LỜI CẢM ƠN

Trước tiên em xin gửi lời cám ơn chân thành sâu sắc tới các thầy cô giáo trong trường Đại học Dân Lập Hải Phòng nói chung và các thầy cô giáo trong khoa Công nghệ Thông tin nói riêng đã tận tình giảng dạy, truyền đạt cho em những kiến thức, kinh nghiệm quý báu trong suốt thời gian qua.

Đặc biệt em xin gửi lời cảm ơn đến thầy Đỗ Xuân Toàn đã tận tình giúp đỡ, trực tiếp chỉ bảo, hướng dẫn em trong suốt quá trình làm đồ án tốt nghiệp. Trong thời gian làm việc với thầy, em không ngừng tiếp thu thêm nhiều kiến thức bổ ích mà còn học tập được tinh thần làm việc, thái độ nghiên cứu khoa học nghiêm túc, hiệu quả, đây là những điều rất cần thiết cho em trong quá trình học tập và công tác sau này.

Đồng thời xin chân thành cảm ơn, trường Đại học Dân Lập Hải Phòng đã tạo mọi điều kiện về cơ sở vật chất giúp em có một môi trường tốt để thực hiện đề tài.

Sau cùng xin gửi lời cảm ơn chân thành tới gia đình, bạn bè đã động viên, đóng góp ý kiến và giúp đỡ trong quá trình học tập, nghiên cứu và hoàn thành đồ án tốt nghiệp.

Hải Phòng, tháng *07/2009*

Nguyễn Thị Thuỳ Dương

MỤC LỤC

Lời Mở Đầu

Ngày nay, sự phát triển về nhu cầu sở hữu các thiết bị kỹ thuật số mà trong đó thiết bị di động có thị phần khá lớn. Sự đòi hỏi về mẫu mã, chất lượng dịch vụ mà đặc biệt là tính năng của chiếc điện thoại, các phần mềm tiện ích đi kèm đã kéo theo sự phát triển của các Hệ điều hành để các nhà phát triển ứng dụng có thể thực hiện các ý tưởng của mình.Các hệ điều hành phổ biến đó như: **Windows Mobile, Linux Mobile, Symbian…**

Cùng với tốc độ phát triển đó là những tiến bộ vượt bậc về tốc độ xử lý. Nhờ đó lập trình các ứng dụng cho loại thiết bị này tăng lên nhanh chóng, đặc biệt là các dịch vụ giá trị gia tăng trên mạng di động như SMS, RSS, WAP và ứng dụng dịch vụ game. Qua tìm hiểu về, em thấy thị trường Game di động tại Việt Nam đang phát triển và có tiềm năng lớn; đó là lý do em chọn đề tài này.

Hiển nhiên có nhiều hạn chế cho game hơn là ứng dụng trên điện thoại bởi vì có nhiều sự tương tác giữa bàn phím, hình ảnh, sự sinh động, âm thanh và hiệu ứng rung. Hơn nữa, khi lập trình bạn không chỉ quan tâm đến sự khác nhau của từng nhà sản xuất mà còn đến sự khác nhau của các dòng sản phẩm của cùng 1 nhà sản xuất

Mặc dù hầu hết các điện thoại đời mới trên thị trường hiện nay đều hỗ trợ MIDP 2.0, tuy nhiên nếu bạn sử dụng điện thoại cũ hơn thì có thể nó chỉ hỗ trợ MIDP 1.0

Chương 1. Cơ sở lý thuyết.

1.1. Máy ảo java cho các điện thoại di động

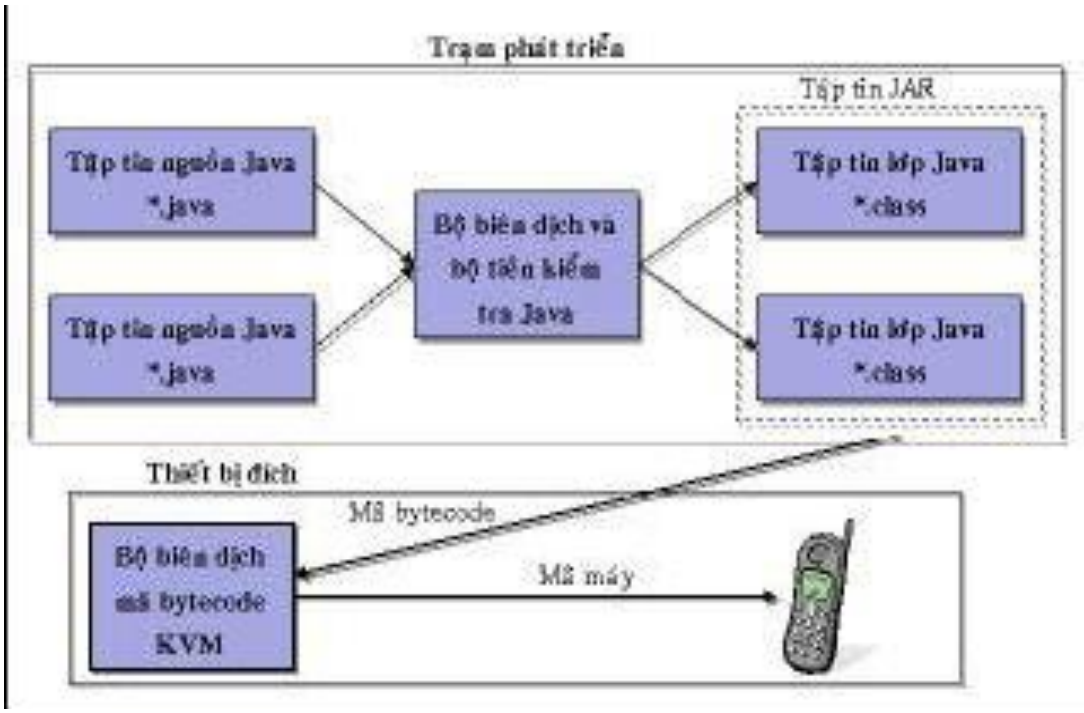
Để có một chiếc điện thoại tốt luôn đòi hỏi những phần mềm cao cấp đi kèm. Nhưng vấn đề lại đặt ra là có quá nhiều nhà sản xuất điện thoại sử dụng nhiều công nghệ khác nhau.

Chính vì thế, việc tạo ra các ứng dụng chạy được trên tất cả các dòng sản phẩm là một vấn đề không đơn giản. Nhưng với sự ra đời của J2ME, nó không những đáp ứng được các vấn đề nêu trên mà còn tạo nên tiền đề quan trọng trong việc phát triển và đẩy mạnh các ứng dụng cho Mobile.

Độc lập với phần cứng, chạy trên mọi nền tảng khác nhau của các nhà sản xuất khác nhau, đây cũng là một mục tiêu đồng thời cũng là thế mạnh mà J2ME đã mang lại.

Khi mã nguồn Java được biên dịch nó được chuyển đổi thành mã bytecode. Mã bytecode này sau đó được chuyển thành mã ngôn ngữ máy của thiết bị di động. Tầng máy ảo Java bao gồm KVM (K Virtual Machine) là bộ biên dịch mã bytecode có nhiệm vụ chuyển mã bytecode của chương trình Java thành ngôn ngữ máy để chạy trên thiết bị di động.

Vai trò của máy ảo Java hay KVM là dịch mã bytecode được sinh ra từ chương trình Java đã biên dịch sang ngôn ngữ máy. Chính KVM sẽ chuẩn hóa output của các chương trình Java cho các thiết bị di động khác nhau có thể có bộ vi xử lý và tập lệnh khác nhau. Không có KVM, các chương trình Java phải được biên dịch thành tập lệnh cho mỗi thiết bị di động. Như vậy lập trình viên phải xây dựng nhiều đích cho mỗi loại thiết bị di động.



Hình 1 Biểu diễn tiến trình xây dựng ứng dụng MIDlet hoàn chỉnh và vai trò của KVM.

1.2. Lập trình java cho Mobile

1.2.1. Ngôn ngữ java

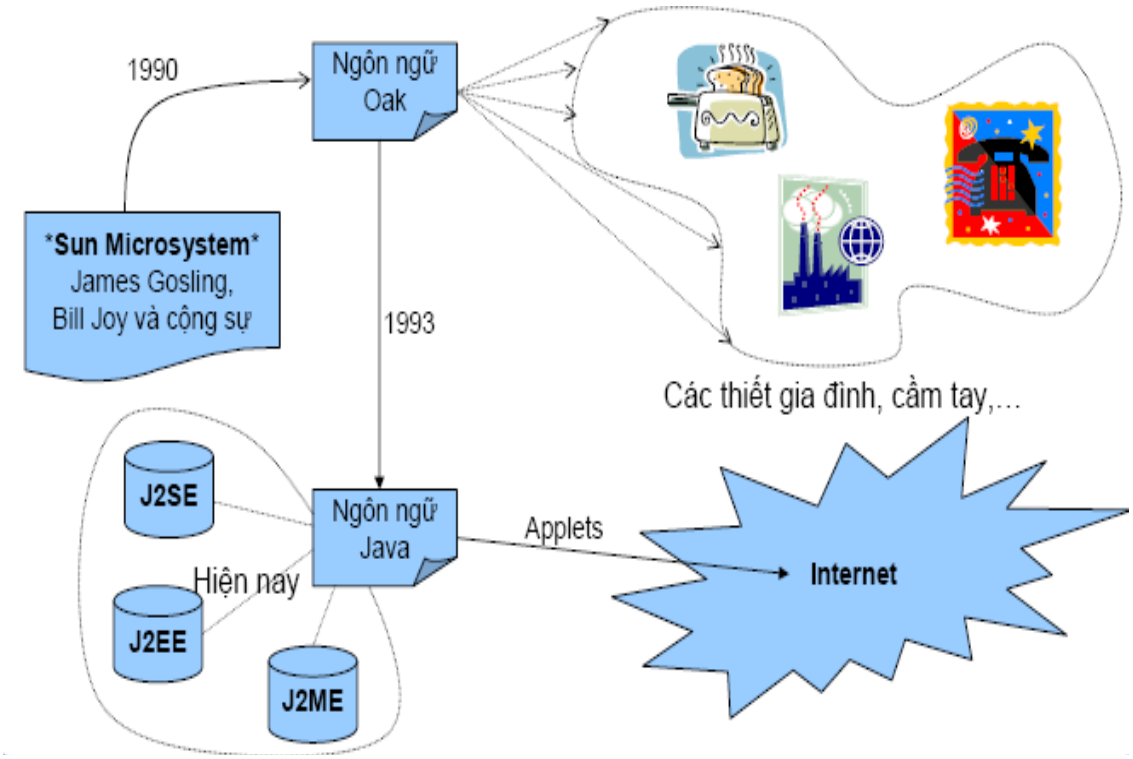
a. Java là một ngôn ngữ biên dịch. Tuy nhiên khác với các ngôn ngữ biên dịch phổ biến khác như C/C++, chương trình nguồn Java không được biên dịch trực tiếp sang một mã máy đích cụ thể nào mà được biên dịch sang mã máy ảo Java. Mã má ảo được thực hiện bởi máy ảo Java khi cần có thể được thông dịch sang hệ máy cụ thể.

Ưu điểm dễ nhận thấy của phương thức này là mã đích không phụ thuộc vào phần cứng hay hệ điều hành cụ thể do đó đảm bảo tính khả chuyển của chương trình.

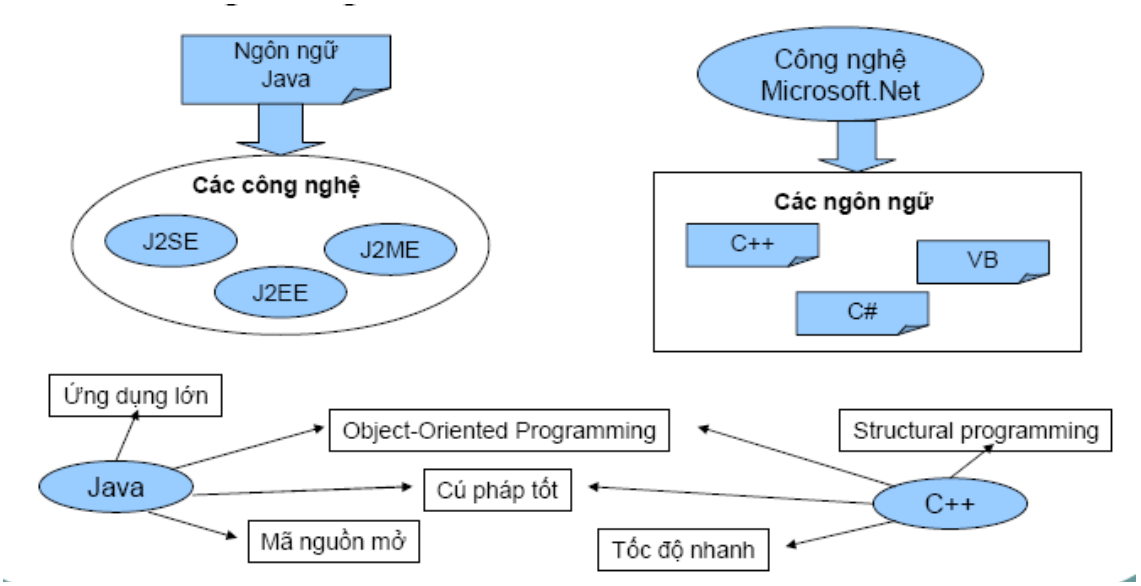
Nhược điểm của nó là do phải thực hiện trong môi trường máy ảo nên tốc độ sẽ chậm hơn so với nếu được dịch sang mã máy và thực hiện trực tiếp.

Một ưu điểm quan trọng khác của cơ chế máy ảo là nó cho phép kiểm soát sự truy cập đến các tài nguyên hệ thống.

b.Quá trình phát triển của Java



c. So sánh công nghệ Java và Microsoft.Net



1.3. Giới thiệu về J2ME

1.3.1. J2ME(Java 2 Micro Edition):

J2ME được phát triển từ kiến trúc Java Card, Embeded Java và Personal Java của phiên bản Java 1.1. Đến sự ra đời của Java 2 thì Sun quyết định thay thế Personal Java và đươc gọi với tên mới là Java 2 Micro Edition, hay viết tắt là J2ME. Đúng với tên gọi, J2ME là nền tảng cho các thiết bị có tính chất nhỏ, gọn .

