

## **Mục lục**

Mục lục.....	1
PHẦN 1 – MỞ ĐẦU .....	4
1. Lý do chọn đề tài.....	4
2. Mục tiêu.....	4
3. Phạm vi nghiên cứu.....	4
PHẦN 2 – TỔNG QUAN.....	5
PHẦN 3 – CƠ SỞ LÝ THUYẾT.....	7
CHƯƠNG 1: CÔNG NGHỆ J2ME.....	7
1.2 Các thành phần của J2ME .....	8
1.2.1 Profile .....	8
1.2.2 Configuration .....	9
1.2.3 Máy ảo java (java virtual machine) .....	10
1.3 MIDlet và Display .....	10
1.3.1 MIDlet.....	10
1.3.2 Display .....	14
1.3.2.1 Text Box .....	16
1.3.2.2 List Box .....	17
1.3.2.3 Alert .....	19
1.3.2.4 Form.....	20
1.3.2.4.1 Choice Group.....	20
1.3.2.4.2 Date Field .....	21
1.3.2.4.3 Text Field .....	21
1.3.2.4.4 Gause .....	22
1.3.2.4.5 Spacer.....	22
1.3.2.4.6 CustomItem .....	22
1.3.2.4.7 ImageItem.....	22
1.3.2.4.8 StringItem.....	23
1.3.2.4.9 Sticket .....	23
1.3.2.5 Lớp Canvas.....	23

1.3.2.6 Lớp Graphics .....	27
1.4 Xử lý sự kiện .....	33
1.4.1 Kiến thức chung .....	33
1.4.2 Command và CommandListener .....	33
1.4.3 Item và ItemStateListener .....	36
1.5 MIDlet suite .....	37
1.5.1 Java Archive File (JAR).....	37
1.5.2 Jad file .....	38
1.6 Lưu trữ dữ liệu trên thiết bị di động (RMS):.....	39
CHƯƠNG 2: MÔ HÌNH CLIENT – SERVER.....	44
2.1 Định nghĩa .....	44
2.2 Các thành phần chính của lập trình mạng Client – Server.....	44
2.2.1 Giao tiếp Client – Server.....	44
2.2.2 Giao tiếp với nhiều client .....	46
2.3 Thiết kế giải thuật Client – Server.....	47
2.3.1 Thiết kế cho Client: .....	47
2.3.1.1 Giao thức TCP.....	47
2.3.1.2 Giao thứcUDP .....	49
2.3.2 Thiết kế giải thuật cho Server.....	51
2.4 Lập trình socket trên java .....	55
PHẦN 4 – NGHIÊN CỨU THỰC NGHIỆM ÁP DỤNG CHƯƠNG TRÌNH CỔ VẤN HỌC TẬP ĐIỆN TỬ.....	58
CHƯƠNG 1 – CÁC YÊU CẦU.....	58
1.1 Yêu cầu chức năng.....	58
1.2 Yêu cầu phần cứng: .....	59
1.3 Yêu cầu phần mềm: .....	59
1.4 Môi trường thực thi (thiết bị) .....	59
CHƯƠNG 2 – THỰC THI .....	60
2.1 Thiết kế CSDL: các bảng dữ liệu .....	60
2.3 Xử lý vấn đề của bài toán .....	65

2.4 Xây dựng các giao diện và font chữ.....	68
2.4.1 Giao diện.....	68
2.4.2 Font chữ.....	69
2.5 Thiết kế các mô hình giao tiếp giữa Client – Server.....	71
2.5.1 Việc gửi nhận dữ liệu giữa Client – Server.....	71
CHƯƠNG 3 – DEMO ỨNG DỤNG.....	75
PHẦN 5 – KẾT QUẢ THỰC NGHIỆM ĐỀ TÀI.....	81
PHẦN 6 – HƯỚNG PHÁT TRIỂN.....	82
PHẦN 7 – TÀI LIỆU THAM KHẢO.....	83
PHẦN 8 – PHỤ LỤC.....	84

## **PHẦN 1 – MỞ ĐẦU**

### **1. Lý do chọn đề tài:**

Cùng với việc hệ thống giáo dục theo tín chỉ ra đời thay thế cho hệ niên chế giúp cho sinh viên chủ động hơn trong việc học tập, rút ngắn thời gian học tập tại trường đại học cũng như tập trung cho chuyên môn của mình làm nền tảng kiến thức sau khi ra trường. Việc này giúp sinh viên có nhiều sự chọn lựa hơn về môn học của mình. Và việc chọn lựa như thế nào cho phù hợp với khả năng cũng như ngành học của mình thì cần những cố vấn học tập thường là các thầy giáo bộ môn do khoa phân.

Một Giảng viên bộ môn thường chỉ chuyên về bộ môn đó, trong khi một tập thể Sinh viên lại chọn lựa theo nhiều chuyên môn khác nhau, hơn nữa Giảng viên không thể hướng dẫn hết cho từng Sinh viên trong một đợt đăng kí tín chỉ diễn ra nhanh trong vài ngày.

Yêu cầu đặt ra là làm thế nào để có thể tư vấn cho sinh viên trong mỗi đợt đăng kí một cách nhanh nhất và hiệu quả nhất. Theo đó, việc tạo ra một hệ tri thức hoàn chỉnh để tư vấn cho sinh viên qua mỗi đợt đăng kí tín chỉ và được sử dụng trên thiết bị cầm tay (điện thoại cấu hình thấp) mà hầu như sinh viên nào cũng có.

### **2. Mục tiêu**

Cung cấp cho sinh viên một công cụ hữu ích ngoài ứng dụng web trong việc lựa chọn môn học thích hợp đăng kí cho mỗi học kì cũng như biết được kết quả học tập của mình.

Xây dựng được một mô hình Client – Server đơn giản để gián sử dụng giao TCP với lập trình socket.

### **3. Phạm vi nghiên cứu**

Trong khuôn khổ đề tài được xây dựng trên mô hình Client – Server. Client là thiết bị cấu hình thấp chạy trên nền tảng J2ME và Server là máy PC chạy trên nền tảng J2SE.

## PHẦN 2 – TỔNG QUAN

Với một sinh viên, việc xem thời khóa biểu, đăng kí môn học theo mỗi đợt hoặc xem điểm sau khi kết thúc môn học hiện tại đang diễn ra dưới dạng web application, tức là thông qua server web. Đề tài này đề cập theo một hướng khác là xây dựng riêng một hệ thống Client – Server giao tiếp qua socket chạy trên môi trường thiết bị di động cấu hình thấp, trong đó ứng dụng chính là cổ vấn học tập cho sinh viên. Có 2 điểm khác biệt so với phương pháp đang sử dụng là:

- Với mục đích nghiên cứu công nghệ mới thay vì sử dụng web server
- Thiết bị cầm tay được sử dụng thay vì máy tính cá nhân có kết nối internet
- Xây dựng thêm hệ tư vấn học tập cho sinh viên

Đề tài này là kết hợp của 2 nền tảng công nghệ cùng với việc xây dựng một hệ chuyên gia tư vấn đơn giản

Các nội dung chính được nêu trong luận văn này bao gồm:

- ❖ **Phần mở đầu:** Dẫn dắt đề tài, về nội dung cần nghiên cứu trong toàn bộ luận văn.
- ❖ **Phần cơ sở lý thuyết:** bao gồm những lý thuyết được áp dụng để phát triển ứng dụng của đề tài. Công nghệ J2ME, hệ thống client – server.
  - **Chương 1:** J2ME: Giới thiệu về công nghệ J2ME
    - + Các thành phần chính của công nghệ J2ME.
    - + Chạy một chương trình MIDlet.
    - + Display : bao gồm các thành phần đồ họa bậc cao và bậc thấp: Canvas, Graphics.
    - + Xử lý sự kiện trong J2ME.
    - + Lưu trữ dữ liệu RMS.
  - **Chương 2:** Giới thiệu mô hình Client – Server

Giới thiệu các kiến thức về lập trình mạng Client – Server trên java. Lập trình socket với các giao thức: TCP và UDP trên client – server.

- + Định nghĩa
- + Ứng dụng của mô hình Client – Server
- + Lập trình socket
- + Lập trình socket cho Client bao gồm 2 giao thức là TCP và UDP
- + Lập trình socket cho Server
- + Kết nối và transfer dữ liệu giữa Client và Server

❖ **Phần nghiên cứu thực nghiệm:**

- Xây dựng ứng dụng là chương trình cố vấn học tập điện tử dựa trên mô hình Client – Server tư vấn cho Sinh viên những môn học nên đăng kí trong một đợt đăng kí học phần.
  - + Phân tích yêu cầu đề tài
  - + Thiết kế Cơ sở dữ liệu quan hệ
  - + Thiết kế các chức năng chương trình
  - + Thực thi chương trình
  - + Demo ứng dụng

❖ **Phần kết quả thực nghiệm:** Kết quả đạt được sau khi kết thúc nghiên cứu đề tài.

❖ **Phần Hướng phát triển:** Nêu ra những tồn tại, những vấn đề còn chưa giải quyết được của yêu cầu đề tài đặt ra và hướng phát triển, xây dựng đề tài hoàn thiện hơn.

❖ **Phần tài liệu tham khảo:** Những tài liệu dùng để nghiên cứu đề tài. Bao gồm phần lý thuyết và thực hành áp dụng.

❖ **Phần phụ lục:** Những từ viết tắt dùng trong đề tài.

## PHẦN 3 – CƠ SỞ LÝ THUYẾT

### CHƯƠNG 1: CÔNG NGHỆ J2ME

#### 1.1 Tổng quan về công nghệ J2ME

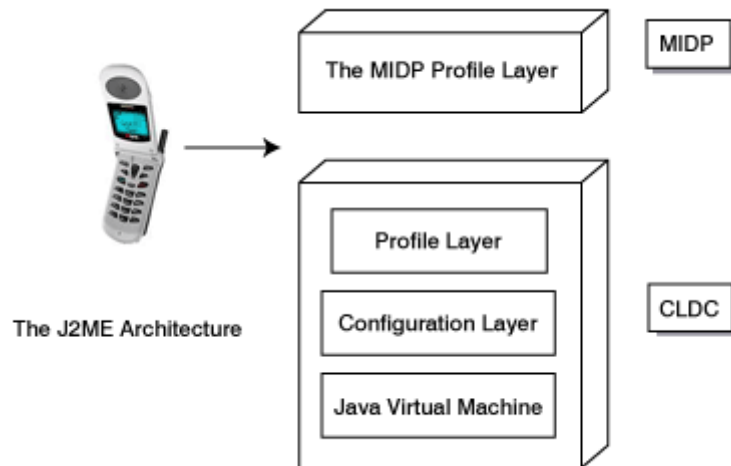
Nền tảng J2ME ra đời thừa hưởng từ những nền tảng trước đã phát triển mạnh mẽ của java là J2SE và J2EE. Điểm căn bản đặc trưng của java là viết một lần và chạy trên môi trường (Có hỗ trợ máy ảo java).

Công nghệ J2ME xây dựng nên các ứng dụng cho điện thoại di động có cấu hình thấp. Các dòng máy như Nokia, Samsung... đều áp dụng công nghệ J2ME.

Ứng dụng nhắn tin, danh bạ, các ứng dụng tra cứu thông tin, lưu trữ và chia sẻ dữ liệu, wifi, wireless...



*H2.1 : Các thiết bị J2ME hỗ trợ*



*H1.1 Nền tảng công nghệ J2ME*

## 1.2 Các thành phần của J2ME

### 1.2.1 Profile

Profile là một khái niệm mở rộng của Configuration. Profile định nghĩa các thư viện giúp lập trình viên phát triển các ứng dụng cho một dạng thiết bị nào đó. Ta có các profile như : MIDP, PDA...

MIDP định nghĩa các hàm API cho các thành phần giao diện, nhập liệu và xử lý sự kiện, lưu trữ, kết nối mạng, quản lý thời gian... phù hợp với màn hình hiển thị và khả năng xử lý của các thiết bị di động. Profile được định nghĩa trên nền tảng CLDC.

MIDP cung cấp các thư viện giao diện người dùng có các đặc trưng :

- Đủ bộ nhớ để chạy các ứng dụng của MIDP
- Độ phân giải màn hình tối thiểu 96 x 56 pixels, màu hoặc trắng đen
- Bộ phím : Keypad, keyboard, hoặc touch screen



- Kết nối mạng không dây 2 chiều ( bluetooth)

Với bộ MIDP 3.0 có các tính năng :

- Hỗ trợ tốt cho thiết bị có màn hình rộng
- Cho phép MIDlets tương tác với Display thứ 2 (khác current display)
- Nâng cao hiệu suất cho các thiết bị
- Hỗ trợ bảo mật cho lưu trữ RMS
- Removable/Remote RMS
- IPv6
- Hỗ trợ nhiều giao thức mạng cho cùng một thiết bị

### 1.2.2 Configuration

Không cung cấp giao diện người dùng bao gồm CLDC và CDC: Được xem như là một dạng máy ảo đặc trưng của java có các đặc trưng về quản lý bộ nhớ, độ phân giải màn hình, giao thức kết nối. Có 2 bộ Configuration trong J2ME.

- **CLDC (Connected Limited Device Configuration - Cấu hình thiết bị kết nối giới hạn):** được thiết kế để nhắm vào thị trường các thiết bị cấp thấp (low-end), các thiết bị này thông thường là máy điện thoại di động và PDA với khoảng 512 KB bộ nhớ. Vì tài nguyên bộ nhớ hạn chế nên CLDC được gắn với Java không dây (Java Wireless ), dạng như cho phép người sử dụng mua và tải về các ứng dụng Java, ví dụ như là Midlet.
  - Tổng bộ nhớ : 160 - 512Kb
  - Bộ xử lý : 16 - 32 bit
  - Tiêu thụ năng lượng thấp, dùng pin
  - Kết nối mạng với băng thông giới hạn
- **CDC (Connected Device Configuration - Cấu hình thiết bị kết nối):** CDC được đưa ra nhằm đến các thiết bị có tính năng mạnh hơn dòng thiết bị thuộc CLDC nhưng vẫn yếu hơn các hệ thống máy để bàn sử dụng J2SE. Những thiết

bị này có nhiều bộ nhớ hơn (thông thường là trên 2Mb) và có bộ xử lý mạnh hơn. Các sản phẩm này có thể kể đến như các máy PDA cấp cao, điện thoại web, các thiết bị gia dụng trong gia đình ...

### **1.2.3 Máy ảo java (java virtual machine)**

Với tính năng đã được đề cập trước đó là môi trường trung gian giữa ứng dụng và máy thực (môi trường thực thi chính của ứng dụng) và mang tính đặc trưng của ngôn ngữ java.

Với CDC, máy ảo java có cùng đặc tính với J2SE. Tuy nhiên với CLDC Sun (giờ đã được sát nhập với Oracle) đã phát triển riêng một dạng máy ảo chuyên biệt dành cho các thiết bị có cấu hình thấp là K Virtual machine, gọi tắt là KVM.

KVM có đặc tính:

- Chỉ cần 40 – 80 Kb bộ nhớ
- Chỉ đòi hỏi tối thiểu 20 – 40 Kb bộ nhớ động
- Có thể chạy với bộ vi xử lý của thiết bị 16 bit và xung nhịp 25 MHz
- Nếu chương trình được biên dịch với CDC, chương trình sẽ chạy máy ảo truyền thống JVM (Java Virtual Machine) và mang các đặc tính như J2SE
- Nếu chương trình được biên dịch với CLDC, chương trình sẽ chạy máy ảo KVM (K Virtual Machine) và mang các đặc tính của J2ME.
- Vì tính đặc thù nhỏ gọn của trình biên dịch CLDC nên một số chương trình viết bằng MIDP sẽ không chạy được trong môi trường J2SE và ngược lại.

