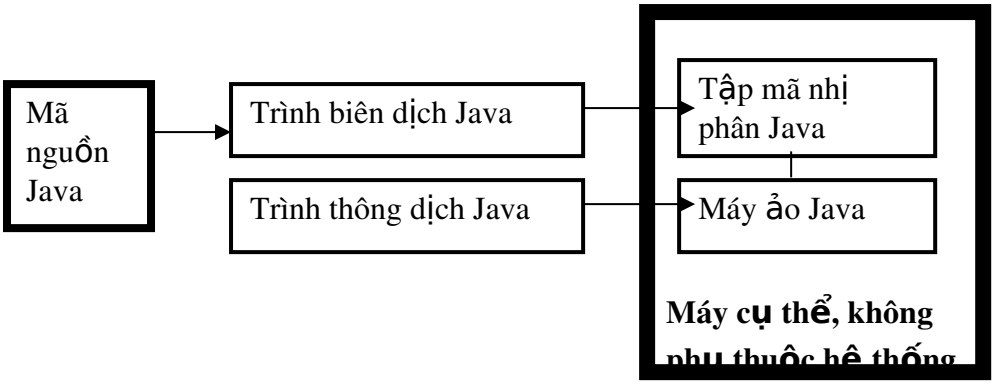
CHƯƠNG 1: LÀM QUEN VỚI MÔI TRƯỜNG J++6.0

1.1 Nguồn gốc Java

Ngôn ngữ lập trình Java được phát triển bởi hãng Sun Microsystem dưới dạng một ngôn ngữ thuần hướng đối tượng. Ngôn ngữ này ngoài việc dùng để thiết kế các chương trình giải quyết các vấn đề kinh tế, khoa học, kỹ thuật, nó còn được dùng trong lĩnh vực Internet World Wide Web (WWW). Có lẽ không có ngôn ngữ nào "gặp may" như ngôn nhữ này. Sun Microsystem vừa công bố lần đầu vào năm 1995 (có lẽ các thành viên chính của hãng Sun rất thích uống cà phê xuất xứ từ đảo Java của Indonexia nên ngôn ngữ này được mang tên Java chăng ?) đã lập tức gây được sự chú ý trên toàn thế giới. Hãng Sun Microsystem dường như đã nhìn thấy trước mối liên hệ giữa sự phát triển nhảy vọt của mạng Internet và nhất là của mạng World Wide Web với ý tưởng mà họ gửi gắm vào ngôn ngữ này: Tham gia vào mạng Internet là máy tính thuộc đủ mọi loại, chẳng lẽ cứ phải chờ khi nào "anh" có máy tính cùng loại như "tôi" và hệ điều hành của "anh" cũng giống như của "tôi" thì chúng ta mới "nói chuyện" được với nhau ư?. Java xoá bỏ ranh giới đó bằng công nghệ mà Sun Microsystem gọi là "máy ảo Java". Nghĩa là chương trình được biên dịch thành "mã máy" của máy ảo Java, sau đó bất cứ máy cụ thể nào cũng có thể chạy được miễn là máy đó có chứa "máy ảo Java". Máy ảo Java là một tập các quy ước về chỉ thị, dạng thức lưu trữ, thanh ghi v.v.. như máy tính thực sự và trình thông dịch Java tạo ra máy ảo Java này.



Trình thông dịch có nhiệm vụ chuyển đổi chỉ thị của máy ảo Java thành máy tính cụ thể vào lúc chạy trình ứng dụng. Như vậy Java vừa biên dịch lại vừa thông dịch. Rõ ràng trình ứng dụng muốn chạy được phải qua hai giai đoạn: Biên dịch để tạo tập mã của máy ảo Java và thông dịch để tạo tập mã của máy đặc thù.

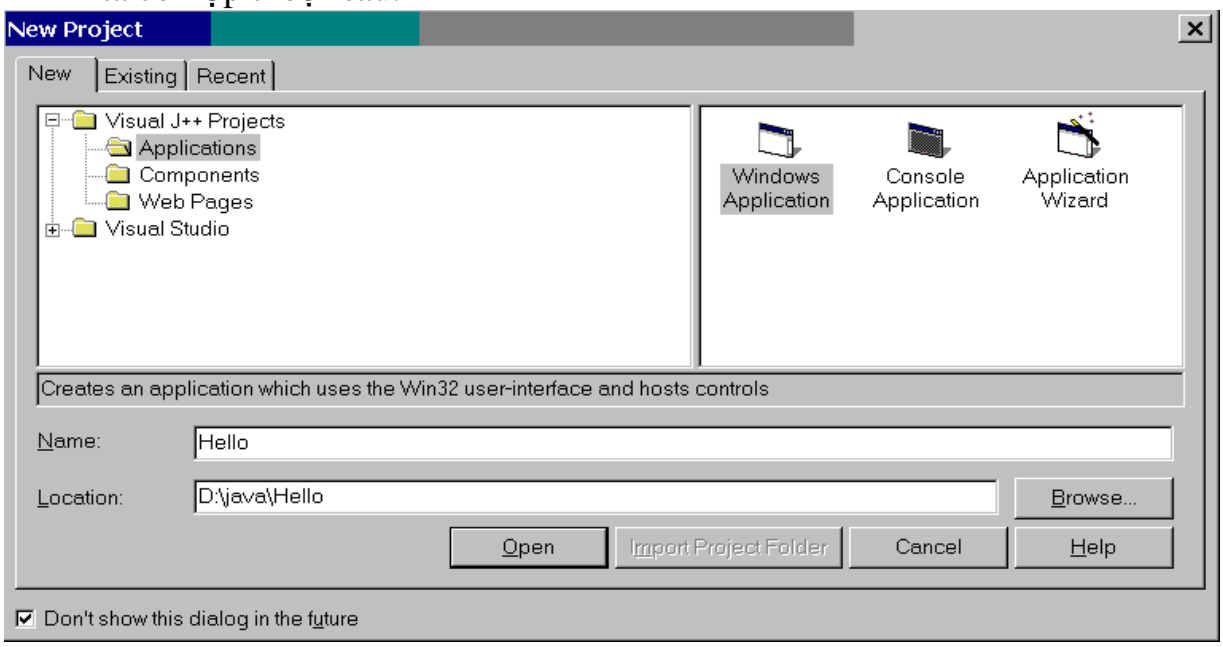
Thành công của Netscape về trình duyệt thức tỉnh Sun, hãng đã bắt tay vào thiết kế một trình duyệt có khả năng đọc và thực thi mã Java đặt trên Web. Trình duyệt này gọi là Hotjava. Bằng cách này văn bản Web không còn là những câu văn hay hình ảnh bất động và thế là Java đã đi tiên phong trong việc thiết kế các Web động- thực chất là chạy các Applet Java trên Web.

Hãng Microsoft thấy được tính ưu việt của Java nên đã bắt tay xây dựng môi trình lập trình cho ngôn ngữ này. Sau khi đã mua bản quyền, Microsoft đã tung ra công cụ lập trình Java có tên J++ và đến nay đã nâng cấp lên Visual J++6.0. Chúng ta sẽ lập trình Java trên môi trường Visual J++6.0 này.

1.2 Tạo một Project mới

1. Khởi động Visual J++6.0:

Chọn **Start/Program/Microsoft Visual Studio/Visual J++6.0,** sau đó ta có hộp thoại sau:



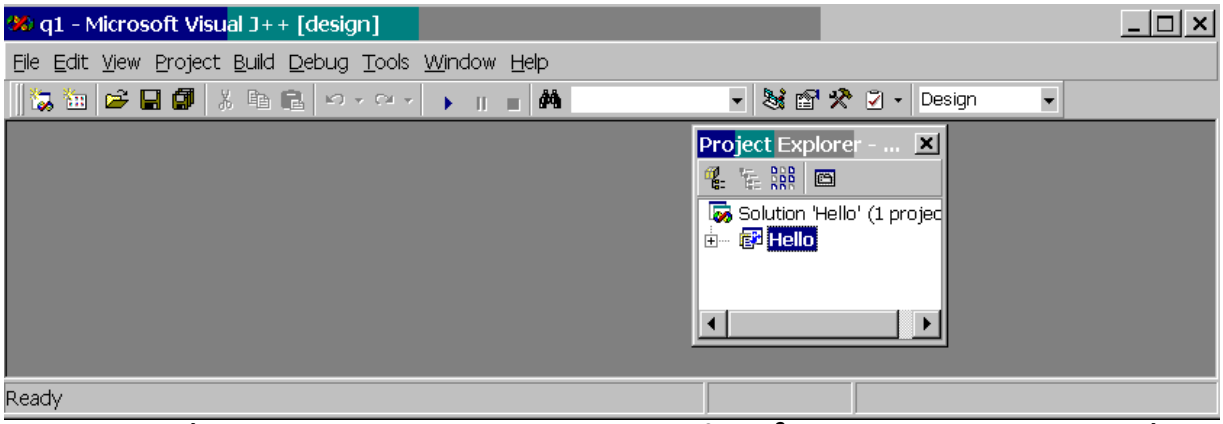
Chọn **Applications** sau đó chọn **Windows Applications**

Chọn thư mục ở **Location, (D:\Java)**

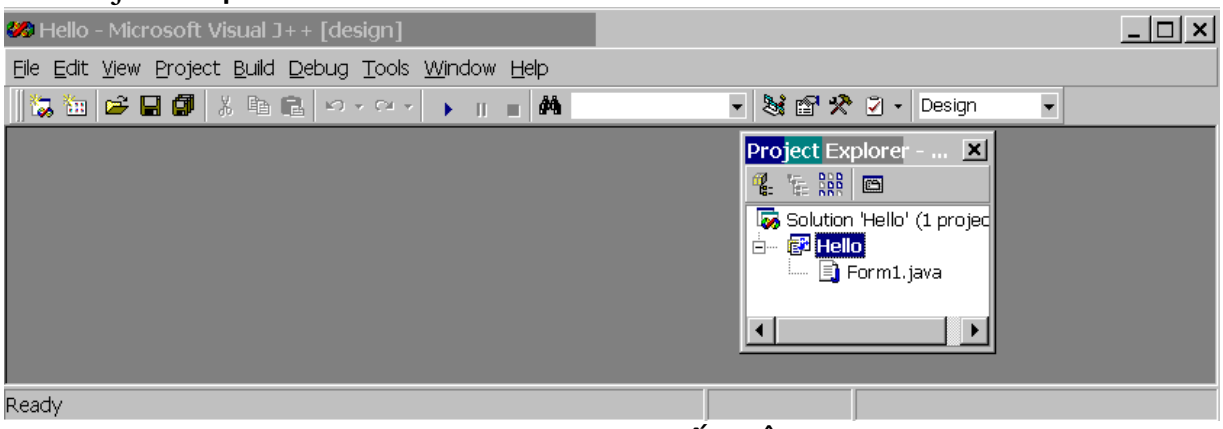
Gõ tên **Hello** vào hộp Name

Chọn Open

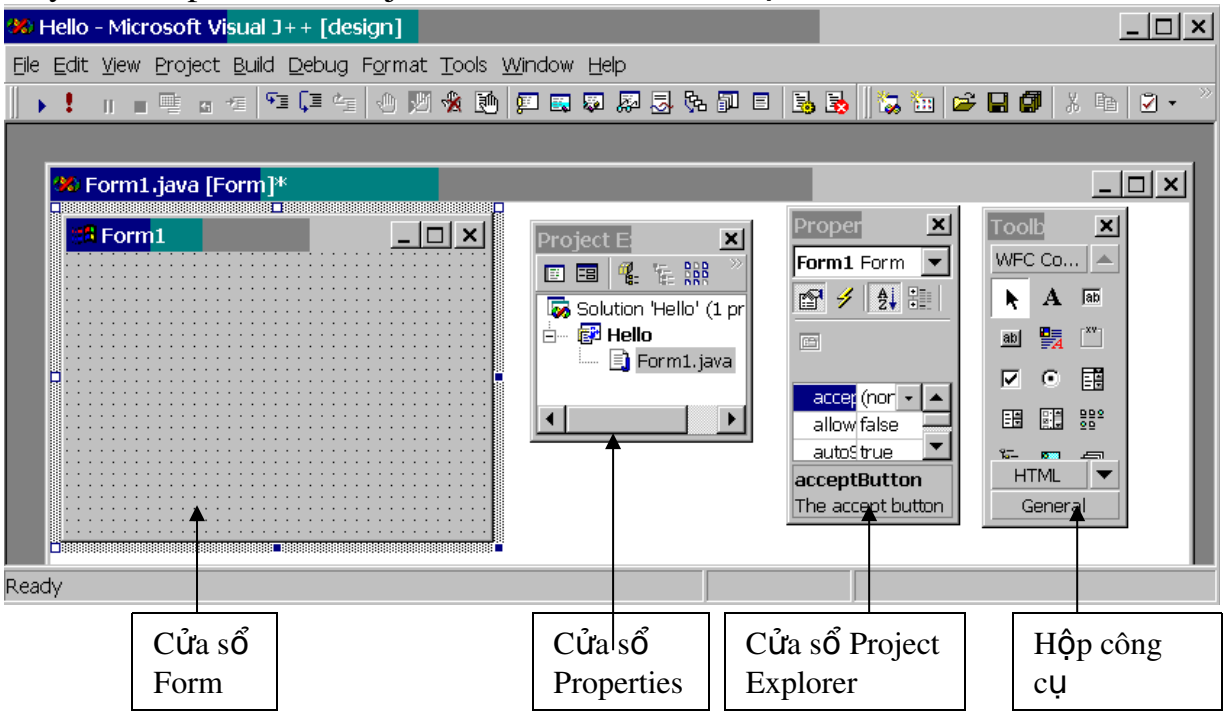
Sau đó ta có hình sau:



Kích vào dấu cộng (+) (trước Hello) trong cửa sổ Project Explorer, ta thấy Form1.java hiện ra:



Hãy kích đúp vào Form1.java, và hình sau xuất hiện:



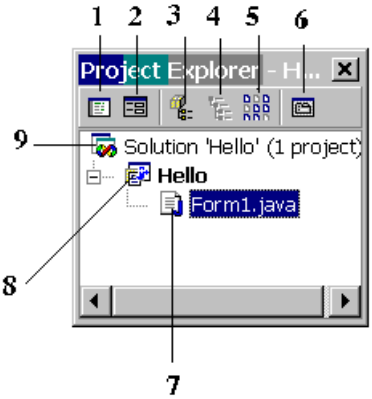
Như vậy ta có một Project có tên **Hello** trong thư mục **D:\java\Hello**.
(Ta quy định là các tệp để trong thư mục D:\Java- Ta đã tạo trước).

2. IDE là gì?

Visual J++6.0 là một thành phần trong họ Microsotf Visual Studio 6.0. Trong họ này bao gồm: Visual Basic, Visual FoxPro, Visual C++, Visual InterDev,Visual SourceSafe và MSDN Library. Các thành phần này có chia sẻ không gian làm việc chung gọi là Integrated Development Environment (môi trường phát triển tích hợp-IDE). Ta nhận thấy cửa sổ trên có chứa nhiều cửa sổ khác và nó là một IDE, sau đây là một số thành phần trong IDE

Project Explorer

Chọn View/ **Project Explorer** để hiện nó. Dạng của Project Explorer như sau:



Khi ta chọn Form1.java thì 6 thành phần trên hiện ra. Ý nghĩa của các thành phần đó như sau:

View Code: Hiển thị mã

View Designer: Kích hoạt Form

Package View: Hiển thị gói

Directory View: Hiển thị các thư mục

Show all Files : Hiển thị các File

Properties: Hiện thị cửa sổ Properties

Form1.java

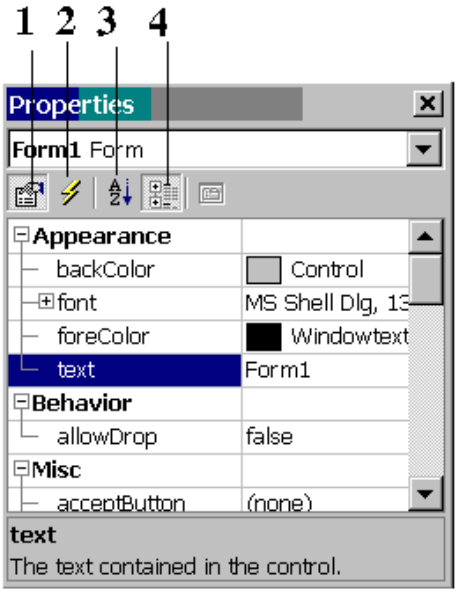
Project có tên Hello

Solution có tên Hello chứa một project

Dùng Project Explorer để điều hành qua lại giữa các bộ phận là rất thuận lợi và nhanh chóng.

Cửa sổ Properties

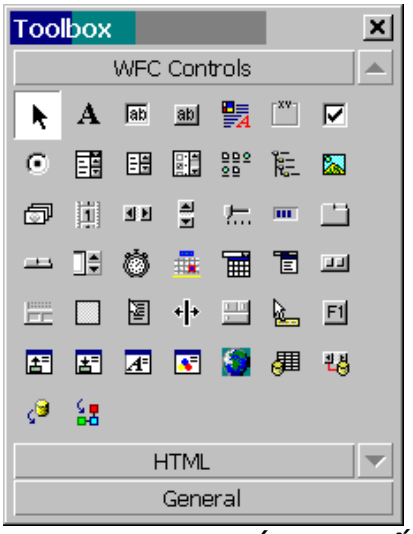
Hiện nó bằng cách: Chọn View/Properties Window. Dùng cửa sổ này để đặt các thuộc tính cho các đối tượng. Cửa sổ này có dạng:



Nếu chọn 1 các thuộc tính hiện ra trong cửa sổ, còn nếu chọn 2 các sự kiện (events) sẽ hiện ra.

Hộp công cụ- Toolbox

Để hiện hộp công cụ hãy chọn: View/Toolbar, ta có dạng sau:



Hộp công cụ có chứa các đối tượng, bạn tìm hiểu đặc tính của các đối tượng ở những phần sau, nhưng trước hết hãy tìm hiểu qua một ví dụ nhỏ của mục 3.

3. Ứng dụng đầu tiên

Chọn New Project

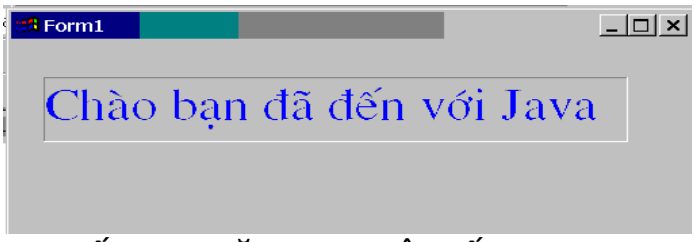
Gõ tên Project là Hello1

Hiện Form1

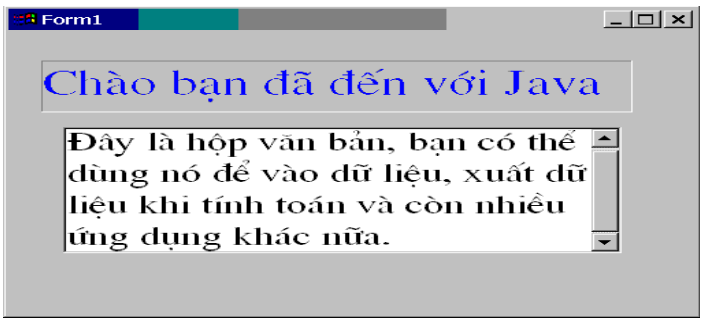
Hiện ToolBox và đặt một Label lên Form

Mở Properties, chọn Text và gõ vào:"Chào bạn đã đến với Java"

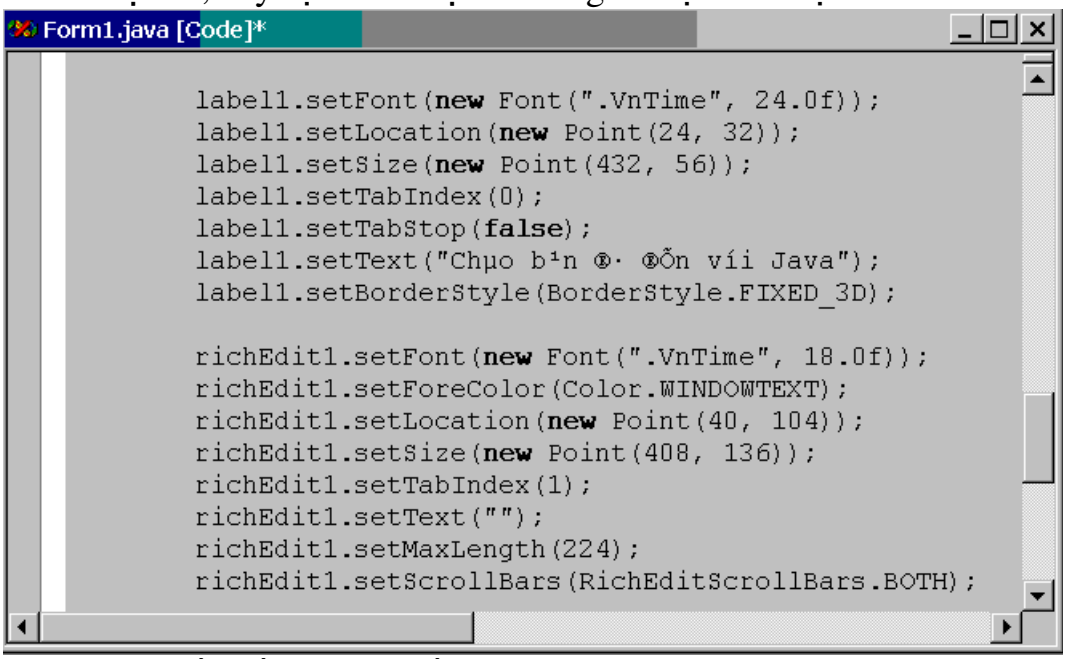
Biên dịch: Chọn Build/Build
Chạy: Chọn Debug/Start, sau đó ta có hình ảnh:



Nếu bạn đặt thêm một đối tượng khác, giả sử đối tượng RichEdit (hộp văn bản), khi chạy bạn có thể gõ dữ liệu vào như sau:



Bạn có thể thắc mắc là mã của chương trình này ở đâu? trong môi trường J++, mọi thứ đều trực quan và riêng biệt. Bạn kích đúp vào Form1 cửa sổ Code hiện ra, hãy dịch lên một số dòng và bạn có đoạn mã sau:



Khi bạn thiết kế với các đối tượng như Form, Label, RichEdit v.v.. thì máy đủ "khôn ngoan" để "dịch" ra câu lệnh ứng với mỗi thao tác của bạn. Điều đó chúng ta sẽ còn đề cập đến ở các chương sau.

Chú ý:

a. Qua các phần trên bạn thấy khái niệm "tệp" rất phong phú, cụ thể:

Solution: Giải pháp, tệp giải pháp (*.sln=Solution)

Project: Dự án, tệp dự án (vjp=Visual J++ Project)

Trong một Solution có thể có nhiều Project, trong một Project có thể có nhiều dạng File khác nhau .

b. Hãy phân biệt các dạng File trong J++ thông qua giao diện File như sau:

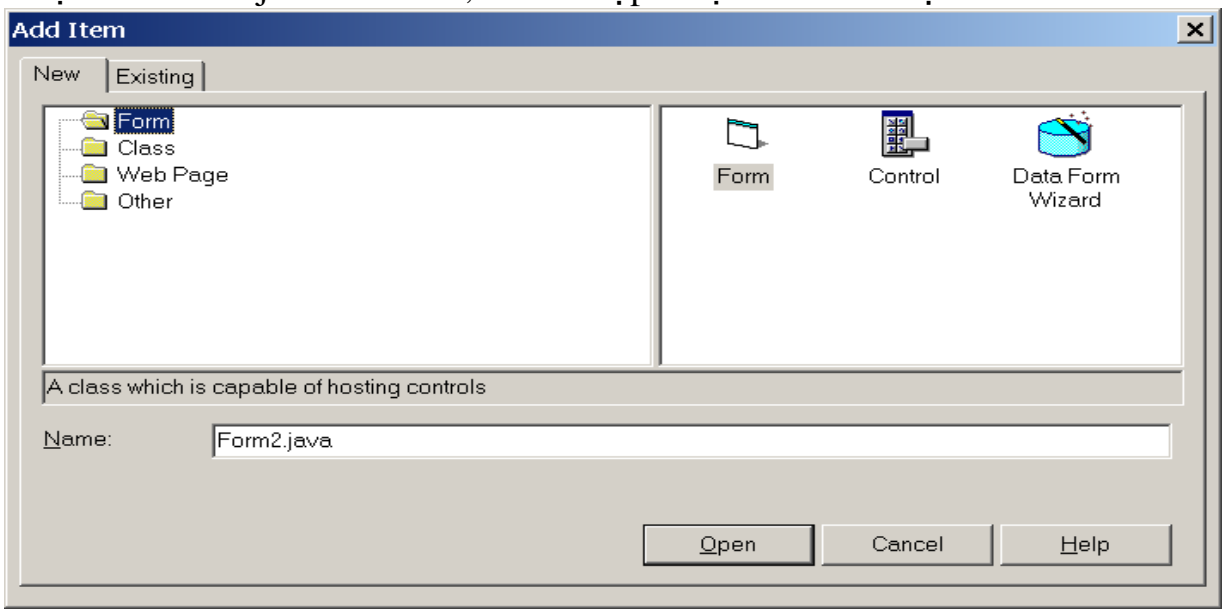
All Project Files (*.sln;*.dsw;*.vjp;*.dbp;*.pkp;*.vup;*.dsp)
Solution Files (*.sln)
Compatible Workspace Files (*.dsw)
Visual J++ Project Files (*.vjp)
Database Project files (*.dbp)
Distribution Units (*.pkp)
Utility Project files (*.vup)
Compatible Project Files (*.dsp)

4. Bổ sung các File vào Project

Có thể có nhiều nguồn tài nguyên khác nhau tạo nên một Project ví dụ: các File Java, các trang HTML, các điều khiển và các thành phần khác. Nhiều khi ta muốn bổ sung một tệp ở ngoài vào Project, ta thao tác như sau:

Mở Project Explorer, chọn tên Project (giả sử Hello)

Chọn Menu: Project/Add Item, sau đó hộp thoại sau xuất hiện:

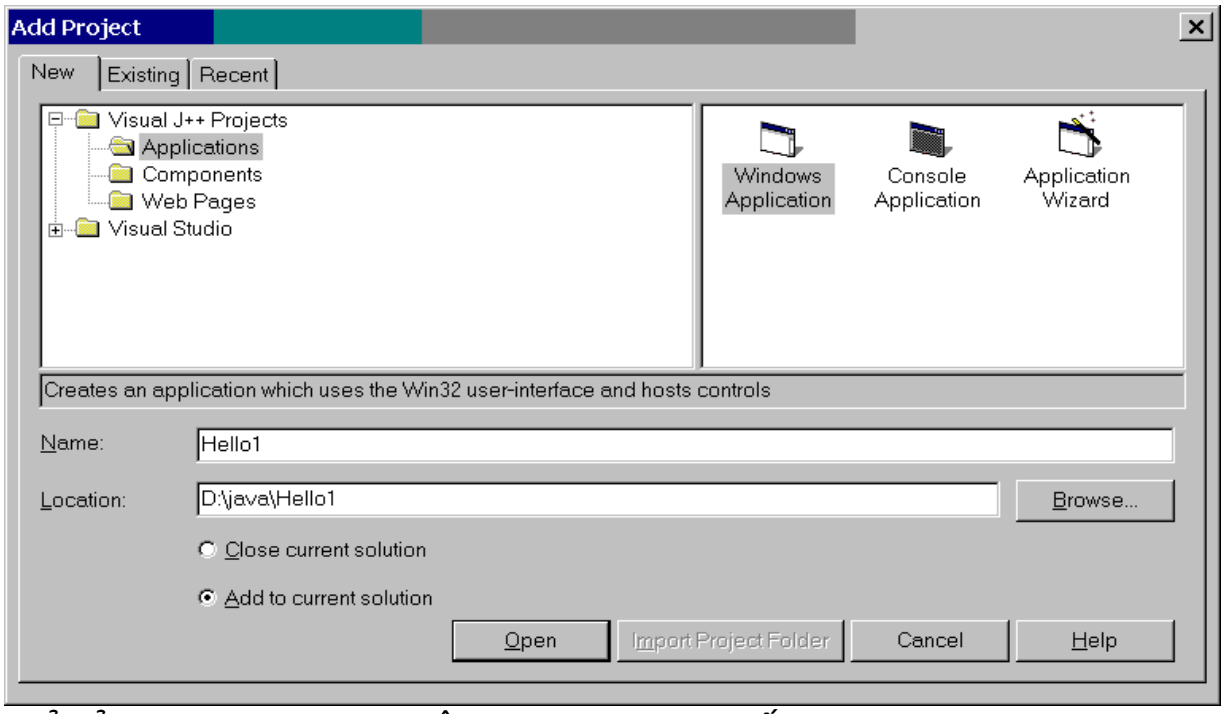


Chọn Form, tên trong hộp Name là Form2.java, chọn Open. Như vậy ta đã tải tệp Form2.java vao tệp Ptoject Hello

5. Bổ sung các Project vào Solution

Một Solution có thể có nhiều Project, nhiều khi ta muốn bổ sung một Project ở ngoài vào một Solution nà đó, hãy thao tác:

Chọn Solution trong Project Exprorer
Chọn File/Add Project, sau đó ta có hộp thoại:



Giả sử bạn chọn như trên hộp thoại, hãy chọn tiếp Open.
Như vậy bạn đã đưa tệp Project có tên Hello1 vào Solution

6. Đặt lại tên cho Solution và Project

Lúc đầu sau khi tạo, Solution tự đặt tên cho mình trùng tên Project, ví dụ trên là tên Hello. Giả sử có hai Project trong Solution bạn nên đặt tên lại cho Solution thì hợp lý hơn. Hãy đổi tên như sau:
Kích phím chuột phải vào biểu tượng Solution và chọn Rename sau đó hãy gõ tên mới vào. Với Project cũng tương tự.