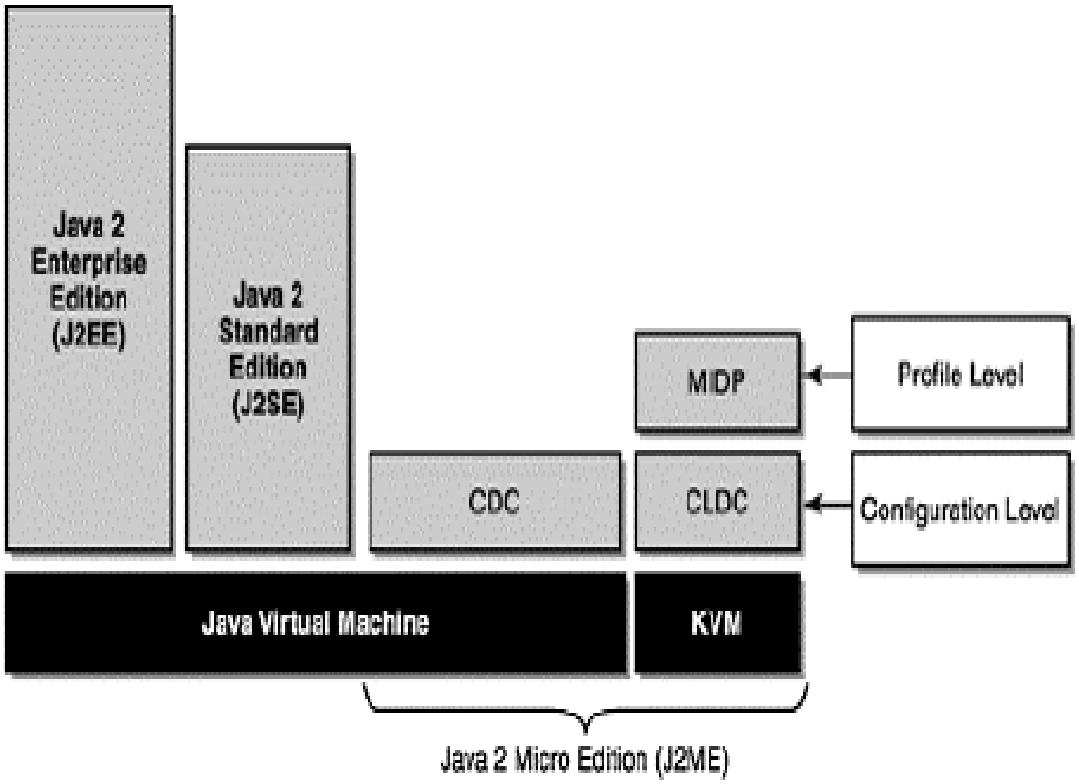
Java ban đầu được thiết kế dành cho các máy với tài nguyên bộ nhớ hạn chế. Thị trường của J2ME được mở rộng ra cho nhiều chủng loại thiết bị như:

- Các loại thẻ cá nhân như Java Card
- Máy điện thoại di động
- Máy PDA (Personal Digital Assistant- thiết bị trợ giúp cá nhân) .
- Các hộp điều khiển dành cho tivi ,thiết bị giải trí gia dụng .

Các bộ công cụ phát triển J2ME hầu như được cung cấp miễn phí . Đây sẽ là một thuận lợi cho những người mới bắt đầu với J2ME.

Một số tính năng ưu việt của J2ME: Cung cấp nội dung linh động, bảo mật, tương thích nền tảng, tính năng nâng cao,truy cập ngoại tuyến, và mang tính năng mạnh mẽ của một ngôn ngữ lập trình hướng đối tượng hiện đại.

1.3.2. Kiến trúc của J2ME



Hình 2 Kiến trúc của J2ME

Kiến trúc J2ME bao gồm các thành phần: ***Cấu hình*** (configuration), ***hiện trạng*** (profile), ***các gói tuỳ chọn*** (optional package) cho việc xây dựng hoàn thiện môi trường thực thi Java mà đáp ứng các yêu cầu cho một phạm vi lớn của các thiết bị và thị trường đích. Các nhà sản xuất thiết bị phát triển các tính năng mới trong các thiết bị của họ hoặc cung cấp các dịch vụ, ứng dụng. Các cấu hình có thể được mở rộng với các thư viện bổ sung cho từng thiết bị.

Cấu hình:

Cấu hình bao gồm máy ảo Java và một tập hợp rất nhỏ các hàm API theo đặc tả của một loại thiết bị nhất định.

Hình trạng:

Để cung cấp một môi trường thực thi hoàn thiện tại các loại thiết bị nhất định, các cấu hình phải được kết nối với một tập các API mức cao hơn.

Gói tuỳ chọn:

Nền tảng J2ME có thể được mở rộng hơn nữa bằng các gói tùy chọn với cấu hình CLDC, CDC, và các hình trạng tương ứng. Các gói tuỳ chọn đưa ra các chuẩn cho việc sử dụng cả công nghệ đã có và công nghệ đang được sử dụng rộng rãi là Bluetooth, dịch vụ Web, nhắn tin không dây, đa phương tiện, và kết nối cơ sở dữ liệu. Các nhà sản xuất thiết bị có thể kèm theo các gói tùy chọn để thúc đẩy các tính năng của các thiết bị.

1.3.3 - MIDP(Mobile Information Device Profile)

Tầng J2ME cao nhất là tầng hiện trạng và mục đích của nó là định nghĩa các API cho các thiết bị di động. Một thiết bị di động có thể hỗ trợ nhiều hiện trạng. Một hiện trạng có thể áp đặt thêm các giới hạn trên các loại thiết bị di động (như nhiều bộ nhớ hơn hay độ phân giải màn hình cao hơn). Hiện trạng là tập các API hữu dụng hơn cho các ứng dụng cụ thể. Lập trình viên có thể viết một ứng dụng cho một hiện trạng cụ thể và không cần quan tâm đến nó chạy trên thiết bị nào.

Hiện tại hiện trạng được công bố là MIDP (Mobile Information Profile) với đặc tả JSR - 37. Có 22 công ty là thành viên của nhóm chuyên gia tạo ra chuẩn MIDP. MIDP cung cấp các API cho phép thay đổi trạng thái chu kỳ sống ứng dụng, đồ họa (mức cao và mức thấp), tuyến đoạn, timer, lưu trữ bền vững (persistent storage), và mạng. Nó không định nghĩa cách mà ứng dụng được nạp trong thiết bị di động. Đó là trách nhiệm của nhà sản xuất. Nó cũng không định nghĩa bất kỳ loại mô hình bảo mật end-to-end nào, vốn cần thiết cho ứng dụng kinh doanh nhận số thẻ tín dụng của người dùng. Nó cũng không bắt buộc nhà sản xuất cách mà lớp MIDP được thực hiện.

❖ MIDP là một Profile được định nghĩa dành riêng cho các thiết bị di động và là thành phần chính trong J2ME, nó cung cấp các chức năng cơ bản cho hầu hết các dòng thiết bị di động phổ biến nhất hiện nay như các máy điện thoại di động và các máy PDA.

❖ Các ứng dụng J2ME được gọi là MIDlet (Mobile Information Device applet).

❖ Ứng dụng của MIDP với di động

Do MIDP là chuẩn phát triển ứng dụng Java trên các điện thoại Symbian nên hiện nay,có rất nhiều hãng điện thoại lớn sử dụng công nghệ J2ME trên nền Symbian có hỗ trợ MIDP.

Vào tháng 11 năm 2003, Sun đã tung ra phiên bản mới nhất là MIDP 2.0 với hàng loạt tính năng được cung cấp so với MIDP 1.0 như : nâng cao tính bảo mật, hỗ trợ Form tốt hơn, chụp ảnh RGB…v.v…

Các thiết bị nhắm tới J2ME : Các hãng Điện thoại di động như : NOKIA, SAMSUNG, MOTOROLA, SIEEMS, TOSIBA, SHARP...

SAMSUNG

SGH-730

SGH-D500

SGH-E630

NOKIA

N6230

N7610

N6600

1.3.4. Những hạn chế của lập trình di động

Sự khác nhau lớn nhất giữa lập trình cho di động so với lập trình cho PC là ta phải quan tâm lớn đến bộ nhớ, kích thước màn hình hay thậm chí là màu sắc của thiết bị. Những vẫn đề đó lại không quan trọng bậc nhất đối với PC. Hơn thế nữa ta còn phải quan tâm đến sự khác nhau giữa các hãng sản xuất cũng như dòng máy di động của hãng. Các dòng máy khác nhau sẽ có bộ nhớ, kích thước, màu sắc màn hình và cả hỗ trợ cho lập trình cũng khác nhau.Các loại bộ nhớ

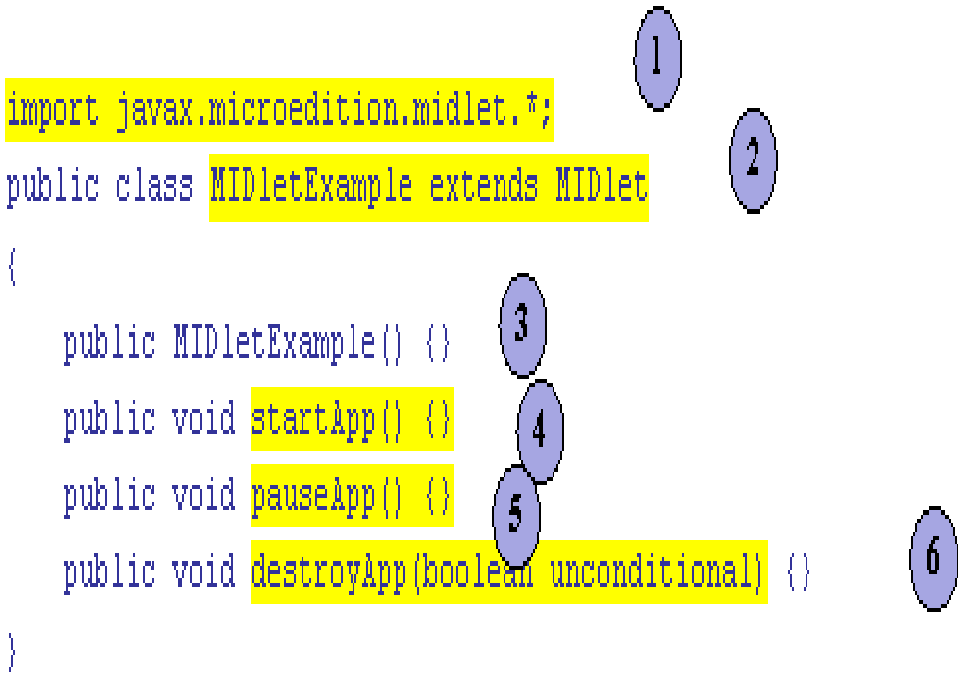
Bộ nhớ làm việc hay còn gọi là heap memory là phần bộ nhớ được dùng trong suốt quá trình chạy và được giải phóng khi ứng dụng kết thúc. Ứng dụng dành cho di động phải có bô nhớ nhỏ hơn vùng nhớ này mới có thể chạy được. Một loại bộ nhớ khác dùng để lưu trữ là bộ nhớ RMS (Record Management System).

RMS là bộ nhớ mà thông tin vẫn còn được lưu khi ta tắt ứng dụng. Lấy một ví dụ trong game ta dùng bộ nhớ này để lưu top những người chơi cao điểm nhất. Các thông tin đó vẫn được cập nhật cho lần chơi tiếp theo.

1.3.5.Tìm hiểu về một ứng dụng trong ĐTDĐ

1.3.5.1.Căn bản lập trình J2ME

1 chương trình trên mobile được gọi là **1 MIDLet .** Chương trình này được đong gói vào một lớp kế thừa từ lớp có sẵn của **JAVA là MIDLET**



Thành phần

Import: Khai báo sử dụng các lớp cần thiết trong thư viện MIDP và CLDC.Hàm khởi tạo

MIDletExample( ): Hàm khởi tạo được thực hiện một lần khi MIDlet khởi động.

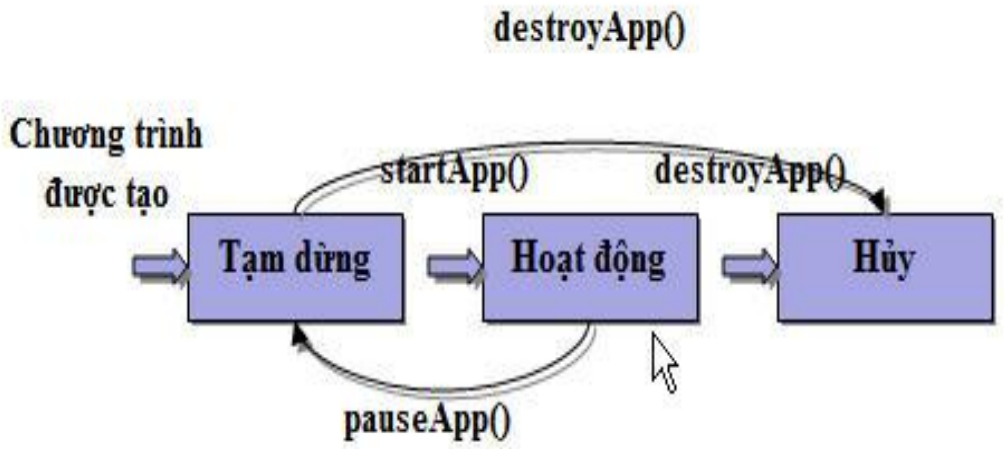
startApp( ): phương thức này được gọi bởi bộ quản lý ứng dụng khi khởi tạo MIDlet và mỗi khi MIDlet trở về từ trạng thái paused.

pauseApp( ) phương thức này gọi bộ quản lý ứng dụng mỗi khi ứng dụng cần tạm dừng ứng dụng đang thực thi. Khi sử dụng pauseApp ta giải phóng một phần tài nguyên của MIDlet để dành bộ nhớ cho các ứng dụng khác.

destroyApp( ): Phương thức này được dùng khi thoát khỏi MIDlet. Trước đó phải giải phóng hoàn toàn bộ nhớ được lấy bởi MIDlet.Có chú ý là

tất cả các ứng dụng MIDlet được viết ra đều kế thừa từ lớp MIDlet có sẵn trong MIDP. Lớp MIDlet nằm trong gói: *javax.microedition.midlet.** .