## **1.3 MIDlet và Display**

### **1.3.1 MIDlet**

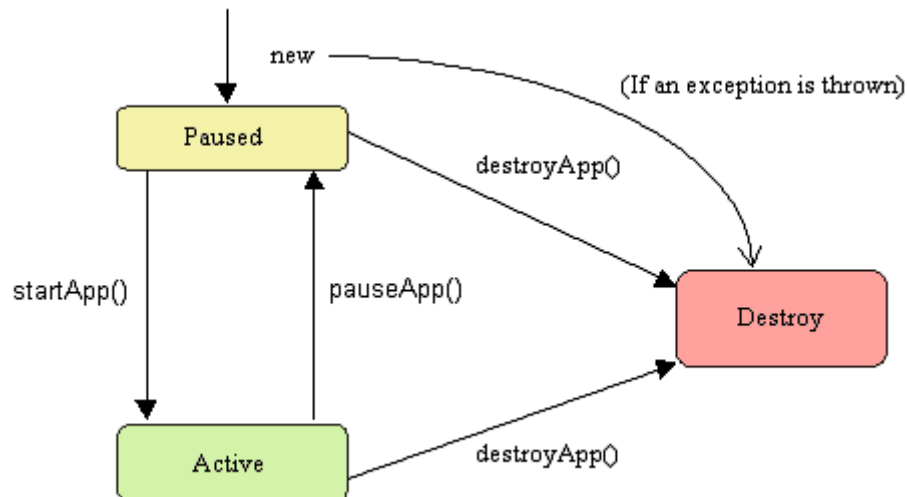
MIDlet được tạo ra trong một ứng dụng J2ME để thực thi chương trình. MIDP không bắt đầu từ phương thức *static main* (giống như bình thường), thay vào đó nó sử dụng một MIDlet.

MIDlet : MIDP không bắt đầu chạy từ phương thức **static main** cũng không gọi phương thức **System.exit()** để thoát ứng dụng. Thay vào đó MIDP sử dụng MIDlet.

Mỗi ứng dụng J2ME phải kế thừa lớp **javax.microedition.midlet.MIDlet** để cho phép quản lý ứng dụng.

Các package hỗ trợ cho MIDlet

- *java.lang*
- *java.util*
- [javax.microedition.io](#)
- *javax.microedition.lcdui*
- *javax.microedition.lcdui.game*
- *javax.microedition.media*
- *javax.microedition.media.control*
- *javax.microedition.midlet*
- *javax.microedition.pki*
- *javax.microedition.rms*



*H1.3.1 Vòng đời 1 MIDlet*

Ví dụ một MIDlet

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 * @author Than Dang
 */
public class MidletHelloWorld extends MIDlet {
    private TextBox textbox;
    public MidletHelloWorld() {
        textbox = new TextBox("", "Midlet Hello World", 25, 0);
    }
    public void startApp() {
        Display.getDisplay(this).setCurrent(textbox);
    }
    public void pauseApp() {
    }
    public void destroyApp(boolean unconditional) {
    }
}
```

Kết quả của chương trình Hello word này là:



*H1.3.2 – Kết quả thực hiện hello word*

- ❖ Phát biểu *Import* : Khai báo các thư viện cần thiết cho ứng dụng.
  - ❖ Dòng khai báo lớp : Khai báo tên lớp, thường là mặc định khi tạo lớp bằng tay.
- Hàm khởi tạo (Constructor) : luôn cùng tên với MIDlet. Nó chỉ được gọi một lần duy nhất bởi bộ quản lý ứng dụng trong suốt vòng đời của MIDlet vào thời điểm MIDlet được nạp bởi bộ quản lý ứng dụng. Ta chỉ muốn thực thi một lần duy nhất

vào thời điểm chương trình khởi động như khởi tạo giá trị ban đầu cho các biến, khai báo và xác lập các thành phần giao diện.

- ❖ **StartApp()** : hàm sẽ được gọi bởi bộ quản lý ứng dụng khi MIDlet chuyển từ trạng thái paused (tạm ngừng) sang trạng thái hoạt động. Hàm này có thể được gọi đi gọi lại nhiều lần vì trong suốt vòng đời của mình MIDlet có thể chuyển đổi qua lại nhiều lần giữa trạng thái tạm ngừng và trạng thái hoạt động.
- ❖ **Pauseapp()** : Hàm này sẽ được gọi bởi bộ quản lý ứng dụng khi MIDlet chuyển từ trạng thái hoạt động sang trạng thái paused. Ví dụ khi bạn đang chơi game thì đột nhiên có tin nhắn hoặc cuộc gọi đến. Ứng dụng J2ME phải tạm ngừng lại để cho phép người dùng nhận cuộc gọi hoặc nhắn. Vào thời điểm ứng dụng chuyển từ trạng thái hoạt động sang tạm ngừng, hàm này sẽ được gọi.
- ❖ **DestroyApp()** : Hàm này được gọi bởi bộ quản lý ứng dụng khi ứng dụng chuẩn bị thoát. Tất cả những tài nguyên nào mà ứng dụng đang nắm giữ cũng cần được giải phóng vào thời điểm này. Bạn sẽ đặt các mã lệnh giải phóng tài nguyên ở đây.
- ❖ Ngoài 3 phương thức kể trên ta còn có thêm những phương thức khác nữa là : **resumeRequest()**, **notifyPause()**, **notifyDestroy()**.

### 1.3.2 Display

Mỗi MIDlet tham chiếu đến một đối tượng Display. Đối tượng này cung cấp các thông tin về màn hình và các phương thức cần thiết cho việc hiển thị các đối tượng lên màn hình thiết bị. Chức năng của lớp Display là quyết định xem thành phần nào sẽ được hiển thị lên màn hình thiết bị.

Các hàm và phương thức chính trong lớp Display

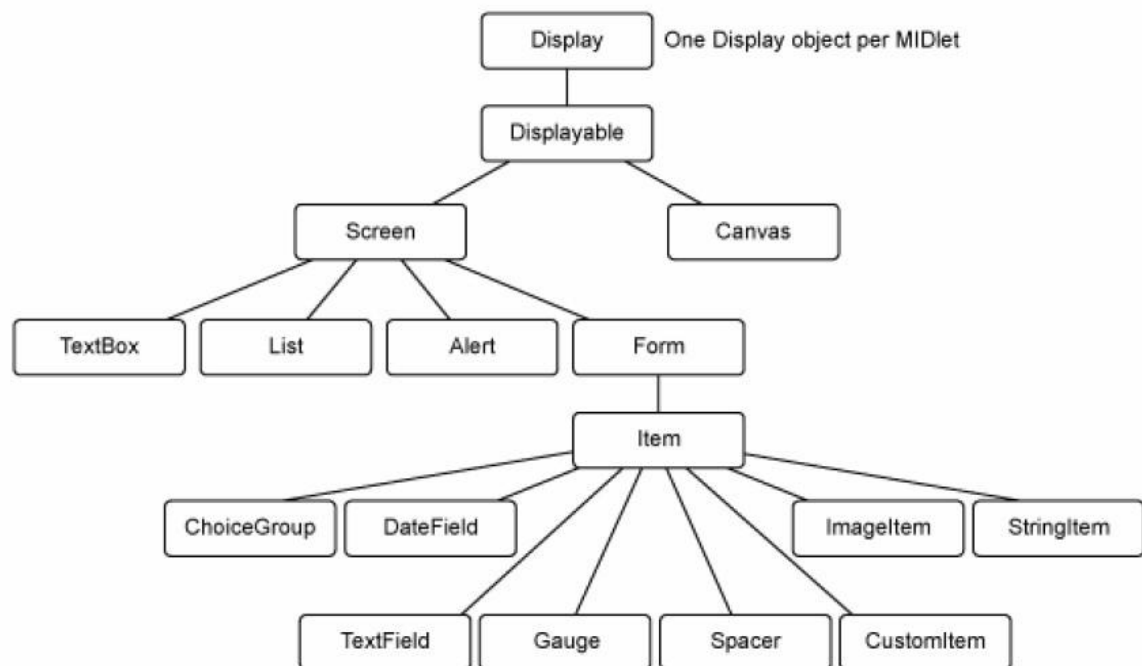
- *static Display getDisplay(MIDlet m):*
- *Displayable getCurrent()*
- *void setCurrent(Alert alert, Displayable nextDisplayable)*
- *void setCurrent(Displayable nextDisplayable)*

- *boolean isColor()*
- *int numColors()*
- *void callSerially(Runnable r)*

Ta sẽ tiến hành tạo một ví dụ về Display để dễ hiểu: Đoạn lệnh bên dưới tạo một *Display*

```
public class DisplayStats extends MIDlet
{
    private Display display; // Reference to Display object
    // MIDlet constructor
    public DisplayStats()
    {
        display = Display.getDisplay(this);
        ...
    }
    ...
}
```

**Các thành phần của Display** : các thành phần đồ họa mức cao và các thành phần đồ họa mức thấp.



### H1.3.3. Các thành phần đồ họa mức cao

#### 1.3.2.1 Text Box

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()*.

Các hàm trong lớp *TextBox* : *javax.microedition.lcdui.TextBox*

Phương thức	Mô tả
<b><i>TextBox</i></b> ( <i>String title, String text, int maxSize, int constraints</i> )	Hàm khởi tạo với các tham số
<i>void delete</i> ( <i>int offset, int length</i> )	Xóa textbox
<i>void insert</i> ( <i>String src, int position</i> )	Chèn ký tự vào textbox



<i>void insert(char[] data, int offset, int length, int position)</i>	Chèn dãy kí tự vào textbox theo vị trí định sẵn
<i>void setChars(char[] data, int offset, int length)</i>	Định lại dãy kí tự của mảng
<i>int getChars(char[] data)</i>	Lấy chuỗi theo mảng
<i>String getString()</i>	Lấy chuỗi
<i>void setString(String text)</i>	Định lại chuỗi
<i>int getConstraints()</i>	Lấy đối tượng Constraints
<i>void setConstraints(int constraints)</i>	Định đối tượng Constraints
<i>int getMaxSize()</i>	Lấy kích thước lớn nhất
<i>int setMaxSize(int maxSize)</i>	Định kích thước lớn nhất
<i>int getCaretPosition()</i>	Trả về vị trí nhập textbox
<i>int size()</i>	Kích thước hiện có trong textbox

### 1.3.2.2 List Box

Lớp List (danh sách) là một Screen chứa danh sách các lựa chọn chẳng hạn như các radio button. Người dùng có thể tương tác với danh sách và chọn một hay nhiều item. MIDP định nghĩa ba loại danh sách:

IMPLICIT - Loại danh sách này tạo ra một thông báo (notify) ngay lập tức cho ứng dụng nếu nó đã được đăng ký một đối tượng CommandListener. Mục đang focus sẽ được lựa chọn.

- EXCLUSIVE - Loại danh sách này chỉ cho phép lựa chọn một mục đơn. Ứng dụng sẽ được thông báo và danh sách phải được kiểm tra sau đó để xác định mục nào đã được chọn.
- MULTIPLE - Loại danh sách này cho phép lựa chọn nhiều mục và bật tắt trạng thái khi lựa chọn. Ứng dụng sẽ không được thông báo và danh sách sẽ được kiểm tra sau để xác định mục nào được chọn.

Ta có các phương thức của lớp `ListBox` : `javax.microedition.lcdui.ListBox`

Phương thức	Mô tả
<b>List</b> (String title, int listType)	Hàm khởi tạo
<b>List</b> (String title, int listType, String[] stringElements, Image[] imageElements)	Hàm khởi tạo có thêm thông số
int <b>append</b> (String stringPart, Image imagePart)	Thêm một đối tượng vào list kèm theo imageicon
void <b>delete</b> (int elementNum)	Xóa một item
void <b>insert</b> (int elementNum, String stringPart, Image imagePart)	Chèn một phần tử vào vị trí xác định
void <b>set</b> (int elementNum, String stringPart, Image imagePart)	Gán giá trị cho phần tử ở vị trí xác định
String <b>getString</b> (int elementNum)	Lấy giá trị phần tử ở vị trí xác định
Image <b>getImage</b> (int elementNum)	Lấy biểu tượng hình ảnh của phần tử
int <b>getSelectedIndex</b> ()	Lấy vị trí phần tử được chọn
void <b>setSelectedIndex</b> (int elementNum, boolean selected)	MULTIPLE: Gán giá trị được chọn hay không ( <i>selected</i> ) cho phần tử ở vị trí xác định. EXCLUSIVE, IMPLICIT: Gán giá trị cho phần tử ở vị trí xác định là được chọn (không phụ thuộc vào <i>selected</i> )
int <b>getSelectedFlags</b> ( boolean[] selectedArray_return)	Lưu thông tin kết quả lựa chọn vào mảng
Void <b>setSelectedFlags</b> (boolean[] selectedArray)	Gán kết quả đối tượng chọn cho List
boolean <b>isSelected</b> (int elementNum)	Kiểm tra xem phần tử ở vị trí xác định có

	được chọn không.
<i>int size()</i>	Định nghĩa kích thước của List

### 1.3.2.3 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.

Các hàm của lớp Alert : *javax.microedition.lcdui.Alert*

Phương thức	Mô tả
<b><i>Alert(String title)</i></b>	Tạo đối tượng Alert với kiểu mặc định của thiết bị
<b><i>Alert(String title, String alertText, Image alertImage, AlertType, alertType)</i></b>	Tạo đối tượng Alert với tên, nội dung và loại Alert
<b><i>Image getImage()</i></b>	Lấy đối tượng Image của Alert
<b><i>void setImage(Image img)</i></b>	Gán đối tượng Image cho Alert
<b><i>String getString()</i></b>	Lấy nội dung thông báo cho Alert
<b><i>void setString(String str)</i></b>	Gán nội dung thông báo cho Alert
<b><i>int getDefaultTimeout()</i></b>	Lấy thời gian Alert được phép hiển thị mặc định của thiết bị
<b><i>int getTimeout()</i></b>	Lấy thời gian Alert được phép hiển thị
<b><i>void setTimeout(int time)</i></b>	Gán giá trị thời gian cho Alert được phép hiển thị tính theo mili giây
<b><i>AlertType getType()</i></b>	Lấy kiểu của Alert
<b><i>void setType(AlertType type)</i></b>	Gán kiểu cho Alert

### 1.3.2.4 Form

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.

Thẻ hiện của Form là các Item, bao gồm : ChoiceGroup, DateField, TextField, Gauge, Spacer, CustomItem ImageItem và String Item

Các phương thức của đối tượng Form

Phương thức	Mô tả
<b>Form</b> ( <i>String title</i> )	Hàm khởi tạo
<b>Form</b> ( <i>String title, Item[] items</i> )	Hàm khởi tạo có tên Form và Items
<i>int</i> <b>append</b> ( <i>Image img</i> )	Thêm một đối tượng Image vào Form
<i>int</i> <b>append</b> ( <i>Item item</i> )	Thêm một đối tượng Item vào Form
<i>int</i> <b>append</b> ( <i>String str</i> )	Thêm một đối tượng String vào Form
<i>void</i> <b>delete</b> ( <i>int itemNum</i> )	Xóa một Item
<i>void</i> <b>insert</b> ( <i>int itemNum, Item item</i> )	Thêm một đối tượng Item có đánh chỉ số
<i>Item</i> <b>get</b> ( <i>int itemNum</i> )	Lấy một Item theo chỉ số
<i>void</i> <b>set</b> ( <i>int itemNum, Item item</i> )	Đặt chỉ số cho 1 Item trong Form
<i>void</i> <b>setItemStateListener</b> ( <i>ItemStateListener iListener</i> )	Đặt listener cho Item
<i>int</i> <b>size</b> ()	Hàm tính kích thước đối tượng

#### 1.3.2.4.1 Choice Group

*public class ChoiceGroup extends Item Implements Choice*

*ChoiceGroup* cung cấp một nhóm các *radio-button* hay *checkbox* cho phép lựa chọn đơn hay lựa chọn nhiều.

Giao diện *Choice* (cũng được lớp *List* sử dụng) định nghĩa các phương thức có thể được dùng để xử lý danh sách các item. Danh sách các item là một chuỗi và cũng có thể bao gồm cả hình ảnh. Ứng dụng có thể chèn (*insert*), thêm vào cuối (*append*), và xóa (*delete*) các item sau khi đối tượng *Choice* đã được tạo. Các item này trong danh sách sẽ được tham chiếu bởi số chỉ mục (bắt đầu từ 0).

*ChoiceGroup* có thể là EXCLUSIVE hay MULTIPLE. Người dùng có thể chọn một mục hoặc nhiều mục. Chế độ IMPLICIT (được hỗ trợ trong List Screen) không được *ChoiceGroup* hỗ trợ. Do đó *CommandListener* sẽ không được gọi khi người dùng lựa chọn.

#### 1.3.2.4.2 Date Field

Thành phần *DateField* cung cấp một phương tiện trực quan để thao tác đối tượng *Date* được định nghĩa trong *java.util.Date*. Khi tạo một đối tượng *DateField*, bạn cần chỉ rõ là người dùng chỉ có thể chỉnh sửa ngày, chỉnh sửa giờ hay đồng thời cả hai. Các phương thức dựng của lớp *DateField* gồm:

*DateField(String label, int mode)*

*DateField(String label, int mode, TimeZone timeZone)*

Các mode tương ứng của lớp *DateField* gồm:

*DateField.DATE\_TIME*: cho phép thay đổi ngày giờ

*DateField.TIME*: chỉ cho phép thay đổi giờ

*DateField.DATE*: chỉ cho phép thay đổi ngày

#### 1.3.2.4.3 Text Field

*public class Gauge extends Item*

Lớp *Gauge* cung cấp một hiển thị thanh (bar display) của một giá trị số học. *Gauge* có thể có tính tương tác hoặc không. Nếu một *gauge* là tương tác thì người dùng có thể thay đổi giá trị của tham số qua *gauge*. *Gauge* không tương tác chỉ đơn thuần là để hiển thị.

#### 1.3.2.4.4 Gause

Dùng để định vị trí cho các thành phần UI bằng cách đặt một số khoảng trống giữa chúng. Thành phần này không thể nhìn thấy được.

#### 1.3.2.4.5 Spacer

Dùng để định vị trí cho các thành phần UI bằng cách đặt một số khoảng trống giữa chúng. Thành phần này không thể nhìn thấy được.

#### 1.3.2.4.6 CustomItem

CustomItem là một lớp trừu tượng cho phép việc tạo ra các lớp con có các thuộc tính của riêng chúng, tương tác của riêng chúng, và cơ chế thông báo của riêng chúng. Nếu bạn yêu cầu một thành phần UI khác với thành phần được cung cấp sẵn, bạn có thể tạo ra lớp con của CustomItem rồi thêm vào Form.

#### 1.3.2.4.7 ImageItem

Một item chứa một ảnh! Cũng như StringItem, lớp Form cung cấp một phương thức rút gọn cho việc thêm một ảnh mới: `append(Image image)`.

Các phương thức dựng cho lớp Image và ImageItem

*Image createImage(String name)*

*Image createImage(Image source)*

*Image createImage(byte[] imageData, int imageOffset, int imageLength)*

*Image createImage(int width, int height)*

*Image createImage(Image image, int x, int y, int width,  
int height, int transform)*

*Image createImage(InputStream stream)*

*Image createRGBImage(int[] rgb, int width, int height,  
boolean processAlpha)*

*ImageItem(String label, Image img, int layout, String altText)*

#### 1.3.2.4.8 StringItem

Một nhãn không thể thay đổi bởi người dùng. Item này có thể chứa tiêu đề văn bản, cả hai nội dung này đều có thể là null đóng vai trò là chỗ chứa – placeholder. Lớp Form cung cấp một cách viết ngắn gọn khi thêm một StringItem cùng tiêu đề  
rỗng: *append(String text) – DateField*: Cho phép người dùng nhập vào ngày tháng/thời gian theo 3 định dạng: DATE, TIME, hoặc DATE\_TIME – *TextField*: Tương tự như Textbox.

#### 1.3.2.4.9 Sticket

Một màn hình có thể có một ticker là một chuỗi văn bản chạy liên tục trên màn hình. Hướng và tốc độ là do thực tế qui định. Nhiều màn hình có thể chia sẻ cùng một ticker. Gọi một Sticket:

```
private String str =  
    "This text keeps scrolling until the demo stops...";  
private Ticker ticker = new Ticker(str);
```

#### 1.3.2.5 Lớp Canvas

Lớp Canvas thuộc đồ họa mức thấp của J2ME là được sử dụng thực sự hiệu quả trong việc tạo đối tượng khung, hỗ trợ cho hầu hết các game được thiết kế trong J2ME.

##### ❖ Tạo đối tượng Canvas

Tạo một lớp thừa kế từ lớp Canvas và được thể hiện bởi lớp *Displayable*

```
class AnimationCanvas extends Canvas  
    implements CommandListener  
{  
    private Command cmExit;    // Exit midlet  
    ...  
    cmExit = new Command("Exit", Command.EXIT, 1);  
    addCommand(cmExit);
```

```
setCommandListener(this);  
...  
protected void paint(Graphics g)  
{  
...  
}  
}  
AnimationCanvas canvas = new AnimationCanvas(this);  
display.setCurrent(canvas);
```

❖ **Vẽ lên đối tượng Canvas :**

Một lớp **Displayable** là một phần tử UI có thể được thể hiện ra trên màn hình của thiết bị trong khi lớp **Display** trừu tượng chức năng hiển thị của một màn hình thiết bị thực tế. Lớp **Displayable** cung cấp các phương thức lấy về thông tin màn hình và đưa ra hay thay đổi phần tử UI hiện hành mà bạn muốn được hiển thị. Vì thế, một MIDlet trình bày một phần tử UI **Displayable** trên một **Display** bằng việc sử dụng phương thức **setCurrent(Displayable current)** của lớp **Display**.

Ví dụ : một canvas vẽ một hình vuông màu đen ở giữa màn hình thiết bị

```
import javax.microedition.lcdui.Canvas;  
import javax.microedition.midlet.MIDlet;  
import javax.microedition.lcdui.Display;  
import javax.microedition.lcdui.Graphics;  
  
public class CanvasExample  
    extends MIDlet {  
    Canvas myCanvas;  
  
    public CanvasExample() {
```



```
myCanvas = new MyCanvas();
}

public void startApp() {
    Display display = Display.getDisplay(this);

    // remember, Canvas is a Displayable so it can
    // be set on the display like Screen elements
    display.setCurrent(myCanvas);

    // force repaint of the canvas
    myCanvas.repaint();
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}
}

class MyCanvas extends Canvas {
    public void paint(Graphics g) {
        // create a 20x20 black square in the center
        g.setColor(0x000000); // make sure it is black
        g.fillRect(
            getWidth()/2 - 10,
            getHeight()/2 - 10,
```



Lớp *MyCanvas* kế thừa *Canvas* và ghi đè phương thức *paint()*. Mặc dù phương thức này được gọi ngay khi canvas tạo ra thành phần *displayable* hiện hành (bằng *setCurrent(myCanvas)*), đó là một ý tưởng tốt để gọi phương thức *repaint()* trên canvas này sớm về sau. Phương thức *paint()* chấp nhận một đối tượng *Graphics*, là đối tượng cung cấp các phương thức cho việc vẽ các đối tượng 2D lên màn hình thiết bị. Như ví dụ trên một hình vuông màu đen được tạo ra ở giữa màn hình sử dụng đối tượng *Graphics* này. Chú ý rằng trước khi vẽ hình vuông bằng phương thức *fillRect()*, màu hiện hành được chọn là màu đen bằng phương thức *g.setColor()*. Điều này không

cần thiết vì màu mặc định là màu đen rồi, ví dụ này chỉ minh họa cách thay đổi màu nếu ta muốn làm sau này.

Các hàm trong phương thức *Paint()*: *javax.microedition.lcdui.Canvas*

Phương thức	Mô tả
<i>abstract void paint(Graphics g)</i>	Vẽ đối tượng lên Canvas
<i>final void repaint()</i>	Yêu cầu vẽ lên Canvas
<i>final void repaint(int x, int y, int width, int height)</i>	Yêu cầu vẽ một vùng Canvas theo kích thước và tọa độ xác định
<i>final void serviceRepaints()</i>	Xử lý yêu cầu vẽ còn treo
<i>boolean is DoubleBuffered()</i>	Kiểm tra thiết bị có hỗ trợ Buffer không

Phương thức *repaint()* sẽ gọi hàm *paint()* để vẽ lại toàn bộ hay một phần màn hình. Phương thức *serviceRepaints()* yêu cầu tất cả các yêu cầu vẽ trước đó phải được thực hiện ngay. Do đó, khi gọi phương thức này, tất cả các tiến trình khác sẽ bị block cho đến khi tất cả các phương thức vẽ được thực hiện.

### 1.3.2.6 Lớp Graphics

Là đối tượng đồ họa dùng làm công cụ để vẽ lên đối tượng Canvas đã được định nghĩa. Có hơn 30 cách vẽ khác nhau về màu, định dạng font... rất phong phú.

Các hàm được sử dụng trong lớp Graphics:

❖ **Các hàm về Color** trong gói *javax.microedition.lcdui.Graphics*

Phương thức	Mô tả
<i>void setColor(int RGB)</i>	Gán giá trị màu sắc cho đối tượng
<i>void setColor(int red, int green, int blue)</i>	Gán giá trị màu sắc theo tham số riêng
<i>int getColor()</i>	Lấy giá trị màu
<i>int getBlueComponent()</i>	Lấy giá trị màu phân xanh dương của

	đối tượng
<i>int getGreenComponent()</i>	Lấy giá trị màu phần màu xanh lục của đối tượng
<i>int getRedComponent()</i>	Lấy giá trị màu phần màu đỏ của đối tượng
<i>void setGrayScale(int value)</i>	Gán giá trị thay đổi Gray
<i>int getGrayScale()</i>	Lấy giá trị scale

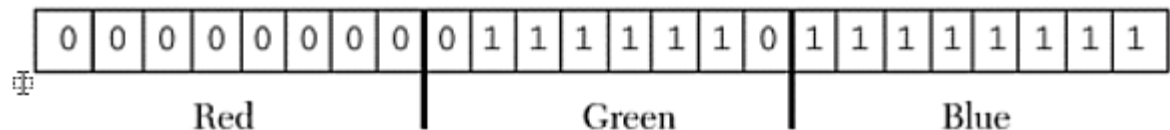
Các giá trị màu bao gồm có 3 màu chính là Red, Green và Blue tương ứng trong các hệ số sau.

	Decimal	Hexadecimal	Binary
Red	0	0x0	0000 0000
Green	126	0x7E	0111 1110
Blue	255	0xFF	1111 1111

Để lấy màu ta gọi hàm :

```
g.setColor(red, green, blue);
```

Trong đó g là đối tượng graphics



Ví dụ :

```
int colors, red, green, blue;
colors = g.getColor();

// Return the highest 8 bits
red = colors & 0xFF0000
// Return middle eight bits
green = colors & 0xFF00;
```

```
// Return lowest 8 bits
```

```
blue = colors & 0xFF
```

#### ❖ Phương thức Stroke

Là phương thức vẽ theo nét. Ta có thể chọn là nét vẽ liền Graphics.SOLID hay nét đứt Graphics.DOTTED. Mặc định của kiểu vẽ là nét liền

Ta có các phương thức trong gói *javax.microedition.lcdui.Graphics*

Phương thức	Mô tả
<i>int getStrokeStyle()</i>	Lấy kiểu nét vẽ
<i>void setStrokeStyle(int style)</i>	Gán kiểu nét vẽ
<i>void drawLine(int x1, int y1, int x2, int y2)</i>	Vẽ đường thẳng
<i>void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)</i>	Vẽ đường cong góc arcAngle
<i>void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)</i>	Tô đường cong góc arcAngle nội tiếp
<i>void drawRect(int x, int y, int width, int height)</i>	Vẽ hình chữ nhật
<i>void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)</i>	Vẽ hình chữ nhật với các góc tròn
<i>void fillRect(int x, int y, int width, int height)</i>	Tô hình chữ nhật
<i>void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)</i>	Tô hình chữ nhật góc tròn

#### ❖ Xác định font chữ và vẽ chữ

*drawString(String str, int x, int y, int anchor)* : anchor là tham số chỉ ra vị trí tương đối cần hiển thị chuỗi so với tọa độ (x, y)

Các hàm xác định font:

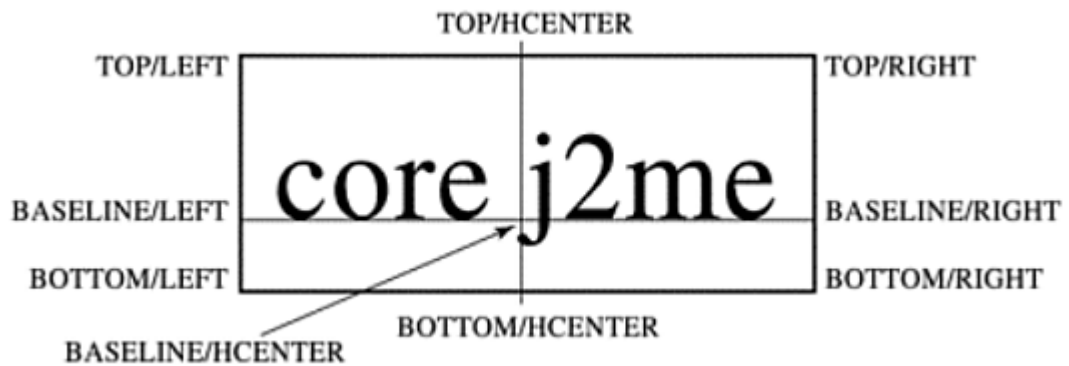
```
Font font = Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN,
Font.SIZE_MEDIUM);
```

Các phương thức hỗ trợ vẽ chữ trong gói

Phương thức	Mô tả
<i>void <b>drawChar</b>(char character, int x, int y, int anchor)</i>	Vẽ chuỗi kí tự
<i>void <b>drawChars</b>(char[] data, int offset, int length, int x, int y, int anchor)</i>	Vẽ chuỗi kí tự với chiều dài và vị trí xác định
<i>void <b>drawString</b>(String str, int x, int y, int anchor)</i>	Vẽ chuỗi kí str lên màn hình
<i>void <b>drawSubstring</b>(String str, int offset, int len, int x, int y, int anchor)</i>	Vẽ chuỗi con có chiều dài và vị trí xác định
<i>Font <b>getFont</b>()</i>	Trả về font của chuỗi
<i>void <b>setFont</b> (Font font)</i>	Gán font cho chuỗi

Giá trị của các Anchor:

Tên Anchor	Mô tả	Vị trí
LEFT	Lề trái của kí tự	Chiều ngang
HCENTER	Điểm giữa của kí tự	Chiều ngang
RIGHT	Lề phải của kí tự	Chiều ngang
TOP	Điểm cao nhất của kí tự	Chiều dọc
BASELINE	Dòng chuẩn của kí tự	Chiều dọc
BOTTOM	Điểm thấp nhất của kí tự	Chiều dọc



Ví dụ vẽ chuỗi kí tự hiện thời gian lên màn hình

```
public void paint(Graphics g) {  
    int w = getWidth();  
    int h = getHeight();  
  
    g.setColor(0xffffffff);  
    g.fillRect(0, 0, w - 1, h - 1);  
  
    Font f = Font.getFont(Font.FACE_PROPORTIONAL,  
Font.STYLE_BOLD, Font.SIZE_LARGE);  
    g.setFont(f);  
    g.setColor(0x0077ff);  
    long now = System.currentTimeMillis();  
    Date d = new Date(now);  
    Calendar c = Calendar.getInstance();  
    c.setTime(d);  
    g.drawString(c.get(Calendar.HOUR_OF_DAY) + ":" +  
c.get(Calendar.MINUTE) + ":" + c.get(Calendar.SECOND) + "." +  
c.get(Calendar.MILLISECOND), w / 2, h / 2, Graphics.HCENTER /  
Graphics.BOTTOM);  
}
```

}



Ta thường dùng biến *g* để thể hiện một đối tượng đồ họa. Ví dụ

```
g.setColor(r, g, b);
```

```
g.drawLine(startx, starty, endx, endy);
```

Các lấy một đối tượng *Graphics* : Để vẽ lên đối tượng *Canvas* sử dụng phương thức *paint( Graphics g)* là phương thức chính vẽ lên đối tượng.