7. Lưu các Solution, Project, các File khác

Lưu toàn bộ

Chọn File/Save All

Lưu một File cụ thể

Vào cửa sổ chứa File cần lưu, chọn File/Save

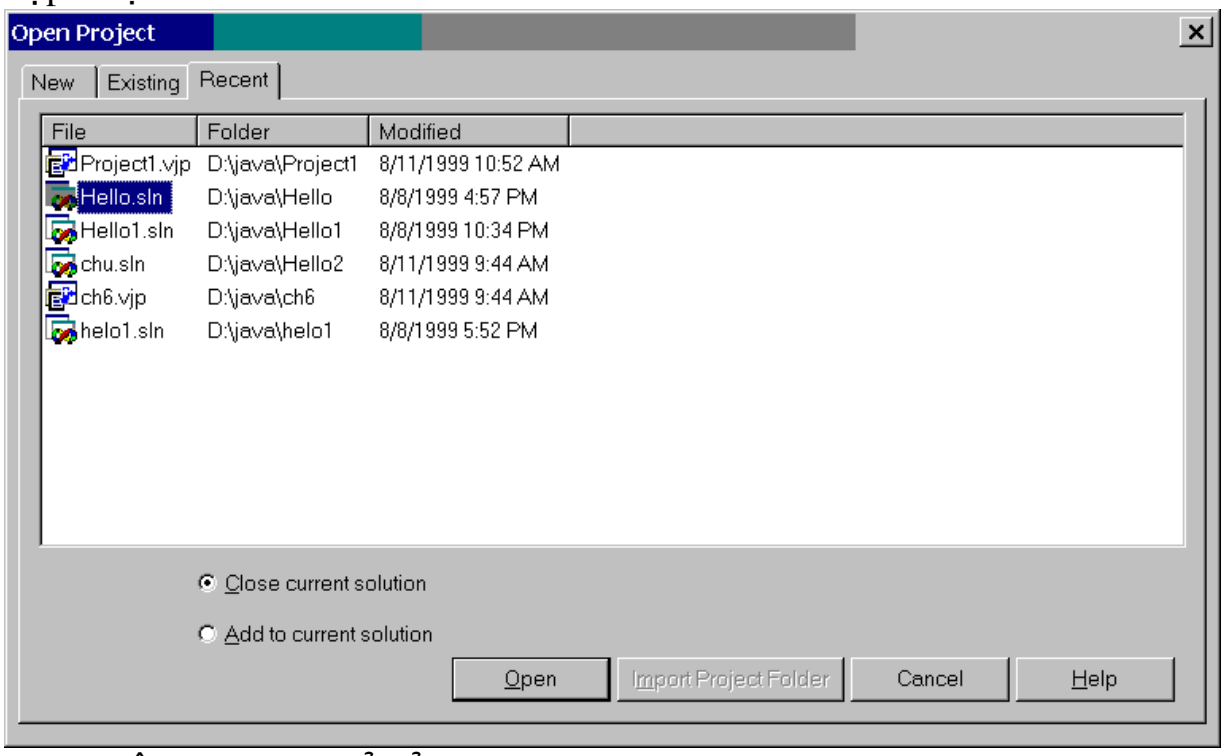
8. Đóng các Solution, Project, các File khác

Chọn File Close all

9. **Mở các Solution , Project**

Khi mở một Solution, đồng thời bạn mở luôn các thứ lưu trong đó. Nhưng nếu bạn mở một Project mà không mở Solution thì Project đó được nằm trong một Solution trống, bởi vì Project mà bạn định mở có thể có nhiều Project khác trong cùng Solution. Vì vậy bạn nên mở Solution thay vì mở Project.

Chọn File/Open Project, một hộp thoại hiện ra và hãy chọn Recent ta có hộp thoại:



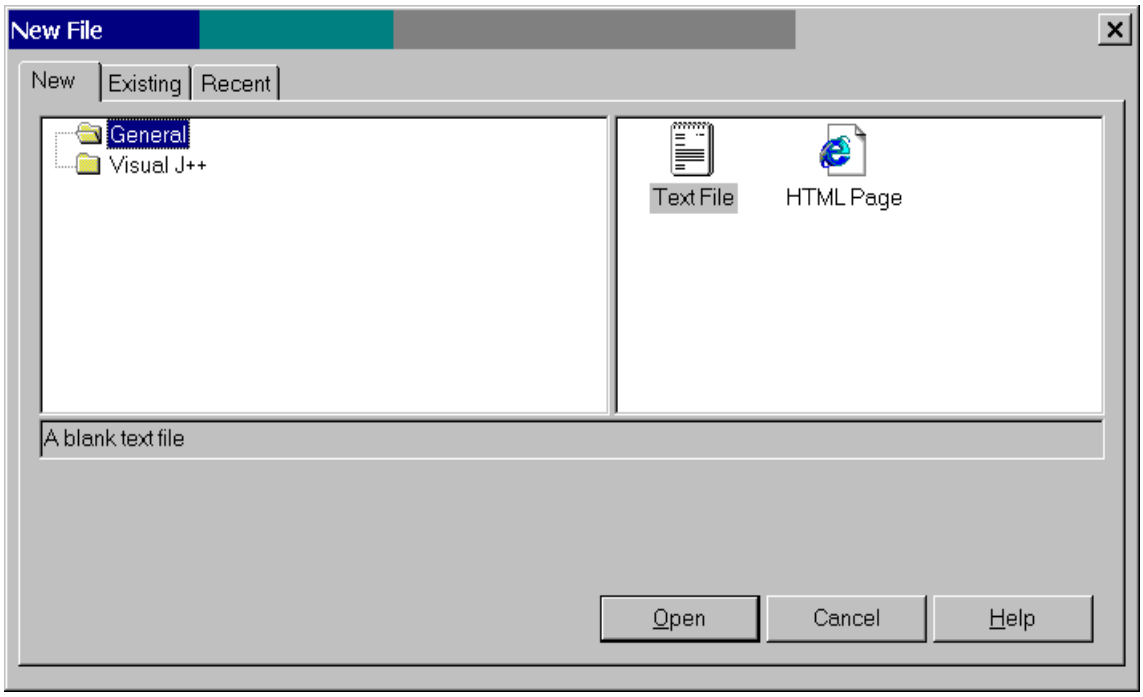
Chọn một Solution (giả sử chọn Hello.sln) và chọn Open

10. Tạo và mở các File riêng biệt

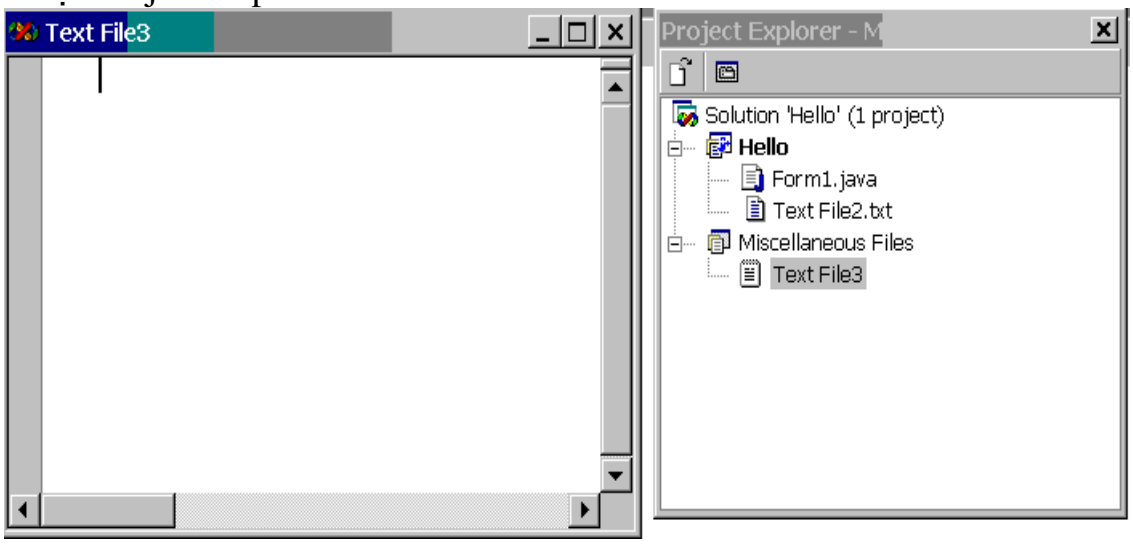
Chương trình Visual J++ được tao ra bằng cách sử dụng Solution và Project. Một Solution có thể chứa 1 hoặc nhiều Project, trong Project lại có thể chứa 1 hoặc nhiều dạng File khác. Dùng cấu trúc Solution giúp bạn dễ quản lý. Tuy nhiên có lúc bạn cần chỉ tạo một File chứ không phải tạo mội thứ trong Solution. Visual j++ cho phép bạn tạo và mở các file riêng biệt vốn không phải là thành phần của một Project nào đó. Đồng thời bạn cũng phải hiểu là các File trong Visual J++ đều phải có "xuất xứ" (tức là ở trong một Project cụ thể), cho nên tam thời chấp nhận tên Project là Miscellaneous Files (khôngđồng loại).

Để tạo một File riêng biệt:

Chọm File/NewFile sau đó ta có hộp thoại:



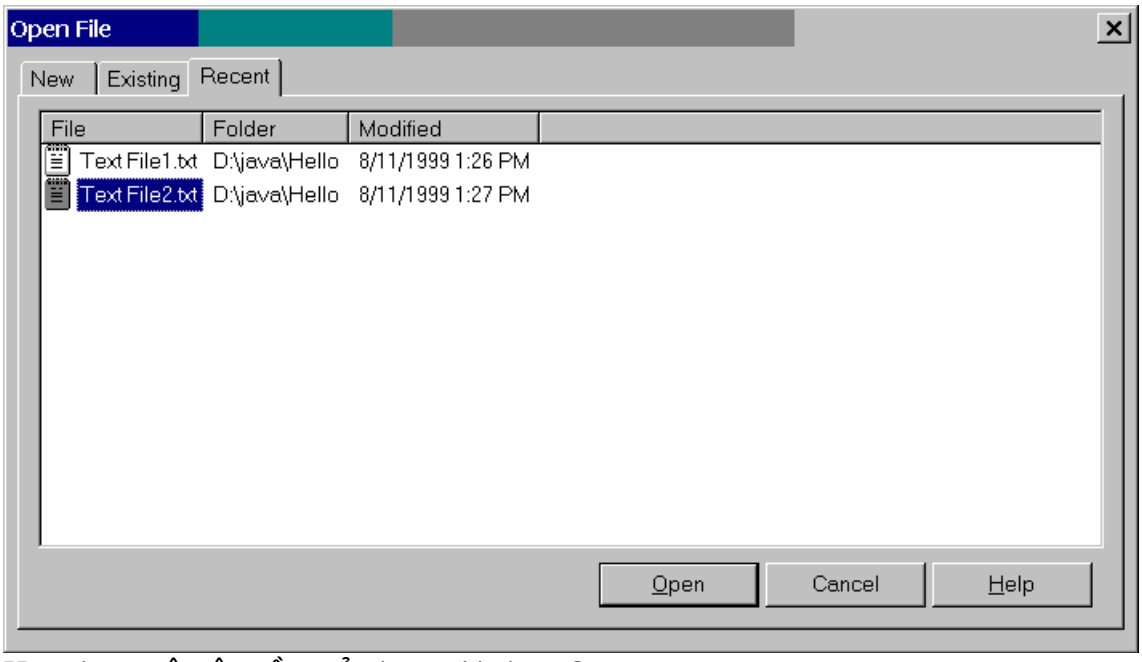
Ở trên, giả sử bạn chọn General và TextFile, hãy chọn Open ta có hộp thoại Project Exporer như sau:



Vì Solution đang mở nên bạn đã gắn thêm một Project **Miscellaneous Files** vào đó, trong Project này có File TextFile3, bạn gõ nội dung vào cửa sổ File !

b. Để mở File riêng biệt

Chọn File/Open File, ta có hộp thoại:



Hãy chọn một tệp cần mở và sau đó chọn Open.

11. Thoát khỏi Visual J++

Chọn **Exit/File** hoặc kích vào nút **Close** (X) của cửa sổ **Designer**

Bài tập chương 1:

1. Ngôn ngữ Java khác các ngôn ngữ biên dịch khác ở những điểm nào?
2. Thế nào là máy ảo Java ?
3. Nói rằng Visual j++6.0 là ngôn ngữ Java, đúng hay sai?
4. Thế nào là trình duyệt có khả năng Java?
5. Tuy chưa giới thiệu gì nhiều về môi trường Visual J++6.0, nhưng bạn hãy tự tìm hiểu về các điều khiển trên Toolbar!
6. Solution và Project khác nhau như thế nào?
7. Hãy tạo một Project mới trong một Solution và đặt điều khiển Edit lên Form.Hãy đặt cho Edit một số thuộc tính bằng cửa sổ Properties. Sau đó chạy dự án đó.
8. Hãy tạo một Project mới trong một Solution và đặt điều khiển PictureBox lên Form.Hãy đặt cho PictureBox này một số thuộc tính trong đó có thuộc tính Image. Sau đó chạy dự án đó.
9. Hãy thao tác để quen với hai hành động sau:
   Mở một Project mói
   Mở một Project hiện có
10. Hãy thao tác để đưa một Project vào một Solution
11. Hãy thao tác để đưa một File vào một Project
12. Hãy tạo một File Text không nằm trong một Project

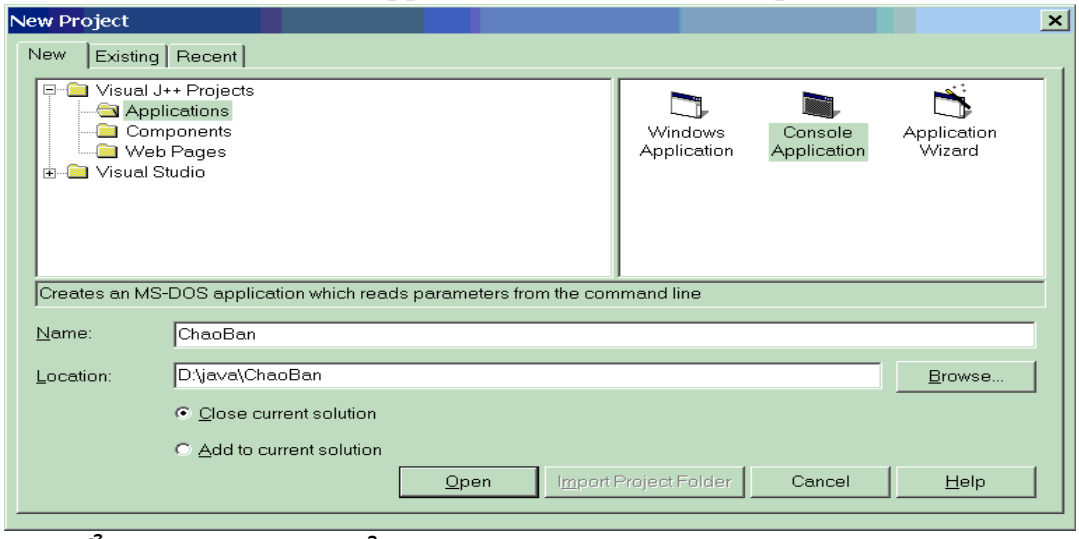
CHƯƠNG 2: LẬP TRÌNH JAVA

2.1 Tạo và chạy một chương trình Java

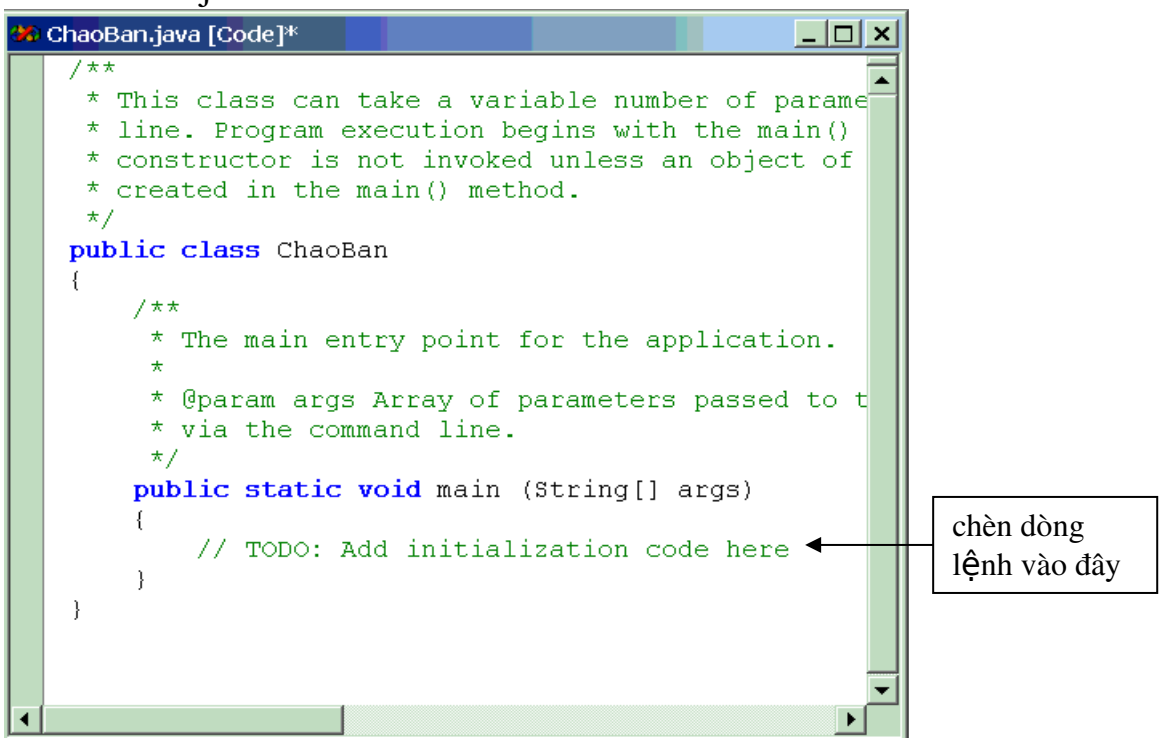
2.1.1 Tạo chương trình

Để làm quen với một ngôn ngữ mới, chúng ta nên bắt đầu tạo một chương trình đơn giản. Thường thì J++6.0 đã tạo cho chúng ta một "khuôn dạng" sẵn, chúng ta chỉ cần thêm những lệnh vào những vị trí thích hợp. Hãy thao tác theo các bước sau:

Khởi động J++6.0, chọn File/New Project, hộp thoại New Project xuất hiện, chọn Console Application, đưa vào tên tệp **Chaoban** như hình sau:



Đổi tên Class1.java ở Project Explorer thành ChaoBan.java. Kích đúp vào ChaoBan.java ta có cửa sổ Code có chứa mã như sau:



Tạm thời xoá các chú thích (các dòng nằm trong cặp /*.....*/ hoặc sau // và đưa dòng lệnh ***System.out.println****("Chào bạn đã đến với Java")* vào thân hàm **main**, ta có chương trình sau:

```
public class ChaoBan
   {
      public static void main (String[] args)
           {
             System.out.println("Chào bạn đã đến với Java");
           }
   }
```

Rồi bạn sẽ tìm hiểu cấu trúc sau, nhưng nhìn chung chương trình này có chứa một lớp (**class**) có tên **ChaoBan**, thân của lớp là hàm **main**, hàm này chỉ có một nhiệm vụ là hiển thị câu "Chào bạn đã đến với Java" ra màn hình. Như vậy một chương trình về hình thức là một class.

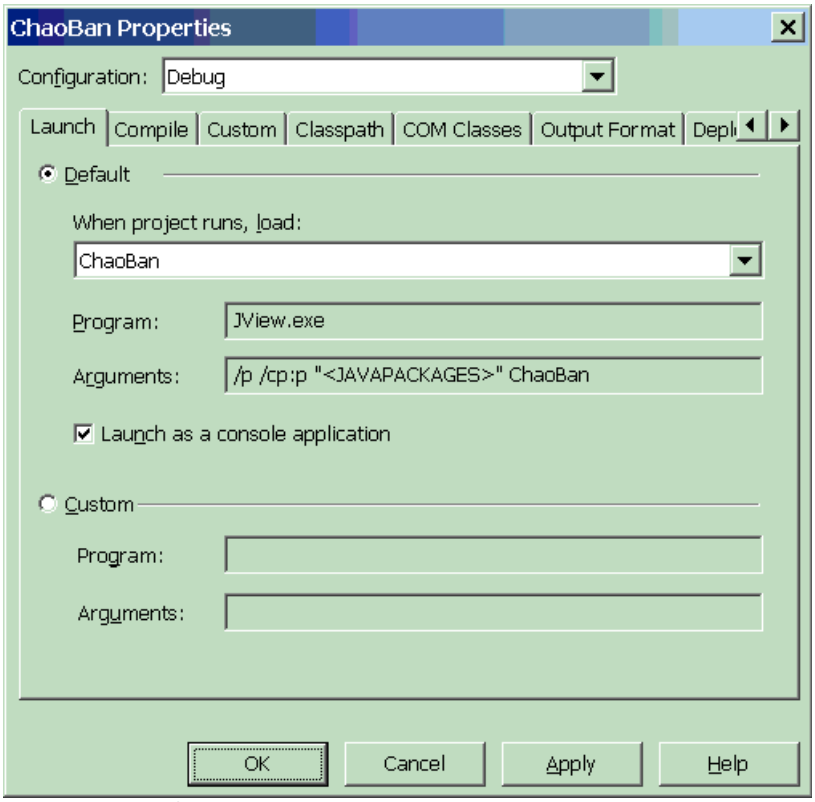
2.1.2 Chạy chương trình

Sau khi soạn và lưu chương trình xong, chúng ta phải thực hiện một số thao tác trước khi có thể xem kết quả chạy chương trình.

J++6.0 có chứa hai trình diễn dịch Java là **jview** và **wjview**. jview được dùng trong môi trường dòng lệnh, như MSDOS chẳng hạn. Wjview dùng trong môi trường window với giao diện đồ hoạ (GUI).

1. **Chạy chương trình ChaoBan.java với jview**

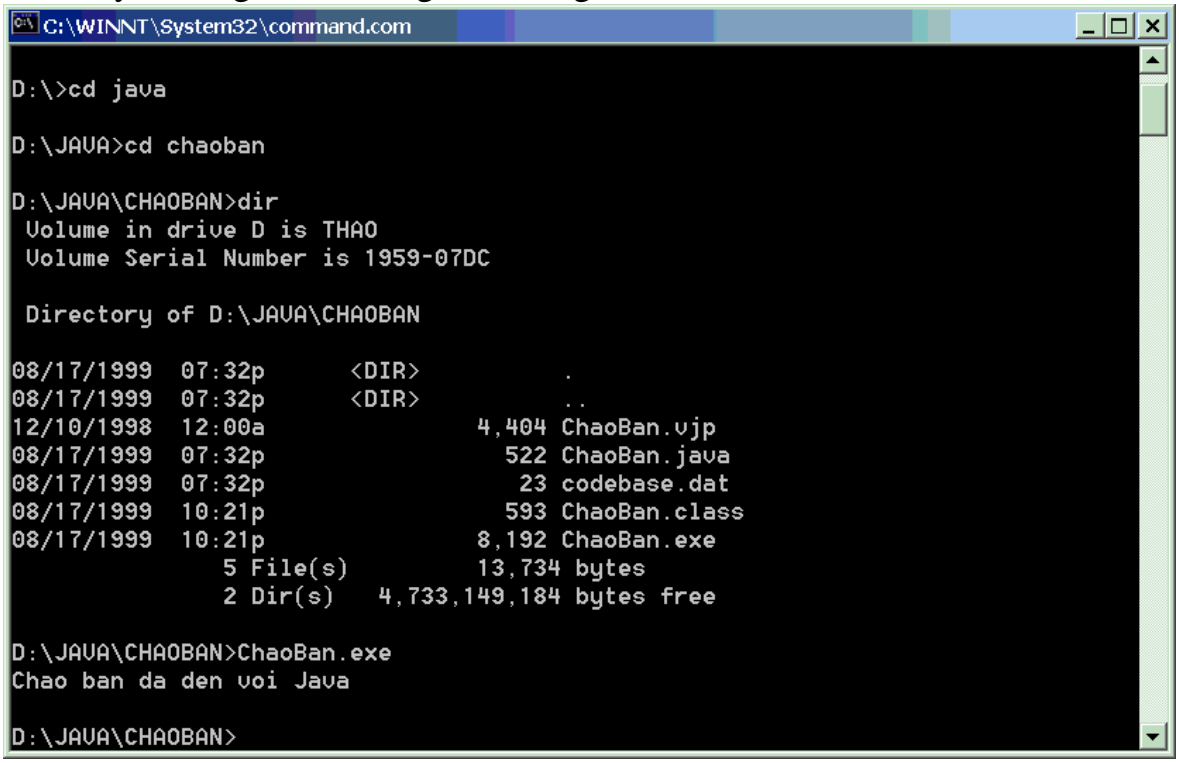
Chọn **Project/ChaoBan Properties**, sau đó ta có hộp thoại:



Đặt nút kiểm tại **Launch as a console application** và chọn **OK**

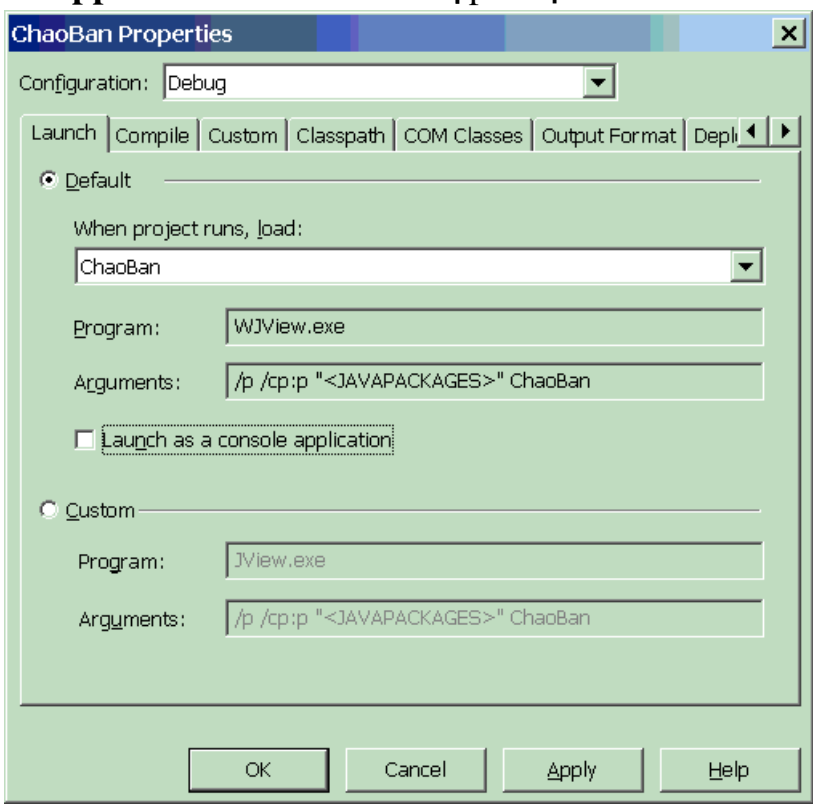
Chọn **Build/Build**

Chuyển sang môi trường DOS và gõ như hình sau:



2. Chạy chương trình ChaoBan.java với wjview

Chọn **Project/ChaoBan Properties**, huỷ hộp kiểm **Launch as a console application** sau đó ta có hộp thoại:

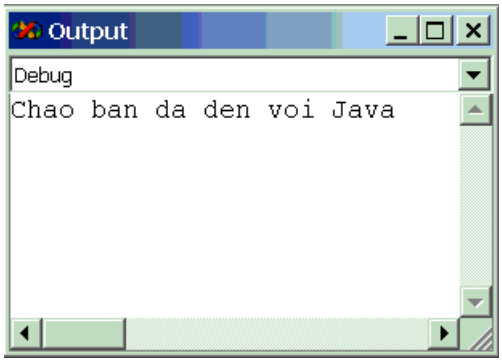


Chọn **OK**

Chọn **Build/Rebuild**

Chọn **View/Other Windows/Output**

Chọn **Debug/Start**, sau đó ta có kết quả:



2.1.3 Chú thích trong chương trình

Để giúp người đọc dễ nhận biết đoạn chương trình, bạn nên đưa vào các chú thích. Chú thích trên một dòng, hãy dùng //, trên nhiều dòng hãy dùng cặp:/*...*/

Ví dụ:

```
/*
 Đây là  class có tên Class1
Bạn có thể xem qua chương trình này  */
```

```
public class Class1
  {
      //  Sau đây là hàm main
      public static void main (String[] args)
      {
            // Thân hàm
      System.out.println("Chào bạn đã đến với Java");
      }
  }
```

Chú ý:

Bạn có thể sửa chương trình lại theo ý của bạn, nhưng sau đó phải dịch lại thì mới có tác dụng.

2.2 Hằng, biến, phép toán, biểu thức và lệnh gán trong Java

1. Hằng

Hằng là giá trị không đổi trong quá trình chạy chương trình. Có các loại hằng sau:

Hằng văn bản: Được bao bởi các cặp dấu nháy kép, ví dụ: "Chào bạn", "Tin học" v.v..

Hằng ký tự: Cũng là văn bản nhưng chỉ chứa một ký tự, được bao bởi dấu nháy đơn, ví dụ: 'a', 'A', 'x' v.v..