Lớp này quản lý chu kì sống của các ứng dụng MIDlet.



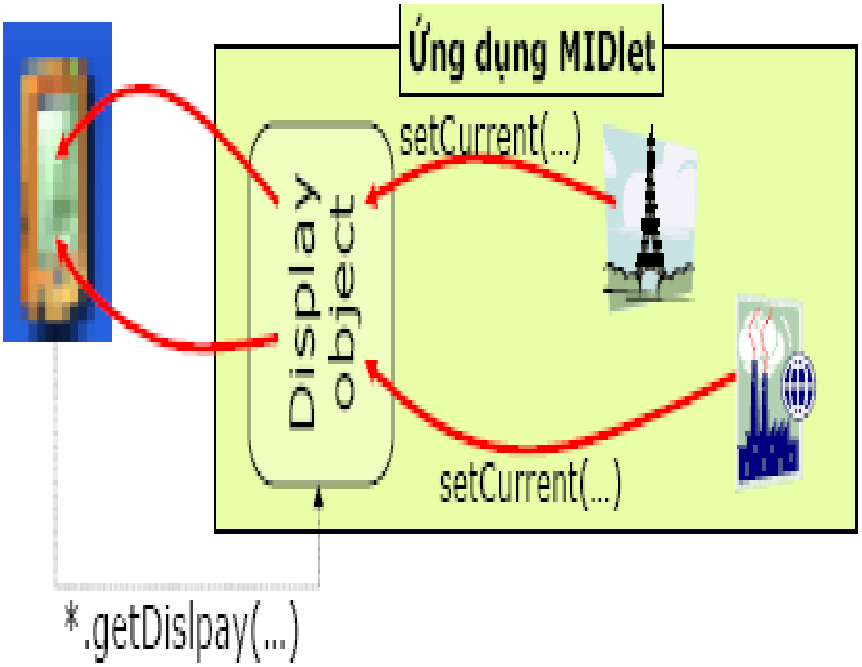
Hình 3 chu kì sống của các ứng dụng MIDlet

Khi người dùng yêu cầu khởi động ứng dụng MIDlet, bộ quản lý ứng dụng sẽ thực thi MIDlet (thông qua lớp MIDlet). Khi ứng dụng thực thi, nó sẽ được xem là đang ở trạng thái tạm dừng. Bộ quản lý ứng dụng gọi hàm tạo và hàm startApp(). Hàm startApp() có thể được gọi nhiều lần trong suốt chu kỳ sống của ứng dụng. Hàm destroyApp() chỉ có thể gọi từ trạng thái hoạt động hay tạm dừng. Lập trình viên cũng có thể điều khiển trạng thái của MIDlet. Các phương thức dùng để điều khiển các trạng thái của MIDlet: resumeRequest(): Yêu cầu vào chế độ hoạt động. Ví dụ: Khi MIDlet tạm dừng, và một sự kiện timer xuất hiện. notifyPaused(): Cho biết MIDlet tự nguyện chuyển sang trạng thái tạm dừng Ví dụ: Khi đợi một sự kiện timer. notifyDestroyed(): Sẵn sàng để hủy Ví dụ: Xử lý nút nhấn Exit Lập trình viên có thể yêu cầu tạm dừng MIDlet trong khi đợi một sự kiện timer hết hạn. Trong trường hợp này, phương thức notifyPaused() sẽ được dùng để

yêu cầu bộ quản lý ứng dụng chuyển ứng dụng sang trạng thái tạm dừng. 1.3 Tập tin JAR Các lớp đã biên dịch của ứng dụng MIDlet được đóng gói trong một tập tin JAR (Java Archive File). Đây chính là tập tin JAR được download xuống điện thoại di động. Tập tin JAR chứa tất cả các tập tin class từ một hay nhiều MIDlet, cũng như các tài nguyên cần thiết. Hiện tại, MIDP chỉ hỗ trợ định dạng hình .png (Portable Network Graphics). Tập tin JAR cũng chứa tập tin kê khai (manifest file) mô tả nội dung của MIDlet cho bộ quản lý ứng dụng. Nó cũng phải chứa các tập tin dữ liệu mà MIDlet cần. Tập tin JAR là toàn bộ ứng dụng MIDlet. MIDlet có thể load và triệu gọi các phương thức từ bất kỳ lớp nào trong tập tin JAR, trong MIDP, hay CLDC. Nó không thể truy xuất các lớp không phải là bộ phận của tập tin JAR hay vùng dùng chung của thiết bị di động.

1.3.5.2. Cách quản lý màn hình của ĐTDĐ :

Điện thoại di động không quản lý trực tiếp trên màn hình như máy tính mà phải thông qua một đối tượng Display được lấy qua từ câu lệnh : **Display.getDisplay(this)**



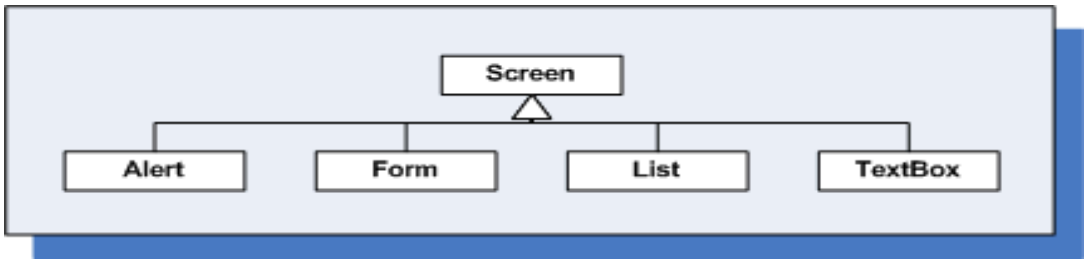
Hình 5 quản lý màn hình của ĐTDĐ

1.3.5.3. Kiến trúc tổng quan giao diện người dùng trong MIDP

MIDP bao gồm các thành phần giao diện và được định nghĩa trong gói **javax.microedition.lcdui**. Chú ý từ viết tắt lcdui chính là LCD UI (liquid crystal display user interface – giao diện người dùng cho màn hình tinh thể lỏng)

Đồ họa trong J2ME

Các lớp giao diện trong gói javax.microedition.lcdui của MIDP có thể được chia thành 2 nhóm là nhóm giao diện cấp cao và nhóm giao diện cấp thấp. Những lớp trong nhóm cấp cao thích hợp khi phát triển các ứng dụng MIDlet có khả năng chạy trên tất cả các thiết bị, bởi vì các lớp cấp cao này không cung cấp khả năng quản lý việc hiển thị trên màn hình, nghĩa là việc quyết định hiển thị các thành phần giao diện này là do điện thoại quản lý sao cho phù hợp nhất với đặc thù của từng điện thoại.



Hình 6 Đồ họa mức cao

Đồ họa mức cao Là các đối tượng của lớp Screen

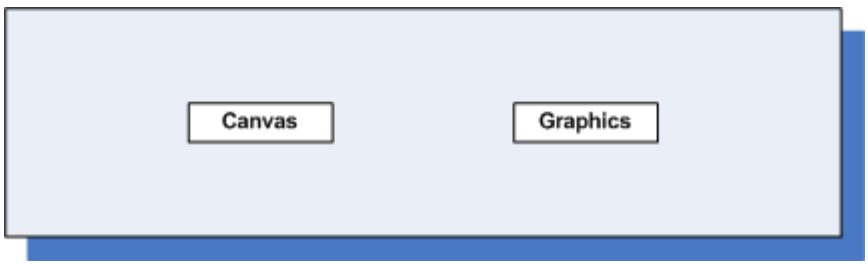
a. TextBox Lớp TextBox cho phép người dùng nhập và soạn thảo văn bản. Lập trình viên có thể định nghĩa số ký tự tối đa, giới hạn loại dữ liệu nhập (số học, mật khẩu, email,…) và hiệu chỉnh nội dung của textbox. Kích thước thật sự của textbox có thể nhỏ hơn yêu cầu khi thực hiện thực tế (do giới hạn của thiết bị). Kích thước thật sự của textbox có thể lấy bằng phương thức getMaxSize().

b. Form Form là lớp hữu dụng nhất của các lớp Screen bởi vì nó cho phép chứa nhiều item trên cùng một màn hình. Các item có thể là DateField, TextField, ImageItem, TextItem, ChoiceGroup.

c. List Lớp List là một Screen chứa danh sách các lựa chọn chẳng hạn như các radio button. Người dùng có thể tương tác với list và chọn một hay nhiều item. **d. Alert** Alert hiển thị một màn hình pop-up trong một khoảng thời gian. Nói chung nó dùng để cảnh báo hay báo lỗi. Thời gian hiển thị có thể được thiết lập bởi ứng dụng. Alert có thể được gán các kiểu khác nhau (alarm, confirmation, error, info, warning), các âm thanh tương ứng sẽ được phát ra.

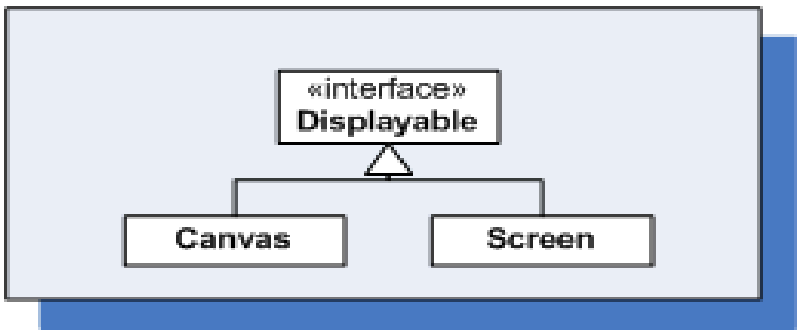
e. Form và các Form Item Sử dụng form cho phép nhiều item khác nhau trong cùng một màn hình. Lập trình viên không điều khiển sự sắp xếp các item trên màn hình. Sau khi đã định nghĩa đối tượng Form, sau đó sẽ thêm vào các item.

Còn các lớp cấp thấp thì thích hợp cho những ứng dụng MIDlet cần điều khiển chính xác việc hiển thị cũng như toạ độ của các thành phần giao diện. Tất nhiên nếu ta điều kiển sâu vào việc hiển thị và toạ độ thì ứng dụng sẽ bớt đi tính khả chuyển, tức là ứng dụng sẽ chạy được trên một ít thiết bị mà thôi. Thường thì khi dùng các lớp cấp thấp này thì bạn cần viết lại ứng dụng cho từng đời điện thoại. Hình vẽ sau cho thấy 2 lớp giao diện cấp thấp là Canvas và Graphics



Hình 7 Canvas và Graphics

Cho dù ta sử dụng giao diện cấp cao hay cấp thấp thì tất cả các thành phần giao diện đều phải kế thừa từ giao tiếp tên là Displayable. Một lớp Displayable có thể có các thành phần như tiêu đề, các lệnh và các mục giao diện khác. Vì vậy nên 2 lớp Screen và Canvas đều kế thừa giao tiếp Displayable như trong hình vẽ sau (chú ý là lớp Graphics không kế thừa giao tiếp Displayable)



Hình 8

Đồ họa mức thấp (Lớp Canvas)

Đồ họa mức thấp là lớp con của lớp Canvas. Lớp này cung cấp các phương thức đồ họa cho phép vẽ lên màn hình hay vào một bộ đệm hình cùng với các phương thức xử lý sự kiện bàn phím. Lớp này dùng cho các ứng dụng trò chơi cần điều khiển nhiều về màn hình.

Với phiên bản MIDP 2.0, công việc lập trình di động nói chung và lập trình game nói riêng sẽ dễ dàng hơn rất nhiều.

*** Lớp GameCanvas** mới có thể vẽ lên màn hình và đáp ứng lại dữ liệu nhập trong phần thân của vòng lặp game, thay vì dựa vào các thread vẽ và nhập liệu của hệ thống. **GameCanvas** là một **Canvas** có thêm một số khả năng; nó cung cấp các phương thức để vẽ tức thời và kiểm tra trạng thái bàn phím thiết bị.

*** Một API** dùng để quản lý layer mạnh và linh hoạt giúp việc xây dựng các cảnh game phức tạp một cách hiệu quả hơn.

Sử dụng các **luồng (thread)** xử lý các đối tượng trong một ứng dụng game chạy song song với nhau: ví dụ như các luồng vẽ đồ họa trong một

đối tượng đồ họa,luồng âm thanh trong đối tượng âm thanh ,…sẽ cùng được chạy trong cùng một thời điểm khi bắt đầu trò chơi trò chơi .

Ngoài ra, việc sử dụng đa luồng (Multithreading) này giúp cho tài nguyên hạn chế của thiết bị di động được sử dụng một cách tích cực hơn.

Chương 2: J2ME game API

Phần một đã nói tổng quan về lập trình cho di động, sau đây em xin được giới thiệu về một mảng kiến thức rất thú vị của J2ME là Game API. Phần hai gồm hai mục:

• Giới thiệu lý thuyết Game API của J2ME

• Công cụ lập trình Java Game của Netbeans

1. J2MEgame API

1. Lớp Canvas
2. Lớp Layer
3. Lớp Sprite
4. Lớp TiledLayer
5. Lớp LayerManager

Thực tế ta hoàn toàn có thể làm game J2ME với kiến thức về lập trình di động J2ME, và đặc biệt là thành phần giao diện mức thấp Canvas của nó. Lớp Canvas cho phép ta vẽ lên màn hình điện thoại, nó hỗ trợ vẽ các đối tượng như đường thẳng, đường tròn, tô màu một vùng màn hình, hay có thể copy một vùng màn hình vào vùng đệm và đặt nó vào chỗ khác …. Với những công cụ như thế ta có thể làm được một chương trình game từ đơn giản đến phức tạp. Tuy nhiên thời gian làm game sẽ rất tốn kém và khối lượng công việc liên quan đến vẽ giao diện là rất lớn.Qua tìm hiểu em thấy J2ME có một gói hỗ trợ tốt cho lập trình game gọi là chung là Game API. Các lớp của nó nằm trong javax.microedition.lcdui.game.