```
class AnimationCanvas extends Canvas
```

```
implements CommandListener
```

```
{
```

```
private Command cmExit;      // Exit midlet
```

```
...
```

```
cmExit = new Command("Exit", Command.EXIT, 1);
```

```
addCommand(cmExit);
```



```
setCommandListener(this);  
...  
protected void paint(Graphics g)  
{  
...  
}
```

## 1.4 Xử lý sự kiện

### 1.4.1 Kiến thức chung

Việc xử lý sự kiện phát sinh là vấn đề cơ bản đối với thao tác ứng dụng người dùng đối với bất kỳ ứng dụng nào được cài lên thiết bị. Quá trình xử lý các sự kiện phát sinh bao gồm 3 quá trình cơ bản:

- Phần cứng (device) : phải tương thích nhận được sự kiện phát sinh, như bấm phím rê chuột của người dùng lên thiết bị, rút cắm cáp...
- Hệ thống của thiết bị ( Phần mềm HĐH) phải nhận biết được sự kiện được phát sinh từ phần cứng.
- Hệ điều hành chuyển thông tin về sự kiện cho ứng dụng. Việc của nhà phát triển ứng dụng là bắt những sự kiện này làm theo ý muốn của họ.

### 1.4.2 Command và CommandListener

Đối tượng *Command* là một đối tượng giữ thông tin về một sự kiện. Là điểm khởi động để phát sinh một sự kiện tương ứng.

Các bước để thêm một đối tượng *Command* bao gồm:

- Tạo Command để lưu thông tin sự kiện
- Add Command này vào Form, TextBox, Alert hay một Canvas
- Thêm listener vào các Form, TextBox, Alert hay Canvas trên

Khi phát hiện một sự kiện được kích hoạt, bộ listener sẽ gọi hàm *CommandListener()* và truyền thông tin sự kiện làm thông số cho hàm.

Ví dụ về Command Exit:

```
private Form fmMain;    // A Form
private Command cmExit; // A Command to exit the MIDlet
...
fmMain = new Form("Core J2ME");    // Form object
cmExit = new Command("Exit", Command.EXIT, 1); // Command object
...
fmMain.addCommand(cmExit);    // Add Command to Form
fmMain.setCommandListener(this); // Listen for Form events
...
public void commandAction(Command c, Displayable s)
{
    if (c == cmExit)
    {
        destroyApp(true);
        notifyDestroyed();
    }
}
```

Trong đó, hàm *commandAction(Command C, Displayable s)* là hàm chính xử lý sự kiện trước khi được khởi tạo bằng hàm

```
cmExit = new Command("Exit", Command.EXIT, 1);
```

```
cmHelp = new Command("Help", Command.HELP, 1);
                |           |           |
                label      type      priority
```

#### *Cấu trúc hàm command*

Các thông số bao gồm :

- Label : Nhãn của command
- Type : Kiểu của command, kiểu này ánh xạ một command với một nút trên thiết bị.

Ta có các type được liệt kê bên dưới:

Giá trị	Mô tả
BACK	Quay lại màn hình trước đó
CANCEL	Hủy thao tác đang thực hiện
EXIT	Thoát ứng dụng
HELP	Cung cấp thông tin trợ giúp
ITEM	Dùng để ánh xạ một Command với một Item trên màn hình. Giả sử khi ta dùng List, khi chọn một Item có thể gắn Item này với một Command để phát sinh một sự kiện nào đó
OK	Xác nhận một yêu cầu
SCREEN	Thể hiện một thông báo
STOP	Dừng một công việc đang thực hiện

- Priority : Độ ưu tiên dùng để sắp xếp các command từ trên xuống dưới hay từ trái sang phải khi thể hiện ở dạng menu

Các hàm chính của lớp Command trong gói *javax.microedition.lcdui.Command*

<b>Command Class: javax.microedition.lcdui.Command</b>	
Phương thức	Mô tả
<b>Command</b> (String label, int commandType, int priority)	Khởi tạo một command
int <b>getCommandType</b> ()	Trả về kiểu command
String <b>getLabel</b> ()	Trả về nhãn của command
int <b>getPriority</b> ()	Trả về độ ưu tiên của command
<b>CommandListener Interface: javax.microedition.lcdui.CommandListener</b>	
void <b>commandAction</b> (Command c, Displayable d)	Được gọi khi khởi tạo sự kiện

### 1.4.3 Item và ItemStateListener

Sự kiện không chỉ được phát sinh thông qua kích hoạt Commands mà còn có thể được kích hoạt qua các Items. Khác với Commands, Item chỉ được sử dụng như một thành phần của Form. Ta có các Items: ChoiceGroup, DateField, Gauge, ImageItem, StringItem và TextField.

Khi chúng ta thêm một Item vào Form, để xử lý được các sự kiện phát sinh ta phải cài đặt một Listener (Ở đây là *ItemStateListener*). Khi có một thay đổi trên Item (check field trong ChoiceGroup chẳng hạn) thì đối tượng listener sẽ được thông báo có một sự kiện phát sinh cùng các thông tin về sự kiện này. Sự kiện sẽ kích hoạt hàm *itemStateChanged()*.

Ví dụ tạo một Item DateField:

```
private DateField dfDate; // Display the date

// Create the date and populate with current date
dfDate = new DateField("Date is:", DateField.DATE);
dfDate.setDate(new java.util.Date());

fmMain = new Form("Core J2ME");
fmMain.append(dfDate);

// Capture Command events (cmExit)
fmMain.setCommandListener(this);
```

Các hàm chính của lớp Item trong gói *javax.microedition.lcdui.Item* và *ItemStateListener Interface*

<i>javax.microedition.lcdui.Item</i>	
Phương thức	Mô tả
String <b><i>getLabel()</i></b>	Trả về nhãn của Items

<code>void <b>setLabel</b>(String label)</code>	Đặt nhãn cho Item
<code>javax.microedition.lcdui. ItemStateListener</code>	
<code>void <b>itemStateChanged</b>(Item item)</code>	Được gọi khi một Item thay đổi trạng thái

## 1.5 MIDlet suite

Chúng ta thường gọi chương trình Java chạy trên thiết bị di động là một MIDlet.

MIDlet sẽ sử dụng các lớp được cung cấp bởi CLDC và MIDP

MIDlet Suite được tạo từ 1 hoặc nhiều MIDlets. Thành phần của MIDlet Suite phải bao gồm 2 phần: Java Archive File (JAR) và Java Application Descriptor(JAD)

### 1.5.1 Java Archive File (JAR)

Thành phần quan trọng nhất trong file Jar chính là tập tin manifest. Là tập tin mô tả toàn bộ nội dung của file Jar được gọi là Manifest.mf

Ta có danh sách các thuộc tính sau :

Đặc tính	Mục đích	Yêu cầu
IDlet – Name	Tên của MIDlet	Có
MIDlet – Version	Phiên bản của MIDlet	Có
MIDlet – Vendor	Người xây dựng	Có
MIDlet – <n>	Tham chiếu đến các MIDlet trong bộ Jar file, mỗi MIDlet cần một mẫu thông tin này, nó gồm 3 phần <ul style="list-style-type: none"> <li>- Tên MIDlet</li> <li>- File Icon</li> <li>- Tên lớp sẽ được nạp khi thực thi MIDlet</li> </ul>	Có
MicroEdition-Profile	Tên profile cần thiết để chạy MIDlet, thường là MIDP 1.0, phiên bản mới nhất	Có

	là MIDP 2.0	
MicroEditionConfiguration	Configuration, thường là CLDC dùng để chạy MIDlet	Có
MIDlet – Icon	File ảnh tượng trưng cho MIDlet. Chỉ có duy nhất định dạng PNG được hỗ trợ cho loại thiết bị nhỏ gọn này	Không
MIDlet – Description	Mô tả của MIDlet	Không
MIDlet – Info - URL	Địa chỉ trang web nhà phát triển	Không

Một ví dụ đơn giản :

```

MIDlet-Name: Todo List
MIDlet-Version: 1.0
MIDlet-Vendor: Core J2ME
MIDlet-1: TodoList, /images/ToDo.png, Todo.ToDoMIDlet
MicroEdition-Profile: MIDP-1.0
MicroEdition-Configuration: CLDC-1.0
Java Application Descriptor File (JAD)

```

### 1.5.2 Jad file

Ngoài file Jar ra còn có Jad file đi kèm có chức năng:

- Cung cấp thông tin về nội dung file Jar. Nhờ đó, bộ quản lý ứng dụng trên thiết bị mới quyết định việc phù hợp của ứng dụng khi chạy trên thiết bị hay không
- Cung cấp tham số dùng cho MIDlet để tránh thay đổi file Jar. File Jar chứa mã ứng dụng nên cần tránh bị thay đổi

Ta có danh sách các thuộc tính của file Jar

Thuộc tính	Mục đích sử dụng	Yêu cầu
MIDlet – Name	Tên MIDlet	Có

MIDlet – Version	Phiên bản	Có
MIDlet – Vendor	Tác giả của MIDlet Suite	Có
MIDlet – <n>	Tham chiếu đến các MIDlet trong bộ Jar file, mỗi MIDlet cần một mẫu thông tin này, nó gồm 3 phần <ul style="list-style-type: none"> <li>- Tên MIDlet</li> <li>- File Icon</li> <li>- Tên lớp sẽ được nạp khi thực thi MIDlet</li> </ul>	Có
MIDlet – Jar – URL	Địa chỉ URL của file Jar	Có
MIDlet – Jar – Size	Kích thước của file Jar (tính bằng byte)	Có
MIDlet – Description	Mô tả MIDlet	Không
MIDlet – Delete – Confirm	Cảnh báo khi xóa MIDlet	Không
MIDlet – Install - Modify	URL nhận thông báo về quá trình cài đặt	Không
MIDlet – Data - Size	Kích thước tối thiểu (tính bằng byte) để ghi các dữ liệu của ứng dụng	không

Một ví dụ đơn giản:

```

MIDlet-Name: Todo List
MIDlet-Version: 1.0
MIDlet-Vendor: Core J2ME
MIDlet-Jar-URL: http://www.corej2me.com/ToDoMIDlet.jar
MIDlet-Jar-Size: 17043
MIDlet-1: TodoList, /images/ToDo.png, Todo.ToDoMIDlet
    
