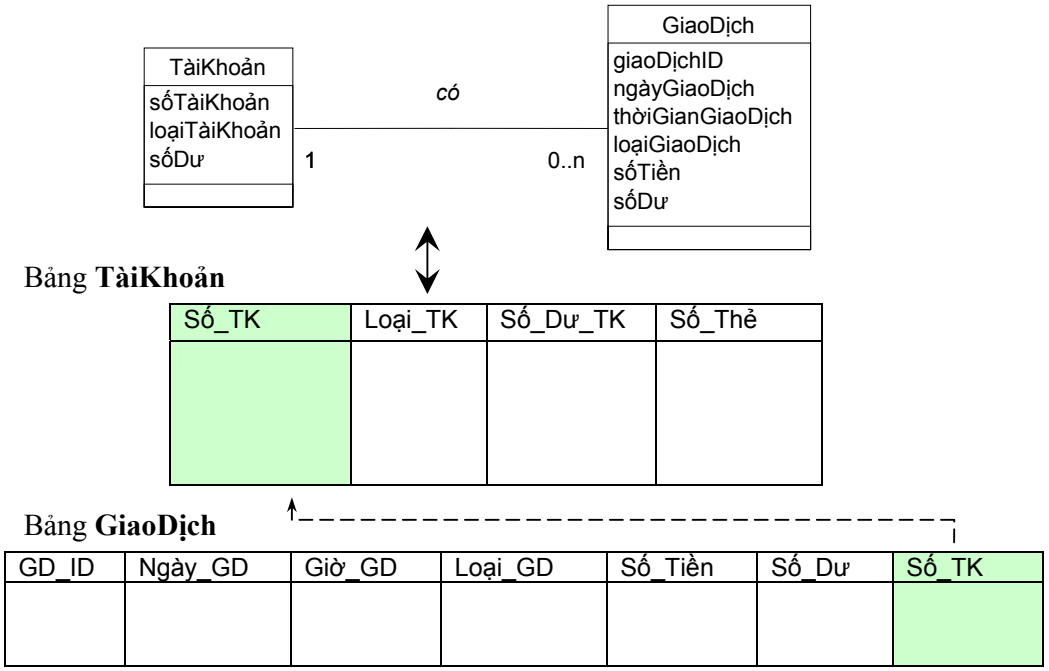
Chuyển đổi liên kết kết hợp

Trường hợp 1



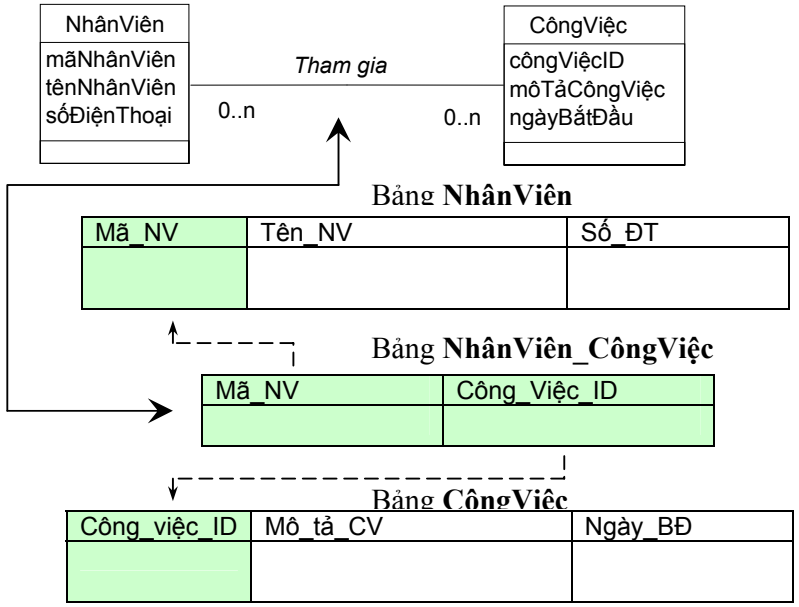
Trong chuyển đổi mối kết hợp dạng 1 – 1, chúng ta có thể lấy cột khoá chính trong một bảng chuyển qua bảng khác làm khoá ngoại. Trong mối kết hợp 1 – 1 trên (giữa lớp KháchHàng và TàiKhoản), chúng ta có thể lấy khoá của bảng KháchHàng (Số_Thẻ) đưa vào bảng TàiKhoản là khoá ngoại. Hoặc ngược lại, chúng ta có thể lấy khoá của bảng TàiKhoản (Số_TK) đưa vào bảng KháchHàng làm khoá ngoại. Hoặc thực hiện cả hai trường hợp.

Trường hợp 2



Trường hợp 3

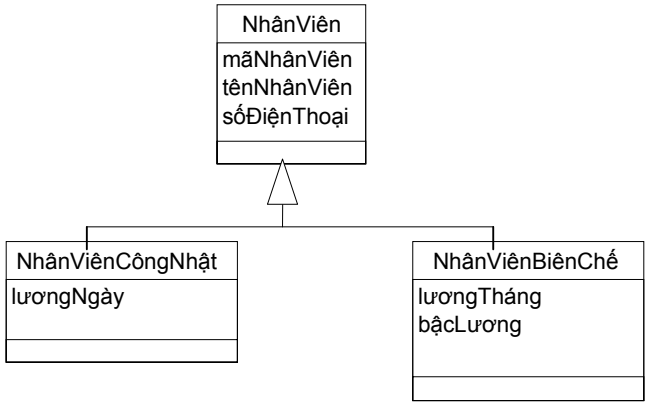
Trong chuyển đổi mối kết hợp dạng 1 – n, chúng ta lấy cột khoá chính của bảng ứng với lớp phía 1 trong mối kết hợp đưa vào bảng ứng với lớp phía n làm khoá ngoại. Trong ví dụ trên, bảng GiaoDịch sẽ lấy thuộc tính Số_TK trong bảng TàiKhoản làm khoá ngoại.



Trong chuyển đổi mối kết hợp n – n, chúng ta tạo ra một bảng cho mối kết hợp đó bằng cách lấy các khoá chính của các bảng đưa vào bảng mới này như là các khoá ngoại. Trong ví dụ trên, mối kết hợp *Tham gia* giữa lớp NhânViên và lớp CôngViệc sẽ được mô tả bởi một bảng NhânViên_CôngViệc bao gồm các cột Mã_NV và Công_Việc_ID lần lượt là khoá chính của bảng NhânViên và bảng CôngViệc.

Chuyển đổi liên kết kế thừa

Trong lược đồ quan hệ không có khái niệm kế thừa mà chúng ta thường dùng liên kết khoá chính – khoá ngoại để diễn đạt điều này. Sau đây là các trường hợp mà chúng ta có thể quyết định chọn một:



Trường hợp 1

Bảng **NhânViên**

Mã_NV	Tên_NV	Điện_Thoại	Lương_Ngày	Lương_Tháng	Bậc_Lương	***Loại_NV***

Chỉ sử dụng một bảng lưu trữ tất cả các loại nhân viên. Do đó, các thuộc tính của bảng được hình thành từ các thuộc tính của lớp NhânViên, NhânViênCôngNhật và Nhân ViênBiênChế. Ngoài ra chúng ta cũng đưa vào thêm một thuộc tính nhằm phân loại dòng này thuộc đối tượng của lớp nào.

Trường hợp 2

Bảng **NhânViên**

Mã_NV	Tên_NV	Điện_Thoại

Bảng **NhânViênCôngNhật**

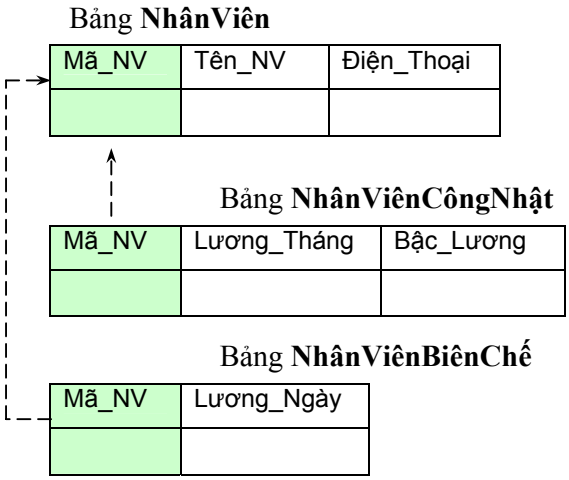
Mã_NV	Tên_NV	Điện_Thoại	Lương_Tháng	Bậc_Lương

Bảng **NhânViênBiênChế**

Mã_NV	Tên_NV	Điện_Thoại	Lương_Ngày

Sử dụng ba bảng tương ứng cho ba lớp. Tuy nhiên, nhằm mô tả sự thừa kế trong các bảng NhânViênCôngNhật và NhânViênBiênChế chúng ta thêm vào tất cả các thuộc tính của bảng nhân viên. Các thể hiện tương ứng của các nhân viên công nhật hay biên chế sẽ chỉ lưu trong bảng tương ứng.

Trường hợp 3



Giống như trường hợp 2. Ở hai bảng NhânViênCôngNhật và NhânViênBiênChế chỉ lưu khoá ngoại tham chiếu đến bảng nhân viên để tham khảo thông tin thừa kế.

Trường hợp 4

Bảng **NhânViênCôngNhật**

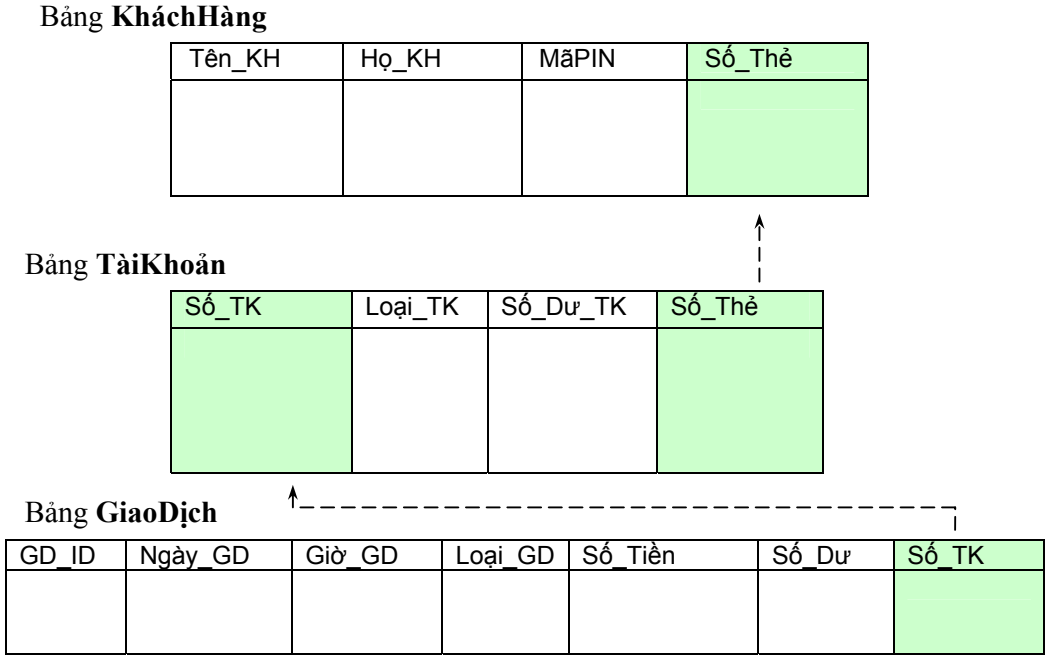
Mã_NV	Tên_NV	Điện_Thoại	Lương_Tháng	Bậc_Lương

Bảng **NhânViênBiênChế**

Mã_NV	Tên_NV	Điện_Thoại	Lương_Ngày

Chỉ dùng hai bảng NhânViênCôngNhật và NhânViênBiênChế. Tuy nhiên, tất cả các thuộc tính của lớp NhânViên sẽ được đưa vào hai bảng này nhằm mô tả sự thừa kế. Nếu chúng ta muốn truy xuất thông tin về lớp NhânViên thì chúng ta có thể tạo một khung nhìn (view) hội của hai bảng này.

Sơ đồ cài đặt các đối tượng persistent của hệ thống ATM dùng mô hình quan hệ như sau:



Trong lược đồ trên của máy ATM, chúng ta áp dụng trường hợp 1 cho cấu trúc các lớp GiaoDịch, GiaoDịchGửi và GiaoDịchRút.

Xác định lớp ở tầng truy cập dữ liệu

Mục tiêu chính của việc tạo ra một tầng truy cập dữ liệu chính là để tạo ra các lớp có nhiệm vụ truy cập tới các vị trí mà dữ liệu thực sự được lưu trữ nhằm giúp cho tầng nghiệp vụ không quan tâm đến vị trí cũng như hình thức lưu trữ của dữ liệu (dạng tập tin, cơ sở dữ liệu quan hệ, cơ sơ dữ liệu đối tượng, internet, DCOM,…). Các đối tượng ở tầng này phải có trách nhiệm cung cấp một liên kết giữa nghiệp vụ (cách nhìn theo đối tượng) và dữ liệu lưu trữ. Tiến trình xác định các lớp ở tầng này gồm các bước sau:

- Với mỗi lớp persistent ở tầng nghiệp vụ, tạo một lớp tương ứng ở tầng truy cập dữ liệu. Ví dụ, nếu chúng ta có ba lớp ở tầng nghiệp vụ là Class1, Class2, Class3 thì chúng ta tạo ra ba lớp tương ứng ở tầng truy cập dữ liệu là: ClassDB1, ClassDB2, ClassDB3.
- Xác định mối kết hợp: tương tự như xác định mối kết hợp giữa các lớp ở tầng nghiệp vụ
- Đơn giản hoá các lớp và mối kết hợp: mục tiêu chính là để loại các lớp và cấu trúc dư thừa hoặc không cần thiết. Thông thường, chúng ta kết hợp nhiều lớp thành một và đơn giản hoá cấu trúc lớp cha – lớp con.
    - Các lớp dư thừa: nếu chúng ta có hai lớp trở lên cùng cung cấp các dịch vụ tương tự nhau, chúng ta giữ lại một và loại đi một.
    - Các method: xem lại các lớp chỉ có một hoặc hai method xem có thể có thể kết hợp với các lớp khác? Thông thường, chúng ta chỉ quan tâm đến các method có nhu cầu truy cập đến dữ liệu lưu trữ. Các method đó là: đọc dữ liệu từ dữ liệu lưu trữ của đối tượng, xoá dữ liệu của đối tượng khỏi dữ liệu lưu trữ và cập nhật các thay đổi của đối tượng xuống dữ liệu lưu trữ.
- Tạo mối kết hợp giữa lớp tầng truy cập dữ liệu và lớp của nó tương ứng ở tầng nghiệp vụ là mối kết hợp dạng thành phần (aggregation). Tạo thuộc tính tham chiếu cho các

lớp của tầng nghiệp vụ tham chiếu đến lớp ở tầng truy cập dữ liệu dựa trên mối liên kết vừa xác định.