Một số lớp quan trọng của nó như: GameCanvas, Layer, Sprites, TiledLayer, và LayerManager.

• GameCanvas: kế thừa từ lớp Canvas, có thể coi đay là màn hình điện thoại.

• Layer: ít được sử dụng trực tiếp nhưng được kết bởi lớp Sprite và TiledLayer.

• Sprites: lớp để tạo các nhân vật và các đối tượng chuyển động.

• TiledLayer: lớp để tạo các khung cảnh khác nhau của game và tạo bản đồ.

• LayerManager: lớp quản lý các Sprites và các TiledLayer

1. GameCanvas class

GameCanvas kết thừa các thuộc tính và phương thức của lớp Canvas và có thêm một số phương thức quan trọng giúp cho việc giảm thời gian tính toán nhờ đó tăng tốc độ refresh của màn hình. Nó có thêm hai mở rộng so với Canvas là

Polling Input – lược bỏ đầu vào và *Frame Buffer* – dùng vùng đệm để chứa nội dung Canvas.

Lớp GameCanvas: không phải đợi để thực hiện keyPressed() vì ta có phương thức getKeysState(). Có kỹ thuật gọi là "*double buffering*"bạn có thể thực hiện vẽ ở "off-screen buffer" khi đó việc copy từ buffer tới các canvas nhanh hơn nhiều. Có thể sử dụng Graphic từ gọi hàm getGraphics(). flushGraphics() để giải phóng Graphics…

2. Layer class

Trước khi đi đến hai lớp quan trọng tiếp theo là Sprite và TiledLayer ta phải nói về khái niệm Layer. Lớp layer của Game API rất đơn giản nhưng nó được kế thừa bởi hai lớp Sprite và TiledLayer. Layer được hiểu là một khối hình ảnh trong Game. Tất cả các hình ảnh có thể hiện được trên màn hình đều kế thừa lớp này. Các Layer là những ảnh mà ta có thể vẽ lên màn hình, ẩn đi, di chuyển hay là xắp xếp chúng theo độ xâu (tức là khi nhiều ảnh chồng chéo lên nhau thì ảnh có độ sâu lớn hơn sẽ bị ảnh có độ xâu nhỏ hơn che khuất). Các phương thức của lớp Layer là

Public void setPosition(int x, int y): đặt vị trí của Layer

Public void Move (int dx, int dy): dịch chuyển Layer

Public void setVisible (boolean visible): thiết lập hiện/ẩn LayerCác phương thức này đều được kế thừa bởi các lớp Sprite và TiledLayer.

3. Sprite Class

Trong Game thì đồ hoạ làm nên thành công rất lớn. Hầu hết các đối tượng đồ hoạ được phân loại như các dạng đặc biệt của đồ hoạ gọi là các

Sprite. Một Sprite có thể là bullet, monster, enemies, keys và doors và một vài cái gì đó…

Các Sprite được nhân nhiều lên là các graphic động, các graphic động này được tạo nên từ cùng một Sprite nhưng nhìn từ các góc khác nhau. Đây là một bộ Sprite:

Hình 9

Sprite Constructor

Có 3 hàm khởi tạo với lớp Sprite

Sprite (Image image); // Tạo ra khung Sprite đơn, không động

Sprite (Sprite sprite); //Tạo ra Sprite mới từ một Sprite

Sprite (Image image,int frameWidth, int frameHeight); //Tạo ra Sprite động với từ 2 frame trở lên, frameWidth và frameHeight là độ rộng và chiều cao của 1 Sprite

Ta có tổng độ rộng là 160 pixels, độ rộng của 1 frame là 32 pixels, chiều cao là 32pixels. Ta có frameWidth và frameHeight là giống nhau cho 1bộ Sprite (các Sprite khác thì khác nhau).

Các Graphic thì bao gồm các Sprite mà độ rộng và chiều cao là hằng số, vì số các pixel thì liên quan đến số màu: nếu 1pixel là 8-bit, 16-bit, 24-bit… thì 28=256, 216=65536… màu

Animation – chuyển động: Tạo hình ảnh chuyển động của nhân vật bằng cách gép nối liên tiếp các hình ảnh gần giống nhau. Ví dụ trong game em làm có sử dụng hình ảnh nhân vật (Player) và chuyển động.

Transforms – quay góc độ: Sprite cho phép chuyển xoay hình ảnh đi một góc nào đó bằng phương thức :

public void setTransform(int transform);

Có 8 kiểu xoay:

TRANS_NONE, tương ứng với 0 chiều kim đồng hồ

TRANS_ROT90, tương ứng với 90 chiều kim đồng hồ

TRANS_ROT180, tương ứng với 180 chiều kim đồng hồ

TRANS_ROT270, tương ứng với 270 chiều kim đồng hồ

TRANS_MIRROR, tương ứng với 0 xoay ngược theo gương, rồi xoay một góc nào đó theo chiều kim đồng hồ.

TRANS_MIRROR_ROT90, tương ứng với 90

TRANS_MIRROR_ROT180, tương ứng với 180

TRANS_MIRROR_ROT270: tương ứng với 270

• Image ở đây là ảnh lớn (kích thước 64)

• FrameWidth là chiều rộng của mỗi ảnh nhỏ (16)

• FrameHeight là chiều cao của mỗi ảnh nhỏ (16)

public Sprite(Image image, int frameWidth, int frameHeight);

bây giờ nếu muốn có ảnh chuyển động ta chỉ tạo một mảng có số phần tử là số ảnh để tạo chuyển động, các giá trị của mảng là chỉ số của các ảnh nhỏ cần ghép tương ứng. Ta gọi chuỗi đó là *Frame Sequences.*

Ví dụ để tạo một Sprite chuyển động gồm 16 ảnh nhỏ từ ảnh lớn trên để tạo thành một chuỗi hình ảnh chuyển động quay tròn ngược chiều kim đồng hồ, trước hết ta phải tạo ra đối tượng sprite

Sprite sprite = new Sprite (playerImage, 16, 16);

Sau đó tạo ra mảng các chỉ số thể hiện cho một chuỗi các frames như sau:

Int[ ] frameSequence = { 0, 1, 2, 3};

Nếu muốn cho Sprite hiển thị một frame cụ thể ta dùng phương thức:

Sprite.setFrame(int frameIndex);

Ta có thể định hướng chuyển động bằng các phương thức:

public void nextFrame()

public void prevFrame()

4. TiledLayer class

Mỗi lớp TiledLayer có một ảnh, ảnh này được chia nhỏ thành các tile theo quy tắc ở trên, một TiledLayer nó gồm một mảng các tile đặt theo thứ tự nhất định. Ảnh của TiledLayer được phân chia ra nhiều tiles và được đánh số. Một bản đồ có thể được tạo nên từ một TiledLayer hay nhiều TiledLayer.Hàm tạo của lớp TiledLayer như sau:

public TiledLayer(int columns, int rows, Image image, int tileWidth, int tileHeight);

• Colums là số cột của TiledLayer, hay số tiles trong một hàng

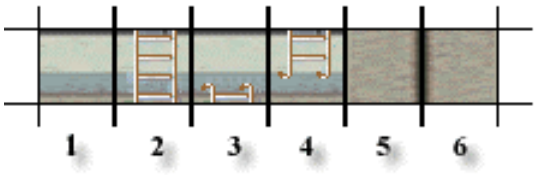
• Rows là số hàng của TiledLayer, hay số tiles trong một cột

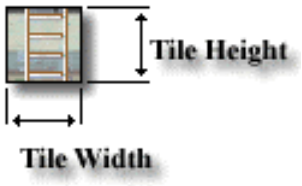
• Image là ảnh của TiledLayer

• tileWidth và tileHeight cho biết kích thước của một tile

Một TiledLayer là một lưới các ô chia ra từ 1 ảnh.

Ví dụ: hình dưới được chia thành 6 vùng, ta chỉ ra các Tiled có 32x32 pixel. Tạo nên 1 lớp TiledLayer, mỗi 1 tile này được đánh số (bắt đầu từ 1). Đánh số từ trái sang phải rồi từ trên xuống dưới.





Ta cũng có thể đặt một miền chữ nhật trong TiledLayer bằng một tile bằng phương thức:

public void fillCells(int col, int row, int numCols, int numRows, int tileIndex);

• col, row, numCols, nunRows xác định miền chữ nhât cần phủ tile

• tileIndex là chỉ số của tile đem phủ.

Tile Animation - tile động

Tile phần lớn được dùng để tạo ra các đối tượng tĩnh như bản đồ, tuy nhiên để tạo ra một khung nền thú vị ta nên có các tile động (các tile này đơn giản là các tiles thay đổi hình ảnh của nó sau một chu kì vẽ lại màn hình). Các tile động là những đối tượng hoạt động không chịu sự điều khiển của người chơi mà hoạt động theo chu kì. Các bước để tạo một tile động như sau:Để xác định tile nào là tile động trong một TiledLayer ta dùng phương thức:

public int createAnimatedTile(int staticTileIndex);

staticTileIndex : là chỉ số của tile trong TiledLayer

Sau mỗi lần vẽ lại màn hình, các TiledLayer được vẽ lại và do đó các tile của nó cũng được cập nhật theo. Để thay đổi vị trí của một tile với một tile ở vị trí khác ta dùng phương thức:

public void setAnimatedTile(int animatedTileIndex, int staticTileIndex);

Trong phương thức trên thì tile có chỉ số animatedTileIndex sẽ được cập nhật hình ảnh của tile có chỉ số staticTileIndex.

5. LayerManager class

Lớp này quản lý tất cả các đối tượng khác, thay vì trực tiếp xác định vị trí để vẽ các đối tượng lên màn hình bằng phương thức paint(), ta tạo một đối tượng LayerManager và cho vào nó tất cả các đối tượng bằng phương thức:

public void append(Layer l);

Đối tượng *append* có thể là đối tượng *Layer* hoặc những đối tượng của lớp kế thừa từ Layer như *Sprite* và *TiledLayer*. Ở đây ta vẫn vẽ từng đối tượng lên màn hình nhưng khác so với vẽ trực tiếp lên màn hình ở chỗ các quá trình vẽ được thâu tóm lại chỉ trong đối tượng LayerManager mà không vẽ rời rạc từng thành phần một. Việc vẽ tất cả các đối tượng đồng loạt như vậy sẽ tốt cho tính thống nhất xuất hiện của các đối tượng khác nhau trên màn hình.Lớp LayerManager cung cấp các phương thức quản lý Layer như:

• *insert (Layer l, int index)*

• *remove (Layer l)*

• *getLayerAt (int index)*

Phương thức thứ nhất chèn một Layer vào lớp ở vị trí có chiều xâu index. Ở đây một layer chèn vào có tính đến chiều sâu mà nó được đặt (layer có chiều sâu lớn hơn sẽ bị che bởi những layer có chiều xâu nhỏ hơn). Do đó việc đặt một layer lên màn hình khi sử dụng LayerManager sẽ không phụ thuộc vào thứ tự viết code, mà phụ thuộc vào biến index mà truyền vào.

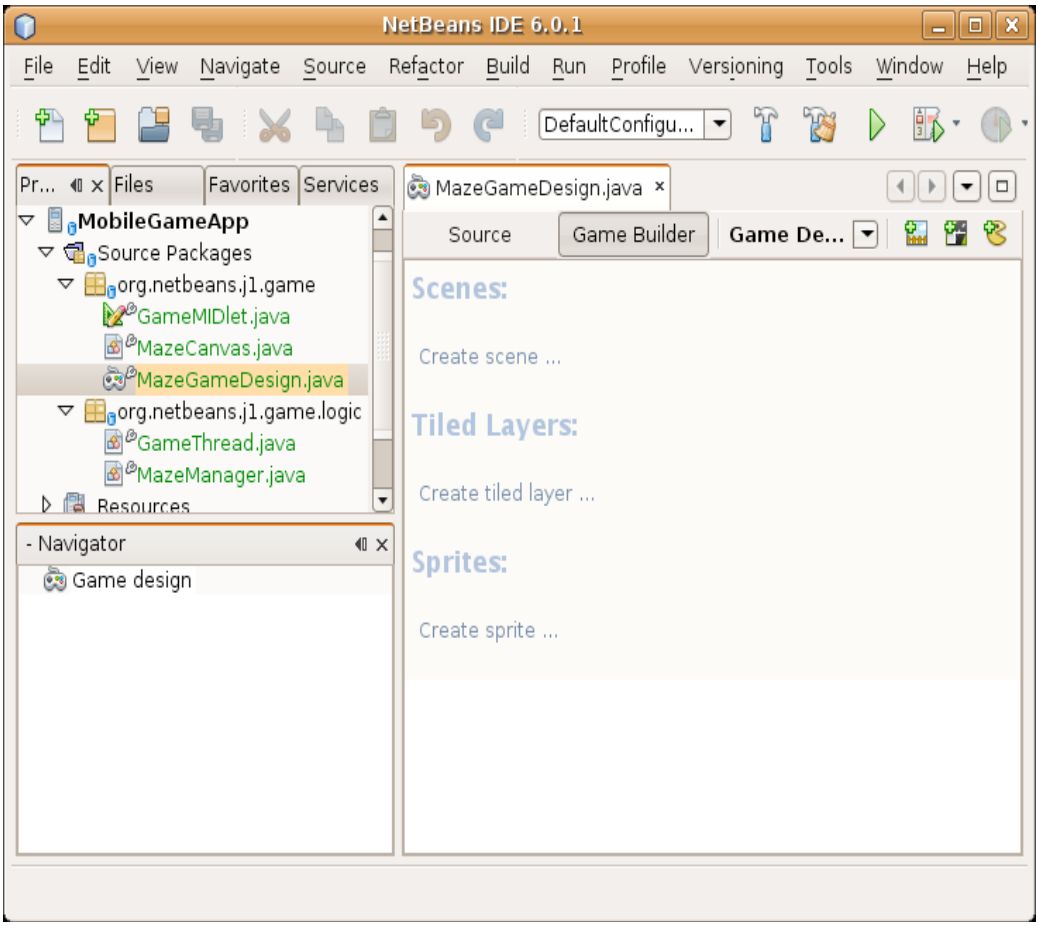
6. Công cụ lập trình của Netbeans

NetBean IDE là một "môi trường phát triển tích hợp" (Integrated Development Environment) kiểu như Visual Studio của Microsoft và được xem là một một bộ ứng dụng "must-download" dành cho các nhà phát triển phần mềm.

