

Chương 6

APPLETS

Sau khi học xong chương này, bạn có thể nắm được các nội dung sau:

- Hiểu được các Applet của Java
- Phân biệt applet và các ứng dụng application
- Tìm hiểu chu trình sống của một applet
- Tạo các applet
- Hiển thị các hình ảnh sử dụng applet
- Truyền tham số cho applet
- Tìm hiểu ứng dụng của applet trong GUI

6.1 Java Applet

Applet là một chương trình Java có thể chạy trong trình duyệt web. Tất cả các applet đều là các lớp con của lớp 'Applet'.

Lớp Applet thuộc package 'java.applet'. Lớp Applet bao gồm nhiều phương thức để điều khiển quá trình thực thi của applet. Để tạo applet, bạn cần import hai gói sau:

- java.applet
- java.awt

6.2 Cấu trúc của một Applet

Một Applet định nghĩa cấu trúc của nó từ 4 sự kiện xảy ra trong suốt quá trình thực thi. Đối với mỗi sự kiện, một phương thức được gọi một cách tự động. Các phương thức này được minh họa trong bảng 6.1

Điều quan trọng là không phải lúc nào applet cũng bắt đầu từ ban đầu. Mà nó bắt đầu từ vị trí tiếp theo của quá trình thực thi trước đó.

Ngoài những phương thức cơ bản này, còn có những phương thức 'paint()' và 'repaint()'. Phương thức paint() dùng để hiển thị một đường thẳng (line), text, hoặc một hình ảnh trên màn hình. Đối số của phương thức này là đối tượng của lớp Graphics. Lớp này thuộc gói java.awt. Câu lệnh sau được dùng để import lớp Graphics:

import java.awt.Graphics;

Phương thức	Chức năng
init()	Được gọi trong quá trình khởi tạo applet. Trong quá trình khởi tạo, nó sẽ tạo đối tượng để cung cấp cho applet. Phương thức này được dùng để tải các hình ảnh đồ họa, khởi tạo các biến và tạo các đối tượng.
start()	Được gọi khi một applet bắt đầu thực thi. Một khi quá trình khởi tạo hoàn tất, thì applet được khởi động. Phương thức này được dùng để khởi động lại applet sau khi nó đã ngừng trước đó
stop()	Được gọi khi ngừng thực thi một applet. Một applet bị ngừng trước khi nó bị huỷ.
destroy()	Được dùng để huỷ một applet. Khi một applet bị huỷ, thì bộ

	nhớ, thời gian thực thi của vi xử lý, không gian đĩa được trả về cho hệ thống.
--	--

Bảng 6.1: Các phương thức của một applet

Phương thức `repaint()` được dùng khi cửa sổ cần cập nhật lại. Phương thức này chỉ cần một thông số. Tham số này là đối tượng của lớp Graphics.

Applet sử dụng phương thức `showStatus()` để hiển thị thông tin trên thanh trạng thái. Phương thức có tham số thuộc kiểu dữ liệu String. Để lấy các thông tin của applet, user có thể override phương thức `getAppletInfo()` của lớp Applet. Phương thức này trả về 1 đối tượng kiểu String.

Các phương thức của applet `init()`, `start()`, `stop()`, `destroy()`, và `paint()` được thừa kế từ một applet. Nhưng mặc định những phương thức này không thực thi một thao tác nào cả. Đây là ví dụ đơn giản của applet. Câu lệnh sau tạo một lớp có tên là `Applet1`, lớp này sẽ kế thừa tất cả các phương thức và biến của lớp `Applet`.

public class Applet1 extends Applet

Phương thức `init()` và `paint()` thường được dùng để thực hiện một số hàm để khởi tạo và vẽ applet. Phương thức `g.drawString()` chỉ ra vị trí mà đoạn văn bản được vẽ ở đâu trên màn hình.

Chương trình 6.1 hiển thị một chuỗi ở dòng 70 và cột 80:

Chương trình 6.1

```
import java.awt.*;
import java.applet.*;
public class Applet1 extends Applet
{
    int num;
    public void init()
    {
        num = 6;
    }
    public void paint (Graphics g)
    {
        g.drawString ("Hello to Applet. Chapter " + num, 70, 80);
        showStatus (getAppletInfo());
        //Hiển thị một chuỗi được trả về từ hàm getAppletInfo() trên thanh trạng thái
    }
    public String getAppletInfo() //user overrides
    {
        return "Created by Aptech";
    }
}
```

Sử dụng cú pháp sau để dịch một Applet:

javac Applet1.java

Để thực thi một applet, ta cần tạo một file HTML. File HTML này sử dụng thẻ applet. Thẻ applet này lấy tham số đầu tiên là đường dẫn của file applet.

Thẻ applet có hai thuộc tính sau:

- Width
- Height

Để truyền tham số vào applet, sử dụng param, sau đó là thẻ value. Sau đây là ví dụ của thẻ applet:

```
<applet code=Applet1 width=300 height=200>  
</applet>
```

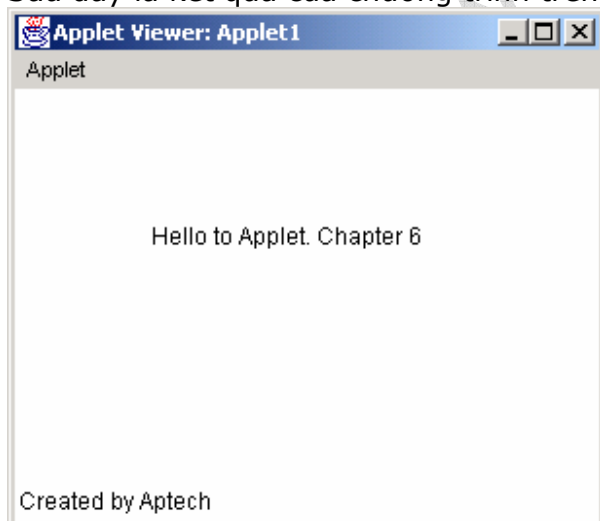
Lúc này, ta có thể thực thi applet này bằng cách dùng trình xem applet. Đây là công cụ của JDK. Để chạy file HTML trong trình xem applet, ta gõ câu lệnh sau:

Appletviewer abc.html // 'abc.html' là tên của file HTML

Một tùy chọn khác của applet là ta thêm thẻ applet như là một dòng chú thích trong đoạn code. Lúc đó, applet được dịch, và thực thi bằng cách sử dụng lệnh sau:

Appletviewer Applet1.java

Sau đây là kết quả của chương trình trên:



Hình 6.1 Applet

6.2.1 Sự khác nhau giữa Application và Applet

Sau đây là sự khác nhau giữa application và applet:

- Để thực thi các application chúng ta dùng trình thông dịch Java, trong khi đó applet có thể chạy được trên các trình duyệt (có hỗ trợ Java) hay sử dụng công cụ AppletViewer, công cụ này đi kèm với JDK.
- Quá trình thực thi của application bắt đầu từ phương thức 'main()'. Tuy nhiên applet thì không làm như vậy.
- Các application sử dụng 'System.out.println()' để hiển thị kết quả ra màn hình trong khi đó applet sử dụng phương thức 'drawstring()' để xuất ra màn hình.

Một điều đáng lưu ý là một chương trình Java đơn lẻ thì có thể vừa là application vừa là applet. Chức năng của applet được bỏ qua khi nó được thực thi như là một application và ngược lại.

Chương trình 6.2 sẽ minh họa điều này

Chương trình 6.2

```
import java.applet.Applet;
import java.awt.*;
/*
<applet code = "both" width = 200 height = 100>
</applet>
*/

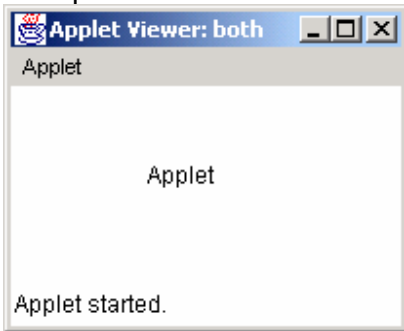
public class both extends Applet
{
    Button btn;
    public void init()
    {
        btn = new Button ("Click");
    }

    public void paint (Graphics g)
    {
        g.drawString ("Applet", 70, 50);
    }
    public static void main (String args[])
    {
        both app = new both();
        app.init();
        System.out.println("Application Main");
    }
}
```

Sau khi biên dịch chương trình, nó có thể được thực thi như là một applet bằng cách sử dụng cú pháp sau:

appletviewer both.java

Kết quả như sau:

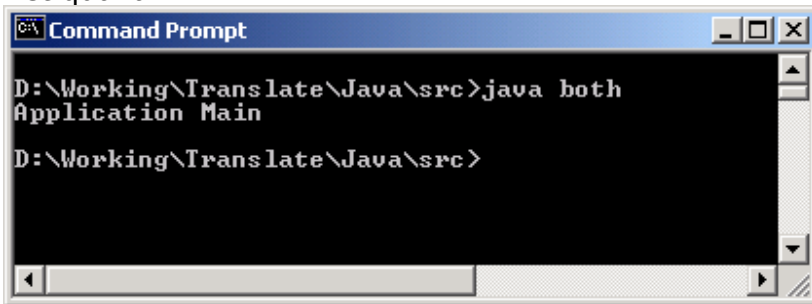


Hình 6.2 Applet

Nếu chạy chương trình trên như một application, thì sử dụng cú pháp sau:

java both

Kết quả là:



Hình 6.3 Application

Khi applet chạy trên trình duyệt web, đặc điểm này thực sự hữu ích khi bạn muốn tải applet trong một frame mới. Ví dụ: trong applet được tạo để chat, một số website sử dụng một cửa sổ chat riêng biệt để chat. Bạn cũng có thể kết hợp các đặc điểm của frame và applet vào trong một chương trình.

6.2.2 Những giới hạn bảo mật trên applet

Có một số hạn chế mà applet không thể làm được. Bởi vì các applet của Java có thể phá hỏng toàn bộ hệ thống của user. Các lập trình viên Java có thể viết các applet để xóa file, lấy các thông tin các nhân của hệ thống...

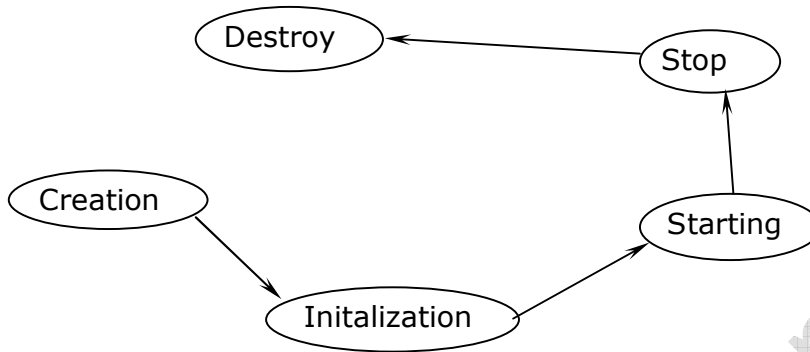
Vì thế, các applet của java không thể làm các việc sau:

- Không thể đọc hoặc ghi file trên hệ thống file của user.
- Không thể giao tiếp với các site internet, nhưng chỉ có thể với các trang web có applet mà thôi.
- Không thể chạy bất cứ chương trình gì trên hệ thống của người đọc.
- Không thể tải bất cứ chương trình được lưu trữ trong hệ thống của user.

Những giới hạn trên chỉ đúng khi các applet được chạy trên trình duyệt Netscape Navigator hoặc Microsoft Internet Explorer.

6.3 Chu trình sống của một Applet

Chu trình sống của một Applet được mô tả ở sơ đồ dưới đây:



Hình 6.4 Chu trình sống của một applet

Trước tiên, applet được tạo.

Bước kế tiếp là khởi tạo. Điều này xảy ra khi một applet được nạp. Quá trình này bao gồm việc tạo các đối tượng mà applet cần. Phương thức `init()` được override để cung cấp các hành vi để khởi tạo.