Hằng nguyên: Viết bình thường như trong toán học, ví dụ: 12, -3, 234 v.v..

Hằng thực tĩnh: Dùng dấu chấm thập phân, ví dụ: 12.4, 56.09, -12.0 v.v..

Hằng thực động: Dùng ký tự e hoặc E để viết, ví dụ: 1.235e1=12.345,
4.3456E-3=0.0043456, v.v..hoặc cũng có thể dùng f hoặc F, d hoặc D đặt phía sau số, ví dụ: 2.34f, -45.234d, v.v..

Hằng Logic: Có hai giá trị: đúng (true) và sai (false)

Các hằng đặt biệt (bảng sau)

Ký tự đặc biệt	**ý nghĩa**
\n	ký tự xuống dòng
\t	ký tự canh cột
\b	ký tự xoá trái
\r	ký tự về đầu dòng
\f	ký tự qua trang
\\	ký tự \

\’	ký tự nháy đơn
\”	ký tự nháy kép
\ddd	ký tự ứng với mã ASCII hệ 8
\xdd	ký tự ứng với mã ASCII hệ 16
\udddd	ký tự ứng với mã unicode

2. Biến

Biến là một vùng được định danh trong bộ nhớ để chứa dữ liệu, trước khi dùng ta phải đặt tên biến và khai báo kiểu cho nó. T*ên biến là một xâu chữ cái hoặc chữ cái và chữ số nhưng chữ cái (hoặc dấu gạch dưới hoặc dấu $) đứng đầu, không được dùng dấu cách, khi đặt ta nên tránh một số ký tự như: %, *, @ ...vốn được dùng trong Java với ý nghĩa riêng.*

Trong Java tên biến, tên hằng được phân biệt qua chữ hoa và chữ thường, ví dụ: X12 và x12 là hai biến khác nhau.

int i, j;

Biến i và j là nguyên.

float x,y,z;

Biến x,y,z là thực.

double x1,x2;

Biến x1,x2 là thực dài.

char c;

Biến c có kiểu ký tự.

String s=”

v.v..

Khi khai báo biến bạn có thể gán luôn giá trị nếu thấy cần thiết, ví dụ:

int i=12;

float x=1f, y=4d;

3. Phép toán

a. Phép toán số học

+ cộng, ví dụ: 1+2 cho kết quả 3

- trừ, ví dụ: 3-5 cho kết quả -2

* nhân, ví dụ: 2*5 cho kết quả 10

/ chia, ví dụ: 5/2 cho kết quả 2.5

% lấy phần dư, ví dụ 5 % 2 cho kết quả 1

b. Phép toán logic:

== bằng

!= không bằng

< nhỏ hơn

> lớn hơn
<= nhỏ hơn hoăc bằng
>= lớn hơn hoặc bằng
& hoặc && phép and
| hoặc || phép or
! phép not
Ví dụ về sự khác nhau giữa & và &&:
(1>3) & (4<6) máy sẽ tính từng biểu thức một, cụ thể:(1>3) cho false, (4<6) cho true. Như vậy false & true cho kết quả false.
còn nếu dùng &&:
(1>3) && (4<6) máy sẽ tính (1>3) là false và kết luận luôn là false
| và || cũng tương tự.

4. Biểu thức

Các toán hạng nối với nhau bởi phép toán gọi là một biểu thức, ví dụ:
a+b
(x+y)*(x-y)
(t<h) & (u<k)

5. Lệnh gán

Ví dụ: x=2+5;
x được gán giá trị 7, tổng quát:
<biến>=<Biểu thức>
<Biểu thức> có thể là một hằng, ví dụ: a=12.6f
Chú ý: Với Java ta có thể viết gọn trong một số trường hợp như sau:
x=x+y; viết gọn x+=y
x=x-y; viét gọn x-=y
*x=x*y; viết gọn x*=y*
x=x/y; viét gọn x/=y
x=x%y; viết gọn x%=y
Cách viết y=x++ có nghĩa x được gán cho y sau đó x được tăng thêm 1
Cách viết y=++x có nghĩa x được tăng thêm 1 rồi mới gán cho y
x=x+1có thể viết x++ (phép toán ++ gọi là phép tăng)
x=x-1có thể viết x-- (phép toán -- gọi là phép giảm)
Có thể bạn cho rằng việc đưa ra phép toán tăng hoặc giảm và cách viết lệnh gán rút gọn chỉ làm phức tạp vấn đề và không cần thiết, tất nhiên bạn có thể dùng hoặc không, nhưng nếu bạn không quen cách viết đó, bạn sẽ gặp trở ngại khi đọc một đoạn mã có cách viết đó, thôi thì trước lạ sau quen, bạn cố nhớ cho xong.

6. Ví dụ về lập trình

Hãy tạo lớp có tên TinhToan bằng cách khởi động J++6.0, chọn File/New Project , v.v.. như đã nói ở phần đầu. Sau đó gõ đoạn mã sau:

Ví dụ 1:

```
public class TinhToan
   {
        public static void main(String[] args)
              {
                int s,a,b,c;
                a=1;
                b=2;
                c=3;
                s=a+b+c;
                System.out.println("S="+s);
              }
   }
```

Ví dụ 2:

```
public class GanRutGon
   {
        public static void main(String[] args)
              {
                int i;
                i=1;
                i+=1;
                System.out.println("I="+i);
              }
   }
```

2.3 Câu lệnh if và switch

Câu lệnh if có hai dạng.

Dạng 1:

```
if (<điều kiện>)
  {
        <các câu lệnh>
  }
```

Ví dụ:

```
if (i<=12)
  {
        System.out.println("i="+i);
        System.out.println(“Dung la i nho hon 12”);
```

```
   }
```
Dạng 2:
```
if (<điều kiện>)
   {
        <Các lệnh 1>
   }
else
   {
       <Các lệnh 2>
   }
```
Ví dụ:
```
public class ViDuElse
   {
        public static void main(String[] args)
              {
                int i;
                i=1;
                if (i<1)
                   {
                        System.out.println("I="+(i+3));
                        System.out.println("I="+i);
                   }
                else
                   {
                        System.out.println("I="+(i+5));
                        System.out.println("I="+(i+3));
                    }

              }
     }
```
Chú ý: nếu sau if hoặc sau else chỉ có một lệnh thì không cần dấu { và }.
Ví dụ:
```
public class ViDuElse
   {
        public static void main(String[] args)
              {
                int i;
                i=1;
                if (i<1)
```

```
                    System.out.println("I="+(i+3));
              else
              System.out.println("I="+(i+3));
            }
     }
```

sau **else** cũng có thể có lệnh **if**, ví dụ:

```
public class vidu
   {
              public static void main(String[] args)
            {
              int i;
              i=1;
              if (i<1)
                  {
                        System.out.println("I="+(i+3));
                        System.out.println("Đúng là i<1”);
                  }
              else
                     if (i>1)
                       {
                            System.out.println("I="+(i+5));
                            System.out.println("Đúng là i>1”);
                        }
             else
                    System.out.println("I="+(i+3));
          }
   }
```

Lệnh switch có dạng:

```
switch (test)
   {
        case 1:
          <các câu lệnh 1>;
              break;
        case 2:
          <các câu lệnh 2>;
              break;
        .....
        case n:
```

```
        <các câu lệnh n>;
              break;
        default:
      <các câu lệnh n+1>;
```

Nếu không có lệnh **break** java sẽ thông thoát khỏi lệnh **switch.**
Nếu test=1 thì thực hiện các lệnh sau case 1:
Nếu test=2 thì thực hiện các lệnh sau case 2:
....
Nếu test=n thì thực hiện các lệnh sau case n:
nếu không thì thực hiện lệnh sau default:
Ví dụ 1:

```
public class LenhChon
  {
      public static void main(String[] args)
        {
            int i;
            i=11;
            switch (i)
              {
              case 1:
                 System.out.println("I="+(i+9));
                   break;
              case 2:
                   System.out.println("I="+(i+5));
                   break;
              default:
                   System.out.println("I="+(i+3));
               }
        }
                                                                    }
```

Khi chạy máy sẽ cho kết quả I=14.
Bạn cũng có thể không dùng default: nhưng với chương trình trên khi chạy máy sẽ không in kết quả nào cả.

2.4 Vòng lặp for, while và do...while

Java có 3 vòng lặp: for, while và do...while, ta lần lượt xét 3 loại lặp đó.

1. Lặp for

Giả sử để tính tổng s=1+2+3+...+1000, ta lập trình như sau:

```
public class TinhTong
```

```
{
    public static void main(String[] args)
    {
        int s,i;
        s=0;
        for (i=1;i<=1000;i++)
            s=s+i;
        System.out.println("s="+s);
    }
}
```

Tổng quát lệnh for có dạng:

for (<Biểu thức gán giá trị đầu>;<Biểu thức điều kiện>; <Biểu thức tăng>)

Ví dụ: Tính p=14!=1*2*3*..*14 (giai thừa)

```
public class GiaiThua
{
    public static void main(String[] args)
    {
        int i;
        float p;
        p=1;
        for (i=1; i<=14; i=i+1)
            p=p*i;
        System.out.println("p="+p);
    }
}
```

Lệnh for có thể thiếu một biểu thức như sau:

```
public class TinhToan
{
    public static void main(String[] args)
    {
        int i;
        float p;
        p=1;i=1;
        for (; i<=14;i=i+1 )
            p=p*i;
        System.out.println("p="+p);
    }
}
```

Có thể có nhiều vòng for lồng nhau, ví dụ:

```
for (i=1; i<=10; i++)
    for (j=1; j<=10; j++)
```

Hoạt động của nó như sau:
Với i=1 thì j=1,2,3,4,..,10
Với i=2 thì j=1,2,3,4,..,10
...
Với i=10 thì j=1,2,3,4,..,10
Như vậy có 100 lần lặp.
Các bạn đã quen bài toán:
Trăm trâu trăm cỏ
Trâu đứng ăn năm
Trâu nằm ăn 3
Lụ khụ trâu già
Ba con một bó
Hỏi mỗi loại trâu có bao nhiêu con?.
Nếu ta gọi trâu đứng, nằm và già là *dung, nam* và *gia* thì ta có hệ phương trình vô định sau:
dung+nam+gia=100
5*dung+3*nam+1/3*gia=100
Vì là nghiệm nguyên dương nên ta quy đồng để tránh phép chia cho 3:
dung+nam+gia=100
15*dung+9*nam+*gia=300
Vì trâu đứng, nằm và già chỉ có thể nằm trong khoảng từ 1 đến 100, nên ta "mò" dần bằng 3 lệnh for lồng nhau. Sau đây là chương trình:

```
public class TinhToan
   {
       public static void main(String[] args)
         {
               int trau, co, dung, nam, gia;
                for (dung=1;dung<=100;dung++)
                  for (nam=1;nam<=100;nam++)
                    for (gia=1;gia<=100;gia++)
                       {
                       trau=dung+nam+gia;
                       co=15*dung+9*nam+gia;
                       if ((co==300) & (trau==100))
                             {
                                System.out.println("Trau dung: "+dung);
```

```
                        System.out.println("Trau nam: "+nam);
                        System.out.println("Trau gia: "+gia);
                        System.out.println("--------------------");
                    }

                }
            }
    }
```

Khi chạy sẽ cho ta kết quả:

```
Trau dung: 4
Trau nam: 18
Trau gia: 78
--------------
Trau dung: 8
Trau nam: 11
Trau gia: 81
--------------
Trau dung: 12
Trau nam: 4
Trau gia: 84
```

2. Lặp while

Có dạng:

```
while (<điều kiện>)
    {
        <các lệnh>;
    }
```

Chừng nào <điều kiện> còn đúng thì còn lặp.

Ví dụ:

```
public class TinhTong
   {
        public static void main(String[] args)
          {
                int s,i;
                s=0; i=1;
                while (i<=1000)
                  {
                     s=s+i;
                     i=i+1;
                  }
            System.out.println("s="+s);
```

```
        }
    }
```

3. Lặp do...while

Lệnh do...while có dạng như sau:

```
do
  {
     <các lệnh>
   }  while <điều kiện>;
```

Chừng nào <điều kiện> còn đúng thì còn lặp. Ví dụ sau:

```
public class TinhToan
    {
          public static void main(String[] args)
           {
                  int s,i;
                  s=0; i=1;
                 do
                   {
                       s=s+i;
                       i=i+1;
                    } while (i<=1000);
              System.out.println("s="+s);
           }
    }
```

2.5 Mảng

Trong các ngôn ngữ lập trình như Pascal, C, C++...), mảng được tạo ra với kích thước cố định và lập tức chiếm cứ một vùng bộ nhớ đủ để chứa các phần tử mảng. Với Java có hơi khác, trước tiên bạn định nghĩa một biến làm tên gọi mảng và kiểu của các phần tử, nhưng không cần xác định ngay kích thước của mảng. Kích thước mảng tuỳ thuộc vào thực tế và sẽ được bàn giao sau. Ví dụ:

int a[]; định nghĩa mảng **a** dùng để quản lý các số nguyên.

String chr[]; định nghĩa mảng **chr** dùng để quản lý văn bản .

bạn có thể viết cách khác như sau:

int[] a;

String[] chr;

Hai cách viết trên hoàn toàn tương đương.

2.5.1 Khởi tạo mảng

Khởi tạo mảng là tạo ra các đối tượng (phần tử) và trao cho mảng quản lý. các đối tượng ấy chính là số phần tử của mảng, hay cũng gọi là kích thước mảng. Ví dụ khởi tạo mảng trực tiếp:

a={1, 2, 4, 6}; mảng nguyên a có 4 phần tử nguyên.

chr={"tin", "học", "văn", "phòng"}; mảng chr có 4 phần tử, mỗi phần tử là một xâu.

Chúng ta cũng có thể khởi tạo mảng theo cách sau:

a=**new int**[10]; mảng nguyên a có 10 phần tử, giá trị ban đầu là 0.

chr=**new String**[8]; mảng chr có 8 phần tử, giá trị ban đầu là null.

Ví dụ 1: Tính tổng mảng

```
public class TongMang
   {
        public static void main(String[] args)
         {
                float [] a;
                float s;
                int i;
                 a=new float[5];
                 a[0]=12.6f;
                 a[1]=10.7f;
                 a[2]=15.9f;
                 a[3]=1.7f;
                 a[4]=18.9f;
                 for (i=0; i<5; i++)

                 s=s+a[i];
              System.out.println("s="+s);
         }
   }
```

Ví dụ 2: Mảng xâu

```
public class Mangxau
   {
        public static void main(String[] args)
         {
                String [] chr;
                int i;
                 chr=new String[5];
```

```
                chr[0]="Toi di tim cai nua cua toi";
                chr[1]="Ma tim mai den bay gio cha thay";
                chr[2]="Tinh yeu cua toi oi anh la ai vay";
                for (i=0; i<5; i++)
                  System.out.println(chr[i]);
        }
    }
```

Sau khi chạy ta có kết quả:

```
Toi di tim cai nua cua toi
Ma tim mai den bay gio cha thay
Tinh yeu cua toi oi anh la ai vay
null
null
```

2.5.2 Mảng nhiều chiều

Mảng nhiều chiều trong Java được xây dựng trên mảng một chiều.

Nếu xem mỗi dòng của mảng hai chiều là một mảng, vậy có bao nhiêu dòng của mảng hai chiều thì có bấy nhiêu mảng một chiều. Ví dụ ta muốn khai báo một mảng hai chiều có 3 dòng và 2 cột:

```
int[][] a;
a=new int[3][];
a[0]=new int[2];
a[1]=new int[2];
a[2]=new int[2];
```

Sau đây là ví dụ đơn giản tính tổng mảng nguyên 2 chiều:

```
public class TongMangHaiChieu
   {
      public static void main(String[] args)
           {
              int a[][]; //Khai báo mảng 2 chiều
              int s, i, j;
               a=new int[3][]; //Khai báo 3 dòng, số phần tử dòng thì chưa
               a[0]=new int[2]; //Dòng thứ nhất có 2 phần tử
               a[1]=new int[2]; //Dòng thứ hai có 2 phần tử
               a[2]=new int[2]; //Dòng thứ ba có 2 phần tử
                   // Khởi tạo giá trị cho mảng
                   a[0][0]=1;
                   a[0][1]=2;
                   a[1][0]=3;
                   a[1][1]=4;
```

```
                a[2][0]=5;
                a[2][1]=6;
                s=0;
            for (i=0; i<3; i++)
                for (j=0;j<2;j++)
                s=s+a[i][j];
                System.out.println("s="+s);
        }
    }
```

2.6 Method

Một Method là một dãy các câu lệnh thực hiện một tác vụ nào đó. Method giống như hàm (Function) trong C/C++, nhưng người ta không dùng từ Function vì nó "không có vẻ hướng đối tượng" lắm. Thật ra không việc gì phải "kiêng kỵ", bạn có thể dùng Funtion cũng được. Tuy nhiên trong giáo trình này ta dùng thuật ngữ Method, bạn có thể hiểu nó như một "thủ tục" hoặc "hàm" tuỳ ý. Một số tài liệu khác có dùng thuật ngữ "phương thức" cũng không sao, miễn là bạn hiểu được chức năng của nó, còn thuật ngữ nhiều khi cũng chỉ là "tương đối".

Qua các ví dụ về lập trình đã trình bày ở trên, bạn thấy một chương trình Java thực chất là một class, trong class có chứa Method có tên là **main**. Ngoài **main** ra tất nhiên còn có những Method khác, tên của nó tuỳ bạn đặt. Phần này chúng ta sẽ đề cập đến vấn đề đó.

2.6.1 Method không đối số

Chương trình sau có tên lớp là **HaiMethod** (Hai Method), ý muốn nói class này có chứa 2 Method: **main** và **TenDiaChi** (Tên Địa Chỉ). Method main chỉ có một nhiệm vụ là gọi Method TenDiaChi() để in tên, địa chỉ và điện thoại. Sau đây là chương trình:

Ví dụ 1:

```
public class HaiMethod
    {
        public static void main(String[] args)
            {
               TenDiaChi();
             }
        public static void TenDiaChi()
             {
                System.out.println("Truong DHQL&KD ha noi");
                System.out.println("1B Cam hoi- Lo duc Ha noi");
```

```
                System.out.println("Tel 9712932");
            }
    }
```

Như bạn đã thấy trong chương trình, một Method dù là main hay không phải main đều có chung một cấu trúc là:

Phần khai báo
Dấu {
Các lệnh
Dấu }

Ở trên phần khai báo: **public static void** TenDiaChi()

public: Toàn cục

static: Tĩnh

void: Method không nhận giá trị

TenDiaChi: Tên Method

Thân Method:

```
            {
                System.out.println("Truong DHQL&KD ha noi");
                System.out.println("1B Cam hoi- Lo duc Ha noi");
                System.out.println("Tel 9712932");
            }
```

Ví dụ 2:

Để làm quen với cấu trúc một class, ta thiết kế một chương trình đơn giản có hai Method để tính tổng và tích, Method main gọi hai Method này.

```
public class ViDu2
  {
      public static void main(String[] args)
            {
              TinhTong();
              TinhTich();
             }
        public static void TinhTong()
             {
              int s,i;
              s=0;
              for (i=1;i<=100;i++)
              s=s+i;
              System.out.println("tong="+s);
            }
      public static void TinhTich()
```

```
                    {
                       int p,i;
                       p=1;
                       for (i=1;i<=10;i++)
                       p =p*i;
                      System.out.println("tich="+p);
                    }

   }
```

Kết quả mà Method TinhTong() hoặc Method TinhTich() tính được, có thể chuyển cho biến toàn cục là **tong** và **tich** để hiển thị trong method **main**, hãy quan sát chương trình sau:

```
public class TinhToan
  {
        static int tong, tich;
       public static void main(String[] args)
              {
                  TinhTong();
                  TinhTich();
                  System.out.println("tong="+tong);
                  System.out.println("tich="+tich);
               }
         public static void TinhTong()
              {
                 int s,i;
                  s=0;
                  for (i=1;i<=100;i++)
                  s=s+i;
                 tong=s;
              }
         public static void TinhTich()
              {
                    int p,i;
                    p=1;
                    for (i=1;i<=10;i++)
                    p=p*i;
                    tich=p;
            }
```

```
    }
```

2.6.2 Method có một đối số

Khi Method có một đối số, có nghĩa Method có thể nhận giá trị hoặc không và nếu nhận giá trị ta bỏ từ void và thêm vào kiểu của Method, đồng thời khai báo luôn kiểu cho đối số đó. Giả sử ta tính tổng s=1+2+3+...+h (h chưa xác định, sẽ chuyển giá trị khi gọi Method). Hãy quan sát chương trình sau:

```
public class TinhTong
  {
       public static void main(String[] args)
            {
                 int k;
                 k=1000;
                System.out.println("Tong="+TinhTong(k));
              }
       public static int TinhTong(int h)
            {
                 int s, i;
                 s=0;
                 for (i=1; i<=h; i++)
                 s=s+i;
                return s; //trả giá trị s về cho method
            }
  }
```

Dòng khai báo Method: **public static int** TinhTong(**int** h) cho bạn biết kiểu của Method là **int**, kiểu của đối số h cũng **int**.

Trường hợp bạn muốn nhận giá trị do Method tạo ra để còn tiếp tục tham gia xử lý tiếp thì vẫn bình thường như khi hiển thị, ví dụ:

```
public class TinhToan
  {
       public static void main(String[] args)
            {
                 int k,n;
                 k=3;
                      n=TinhTong(k)*10; //Lời gọi Method để xử lý tiếp
                      System.out.println("Tong="+n);
              }
       public static int TinhTong(int h)
            {
```

```
            int s,i;
                s=0;
                for (i=1; i<=h; i++)
                s=s+i;
                return s;
        }
}
```

Còn nếu bạn không muốn nhận giá trị thì:

```
public class TinhToan
  {
      public static void main(String[] args)
          {
                int k,n;
                k=30;
              TinhTong(k);
          }
      public static void TinhTong(int h)
          {
            int s,i;
                s=0;
                for (i=1; i<=h; i++)
                s=s+i;
                System.out.println("Tong="+s);
          }
}
```

Tóm lại, tuỳ ý định của bạn mà việc thiết kế các Method có thể nhận giá trị qua tên Method (dùng **return** <biểu thức>), qua biến toàn cục hoặc không nhận giá trị v.v..

2.6.3 Method có nhiều đối số

Dù nhiều đối số cũng vậy, bạn phải khai báo từng đối số một, và tuỳ theo có nhận giá trị hay không mà khai báo **void** hay kiểu của Method. Ví dụ:

```
public class TinhToan
  {
      public static void main(String[] args)
              {
                double a, b, c;
                    a=1.5;
                    b=2.5;
                    c=3.5;
```

```
                System.out.println("Tong="+TinhTong(a,b,c));
            }
    public static double TinhTong(double x, double y, double z)
        {
          double s;
              s=x+y+z;
            return  s;
          }
}
```

2.7 Đặc trưng hướng đối tượng của Java

Hướng đối tượng (**object oriented**) là một phương pháp luận lập trình, giúp ta cách thức xây dựng phần mềm, vừa gần gũi với thực tế lại vừa có tính khái quát cao. Hướng đối tượng được xem là một thành tựu vĩ đại của công nghệ lập trình. Java cũng như các ngôn ngữ khác cũng không thể nằm ngoài khuynh hướng đó.

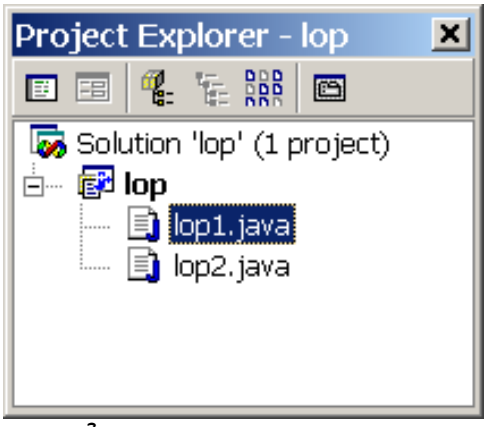
Phần này sẽ trình bày khái quát bản chất hướng đối tượng: lớp (class) và đối tượng (object); hai bộ phận cơ bản của class là dữ liệu và phương thức (method), từ đó dẫn đến khái niệm kế thừa (inheritance) và đa hình (polymorphism).

2.7.1 Hướng đối tượng là gì?

Để hiểu khái niệm này ta hãy đưa ra một ví dụ: Chắc bạn đã biết trò chơi ghép hình của trẻ em (và cũng có thể là của chúng ta nữa). Bộ đồ chơi này bao gồm nhiều khối nhựa nhỏ có lỗ, chốt v.v.. sao cho chúng có thể khớp nối được với nhau. Chỉ với các khối nhựa nhỏ đó mà "bọn trẻ" có thể xếp thành nhiều hình thù như ngôi nhà, ô tô, xe tăng, đại bác v.v..vậy là để có được một sản phẩm (ngôi nhà chẳng hạn) bạn phải có hai thứ: "khối nhựa nhỏ"(dữ liệu) và "cung cách thao tác" (phương thức-Method). Hai thứ đó thuộc về lớp (class) đồ chơi ghép hình. Vậy là bạn hình dung ra từ "**lớp**" (class)!.Hay nói một cách khác là bạn nghĩ đến những đối tượng (Object), nhờ nó bạn tạo ra được sản phẩm theo ý mình. Rõ ràng đối tượng càng phong phú và chuẩn mực bao nhiêu thì sự ra đời các sản phẩm càng nhanh và chính xác bấy nhiêu. Trong lập trình cũng vậy, để có được một sản phẩm bạn phải "kết dính" nhiều đối tượng vào đó. Java có những object như vậy để bạn tạo ra chương trình một cách nhanh chóng và chính xác. Hiểu một cách "nôm na" là cung cách "lắp ghép" các đối tượng trong Java để có một phần mềm được gọi là lập trình hướng đối tượng (OOP=Object Oriented Programming). Vậy là để lập trình trong Java bạn phải nghĩ ngay đến lớp, trong lớp có các đối tượng v.v..

2.7.2 Xây dựng lớp (class)

Chúng ta thiết kế một Solution có một Project trong Project này có hai tập tin lớp (lop1.java và lop2.java) như hình sau:



Mã của class lop1.java:

```
class lop1
      {
        int x, y, s;
// Method tong()
            void tong()
                  {
                    s=x+y;
                  }
      }
```

Bạn thấy class trên có hai phần:

Phần dữ liệu:

```
int x, y, s;
```

Phần Method:

```
void tong()
      {
        s=x+y;
      }
```

Nếu bạn cố chạy, máy sẽ báo là không thấy main trong lớp này. Bởi vì chúng ta chỉ mới xây dựng lớp chứ chưa có lớp cụ thể, mà bạn thấy đấy cũng chưa có dữ liệu và tất nhiên phải có Method main.

Hãy viết mã cho lớp 2 như sau:

```
class lop2
        {
         public static void main(String[] args)
          {
```

```
            lop1 m=new lop1(); //tạo lớp m cụ thể từ lop1
            m.x=1;
            m.y=2;
            m.tong();
            System.out.println("s="+m.s);
    }
  }
```

Lop2 này chỉ chứa Method main. Bạn xem dòng lệnh:
lop1 **m=new** lop1(); Cách viết này có nghĩa là tạo ra đối tượng **m** từ lop1.
Bây giờ bạn chạy chương trình và kết quả sẽ là s=3.
Nếu bạn khai báo thêm một đối tượng **n** nữa như sau:

```
class lop2
      {
       public static void main(String[] args)
        {
            lop1 m=new lop1();
            lop1 n=new lop1();
            m.x=1;
            m.y=2;
            m.tong();
            System.out.println("s="+m.s);
            n.x=111;
            n.y=222;
            n.tong();
            System.out.println("s="+n.s);
      }
  }
```

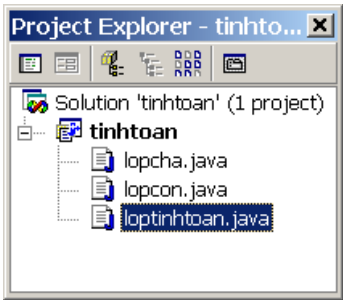
Thì khi chạy sẽ có kết quả:
s=3
s=333

2.7.3 Kế thừa

Trong thực tế người ta thường xây dựng một lớp từ một lớp đã có. Dùng lớp có sẵn để xây dựng một lớp mới là một trong những nội dung quan trọng nhất của lập trình hướng đối tượng. Nhờ khả năng này mà chúng ta xây dựng nên những lớp ngày càng phức tạp giống như sự di truyền trong thế giới sinh vật.

Để hiểu rõ tính kế thừa, tốt nhất vẫn nên thông qua một ví dụ đơn giản như sau: Một Solution có chứa một Project, trong Project này ta xây dựng 3

lớp có tên **lopcha.java, lopcon.java và loptinhtoan.java**. Lớp con được kế thừa từ lớp cha và **loptinhtoan.java** dùng để hiển thị tên cha, tuổi cha, tên con, tuổi con và tuổi trung bình của cha và con.