- Lặp lại tiến trình này

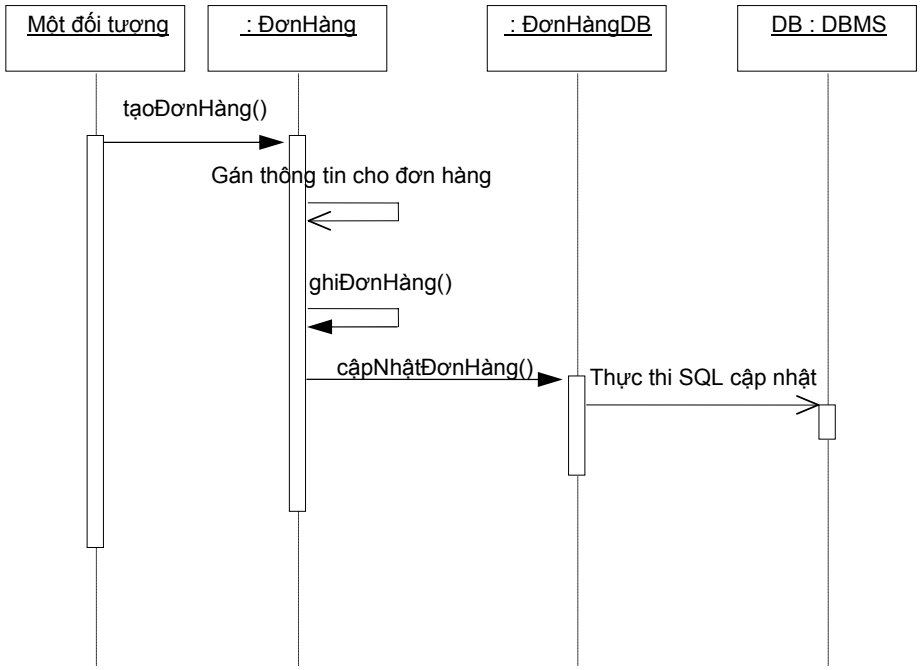
Xác định method các lớp tầng truy cập dữ liệu

Trong thiết kế hướng đối tượng nhằm đảm bảo tính bao bọc trong các cài đặt chi tiết, chúng ta mong muốn các đối tượng persistent được quan sát giống như một đối tượng tạm thời (transient) trong hệ thống. Nghĩa là chúng ta phải tạo được một cách nhìn trong suốt cho các đối tượng trong hệ thống mà không phân biệt và xử lý khác nhau giữa đối tượng persistent và bất kỳ đối tượng nào khác. Tuy nhiên, đây cũng là một vấn đề của hệ thống bởi vì dữ liệu và trạng thái của các đối tượng persistent được quản lý tách biệt khỏi chương trình ứng dụng. Do đó, sự nhất quán giữa đối tượng trong ứng dụng và trạng thái của nó trong cơ sở dữ liệu phải luôn luôn được đặt ra trong thiết kế hệ thống hướng đối tượng. Để đảm bảo điều này thì có nhiều mặt một ứng dụng phải kiểm soát:

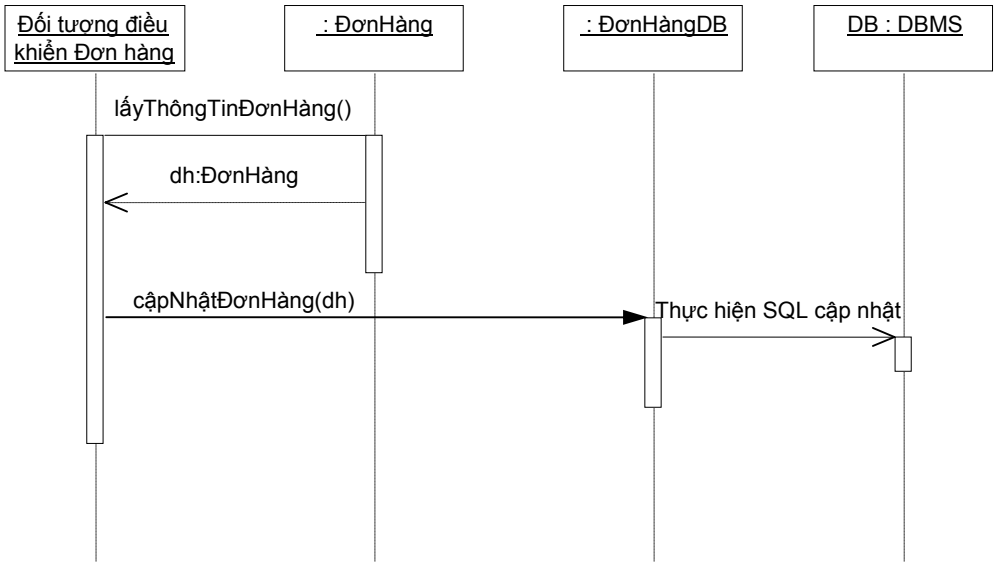
- Đọc và lưu các đối tượng persistent
- Xoá các đối tượng persistent
- Quản lý giao tác trên các đối tượng persistent
- Kiểm soát cơ chế khoá và truy cập đồng hành

Ghi đối tượng persistent

Có hai trường hợp cần xem xét: thời điểm ban đầu khi đối tượng được tao ra và phải ghi vào cơ sở dữ liệu; các thời điểm tiếp theo khi chương trình cập nhật trạng thái của đối tượng và thay đổi này cũng được ghi vào cơ sở dữ liệu.



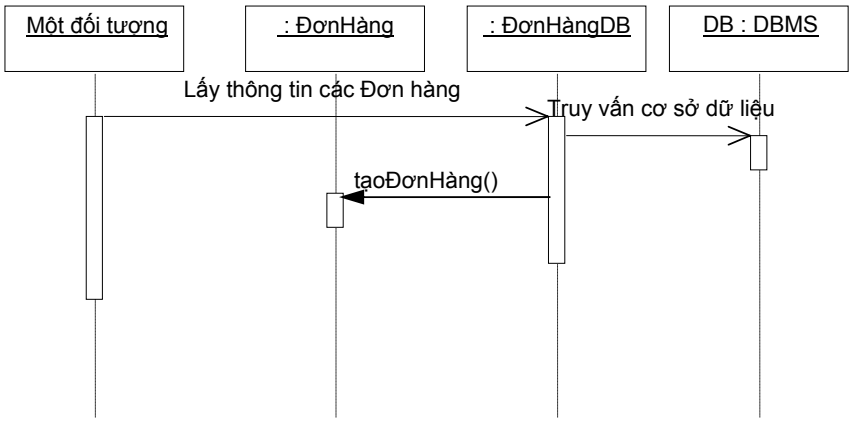
Sơ đồ tuần tự trên mô tả một trong những giải pháp cập nhật đối tượng persistent. Khi một đối tượng đơn hàng được tạo ra trong ứng dụng, đối tượng này ngay lập tức được ghi vào cơ sở dữ liệu bởi lớp ĐơnHàngDB tầng truy cập dữ liệu. Tương tự cho hoạt động cập nhật, khi một đối tượng trong ứng dụng thay đổi trạng thái của đối tượng đơn hàng. Ngay lập tức sự thay đổi này được cập nhật vào cơ sở dữ liệu bởi ĐơnHàngDB.



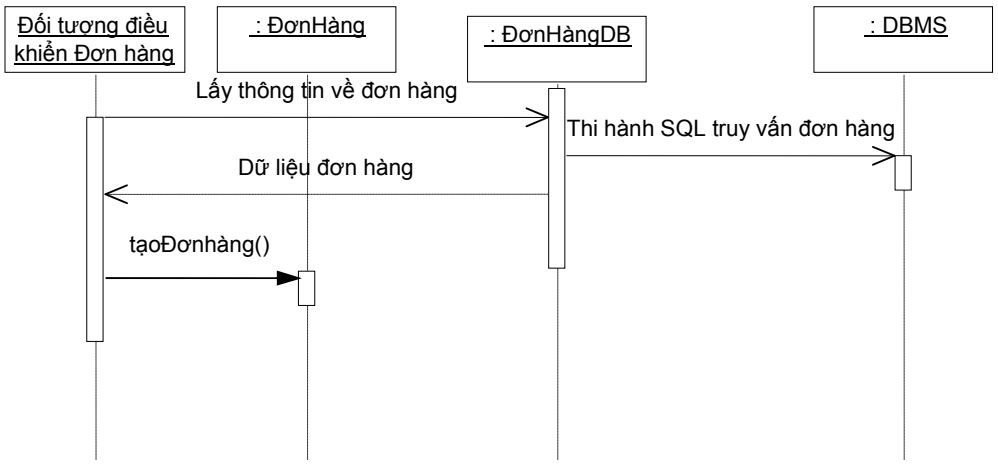
Một giải pháp khác cho cập nhật đối tượng persistent. Chúng ta xây dựng một đối tượng điều khiển và khi nào thông tin đối tượng persistent đuợc cập nhật sẽ được quản lý bởi đối tượng này. Method cậpNhậtĐơnHàng() sẽ được thiết kế thông minh để nhận ra một đơn hàng là tạo mới hay cập nhật. Trong giải pháp này, việc cập nhật được quản lý một cách tường minh và hệ thống chấp nhận các khoảng thời điểm thiếu sự nhất quán giữa đối tượng và dữ liệu lưu trữ về đối tượng.

Đọc đối tượng persistent

Việc lấy thông tin về các đối tượng persistent trong sơ sở dữ liệu cần thiết trước khi một ứng dụng gởi các thông điệp tới đối tượng. Vấn đề là khi gửi thông điệp cho một đối tượng thì phải đảm bảo đối tượng đó đã tồn tại trong bộ nhớ. Do vậy, chúng ta cần thiết kế một method truy vấn cơ sở dữ liệu để nhận đúng thông tin về đối tượng. Thông thường các đối tượng trong hệ thống liên kết với nhau qua mối kết hợp và vì vậy chúng ta đọc thông tin đầu tiên cho đối tượng **gốc**, rồi sau đó chúng ta sẽ đọc tiếp theo các đối tượng còn lại dựa theo mối kết hợp với đối tượng gốc.

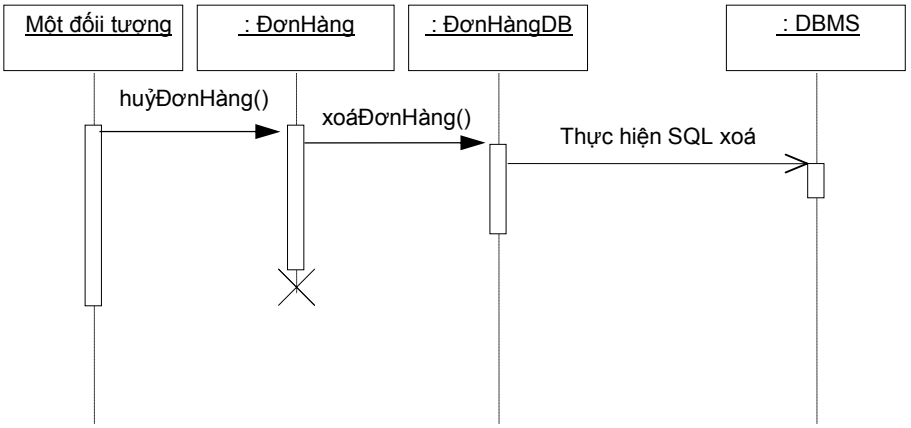


Một giải pháp khác là dùng một đối tượng điều khiển quản lý việc độc và tạo đối tượng đơn hàng được minh hoạ theo sơ đồ sau:

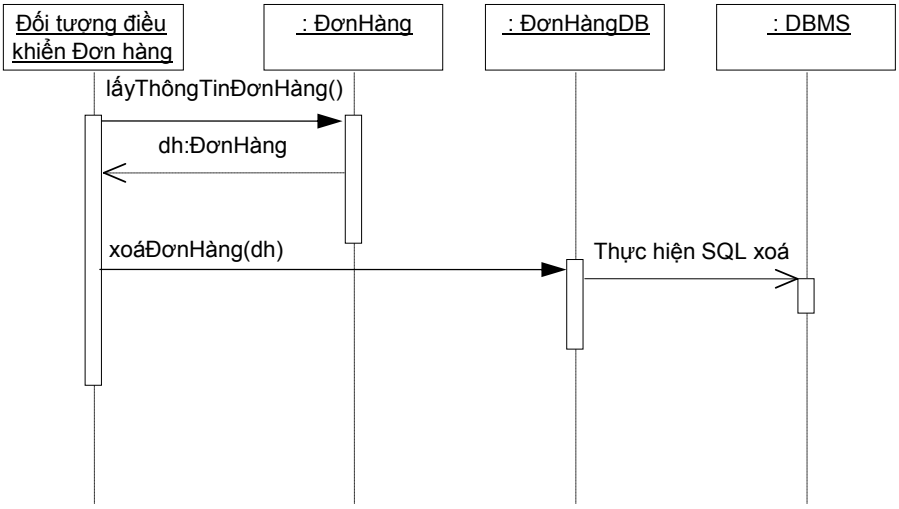


Xoá đối tượng persistent

Không giống như các đối tượng tạm thời (transtient) trong hệ thống, việc kết thúc sự hiện diện của đối tượng trong ứng dụng sẽ kết thúc vòng đời của đối tượng đó. Ngược lại, đối với đối tượng persistent, việc kết thúc vòng đời của nó đòi hỏi phải huỷ bỏ dữ liệu của nó khỏi nơi lưu trữ (hoặc ít nhất đánh dấu nó không còn hoạt động).



Một giải pháp khác dùng một đối tượng điều khiển

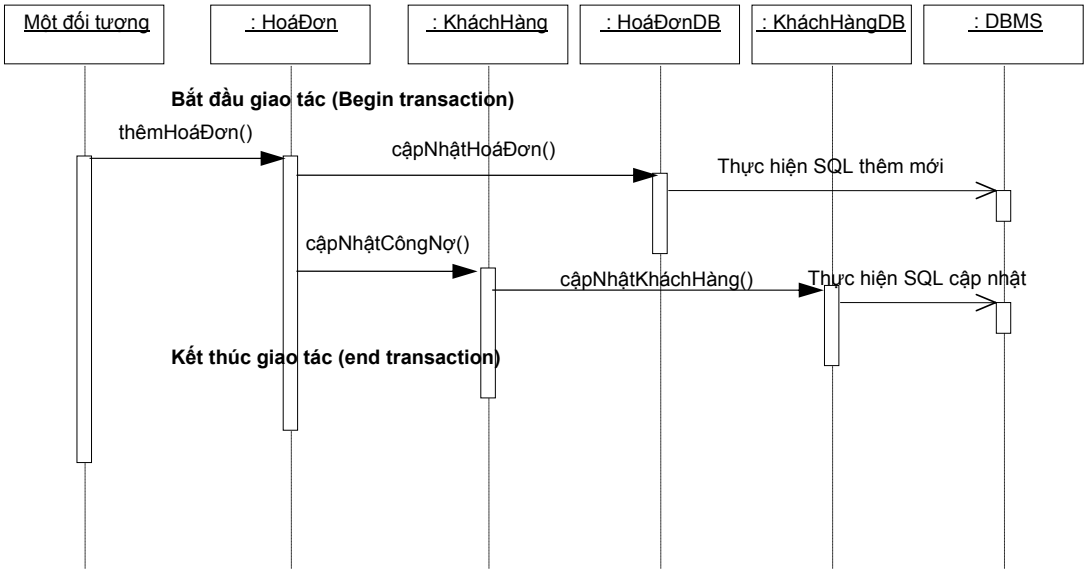


Quản lý giao tác (transaction)

Một giao tác xác định một tập các phép toán cập nhật trong cơ sở dữ liệu thành một đơn vị nhằm đáp ứng cho một yêu cầu cập nhật nghiệp vụ mà trong đó, hệ thống đòi hỏi phải cập nhật nhiều nguồn dữ liệu của nhiều đối tượng persistent khác nhau. Tất cả các phép toán cập nhật một giao tác hoặc thành công (thì giao tác thành công) hoặc không một phép toán nào thành công (thì giao tác không thành công).