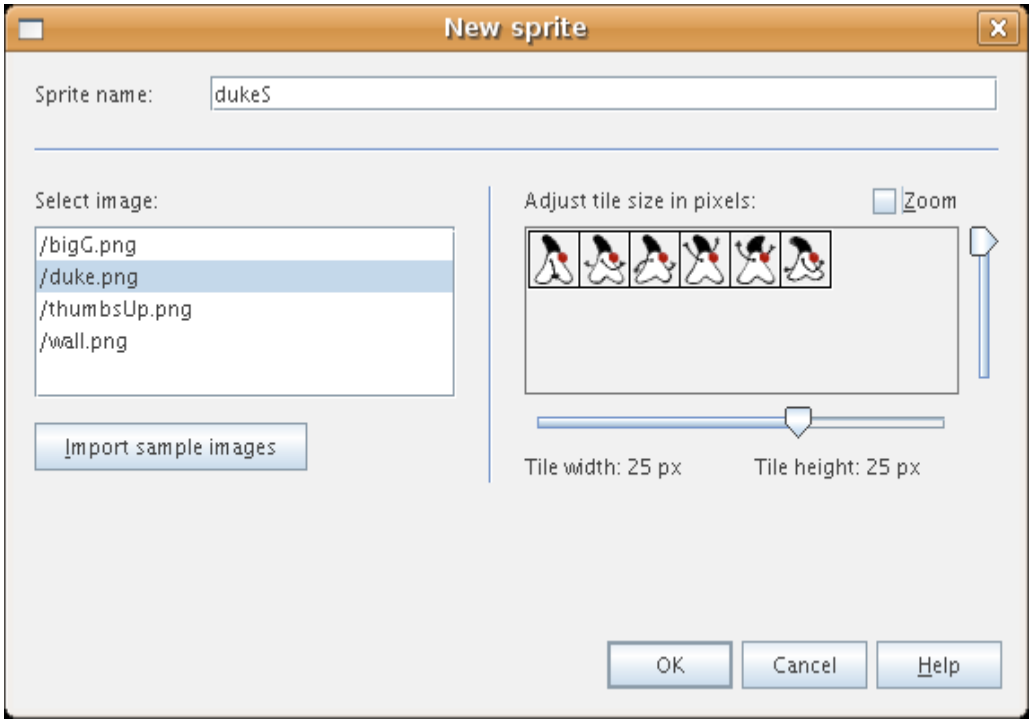
NetBean IDE hỗ trợ nhiều hệ điều hành khác nhau như Windows, Mac, Linux, và Solaris. NetBean bao gồm một IDE mã nguồn mở và một nền tảng ứng dụng cho phép nhà phát triển nhanh chóng tạo nên các ứng dụng dành cho web, doanh nghiệp, desktop và thiết bị di động bằng các ngôn ngữ lập trình Java, C/C++, JavaScript, Ruby, Groovy, và PHP.

NetBean IDE hỗ trợ hai công cụ thiết kế game là GameDesign và VisualMidlet. Trong đó GameDesign làm phần giao diện game còn phần VisualMIDlet làm phần menu cho game.

Sau khi tìm hiểu về lý thuyết GAME API ta đến với việc thực hành những kiến thức đó trên ngôn ngữ và công cụ cụ thể. Đây là khung màn hình GameDesign của Netbeans.

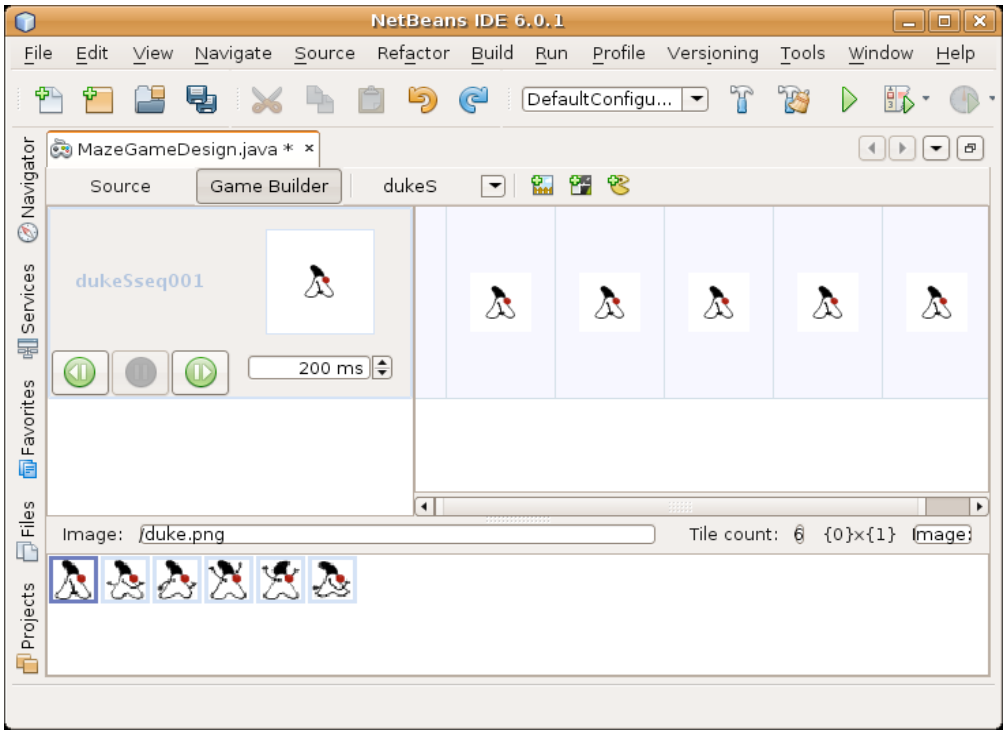


Hình 10 GameDesign của Netbeans



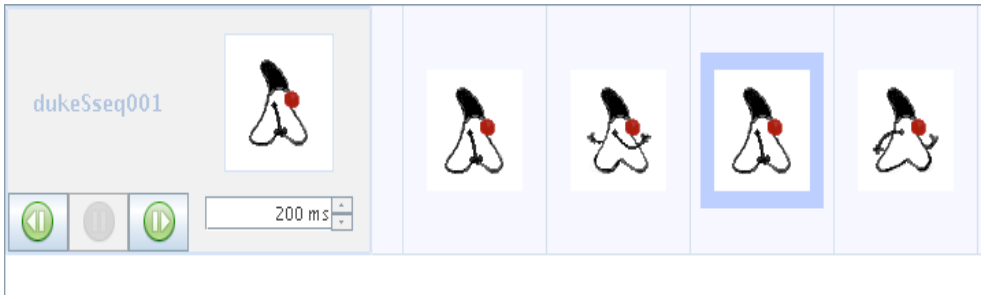
Hình 11

Hình trên là tạo ra Sprite với Image duke.png. Có thể tự điều chỉnh độ rộng độ dài của Frame



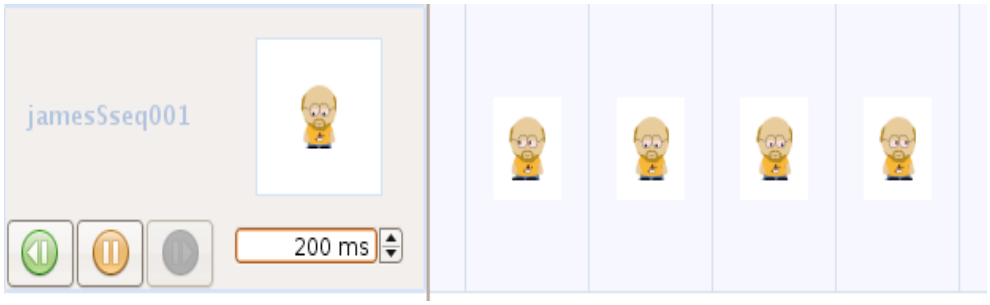
Hình 12 Tạo ra Sprite

Tạo ảnh chuyển động của nhân vật.



Hình 13

Để tạo một nhân vật chuyển động ta chỉ cần kéo những tile trên vào các ô tương ứng để tạo thành một chuỗi các ảnh.

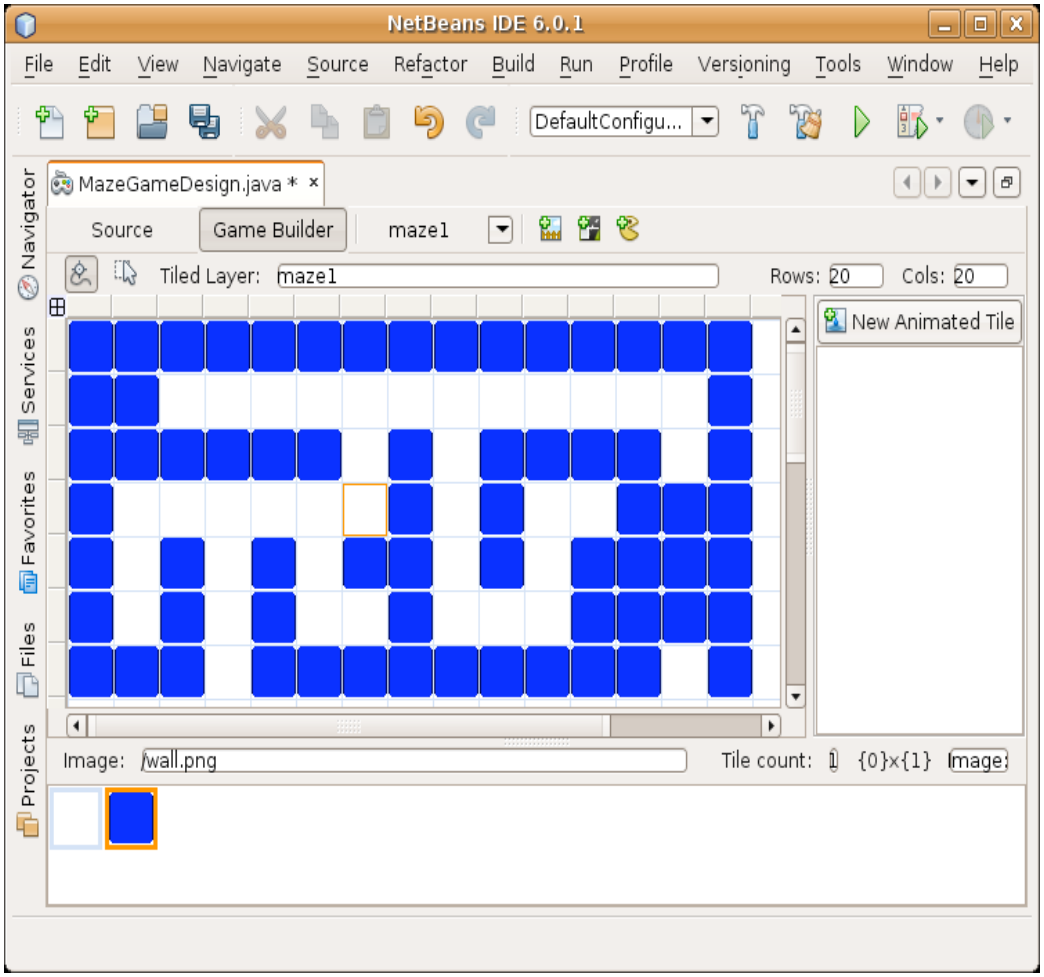


Hình 14

GameDesign hỗ trợ test thử bằng cách nhấn nút play hay pause. Số 100ms ở trên có nghĩa là độ trễ giữa hai ảnh kế tiếp nhau. Sau khi chạy hết một lượt, quá trình được lặp lại.