Một khi applet được khởi tạo, applet sẽ được khởi động. Applet có thể khởi động ngay cả khi nó đã được ngừng trước đó. Ví dụ, nếu trình duyệt nhảy đến một liên kết nào đó ở trang khác, lúc đó applet sẽ bị ngừng, và được khởi động trở lại khi user quay về trang đó.

Sự khác nhau giữa quá trình khởi tạo và quá trình khởi động là một applet có thể khởi động nhiều lần, nhưng quá trình khởi tạo thì chỉ xảy ra một lần.

Phương thức `'start()'` được override để cung cấp các thao tác khởi động cho applet.

Phương thức `'stop()'` chỉ được gọi khi user không còn ở trang đó nữa, hoặc trang đó đã được thu nhỏ lại ở dưới thanh taskbar.

Kế tiếp là phương thức `'destroy()'`. Phương thức này giúp applet dọn dẹp trước khi nó được giải phóng khỏi vùng nhớ, hoặc trước khi duyệt duyệt kết thúc. Phương thức này được dùng để hủy những luồng (thread) hay quá trình đang chạy.

Phương thức `'destroy()'` khác với phương thức `finalize()` là phương thức `destroy()` chỉ dùng cho applet, trong khi `finalize()` là cách tổng quát để dọn dẹp applet.

Phương thức `paint()` cũng là một phương thức quan trọng khác. Phương thức này cho phép ta hiển thị một cái gì đó trên màn hình. Có thể là text, đường thẳng, màu nền, hoặc hình ảnh. Phương thức này xảy ra nhiều lần trong suốt quá trình applet tồn tại. Phương thức này thực thi một lần sau khi applet được khởi tạo. Nó sẽ lặp đi lặp lại khi di chuyển từ cửa sổ trình duyệt sang cửa sổ khác. Nó cũng xảy ra khi cửa sổ trình duyệt thay đổi vị trí của nó trên màn hình.

Phương thức 'paint()' có một tham số. Tham số này là đối tượng của lớp Graphics. Lớp Graphics thuộc lớp java.awt, chúng ta phải import trong đoạn code của applet. Chúng ta có thể sử dụng đoạn mã sau:

```
import java.awt.Graphics;
```

6.4 Truyền tham số cho Applet

Trong chương trình sau, chúng ta sẽ truyền tham số cho applet. Thành phần nút 'bNext' có tên được truyền như là một tham số. Phương thức 'init()' sẽ kiểm tra tham số có tên là 'mybutton'. Sau đó, nó tạo một nút với chuỗi đó như là tên của nút. Nếu không có tham số truyền vào, nút đó có tên mặc định là 'Default'.

Bây giờ chúng ta định nghĩa thẻ <PARAM> trong đoạn mã HTML như sau:

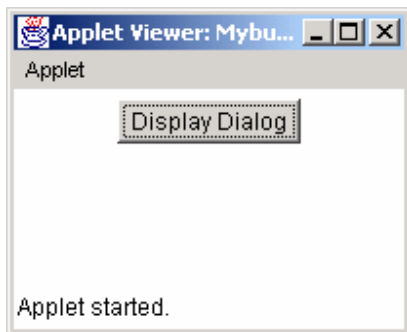
```
/*
<applet code="Mybutton1" width="100" height="100">
<PARAM NAME="mybutton" value="Display Dialog">
</applet>
*/
```

Chương trình 6.3

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Mybutton1" width="200" height="100">
<PARAM NAME="mybutton" value="Display Dialog">
</applet>
*/

public class Mybutton1 extends Applet
{
    Button bNext;
    public void init()
    {
        /*getParameter returns the value of the specified parameter in the form of a String
        object*/
        String str = getParameter("mybutton");
        //when no parameter is passed
        if (str==null)
            str = new String ("Default");
        //when parameter is passed
        bNext = new Button(str);
        add (bNext);
    }
}
```

Sau đây là kết quả của chương trình trên:



Hình 6.5: truyền tham số cho applet

Bây giờ chúng ta sẽ sử dụng lớp Graphics để vẽ các hình chẳng hạn như: đường thẳng, hình oval, và hình chữ nhật. Chúng ta sẽ học lớp Font trong các phần sau. Lớp này có thể dùng để in văn bản bằng bất cứ font nào.

6.5 Lớp Graphics

Java cung cấp gói AWT cho phép ta vẽ các hình đồ họa. Lớp Graphics bao gồm tập hợp rất nhiều phương thức. Nhưng phương thức này được sử dụng để vẽ bất cứ hình nào trong các hình sau:

- Oval
- Rectangle
- Square
- Circle
- Lines
- Text

Bạn có thể vẽ những hình này bằng bất cứ màu nào. Frame, Applet và canvas là các môi trường để hiển thị đồ họa.

Để vẽ bất cứ hình ảnh nào chúng ta cần phải có nền đồ họa (Graphical Background). Để có được một nền đồ họa, chúng ta gọi phương thức 'getGraphics()' hay bất cứ phương thức nào trong các phương thức sau đây:

- repaint()

Được gọi khi cần vẽ lại những đối tượng đã vẽ.

- update(Graphics g)

Được gọi một cách tự động bởi phương thức 'repaint()'

Phương thức này sẽ xóa những đối tượng đã vẽ, và truyền nó cho đối tượng của lớp Graphics để gọi phương thức 'paint()';

- paint(Graphics g)

Được gọi bởi phương thức update().

Đối tượng được truyền cho phương thức này được dùng để vẽ. Phương thức này dùng để

vẽ các hình ảnh đồ hoạ khác nhau.

Việc gọi phương thức `paint()` lặp đi lặp lại thông qua phương thức `repaint()` sẽ xoá đi các hình đã vẽ trước đó. Để vẽ các hình mới mà vẫn giữ lại những hình đã vẽ trước đó, chúng ta cần override lại phương thức `update()`.