```

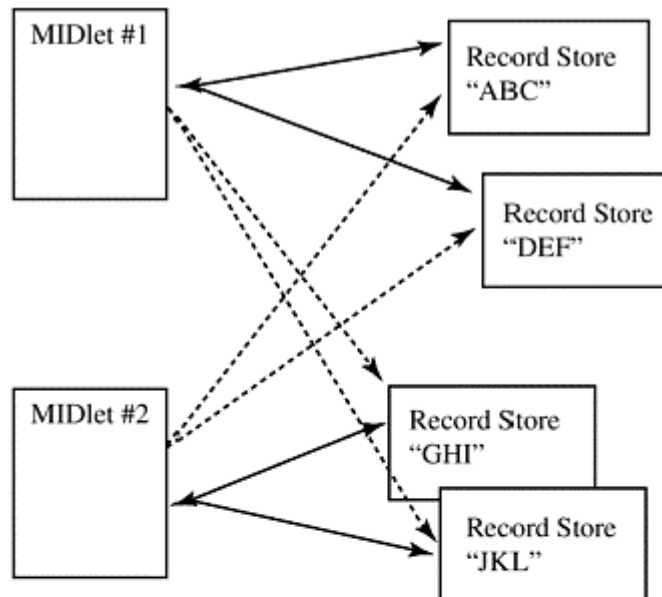
## 1.6 Lưu trữ dữ liệu trên thiết bị di động (RMS):

RMS (Record Management System) là hệ thống quản lý bản ghi dưới dạng record. Mỗi Record có thể chứa bất kỳ loại dữ liệu nào, chúng có thể là kiểu số nguyên, chuỗi ký tự hay có thể là một ảnh và kết quả là một Record là một chuỗi (mảng) các byte.

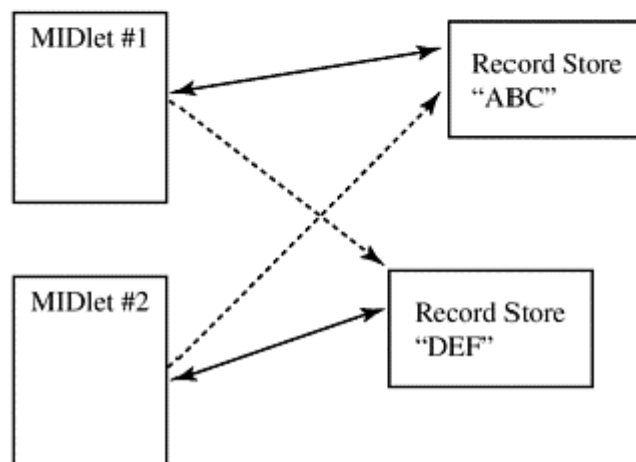
Một tập các RecordStore là tập hợp các Record được sắp xếp có thứ tự. Mỗi Record không thể đứng độc lập mà nó phải thuộc vào một RecordStore nào đó, các thao tác trên Record phải thông qua RecordStore chứa nó. Khi tạo ra một Record trong RecordStore, Record được gán một số định danh kiểu số nguyên gọi là Record ID. Record đầu tiên được tạo ra sẽ được gán Record ID là 1 và sẽ tăng thêm 1 cho các Record tiếp theo.



### **MIDLET SUITE ONE**



### **MIDLET SUITE TWO**



*H1.3 Cơ chế hoạt động của RMS*

Đường liền thể hiện việc truy xuất Record store do MIDlet đó tạo ra, đường nét đứt là Record store do MIDlet khác tạo ra.

Trong MIDLET Suite One, MIDlet #1 và MIDlet #2 cùng có thể truy xuất 4 Record store. MIDLET Suite One không thể truy xuất Record store trong Suite Two. Trong

MIDlet Suite One tên của các Record store là duy nhất, tuy nhiên Record store trong các MIDlet Suite khác nhau có thể dùng chung một tên.

Với ứng dụng Cố vấn học tập

- Dữ liệu sau khi nhận được từ server response lại Client sẽ được lưu lại để người dùng có thể xem lại nó.
- Sau mỗi lần có kết quả mới từ một request mới thì dữ liệu cũ sẽ bị ghi chồng lên. Tức là vùng nhớ chứa dữ liệu cũ sẽ bị thay thế để đảm bảo việc chiếm dụng vùng nhớ không nhiều đối với thiết bị có cấu hình thấp.

Các hàm trong RMS

**RMS** không có hàm khởi tạo. Ta liệt kê các hàm và phương thức sau

<i>javax.microedition.rms.RecordStore</i>	
static RecordStore <b>openRecordStore</b> (String recordStoreName, boolean createIfNecessary)	Mở một Recordstore, có tham số tạo Record store nếu nó chưa tồn tại.
void <b>closeRecordStore</b> ()	Đóng RecordStore
static void <b>deleteRecordStore</b> (String recordStoreName)	Xóa RecordStore
static String[] <b>listRecordStores</b> ()	Danh sách các RecordStore trong MIDlet Suite
int <b>addRecord</b> (byte[] data, int offset, int numBytes)	Thêm một Record vào RecordStore
void <b>setRecord</b> (int recordId, byte[] newData, int offset, int numBytes) record.	Đặt một Record trong RecordStore
void <b>deleteRecord</b> (int recordId)	Xóa một Record trong RecordStore

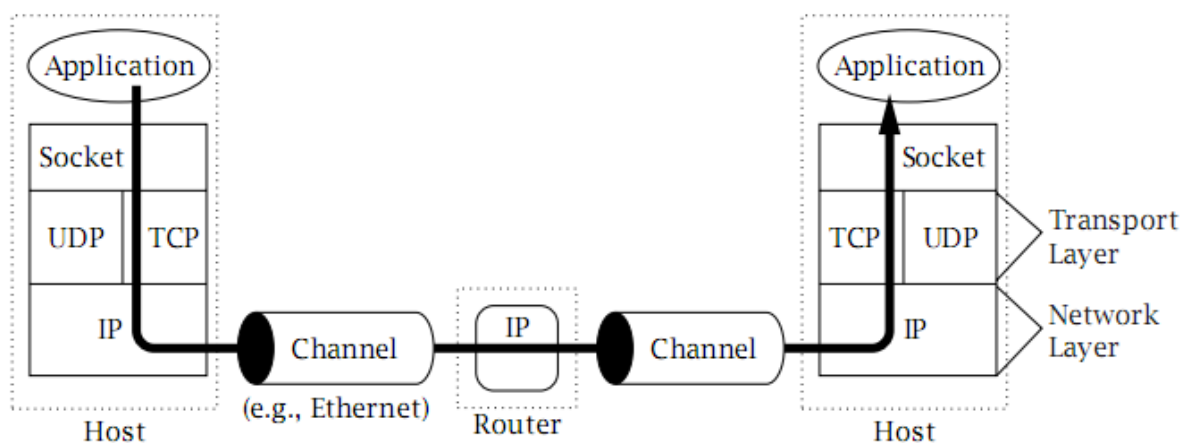
byte[] <b>getRecord</b> (int recordId)	Trả về một Record trong RecordStore
int <b>getRecord</b> (int recordId, byte[] buffer, int offset)	Lấy nội dung của Record vào dãy byte
int <b>getRecordSize</b> (int recordId)	Lấy kích thước của Record
int <b>getNextRecordID</b> ()	Lấy Id của Record mới
int <b>getNumRecords</b> ()	Lấy số lượng các Record
long <b>getLastModified</b> ()	Thời gian thay đổi gần nhất
int <b>getVersion</b> ()	Lấy phiên bản của RecordStore
String <b>getName</b> ()	Lấy tên của RecordStore
int <b>getSize</b> ()	Kích thước của RecordStore
int <b>getSizeAvailable</b> ()	Số byte trống của RecordStore
RecordEnumeration <b>enumerateRecords</b> (RecordFilter filter, RecordComparator comparator, boolean keepUpdated)	Xây dựng enumerator dùng để duyệt RecordStore
void <b>addRecordListener</b> (RecordListener listener)	Add listener để thực thi RecordStore
void <b>removeRecordListener</b> (RecordListener listener)	Xóa Listener

## CHƯƠNG 2: MÔ HÌNH CLIENT – SERVER

### 2.1 Định nghĩa

Mô hình Client – Server là mô hình tương tác giữa một bên là máy khách và một bên là máy chủ. Trong đó Client là máy khách thực hiện gửi một yêu cầu lên server thông qua socket. Server thực nhận dạng socket này và thực hiện tạo tác xử lý sau đó trả về thông qua socket trong cùng cổng (port).

Socket là giao diện do ứng dụng tạo ra trên máy trạm, quản lý bởi hệ điều hành qua đó các ứng dụng có thể gửi và nhận các thông điệp đến/từ các ứng dụng khác. Client và Server giao tiếp thông qua socket.

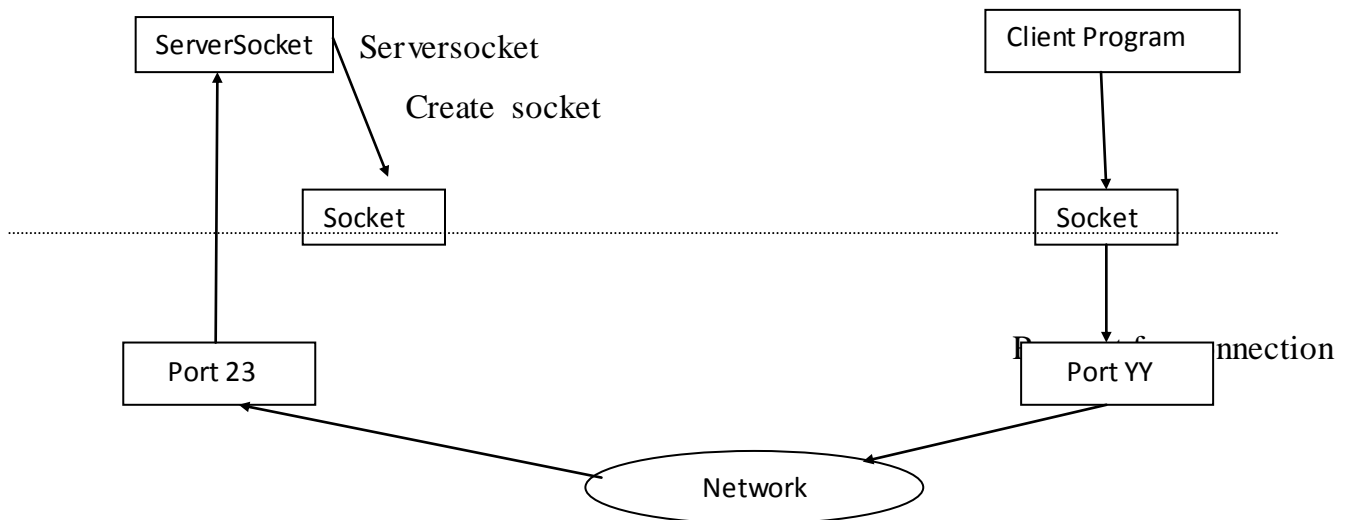


H2.1 Mô hình Client - Server

### 2.2 Các thành phần chính của lập trình mạng Client – Server

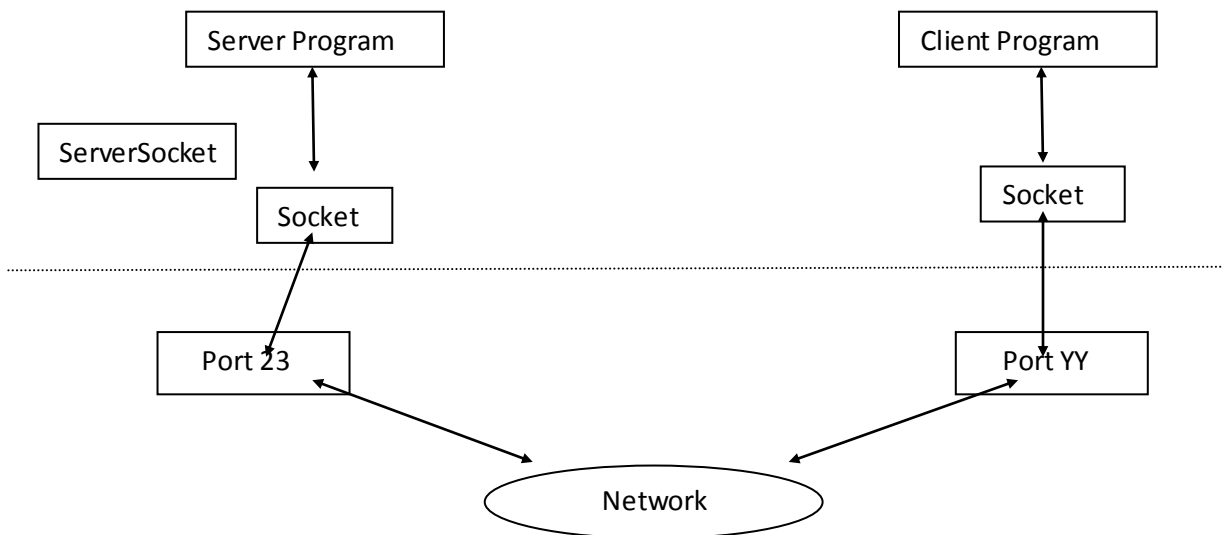
#### 2.2.1 Giao tiếp Client – Server

Yêu cầu kết nối:



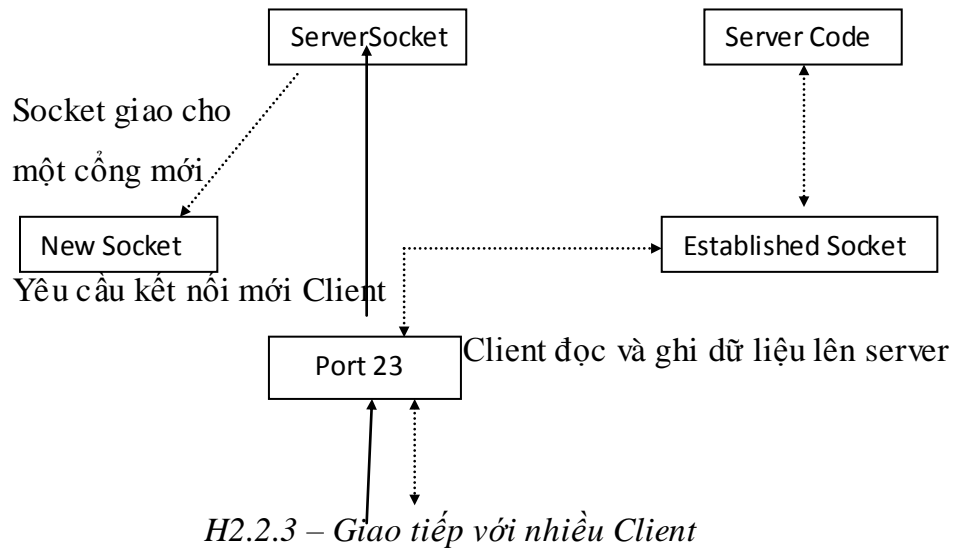
*H2.2.1 – Yêu cầu kết nối Client – Server*

**Giao tiếp Client – Server:**



*H2.2.2 – Giao tiếp Client – Server*

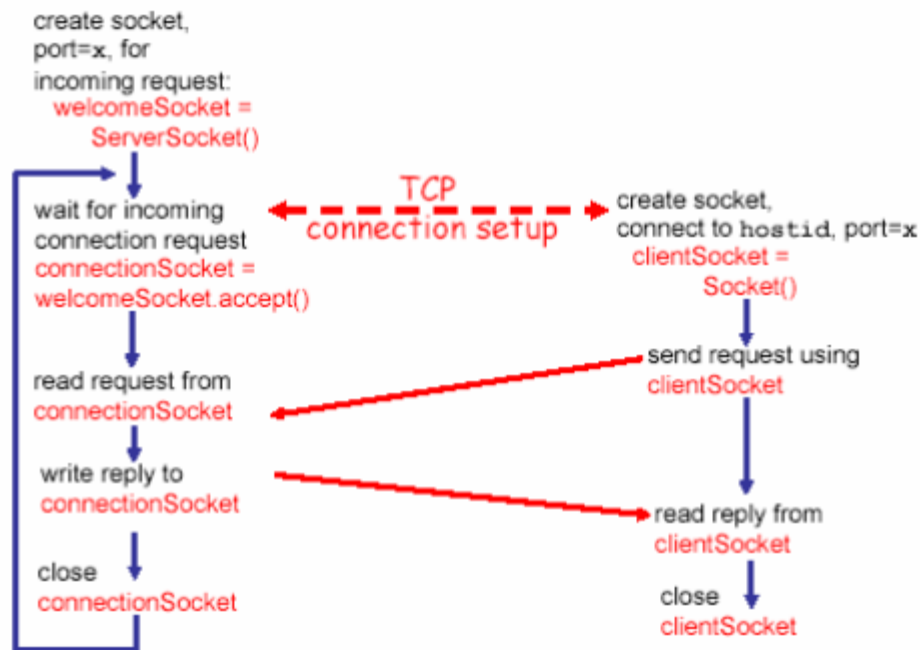
## 2.2.2 Giao tiếp với nhiều client



## 2.3 Thiết kế giải thuật Client – Server

### 2.3.1 Thiết kế cho Client:

#### 2.3.1.1 Giao thức TCP



H2.3.1 – Mô hình Client – Server giao thức TCP

Có 2 phương thức kết nối là TCP truyền theo byte dữ liệu và UDP truyền theo các gói package. Khi Server nhận yêu cầu kết nối nó sẽ chấp nhận yêu cầu và khởi tạo socket mới để giao tiếp với Client. Có thể chấp nhận nhiều Client tại cùng một thời điểm.

- **Client:** phải gửi yêu cầu tới Server bằng cách tạo một socket trên máy khách. Chỉ rõ port number trên máy Server. Khi Client tạo socket client liên kết với server
  - Khởi tạo TCP socket
  - Xác định IP address, port number của Server
  - Thiết lập kết nối đến Server
- **Server:** Tiến trình máy chủ phải đang thực thi. Máy chủ phải đang mở socket để nhận yêu cầu từ Client

- Server process phải chạy trước
- Server phải tạo một socket để lắng nghe và chấp nhận các kết nối từ Client

Ví dụ với giao thức TCP cho client

```
import java.io.*;
import java.net.*;

public class TCPClient {
    public static void main(String arg[]) throws EOFException,
    IOException{
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser = new BufferedReader(new
        InputStreamReader(System.in));

        Socket clientSocket = new Socket("hostname",6789);
        DataOutputStream outToServer = new
        DataOutputStream(clientSocket.getOutputStream());

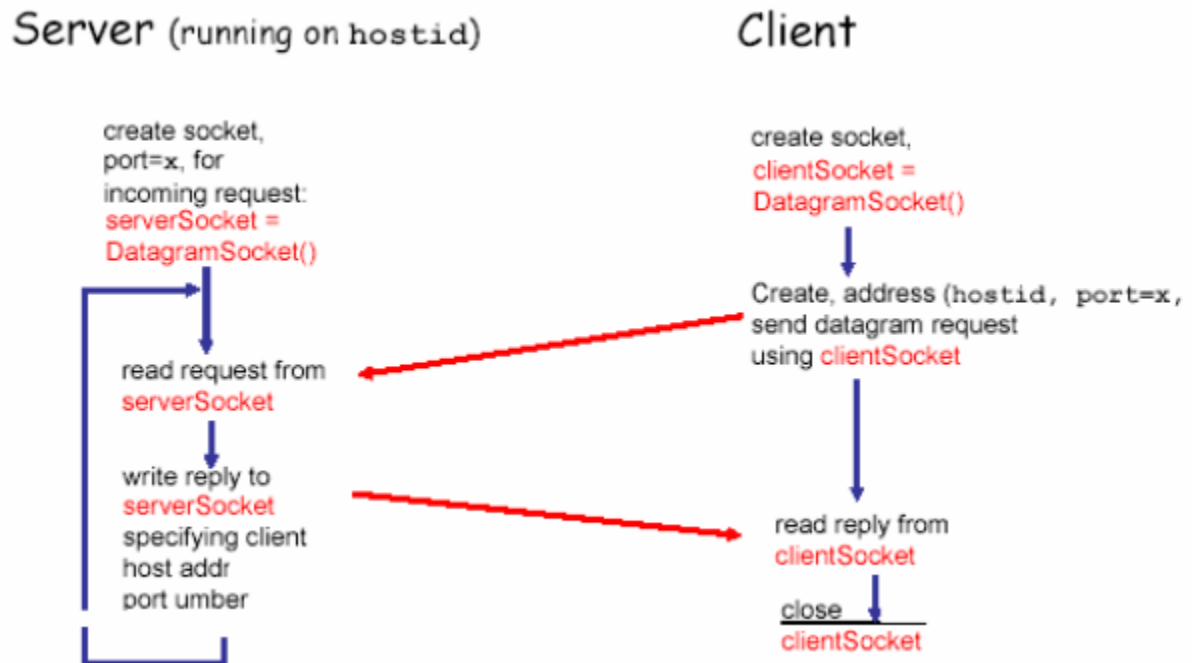
        BufferedReader inFromUser = new BufferedReader(new
        inputStreamReader(clientSocket.getInputStream()));

        sentence = inFromUser.readLine();
        outToServer.writeBytes(sentence + '\n');
        modifiedSentence = inFromUser.readLine();
        System.out.println("FROM SERVER: " + modifiedSentence);
    }
}
```