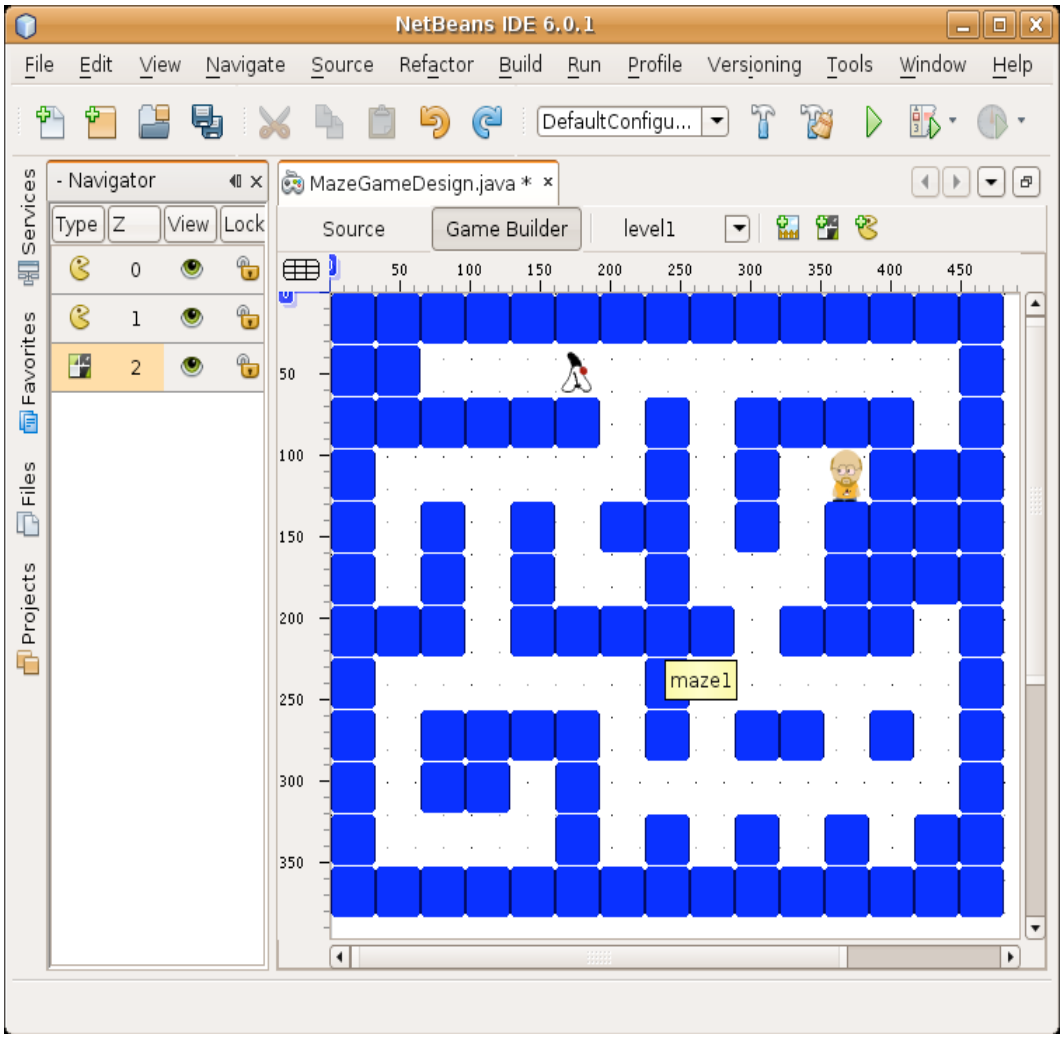
Tạo các TiledLayer

Tạo các TiledLayer bằng cách chọn những tile ở dưới và đặt lên khung hình trắng ở trên. Kích thước của TiledLayer có thể thay đổi được.



Hình 15 Tạo các TiledLayer

Tạo ra khung hình trong game. Bước cuối cùng là tạo ra tạo ra bản đồ cho game bằng cách kết hợp các TiledLayer và các nhân vật .



Hình 16 Tạo scene

Chương 3: Phân Tích và thiết kế Game đơn giản

1. Mô tả luật chơi của trò chơi

Tên trò chơi: **Duke & Jame** .

Logic trò chơi rất đơn giản :

1. Trò chơi có 2 nhân vật là: Duke và Jame

Duke

Jame

2. Nhân vật Duke sẽ đi bộ trong khung để tìm Jame .
3. Duke có thể đi, nhảy, hoặc rơi. Jame thì đứng một chỗ và đảo mắt xung quanh.
4. Jame có thể đứng ở bất kỳ đâu trong khung .
5. Khi Duke thấy Jame thì trò chơi kết thúc.
6. Thời gian mà Duke đi tìm Jame thì sẽ được tính để ghi điểm người chơi.

Thời gian mà Duke tìm thấy càng nhanh thì điểm càng cao.

2. Một số luồng quan trọng trong chương trình

Chương trình có 3 lớp làm về giao diện:

1. **GameDesign:** gồm các thuộc tính và các phương thức public chuyên về tạo ra các đối tượng đồ họa như Sprite, TiledLayer, Image ...

2.**VisualMIDlet:** bản chất đây là một MIDlet vì nó được mở rộng từ lớp MIDlet của J2ME. Lớp này có nhiệm vụ tạo ra giao diện menu của game, đồng thời có một số phương thức cho chức năng High Score của game.Chức năng High Scores lưu giữ điểm của 3 người chơi có điểm số cao nhất trong máy.

Các điểm số được lưu trong bộ nhớ RMS của máy, vì thế các thông số này vẫn lưu lại ngay cả khi ta tắt ứng dụng.

3.**GameScreen:** bản chất đây là một GameCanvas lớp này chứa tất cả các đối tượng đồ họa lấy từ lớp GameDesign.

Lớp này có rất nhiều phương thức như:

1.Vẽ các đối tượng lên màn hình

2.Chờ và xác nhận phím bấm

3.Tạo các hình ảnh chuyển động: Các nhân vật

4.Xác nhận trạng thái của game: Game Over, Running

5.Lớp này là có một phương thức run() trong đó có vòng lặp While (Running){ … } thực hiện tất cả các thao tác ở trong vòng lặp đó, kể cả gọi Thread của các lớp nhân vật.

Mối quan hệ của ba lớp được mô tả qua hình vẽ

Hình 17 3 lớp làm về giao diện

Được hiểu là lớp MIDlet và lớp GameScreen sẽ có quan hệ kết hợp (Association) với nhau.

(VisualMIDlet <-> menu game và GameScreen <->giao diện chơi game có thể gửi thông diệp đến nhau). Còn lớp GameDesign chỉ có quan hệ một chiều với lớp GameScreen.

Trong lớp GameDesign hỗ trợ API dùng để quản lý layer mạnh và linh hoạt giúp việc xây dựng các cảnh game phức tạp một cách hiệu quả hơn. Tạo các đối tượng đồ hoạ để lớp GameScreen cập nhật lên màn hình.

Trong lớp GameScreen (GameCanvas) có một luồng chính thực hiện tất cả các công việc trong đó – Main Loop

While (running){

+*Xử lý sự kiện nhấn phím*

+*Cập nhật màn hình*

+*Vẽ lại màn hình*

+*Xử lý thời gian tính toán cho mỗi vòng lặp*}

Lớp GameCanvas mới có thể vẽ lên màn hình và đáp ứng lại dữ liệu nhập trong phần thân của vòng lặp game, thay vì dựa vào các thread vẽ và nhập liệu của hệ thống.Đầu tiên, GameCanvas cho phép bạn truy xuất trực tiếp đối tượng Graphics của nó sử dụng phương thức getGraphics(). Bất kỳ công việc vẽ nào trên đối tượng Graphics được trả về sẽ được hoàn thành trong bộ đệm ngoài màn hình (offscreen). Sau đó bạn có thể chép bộ đệm vào màn hình sử dụng flushGraphics(), không cần phải đợi cho đến khi màn hình được cập nhật. Cách tiếp cận này cho phép bạn điều khiển tốt hơn việc gọi repaint(). Phương thức repaint() trả về ngay lập tức và ứng dụng của bạn không cần phải bảo đảm chính xác khi nào hệ thống sẽ gọi paint() để cập nhật màn hình.

GameCanvas cũng có một phương thức để lấy trạng thái hiện thời của bàn phím thiết bị, một kỹ thuật được gọi là *thăm dò (polling)*. Thay vì chờ cho hệ thống gọi phương thức keyPressed(), bạn có thể xác định ngay lập tức các phím nào được nhấn bằng cách gọi phương thức getKeyStates() của GameCanvas.

Cuối cùng là lớpVisualMIDlet. java sử dụng lớp Canvas này.

2.1.Các lớp đối tượng :

// Đối tượng GameMIDlet là lớp chính trong chương trình.

// gọi đối tượng Display hiện thị Alert nếu màn hình game không hiển thị được.

// hiển thị màn hình menu

// các thao tác thực hiện trên Menu :

Đồng thời gọi luôn đối tượng MazeCanvaz để hiện thị Game

// các hàm gọi để hiển thị màn hình như trên ảnh :màn hình About, GameScore…

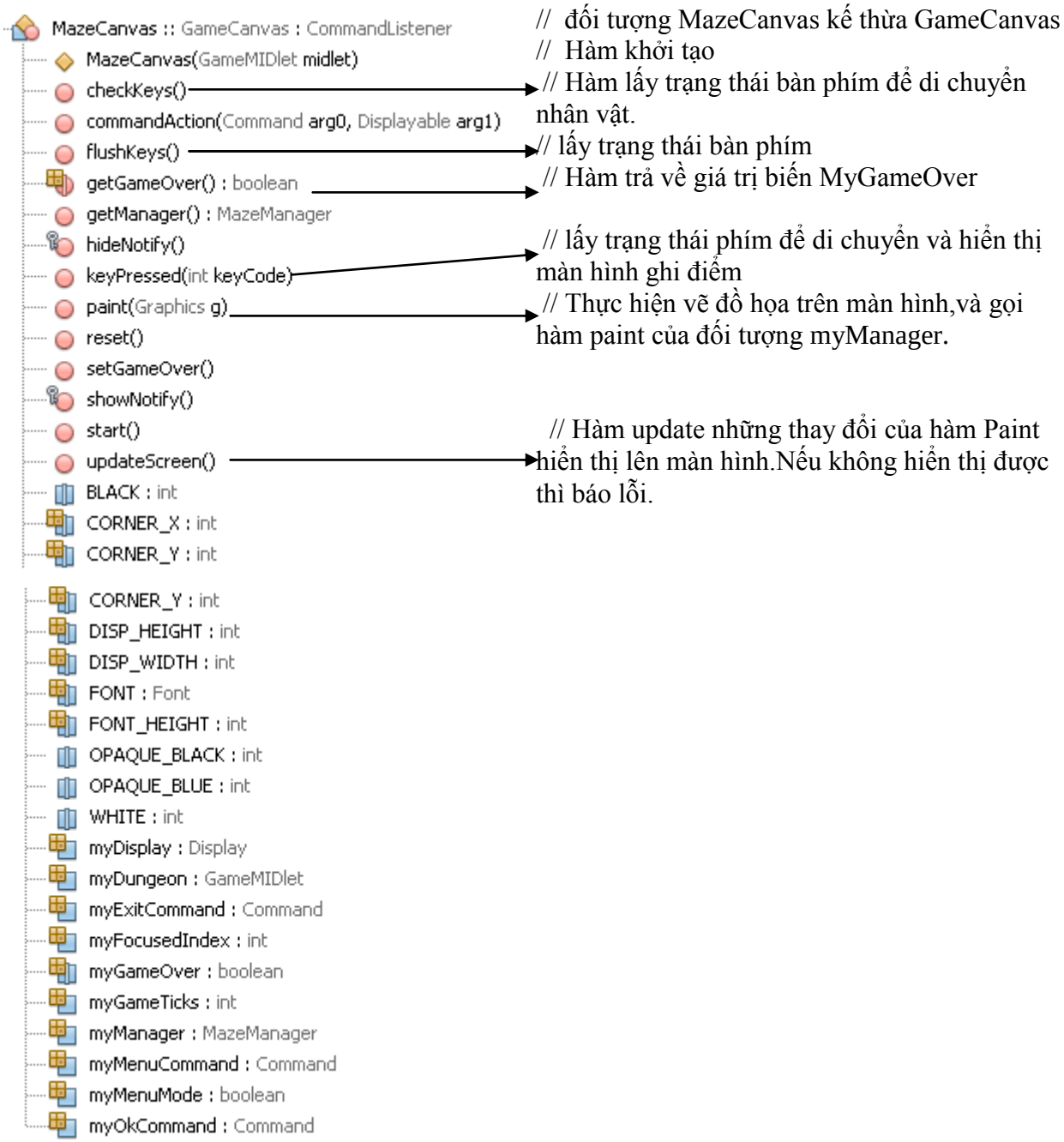
// Hàm thiết lập đối tượng mazeCanvaz

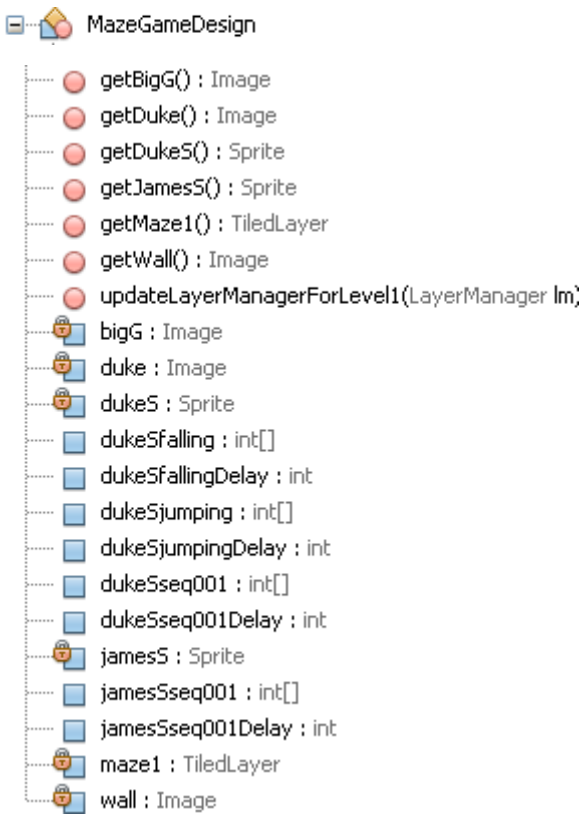
// Gọi đối tượng Display để hiện thị ra màn hình

// Hàm bắt đầu ứng dụng

//Hàm dừng chương trình để bắt các sự kiện timer.