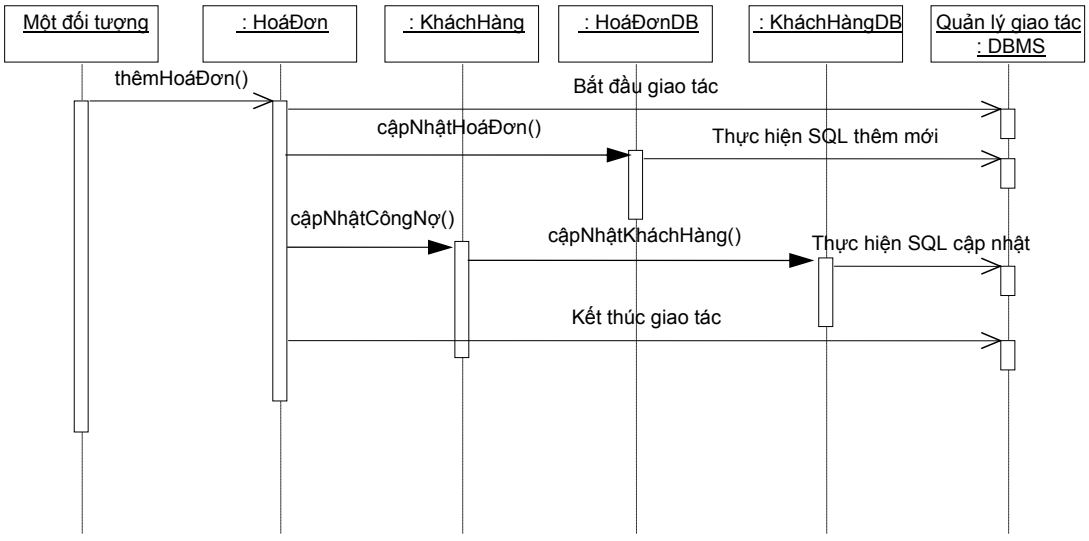
Có hai cách mô hình hoá một giao tác như sau:

Dùng văn bản để chỉ ra bắt đầu và kết thúc một giao tác



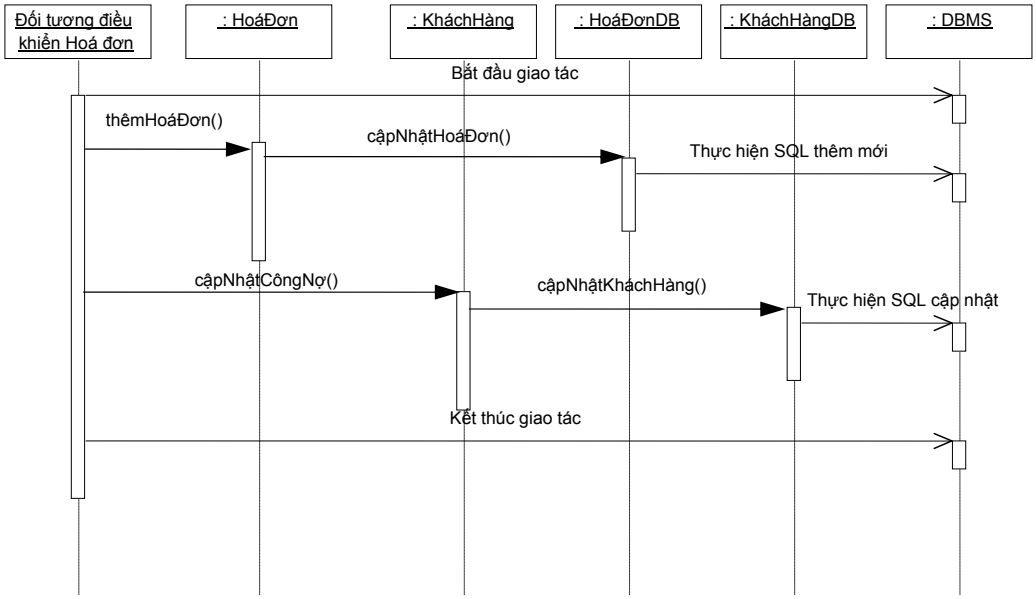
Sơ đồ trên minh hoạ việc quản lý một giao tác thêm một hoá đơn thanh toán. Giao tác này bao gồm hai phép toán cập nhật. Một là thêm mới một hoá đơn và hai là cập nhật lại thông tin công nợ của khách hàng. Giao tác kết thúc thành công nếu cả hai phép toán cập nhật đều thành công. Nếu một trong hai phép toán thực hiện không thành công thì giao tác sẽ không thành công và lúc này tất cả hai phép toán trên xem như chưa được thực hiện.

Dùng đối tượng quản lý giao tác



Chúng ta tạo một đối tượng Quản lý giao tác, tuỳ thuộc vào ngôn nhữ lập trình mà đối tượng này có kiểu khác nhau. Đối với một số ngôn ngữ thì nó là một đối tượng kiểu kết nối (connection) hoặc đối tượng kiểu thành phần hệ quản trị cơ sở dữ liệu trong ngôn ngữ lập trình đó. Sơ đồ minh hoạ trên chọn đối tượng quản lý giao tác là một đối tượng kiểu thành phần hệ quản trị cơ sở dữ liệu.

Một giải pháp dùng đối tượng điều khiển để quản lý giao tác được minh hoạ như sơ đồ dưới đây:



Trong sơ đồ, đối tượng điều khiển sẽ quản lý việc bắt đầu và kết thúc giao tác cũng như nội dung của một giao tác bằng việc quản lý các lệnh cập nhật của giao tác. Giải pháp này so với giải pháp trên thì co ưu điểm sau: làm cho đối tượng hoá đơn độc lập với đối tượng khách hàng. Thay vì làm cho đối tượng hoá đơn phụ thuộc vào đối tượng khách hàng qua việc gọi cậpNhậtCôngNợ() ở giải pháp trên. Giải pháp này chuyển sự phụ thuộc của hoá đơn và khách hàng vào đối tượng điều khiển và giữa đối tượng hoá đơn và khách hàng lúc này hoàn toàn độc lập.

Xác định lớp truy cập dữ liệu cho hệ thống ATM

Từ các lớp persistent của hệ thống ATM được xác định là: KháchHàng, TàiKhoản, GiaoDịch, GiaoDịchRút, GiaoDịchGửi chúng ta tạo ra các lớp truy cập dữ liệu tương ứng:

KháchHàngDB

TàiKhoảnDB

GiaoDịchDB (GiaoDịch, GiaoDịchRút, GiaoDịchGửi)

Các method được xác định:

KháchHàngDB::+đọcKháchHàng()

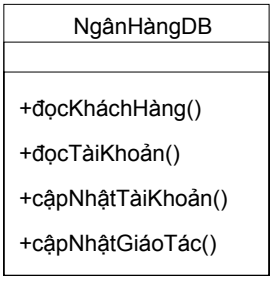
KháchHàng::+đọcTàiKhoản()

TàiKhoảnDB::+cậpNhậtTàiKhoản()

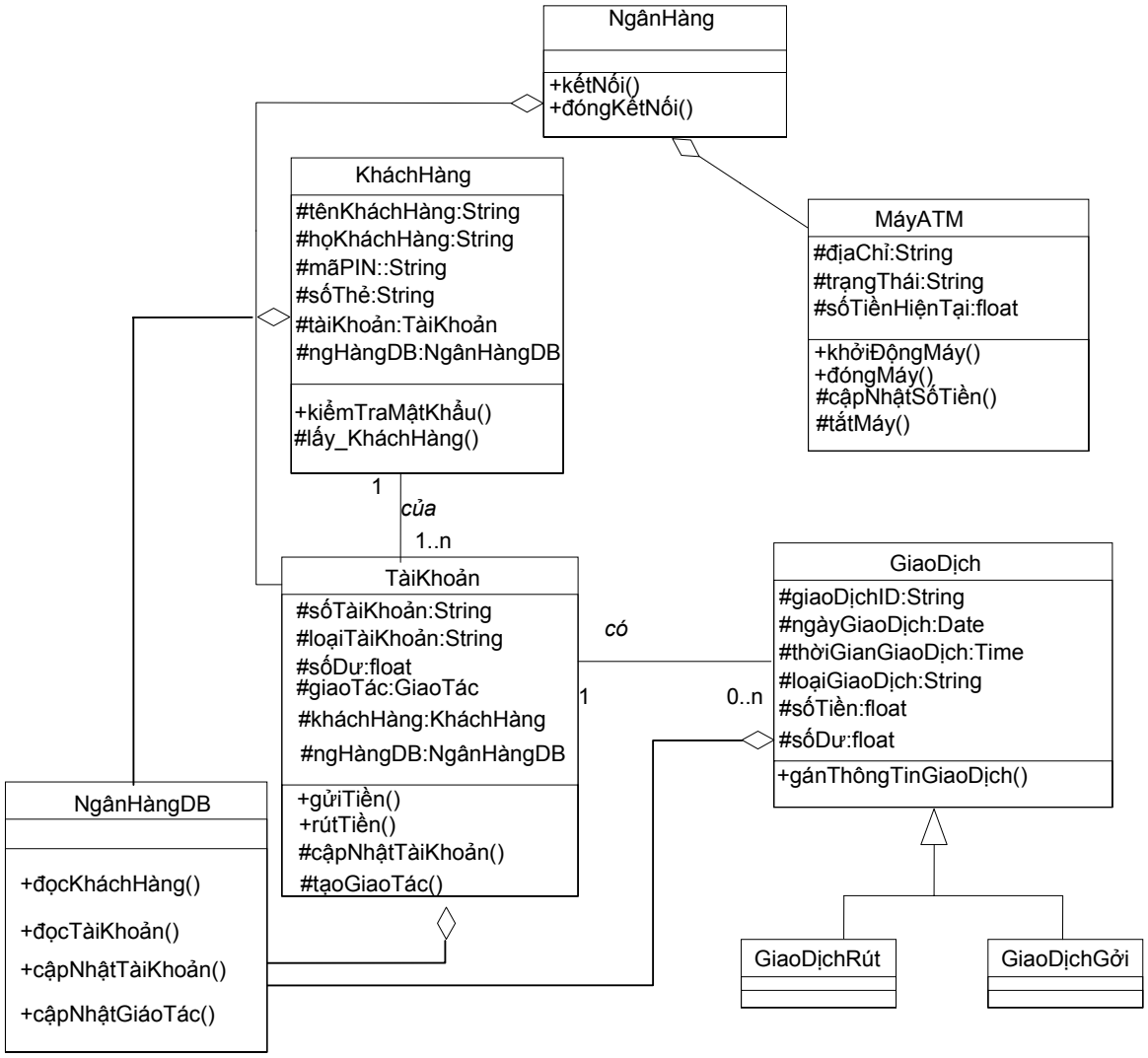
GiaoDịchDB::+cậpNhậtGiaoDịch()

Chúng ta gán mặc định cho các nghi thức ở tầng này là toàn cục (public), vì đa số đều được truy cập bởi những lớp khác, rồi trong quá trình thiết kế chi tiết cho từng method chúng ta sẽ xác định lại nghi thức này cho phù hợp nhằm đảm bảo tính bao bọc.

Bởi vì số lượng các method cho mỗi lớp này là ít (<= 2), cho nên chúng ta kết hợp tất cả lại thanh một lớp chung và gọi là NgânHàngDB.

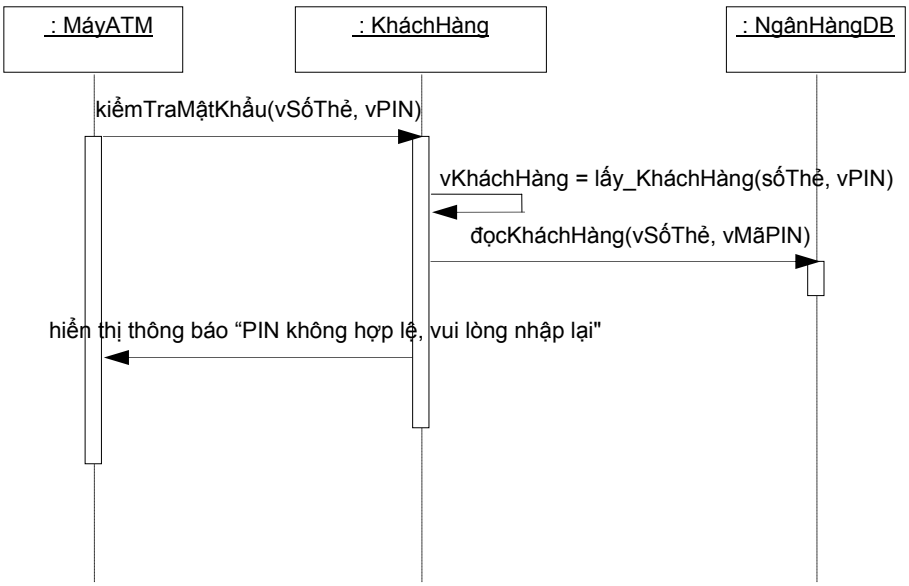


Sau đó chúng ta tạo liên kết aggregation từ lớp NgânHàngDB tới lần lượt các lớp: KháchHàng, TàiKhoản và GiaoDịch. Ứng với mỗi lớp vừa mới tạo liên kết tới lớp NgânHàngDB chúng ta thêm vào một thuộc tính tham chiếu đến lớp này.



Các method của lớp NgânHàngDB lần lượt sẽ được thiết kế chi tiết như sau:

NgânHàngDB::+đọcKháchHàng (vSốThẻ:String, vMãPIN:String): KháchHàng



Tinh chế chi tiết cho use case “Đăng Nhập” liên quan đến truy cập cơ sở dữ liệu theo sơ đồ trên, chúng ta đưa thêm vào một đối tượng NgânHàngDB và việc đọc dữ liệu thực sự về khách hàng sẽ do đối tượng này đả nhận qua method đọcKháchHàng. Sau đây chúng ta minh hoạ đoạn mã tương ứng:

```
KháchHàng::#lấy_KháchHàng(sốThẻ:String, mãPIN:String): KháchHàng
        vNgânHàngDB: NgânHàngDB
        return vNgânHàngDB.đọcKháchHàng (sốThẻ, mãPIN)
```

Ở đây chúng ta cần tạo ra một thể hiện của lớp truy cập NgânHàngDB và rồi gởi thông điệp tới đối tượng này để lấy thông tin về đối tượng khách hàng. Method đọcKháchHàng của NgạnHàngDB sẽ thực sự đọc dữ liệu từ cơ sở dữ liệu. Có thể minh hoạ method này truy cập cơ sở dữ liệu quan hệ bằng SQL như sau:

```
NgânHàngDB::+đọcKháchHàng (vSốThẻ:String, vMãPIN:String): KháchHàng
        -- kết nối cơ dữ liệu ở đây
        return
        SELECT * FROM KháchHàng
        WHERE Số_Thẻ = vSốThẻ AND Mã_PIN = vMãPIN
```

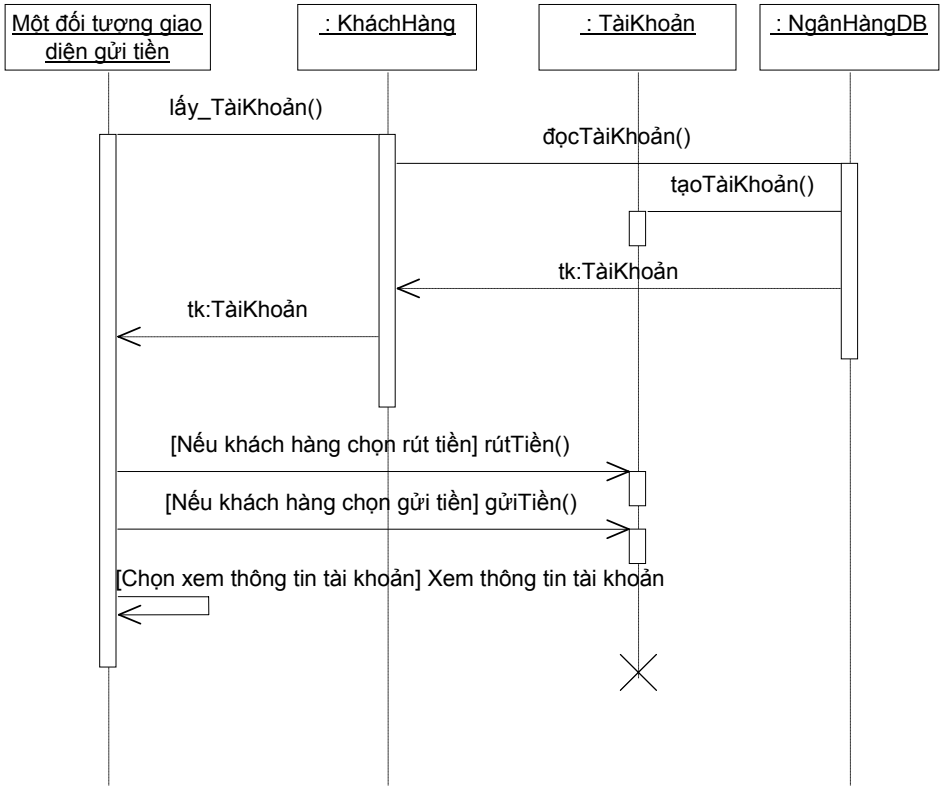
Giả sử rằng, câu truy vấn sẽ trả về dữ liệu khách hàng vào một đối tượng khách hàng trả và về cho method này. Tuỳ thuộc vào ngôn ngữ cài đặt cụ thể được chọn mà chúng ta thay đổi cho phù hợp, đoạn mã lệnh trên đây chỉ minh hoạ ý tưởng và cách viết độc lập ngôn ngữ.