```
        clientSocket.close();  
    }  
}
```

### 2.3.1.2 Giao thứcUDP



#### H2.3.2 – Mô hình Client – Server giao thức UDP

Với giao thức UDP ta có các đặc tính

- Không cần thiết lập kết nối giữa Client và Server
- Sender phải gửi kèm địa chỉ IP và port
- Server khi nhận được dữ liệu sẽ phân tích địa chỉ của sender để gửi lại
- Có thể Server chấp nhận nhiều Client tại cùng một thời điểm

#### Client:

- Xác định địa chỉ Server
- Tạo socket
- Gửi/ nhận dữ liệu theo giao thức lớp ứng dụng đã thiết kế

- Đóng socket

Ví dụ với giao thức UDP cho client:

```
import java.io.*;
import java.net.*;

/**
 *
 * @author Sony
 */
public class UDPClient {
    public static void main(String arg[]) throws EOFException{
        BufferedReader inFromUser = new BufferedReader(new
        InputStreamReader(System.in));

        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = new InetAddress.getByName("hostname");

        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];

        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();

        DatagramPacket sendPacket = new DatagramPacket(sendData,
        sendData.length, IPAddress, 9876);
        clientSocket.send(sendPacket);
    }
}
```

```
DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);

clientSocket.receive(receivePacket);

String modifiedSentence = new String(receivePacket.getData());
System.out.println("FROM Server:" + modifiedSentence);
clientSocket.close();
}
}
```

### 2.3.2 Thiết kế giải thuật cho Server

Chương trình server có hai loại:

- Lặp (iterative)
- Đồng thời (concurrent)

Giao thức cho server có 2 loại

- Connection – oriented
- Connectionless

❖ Giải thuật cho chương trình Server iterative , connection – oriented

- Tạo socket đăng kí địa chỉ socket với hệ thống
- Đặt socket ở trạng thái lắng nghe và chờ và sẵn sàng cho việc kết nối từ client
- Chấp nhận kết nối từ client, gửi/ nhận dữ liệu theo giao thức lớp ứng dụng đã thiết kế
- Đóng kết nối sau khi đã hoàn thành, trở lại trạng thái lắng nghe chờ kết nối mới

❖ Giải thuật cho chương trình server iterative, connectionless

- Tạo socket đăng kí với hệ thống

- Lập công việc đọc dữ liệu từ client gửi đến, xử lý và gửi trả kết quả cho client theo đúng giao thức đã thiết kế
- ❖ Giải thuật cho chương trình server concurrent, connection – oriented
  - Tạo socket, đăng kí với hệ thống
  - Đặt socket ở chế độ chờ, lắng nghe kết nối
  - Khi có request từ client, chấp nhận kết nối, tạo một process con để xử lý. Quay lại trạng thái chờ, lắng nghe kết nối mới
  - Công việc của process mới bao gồm:
    - Nhận thông tin kết nối mới từ client
    - Giao tiếp với client theo giao thức lớp ứng dụng đã thiết kế
    - Đóng kết nối và kết thúc process con
- ❖ Giải thuật cho chương trình server concurrent, connectionless
  - Tạo socket, đăng kí với hệ thống
  - Lập việc nhận dữ liệu từ client, đối với một dữ liệu nhận, tạo mới một process để xử lý. Tiếp tục nhận dữ liệu mới từ client
  - Công việc của process mới:
    - Nhận thông tin từ process cha gửi tới, lấy thông tin socket
    - Xử lý và gửi thông tin về cho client theo giao thức lớp ứng dụng đã thiết
    - Kết thúc

Ví dụ: UDP Server

```
import java.io.*;
import java.net.*;
public class UDPServer {
    public static void main(String arg[]) throws EOFException{
        BufferedReader inFromUser = new BufferedReader(new
        InputStreamReader(System.in));
```

```
DatagramSocket serverSocket = new DatagramSocket();
InetAddress IPAddress = new InetAddress.getByName("hostname");

byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];

while(true){
    DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
    serverSocket.receive(receivePacket);

    String sentence = new String(receivePacket.getData());
    InetAddress IPAddress = new InteAddress();
    int port = receivePacket.getPort();
    String capitalizedSentence = sentence.toUpperCase(); //doi chu hoa
thanh chu thuong
    sendData = capitalizedSentence.getBytes();

    DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, port);
    serverSocket.send(sendPacket);
    }
}
}
```

TCP Server: trả về chữ in hoa cho Client

```
import java.io.*;
```

```
import java.net.*;
import java.util.Locale;

public class TCPServer {
    public static void main(String arg[]) throws EOFException, IOException{
        String clientSentence;
        String capitalizedSentence;

        ServerSocket welcomeSocket = new ServerSocket(6789);

        while(true){
            Socket connectionSocket = welcomeSocket.accept();

            BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));

            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());

            clientSentence = inFromClient.readLine();

            capitalizedSentence = clientSentence.toUpperCase() + '\n';

            outToClient.writeBytes(capitalizedSentence);
        }
    }
}
```

## 2.4 Lập trình socket trên java

### ❖ Gói java.net

- InetAddress
- ServerSocket
- Socket
- URL
- URLConnection
- DatagramSocket

### • Tạo một socket

*Socket(InetAddress address, int port)*

*Socket(String host, int port)*

*Socket(InetAddress address, int port, InetAddress, localAddr, int localPort)*

*Socket(String host, int port, InetAddress, localAddr, int localPort)*

*Socket()*

### • Lấy thông tin về một socket

*InetAddress getInetAddress()* : trả về địa chỉ mà socket kết nối đến

*int getPort()* : trả về địa chỉ mà socket kết nối đến

*InetAddress getLocalAddress()* : trả về địa chỉ cục bộ

*int getLocalPort()* : trả về địa chỉ cục bộ

### • Sử dụng Streams

*public OutputStream getOutputStream() throws IOException*

Trả về một output stream cho việc viết các byte đến socket này

*public InputStream getInputStream() throws IOException*

Trả về một input stream cho việc đọc các byte từ socket này

### ❖ ServerSocket class

Là class mô tả về ServerSocket

- **Tạo một ServerSocket**

*ServerSocket(int port) throws IOException*

*ServerSocket(int port, int backlog) throws IOException*

*ServerSocket(int port, int backlog, InetAddress bindAddr) throws IOException*

- **Các phương thức trong ServerSocket**

*Socket accept() throws IOException* //lắng nghe một kết nối đến socket này và chấp nhận nó

*void close() throws IOException* : đóng socket

*InetAddress getInetAddress()* : trả về địa chỉ cục bộ của socket

*int getLocalPort()* : trả về port mà socket đang lắng nghe

*void setSoTimeout(int timeout) throws SocketException*

*Enable/disable SO\_TIMEOUT với khai báo timeout (milliseconds)*

- **Datetime Server**

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class DayTimeServer {
    public final static int daytimePort = 5000;
    public static void main(String[] args) {
        ServerSocket theServer;
        Socket theConnection;
        PrintStream p;
        try {
            theServer = new ServerSocket(daytimePort);
            while (true) {
                theConnection = theServer.accept();
```



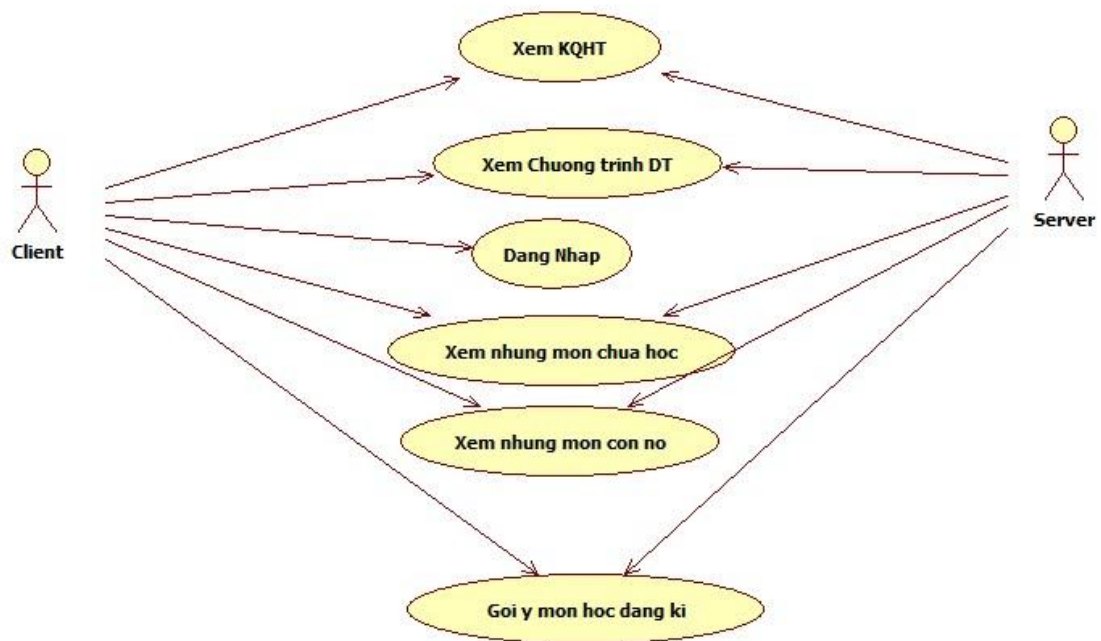
```
p = new PrintStream(theConnection.getOutputStream());  
p.println(new Date());  
theConnection.close();  
theServer.close();  
}  
}catch (IOException e) {  
System.err.println(e);  
}  
}  
}
```

## PHẦN 4 – NGHIÊN CỨU THỰC NGHIỆM ÁP DỤNG CHƯƠNG TRÌNH CỐ VẤN HỌC TẬP ĐIỆN TỬ

### CHƯƠNG 1 – CÁC YÊU CẦU

Đưa ra các yêu cầu bài toán và phân tích, tìm hướng giải quyết

#### 1.1 Yêu cầu chức năng



*H4.1.1 – Mô hình Usecase*

Các chức năng cần được xây dựng bao gồm:

- Đăng nhập : dành cho Client là sinh viên.
- Xem điểm qua các học kì: sinh viên được phép xem điểm sau khi đã đăng nhập.
- Xem điểm tổng kết: sinh viên được phép xem điểm sau khi đã đăng nhập.
- Xem chương trình đào tạo.
- Xem những môn học trong chương trình đào tạo mà sinh viên chưa học.
- Xem những môn học mà sinh viên con nợ.

- Yêu cầu tư vấn môn học nên đăng kí trong học kì này dựa vào các dữ kiện để xử lý bao gồm: thời khóa biểu, kết quả học tập và chương trình đào tạo.
- Sau mỗi lần requestm kết quả trả về từ server sẽ được lưu lại trên thiết bị client, sinh viên có thể xem ở trạng thái offline. Những dữ liệu này sẽ được ghi đè lên ở lần request sau đó với dữ liệu mới.

## 1.2 Yêu cầu phần cứng:

Phần cứng cho device: Thiết bị di động phải có khả năng kết nối internet wifi

Trên máy PC: cấu hình tối thiểu Duo Core, RAM 1Gb dùng cho máy Server.

Ở đây chúng em chạy trên môi trường ảo nên Server trực tiếp sẽ là PC và Client là emulator.

## 1.3 Yêu cầu phần mềm:

Trên máy PC server:

JRE: *jre-7u1-windows-x64*

SDK : bản cài đặt *sun\_java\_me\_sdk-3\_0-win* bản mới nhất

*sun\_java\_wireless\_toolkit-2\_5\_2-windows*: dùng để chạy ứng dụng trên emulator

Trên Client device: Hỗ trợ cho J2ME

## 1.4 Môi trường thực thi (thiết bị)

J2ME trên device client và J2SE trên PC server

Yêu cầu máy client và server phải kết nối thông qua internet wifi




Name	Type	Length	Decimals	Allow Null	
ma_mh	varchar	10	0	<input type="checkbox"/>	
ten_mh	varchar	50	0	<input type="checkbox"/>	
so_tc	int	2	0	<input type="checkbox"/>	
ma_khoa	varchar	10	0	<input type="checkbox"/>	
ma_chuyen_nganh	varchar	10	0	<input checked="" type="checkbox"/>	
tuy_chon	bit	1	0	<input type="checkbox"/>	
ma_tien_quyet	varchar	50	0	<input checked="" type="checkbox"/>	

Bao gồm các thuộc tính :

- mã ngành : khóa chính
- mã môn học: khóa chính
- mã khoa: Để phân biệt các môn cơ sở ngành (không thuộc chuyên ngành)
- tên môn học
- mã môn học tiên quyết,
- tùy chọn: kiểm tra xem là môn học bắt buộc hay môn tùy chọn

#### • Khoa

Quy mô đào tạo Đại học có nhiều khoa quản lý sinh viên. Trong đó, mỗi khoa có ít nhất một ngành học. Khoa thường cố định, ít thay đổi trừ khi trường có thêm ngành mới không liên quan tới các khoa đang có


Name	Type	Length	Decimals	Allow Null	
makhoa	varchar	15	0	<input type="checkbox"/>	 1
tenkhoa	varchar	50	0	<input type="checkbox"/>	

Bao gồm các thuộc tính:

- Mã khoa: khóa chính
- tên khoa. Một khoa có thể có nhiều ngành học

#### • Ngành học


Một ngành học bắt buộc phải thuộc 1 khoa duy nhất và thường ít thay đổi chỉ khi có ngành học mới cần bổ sung từ Phòng Đào Tạo.

Name	Type	Length	Decimals	Allow Null	
ma_chuyen_nganh	varchar	15	0	<input type="checkbox"/>	 1
ma_khoa	varchar	15	0	<input type="checkbox"/>	
ten_chuyen_nganh	varchar	60	0	<input type="checkbox"/>	
tt_chitiet	varchar	5000	0	<input checked="" type="checkbox"/>	

Mỗi ngành học thuộc một khoa nhất định. Các thuộc tính của ngành bao gồm:

- mã ngành: khóa chính dùng để phân biệt các ngành trong cùng một khoa
- tên ngành: nhận dạng ngành với tên ngành
- mã khoa: khóa chính để phân biệt các khoa với nhau

#### • Sinh viên

Name	Type	Length	Decimals	Allow Null	
masv	varchar	10	0	<input type="checkbox"/>	 1
password	varchar	150	0	<input type="checkbox"/>	
hoten	varchar	100	0	<input type="checkbox"/>	
ngaysinh	date	0	0	<input type="checkbox"/>	
ma_chuyen_nganh	varchar	15	0	<input checked="" type="checkbox"/>	
email	varchar	100	0	<input checked="" type="checkbox"/>	
mobile	varchar	12	0	<input checked="" type="checkbox"/>	
ma_khoa	varchar	10	0	<input type="checkbox"/>	



Mỗi sinh viên được phân biệt bởi mã số sinh viên. Mỗi sinh viên học một ngành và thuộc một khoa quản lý nhất định. Các chức năng cần dùng bao gồm:

Thêm, xóa, sửa Sinh viên.

Các thuộc tính của table Sinh viên bao gồm:

- Mã số sinh viên: khóa chính
- tên sinh viên: Quản lý tên của sinh viên
- mã ngành: Quản lý ngành học của sinh viên
- ngày sinh: Quản lý ngày sinh của sinh viên
- điện thoại : Điện thoại liên lạc của sinh viên
- Email: địa chỉ email của sinh viên
- Mã khoa: Khoa quản lý sinh viên

- **Thời khóa biểu**

Name	Type	Length	Decimals	Allow Null	
ma_mh	varchar	10	0	<input type="checkbox"/>	 2
hoc_ky	int	5	0	<input type="checkbox"/>	 1
thu	int	1	0	<input type="checkbox"/>	
tiet_batdau	int	2	0	<input type="checkbox"/>	
sotiet	int	2	0	<input type="checkbox"/>	
ten_phonghoc	varchar	10	0	<input type="checkbox"/>	
tuan_bd	int	2	0	<input type="checkbox"/>	
tuan_kt	int	2	0	<input type="checkbox"/>	
si_so	int	4	0	<input type="checkbox"/>	
so_tc	int	2	0	<input type="checkbox"/>	


Mỗi học kì đều có một thời khóa biểu riêng cho sinh viên. Do chủ ý của chương trình nên chỉ xác định được một thời khóa biểu chính thức trong đó chứa tất cả các môn học sinh viên có thể đăng kí trong kì. Với mỗi học kì mới thì sẽ có thời khóa biểu mới. Khi đó, thời khóa biểu cũ sẽ bị xóa đi. Ta có các tính năng là: thêm và xóa

Các thuộc tính của table thời khóa biểu bao gồm:

- Mã môn học: khóa chính nhằm phân biệt môn học này với môn học khác.
- học kỳ: Mỗi học kì có một thời khóa biểu riêng do phòng đào tạo đưa ra và học kì là một khóa chính.
- phòng học: Nơi bố trí lớp học được sắp xếp theo từng lớp
- tiết bắt đầu: Mỗi buổi học có 6 tiết và một ngày học có 12 tiết tính theo thứ tự.
- số tiết: số tiết dạy của môn đó trong một buổi học.
- tuần bắt đầu: Mỗi một môn học kéo dài trong một số tuần nhất định
- tuần kết thúc.
- số tín chỉ: Số tín chỉ của môn học đó, lấy từ môn học.
- sĩ số: tổng số sinh viên có thể đăng kí cho môn học đó.
- thứ: Thứ học trong tuần.