```
Public void update (Graphics g)
{
    paint (g);
}
```

Ở đây, phương thức `update()` sẽ không xoá những đối tượng đã vẽ, nhưng chỉ gọi phương thức `paint()`. Để làm được điều này, nó truyền đối tượng của lớp `Graphics` hoặc `GraphicsContext` cho phương thức `paint()`. Ở đây, đối tượng của lớp `Graphics` là `'g'`.

6.5.1 Vẽ các chuỗi, các ký tự và các byte

Chương trình sau minh hoạ các vẽ các chuỗi, ký tự và các byte.

Để vẽ hoặc in một chuỗi, lớp `Graphics` cung cấp phương thức `'drawString()'`. Cú pháp như sau:

```
DrawString (String str, int xCoor, int yCoor);
```

Ba tham số là:

- Chuỗi cần vẽ.
- Toạ độ X trên frame, nơi chuỗi cần được vẽ.
- Toạ độ Y trên frame, nơi chuỗi cần được vẽ.

Để vẽ hoặc xuất các ký tự trên frame, lớp `Graphics` cung cấp phương thức `'drawChars'`. Cú pháp như sau:

```
DrawChars (char array[], int offset, int length, int xCoor, int yCoor);
```

Chú thích các tham số:

- Mảng các ký tự.
- Vị trí bắt đầu, nơi các ký tự được vẽ.
- Số các ký tự cần được vẽ.
- Toạ độ X, nơi các ký tự cần được vẽ.
- Toạ độ Y, nơi các ký tự cần được vẽ.

Lớp `Graphics` cung cấp phương thức `'drawBytes()'` để vẽ hoặc in các byte ra frame. Cú pháp của phương thức này như sau:

```
DrawBytes (byte array[], int offset, int length, int xCoor, int yCoor);
```

5 tham số của phương thức trên là:

- Mảng các byte.

- Vị trí offset hay vị trí bắt đầu.
- Số byte cần vẽ.
- Toạ độ X.
- Toạ độ Y.

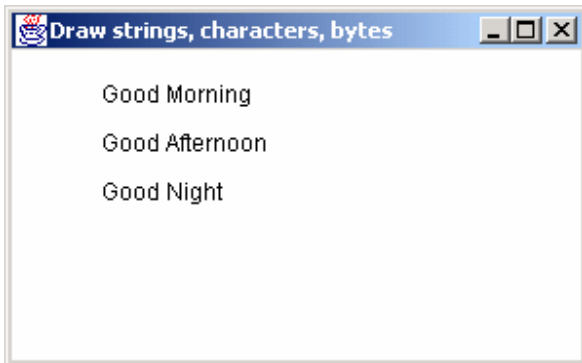
Đối với một ký tự hoặc một mảng các byte, chúng ta có thể in một phần của mảng mà thôi. Ở đây, toạ độ x và y là toạ độ tính theo dòng. Chương trình 6.4 minh hoạ cách vẽ chuỗi, các ký tự và các byte.

Chương trình 6.4