NgânHàngDB::+đọcTàiKhoản(sốThẻ:String): TàiKhoản

Đây là một method đọc dữ liệu từ cơ sở dữ liệu để tạo một đối tượng persistent. Do đó, chúng ta có thể áp dụng một trong hai mẫu sơ đồ đọc đối tượng persistent ở phần trên.

Các use case liên quan đến method đọcTàiKhoản để tạo một đối tượng tài khoản phục vụ cho hoạt động của use case bao gồm: Giao dịch, Gửi tiền, Rút tiền, Truy vấn thông tin tài khoản. Sau đây chúng ta minh hoạ sơ đồ tuần tự thiết kế chi tiết các use case này qua sự tượng tác giữa tầng nghiệp vụ và tầng truy cập dữ liệu, đặc biệt là tương tác nhằm tạo đối tượng tài khoản.

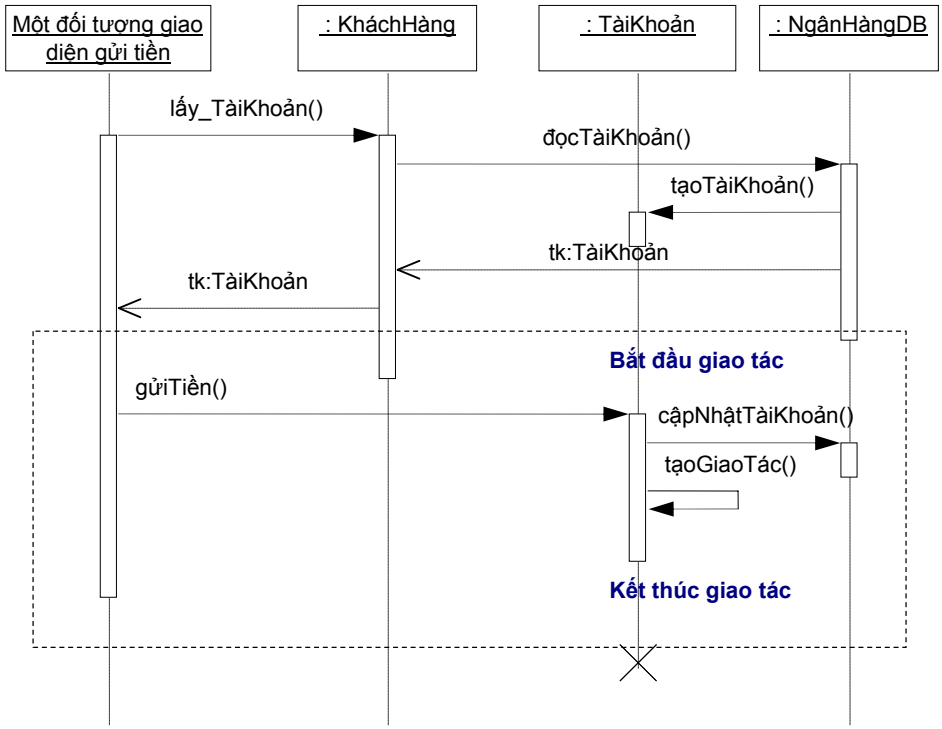
Use case Giao dịch



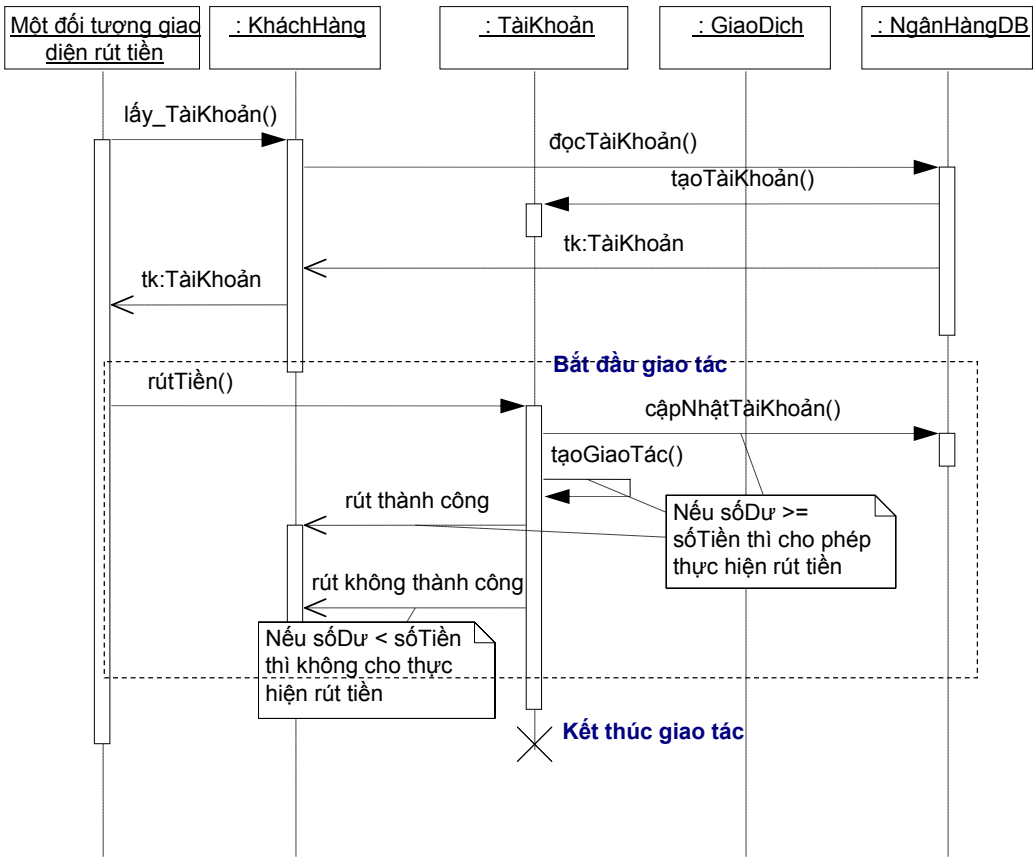
Sơ đồ tuần tự về use case Giao dịch được thiết kế: đầu tiên chúng ta giả lập một đối tượng tầng giao diện sẽ tương tác tới một đối tượng khách hàng (đã được tạo ra khi đăng nhập thành công) để tham khảo đến tài khoản của khách hàng bằng cách gởi thông điệp đến đối tượng truy cập dữ liệu là NgânHàngDB để thực hiện việc đọc dữ liệu từ cơ sở dữ liệu và tạo ra đối tượng tài khoản. Sau đó, đối tượng này sẽ được trả về cho đối tượng giao diện như một đối tượng để tham chiếu. Đối tượng giao diện sẽ được tinh chế trong phần sau của chương này.

Tuỳ theo loại giao dịch mà khách hàng chọn thì use cae này sẽ gọi thực hiện các use case mở rộng tương ứng. Các use case Rút tiền và Gửi tiền sẽ thực hiện dựa trên đối tượng tài khoản vừa đọc đọc được; use case Truy vấn thông tin tài khoản sẽ hiển thị thông tin về tài khoản vừa đọc được, do đó, nó được thực hiện bởi một đối tượng giao diện.

Use case Gửi tiền



Use case Rút tiền



Việc thực hiện gửi tiền hoặc rút tiền sẽ do đối tượng tài khoản đảm nhận, đối tượng này được tạo ra do thừa hưởng từ use case Giao dịch. Sau đó, được thiết kế bằng một giao tác gồm hai

hoạt động: cập nhật lại số dư tài khoản và tạo một giao tác mới. Việc tạo một giao tác mới sẽ được thực hiện bởi method tạoGiaoTác() sẽ được thiết kế trong phần tiếp theo sau.

Sau đây là đoạn mã minh hoạ cho các method:

```
KháchHàng::-lấy_TàiKhoản(sốThẻ:String): TàiKhoản
        vNgânHàngDB: NgânHàngDB
        return vNgânHàngDB.đọaTàiKhoản(sốThẻ)

NgânHàngDB::+đọcTàiKhoản(vSốThẻ:String): TàiKhoản
        -- kết nối cơ sở dữ liệu ở đây
        Return
        SELECT * FROM TàiKhoản
        WHERE Số_Thẻ = vSốThẻ

TàiKhoản::+gửiTiền(sốTiền:float)
        vNgânHàngDB: NgânHàngDB
        NgânHàngDB.cậpNhậtTàiKhoản (sốTàiKhoản, sốDư + sốTiền)
        tạoGiaoTác("gửi", sốTiền, sốDư + sốTiền)

TàiKhoản::+rútTiền(sốTiền:float): String
        vNgânHàngDB: NgânHàngDB
        if sốDư < sốTiền
                return "Số tiền rút vượt quá số dư tài khoản"
        else
                NgânHàngDB.cậpNhậtTàiKhoản (sốTàiKhoản, sốDư + sốTiền)
                tạoGiaoTác("gửi", sốTiền, sốDư + sốTiền)
                return ""
        endif
```

NgânHàngDB::+cậpNhậtTàiKhoản(vSốTàiKhoản:String, vSốDư:float)

Việc sử dụng method này được minh hoạ theo sơ đồ của use case Gửi tiền và Rút tiền. Sau đây là đoạn mã minh hoạ:

```
NgânHàngDB::+cậpNhậtTàiKhoản(vSốTàiKhoản:String, vSốDư:float)
        UPDATE TàiKhoản
        SET sốDư = vSốDư
        WHERE Số_TK = vSốTàiKhoản
```

NgânHàngDB::+cậpNhậtGiaoDịch()

Để thiết kế chi tiết method này chúng ta tiếp tục với các use case Gửi tiền và Rút tiền qua việc mô tả chi tiết method tạoGiaoTác() của đối tượng tài khoản qua sự tương tác giữa tầng nghiệp vụ và tầng truy cập dữ liệu.

TàiKhoản::#tạoGiaoTác(loạiGD:String, sốTiền:float, sốDư: float)

vNgânHàngDB: NgânHàngDB

vGiaoDịch = tạoMới(GiaoDịch)

vGiaoDịch.gánThôngTin(Date(today), Time(now), loạiGD, sốTiền, sốDư, sốTàiKhoản)

vNgânHàngDB.cậpNhậtGiaoDịch(sốTàiKhoản, loạiGD, sốTiền, sốDư)

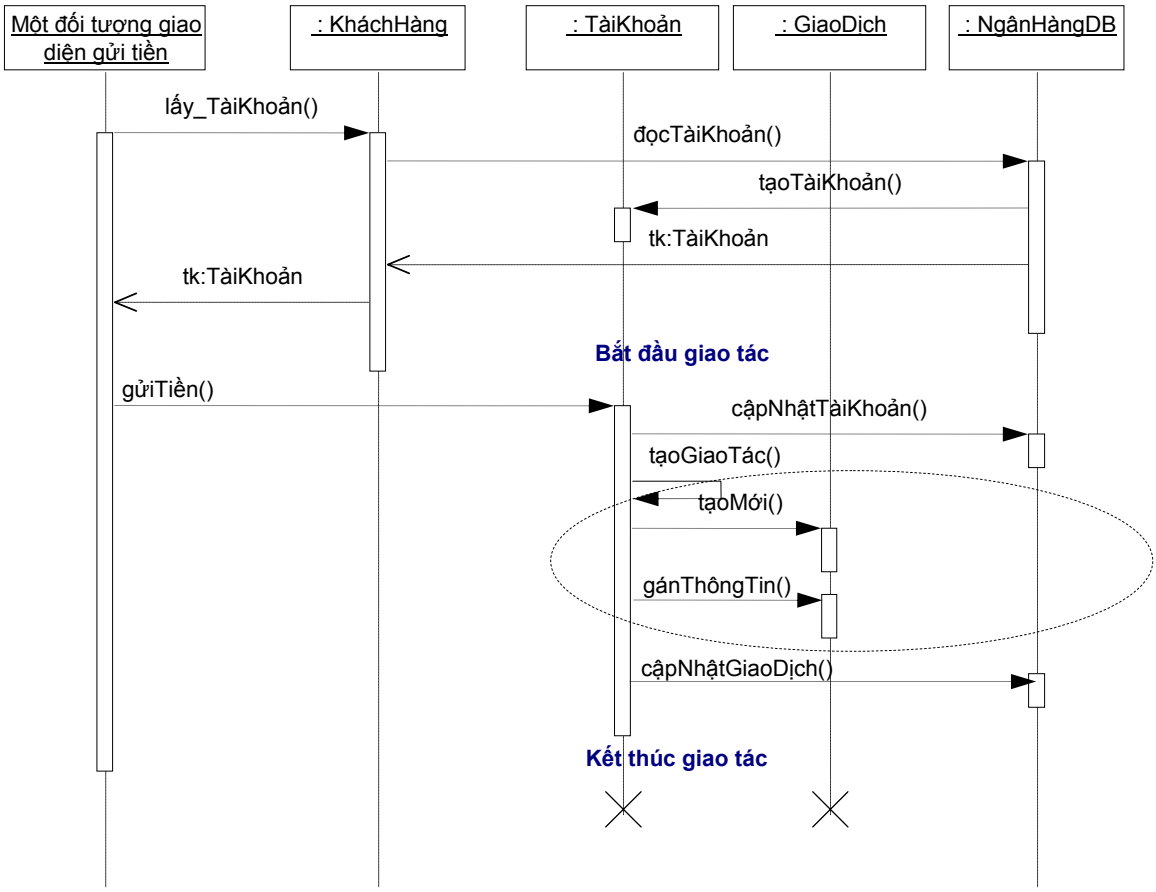
NgânHàngDB::+cậpNhậtGiaoDịch(vSốTàiKhoản:String, vLoạiGD:String, vSốDư:float)

--kết nối cơ sở dữ liệu ở đây

vGD_ID: String

vGD_ID = Tạo_GD_ID()

INSERT INTO GiaoDịch (GD_ID, Ngày_GD, Giờ_GD, Loại_GD, Số_Tiền, Số_Dư, Số_TK) VALUES (vGD_ID, Date(today), Time(now), vLoạiGD, vSốTiền, vSốDư, vSốTàiKhoản)



Xác định lớp tầng giao diện người dùng (user interface layer)

Bao gồm các đối tượng hệ thống mà người dùng sẽ tương tác và các đối tượng được dùng để quản lý hoặc điều khiển giao diện. Các class ở tầng này đảm nhận hai công việc chính:

- *Trả lời tương tác người dùng*: các đối tượng mức này phải được thiết kế để chuyển dịch những hành động của người dùng tới một tình huống xử lý phù hợp. Có thể là mở hoặc đóng một giao diện khác, hoặc gởi một thông điệp xuống tầng tác nghiệp hoặc khởi động một vài tiến trình ở tầng tác nghiệp

- *Hiển thị các đối tượng tác nghiệp*: tầng này phải trình bày một hình ảnh tốt nhất các đối tượng tác nghiệp tới người dùng trong một giao diện. Điều này có nghĩa là một hình ảnh trực quan thao tác được có thể bao gồm: các textbox, listbox, commbobox, page, form, menu, toolbar, ….