- **Môn học**




Môn học được phân theo tổ bộ môn và theo khoa. Môn học thường không thay đổi và chỉ bổ sung từ phòng đào tạo.

Name	Type	Length	Decimals	Allow Null	
▶ mamh	varchar	10	0	<input type="checkbox"/>	 1
tenmh_eng	varchar	150	0	<input type="checkbox"/>	
tenmh_vn	varchar	150	0	<input type="checkbox"/>	
so_tinchi	smallint	6	0	<input type="checkbox"/>	

Mỗi môn học được xác định bằng mã môn học. Các thuộc tính của môn học bao gồm:

- mã môn học: khóa chính để phân biệt môn học này với môn học khác.
- tên môn học Việt Nam: xác định tên môn học để nhận biết.
- tên môn học tiếng Anh: Dùng tên tiếng Anh trong những trường hợp cần thiết.
- số tín chỉ: Nói lên thời gian học của một môn.

- **Điểm môn học:**

Name	Type	Length	Decimals	Allow Null	
▶ masv	varchar	10	0	<input type="checkbox"/>	 1
ma_mh	varchar	10	0	<input type="checkbox"/>	 2
diem	float	3	0	<input checked="" type="checkbox"/>	
hocki	varchar	3	0	<input type="checkbox"/>	 3

Điểm môn học để xác định kết quả học tập của mỗi môn của một sinh viên. Nhờ kết quả điểm môn học này làm cơ sở để xác định sinh viên đã học đủ số tín chỉ hay chưa. Những môn nào đã hoàn thành và những môn tiên quyết liên quan để tư vấn cho sinh viên nên chọn những môn nào đăng kí trong một kì học.

Tương ứng với một Sinh viên sau khi kết thúc một học kì thì *diem\_mon\_hoc* sẽ được cập nhật bổ sung vào table này.

Các thuộc tính của điểm môn học bao gồm:

- Mã sinh viên: là khóa chính để phân biệt điểm giữa các sinh viên.
- Mã môn học: là khóa chính để phân biệt điểm của các môn.



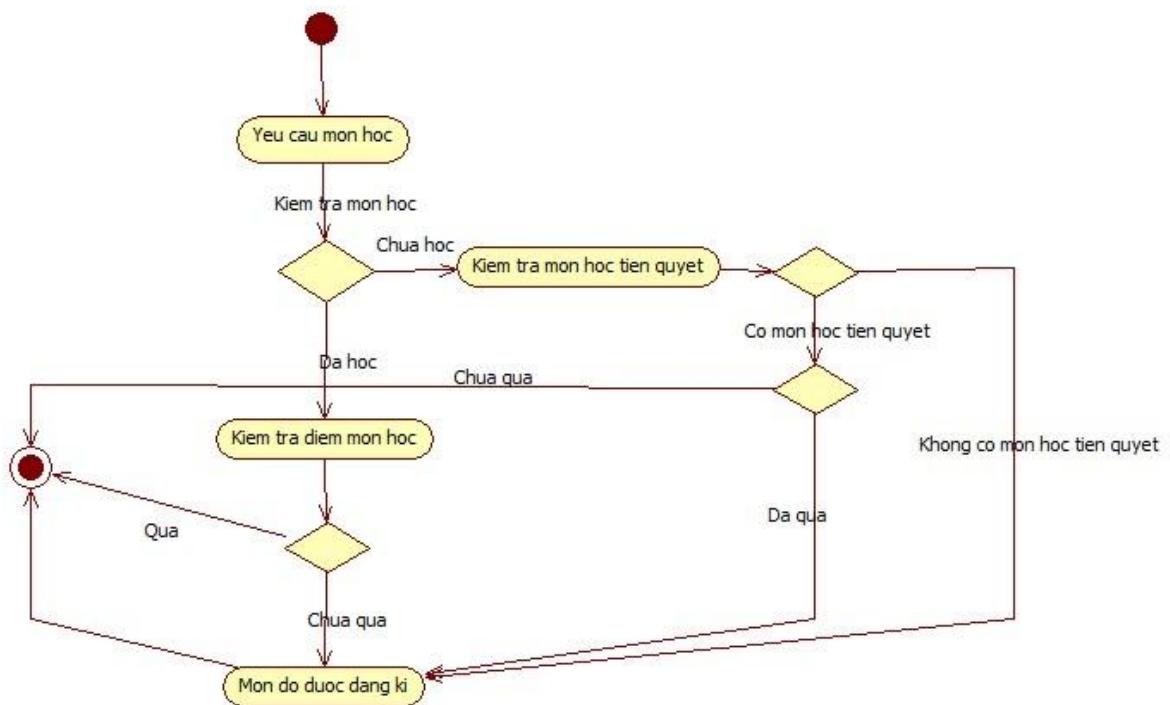
- Điểm: kết quả học tập của một môn học.
- Học kì : Mỗi học kì có một điểm riêng làm khóa chính.

## 2.3 Xử lý vấn đề của bài toán

Bài toán được áp dụng: Xây dựng ứng dụng Cố vấn sinh viên học tập giải quyết các vấn đề sau:

- Giải quyết các môn học mà sinh viên đã thi đậu cho qua.
- Xử lý các môn học mà sinh viên đã được đặt cách cho qua.
- Xử lý các môn học có các môn học tiên quyết.
- Xử lý các môn học mà sinh viên được phép đăng kí trong mỗi kỳ: bao gồm đúng chuyên ngành, chưa học hoặc đã học nhưng chưa qua, môn tiên quyết (nếu có) đã qua.

❖ Ta có một phiên cố vấn



H2.3a Một phiên cố vấn môn học

❖ **Định nghĩa dữ liệu vào**

- Các môn học bắt buộc (nằm trong table *ct\_dao\_tao*).
- Các môn học tự chọn (nằm trong table *ct\_dao\_tao*).
- Các môn học có các môn tiên quyết( nằm trong table *ct\_dao\_tao*).
- Các môn học mà sinh viên đã học xong ( nằm trong table *diem\_mon\_hoc*).
- Các môn học mà sinh viên được phép đăng kí trong mỗi kỳ ( nằm trong table *thoi\_khoa\_bieu*).

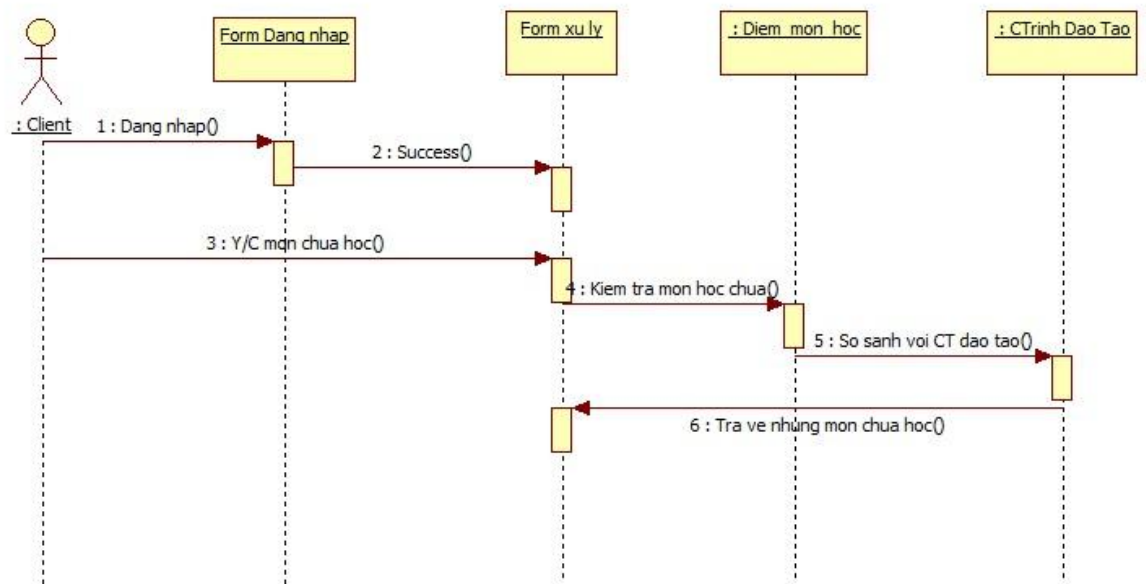
Ở mỗi học kì khi có thời khóa biểu mới, sinh viên có MSSV xxx sẽ được gợi ý những môn học nên đăng kí với những điều kiện:

- Môn đó nằm trong thời khóa biểu.
- Môn đó nằm trong chương trình đào tạo của ngành mình đang theo học.
- Môn đó chưa được học hoặc đã học mà chưa qua.
- Môn đó không có môn học tiên quyết hoặc có môn học tiên quyết mà môn học tiên quyết đã được học và qua.

Ví dụ với một môn trong thời khóa biểu:

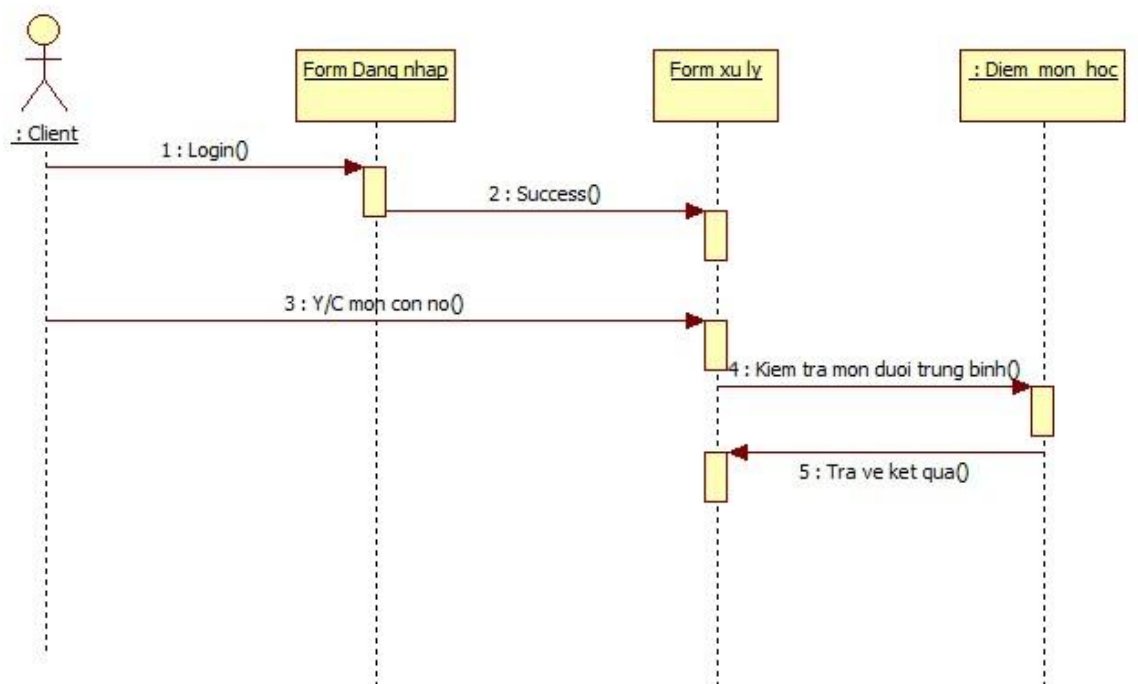
- Môn hệ thống web 2 học kì này (học kì 1 năm 2008 – 2009).
- Môn Hệ thống web 2 chưa được học trước đó (không tồn tại trong table *diem\_mon\_hoc*).
- Môn hệ thống web 2 có môn học tiên quyết hệ thống web 1, môn hệ thống web 1 đã học và qua.
- Vậy sinh viên nên đăng kí môn Hệ thống web 2.

❖ Việc kiểm tra xem những môn nào chưa học, ta có sơ đồ sau:



*H2.3b – Biểu đồ tuần tự kiểm tra những môn chưa học*

❖ Việc kiểm tra những môn còn nợ ta có sơ đồ sau



*H2.3c – Biểu đồ tuần tự kiểm tra những môn còn nợ*

## 2.4 Xây dựng các giao diện và font chữ

### 2.4.1 Giao diện

#### ❖ Các menu chức năng

- Các constructor

```
public CMenu(int x, int y, int w, int h) //khởi tạo menu với tọa độ và kích thước  
public CMenu(int x, int y, int width, int height, boolean enableFocus, Vector data,  
int m_iNumItem) //khởi tạo menu với vị trí, tọa độ, trạng thái, dữ liệu và số item
```

- Các phương thức

```
public void add(String smenu) //thêm menu  
public void add(String id,String caption) // thêm một menu item  
public void setDimension(int _width, int _height) //gán kích thước cho menu  
void setPosition public (int xx, int yy) //gán tọa độ cho menu
```

- Sự kiện trên bàn phím : Lên một item, xuống item bên dưới dựa vào *itemIndex*  
case KEY.DOWN case KEY.UP

#### ❖ Các table view: các bảng dữ liệu thể hiện điểm thi, môn học, sinh viên...

- Ta có các constructor:

```
public CTable(String[]title)// khởi tạo table với tên table  
public CTable(int row, int col): //khởi tạo table với số hàng và số cột  
public CTable(Object data[][], int model[][]) //khởi tạo table với dữ liệu và  
kiểu dữ liệu
```

- Ứng với các phương thức

```
public void insertBlankRowAt(int row) //thêm số hàng với dữ liệu trống  
public void insertRowAt(Object object[], int row)// thêm cột cột có dữ liệu  
public boolean insertColumnAt(Object object[], int column) //kiem tra du lieu  
co duoc insert không
```

```
public boolean addItem(Object object,int model,int row, int column)//kiem tra
việc thêm một item mới

public void setData(Object[][] data)//gán dữ liệu vào

public void appendBlankRow()//thêm 1 hàng trống

public void appendRow(Object rowData[])//thêm một hàng có dữ liệu

public void appendColumn(Object columnData[],String titleName)// thêm một
cột có dữ liệu và tên cột

public void removeRowAt(int rowIndex)//xóa một hàng tại vị trí index

public void removeColumnAt(int columnIndex)//xóa một cột tại vị trí index

public Object getCurrentCell(int row, int column)//lấy dữ liệu tại một vị trí
tương ứng với hàng/cột

public void setColWidth(int[] wid)//gán chiều rộng cho cột

public void setRowHeight(int he[])//gán chiều dài cho hàng
```

- Vẽ các hàng và cột

```
private void paintColumn(Graphics g, int column)

private void paintHeader(Graphics g)
```

- Xử lý sự kiện bàn phím: lên, xuống, sang trái, sang phải  
case KEY.RIGHT, case KEY.UP, case KEY.DOWN, case KEY.FIRE

### 2.4.2 Font chữ

Ứng với mỗi thiết bị di động có hỗ trợ những bộ font riêng biệt. Vì vậy với một ứng dụng có thể bị lỗi font khi cài đặt lên những thiết bị di động khác với thử nghiệm trước đó. Font mới được xây dựng lại trong ứng dụng này là. Với dạng font mới này có thể chạy trên bất kì thiết bị di động nào có hỗ trợ J2ME mà không bị lỗi font. Font này có 2 màu chính là đen (black) và trắng (white)

Ta mã hóa hình một hình ảnh bitmap dưới dạng sau:

Định nghĩa hàm:

```
public FontBitmapH(byte fontType1, byte fontSize, byte fontColor)
```

Với các tham số

- *fontType* : Đọc định dạng font mới dưới dạng một hình bitmap
- *fontSize* : Kích thước của chữ
- *fontColor*:Màu của chữ

Ta có khai báo một *stringMap* từ hình ảnh bitmap. *stringMap* này được mã hóa dưới dạng các kí tự được đọc theo một mảng (array).