```
import java.awt.*;
public class DrawStrings extends Frame
{
    public DrawStrings()
    {
        super ("Draw strings, characters, bytes");
        setSize (300, 300);
        setVisible (true);
    }
    public void paint(Graphics g)
    {
        g.drawString ("Good Morning", 50, 50);
        g.drawString ("Good Afternoon", 50, 75);
        g.drawString ("Good Night", 50, 100);
        char ch[] = {};
    }
    public static void main (String args[])
    {
        new DrawStrings();
    }
}
```

Chương trình trên vẽ chuỗi, ký tự từ một mảng ký tự, và vẽ các byte từ mảng các byte. Bạn phải import gói java.awt để sử dụng các phương thức đồ hoạ có sẵn trong gói này. Ta phải làm điều này vì lớp Graphics nằm trong gói này.

Sau đây là kết quả của chương trình trên:



Hình 6.6 Strings, characters và bytes

6.5.2 Vẽ đường thẳng (Line) và Oval

Sau đây là cú pháp của các phương thức được sử dụng để vẽ đường thẳng và hình oval:

- `drawLine (int x1, int y1, int x2, int y2);`
- `drawOval (int xCoor, int yCoor, int width, int height);`
- `setColor (Color c);`
- `fillOval (int xCoor, int yCoor, int width, int height);`

Phương thức `'drawLine()'` nhận các tham số sau:

- Toạ độ X, nơi bắt đầu vẽ (`x1`).
- Toạ độ Y, nơi bắt đầu vẽ (`y1`).
- Toạ độ X, nơi kết thúc vẽ (`x2`).
- Toạ độ Y, nơi kết thúc vẽ (`y2`).

Phương thức này bắt đầu vẽ tại toạ độ `'x1'` và `'y1'`, và kết thúc tại toạ độ `'x2'` và `'y2'`. Để vẽ những đường thẳng có màu, chúng ta thiết lập một màu nào đó. Phương thức `'setColor'` dùng để thiết lập màu cho hình ảnh đồ hoạ. Trong chương trình này, chúng ta sử dụng câu lệnh sau để chọn màu xanh:

`g.setColor (Color.blue);`

Phương thức `'drawOval()'` nhận 4 thông số sau:

- Toạ độ X.
- Toạ độ Y.
- Chiều rộng của hình Oval.
- Chiều cao của hình Oval.

Đối với hình oval rộng, thì giá trị của chiều rộng lớn hơn chiều cao, và ngược lại đối với hình oval cao.

Phương thức `'fillOval()'` nhận 4 thông số, nhưng nó sẽ tô hình oval. Sử dụng phương thức `setColor` để tô hình oval;

g.setColor(Color.cyan);

Ở đây, hình oval sẽ được tô với màu cyan. Lớp Color cung cấp các màu khác nhau mà hệ thống có hỗ trợ.

6.5.3 Vẽ hình chữ nhật (Rectangle) và hình chữ nhật bo góc (Rounded Rectangle)

Sau đây là cú pháp của các phương thức được dùng để vẽ hình chữ nhật và hình chữ nhật bo góc:

- drawRect (int xCoor, int yCoor, int width, int height);
- fillRect (int xCoor, int yCoor, int width, int height);
- drawRoundRect (int xCoor, int yCoor, int width, int height, int arcwidth, int archeight);
- fillRoundRect (int xCoor, int yCoor, int width, int height, int arcwidth, int archeight);

Phương thức 'drawRect()' được dùng để vẽ hình chữ nhật đơn giản. Phương thức này nhận 4 tham số sau:

- Toạ độ X
- Toạ độ Y
- Chiều rộng của hình chữ nhật
- Chiều cao của hình chữ nhật

Phương thức này vẽ hình chữ nhật có chiều rộng và chiều cao cho trước, bắt đầu tại toạ độ X, Y. Chúng ta có thể thiết lập màu của hình chữ nhật. Ở đây, chúng ta chọn màu đỏ. Câu lệnh sẽ như sau:

g.setColor (Color.red);

Phương thức 'drawRoundRect()' vẽ hình chữ nhật có các góc tròn. Phương thức này nhận 6 tham số, trong đó 4 tham số đầu thì giống với phương thức drawRect. Hai tham số khác là:

- arcwidth của hình chữ nhật
- archeight của hình chữ nhật

Ở đây, 'arcwidth' làm tròn góc trái và góc phải của hình chữ nhật. 'archeight' làm tròn góc trên đỉnh và góc đáy của hình chữ nhật. Ví dụ, arcwidth = 20 có nghĩa là hình chữ nhật được làm tròn cạnh trái và cạnh phải mỗi cạnh 10 pixel. Tương tự, archeight = 40 sẽ tạo ra hình chữ nhật được làm tròn từ đỉnh đến đáy 20 pixel.

Pixel là đơn vị đo. Nó là đơn vị nhỏ nhất trong vùng vẽ.

Để tô hay vẽ hình chữ nhật và hình chữ nhật bo góc, chúng ta sử dụng phương thức 'fillRect()' và 'fillRoundRect()'. Những phương thức này nhận các tham số giống với

phương thức `drawRect()` và `drawRoundRect()`. Những phương thức này vẽ các hình ảnh với một màu cho trước hoặc mới màu hiện hành. Lệnh sau dùng để vẽ hình với màu xanh:

`g.setColor(Color.green);`

6.5.4 Vẽ hình chữ nhật 3D và vẽ hình cung (Arc)

Sau đây là cú pháp của các phương thức dùng để vẽ hình chữ nhật 3D và hình cung:

- `draw3Drect (int xCoord, int yCoord, int width, int height, boolean raised);`
- `drawArc(int xCoord, int yCoord, int width, int height, int arcwidth, int archeight);`
- `fillArc(int xCoord, int yCoord, int width, int height, int arcwidth, int archeight);`

Phương thức `'draw3Drect()'` nhận 5 tham số. 4 tham số đầu thì tương tự với phương thức để vẽ hình chữ nhật. Tuy nhiên, giá trị của tham số thứ 5 quyết định là hình chữ nhật này có 3 chiều hay không. Tham số thứ 5 có kiểu dữ liệu là Boolean. Giá trị này True có nghĩa là hình chữ nhật là 3D.

Phương thức `'drawArc()'` nhận 6 tham số sau:

- Toạ độ x
- Toạ độ y
- Chiều rộng của cung được vẽ.
- Chiều cao của cung được vẽ.
- Góc bắt đầu.
- Độ rộng của cung so với góc ban đầu.

Phương thức `'fillArc()'` cũng nhận 6 tham số giống như phương thức `drawArc()`, nhưng nó vẽ cung và tô cung với màu hiện thời.

6.5.5 Vẽ hình PolyLine

Chương trình sau lấy các điểm từ hai mảng để vẽ một loạt các đường thẳng.

Cú pháp của phương thức này như sau:

- `drawPolyline (int xArray[], int yArray[], int totalPoints);`
- `g.setFont (new Font("Times Roman", Font.BOLD, 15));`

Phương thức `'drawPolyline()'` nhận 3 tham số sau:

- Mảng lưu trữ toạ độ x của các điểm.
- Mảng lưu trữ toạ độ y của các điểm.
- Tổng số điểm cần vẽ.

Để vẽ các đường thẳng ta lấy các điểm từ hai mảng như sau:

`(array1[0], array2[0]) (array1[1], array2[1]) (array1[2], array2[2])...`

Số đường thẳng vẽ được luôn nhỏ hơn số truyền vào thông số thứ 3 của phương thức drawPoyline(). Ví dụ như: totalPoints - 1

Chương trình 6.5 minh hoạ các vẽ polyline.

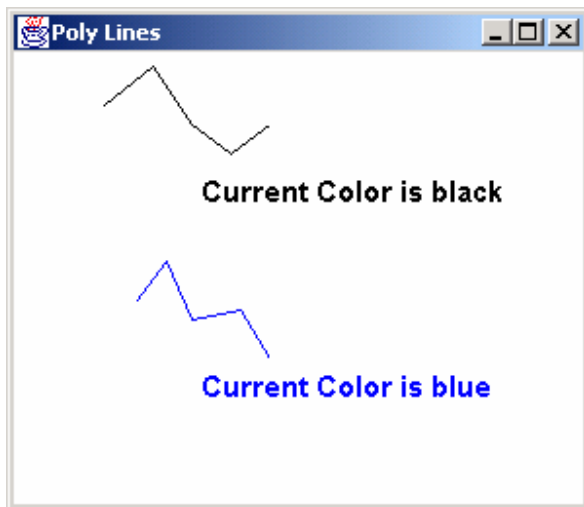
Chương trình 6.5

```
import java.awt.*;

class PolyLines extends Frame
{
    int x1[] = {50, 75, 95, 115, 135};
    int y1[] = {50, 30, 60, 75, 60};
    int x2[] = {67, 82, 95, 120, 135};
    int y2[] = {150, 130, 160, 155, 180};

    public PolyLines()//constructor
    {
        super ("Poly Lines");
        setSize (300, 300);
        setVisible (true);
    }
    public void paint (Graphics g)
    {
        g.drawPolyline (x1, y1, 5);
        g.setFont (new Font("Times Roman", Font.BOLD, 15));
        g.drawString("Current Color is black", 100, 100);
        g.setColor(Color.blue);
        g.drawPolyline (x2, y2, 5);
        g.drawString ("Current Color is blue", 100, 200);
    }
    public static void main (String args[])
    {
        new PolyLines();
    }
}
```

Kết quả của chương trình được minh hoạ ở hình 6.7



Hình 6.7

6.5.6 Vẽ và tô đa giác (Polygon)

Lớp Graphics cung cấp hai phương thức để vẽ đa giác. Phương thức đầu tiên nhận một đối tượng của lớp Polygon. Phương thức thứ 2 lấy hai mảng điểm, và tổng số điểm cần vẽ. Chúng ta sẽ sử dụng phương thức 2 để vẽ đa giác.

Cú pháp của **drawPolygon()** như sau:

drawPolygon(int x[], int y[], int numPoints);

Cú pháp của **fillPolygon()** như sau:

fillPolygon (int x[], int y[], int numPoints);

Chương trình dưới đây lấy các điểm từ 2 mảng để vẽ đa giác. Phương thức 'drawPolygon()' nhận 3 tham số sau giống như phương thức drawPolyline()

- Mảng lưu trữ tọa độ x của các điểm.
- Mảng lưu trữ tọa độ y của các điểm.
- Tổng số điểm cần vẽ.

Chương trình 6.6