Mã của lopcha.java:

```
class lopcha
     {
          String TenCha;
          int TuoiCha;
             void ShowCha()
                 {
                    System.out.println("Ten cha: "+TenCha);
                    System.out.println("Tuoi cha: "+TuoiCha);
                 }
     }
```

Mã của lopcon.java:

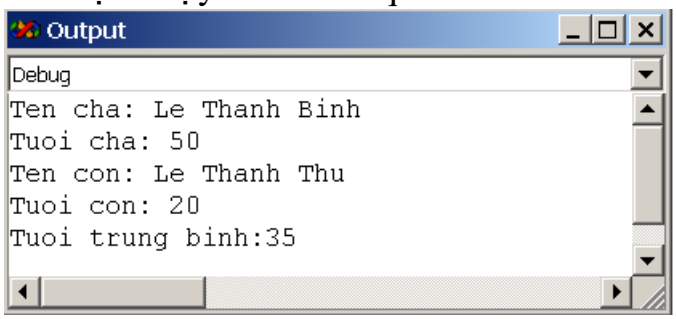
```
class lopcon extends lopcha //lớp con được kế thừa từ lớp cha
    {
         String TenCon;
          int TuoiCon;
         void ShowCon()
            {
                System.out.println("Ten con: "+TenCon);
                System.out.println("Tuoi con: "+TuoiCon);
            }
     }
```

Mã của loptinhtoan.java:

```
public class loptinhtoan
  {
    public static void main(String[] args)
        {
           lopcon Q=new lopcon();//Q là đối tượng lớp con
```

```
            Q.TenCha="Le Thanh Binh";
            Q.TuoiCha=50;
            Q.TenCon="Le Thanh Thu";
            Q.TuoiCon=20;
            Q.ShowCha();
            Q.ShowCon();
          System.out.println("Tuoi trung binh:"+((Q.TuoiCha+Q.TuoiCon)/2));
      }
   }
```

Khi bạn chạy sẽ có kết quả:



Trong lớp con ngoài dữ liệu và Method mới còn chứa toàn bộ dữ liệu và Method của lớp cha nhưng bị che khuất.

2.7.4 Đa hình

Điều gì sẽ xẩy ra khi trong lớp con có một Method trùng tên với lớp cha. Giả sử hai Method ShowCha() và ShowCon() cùng có tên Show(), tuy nội dung khác nhau. Chúng ta viết lại chương trình trên:

```
class lopcha
      {
            String TenCha;
            int TuoiCha;
               void Show()
                   {
                      System.out.println("Ten cha: "+TenCha);
                      System.out.println("Tuoi cha: "+TuoiCha);
                    }
      }
```

Mã của lopcon.java:

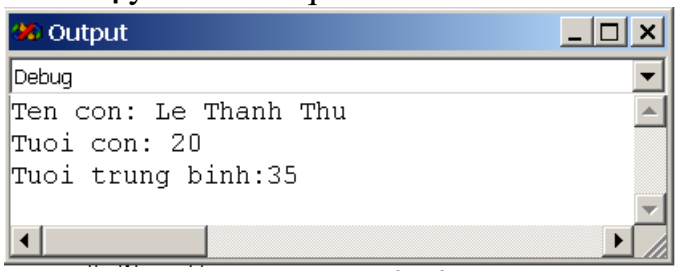
```
class lopcon extends lopcha //lớp con được kế thừa từ lớp cha
    {
          String TenCon;
           int TuoiCon;
```

```
        void Show() //cũng có tên Show()
          {
             System.out.println("Ten con: "+TenCon);
             System.out.println("Tuoi con: "+TuoiCon);
          }
    }
```

Mã của loptinhtoan.java:

```
public class loptinhtoan
  {
   public static void main(String[] args)
       {
          lopcon Q=new lopcon();//Q là đối tượng lớp con
          Q.TenCha="Le Thanh Binh";
          Q.TuoiCha=50;
          Q.TenCon="Le Thanh Thu";
          Q.TuoiCon=20;
          Q.Show();
       System.out.println("Tuoi trung binh:"+((Q.TuoiCha+Q.TuoiCon)/2));
    }
  }
```

Khi chạy ta có kết quả:



Rõ ràng chương trình chỉ sử dụng Show() ở lớp con. Vậy muốn sử dụng Show() của lớp cha thì làm thế nào?. Tất nhiên phải khai báo thêm một đối tượng nữa, cụ thể:

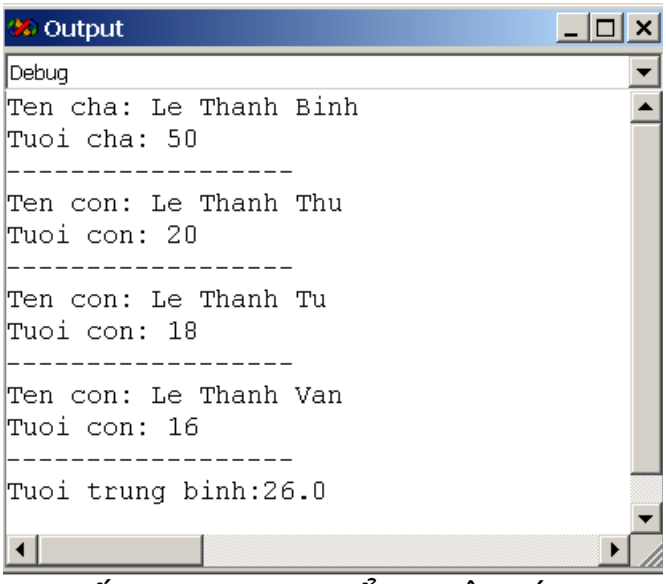
```
public class loptinhtoan
  {
   public static void main(String[] args)
       {
          lopcon Q=new lopcon();//Q là đối tượng lớp con
          lopcon P=new lopcha();//P là đối tượng lớp cha
          Q.TenCha="Le Thanh Binh";
          Q.TuoiCha=50;
          Q.TenCon="Le Thanh Thu";
          Q.TuoiCon=20;
```

```
            P.Show();
            Q.Show();
        System.out.println("Tuoi trung binh:"+((Q.TuoiCha+Q.TuoiCon)/2));
    }
  }
```

Khi chạy ta thấy ổn, nghĩa là chương trình thực hiện Show() ở lớp cha và cả ở lớp con. Một vấn đề khác được đặt ra là: Giả sử gia đình nào đó có từ hai con trở lên thì có dùng được chương trình này không ? tất nhiên là có, bạn chỉ cần khái báo thêm đối tượng và đưa các đối tượng vào mảng, ta viết lại lớp loptinhtoan như sau:

```
public class loptinhtoan
  {
                public static void main(String[] args)
                  {
                        lopcha Q=new lopcha();
                        lopcon P=new lopcon();
                        lopcon R=new lopcon();
                        lopcon S=new lopcon();
                        Q.TenCha="Le Thanh Binh";
                        Q.TuoiCha=50;
                        P.TenCon="Le Thanh Thu";
                        P.TuoiCon=20;
                        R.TenCon="Le Thanh Tu";
                        R.TuoiCon=18;
                        S.TenCon="Le Thanh Van";
                        S.TuoiCon=16;
                        lopcha[] mang={Q, P, R,S}; //Mảng các đối tượng
                        float tb;
                        int i;
                        for (i=0;i<4; i++)
                              {
                                mang[i].Show();
                                System.out.println("------------------");
                              }
                    tb=(Q.TuoiCha+P.TuoiCon+R.TuoiCon+S.TuoiCon)/4;
                    System.out.println("Tuoi trung binh:"+tb);
                }
  }
```

Khi chạy ta có kết quả:



Ta thấy mang[] có kiểu thuộc lớp cha, nhưng cũng chấp nhận các đối tượng thuộc lớp con và với cùng một câu lệnh thực hiện Show(), ta thu được các câu trả lời khác nhau. Nhưng rõ ràng với Show() đời con xử lý khác đời cha. Cơ chế linh hoạt này có được là nhờ gọi Method thông qua đối tượng vào lúc chạy chương trình. Người ta gọi quá trình đó là tính *đa hình* (polymorphism). Bạn cứ tưởng tượng nếu "dòng họ" cứ phát triển mãi, có bao nhiêu thế hệ khác nhau thì có bấy nhiêu tình huống có thể khác nhau nhưng chỉ thông qua một câu lệnh: mang[i].Show().

Như vậy ngoài khái niệm *lớp*, *đối tượng* và *kế thừa*, tính *đa hình* là một nét nữa góp phần làm cho tính hướng đối tượng thêm phong phú.

2.8 Nạp chồng Method

Java cho phép ta định nghĩa trong cùng một lớp nhiều Method trùng tên với điều kiện các Method ấy có danh sách các đối số hoặc kiểu các đối số khác nhau. Khả năng này gọi là nạp chồng Method (Method overloading). Ví dụ sau có hai method tong(), nhưng khác nhau bởi số lượng các đối số của method:

```
public class Class1
{
        public static void main (String[] args)
        {
                int x,y,z;
                x=1;
                y=2;
                z=3;
                tong(x,y);
        }
```

```
public static void tong(int i, int j)
        {
          System.out.println("s="+(i+j));
        }
public static void tong(int i, int j, int k)
        {
           System.out.println("s="+(i+j+k));
        }
}
```

Sau khi chạy kết quả: s=3

2.9 Tình huống mơ hồ

Khi nạp chồng hàm, bạn có thể gặp những tình huống mà bộ biên dịch không thể xác định Method nào được sử dụng, ví dụ:

Trường hợp 1: Đúng ý chúng ta.

```
public class chaoban
{
        public static void main (String[] args)
        {
                double a;
                int i;
                a=6.0;
                i=8;
                KetQua(a);
                KetQua(i);
        }

           public static void KetQua(double a)
                   {
                        System.out.println("ket qua ="+(a+1));
                   }

           public static void KetQua(int i)
                   {
                       System.out.println("ket qua="+(i+2));
                   }

}
```

Khi chạy sẽ cho ta kết quả:

ket qua=7.0

ket qua=10

Trường hợp 2:

```
public class chaoban
{
        public static void main (String[] args)
        {
                int i;
                i=8;
                KetQua(i);

        }

public static void KetQua(double a)
  {
    System.out.println("ket qua="+(a+1));
  }

}
```

Biến nguyên i được chuyển sang double cho phù hợp với khai báo của hàm KetQua(). Kết quả sau khi chạy chương trình sẽ là:

ket qua=9.0 (mặc dù ta muốn là 9 - số nguyên)

Trường hợp 3:

```
public class chaoban
{
        public static void main (String[] args)
        {
                double a;
                a=6.0;
                KetQua(a);

        }

public static void KetQua(int i)
  {
    System.out.println("ket qua="+(i+2));
  }

}
```

Khi chạy sẽ báo lỗi, vì không thể biến có kiểu double chuyển sang int được. Ngoài ra giả sử nếu gặp hai Method sau:
void Tong(int i) và
int Tong(int i)
Máy sẽ không biết gọi Method nào. Tóm lại bạn hãy cảnh giác với việc nạp chồng method để tránh những "tình huống mơ hồ" trên.

Bài tập:
1. Class là gì cho ví dụ
2. Kế thừa là gì cho ví dụ
3. Đa hình là gì cho ví dụ
4. Thế nào là nạp chồng hàm cho ví dụ
5. Hãy cho một ví dụ về tình huống mơ hồ
6. Cho mảng A có 100 phần tử nguyên, hãy gán các phần tử cho mảng và sắp xếp giảm dần
7. Cũng như câu hỏi 1 nhưng các phần tử mảng là văn bản
8. Từ việc giải phương trình bậc hai, kế thừa để giải phương trình trùng phương
9. Từ tuyển sinh khối A cho Trường X, hãy xây dựng đa hình với phương *thức in giấy báo đỗ* để giải quyết cho Trường Y

CHƯƠNG 3: CÁC APPLET

Như các bạn đã biết, ngoài việc Java có thể tạo ra các trình ứng dụng đứng độc lập như C++ hoặc Visual Basic v.v.. còn có một khả năng khác là tạo ra các ứng dụng nhỏ (Applet) để nhúng vào Web, giúp cho việc tạo các trang Web động một cách dễ dàng. Như vậy một Applet được chạy từ trang Web trong khi các ứng dụng đứng độc lập được chạy từ môi trường Windows hoặc DOS. Để tạo một Applet bạn theo các bước sau:

Tạo một Project và viết mã Java với các File có phần mở rộng *.java giống như viết trình ứng dụng đứng độc lập

Biên dịch các file *.java thành Bytecode với phần mở rộng *.class

Thiết kế một tài liệu HTML và đưa mã vừa biên dịch vào một thẻ đặc biệt

Dùng IE hoặc Netscape Navigator hoặc Appletviewer để duyệt tài liệu HTML có Applet vừa tạo.

Các trang Web có chứa các Applet là công việc phổ biến của người lập trình, người sử dụng các Applet có thể sử dụng bộ duyệt Web để thực thi các giao diện với Applet đó, giống như bạn truy cập các WebSite trên mạng Internet. Thẻ đặc biệt chứa mã Applet có dạng:

<APPLET>.....</APPLET>

Ví dụ:

<APPLET CODE=”NameApplet.class” WIDTH=400 HEIGHT=300></APPLET>

Giải thích:

CODE= tên của APPLET

WIDTH= Chiều rộng của APPLET trên màn hình

HEIGHT=Chiều cao của APPLET trên màn hình

Như đã nói ở phần trên, APPLET phải là một đoạn mã JAVA đã được biên dịch thành *.class. Chiểu rộng và chiều cao được đo bằng số điểm Pixel.

3.1 Tạo các APPLET đơn giản

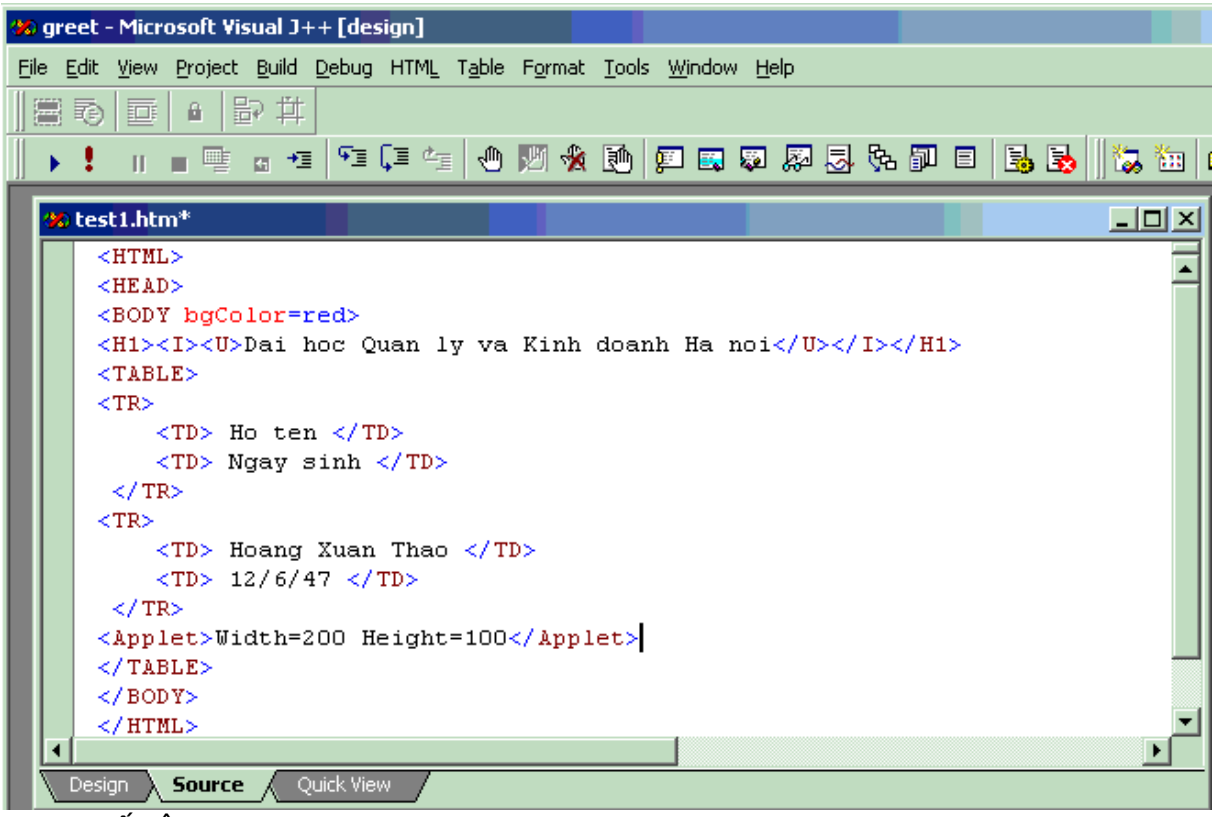
Các tài liệu HTML là các File Text, ta có thể tạo chúng trong bất kỳ hệ soạn thảo nào như NotePad, WordPad v.v..nhưng như thế thì bạn không

xem ngay được kết quả. Có một số trình ứng dụng khác dùng để soạn thảo HTML giúp bạn có thể xem ngay kết quả (ta hay gọi trình ứng dụng này là bộ chỉnh sửa HTML). Một số bộ chỉnh sửa phổ biến như Microsoft FrontPage , Adobe PageMill v.v.. có giao diện đồ hoạ cho phép bạn tạo các trang Web và biết kết quả ngay trong lúc soạn. Trong Visual Studio cũng có bộ chỉnh sửa như vậy gọi là HTML EDITER WINDOW. Cửa sổ HTML Editer chứa ba chế độ để làm việc với tài liệu HTML:

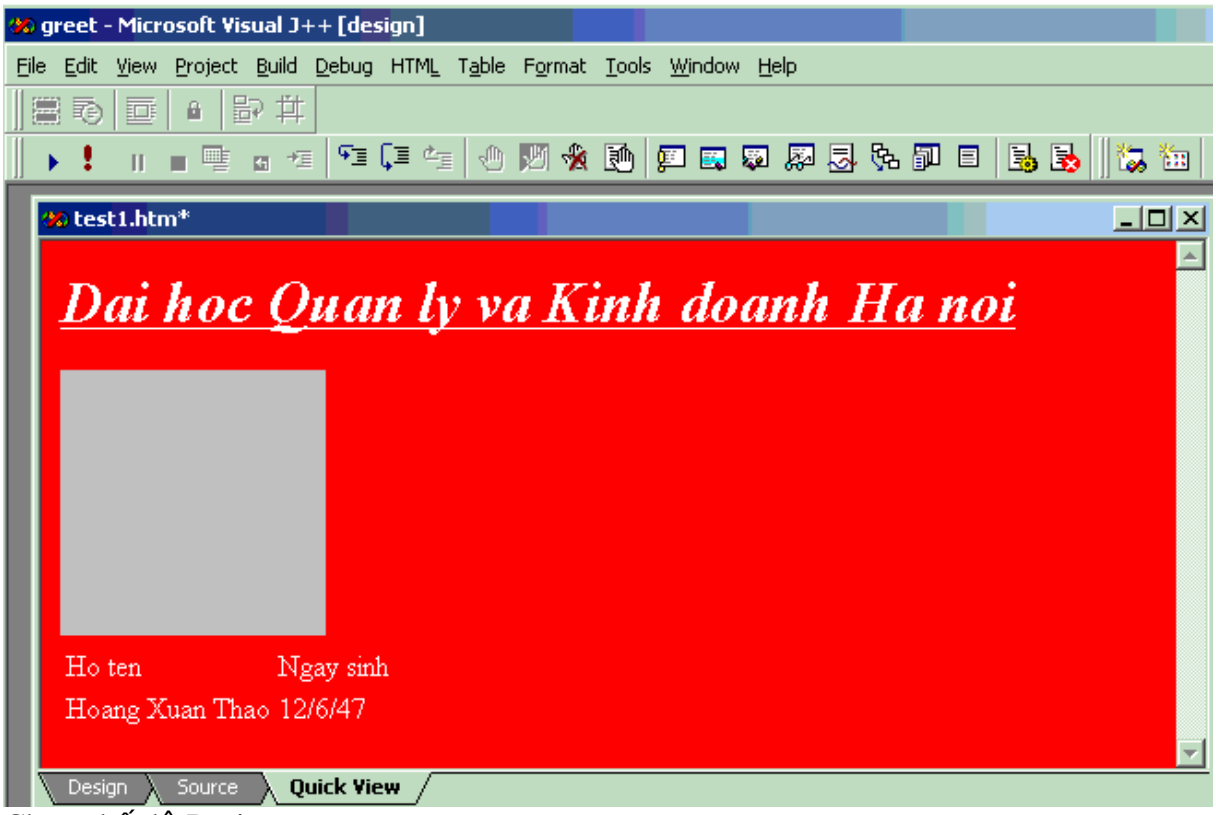
Design view: Dùng để tạo đồ hoạ cho tài liệu HTML

Source view: Cho phép bạn soạn, chỉnh sửa mã nguồn

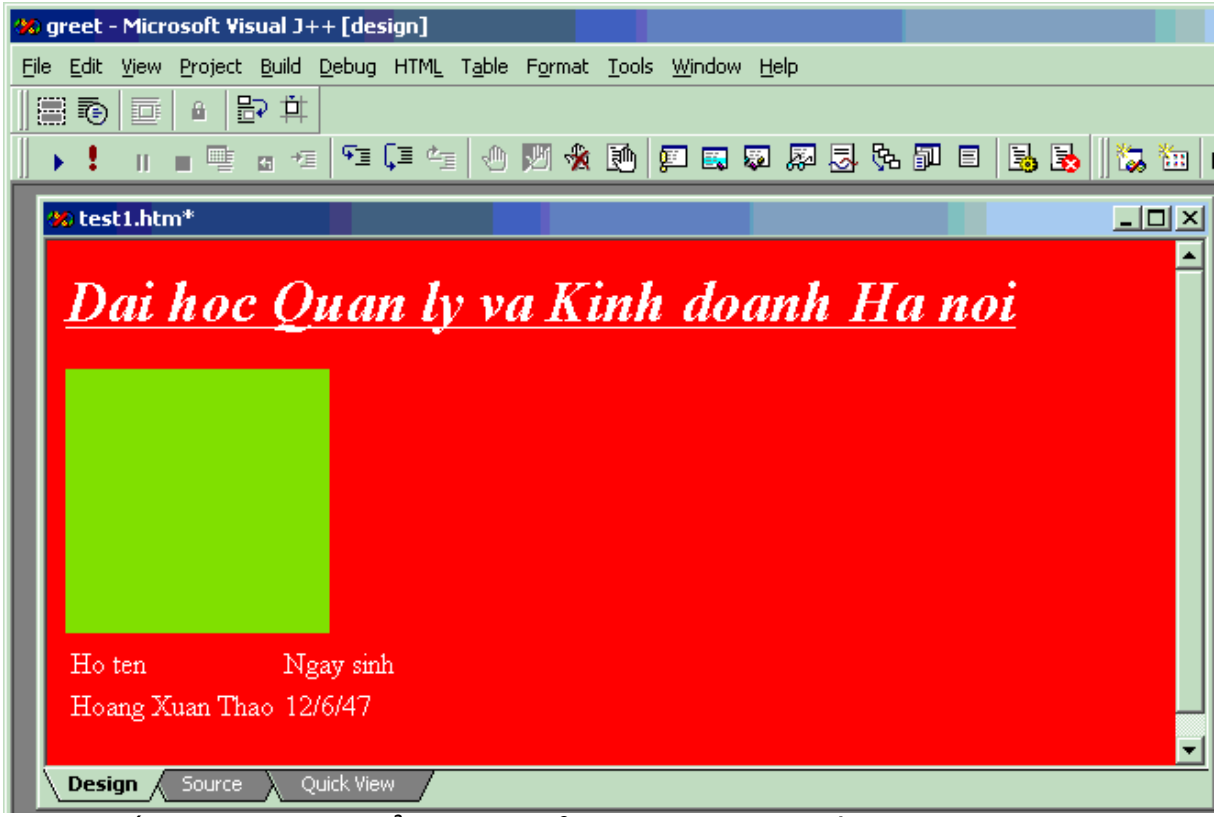
Quick view: Cho kết quả sau khi nó tự động duyệt qua Microsoft Internet Explorer.



Chọn chế độ Quick view ta có:



Chọn chế độ Design:

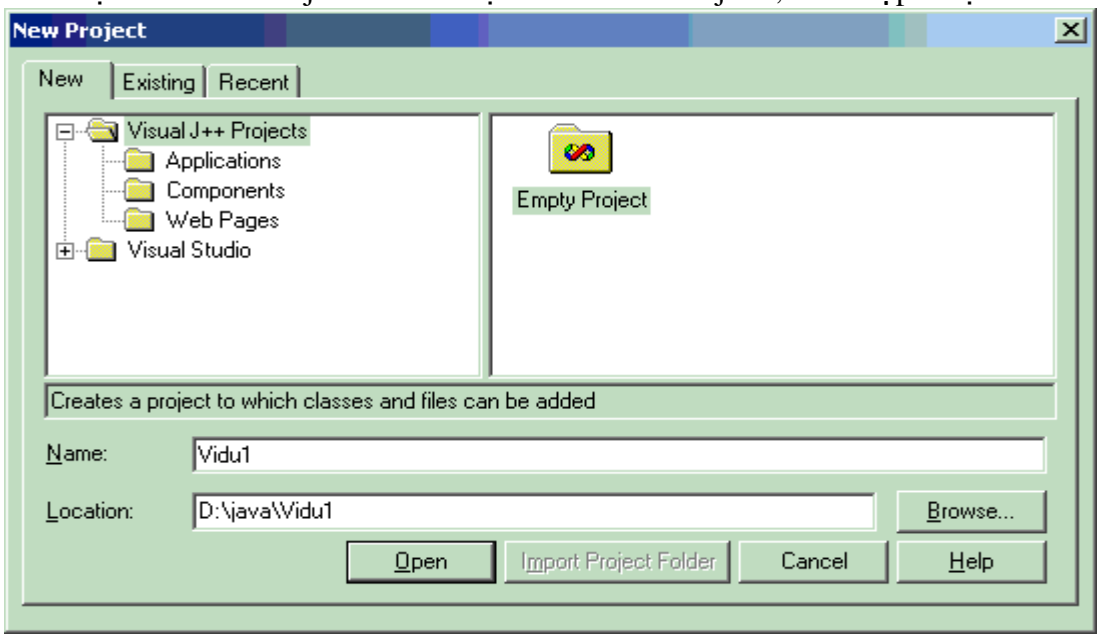


Trong chế Design bạn có thể soạn văn bản như trong môi trường Word. Ví dụ bạn tạo chữ đậm: **Sinh viên** thì trong môi trường Source sẽ có thẻ:
<B> <I> Sinh viên </I></B> v.v..

3.1.1 Tạo một tài liệu HTML trong HTML Editer

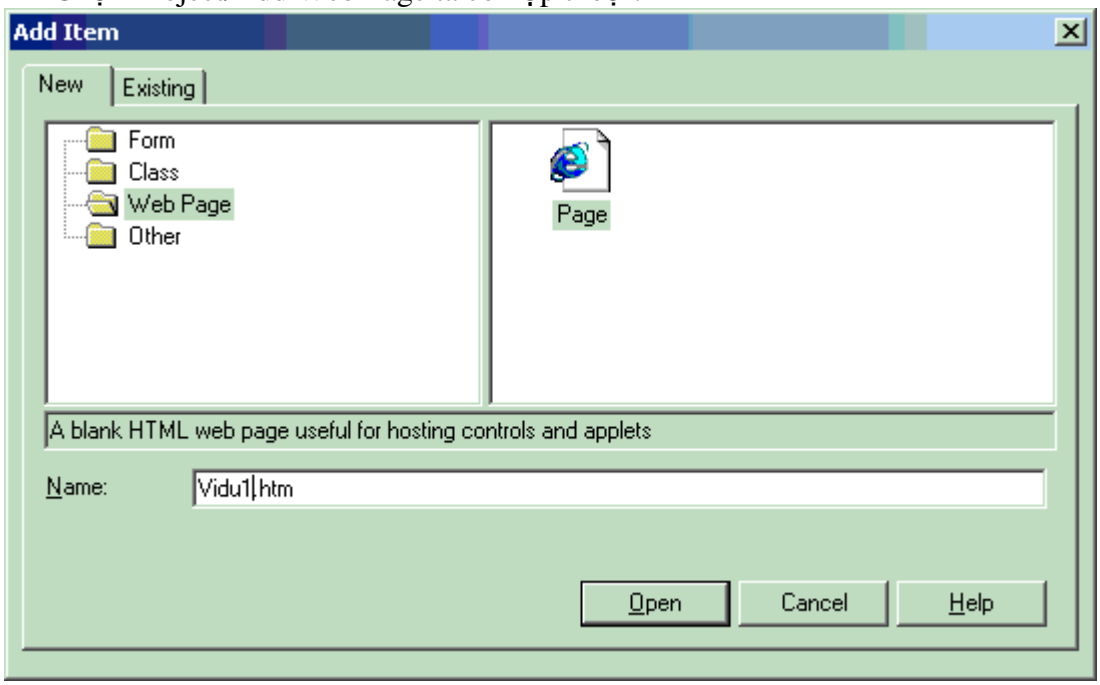
Để tạo môt tài liệu HTML với bộ chỉnh sửa HTML trong Visual Studio (trong J++) ta thao tác như sau:

Chọn File/New Project sau đó chọn Visual J++ Projects, ta có hộp thoại:

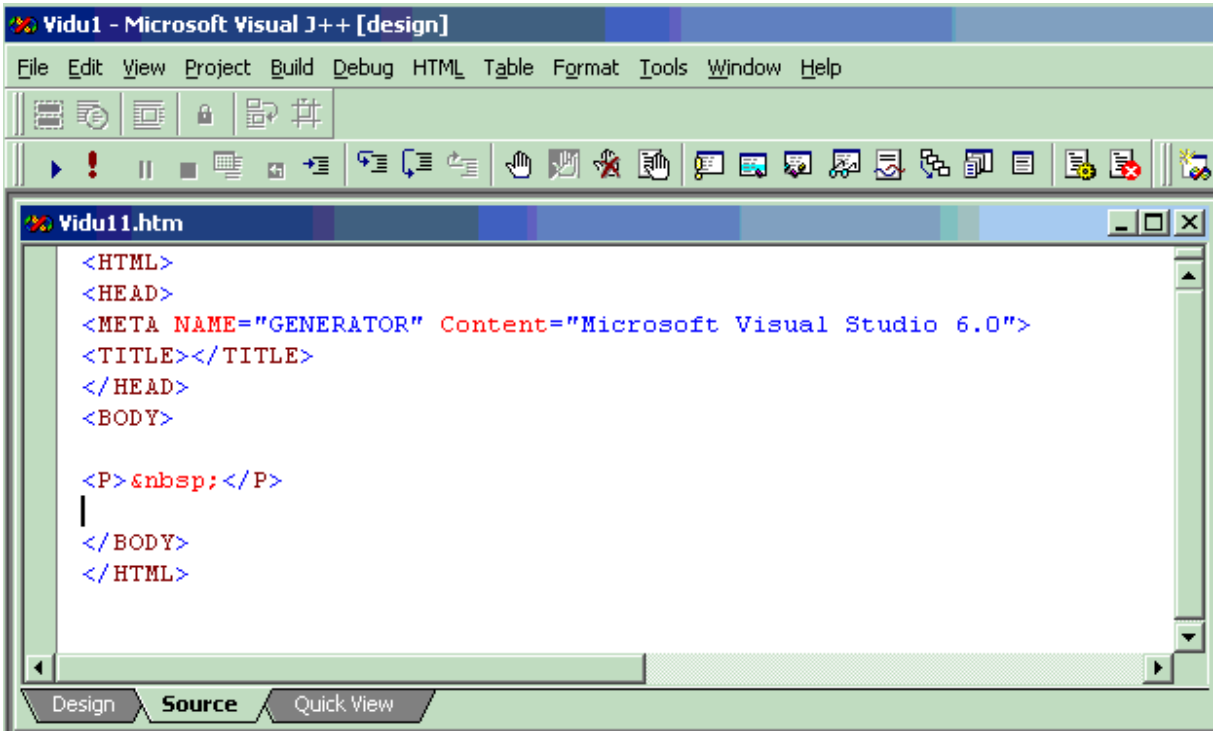


Đổi tên thành Vidu1 ở hộp văn bản Name (nếu cần), chọn Emply Project và chọn Open

Chọn Project/Add Web Page ta có hộp thoại:



Chọn Page, nếu cần thì đổi tên tệp thành Vidu1.html ở hộp văn bản Name và chọn Open, sau đó chọn Source ta có:



Bạn đưa trỏ vào những chỗ thích hợp để gõ các thẻ vào, bạn cũng có thể dùng cửa sổ Properties để đặt các thuộc tính cần thiết khi thiết kế.