Tiến trình thiết kế tầng giao diện

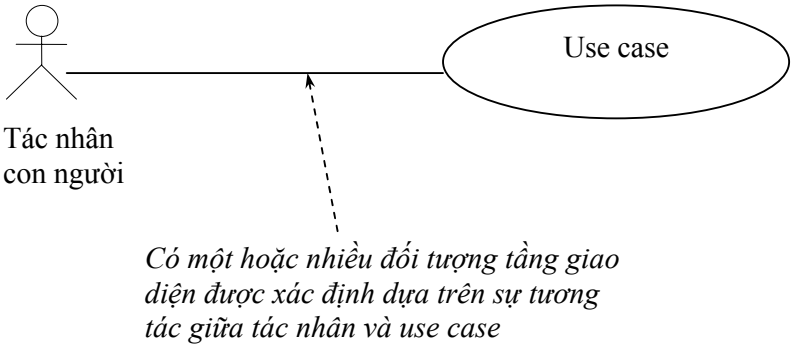
Một tiến trình thiết kế tầng giao diện bao gồm 4 bước sau:

1. Xác định các đối tượng ở tầng giao diện: đây là một hoạt động diễn ra luôn cả trong giai đoạn phân tích hệ thống. Mục tiêu chính là để xác định các lớp tương tác với các tác nhân con người thông qua việc phân tích các use case trong giai đoạn phân tích. Như đã mô tả trong các chương trước, mỗi use case bao gồm các tác nhân và các công việc mà tác nhân muốn hệ thống thực hiện. Do đó, nó mô tả một bức tranh đầy đủ về các yêu cầu giao diện của hệ thống. Trong giai đoạn này chúng ta cũng cần thiết tập trung vào cách thức cài đặt giao diện. Các sơ đồ tuần tự và hợp tác của use case cũng giúp chúng ta xác định chính xác yêu cầu về giao diện.
2. Xác định các lớp tầng giao diện
    a. Xác định các đối tượng tầng giao diện: xây dựng sơ đồ lớp mô tả các đối tượng giao diện
    b. Tạo bản mẫu giao diện (prototype): sau khi xác định mô hình thiết kế, chúng ta chuẩn bị cho một bản mẫu của giao diện. Thông thường các bản mẫu này đã được xác định trong những giai đoạn trước, nếu vậy thì chúng ta hãy cố gắng tích hợp với các đối tượng đã được xác định nhằm thống nhất các lớp giao diện trong sơ đồ lớp và các giao diện thiết kế.
    c. Xác định hành vi và thuộc tính cho các lớp giao diện: phân tích lại hành động của người dùng trong use case và tinh chế lại sơ đồ tuần tự để phát hiện các method cũng như xem xét lại các mối quan hệ của các đối tượng để xác định các thuộc tính (chủ yếu là các thuộc tính tham chiếu) cho lớp tầng giao diện.
3. Thử nghiệm giao diện.
4. Tinh chế và lặp lại các bước trên

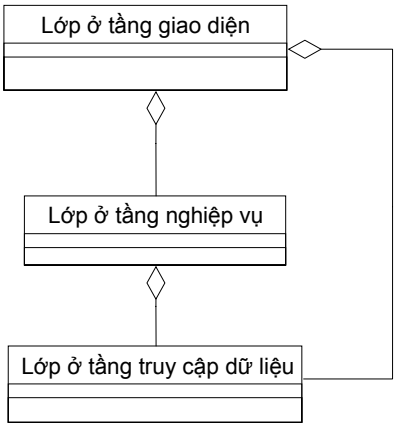
Xác định các lớp tầng giao diện qua việc phân tích use case

Trong một use case, đối tượng tầng giao diện xử lý tất cả trao đổi với tác nhân. Thực sự, các đối tượng này hoạt động như một vùng đệm giữa người dùng và phần còn lại của hệ thống là các đối tượng nghiệp vụ. Một đối tượng giao diện có thể tham gia vào nhiều use case. Bắt đầu từ use case, chúng ta xác định được các mục tiêu và nhiệm vụ của người dùng. Những người dùng khác nhau sẽ có những nhu cầu khác nhau trên giao diện, ví dụ, các người dùng chuyên nghiệp thì cần một giao diện có tính hiệu quả trong khi các người dùng bình thường thì cần có giao diện dễ sử dụng. Do đó, với các đối tượng người dùng khác nhau mà các giao diện được thiết kế sẽ khác nhau về mục tiêu, trách nhiệm, cách vận hành và hình thức trình bày. Việc xác định các lớp tầng giao diện gồm hai bước sau:

- Với mỗi lớp (tầng nghiệp vụ), nếu lớp đó có tương tác với một tác nhân con người trong một use case, chúng ta thức hiện như sau:
    - Xác định các đối tượng giao diện cho lớp đó, các trách nhiệm cũng như các yêu cầu của các đối tượng này. Việc này được thực hiện bằng cách phân tích lại sơ đồ tuần tự hoặc hợp tác của use case.



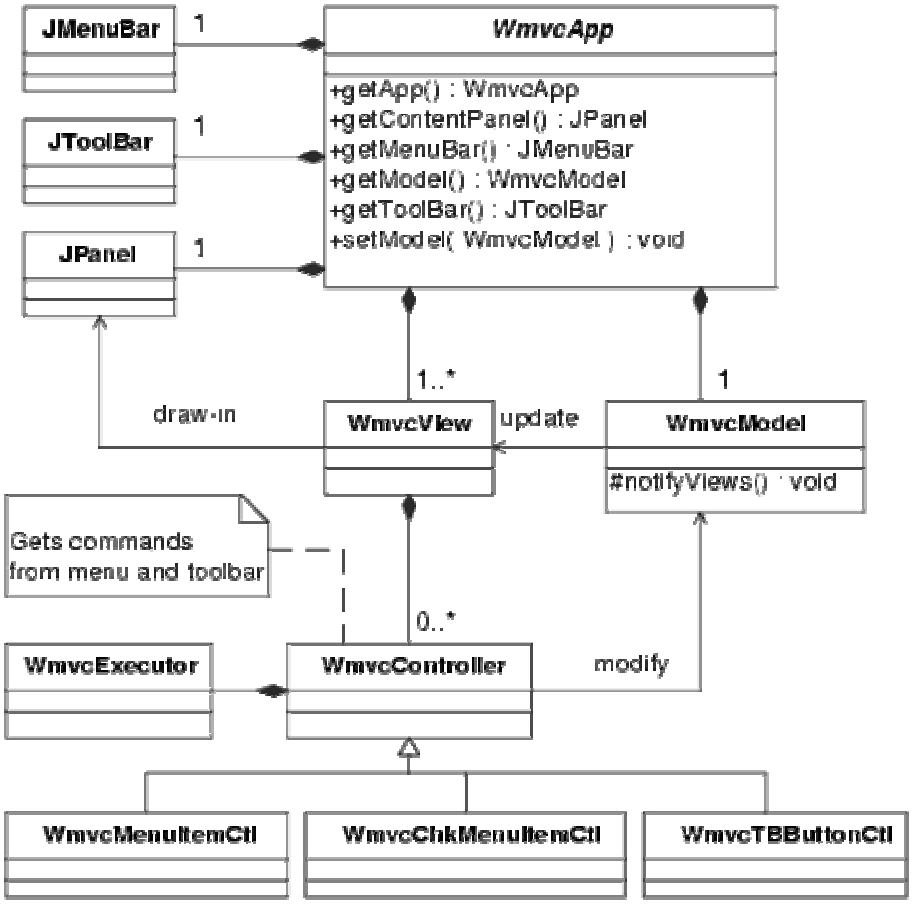
- Xác định sự liên kết giữa các đối tượng giao diện: các lớp giao diện cũng giống như các lớp khác, đều có mối quan hệ với các lớp ở tầng nghiệp vụ cùng tham gia trong một use case với nó. Các mối liên kết này cũng được xác định tương tự như của các lớp trong tầng nghiệp vụ. Sự liên kết này thường theo sơ đồ dưới đây.



- Lặp lại các bước trên và tinh chế

Ưu điểm của của việc sử dụng use case để xác định các đối tượng ở tầng giao diện là nó tập trung vào người dùng và đưa người dùng vào như một phần của kế hoạch thiết kế nhằm tìm ra một giao diện tốt nhất cho người dùng. Khi các đối tượng này đã được xác định, chúng ta phải xác định các thành phần cơ bản hoặc các đối tượng được dùng trong các công việc cũng như là các hành vi và các đặc điểm tạo ra sự khác biệt của mỗi loại đối tượng, bao gồm luôn cả mối quan hệ giữa các đối tượng và giữa đối tượng và người dùng.

Tuy nhiên, trong thiết kế giao diện khi chúng ta đã xác định môi trường phát triển thì chúng ta cũng nên tìm hiểu các khung mẫu (framework) cũng như các thư viện mà môi trường đó hỗ trợ để phát huy việc tái sử dụng trong thiết kế giao diện và tận dụng tối đa các hỗ trợ từ môi trường.



Một dạng khung mẫu Wmvc (Model – View - Controller) được hỗ trợ trong môi trường Java áp dụng cho các thiết kế giao diện ứng dụng với Java

Xây dựng bản mẫu (prototype) cho giao diện

Việc xây dựng bản mẫu giao diện thường được thiết kế trong giai đoạn xác định yêu cầu và phân tích hệ thống. Công việc này được thực hiện sử dụng một công cụ gọi là CASE Tool hoặc các công cụ phát triển. Bao gồm ba bước sau:

- Tạo các đối tượng giao diện (như là các button, các vùng nhập liệu,…)
- Liên kết và gán các hành vi hoặc hành động thích hợp tới các đối tượng giao diện này và các sự kiện của nó.
- Thử nghiệm và lặp lại bước một

Thiết kế tầng giao diện cho hệ thống ATM

Xác định các đối tượng ở tầng giao diện, các yêu cầu và trách nhiệm của nó

Đối với mỗi lớp ở tầng nghiệp vụ: NgânHàng, MáyATM, KháchHàng, TàiKhoản, GiaoDịch, GiaoDịchGửi, GiaoDịchRút chúng ta xác định các lớp giao diện của nó bằng việc quan sát các sơ đồ tuần tự và hợp tác của các use case: Đăng nhập, Đăng nhập không hợp lệ, Giao dịch, Rút tiền, Gửi tiền, Truy vấn thông tin tài khoản, Khởi động hệ thống, Đóng hệ thống. Chúng ta xác định được các lớp tầng giao diện sau đây:

KháchHàngGD: biểu diễn giao diện tương tác giữa khách hàng và use case Đăng nhập, Đăng nhập không hợp lệ

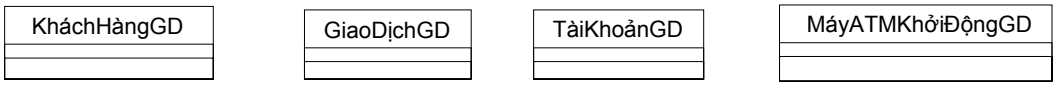
GiaoDịchRútGD: biểu diễn tương tác giữa khách hàng và use case Rút tiền

GiaoDịchGửiGD: biểu diễn tương tác giữa khách hàng và use case Gửi tiền

Tài khoảnGD: biểu diễn tương tác giữa khách hàng và use case Truy vấn thông tin tài khoản

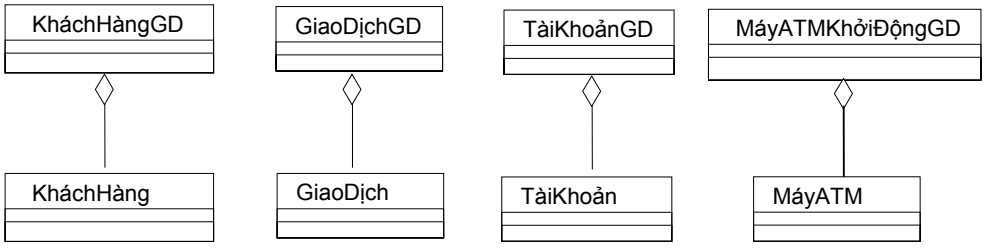
MáyATMKhởiĐộngGD: biểu diễn tương tác giữa nhân viên vận hành và use case Khởi động hệ thống

Các đối tượng giao diện GiaoDịchGửiGD, GiaoDịchRútGD có một hình thức trình bày chung của giao dịch và chỉ thay đổi loại giao dịch. Do đó, chúng ta gom lại thành một lớp và gọi là: GiaoDịchGD.



Xác định sự liên kết các lớp của tầng giao diện với các lớp của tầng nghiệp vụ

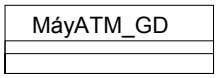
Mối kết hợp được xác lập là mối kết hợp dạng aggregation như sau: với mỗi lớp giao diện chúng ta tạo liên kết aggregation tới lớp tương ứng trong tầng nghiệp vụ. Mối liên kết này cho biết trong các thể hiện của lớp tầng giao diện sẽ sử dụng đối tượng ở tầng nghiệp vụ như là một thành phần để thực hiện gởi thông điệp.



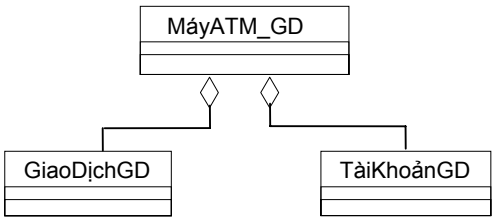
Xác định các đối tượng giao diện điều khiển như là: toolbar, menu, form điều khiển,....

Ngoài các lớp được xác định dựa vào use case, chúng ta xác định thêm các đối tượng giao diện có nhiệm vụ điều khiển chính, giao diện chính, thực đơn, tool bar,…

Với hệ thống ATM chúng ta có thêm một đối tượng giao diện điều khiển chính hoạt động giao diện của máy ATM gọi là MáyATM_GD.



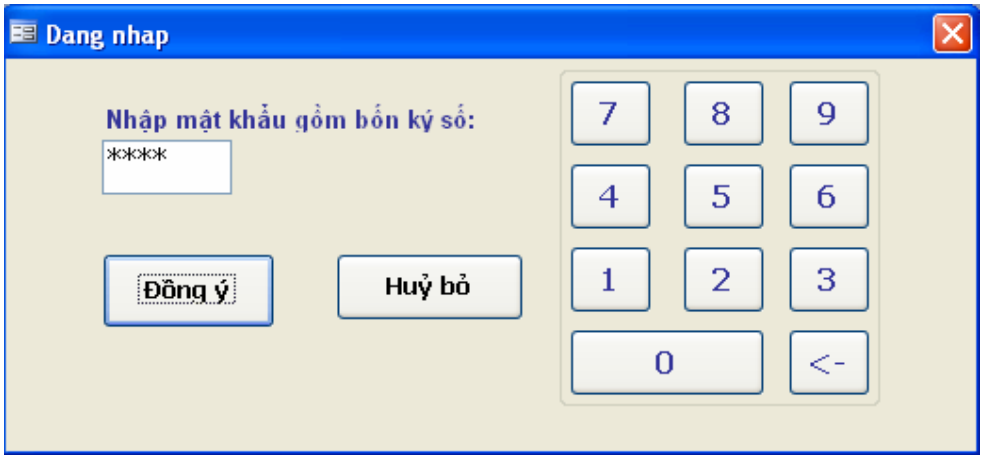
Vì đối tượng của lớp MáyATM_GD sẽ gởi thông điệp đến tất cả các đối tượng giao diện rút, gửi và truy vấn thông tin nhằm điều khiển các giao diện này. Do đó, trước khi gởi thông điệp thì đối tượng lớp MáyATM_GD sẽ phải tạo ra các đối tượng kia như là một thành phần của nó. Chính vì vậy mà chúng ta sẽ xác lập mối kết hợp aggregation từ lớp MáyATM_GD đến các lớp: GiaoDichGD, TaiKhoanGD



Thiết kế giao diện mẫu (prototype)

Chúng ta lần lượt thiết kế bản mẫu cho các đối tượng giao diện:

KháchHàngGD



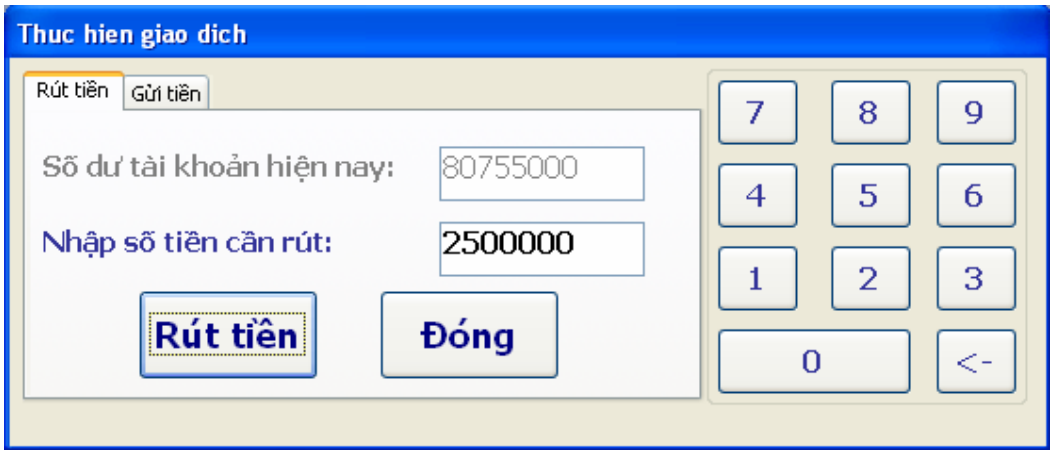
Giao diện đăng nhập tài khoản cho phép khách hàng chọn bốn ký số bằng cách nhấn vào các nút số được thiết kế ngay trên màn hình. Khách hàng có thể chọn đồng ý hoặc huỷ bỏ đăng nhập.