Khởi tạo biến *stringMap*:

```
stringMap = "0123456789.,!()?)+-  
/*'@abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ^#%$&;~  
";
```

Các phương thức

- Vẽ kí tự String

```
public void drawString(Graphics g, String st, int x, int y, int align)
```

- Vẽ kí tự byte

```
public void drawString(Graphics g, byte st[], int x, int y, int align)
```

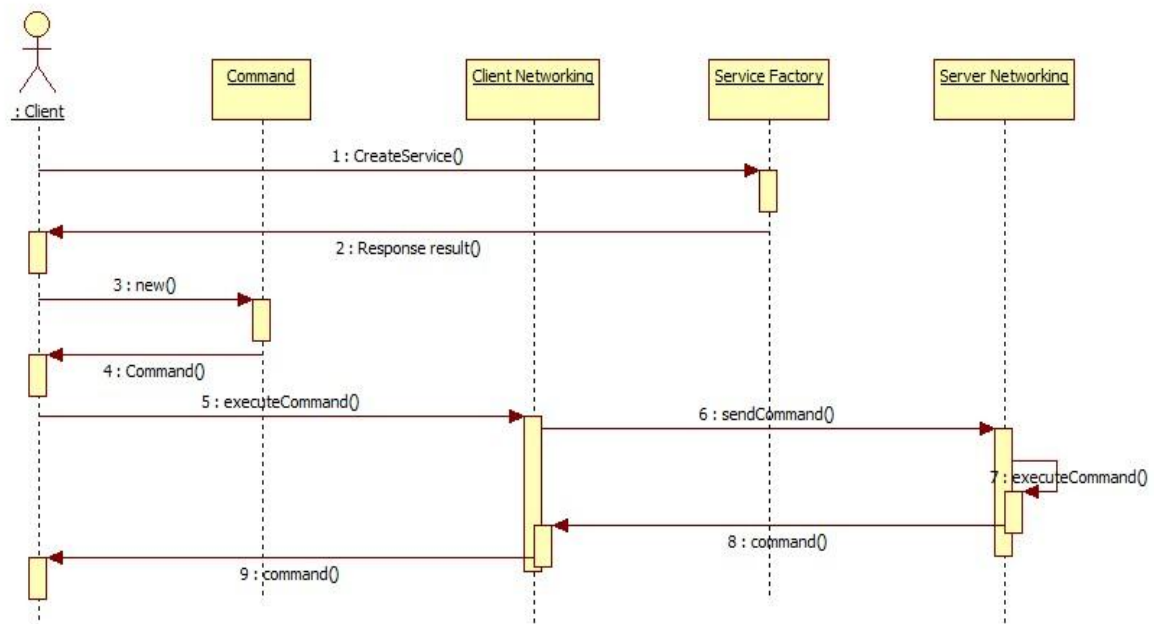
Khởi tạo một font :

```
public static FontBitmapH font_White = new FontBitmapH((byte)0, (byte)0, (byte)0);  
public static FontBitmapH font_Black = new FontBitmapH((byte)0, (byte)1, (byte)1);
```

Gọi và sử dụng font:

```
SFont.font_Black.drawString(g, getMenuText(i+m_iStartIndex),  
x + 10, YstartDraw + i * (item_offset), SFont.ALIGN_LEFT);
```

## 2.5 Thiết kế các mô hình giao tiếp giữa Client – Server



### H2.5 Sự tương tác giữa Client – Server

#### 2.5.1 Việc gửi nhận dữ liệu giữa Client – Server

Giao tiếp giữa Client – Server là trao đổi và xử lý dữ liệu thông qua mảng các byte và được truyền qua socket Connection. Mỗi khi có một request từ Client dưới dạng một command thì Server sẽ xử lý dữ liệu và trả về kết quả dưới dạng mảng các byte. Sau đó dữ liệu này sẽ được chuyển hóa thành dạng dữ liệu thông thường

**Dữ liệu được gửi dưới dạng các Message ứng với các tham số ta có hàm sau:**

```
public CMessage( int commandAction,int serviceType, byte[] data)
```

Trong đó:

- commandAction là lệnh nhận để xử lý
- serviceType
- data được lưu dưới dạng các byte

❖ **Ta có các hàm chuyển đổi**

```
public static byte[] intToBytes(int value)
```

```
{
    byte byteArray[] = new byte[4];
    for (int i = 3; i >= 0; i--)
    {
        byteArray[i] = (byte) value;
        value = value >> 8;
    }
    return byteArray;
}

/**
 * Chuyển đổi mảng byte thành kiểu dữ liệu int
 * @param data
 * @return
 */
public static int bytesToInt(byte[] data)
{
    int res = 0;
    for (int i = 0; i < 4; i++)
    {
        res <<= 8;
        res |= 0xff & data[i];
    }
    return res;
}
```

❖ **Các command nhận lệnh và xử lý:**

Các command được định nghĩa ở hai dạng là request và response

Ví dụ: ta định nghĩa command login:



```
public static final int REQUEST_LOGIN = 1;  
public static final int RESPONSE_LOGIN = -1;
```

Gọi và sử dụng:

```
switch(command)  
{  
    case Command.REQUEST_LOGIN:  
        String userName = SSerializerHelper.readString(message);  
        String pass = SSerializerHelper.readString(message);  
        service.Login(userName, pass);  
        break;  
    case Command.REQUEST_GET_ALL_COURSE_ID:  
        service.doRequestAllCourseID();  
        break;  
    case Command.REQUEST_A_TRANSCRIPT:  
        System.out.println("Yêu cầu bảng điểm");  
        String hocki = SSerializerHelper.readString(message);  
        System.out.println("hoc ki: " + hocki);  
        service.doResponseA_Transcript(hocki);  
        break;
```

Ở mỗi phiên làm việc có thể có nhiều client kết nối với server và request. Ta sử dụng `IOSession` để quản lý user đăng nhập:

```
public void Login(String userID, String passWord)  
{  
    SMessage message = new SMessage(-1, Command.RESPONSE_LOGIN);  
    int status = DBProcess.getInstance().checkValidUserName(userID, passWord);  
    switch(status)
```

```
{  
    //Không đúng userName  
    case -1:  
    case -2://Không đúng pass  
        SSerializerHelper.writeInt(status, message);  
        break;  
    case 1://userName, passWord hợp lệ  
        session.m_strUserID = userID;  
        SSerializerHelper.writeInt(status, message);  
        String userName = DBProcess.getInstance().getUserName(userID);  
        SSerializerHelper.writeString(userName,message);  
        DBProcess.getInstance().updateConnectStatus(userID, 1);  
        break;  
    case 2://Trùng hợp đang được connect  
        SSerializerHelper.writeInt(status, message);  
        break;  
}  
session.sendMessage(message);  
}
```

## CHƯƠNG 3 – DEMO ỨNG DỤNG

Phần này được thể hiện trong file cài đặt

Giao diện chính của chương trình

❖ Giao diện đăng nhập:

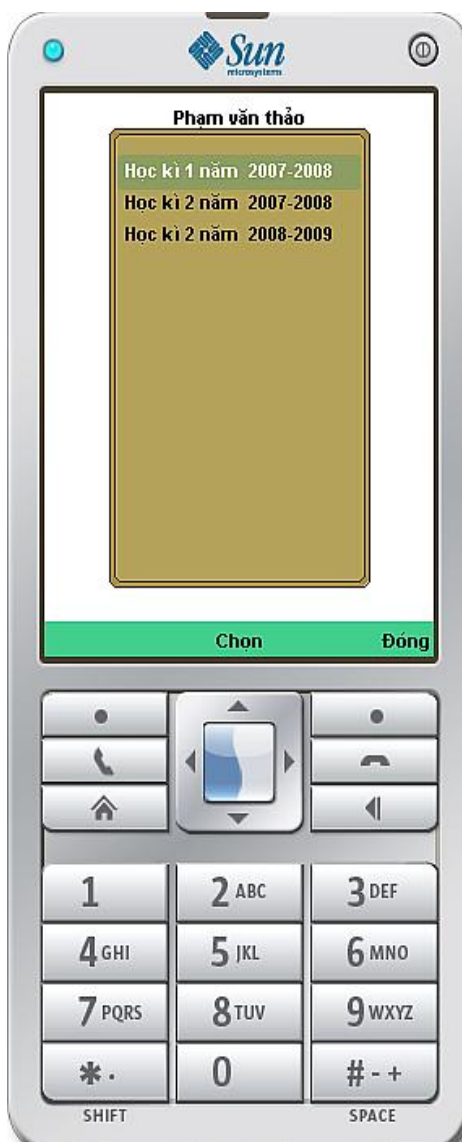


❖ Menu chính

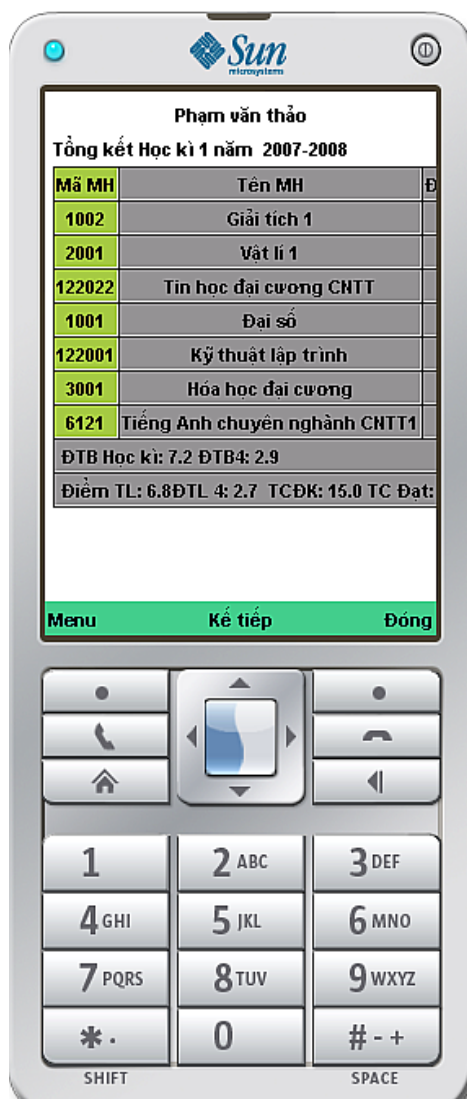


❖ Menu xem điểm:

- Xem điểm theo học kì



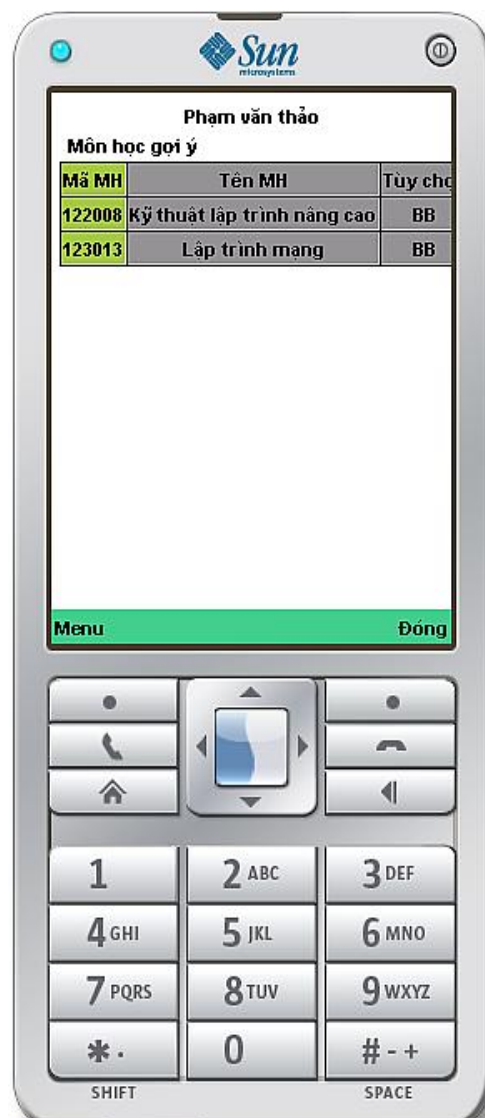
- Xem tất cả điểm



- ❖ Xem chương trình đào tạo



❖ Tư vấn đăng kí môn học





## **PHẦN 5 – KẾT QUẢ THỰC NGHIỆM ĐỀ TÀI**

Với ứng dụng Cố vấn học tập cho sẽ hữu ích cho sinh viên trong chương trình đào tạo mới với hệ thống đào tạo tín chỉ. Một ứng dụng hay không chỉ chạy trên 1 hoặc 1 vài thiết bị mà có thể sử dụng trên nhiều thiết bị. Hệ thống Client – Server làm giảm thiểu thời gian cũng như áp dụng công nghệ mới.

Qua quá trình nghiên cứu lý thuyết cũng như xây dựng ứng dụng minh họa có những ưu điểm và nhược điểm sau:

### **❖ Ưu điểm**

- Nghiên cứu và xây dựng được mô hình Client – Server chạy trên PC server và client và mobile device.
- Xây dựng được cơ sở dữ liệu môn học cho sinh viên, cơ sở tri thức bổ sung bằng một số luật cơ bản để tư vấn đăng kí môn học cho sinh viên.
- Giúp cho Sinh viên có thể chọn lựa đúng môn học theo chuyên ngành của mình và nhanh chóng hơn việc thực hiện trên web.
- Chạy được với nhiều thiết bị cùng lúc.

### **❖ Nhược điểm**

- Chưa xây dựng được một hệ chuyên gia trên cơ sở tri thức xây dựng tư vấn đăng kí môn học cho sinh viên.
- Chưa xây dựng tính năng đăng kí môn học: điều này cần thử nghiệm trên server thực nên gặp khó khăn.

## **PHẦN 6 – HƯỚNG PHÁT TRIỂN**

Trong quá trình thực hiện đề tài còn nhiều vấn đề chưa được giải quyết thỏa đáng. Còn nhiều điểm chưa chuyên sâu

- Chạy được trên môi trường thực tế với nhiều thiết bị client.
- Xây dựng hoàn chỉnh cơ sở dữ liệu, cơ sở tri thức.
- Xây dựng hệ chuyên gia tư vấn đăng kí môn học cho sinh viên. Với hệ thống tư vấn heuristic hỗ trợ đắc lực cho sinh viên trong việc định hướng môn học, ngành học, tư vấn cho sinh viên như những chuyên gia thực thụ.
- Xây dựng tính năng đăng kí môn học.

## PHẦN 7 – TÀI LIỆU THAM KHẢO

### Tiếng Việt

[1]. *Mobile Programming* – TH.S Trịnh Công Duy – Trường ĐH Bách Khoa Đà Nẵng

### Tiếng Anh

[1]. *Core J2ME* – John W. Muchow

[2]. *A Tutorial on Socket Programming in Java* - Natarajan Meghanathan

[3]. *Java™ Network Programming and Distributed Computing* - David Reilly,  
Michael Reilly

## **PHẦN 8 – PHỤ LỤC**

Các từ viết tắt dùng trong báo cáo

- J2ME: Java 2 Platform, Micro Edition
- J2SE : Java 2 Platform, Standard Edition
- CDC: Connected Device Configuration
- CLDC: Connected Limited Device Configuration: Thiết bị di động cấu hình thấp
- KVM : Kevil Virtual Machine – là dạng máy ảo dùng cho thiết bị cấu hình thấp
- MIDP: Mobile Information Device Profile
- RMS: Record Management System
- MIDLet: tên gọi chung cho các ứng dụng của J2ME