```
import java.awt.*;
class PolyFigures extends Frame
{
    int x1[] = {50, 25, 40, 100, 80};
    int x2[] = {80, 30, 50, 150, 100, 170};
    int y1[] = {50, 70, 120, 120, 80};
    int y2[] = {150, 170, 200, 220, 240, 190};

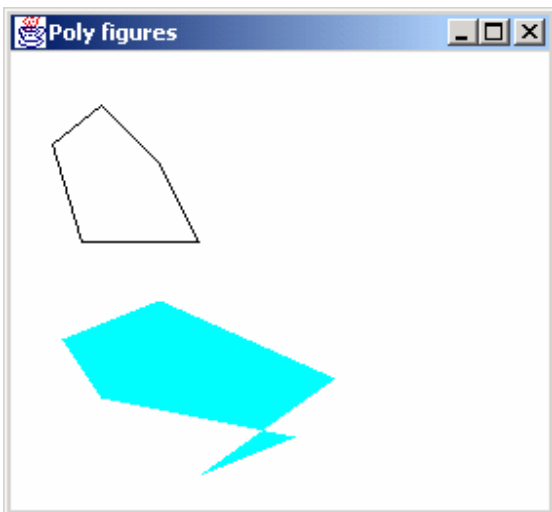
    public PolyFigures()
    {
```

```

        super ("Poly figures");
        setSize(300, 300);
        setVisible (true);
    }
    public void paint (Graphics g)
    {
        g.drawPolygon (x1, y1, 5);
        g.setColor (Color.cyan);
        g.fillPolygon (x2, y2, 6);
    }
    public static void main (String args[])
    {
        new PolyFigures();
    }
}

```

Sau đây là kết quả của chương trình trên:



Hình 6.8 Polygon

6.6 Điều khiển màu

Trong Java, chúng ta điều khiển màu bằng cách dùng 3 màu chính là đỏ (red), xanh lá cây (green), xanh dương (blue). Java sử dụng mô kiểu màu RGB. Đối tượng của lớp Color chứa 3 số nguyên cho các tham số red, green, blue. Bảng sau trình bày giá trị có thể có của các màu đó:

Thành phần	Phạm vi
Red	0-255
Green	0-255
Blue	0-255

Bảng 6.2 Phạm vi giá trị của các thành phần màu

Sử dụng các giá trị trên để tạo ra một màu tùy thích. Cú pháp của hàm dựng để tạo ra một màu như sau:

color (int red, int green, int blue);

Bảng sau hiển thị các giá trị của các màu thường gặp:

Màu	Red	Green	Blue
White	255	255	255
Light Gray	192	192	192
Gray	128	128	128
Dark Gray	64	64	64
Black	0	0	0
Pink	255	175	175
Orange	255	200	0
Yellow	255	255	0
Magenta	255	0	255

Bảng 6.3 Các giá trị RGB

Các đối tượng màu khác nhau có thể được tạo bằng những giá trị này. Những đối tượng này có thể được dùng để vẽ hoặc tô các đối tượng đồ họa. Ví dụ, để tạo màu hồng, ta dùng lệnh sau:

```
color c = new Color (255, 175, 175);
```

Ta có thể thiết lập màu bằng cách dùng lệnh sau:

```
g.setColor (c); //g là đối tượng của lớp Graphics
```

Sử dụng kết hợp các giá trị RGB để tạo ra một màu tùy ý. Để cho dễ hơn, lớp Color cung cấp sẵn một số màu.

color.white	color.black
color.orange	color.gray
color.lightgray	color.darkgray
color.red	color.green
color.blue	color.pink
color.cyan	color.magenta
color.yellow	

Bảng 6.4 Các màu thường gặp

Đoạn mã sau minh họa cách tạo một màu tùy ý:

```
Color color1 = new Color (230, 140, 60);
Color color4 = new Color (90, 210, 130);
g.setColor (color1);
int myred = color1.getRed ();
int mygreen = color1.getGreen ();
int myblue = color1.getBlue();
```

```
color1 = color1.darker();
color4 = color4.brighter();
```

6.7 Điều khiển Font

Java cung cấp lớp Font trong gói java.awt cho phép sử dụng các loại font khác nhau. Lớp này bao gồm một số phương thức.

Để sử dụng font, chúng ta nên kiểm tra xem hệ thống có hỗ trợ hay không. Phương thức 'getAllFont()' trả về tất cả các font mà hệ thống hỗ trợ.

Trước tiên, khai báo một đối tượng của lớp GraphicsEnvironment như sau:

Graphicenvironment ge;

ge = GraphicsEnvironment.getLocalGraphicsEnvironment ();

Đối tượng này sử dụng cú pháp sau để lấy tất cả các font có trong mảng Font:

Font f[] = ge.getAllFonts();

Phương thức getAllFont() được sử dụng ở đây. Phương thức getAllFont() thuộc lớp GraphicsEnvironment. Đây là lớp trừu tượng, do đó ta không thể khởi tạo lớp này. để truy cập phương thức getAllFont(), chúng ta sử dụng phương thức 'getLoacalGraphicsEnvironment()' của lớp GraphicsEnvironment.

ge = GraphicsEnvironment.getLocalGraphicsEnvironment ();

Tham chiếu đến lớp này được gán cho biến ge. Biến này gọi phương thức getAllFont(). Chúng ta sử dụng các font khác nhau để hiển thị các chuỗi khác nhau. Phương thức getFont() trả về font mặc định dùng để hiển thị chuỗi, khi không có chọn font nào cả.

Font defaultFont = g.getFont (); //g là đối tượng Graphics
g.drawString ("Default Font is ", 30, 50);

Dialog là font mặc định của hệ thống.

Để thay đổi font mặc định của hệ thống thành font khác, chúng ta tạo đối tượng của lớp Font. Hàm dựng của Font lấy 3 tham số sau:

- Tên của font. Ta có thể lấy tên thông qua phương thức getFontList().
- Kiểu của font. Ví dụ: Font.BOLD, Font.PLAIN, Font.ITALIC.
- Kích thước font.

Cú pháp sau minh họa những thông số trên:

Font f1 = new Font ("SansSerif", Font.ITALIC, 16);
g.setFont (f1);

Ba tham số được truyền ở đây là: 'SanSerif' – tên của font, Font.BOLD – kiểu font, 14 là kích thước của font. Những thông số này tạo ra đối tượng f1. Chúng ta có thể kết hợp 2 kiểu font lại với nhau. Hãy xét ví dụ sau:

```
Font f3 = new Font ("Monospaced", Font.ITALIC+Font.BOLD, 20);
```

Ở đây kiểu font của f3 vừa đậm, vừa nghiêng.

6.8 Lớp FontMetrics

Lớp này xác định kích thước của các ký tự khác nhau thuộc các loại font khác nhau. Xác định kích thước bao gồm chiều cao (height), baseline, descent, và leading. Điều này rất cần thiết vì các ký tự khi in đều chiếm một kích thước riêng. Bạn cần tính kích thước cần thiết khi in các ký tự để tránh các ký tự ghi đè lên nhau.

- Height: chiều cao của font.
- Baseline (Dòng cơ sở): xác định cơ sở của các ký tự (không kể phần thấp nhất của ký tự)
- Ascent: khoảng cách từ đường baseline đến đỉnh của ký tự.
- Descent: khoảng cách từ baseline đến đáy của ký tự.
- Leading: khoảng cách giữa các dòng chữ in.

Chương trình 6.7 minh họa việc sử dụng các phương thức khác nhau mà lớp FontMetrics có. Trong chương trình này, chúng ta sử dụng các phương thức khác nhau để xem xét chi tiết các loại font khác nhau. Lớp FontMetric là lớp trừu tượng. Phương thức getFontMetrics() có tham số là đối tượng của lớp Font, vì FontMetrics đi đôi với một font nào đó.

```
FontMetrics fm = g.getFontMetrics (f1);
```

Lệnh này tạo đối tượng fm của lớp FontMetrics, cùng với đối tượng f1. Bây giờ, chúng ta sử dụng fm để lấy chi tiết của font.

Các phương thức getHeight(), getAscent(), getDescent(), và getLeading() trả về chi tiết của font. Phương thức getFont() của lớp FontMetrics trả về Font mà kết hợp với đối tượng của lớp FontMetrics. Phương thức getName() của lớp Font trả về tên Font.

Chương trình 6.7

```
import java.awt.*;

class FontMetricsUse extends Frame
{
    public FontMetricsUse()
    {
        super ("Detail oc Fonts");
        setSize (400, 300);
        setVisible(true);
    }
}
```

```

}

public void paint (Graphics g)
{
    Font f1 = new Font ("Times Roman", Font.PLAIN, 22);
    FontMetrics fm = g.getFontMetrics (f1);
    String name = fm.getFont().getName();
    g.drawString ("Details of Font " + name, 30, 50);
    g.drawString ("Leading: " + String.valueOf (fm.getHeight()), 50, 75);
    g.drawString ("Leading: " + String.valueOf (fm.getAscent()), 50, 100);
    g.drawString ("Leading: " + String.valueOf (fm.getDescent()), 50, 125);
    g.drawString ("Leading: " + String.valueOf (fm.getLeading()), 50, 150);

    Font f2 = new Font ("DialogInput", Font.PLAIN, 22);
    fm = g.getFontMetrics (f2);
    name = fm.getFont().getName();
    g.drawString ("Details of Font " + name, 30, 175);
    g.drawString ("Leading: " + String.valueOf (fm.getHeight()), 50, 200);
    g.drawString ("Leading: " + String.valueOf (fm.getAscent()), 50, 225);
    g.drawString ("Leading: " + String.valueOf (fm.getDescent()), 50, 250);
    g.drawString ("Leading: " + String.valueOf (fm.getLeading()), 50, 275);
}
public static void main (String args[])
{
    new FontMetricsUse ();
}
}

```

Kết quả của chương trình trên:



Hình 6.9 Lớp FontMetrics

Chương trình 6.8 minh họa cách lớp FontMetrics được sử dụng để in đoạn văn bản nhiều

font, nhiều dòng. Trong chương trình này, chúng ta cần in văn bản nhiều font trên nhiều dòng. Lớp FontMetrics giúp ta xác định khoảng cách cần thiết để in một dòng văn bản cho một font nào đó. Điều này thật cần thiết, bởi vì dòng thứ 2 được in ngay sau dòng thứ nhất.

Trước tiên chúng ta in msg1 sử dụng font Monospaced. Sau đó, chúng ta xuất msg2 sử dụng font DialogInput. Để làm được điều này, chúng ta cần tính khoảng cách cần thiết để xuất msg1. Phương thức stringWidth() của lớp FontMetrics được dùng để tính ra tổng khoảng cách cần thiết để xuất msg1. Khi chúng cộng thêm khoảng cách này vào biến x, chúng ta sẽ lấy được vị trí mà chúng ta bắt đầu in đoạn văn bản kế tiếp, msg2. Phương thức setFont() được dùng để thiết lập font để in văn bản.

Kế đó, chúng ta xuất msg1 và msg2 trên các dòng khác nhau sử dụng chung 1 font Monospaced. Ở đây, chúng ta cần biết khoảng cách chiều cao của font, để in dòng kế tiếp. Phương thức getHeight() được dùng để làm điều này.

Chương trình 6.8