MáyATM_GD

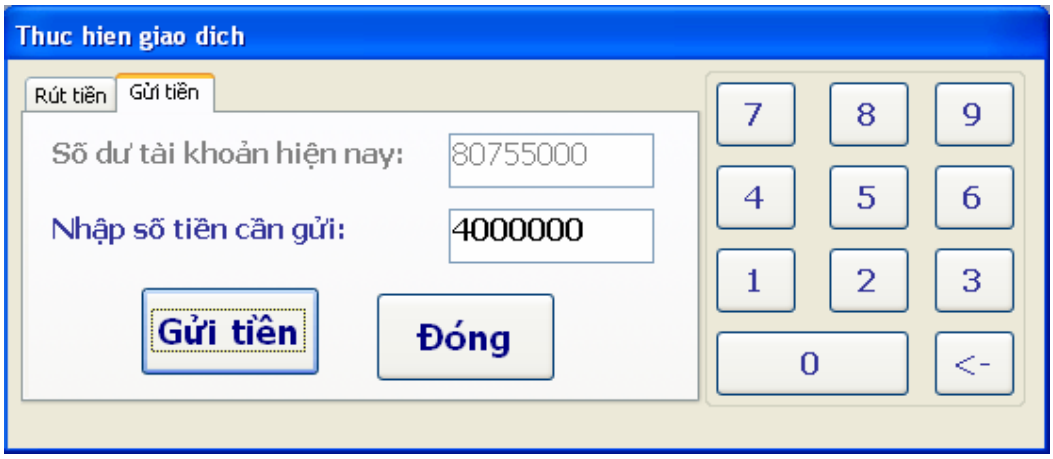
Sau khi đăng nhập thành công, giao diện chính điều khiển của ATM sẽ xuất hiện. Giao diện này sẽ hiển thị các nút cho phép người dùng chọn để thực hiện các dịch vụ tương ứng như là: gửi tiền, rút tiền, và xem thông tin về tài khoản.

GiaoDịchGD

Giao diện rút tiền

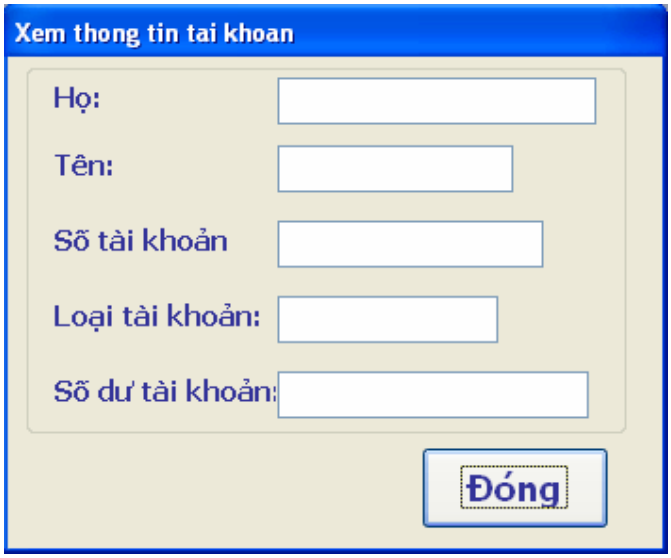


Giao diện gửi tiền



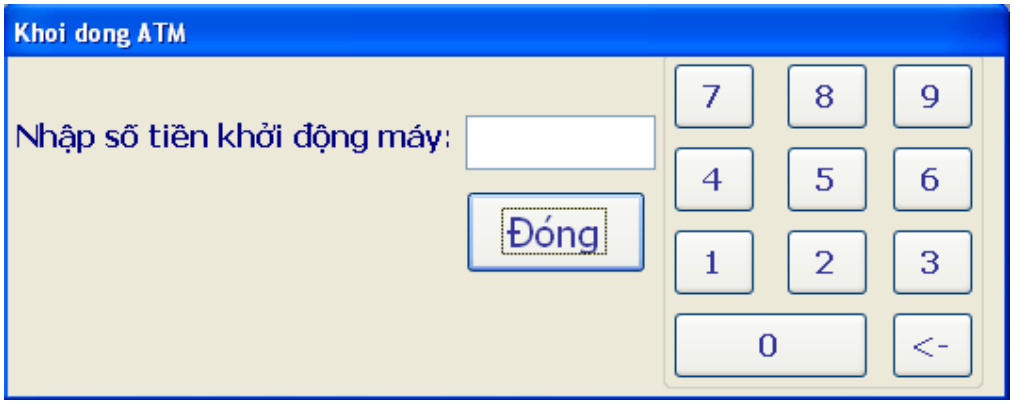
Giao diện thực hiện giao dịch được thiết kế gồm hai thẻ một thẻ cho giao dịch rút và một thẻ cho giao dịch gửi. Tập các số giả lập bàn phím cho phép khách hàng nhập số liệu trực tiếp từ màn hình. Giao dịch sẽ được thực hiện khi khách hàng chọn nút rút tiền hoặc gửi tiền. Khách hàng có thể chọn huỷ bỏ giao dịch bằng cách nhấn nút đóng.

TàiKhoảnGD



Giao diện này cho phép hiển thị thông tin về tài khoản của khách hàng bao gồm: họ, tên, số tài khoản, loại tài khoản, và số dư tài khoản

MáyATMKhởiĐộngGD



Sau khi máy đã được bật lên, giao diện này sẽ hiển thị cho phép nhân viên vận hành nhập vào số tiền khởi động cho máy.

Xác định các method

Mục tiêu của các đối tượng giao diện là cho phép người dùng thao tác với hệ thống nhằm phục vụ cho công việc nghiệp vụ như là: nhấn một nút, nhập vào một ký tự,…. Các thao tác này sẽ được chuyển tới các đối tượng tầng nghiệp vụ để xử lý. Khi hoàn thành xử lý, giao diện sẽ được cập nhật lại thông tin nhằm hiển thị các thông tin mới hoặc mở một giao diện mới,…. Xác định hành vi cho đối tượng giao diện bằng cách xác định các sự kiện mà của hệ thống phải đáp ứng tương ứng tới các thao tác người dùng trên giao diện và các hành động khi sự kiện xảy ra. Một hành động được xem như một hành vi và được hình thành bởi sự kết hợp một đối tượng với một thông điệp gởi tới đối tượng đó. Xem xét lại từng đối tượng giao diện theo từng use use case của hệ thống ATM chúng ta lần lượt xác định các method cho các lớp giao diện .[2]

KháchHàngGD - Use case Đăng nhập

Khi khách hàng đưa thẻ ATM vào máy các sự kiện và hành động:

- Đưa thẻ vào máy -> hiển thị giao diện đăng nhập (KháchHàngGD)
- Khách hàng chọn đồng ý -> kiểm tra mật khẩu (KháchHàng)
  -> hiển thị giao diện điều khiển chính (MáyATM_GD)
  -> thông báo nếu đăng nhập không thành công (KháchHàngGD)
  -> đóng giao diện đăng nhập (KháchHàngGD)
- Khách hàng chọn huỷ bỏ -> đóng giao diện đăng nhập (KháchHàngGD)

Chúng ta xác định được các method:

KháchHàngGD::+hiểnThị()

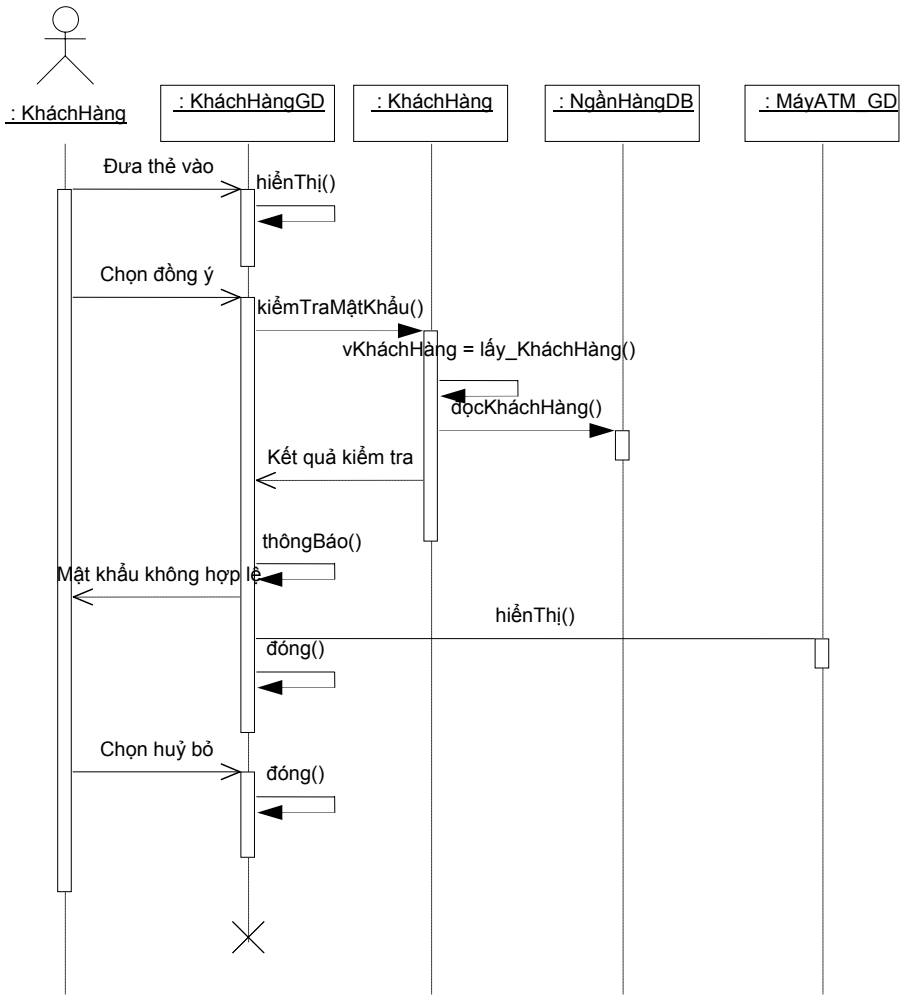
KháchHàngGD::-thôngBáo(thôngBáo:String)

KháchHàngGD::+đóng()

MáyATM_GD::+hiểnThị()

Sơ đồ tuần tự tinh chế tầng giao diện cho use case Đăng nhập

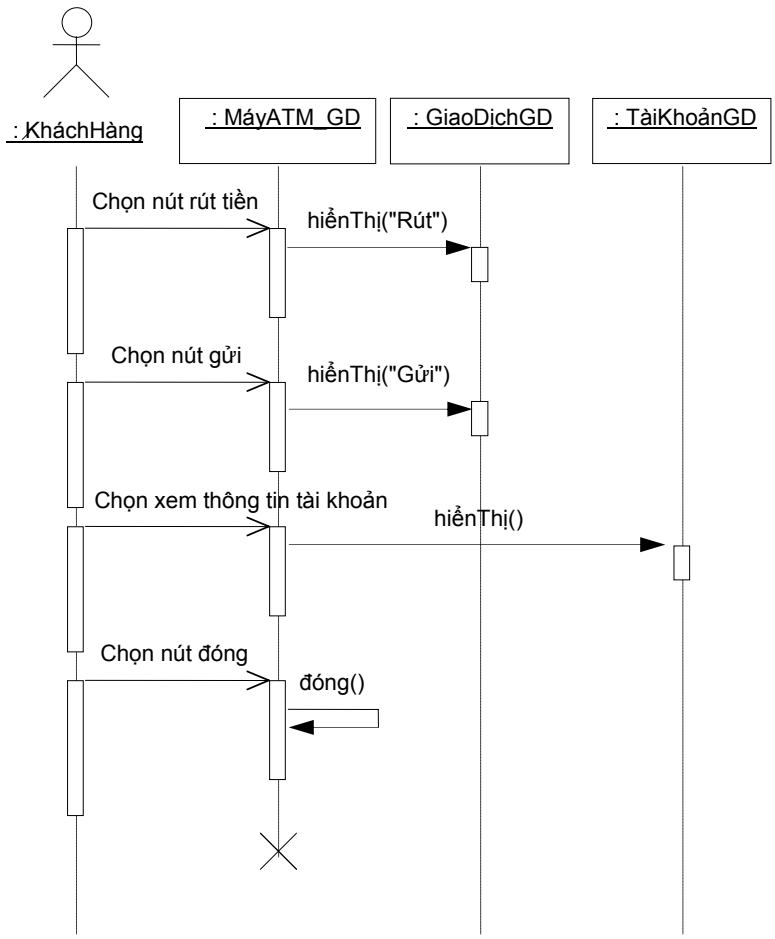
[2] Trong quá trình xác định các method cho tầng giao diện có thể phát sinh thêm các method ở các tầng nghiệp vụ hoặc truy cập dữ liệu và chúng ta phải quay lại để thiết kế thêm các method mới phát sinh này



MáyATM_GD

Khi giao diện chính của máy được hiển thị các tương tác của khách hàng làm phát sinh các sự kiện và các hành động:

- Chọn nút rút tiền -> hiển thị giao diện rút tiền (GiaoDịchGD)
- Chọn nút gửi tiền -> hiển thị giao diện gửi tiền (GiaoDịchGD)
- Chọn nút xem tài khoản (TàiKhoảnGD) -> hiển thị giao diện xem thông tin tài khoản
- Chọn nút thoát -> đóng giao diện chính (MáyATM_GD)



Chúng ta xác định được các method

GiaoDịchGD::+hiểnThị(loạiGD:String)

TàiKhoản::+hiểnThị()

MáyATM_GD::+đóng()

GiaoDịchGD - Use case Rút tiền

Khi khách hàng chọn dịch vụ rút tiền từ giao diện chính các sự kiện và hành động:

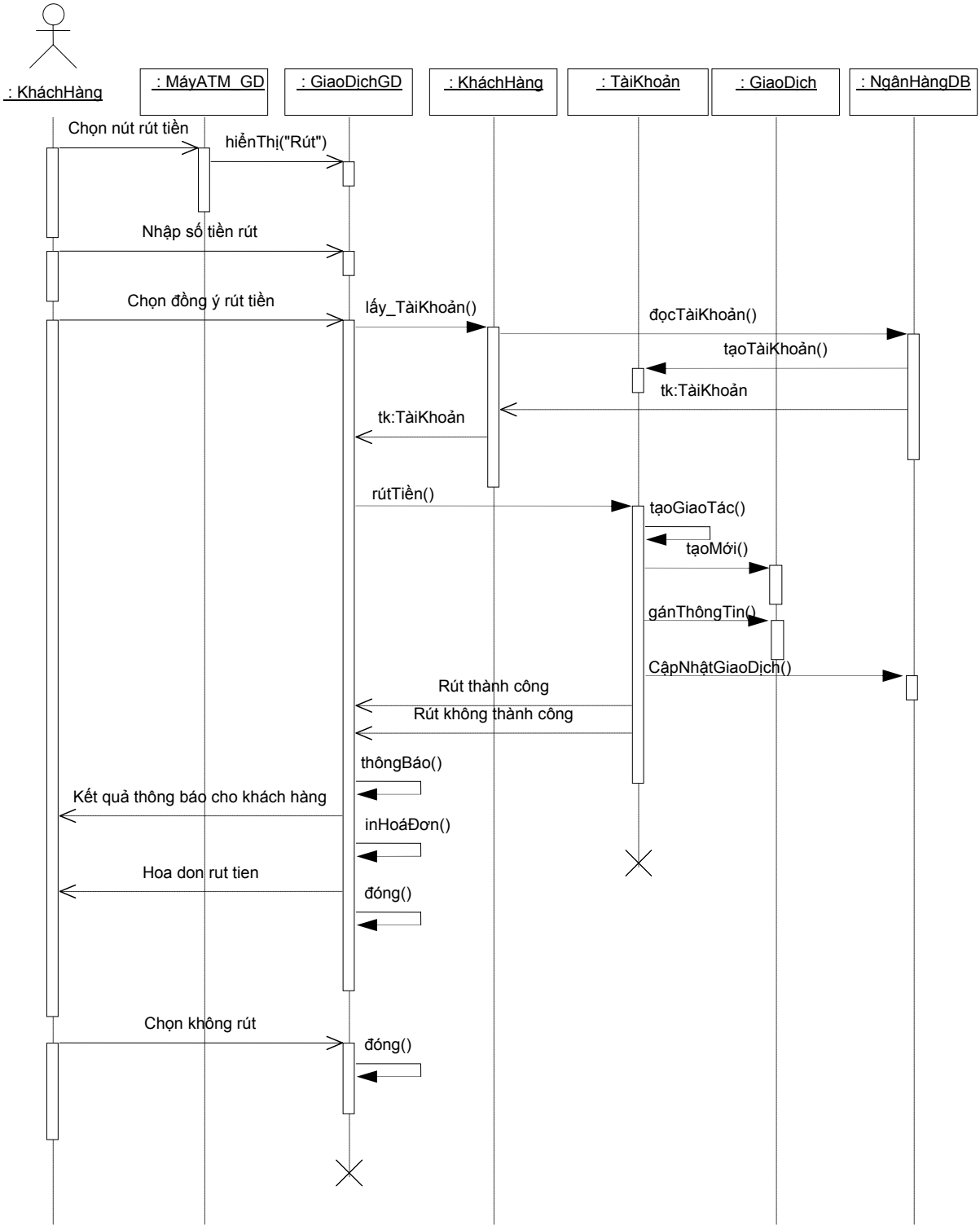
- Chọn rút tiền -> hiển thị giao diện rút tiền (GiaoDịchGD)
- Khách hàng chọn rút tiền -> thực hiện rút tiền (TàiKhoản)
  -> thông báo kết quả (GiaoDịchGD)
  -> in hoá đơn rút (GiaoDịchGD)
  -> đóng giao diện rút tiền (GiaoDịchGD)
- Khách hàng chọn đóng -> đóng giao diện rút tiền (GiaoDịchGD)