Chú ý: Khi ở chế độ **Source** nếu bạn muốn quan sát đồ hoạ hãy chọn: View/View Controls Graphically, muốn quay về chế độ Text hãy chọn: View/View Controls As Text.

3.1.2 Tạo một Applet đơn giản

Để tạo một Applet bạn phải tìm hiểu thêm môt số vấn đề khác, cụ thể:

Phải bổ sung các câu lệnh import

Các Method Applet

Các keyword extends

Và một số vấn đề khác

Các Applet chứa tối thiểu hai câu lệnh:

import java.applet.* và

import java.awt.*

Gói (package) **java.applet** chứa một class có tên **Applet**, mỗi class do bạn tự tạo phải dựa trên class này.

Gói **java.awt** (**A**bstract **W**indows **T**oolkit) chứa các thành phần Windows như nhãn, menu, nút, v.v..Nhãn là một class chứa text mà ta có thể dùng để hiển thị trong Applet.

Trong trình ứng dụng thì method main() sẽ lần lượt gọi các method khác, nhưng trong Applet thì bộ trình duyệt sẽ gọi nhiều method một cách tự động, mỗi applet bao gồm 4 method sau:

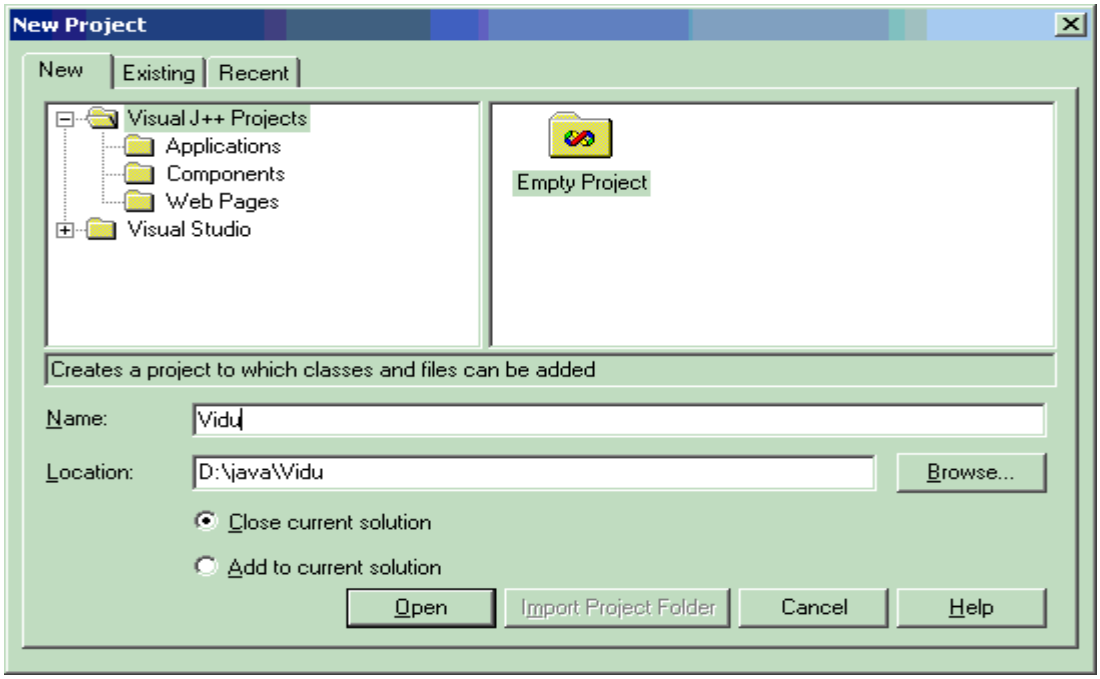
```
public void init()
public void start()
public void stop()
public void destroy()
```

Nếu không có các method này thì java sẽ tạo chúng và chúng chỉ có các nối kết đóng mở mà không có nội dung. Method init() là method đầu tiên được gọi trong một applet, bạn sử dụng nó để khởi tạo các biến hoặc đặt các applet lên màn hình v.v..Ví dụ sau là một lớp có tên Vidu để hiển thị "Chào bạn !" lên màn hình:

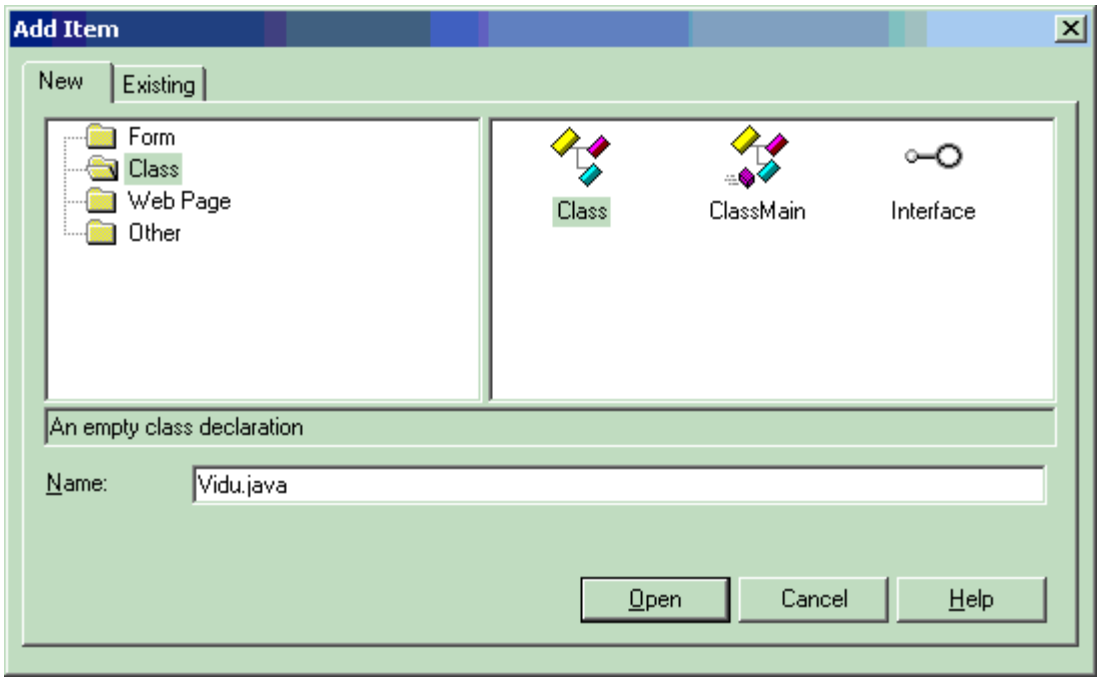
```
import java.applet.*;
import java.awt.*;
public class Vidu extends Applet
   {
        Label Label1=new Label("Chao ban ");
          public void init()
                {
                        add(Label1);
                }
    }
```

Để tạo và chạy một applet bạn theo các bước sau:

Chọn New Project sau đó chọn Visual J++ Projects ta có:



Chọn Emply Project và gõ Vidu vao hộp Name, chọn Open ta có hộp thoại tiếp và tiếp theo chọn tiếp Project/Add Class ta có:



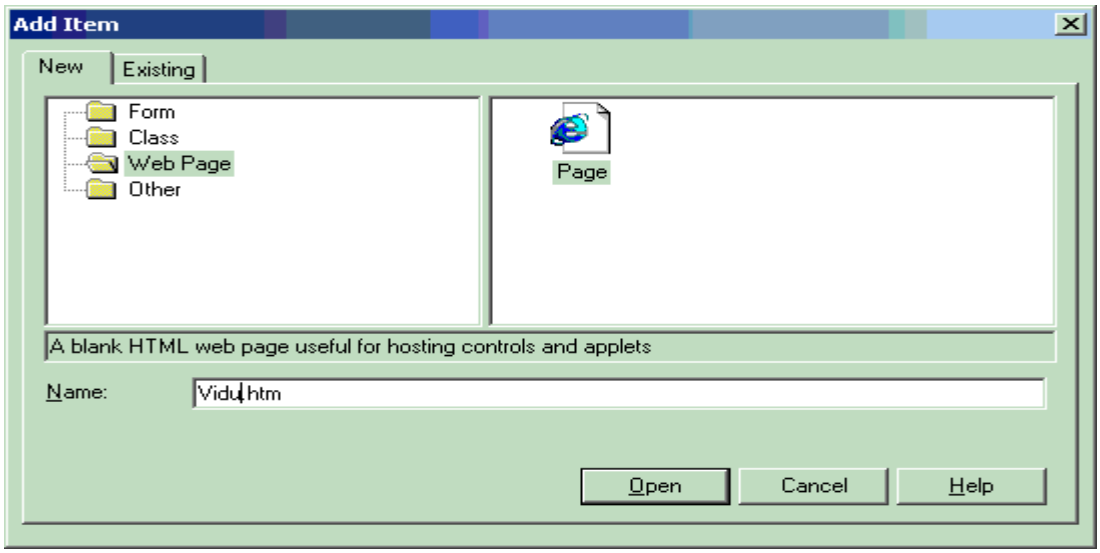
Gõ Vidu.java vào hộp Name và chọn Open ta có mã sau:

```
public class Vidu
{
}
```

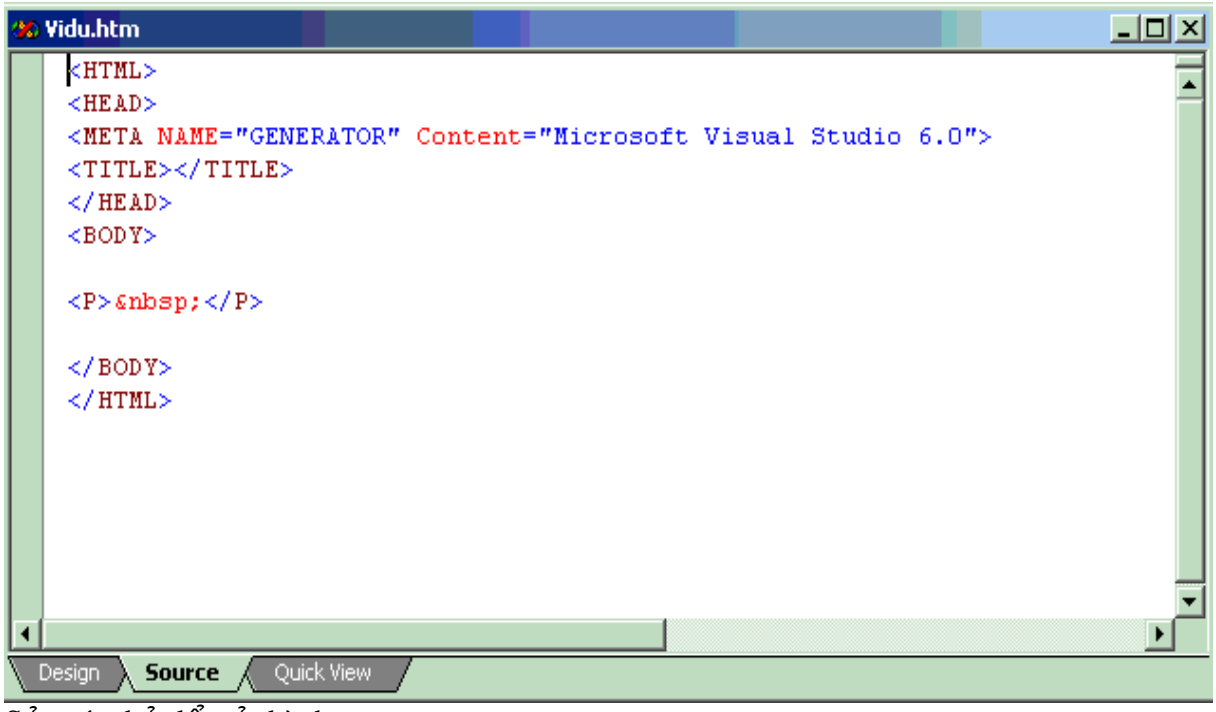
Bỏ mã đó và gõ vào như sau:

```
import java.applet.*;
import java.awt.*;
public class Vidu extends Applet
   {
        Label Label1=new Label("Chao ban ");
          public void init()
                {
                        add(Label1);
                }
    }
```

Tiếp theo chọn Project/ Add Web Page ta có:



Gõ Vidu.htm vào hộp Name và chọn Open, ta có:

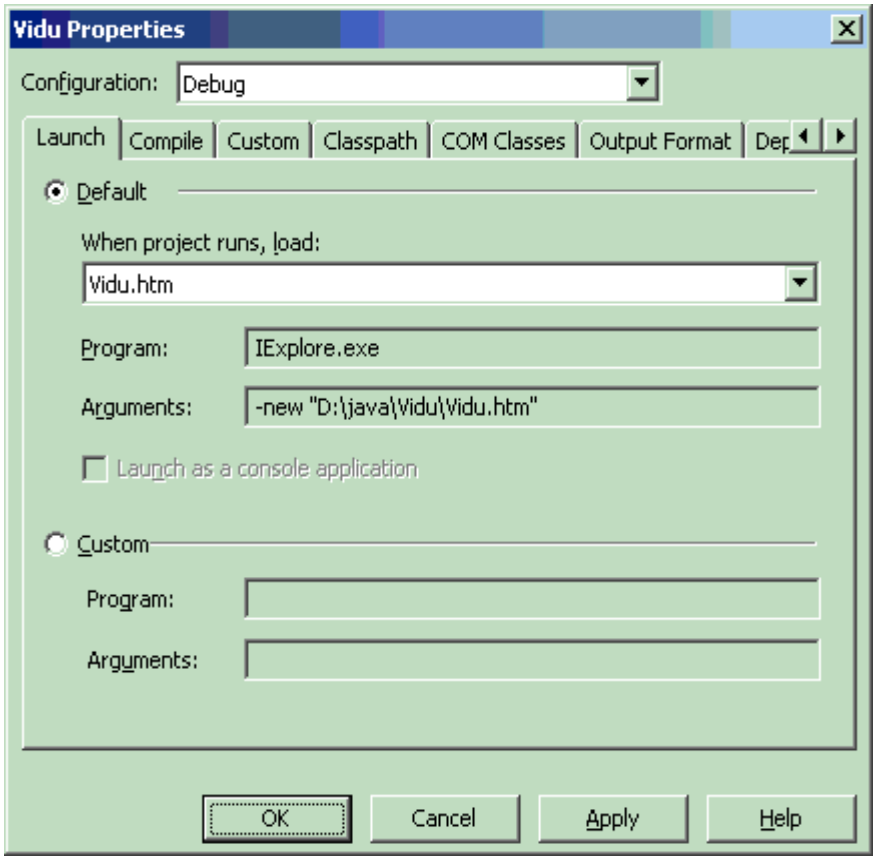


Sửa các thẻ để trở thành:

```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>
<P> </P>
<APPLET
        CODE=Vidu.class
        NAME=Vidu
        WIDTH=320
        HEIGHT=200 >
        <PARAM NAME=background value="008080">
        <PARAM NAME=foreground value="FFFFFF">
```

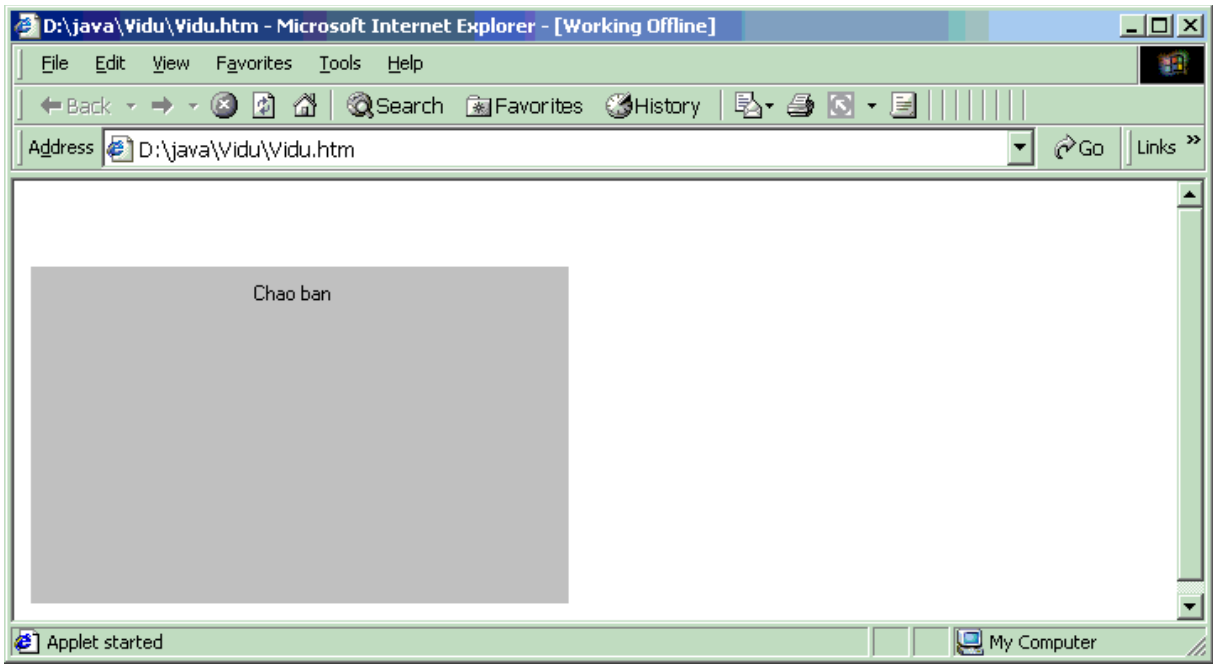
```
        </APPLET>
</BODY>
</HTML>
```

Chọn Project/Vidu Properties và đặt Vidu.htm vào hộp When projects runs, load như sau



Chọn OK

Chạy bằng cách chọn Debug/Start, ta có kết quả:



3.2 Cửa sổ

Java có rất nhiều lớp khác nhau như Frame, Panel, Button, TextField, TextArea v.v.. nhờ nó bạn có thể xây dựng được các ứng dụng riêng của mình. Các lớp này là một phần của lớp AWT (Abstract Window Tooolkit).
AWT chứa nhiều nội dung phong phú, trước hết ta làm quen với lớp Frame trong đó.

3.2.1 Tạo cửa sổ bằng lớp Frame

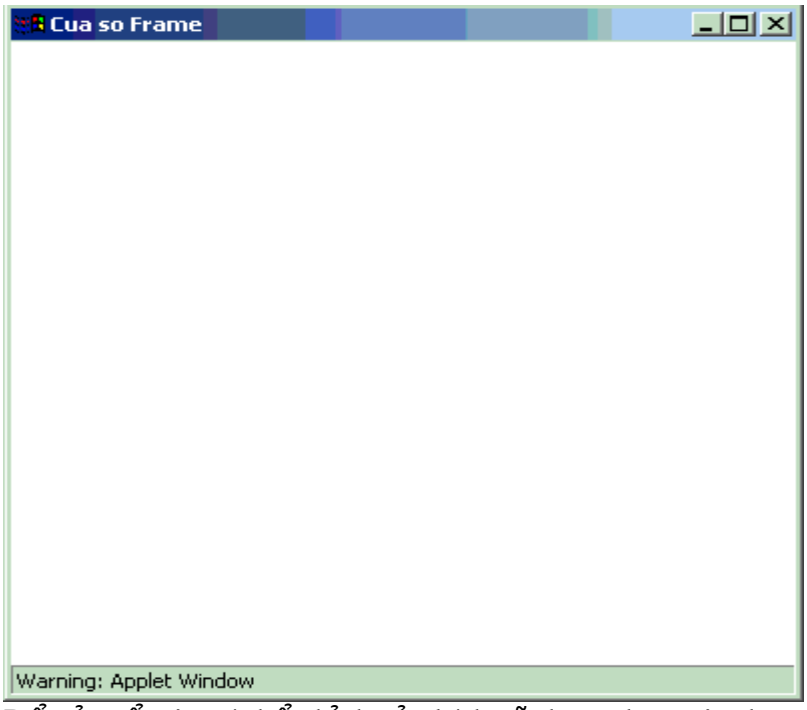
Để tạo một cửa sổ ta dùng lớp Frame. Bạn xem đoạn mã sau:

```
import java.applet.*;
import java.awt.*;
public class Vidu extends Applet
    {
      Frame WinFrame;
        public void init()
              {
                WinFrame=new Frame("Cua so Frame”);
                WinFrame.setSize(400,400);
                WinFrame.setVisible(true);
              }
    }
```

Bạn hãy chú ý các dòng lệnh trên:

- Frame WinFrame; (khai báo biến WinFrame)
- WinFrame=new Frame("Cua so Frame”); (Đây là một constructor: Tạo cửa sổ có tiêu đề “Cua so Frame”)
- WinFrame.setSize(400,400); (Đặt kích cỡ cửa sổ 400 x 400
- WinFrame.setVisible(true); (Đặt true ứng với lộ diện cửa sổ)

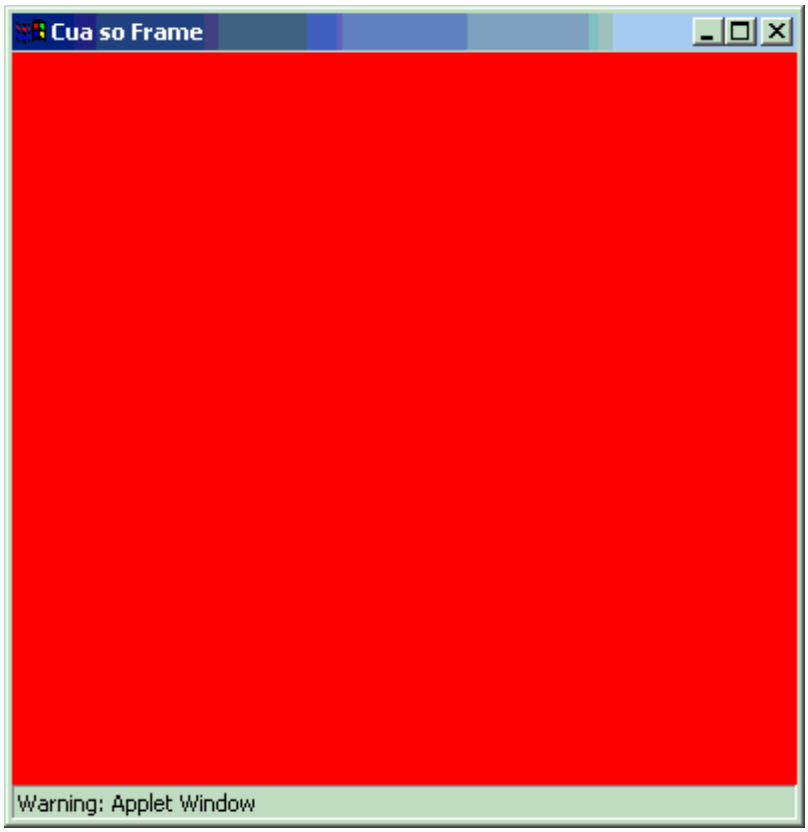
Biên dịch (Build/Build) và chạy (Debug/Starrt) ta có cửa sổ sau:



Để cửa sổ trên có thể chỉnh sửa kích cỡ được, bạn nên đưa vào dòng lệnh:
WinFrame.setResizable(true);
còn nếu WinFrame.setResizable(false); thì không chỉnh cỡ được.
Nếu bạn cần thay đổi tiêu đề cửa sổ hãy dùng phương thức:
WinFrame.setTitle("Day la tieu de moi");
Bạn cũng có thể tạo màu bằng:
WinFrame.setBackground(Color.red);
Đoạn mã trên có thể thêm để trở thành:

```
import java.applet.*;
import java.awt.*;
public class Vidu extends Applet
      {
         Frame WinFrame;
           public void init()
                   {
                     WinFrame=new Frame("Cua so Frame");
                     WinFrame.setSize(400,400);
                     WinFrame.setVisible(true);
                   WinFrame.setResizeable(true);
                   WinFrame.setBackground(Color.red);
                  }
      }
```

Sau khi chạy ta có cửa sổ mới như sau:



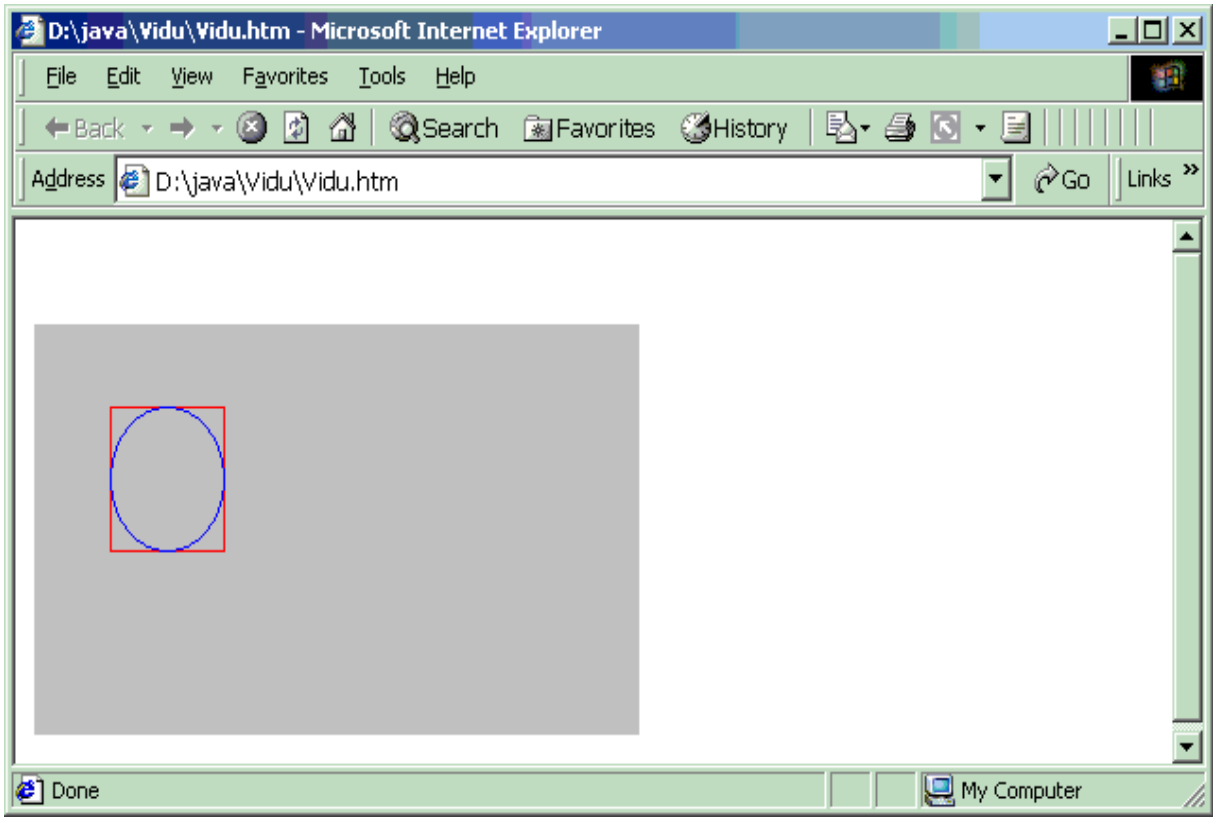
3.2.2 Tính kế thừa trong Applet

Đoạn mã sau dùng để vẽ một hình chữ nhật và một hình ellipse nội tiếp trong hình chữ nhật đó. Bạn quan sát lớp Shape xem nó hoạt động thế nào!

```
import java.applet.*;
import java.awt.*;
public class Vidu extends Applet
        {
        Shape m_Shape=new Shape();
         public void paint(Graphics g)
                    {
                       m_Shape.draw(g);
                    }
                     public class Shape
                        {
                            int x1=40,y1=40;
                            public void draw(Graphics g)
                              {
                                    g.setColor(Color.red);
                                    g.drawRect(x1,y1,60,70);
                                    g.setColor(Color.blue);
                                    g.drawOval(x1,y1,60,70);
                              }
                            public void setPosition(int x, int y)
                                {
```

```
                    x1=x;
                    y1=y;
                }
            public void movePosition(int dx, int dy)
                {
                    x1=x1+dx;
                    y1=y1+dy;
                }
        }
    }
```

