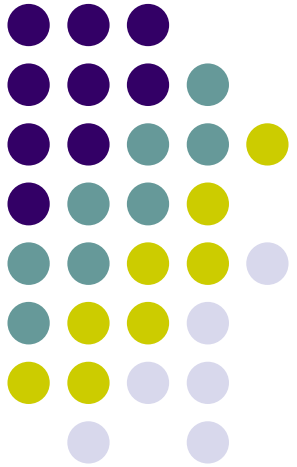
第二十章
AWT的繪圖

認識Java繪圖的基本概念

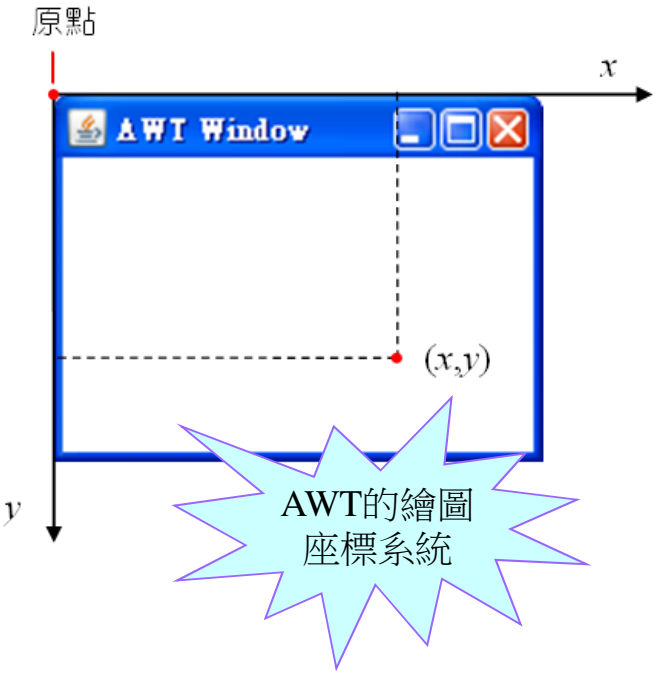
學習設定圖形的顏色與文字的字型

學習用滑鼠繪圖的基本程式設計



座標系統

- 每個點都是由兩個座標來表示
  - x座標
  - y座標
- 視窗的原點
  - 位於視窗的左上角，包括
    - 視窗的標題列
    - 視窗的邊界

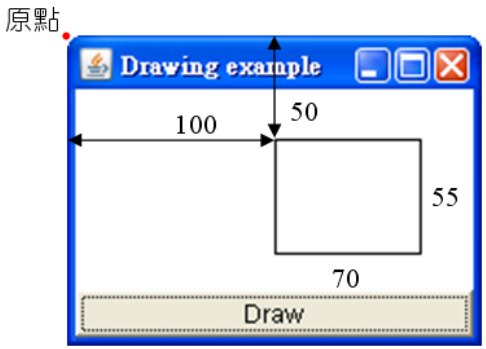


取得繪圖區

- 「圖形內容」（graphics context）
  - Graphics類別產生的物件
  - 「圖形內容」可透過getGraphics() method取得：

```
Graphics g=getGraphics();      // 取得「圖形內容」，也就是視窗的繪圖區
```

- app20_1在視窗內配置一個按鈕，用來觸發繪圖事件



app20_1執行
結果說明圖

- 在繪圖區內繪製長方形，可用drawRect() method：

```
void drawRect(int x, int y, int width, int height)      // 繪出長方形
```

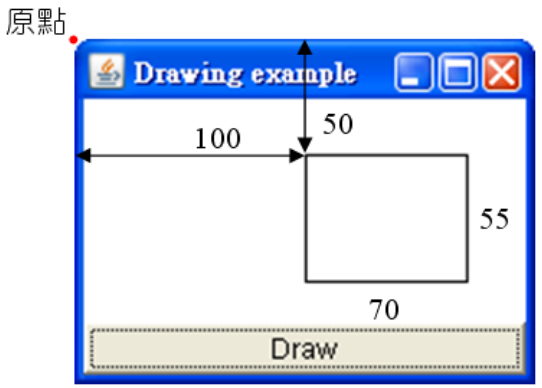
  - 要繪出如圖20.1.2的長方形，可用下面的語法：

```
g.drawRect(100,50,70,55);          // 在繪圖區內繪出長方形
```