Chúng ta xác định được các method:

GiaoDịchGD::-thôngBáo(thôngBáo:String)

GiaoDịchGD::-inHoáĐơn()

GiaoDịchGD::+đóng()



GiaoDịchGD - Use case Gửi tiền

Khi khách hàng chọn dịch vụ gửi tiền từ giao diện chính các sự kiện và hành động:

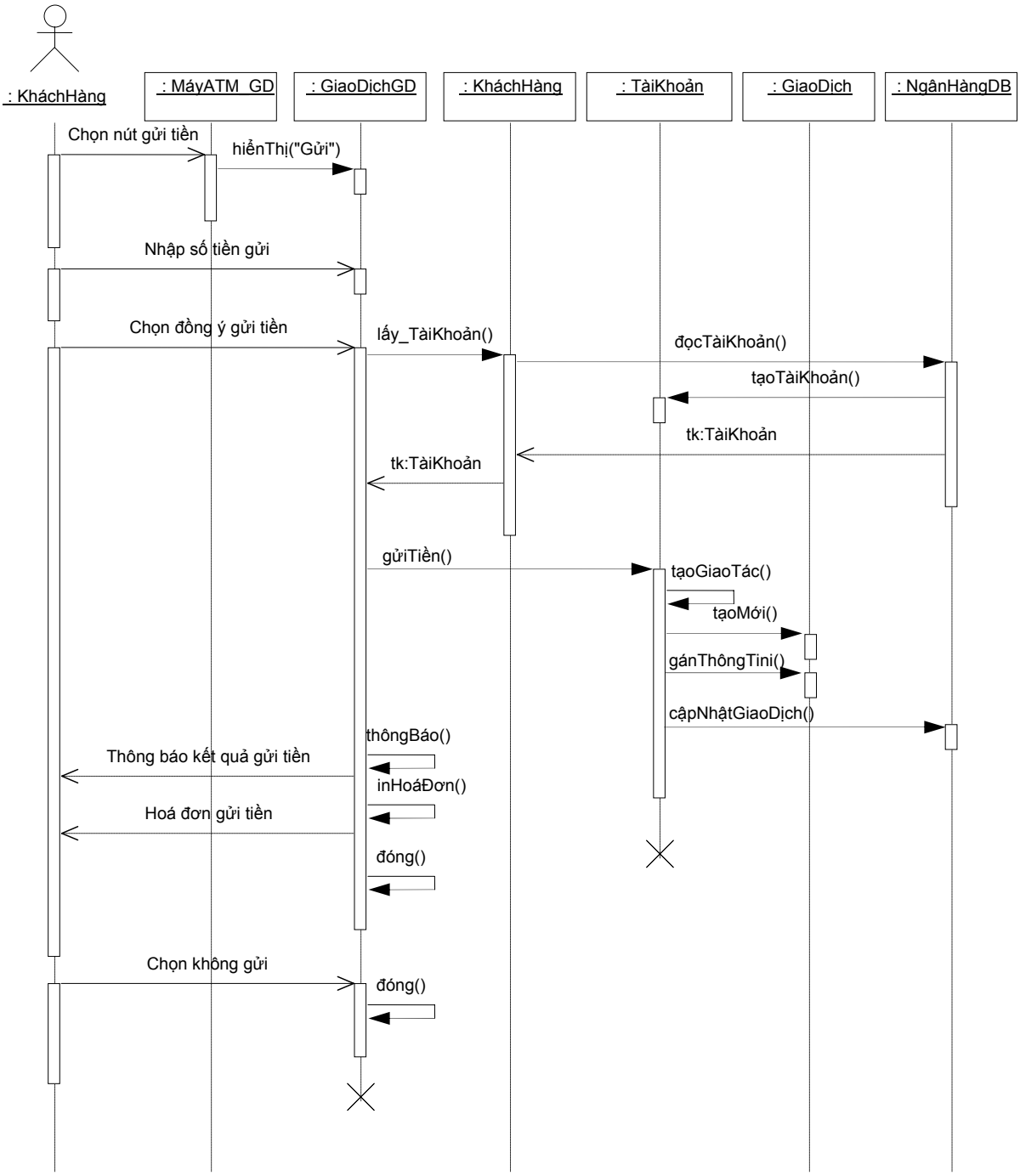
- Chọn gửi tiền -> hiển thị giao diện gửi tiền (GiaoDịchGD)
- Khách hàng chọn gửi tiền -> thực hiện gửi tiền (TàiKhoản)
  -> thông báo kết quả (GiaoDịchGD)

-> in hoá đơn gửi (GiaoDịchGD)

-> đóng giao diện gửi tiền (GiaoDịchGD)

- Khách hàng chọn đóng -> đóng giao diện gửi tiền (GiaoDịchGD)

Các method xác định trong use case này giống như của use case Rút tiền



Trong sơ đồ tuần tự trên cho rút tiền và gưi tiền, chúng ta quan sát ba đối tượng: Kháchhàng (tác nhân), và hai đối tượng giao diện MáyATM_GD, GiaoDịchGD. Bắt đầu các dòng từ tác nhân khách hàng mô tả các thao tác khách hàng trên giao diện và trả lời của hệ thống từ giao diện. Từ các dòng này, các đối tượng giao diện sẽ tương tác với các đối tượng ở tầng nghiệp vụ bằng các thông điệp nhằm thực hiện tác vụ của hệ thống.

TàiKhoảnGD - Use case Truy vấn thông tin tài khoản[3]

Khi khách hàng chọn truy vấn thông tin tài khoản từ giao diện chính các sự kiện và hành động:

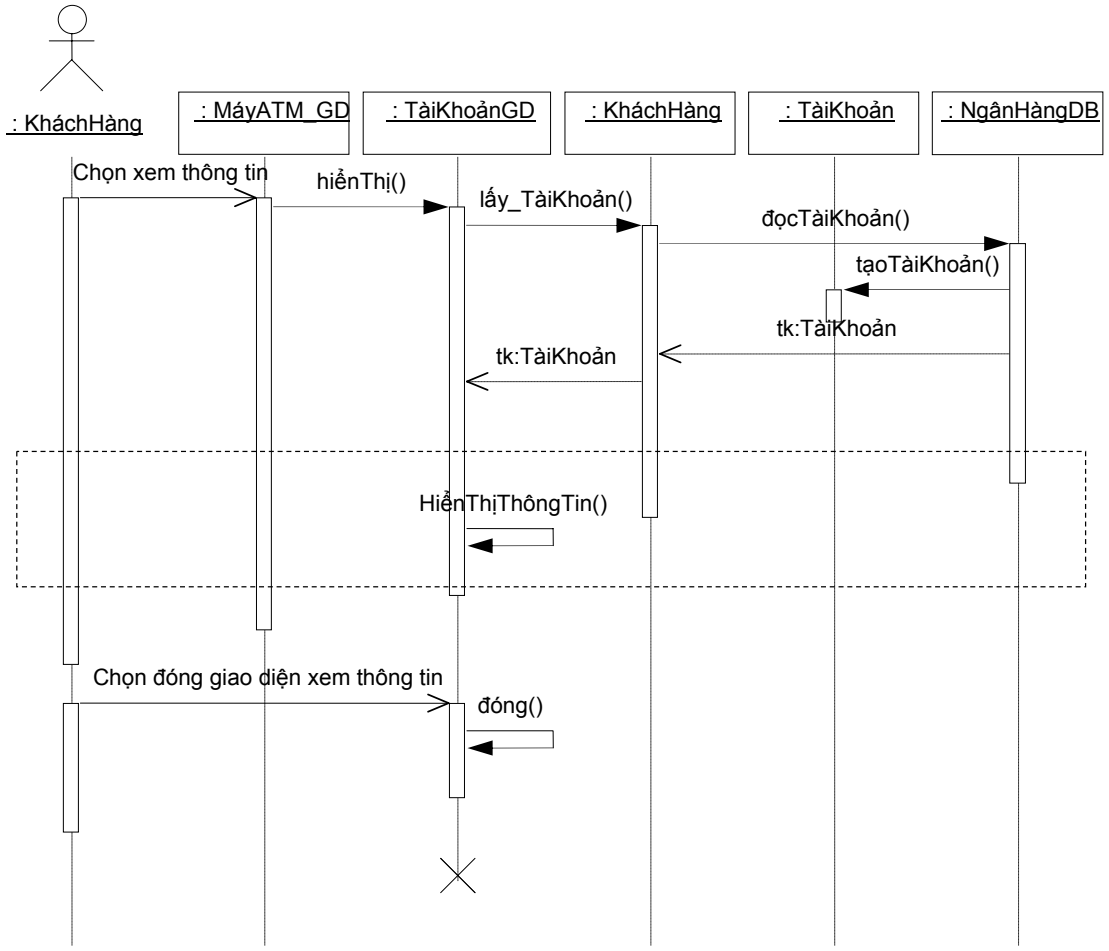
- Chọn xem thông tin tài khoản -> hiển thị giao diện truy vấn (TàiKhoảnGD)
  -> đọc thông tin tài khoản (KháchHàng)
  -> hiển thị thông tin tài khoản (TàiKhoảnGD)
- Khách hàng chọn đóng -> đóng giao diện truy vấn (TàiKhoảnGD)

Chúng ta xác định được các method:

TàiKhoảnGD::+hiểnThị()

TàiKhoảnGD::-hiểnThịThôngTin(tk:TaiKhoan)

TàiKhoảnGD::+đóng()



MáyATMKhởiĐộngGD - Use case Khởi động hệ thống

Khi máy được bật công tắc khởi động các sự kiện và hành động:

- Khởi động máy hoàn thành -> hiện thị giao diện khởi động máy (MáyATMKhởiĐộngGD)

[3] Chú ý rằng: use case "Truy vấn thông tin tài khoản" là một use case mở rộng của use case "Giao dịch", do vậy trong thiết kế use case "Truy vấn thông tin tài khoản" chúng ta thừa kế các hoạt động của use case "Giao dịch".

- Nhân viên chọn đóng -> cập nhật số tiền cho hiện hành cho máy (MáyATM)
  -> thực hiện kết nối tới mạng ngân hàng (NgânHàng)
  -> đóng giao diện khởi động (MáyATMKhởiĐộngGD)

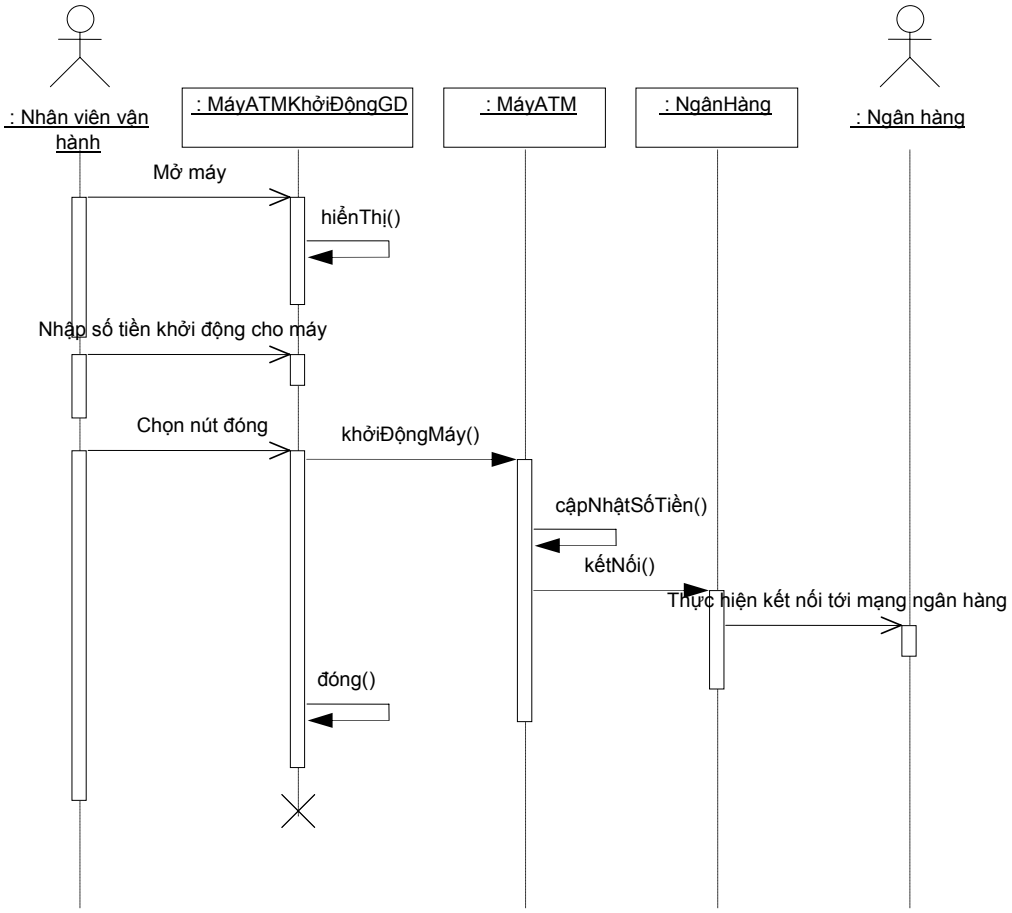
Chúng ta xác định được các method:

MáyATMKhởiĐộngGD::+hiểnThị()

MáyATM::+cậpNhậtSốTiền(sốTiền:float)

NgânHàng::+kếtNối()

MáyATMKhởiĐộngGD::+đóng()



Use case Đóng máy

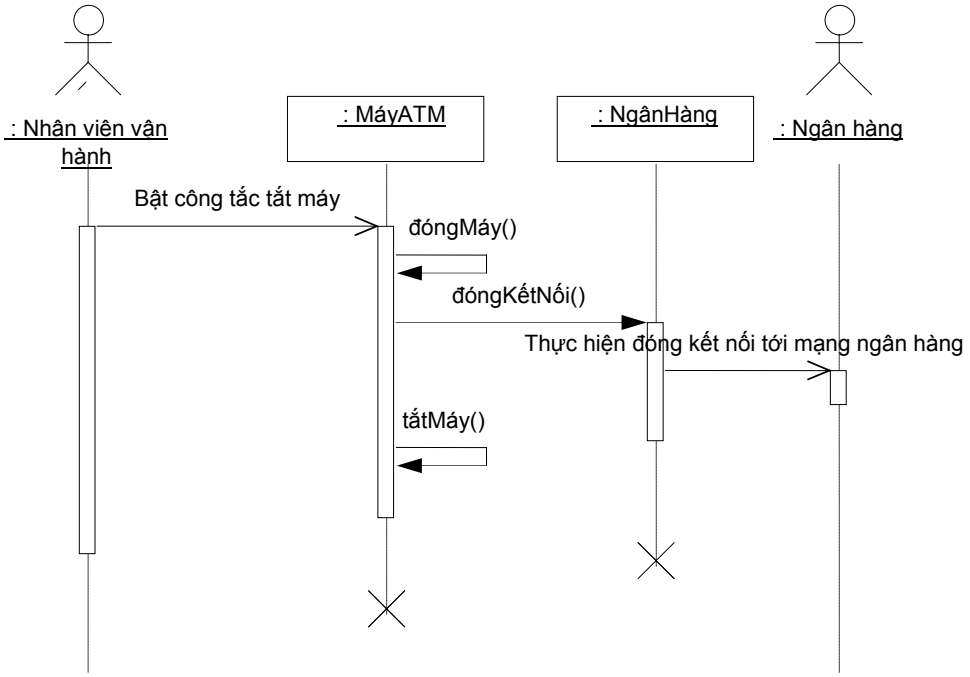
Khi nhân viên bật công tắc tắt máy, các sự kiện và hành động:

- Trước khi tắt máy -> thực hiện đóng kết nối tới mạng ngân hàng (NgânHàng)
  -> tắt máy (MáyATM)

Chúng ta xác định được các method

NgânHàng::+đóngKếtNối()

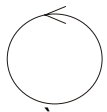
MáyATM::-tắtMáy()



Một giải pháp khác nhằm quản lý việc điều khiển các lớp nghiệp vụ để đáp ứng cho các sự kiện ở tầng giao diện là dùng đối tượng điều khiển. Các thông điệp từ tầng giao diện sẽ được tiếp nhận bởi đối tượng điều khiển và đối tượng này sẽ điều khiển các hoạt động của các đối tượng nghiệp vụ nhằm thực thi cho thông điệp đó. Như vậy đối tượng giao diện cũng có thể hiểu như là một đối tượng bao bọc tầng nghiệp vụ từ các đối tượng tầng giao diện.