Khi chạy ta có kết quả sau:

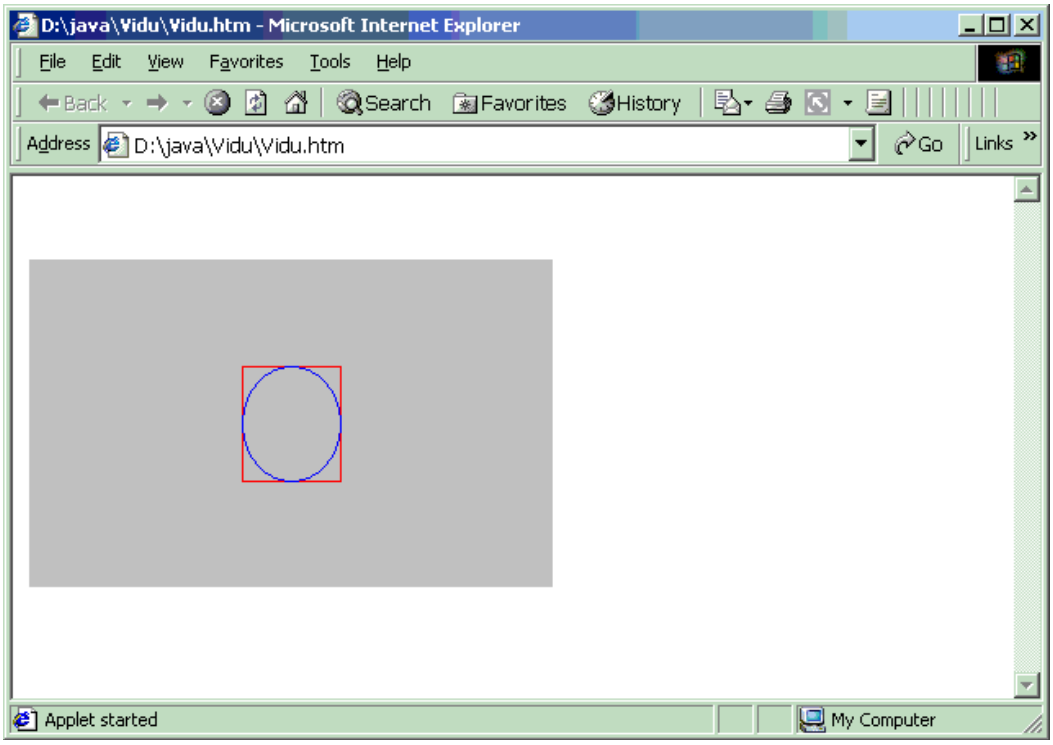


Bây giờ ta kế thừa và thêm một số lệnh (phần bôi đen) để có lớp mới như đoạn mã sau:

```
import java.applet.*;
import java.awt.*;
public class Vidu extends Applet
    {
    // Tạo m_Shape từ lớp kế thừa KeThuaShape
     KeThuaShape m_Shape=new KeThuaShape();
      public void paint(Graphics g)
            {
          // Gọi TrungTamShape() từ lớp kế thừa
              m_Shape.TrungTamShape();
              m_Shape.draw(g);
            }
```

```
public class Shape
{
        int x1=40,y1=40;
        public void draw(Graphics g)
          {
                g.setColor(Color.red);
                g.drawRect(x1,y1,60,70);
                g.setColor(Color.blue);
                g.drawOval(x1,y1,60,70);
          }
        public void setPosition(int x, int y)
           {
                x1=x;
                y1=y;
           }
        public void movePosition(int dx, int dy)
           {
                x1=x1+dx;
                y1=y1+dy;
           }
  }

//Kế thừa
public class KeThuaShape extends Shape
{
        public void TrungTamShape()
        {
                Dimension Size=getSize();
                int newx, newy;
                newx=(Size.width/2)-30;
                newy=(Size.height/2)-35;
                setPosition(newx,newy);
        }
}
}
```

Sau khi chạy ta có kết quả sau:



Lớp **KeThuaShape** có mọi tính năng của **Shape**, chúng ta chỉ thêm phương thức **TrungTamShape**() để đưa hình được tạo ra nằm giữa cửa sổ Applet mà thôi.

3.3 Menu

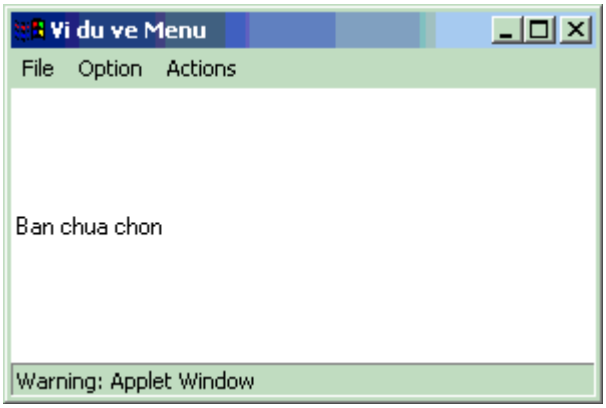
Trong Java có khái niệm **siêu lớp-** super, một lớp đã tồn lại gọi là siêu lớp, lớp kế thừa từ nó gọi là lớp con, có khi siêu lớp còn gọi là lớp cha và cũng có thể là lớp con nếu nó đã được kế thừa (tuỳ ngữ cảnh). Phương thức super() có thể được sử dụng theo hai cách: để gọi phương thức của một siêu lớp, thứ hai để gọi constructor của một siêu lớp.

Ta có thể đưa Menu vào Applet. Trước hết phải tạo một cửa sổ bằng lớp Frame, sau đó đưa Menu vào cửa sổ đó. Hãy đọc đoạn mã sau:

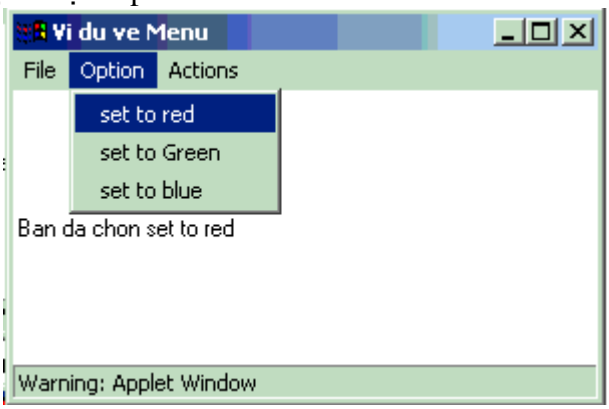
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Vidu extends Applet
    {
     MyFrame m_Frame=new MyFrame("Vi du ve Menu");
     Label m_1Label=new Label("Ban chua chon");
            public void init()
             {
                    m_Frame.add(m_1Label);
                    MyListener Listener=new MyListener();
                    MenuBar menu=new MenuBar();
                    Menu file=new Menu("File");
                    menu.add(file);
                    file.add("Open");
                    file.add("Close");
                    file.addActionListener(Listener);
```

```
                        Menu options=new Menu("Option");
                        menu.add(options);
                        options.add("set to red");
                        options.add("set to Green");
                        options.add("set to blue");
                        options.addActionListener(Listener);
                        Menu actions=new Menu("Actions");
                        menu.add(actions);
                        actions.add("go home");
                        actions.add("go to Work");
                        options.addActionListener(Listener);
                        m_Frame.setMenuBar(menu);
                        m_Frame.setSize(300,200);
                        m_Frame.setVisible(true);
                    }
public class MyListener implements ActionListener
  {
  public void actionPerformed(ActionEvent ae)
    {
          if (ae.getSource() instanceof MenuComponent)
          {
            m_1Label.setText("Ban da chon "+ae.getActionCommand());
          }
    }
  }
public class MyFrame extends Frame
   {
          MyFrame(String s)
          {
            super(s);
            addWindowListener(new WL());
          }
public class WL extends WindowAdapter
  {
 public void WindowClosing(WindowEvent e)
   {
     setVisible(false);
     m_Frame=null;
   }
  }
 }
}
```

Sau khi chạy ta có kết quả:



Nếu bạn chọn Option sẽ có:

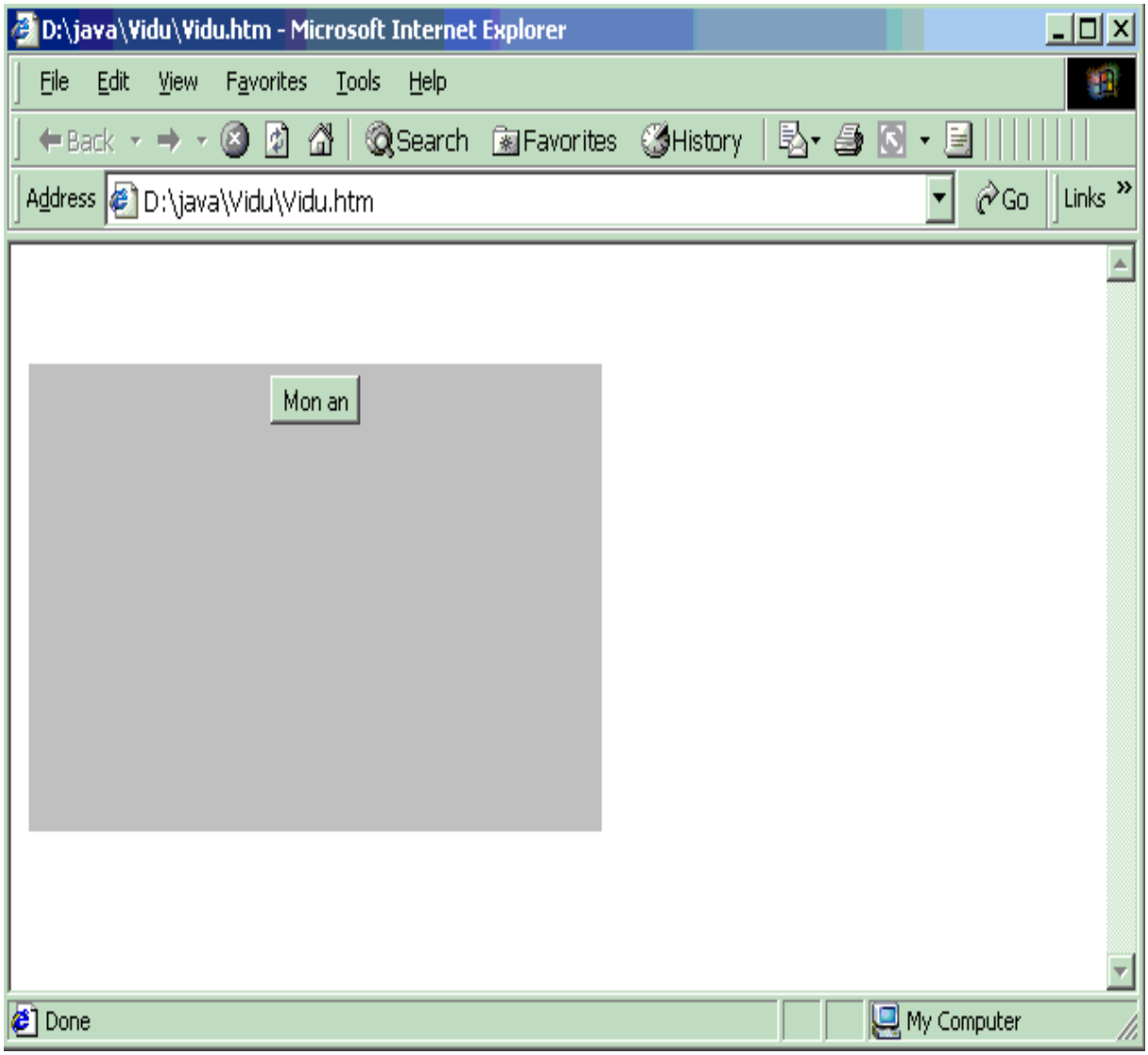


Sau đây là ví dụ về tạo Menu Popup:

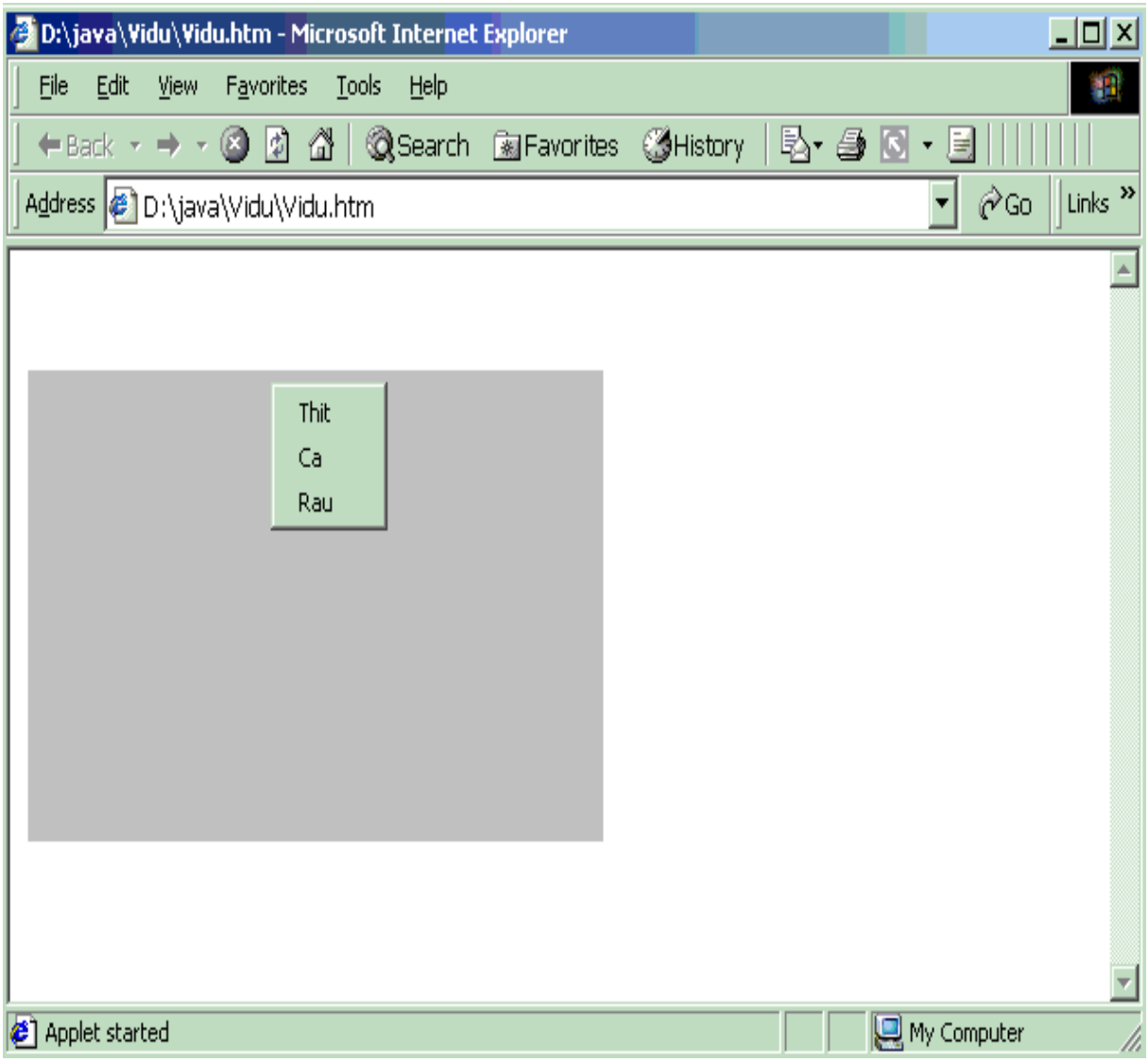
```
import java.applet.*;
import java.awt.*;
import java.awt.event.MouseListener;
import java.awt.event.*;
public class Vidu extends Applet
  {
    PopupMenu m_Popup=new PopupMenu();
       public void init()
        {
          m_Popup.add("Thit");
          m_Popup.add("Ca");
         m_Popup.add("Rau");
         MyButton button=new MyButton("Mon an");
         button.add(m_Popup);
         add(button);
        }
          public class MyButton extends Button
           {
            MyButton(String s)
              {
                 super(s);
               enableEvents(AWTEvent.ACTION_EVENT_MASK);
```

```
                }
        public void processActionEvent(ActionEvent ae)
           {
              m_Popup.show(this,0,0);
            }
     }
}
```

Sau khi chạy ta có kết quả:



Nếu bạn kích vào nút Mon an, 3 thành phần hiện ra như sau:



3.4 Nút nhấn (Button) trong Applet

Button là một thành phần giao tiếp với người dùng rất phổ biến. Bạn có thể tạo nút nhấn bằng hai cách:

Cách 1: Định nghĩa trực tiếp:

Button Button1=new Button("Nut 1");

Cách 2: Định nghĩa và gán nhãn sau:

Button Button1=new Button();

Button1.setLabel("Nut 1");

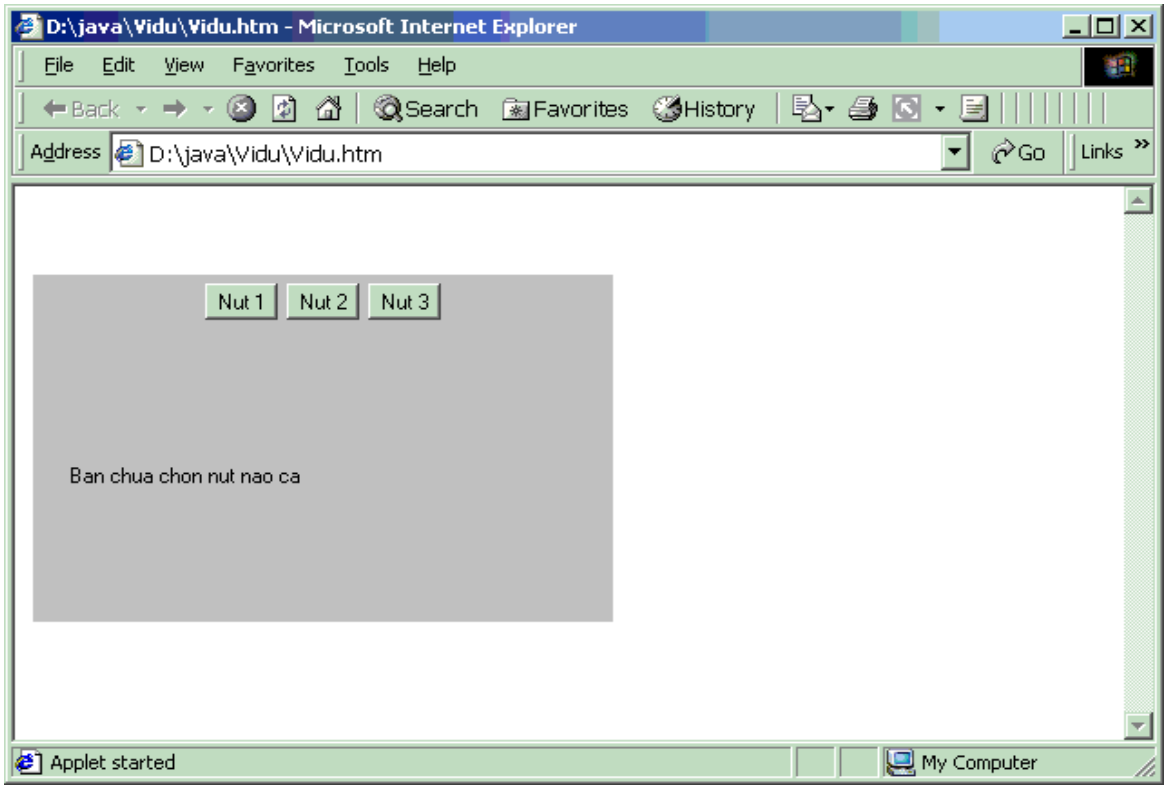
Đoạn mã sau dùng để tạo ra 3 nút có tên: nut1, nut2 và nut3. Ta dùng lớp kế thừa MyButton với phương thức:

public void processActionEvent(ActionEvent ae); để kiểm soát sự kiện kích chuột. Bạn hãy đọc kỹ từng dòng lệnh để biết tính năng của nó!

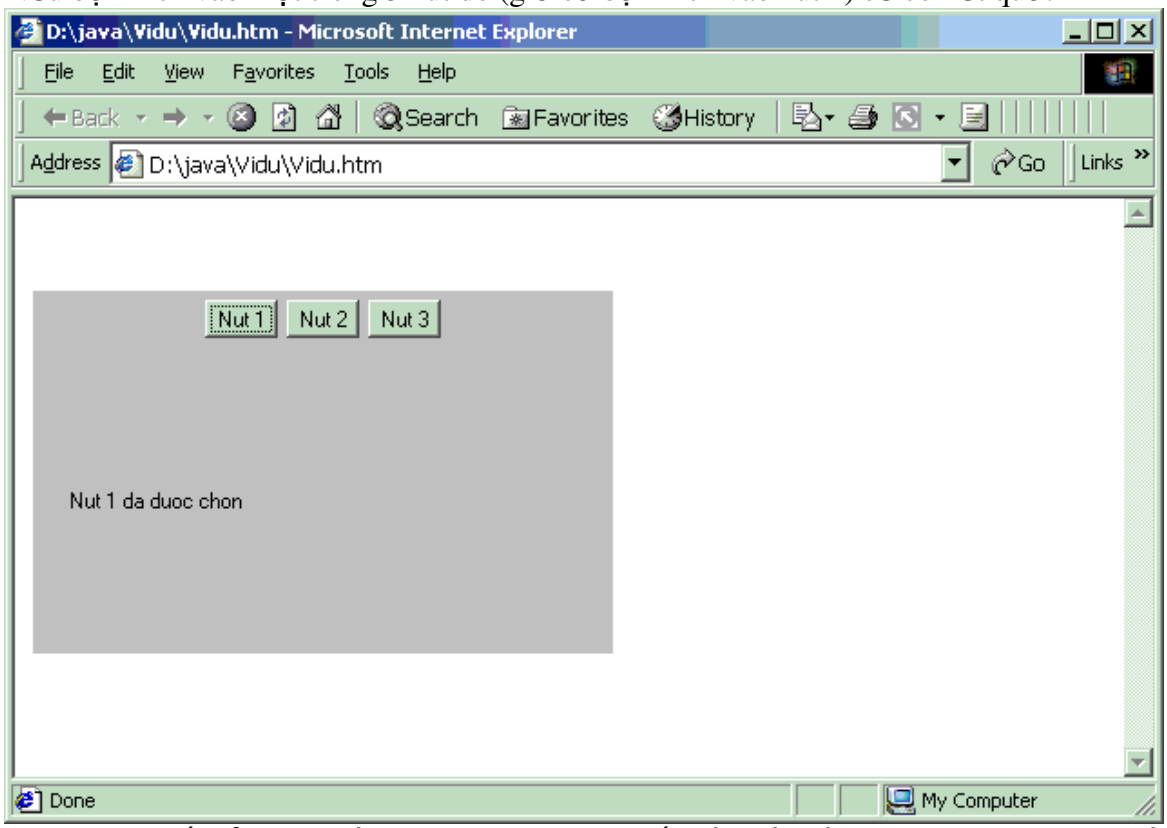
```
import java.applet.*;
import java.awt.*;
```

```
import java.awt.event.*;
public class Vidu extends Applet
  {
      String str="Nut chua nhan";
      MyButton nut1=new MyButton("Nut 1");
      MyButton nut2=new MyButton("Nut 2");
      MyButton nut3=new MyButton("Nut 3");
//Khởi tạo
public void init()
    {
      add(nut1);
      add(nut2);
      add(nut3);
    }
//Phương thức paint
public void paint(Graphics g)
     {
       g.drawString(str,20,120);
     }
//Lớp kế thừa của lớp Button
public class MyButton extends Button
    {
      String strText;
      MyButton(String s)
       {
         super(s);
         enableEvents(AWTEvent.ACTION_EVENT_MASK);
         strText=s;
       }
    public void processActionEvent(ActionEvent ae)
        {
          str=strText+" da duoc chon";
          getParent().repaint();
          super.processActionEvent(ae);
        }
      }
 }
```

Sau khi chạy ta có kết quả:



Nếu bạn kích vào một trong 3 nút đó (giả sử bạn kích vào nut 1) sẽ có kết quả:



Như bạn thấy ở trên, lớp MyButton được kế thừa từ lớp Button. Bên trong lớp MyButton có gọi hai phương thức **super()** và **enableEvents()** để lớp này kiểm soát được các sự kiện. Phương thức **processActionEvent**() kiểm soát sự kiện kích chuột và đưa ra

dòng văn bản và sau đó vẽ lại cha của nó bằng phương thức getParent().repaint(); cuối cùng là gọi phương thức:
super.processActionEvent(ae);

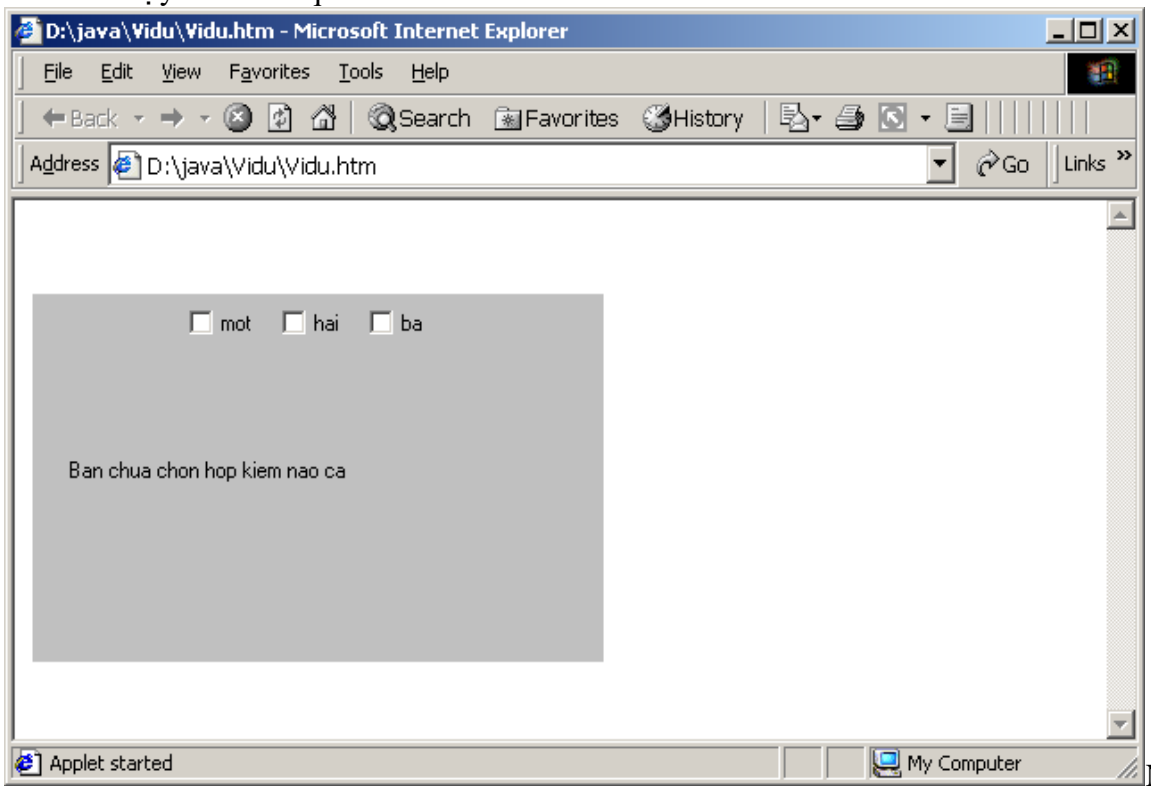
3.5 Hộp kiểm (Checkbox) trong Applet

Hộp kiểm có hai trạng thái: chọn được (true) hay không chọn được (false). Nếu tồn tại nhiều Checkbox thì chúng có thể độc lập hoặc có liên hệ với nhau tuỳ chúng ta thiết lập. Nếu thiết lập mối liên hệ với nhau thì chỉ có một thành phần Checkbox được chọn.
Checkbox có 4 cách tạo (constructor) khác nhau.
Constructor thứ nhất có dạng:
Checkbox Kiem=new Checkbox();
Constructor này không sử dụng đối số và chưa có nhãn, muốn gắn nhãn bạn phải dùng getLabel().
Constructor thứ hai có dạng:
Checkbox Kiem=new Checkbox("Day la Chechbox"**);**
Constructor thứ ba có dạng:
Checkbox Kiem=new Checkbox("Day la Chechbox",groupObject, true**);**
Đối số thứ nhất là nhãn, đối số thứ hai là đối tượng nhóm, đối số thứ 3 là trạng thái khởi động.
Constructor thứ tư có dạng:
Checkbox Kiem=new Checkbox("Day la Chechbox",true**);**
Đối số thứ nhất là nhãn, đối số thứ hai là trạng thái khởi động
Cũng như Button việc kiểm tra sự kiện kích chuột được đảm trách bởi phương thức enableEvents(AWTEvent.ITEM_VEN_MASK) và processItemEvent(). Ví dụ sau là đoạn mã tạo 3 hộp kiểm để bạn chọn và thông báo hộp kiểm bạn chọn mới nhất:

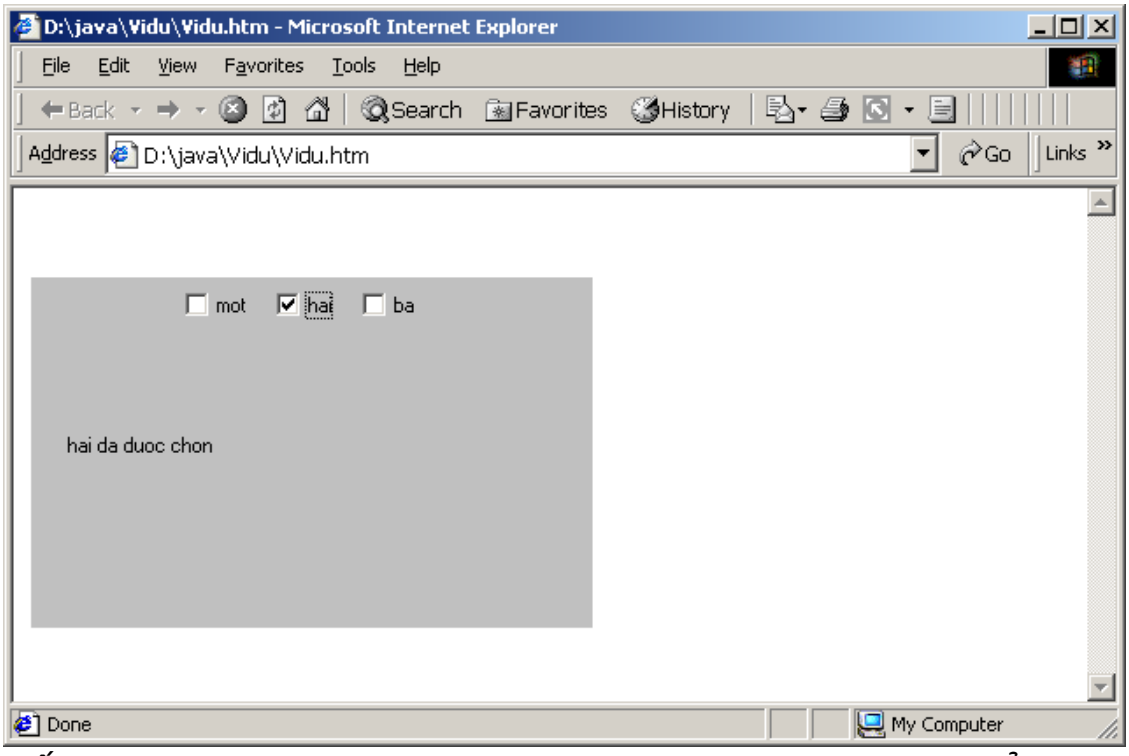
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Vidu extends Applet
  {
   String str="Ban chua chon hop kiem nao ca";
   MyCheckbox Kiem1=new MyCheckbox("mot");
   MyCheckbox Kiem2=new MyCheckbox("hai");
   MyCheckbox Kiem3=new MyCheckbox("ba");
public void init()
    {
      dd(Kiem1);
     add(Kiem2);
       add(Kiem3);
     }
   public void paint(Graphics g)
    {
       g.drawString(str,20,100);
    }
public class MyCheckbox extends Checkbox
 {
```

```
   String strText;
   MyCheckbox(String s)
     {
       super(s);
      strText=s;
       enableEvents(AWTEvent.ITEM_EVENT_MASK);
     }
 public void processItemEvent(ItemEvent ie)
       {
          str=strText+" da duoc chon";
          getParent().repaint();
          super.processItemEvent(ie);
        }
     }
  }
```