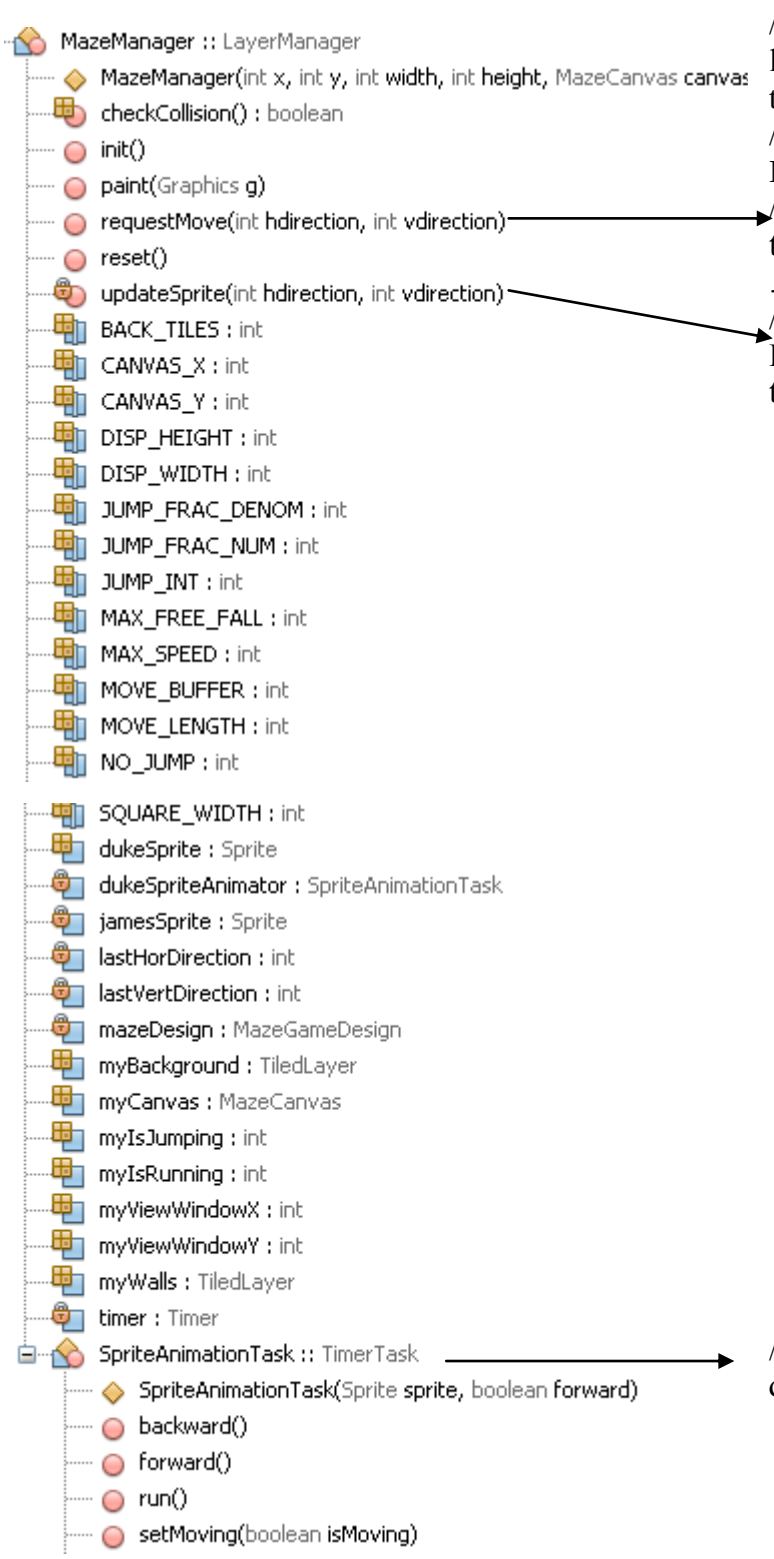
// Thiết lập hàm GameOver để thực hiện ghi điểm cho người chơi .


MazeCanvas :: GameCanvas : CommandListener
MazeCanvas(GameMIDlet midlet)
checkKeys()
commandAction(Command arg0, Displayable arg1)
flushKeys()
getGameOver() : boolean
getManager() : MazeManager
hideNotify()
keyPressed(int keyCode)
paint(Graphics g)
reset()
setGameOver()
showNotify()
start()
updateScreen()
BLACK : int
CORNER_X : int
CORNER_Y : int
CORNER_Y : int
DISP_HEIGHT : int
DISP_WIDTH : int
FONT : Font
FONT_HEIGHT : int
OPAQUE_BLACK : int
OPAQUE_BLUE : int
WHITE : int
myDisplay : Display
myDungeon : GameMIDlet
myExitCommand : Command
myFocusedIndex : int
myGameOver : boolean
myGameTicks : int
myManager : MazeManager
myMenuCommand : Command
myMenuMode : boolean
myOkCommand : Command
// đối tượng MazeCanvas kế thừa GameCanvas
// Hàm khởi tạo
// Hàm lấy trạng thái bàn phím để di chuyển nhân vật.
// lấy trạng thái bàn phím
// Hàm trả về giá trị biến MyGameOver
// lấy trạng thái phím để di chuyển và hiển thị màn hình ghi điểm
// Thực hiện vẽ đồ họa trên màn hình,và gọi hàm paint của đối tượng myManager.
// Hàm update những thay đổi của hàm Paint hiển thị lên màn hình.Nếu không hiển thị được thì báo lỗi.



// đối tượng MazeGameDesign để thực hiện các đối tượng đồ họa .

// Các hàm để load các đối tượng : Image, Sprite,TiledLayer



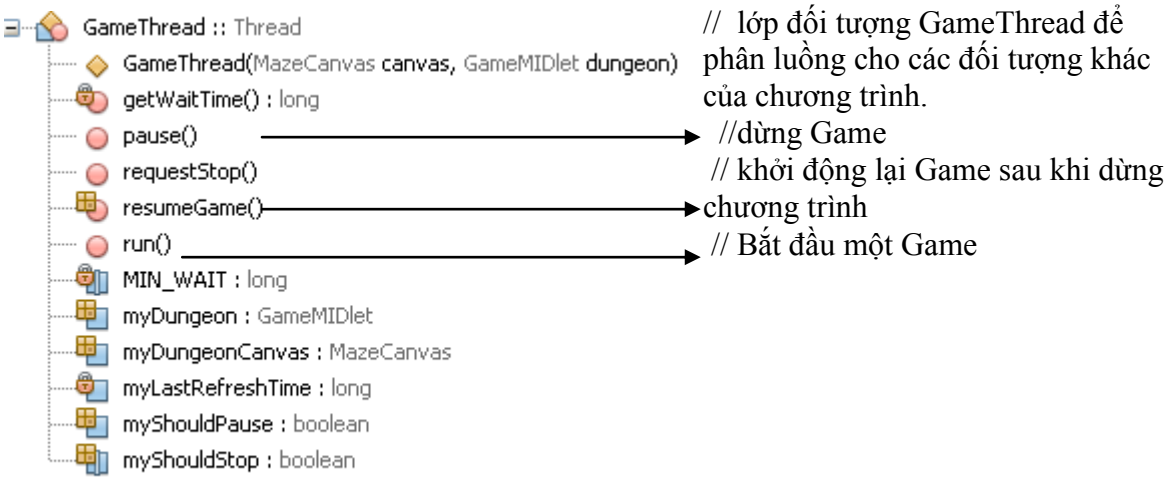
// Đối tượng MazeManager kế thừa lớp thực hiện đồ họa của chương trình

// hàm khởi tạo gọi đối tượng MazeDesign

// Hàm di chuyển nhân vật Duke trong Game, gặp xung đột thì quay lại .Gặp nhân vật Jame thì thắng .

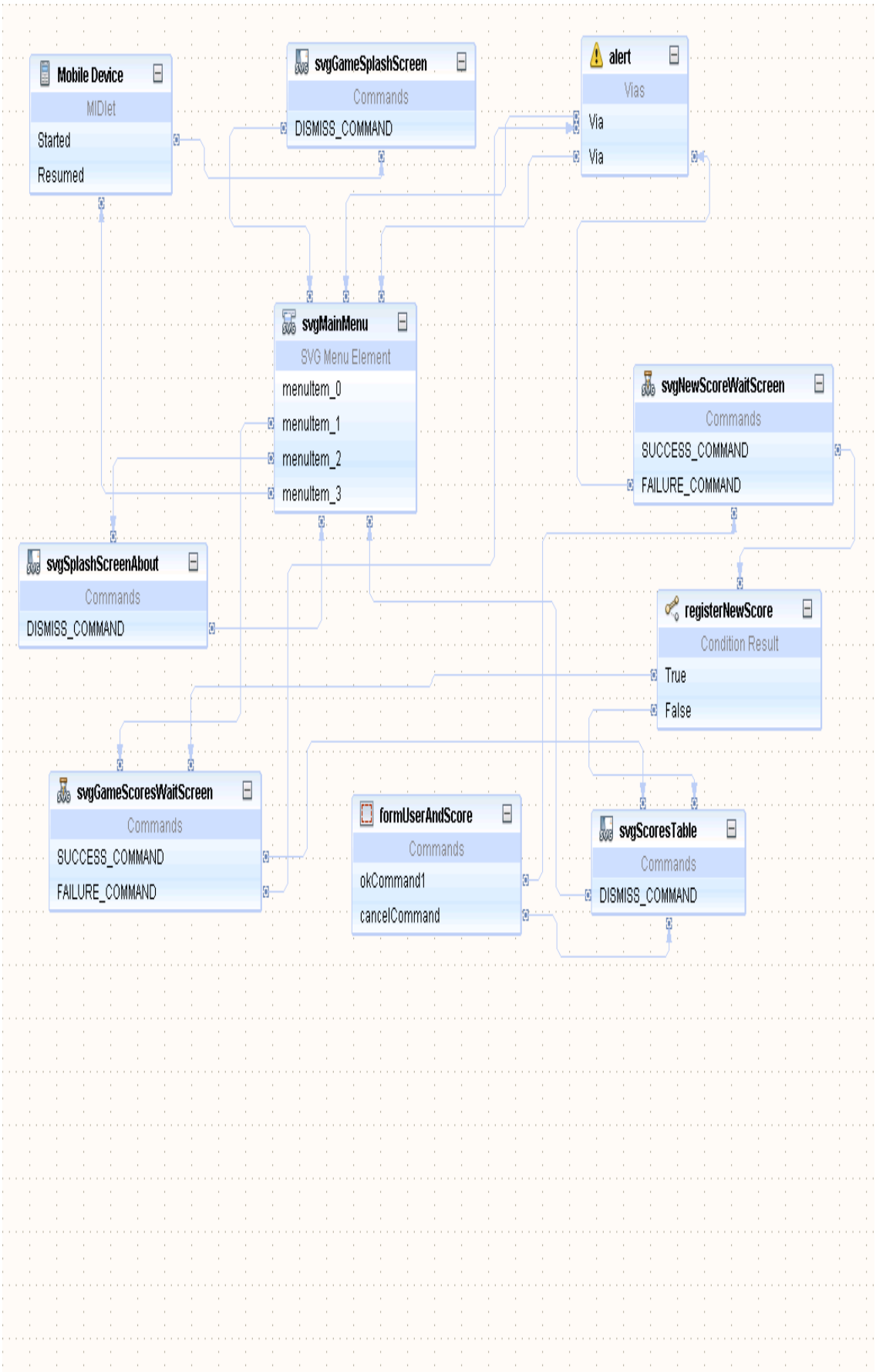
// Hàm thiết lập sự chuyển đổi của Duke .Trong khi Duke di chuyển hiển thị Sprite.

// lớp đối tượng thiết lập sự chuyển động của SpriteDuke

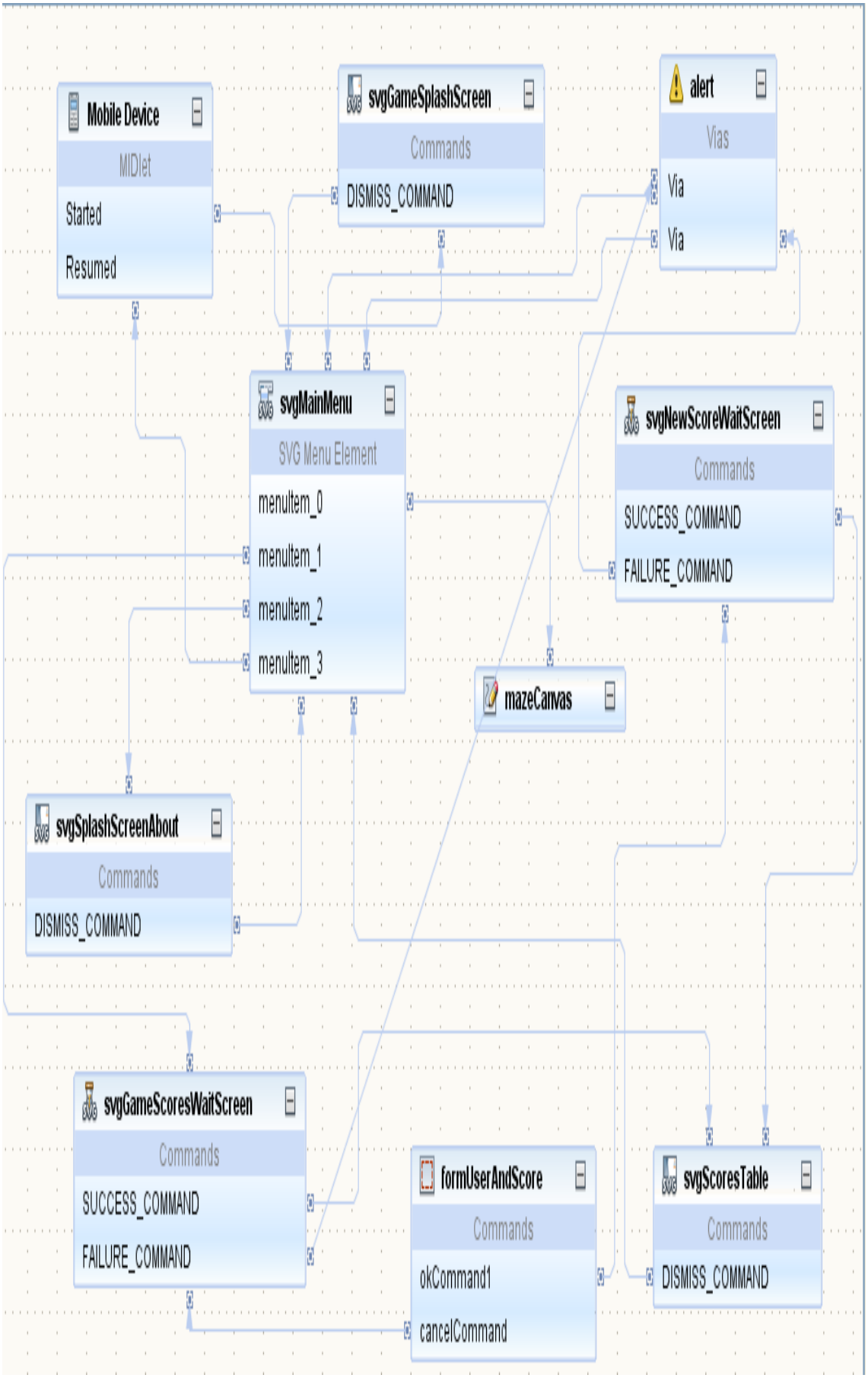


2.2. VisualMIDlet

Đây là công cụ tạo ra các ứng dụng J2ME bằng lưu đồ thao tác rất trực quan. Sau đây là cơ cấu MIDlet em sử dụng trong Game của mình.