```
import java.awt.*;
class MultiFontMultiLine extends Frame
{
    public MultiFontMultiLine()
    {
        super ("Multiline Text");
        setSize (450, 200);
        setVisible (true);
    }

    public void paint (Graphics g)
    {
        Font f1 = new Font ("MonoSpaced", Font.BOLD, 18);
        Font f2 = new Font ("DialogInput", Font.PLAIN, 14);

        int x = 20;
        int y = 50;
        String msg1 = "Java Language";
        String msg2 = "A new approach to programming";

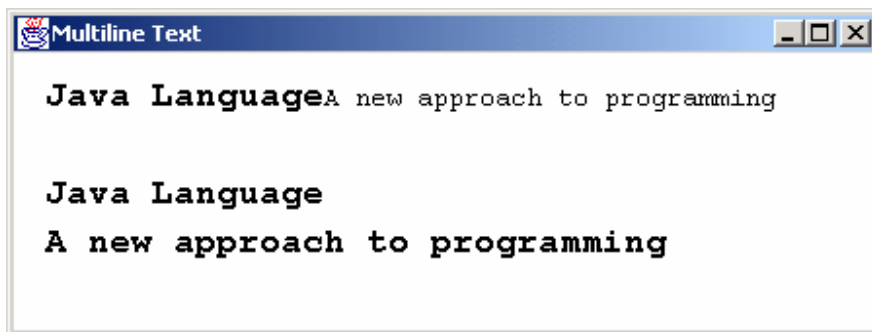
        FontMetrics fm = g.getFontMetrics(f1);
        g.setFont(f1);
        g.drawString (msg1, x, y);
        x = x + fm.stringWidth(msg1);
        g.setFont(f2);
        g.drawString (msg2, x, y);
        g.setFont(f1);
        y = 100;
        x = 20;
        int height = fm.getHeight();
    }
}
```

```

        g.drawString (msg1, x, y);
        y += height;
        g.drawString (msg2, x, y);
    }
    public static void main (String args[])
    {
        new MultiFontMultiLine ();
    }
}

```

Kết quả của chương trình trên:



Hình 6.10 Văn bản được xuất nhiều font, nhiều dòng

6.9 Chọn mode để vẽ

Các đối tượng được vẽ bằng cách sử dụng mode vẽ. Khi một đối tượng mới được vẽ, nó sẽ đè lên các hình đã vẽ trước đây. Tương tự, khi các đối tượng được vẽ đi vẽ lại nhiều lần thì chúng sẽ xoá các đối tượng đã vẽ trước đó. Chỉ hiển thị nội dung của đối tượng mới. Để làm cho nội dung cũ và nội dung mới đều hiển thị trên màn hình, lớp Graphics cung cấp phương thức `setXORMode (Color c)`;

Chương trình 6.9 minh họa tiện lợi của của việc sử dụng phương thức `setXORMode()`. Ở đây, chúng ta sử dụng phương thức `setXORMode()` để tô các hình đồ họa khác nhau, mà không đè lên các hình khác. Kết quả là, khi sử dụng mode XOR thì hiển nhiên là tất cả các hình đều hiển thị đầy đủ. Điều này có nghĩa là các hình mới không đè lên các hình cũ. Thay vào đó, phần chung giữa các hình sẽ được hiển thị thành một màu khác. Nhưng khi không sử dụng mode XOR, hình mới hoàn toàn che khuất những hình trước đó.

Chương trình 6.9

```

import java.awt.*;
class PaintMode extends Frame
{
    public PaintMode()
    {
        super ("Paint Mode");
        setSize (300, 300);
    }
}

```

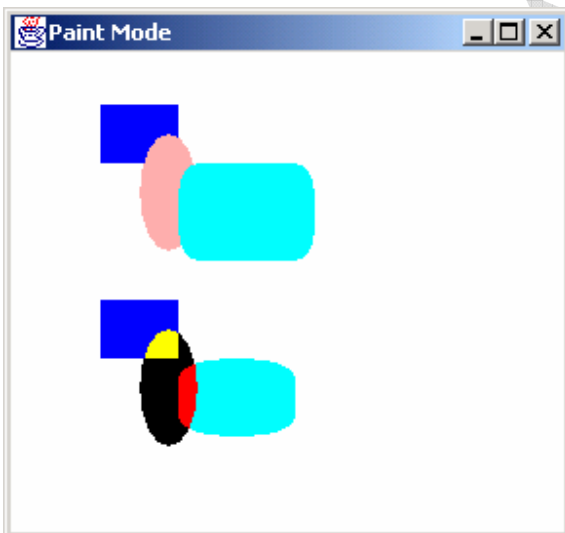
```

        setVisible (true);
    }
    public void paint (Graphics g)
    {
        g.setPaintMode ();
        g.setColor (Color.blue);
        g.fillRect (50,50,40, 30);
        g.setColor (Color.pink);
        g.fillOval (70, 65, 30, 60);
        g.setColor (Color.cyan);
        g.fillRoundRect (90, 80, 70, 50, 20, 30);
        g.setColor (Color.blue);
        g.fillRect (50, 150, 40, 30);
        g.setXORMode (Color.yellow);
        g.fillOval (70, 165, 30, 60);
        g.setXORMode (Color.magenta);
        g.fillRoundRect (90, 180, 60, 40, 50, 20);
    }

    public static void main (String args[])
    {
        new PaintMode();
    }
}

```

Kết quả của chương trình trên:



Hình 6.11 Paint mode

Tóm tắt

- Applet là chương trình Java chạy trong trình duyệt web.
- Chương trình Java đơn lẻ có thể vừa là applet, vừa là application.

- Lớp Graphics nằm trong gói AWT, bao gồm các phương thức được sử dụng để vẽ các hình đồ họa như oval, hình chữ nhật, hình vuông, hình tròn, đường thẳng và văn bản.
- Java sử dụng bảng màu RGB.
- Lớp Font trong gói java.awt cho phép sử dụng nhiều font khác nhau.
- Lớp FontMetrics xác định kích thước của các ký tự.

!!! Check & Exercise !!!

<http://www.ngohaianh.info>