Sau khi chạy ta có kết quả:



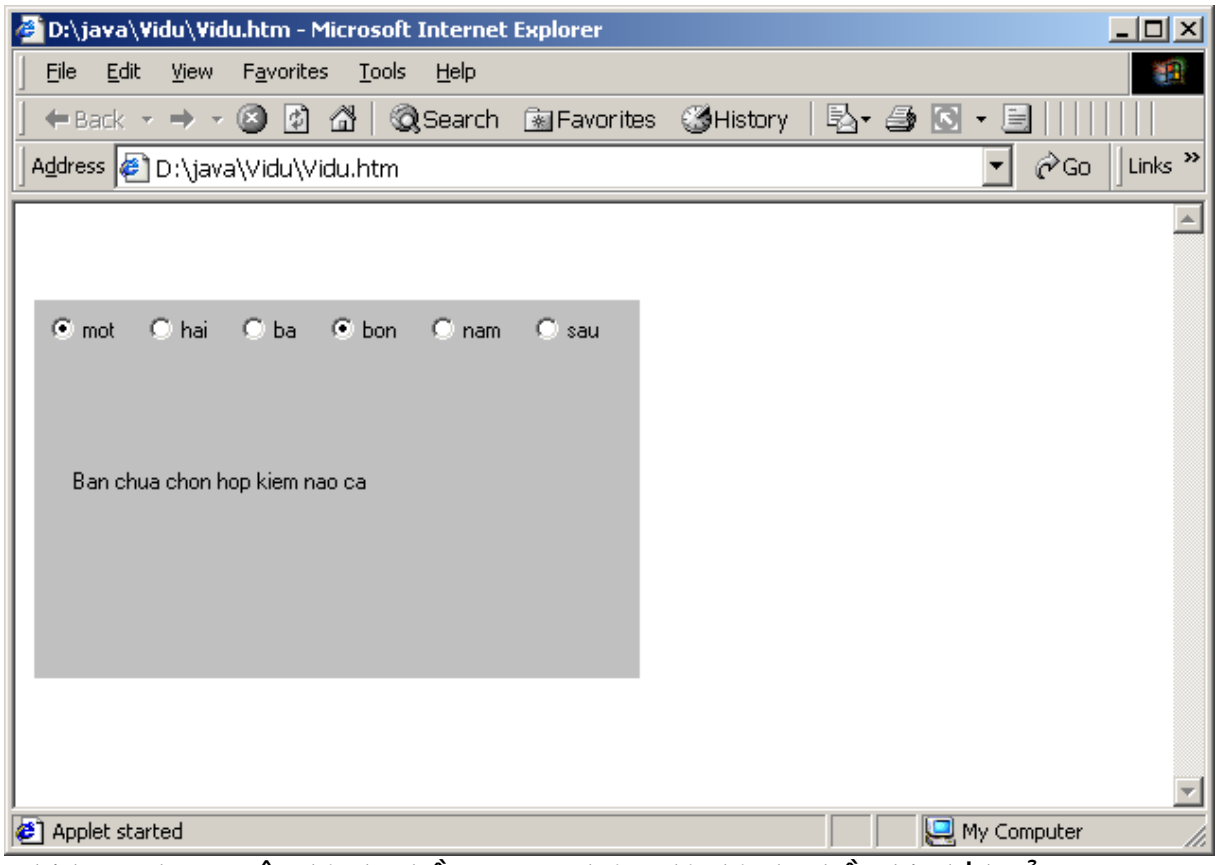
Nếu bạn chọn một hộp se có kết quả:



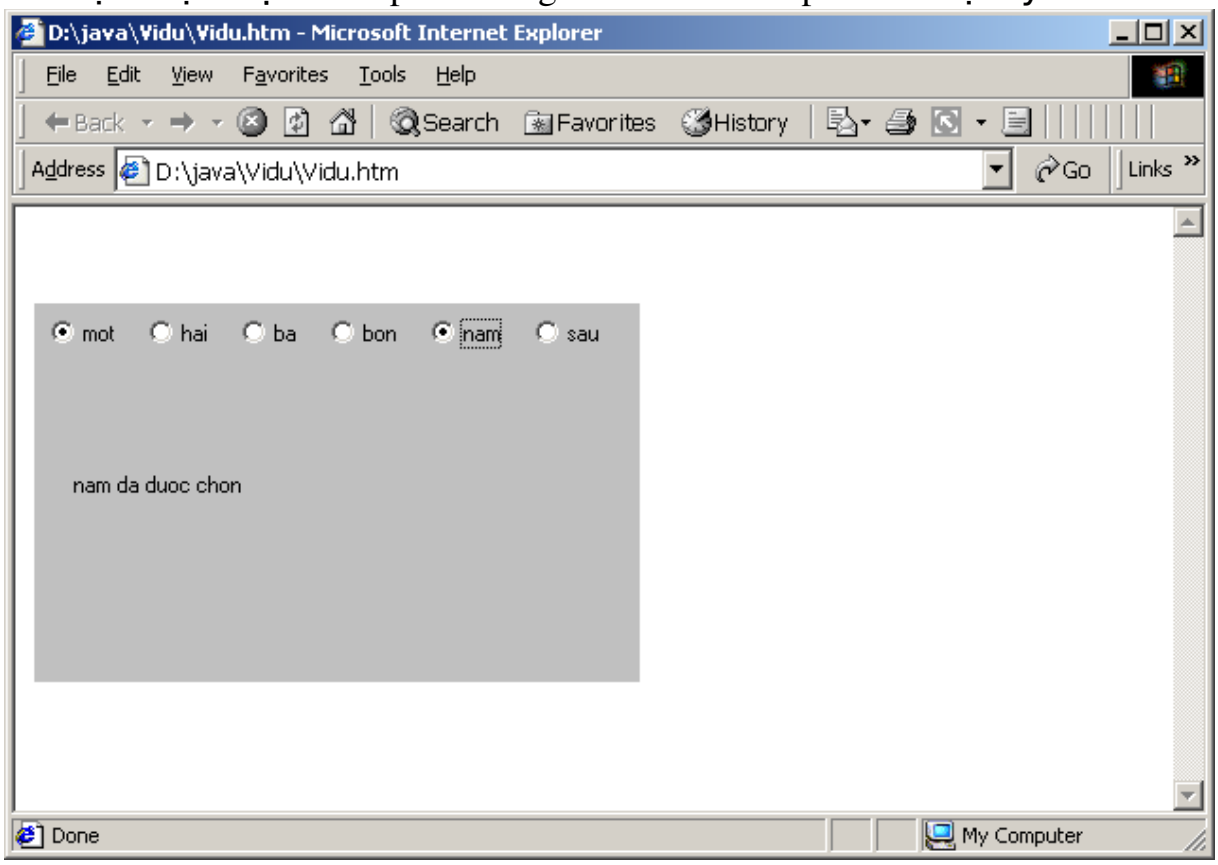
Nếu chúng ta nhóm các Checkbox lại thì hình dạng Checkbox trở thành nút chọn, bạn quan sát đoạn mã sau:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Vidu extends Applet
  {
    String str="Ban chua chon hop kiem nao ca";
    CheckboxGroup nhom1=new CheckboxGroup();
    CheckboxGroup nhom2=new CheckboxGroup();
    Checkbox Kiem1=new Checkbox("mot",nhom1,true);
    Checkbox Kiem2=new Checkbox("hai",nhom1,false);
    Checkbox Kiem3=new Checkbox("ba",nhom1,false);
    MyCheckbox Kiem4=new MyCheckbox("bon",nhom2,true);
    MyCheckbox Kiem5=new MyCheckbox("nam",nhom2,false);
    MyCheckbox Kiem6=new MyCheckbox("sau",nhom2,false);
public void init()
{
  add(Kiem1);
  add(Kiem2);
  add(Kiem3);
  add(Kiem4);
  add(Kiem5);
  add(Kiem6);
```

```
}
public void paint(Graphics g)
   {
     g.drawString(str,20,100);
   }
public class MyCheckbox extends Checkbox
   {
      String strText;
      MyCheckbox(String s,CheckboxGroup group, boolean state)
          {
             super(s,group,state);
              strText=s;
              enableEvents(AWTEvent.ITEM_EVENT_MASK);
          }
public void processItemEvent(ItemEvent ie)
     {
           str=strText+" da duoc chon";
           getParent().repaint();
           super.processItemEvent(ie);
     }
  }
}
```



Khi bạn chọn một thành phần trong nhóm thì thành phần kia bị huỷ:



3.6 Điều khiển Choice

Điều khiển này giống như hộp Combo trong Visual Basic, Choice sẽ tạo ra một danh sách đổ xuống và mỗi lúc chỉ chọn được một phần tử, bản thân Choice không sắp xếp được các phần tử nên khi đưa vào bạn nên sắp xếp trước. Điều khiển này chỉ có một constructor:

Choice ch=new Choice()

Sau khi tạo xong, ta dùng addItem() để gắn các phần tử vào, ví dụ:

Choice.addItem(“ 1. Nguyễn Du”);

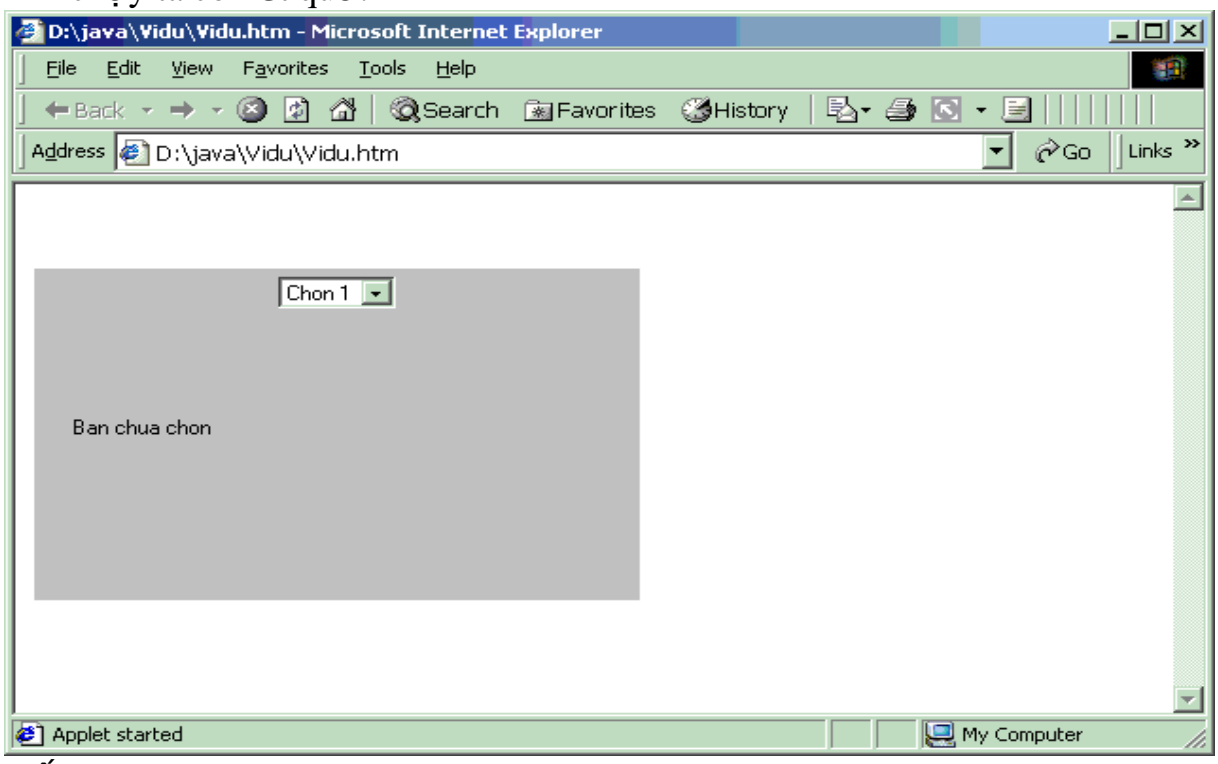
Choice.addItem(“ 2. Tố Hữu”);

Các sự kiện của Choice cũng giống như Checkbox, bạn quan sát đoạn mã sau:

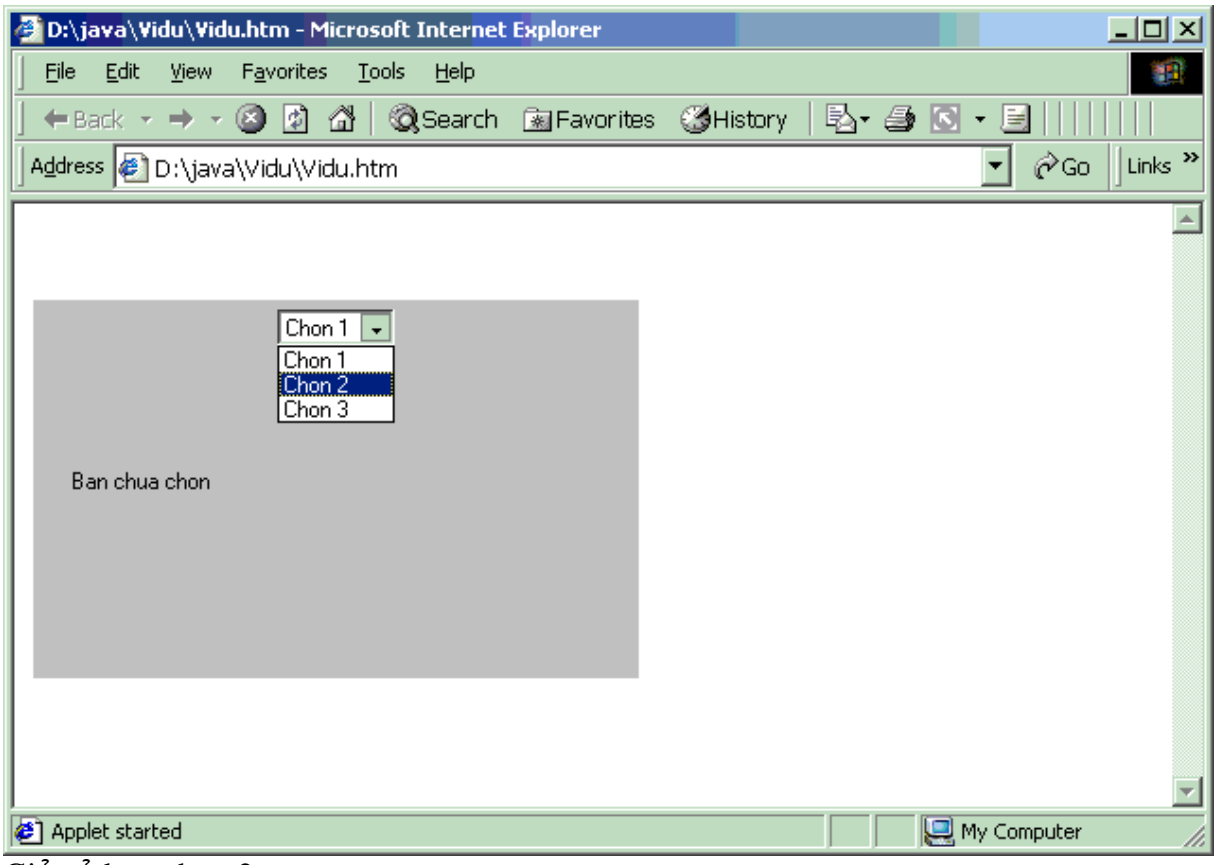
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Vidu extends Applet
{
   String str="Ban chua chon ";
   MyChoice Ch=new MyChoice();
  public void init()
     {
    Ch.add("Chon 1");
    Ch.add("Chon 2");
    Ch.add("Chon 3");
    add(Ch);
  }
public void paint(Graphics g)
  {
     g.drawString(str,20,100);
  }
public class MyChoice extends Choice
   {
       String strText="ban";
        MyChoice()
             {
                super();
                enableEvents(AWTEvent.ITEM_EVENT_MASK);
             }
public void processItemEvent(ItemEvent ie)
     {
       str=strText+" da chon";
```

```
            getParent().repaint();
        }
    }
}
```

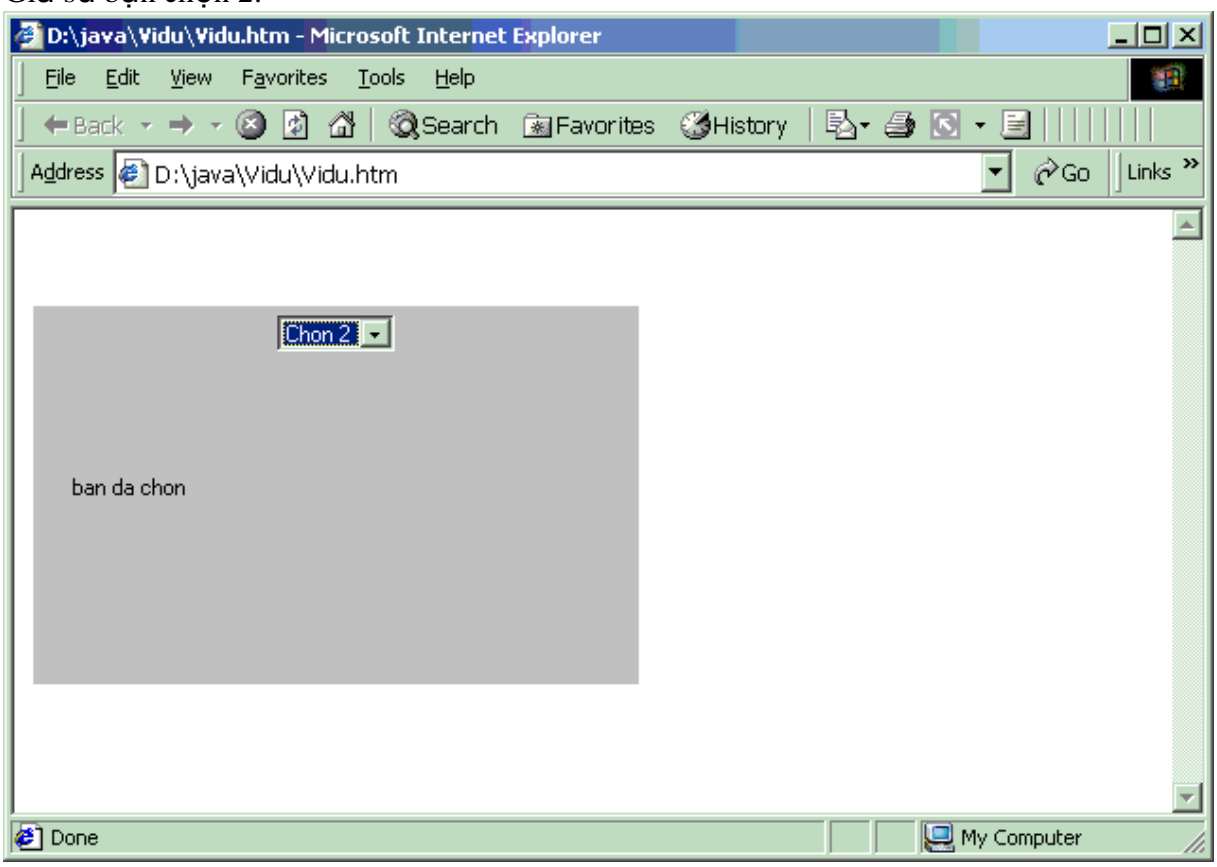
Khi chạy ta có kết quả:



Nếu bạn kích vào mũi tên:



Giả sử bạn chọn 2:



3.7 Điều khiển List

Điều khiển List tạo một danh sách có nhiều lựa chọn. Constructor của nó có hai dạng.

Dạng 1:

List Li=new List();

Dạng 2:

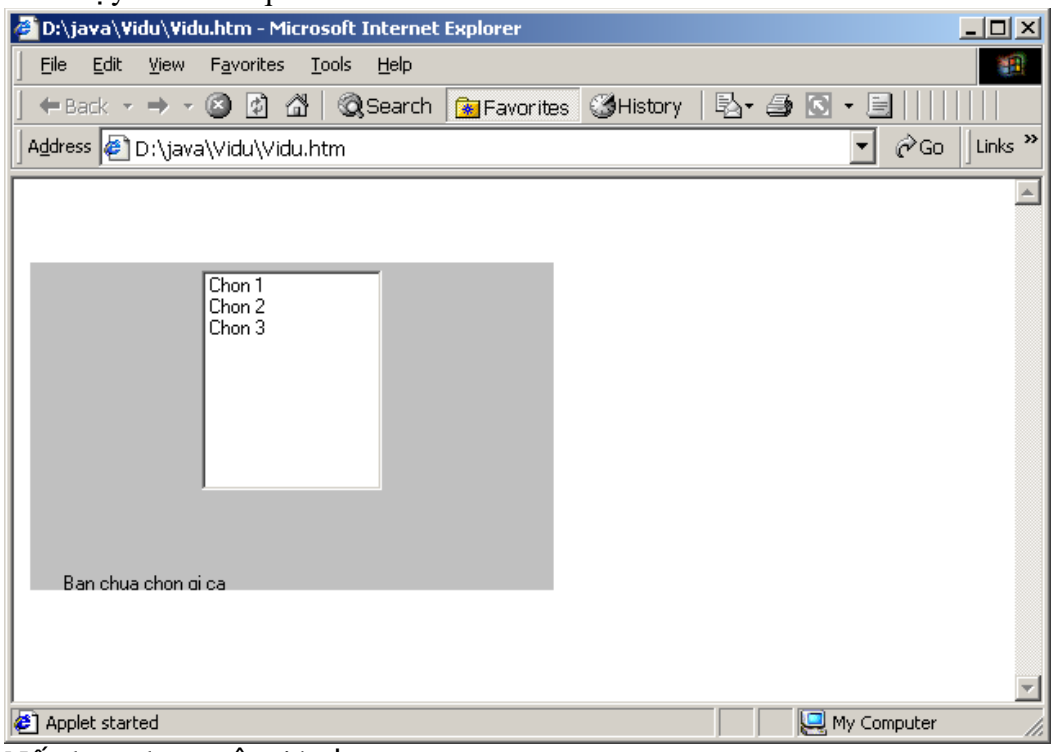
List Li=new List(n,true);

Thông số n chỉ số dòng, thông số thứ hai cho biết có thể chọn nhiều dòng trên List hay không (true chọn nhiều dòng, false chọn 1 dòng).