Hình 17 Lưu đồ các luồng trong thiết kế menu game

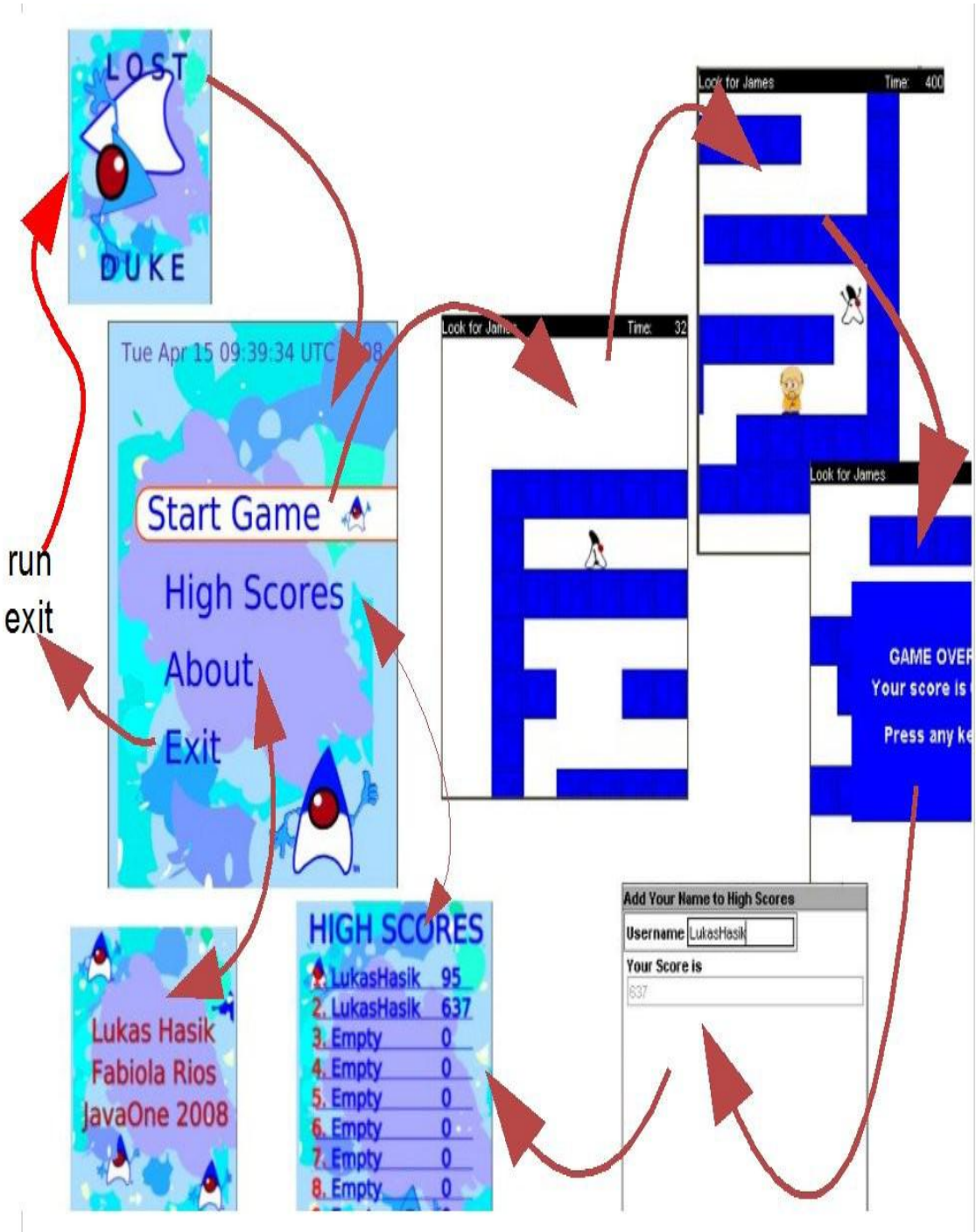


Hình 18 thiết kế luồng ứng dụng MazeCanvas

Điểm bắt đầu là đối tượng Mobile Device, sau khi chạy chương trình sơ đồ chuyển từ started đến đối tượng SplashScreen, đối tượng này thực hiện chức năng tạo một khoảng chờ và phóng một ảnh lên màn hình để quảng cáo về game. Sau một khoảng thời gian sẽ tự động chuyển đến đối tượng SgvMainmenu. Đối tượng SgvMainmenu gồm một danh sách các chức năng của chương trình như: StartGame – bắt đầu chơi, About – tự động chuyển đến đối tượng SgvFlashScreenAbout thông tin tác giả, High Score – điểm số của những người chơi cao nhất, Help – hướng dẫn phím bấm, Exit – thoát. Khi người chơi chọn một trong các chức năng này các Form tương ứng sẽ được hiện lên màn hình.

Chú ý:

Thành phần StartGame trong Menu không chuyển mạch tới đối tượng nào trong menu vì khi chọn StartGame màn hình chuyển đến đối tượng GameCanvas và bắt đầu chơi. Canvas là đối tượng giao diện mức thấp nên không cho phép thao tác trên lưu đồ. Ta phải tự tạo các chức năng trong một Canvas, nếu muốn từ Canvas có thể quay lại Menu chính phải bắt sự kiện phím bấm Back cho Canvas.



Hình 19 Hình ảnh quá trình của trò chơi :

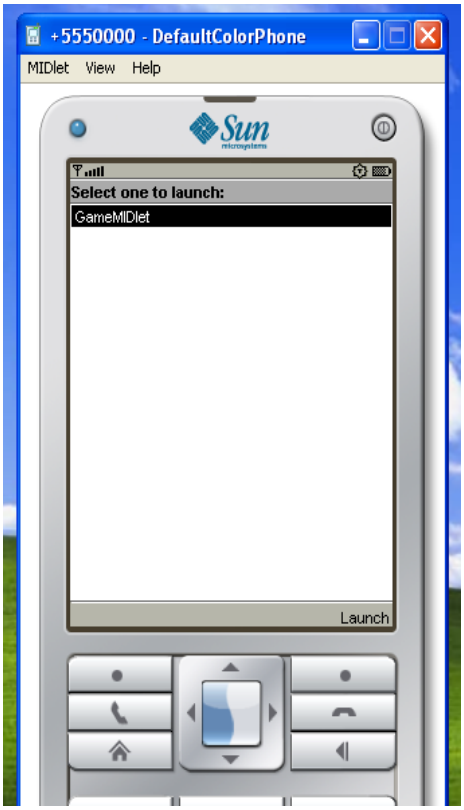
Chương 4: Kết quả đạt được

1. Môi trường cài đặt

- Ngôn ngữ cài đặt: Java là một ngôn ngữ lập trình có hiệu quả cao, cấu trúc độc lập nên các ứng dụng của nó chỉ cần viết sao cho chạy được trên máy ảo Java là có thể cài đặt và chạy tốt trên mọi hệ thống.
- Môi trường soạn thảo: NetBean IDE 6.0
- Môi trường chạy chương trình: Sun J2ME Wireless Toolkit 2.5 (WTK)

2. Chạy ứng dụng Game .

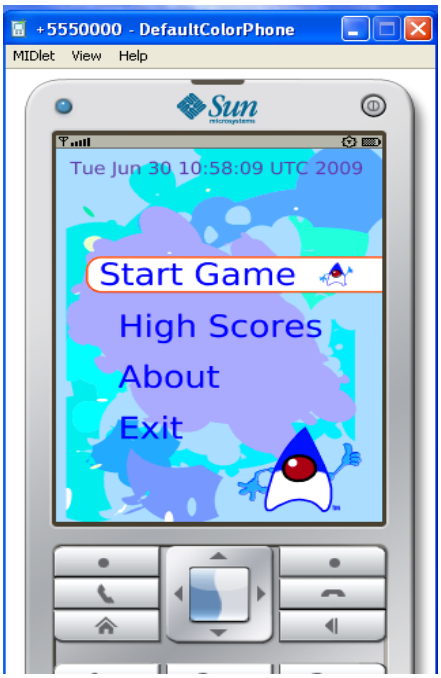
Một số hình ảnh về Game Duke & Jame



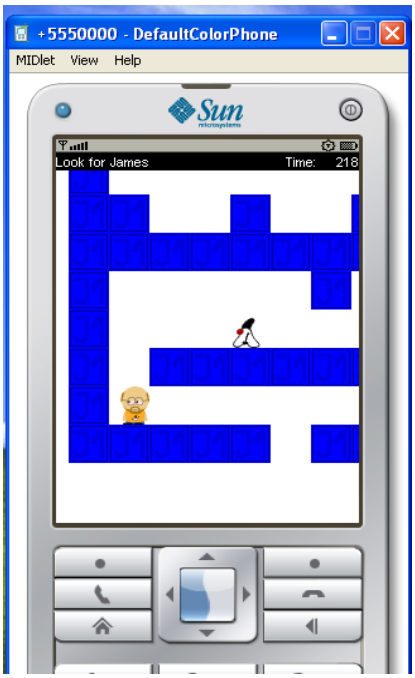
Launch game

Splash screen

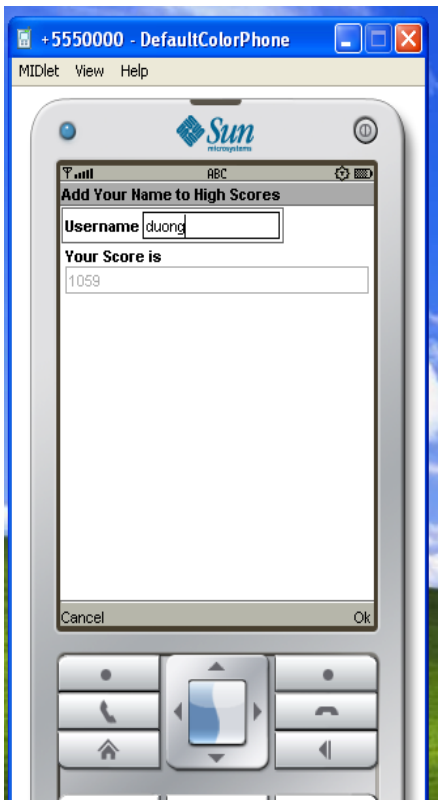
Hình 20



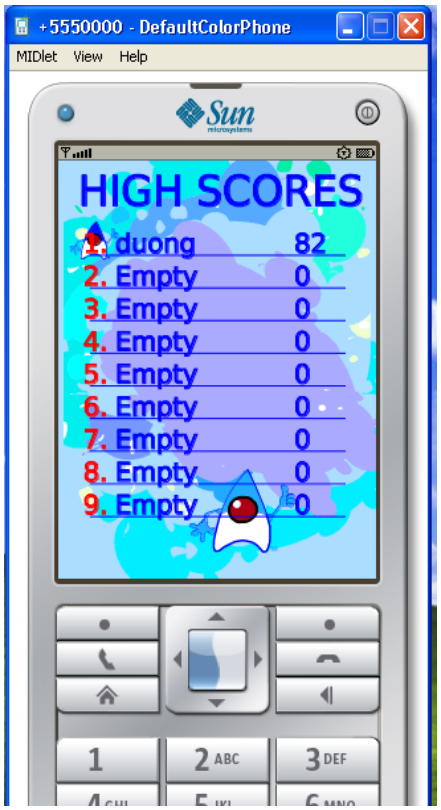
Main menu



Game running



Write Username



High Scores Table

Hình 21

3. Kết luận và hướng phát triển

3.1. Những kết quả đạt được

Sau khi làm xong bài tập thực tập "Lập trình game di động với J2ME" em đã đạt được những kết quả sau:

Về kiến thức

• Đã nắm bắt được cơ chế hoạt động của máy ảo Java trên điện thoại di động, nhờ đó hoàn thiện kỹ năng xây dựng một ứng dụng bằng ngôn ngữ J2ME.

• Học được những kiến thức cơ bản về làm game cũng như lập trình thực tế trên ngôn ngữ J2ME.

Về chương trình

• Kích thước file JAR phù hợp với hầu hết các dòng điện thoại có hỗ trợ MIDP 2.0.

• Chương trình chỉ sử dụng những thư viện cơ bản của J2ME nên có thể chạy trên nhiều dòng máy.

3.2. Những hạn chế

Bên cạnh những kết quả trên, bản thân em thấy những hạn chế cần khắc phục sau:

• Kích thước của chương trình chưa được tối ưu hóa

• Game còn đơn giản

• Chưa có âm thanh cho game

3.3. Hướng phát triển

Hướng phát triển sau này của em là nghiên cứu thêm các kỹ thuật làm game như tạo âm thanh, tạo game 3D, và tìm hiểu những công cụ mạnh hơn cho việc xây dựng đồ họa của Game

TÀI LIỆU THAM KHẢO

1. JavaVietNam.org & Nhà sách Đất Việt , *Lập trình Mobile Games bằng J2ME*, NXB Giao Thông Vận Tải.
2. Phương Lan & Hoàng Đức Hải , *Java(Tập 1)*,NXB Lao Động Xã Hội.
3. Addison Wesley Publisher, *Programming Wireless Devices with the Java™ 2 Platform, Micro Edition, Second Edition*.
4. Sams Publisher, *Java™ 2 Micro Edition Application Development*.
5. Hungry Minds, *Wireless Programming with J2ME™: Cracking the code,* Inc.
6. James White, *Java in Small Things*, Maning Publications
7. *J2ME Game Programming* – by Martin Wells
8. *Wireless Game Development in Java with MIDP.2.0* - by Ralph Barbagallo
9. *J2ME Game Development with MIDP2* – by Jason Lam