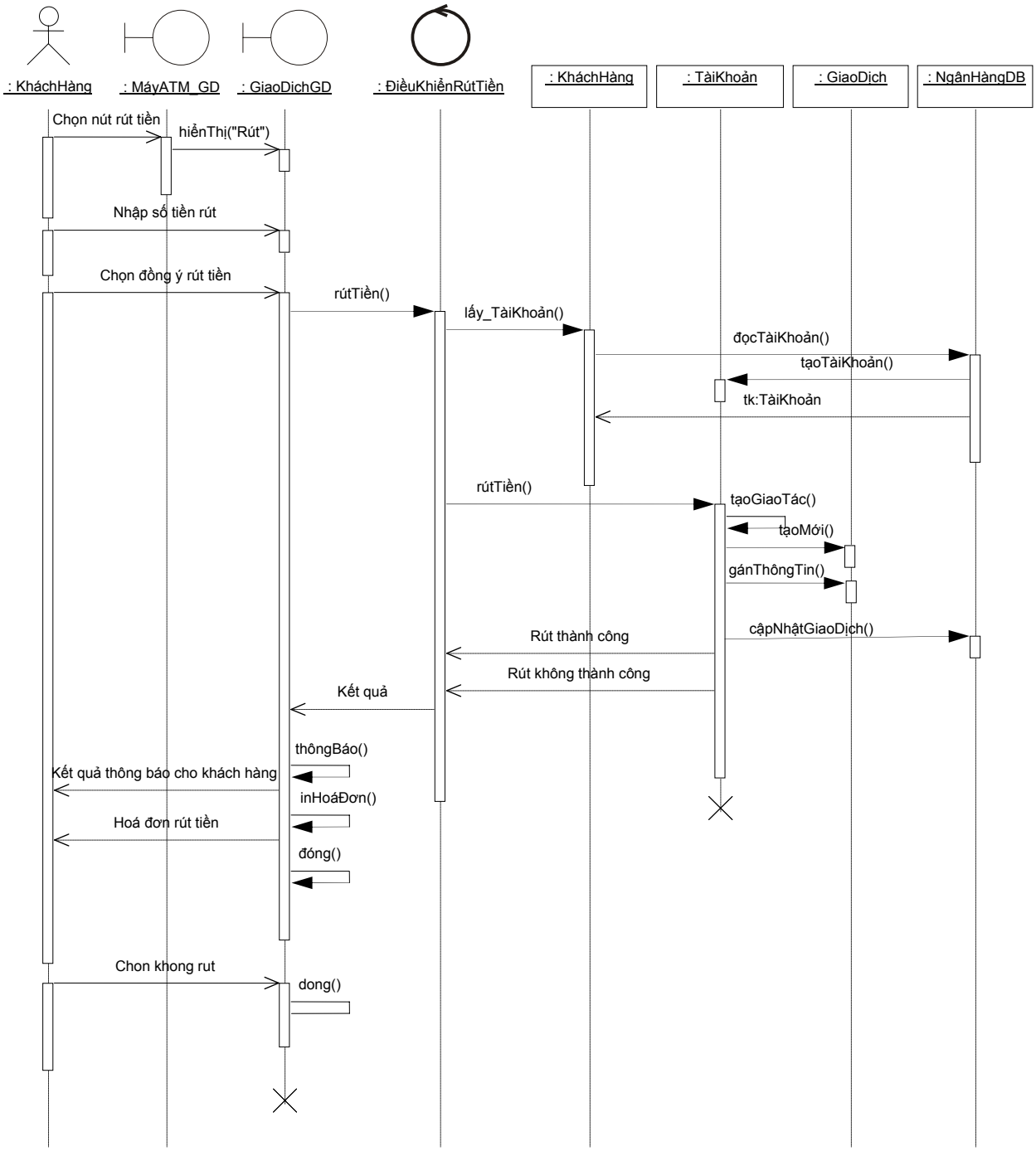
Về quan niệm tổng quát, một đối tượng điều khiển có thể đảm nhận nhiều use case hoặc một use case có thể có nhiều đối tượng điều khiển. Tuy nhiên, không phải tất cả use case đều phải có đối tượng điều khiển, bởi vì ý nghĩa của đối tượng điều khiển là tạo ra sự phối hợp, do đó, trong trường hợp use case chỉ có một lớp thì ý nghĩa phối hợp không còn.

Trong UML lớp điều khiển là một stereotype và có ký hiệu như sau:



Lớp điều khiển

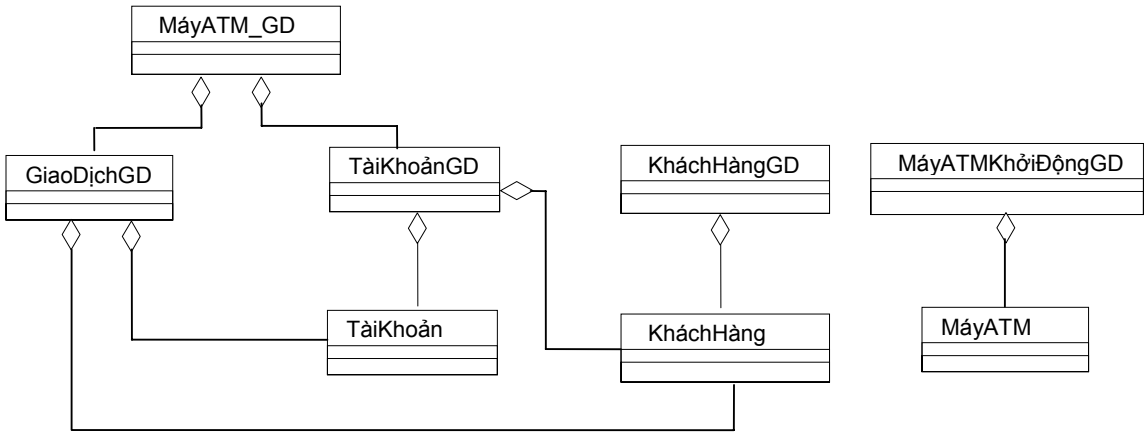
Sơ đồ tuần tự cho use case Rút tiền có đối tượng điều khiển:



Các lớp tầng giao diện được có thể được chọn một kiểu stereotype phù hợp: boundary, form, interface, page, webpage,… Ví dụ, trong sơ đồ use case Rút tiền trên chúng ta chọn stereotype cho các lớp giao diện là boundary. Quan sát sơ đồ trên chúng ta thấy các đối tượng giao diện sẽ tượng tác (rút tiền) với các đối tượng nghiệp vụ qua một đối tượng điều khiển, và đối tượng này sẽ điều phối các hoạt động của các đối tượng nghiệp vụ (KháchHàng, TàiKhoản, GiaoDịch) để thực hiện việc rút tiền thay vì trong các sơ đồ trước đó, việc điều phối này do đối tượng giao diên đảm nhận.

Xác định thuộc tính các lớp tầng giao diện

Các thuộc tính được xác định cho các lớp tầng giao diện chủ yếu là các thuộc tính mô tả tham chiếu. Một lần nữa, dựa vào các sơ đồ tuần tự chúng ta tinh chế lại mối kết hợp giữa các lớp ở tầng giao diện và các lớp tầng nghiệp vụ:



Các thuộc tính tham chiếu của các lớp lần lượt là:

Lớp MáyATM_GD

-giaoDịchGD:GiaoDịchGD

-tàiKhoảnGD:TàiKhoảnGD

Lớp KháchHàngGD

-khácHàng:KháchHàng

Lớp GiaoDịchGD

-tàiKhoản:TàiKhoản

-khácHàng:KháchHàng

Lớp TàiKhoảnGD

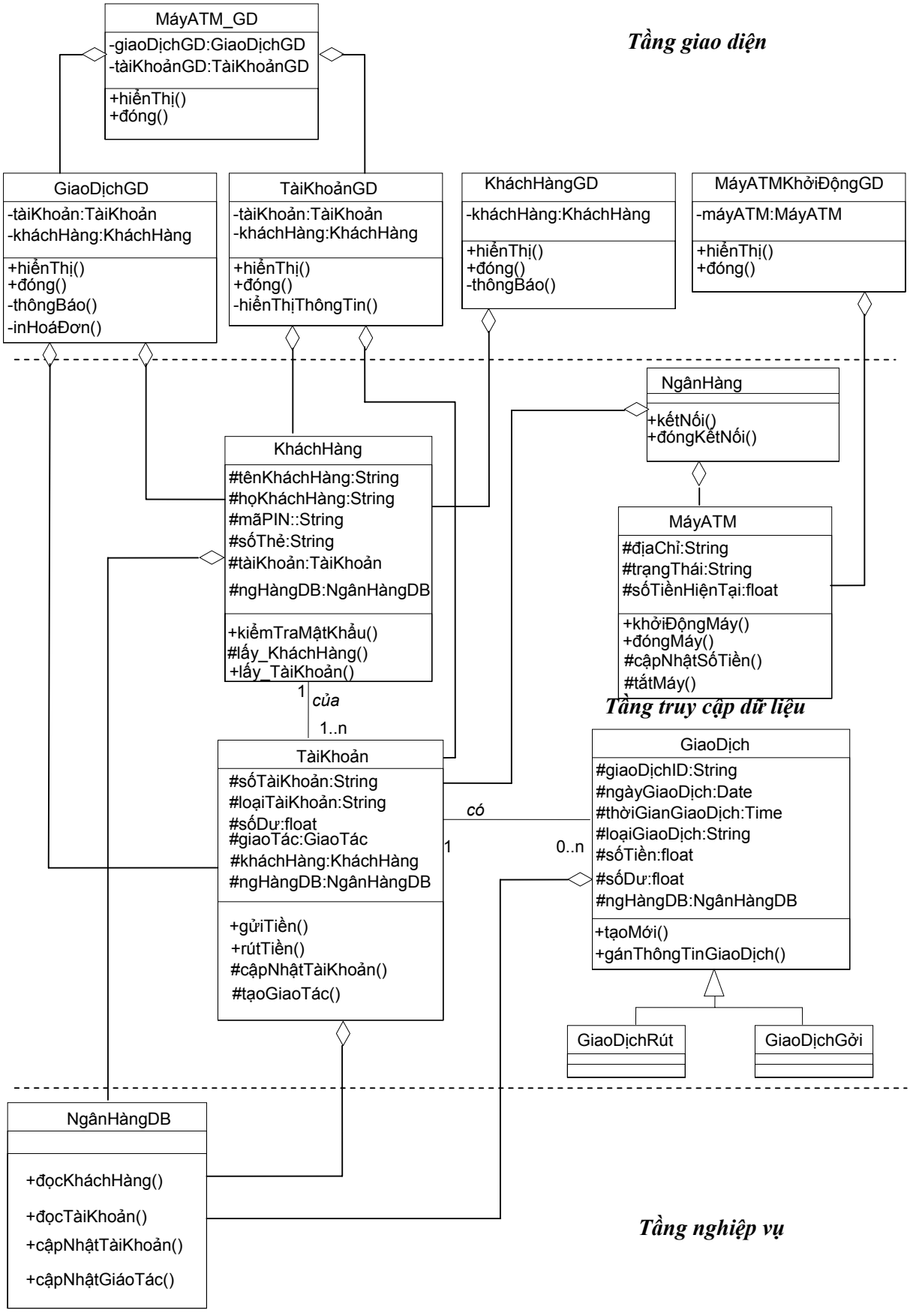
-tàiKhoản:TàiKhoản

-khácHàng:KháchHàng

Lớp MáyATMKhởiĐộngGD

-máyATM:MáyATM

Sơ đồ lớp đầy đủ ba tầng của hệ thống ATM:


MáyATM_GD
-giaoDịchGD:GiaoDịchGD
-tàiKhoảnGD:TàiKhoảnGD
+hiểnThị()
+đóng()
Tầng giao diện
GiaoDịchGD
-tàiKhoản:TàiKhoản
-kháchHàng:KháchHàng
+hiểnThị()
+đóng()
-thôngBáo()
-inHoáĐơn()
TàiKhoảnGD
-tàiKhoản:TàiKhoản
-kháchHàng:KháchHàng
+hiểnThị()
+đóng()
-hiểnThịThôngTin()
KháchHàngGD
-kháchHàng:KháchHàng
+hiểnThị()
+đóng()
-thôngBáo()
MáyATMKhởiĐộngGD
-máyATM:MáyATM
+hiểnThị()
+đóng()
NgânHàng
+kếtNối()
+đóngKếtNối()
KháchHàng
#tênKháchHàng:String
#họKháchHàng:String
#mãPIN::String
#sốThẻ:String
#tàiKhoản:TàiKhoản
#ngHàngDB:NgânHàngDB
+kiểmTraMậtKhẩu()
#lấy_KháchHàng()
+lấy_TàiKhoản()
MáyATM
#địaChỉ:String
#trạngThái:String
#sốTiềnHiệnTại:float
+khởiĐộngMáy()
+đóngMáy()
#cậpNhậtSốTiền()
#tắtMáy()
Tầng truy cập dữ liệu
1
của
1..n
TàiKhoản
#sốTàiKhoản:String
#loạiTàiKhoản:String
#sốDư:float
#giaoTác:GiaoTác
#kháchHàng:KháchHàng
#ngHàngDB:NgânHàngDB
+gửiTiền()
+rútTiền()
#cậpNhậtTàiKhoản()
#tạoGiaoTác()
có
1
0..n
GiaoDịch
#giaoDịchID:String
#ngàyGiaoDịch:Date
#thờiGianGiaoDịch:Time
#loạiGiaoDịch:String
#sốTiền:float
#sốDư:float
#ngHàngDB:NgânHàngDB
+tạoMới()
+gánThôngTinGiaoDịch()
GiaoDịchRút
GiaoDịchGởi
NgânHàngDB
+đọcKháchHàng()
+đọcTàiKhoản()
+cậpNhậtTàiKhoản()
+cậpNhậtGiáoTác()
Tầng nghiệp vụ