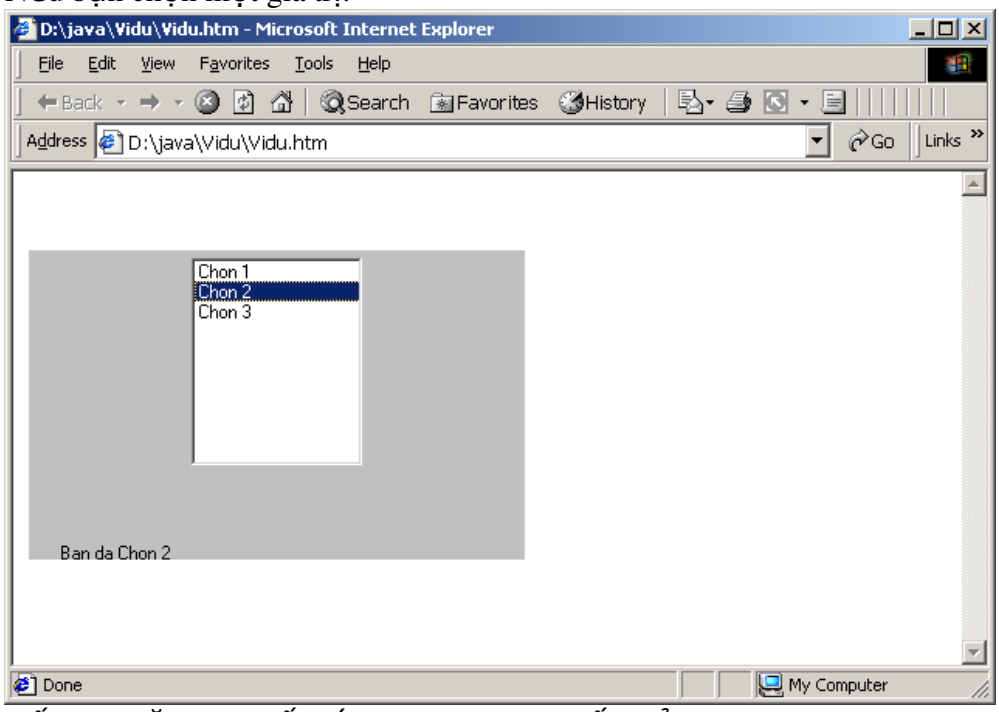
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Vidu extends Applet
  {
    String str="Ban chua chon gi ca";
    MyList M_List=new MyList(10,false);
public void init()
{
  M_List.add("Chon 1");
  M_List.add("Chon 2");
  M_List.add("Chon 3");
   add(M_List);
}
public void paint(Graphics g)
   {
    g.drawString(str,20,200);
   }
public class MyList extends List
   {
     MyList()
         {
            super();
            enableEvents(AWTEvent.ITEM_EVENT_MASK);
         }
          MyList(int nLines, boolean bMultiselect)
                   {
                  super(nLines,bMultiselect);
                   enableEvents(AWTEvent.ITEM_EVENT_MASK);
                   }
    public void processItemEvent(ItemEvent ie)
          {
                int nStateChange=ie.getStateChange();
                if (nStateChange==ItemEvent.SELECTED)
```

```
                {
                    str= "Ban da "+getSelectedItem() ;
                    getParent().repaint();
                }
            }
        }
    }
```

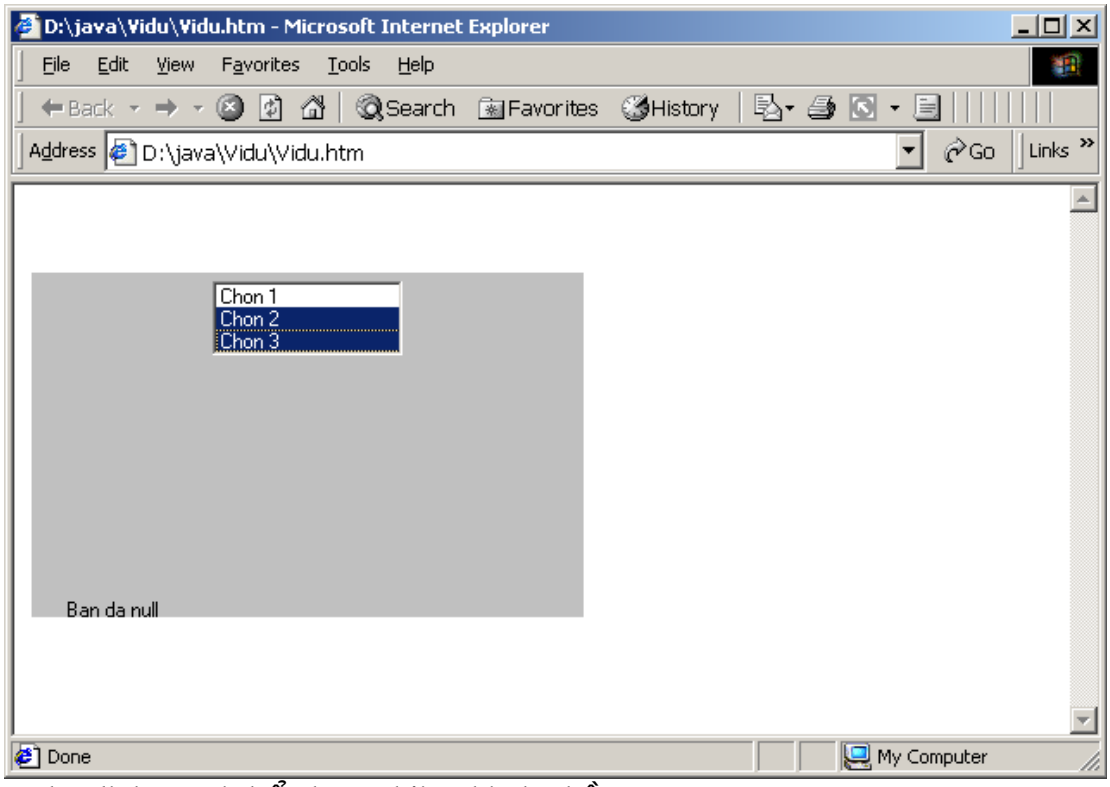
Khi chạy ta có kết quả:



Nếu bạn chọn một giá trị:



Nếu bạn đặt thông số thứ hai là true, ta có kết quả sau khi chạy như sau:



Nghĩa là bạn có thể chọn nhiều thành phần.

3.8 Điều khiển TextField

Điều khiển TextField cho phép người dùng nhập thông tin vào một khung (Field). Điều khiển này có thể được tạo ra rỗng hay khởi đầu bằng một chuỗi. Điều khiển này có thể được định nghĩa để có một số cột khởi đầu. Điều khiển TextField có 4 constructor.

Constructor thứ nhất có dạng:

TextField tf=new TextField()

Constructor thứ hai có thể xác định số cột cho trường văn bản, ví dụ:

TextField tf=new TextField(25)

Constructor thứ 3 nhận một đối số:

TextField tf=new TextField(“Chao ban”)

Constructor thứ 4 có dạng:

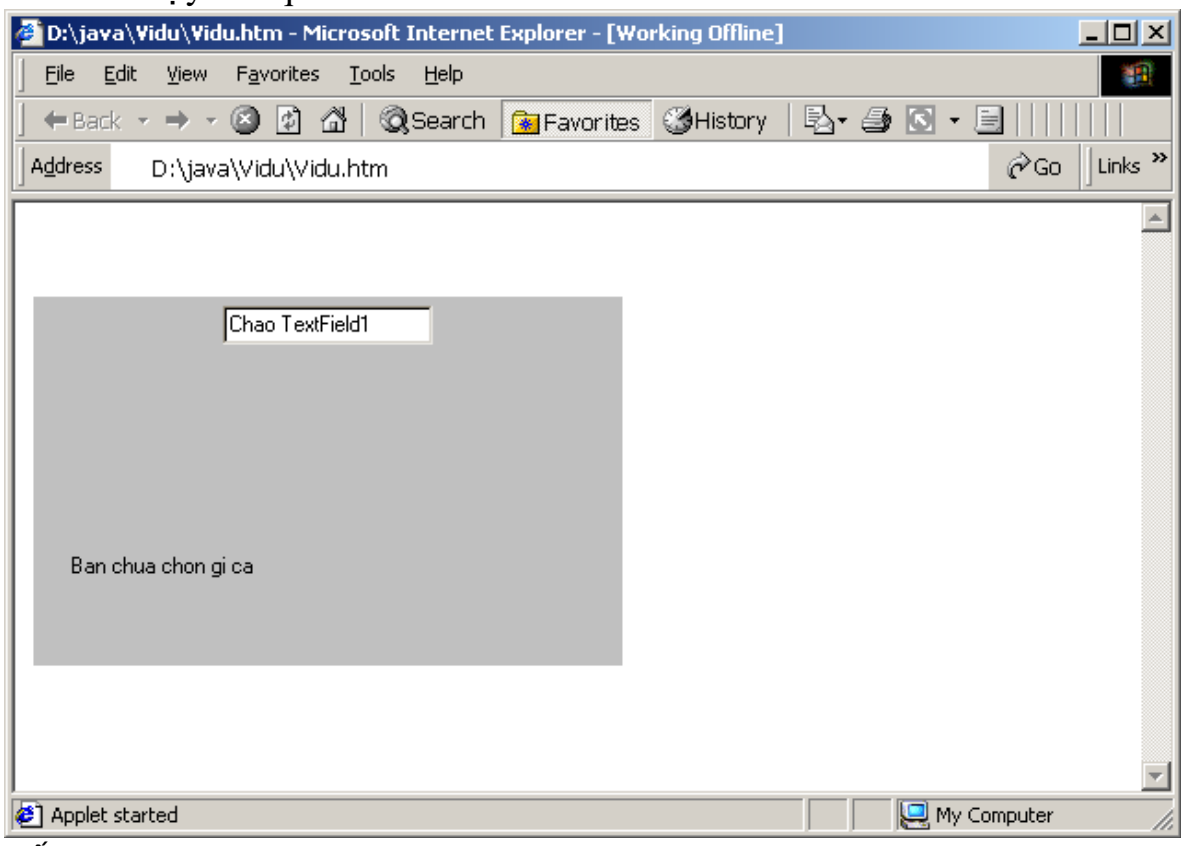
TextField tf=new TextField(“Chao ban”,25)

Tức là vừa có đối số vừa có số cột. Bạn xem đoạn mã sau:

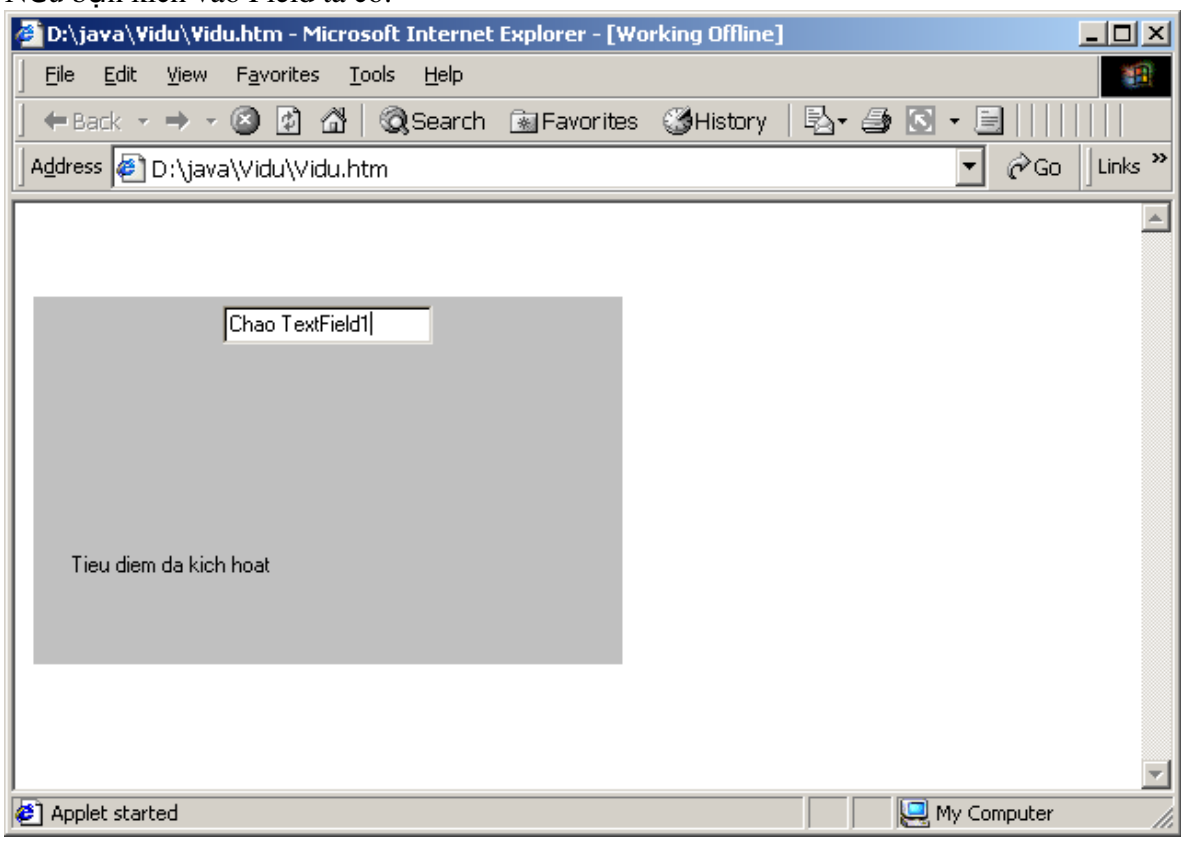
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Vidu extends Applet
  {
    String str="Ban chua chon gi ca";
    MyTextField tf=new MyTextField("Chao TextField1");
public void init()
```

```
  {
   add(tf);
  }
public void paint(Graphics g)
     {
      g.drawString(str,20,150);
     }
public class MyTextField extends TextField
 {
  MyTextField(String s)
   {
    super(s);
    enableEvents(AWTEvent.ACTION_EVENT_MASK |
AWTEvent.FOCUS_EVENT_MASK | AWTEvent.KEY_EVENT_MASK);
   }
public void processKeyEvent(KeyEvent ke)
          {
            if (ke.getID()==ke.KEY_RELEASED)
                    {
                          str= "Ban da gõ: "+ke.getKeyChar() ;
                          getParent().repaint();
                    }
         super.processKeyEvent(ke);
}
public void processActionEvent(ActionEvent ae)
    {
           str= "Ban da kich hoat" ;
           getParent().repaint();
          super.processActionEvent(ae);
    }

public void processFocusEvent(FocusEvent fe)
{
         if (fe.getID()==fe.FOCUS_GAINED)
         {
                  str="Tieu diem da kich hoat";
                  getParent().repaint();
         }
         else if (fe.getID()==fe.FOCUS_LOST)
         {
                  str="Tieu diem da mat";
                  getParent().repaint();
         }
         super.processFocusEvent(fe);
      }
   }
}
```

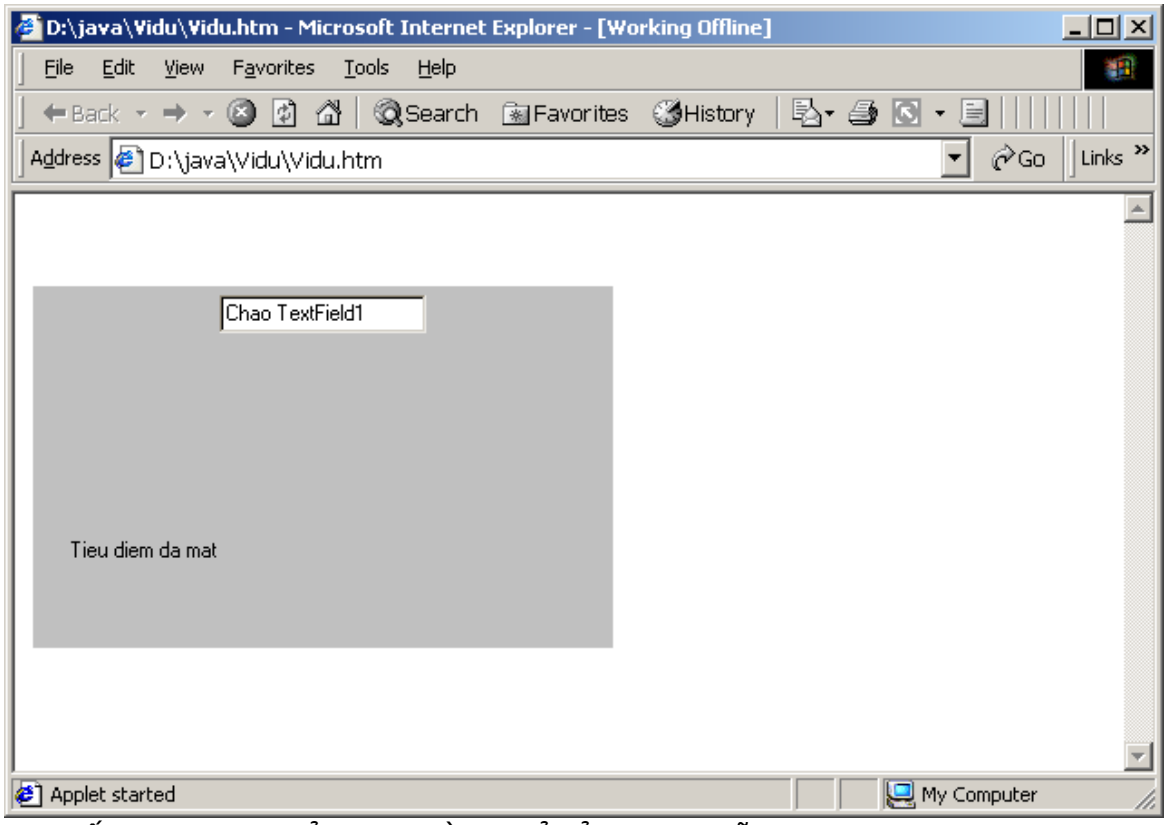
Sau khi chạy kết quả là:



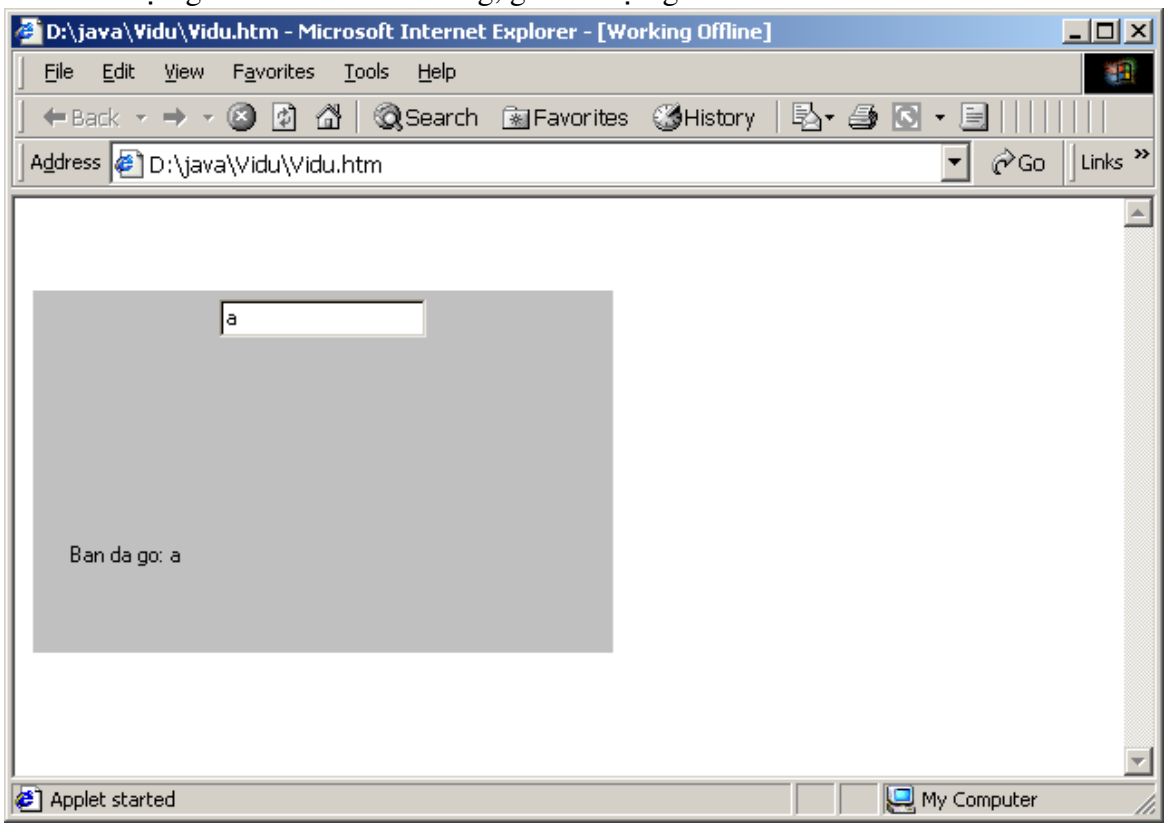
Nếu bạn kích vào Field ta có:



Còn nếu bạn kích vào phần ngoài Field thì:



Còn nếu bạn gõ văm bản vào trường, giả sử bạn gõ chữ a:



3.9 Điều khiển TextArea

TextArea có thể tạo ra với số lượng các dòng và cột. Có 5 Consstructor.

Constructor thứ nhất không chứa đối số:

TextArea ta=new TextArea()

Constructor thứ hai có chứa một đối số:

TextArea ta=new TextArea("Chao ban")

Constructor thứ ba có chứa hai đối số dòng và cột:

TextArea ta=new TextArea(10,20)

Constructor thứ ba có chứa ba đối số:

TextArea ta=new TextArea("Chao ban",10,20)

Constructor thứ năm có chứa bốn đối số:

TextArea ta=new TextArea("Chao ",10,20,
TextArea.SCROLLBARS_BOTH)

Có 4 thông số về thanh cuộn:

SCROLLBARS_BOTH: Hai thanh cuốn

SCROLLBARS_HORIZONTAL_ONLY: Thanh cuốn ngang

SCROLLBARS_VERTICAL_ONLY: Thanh cuốn dọc

SCROLLBARS_NONE: Không thanh cuốn

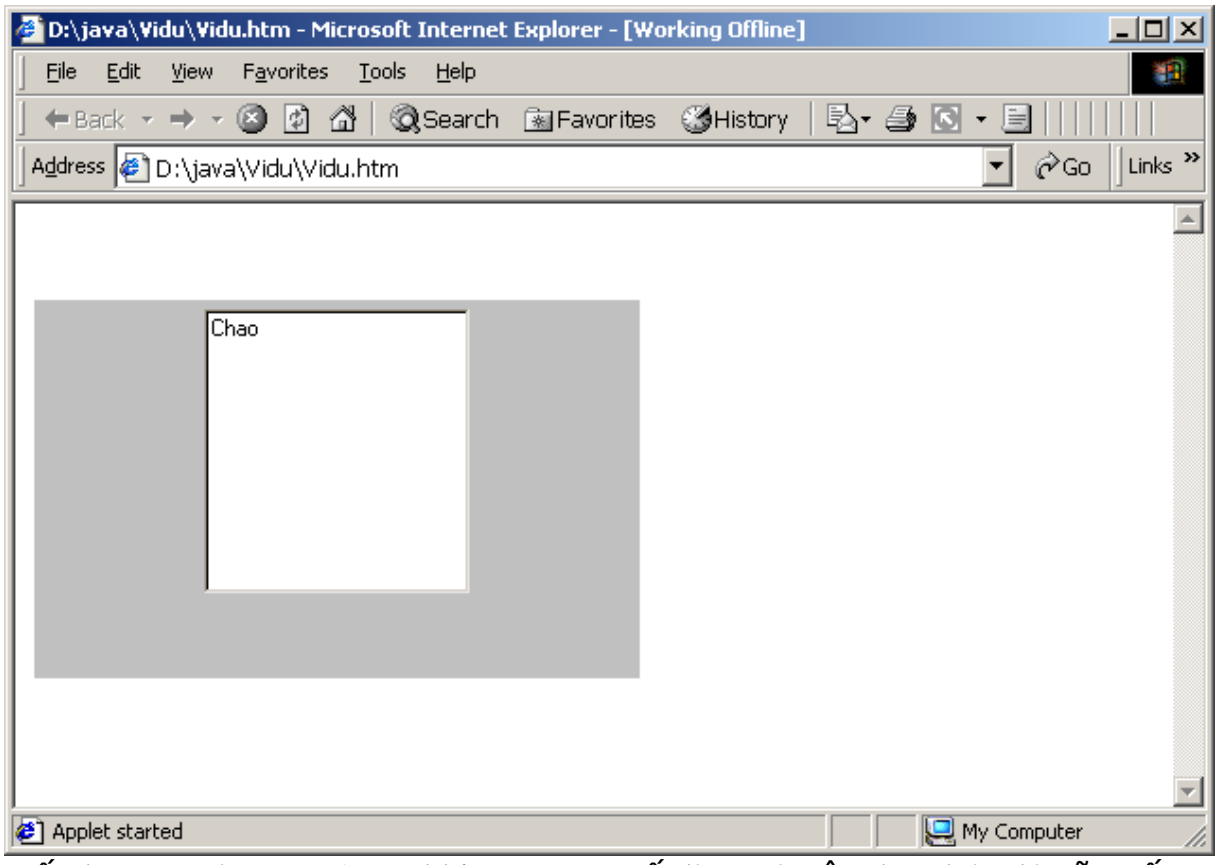
Có các sự kiện bàn phím hoạt động như sự kiện TextField.

Bạn quan sát đoạn mã sau:

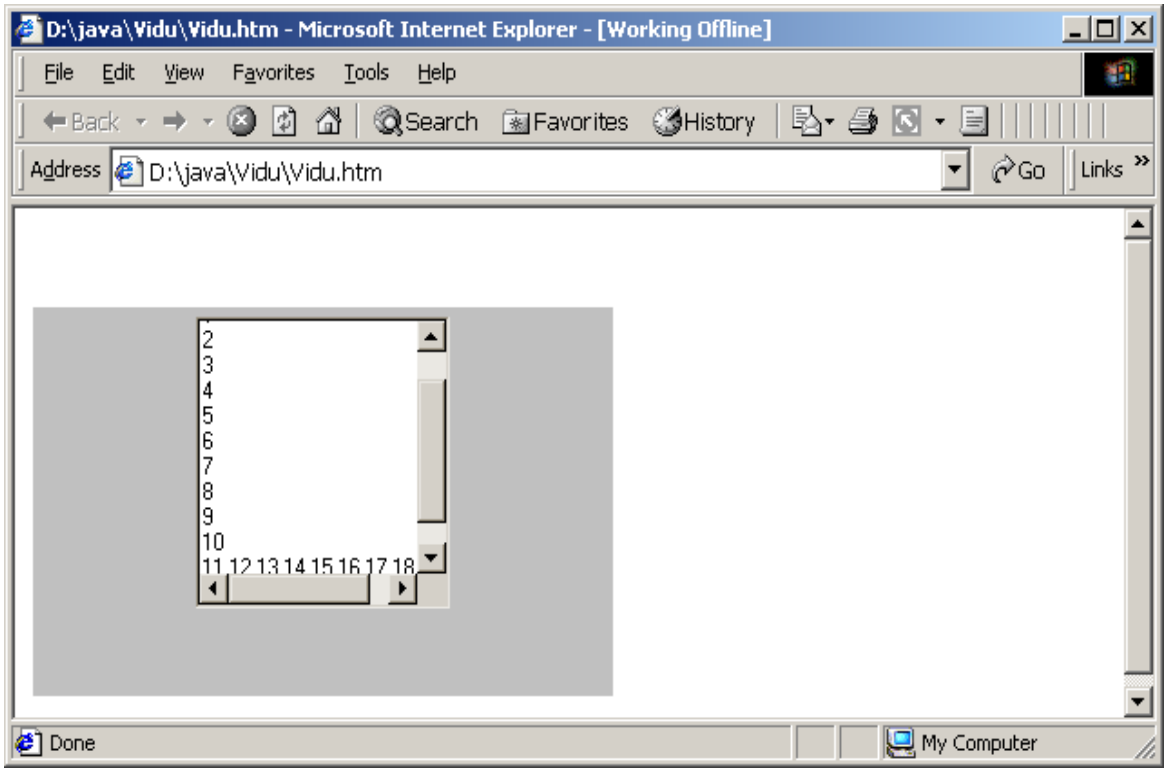
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Vidu extends Applet
  {
    String str="";
    TextArea ta=new
TextArea("Chao",10,20,TextArea.SCROLLBARS_BOTH);
public void init()
 {

  add(ta);
 }
public void paint(Graphics g)
   {
    g.drawString(str,11,30);
    }
```

}

Khi chạy ta có kết quả:



Nếu bạn gõ vào TextArea, khi vượt qua số dòng và cột cho phép thì sẽ xuất hiện thanh cuộn như sau:



3.10 Điều khiển Label

Label thường được sử dụng để đặt một văn bản mang tên của một điều khiển khác. Label có 3 constructor:

Constructor thứ nhất:

Label label1=new Label();

Constructor thứ hai:

Label label1=new Label("Chao Label");

Constructor thứ ba:

Label label1=new Label("Chao Label", Label.LEFT);- căn trái.

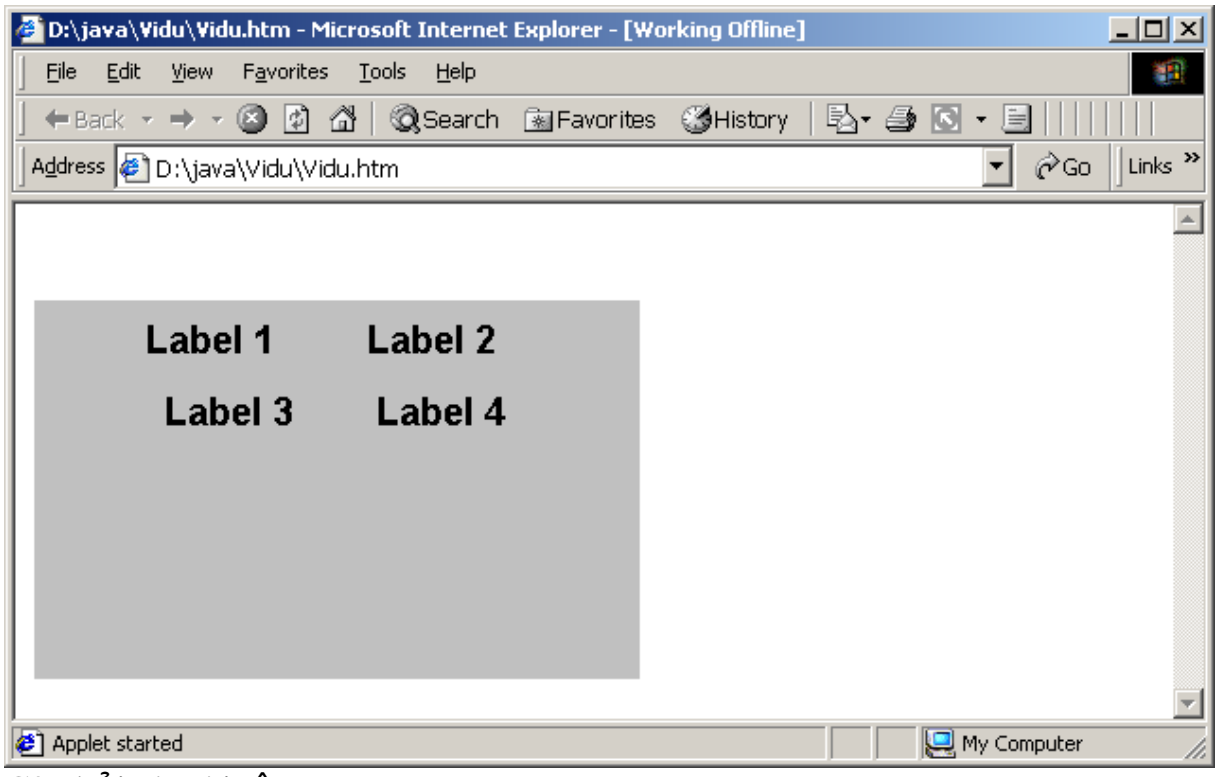
Rất ít khi bạn phải kiểm soát sự kiện nhãn.

Bạn quan sát đoạn mã sau:

```
import java.applet.*;
import java.awt.*;
public class Vidu extends Applet
{
public void init()
  {
  Label label1=new Label("  Nhan1   ");
  Label label2=new Label("  Nhan2   ", Label.LEFT);
  Label label3=new Label("  Nhan3   ", Label.RIGHT);
```

```
 Label label4=new Label("  Nhan 4   ", Label.CENTER);
  Font font=new Font(".vnTime", Font.BOLD,20);
  label1.setFont(font);
  label2.setFont(font);
  label3.setFont(font);
  label4.setFont(font);
  add(label1);
  add(label2);
  add(label3);
  add(label4);
 }
}
```

Khi chạy ta có kết quả:



Câu hỏi và Bài tập

1. Điều khiển Button có bao nhiêu constructor
2. Hãy nêu một phương thức kiểm tra sự kiện Button
3. Điều khiển Checkbox có những constructor nào?
4. Hãy nêu những phương thức kiểm tra sự kiện Checkbox

5. Điều khiển Choice có những constructor nào?
6. Hãy nêu những phương thức kiểm tra sự kiện Choice
7. Muốn CheckBox trở thành Radio thì phải làm thế nào?
8. Nếu bạn không sử dụng phương thức add() thì điều gì sẽ xẩy ra
9. Hãy tạo một Applet có chứa hai điều khiển Button và Checkbox
10. Hãy tạo một Applet có chứa hai điều khiển Button và TextArea
11. Hãy tạo một trang HTML trong đó có thẻ <Applet> chứa nội dung lời quảng cáo (tuỳ ý bạn) vào TextArea.