簡單的繪圖

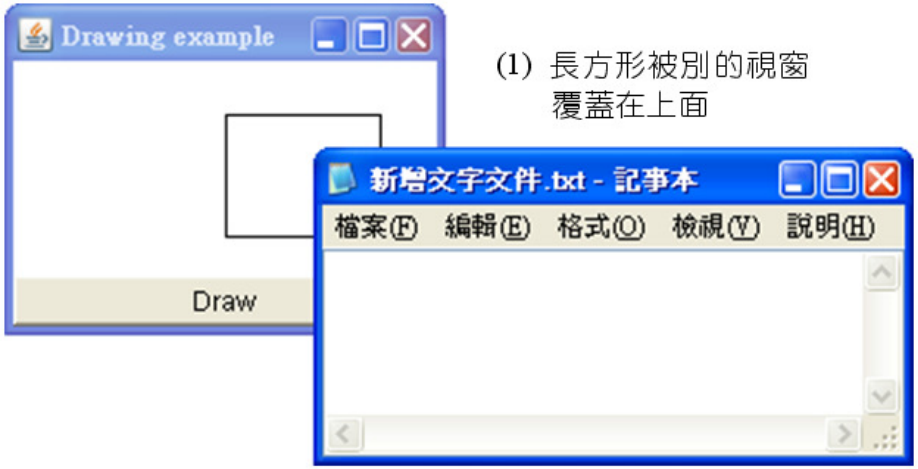
app20_1的程式碼

```
01  // app20_1, 簡單的繪圖
02  import java.awt.*;
03  import java.awt.event.*;
04  public class app20_1 extends Frame implements ActionListener
05  {
06     static app20_1 frm=new app20_1();
07     static Button btn=new Button("Draw");
08
09     public static void main(String args[])
10     {
11        BorderLayout br=new BorderLayout();
12        frm.setTitle("Drawing example");
13        frm.setLayout(br);
14        frm.setSize(200,150);
15        frm.add(btn,br.SOUTH);
16        btn.addActionListener(frm);
17        frm.setVisible(true);
18     }
19     public void actionPerformed(ActionEvent e)
20     {
21        Graphics g=getGraphics();              // 取得視窗的繪圖區
22        g.drawRect(100,50,70,55);              // 繪出長方形
23     }
24  }
```

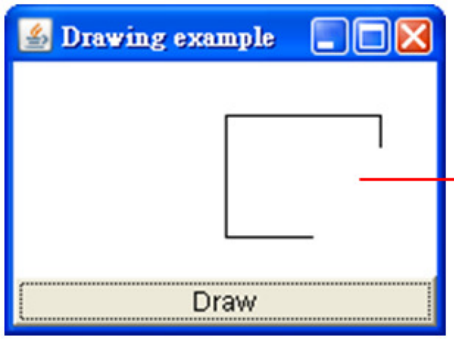


被覆蓋掉的圖形

- app20_1所繪出來的長方形會被覆蓋掉



(1) 長方形被別的視窗覆蓋在上面



(2) 當視窗移開時，被覆蓋的部份會消失

當視窗縮到最小，再放到最大時，整個長方形甚至會完全消失

paint() method

- paint() method
  - 自發性的（spontaneous），在適當的時機會自動執行
  - paint() 在下列的情況發生時會自動執行：
    - 當新建的視窗顯示於螢幕上，或從隱藏變成顯示時
    - 從縮小圖示還原之後
    - 正在改變視窗的大小時
  - paint() 的格式為：

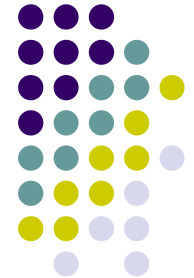
```
public void paint(Graphics g)                    // paint() 的格式
```

修正繪圖區被覆蓋的問題

```
01  // app20_2, app20_1 的修改版
02  import java.awt.*;
03  import java.awt.event.*;
04  public class app20_2 extends Frame implements ActionListener
05  {
06     static app20_2 frm=new app20_2();
07     static Button btn=new Button("Draw");
08
09     public static void main(String args[])
10     {
11        BorderLayout br=new BorderLayout();
12        frm.setTitle("Drawing example");
13        frm.setLayout(br);
14        frm.setSize(200,150);
15        frm.add(btn,br.SOUTH);
16        btn.addActionListener(frm);
17        frm.setVisible(true);
18     }
19     public void actionPerformed(ActionEvent e)
20     {
21        Graphics g=getGraphics();              // 取得視窗的繪圖區
22        paint(g);                              // 呼叫 paint() method
23     }
24     public void paint(Graphics g)
25     {
26        g.drawRect(100,50,70,55);              // 繪出長方形
27     }
28  }
```

把app20_1改寫成app20_2

解決新的問題



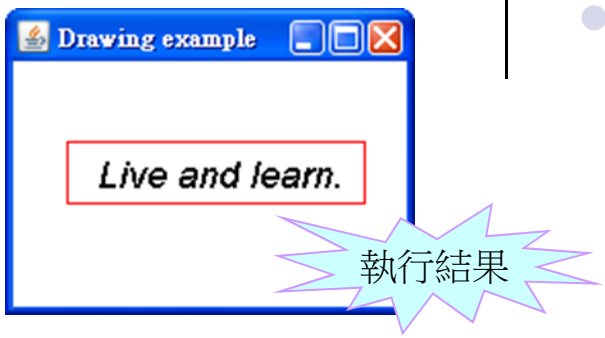
將app20_2改寫成app20_3

```
01  // app20_3, app20_2 的修改版
02  import java.awt.*;
03  import java.awt.event.*;
04  public class app20_3 extends Frame implements ActionListener
05  {
06     static app20_3 frm=new app20_3();
07     static Button btn=new Button("Draw");
08     boolean clicked=false;    // 宣告 boolean 變數，用來判別按鈕是否按下
09
10     public static void main(String args[])
11     {
12        BorderLayout br=new BorderLayout();
13        frm.setTitle("Drawing example");
14        frm.setLayout(br);
15        frm.setSize(200,150);
16        frm.add(btn,br.SOUTH);
17        btn.addActionListener(frm);
18        frm.setVisible(true);
19     }
20     public void actionPerformed(ActionEvent e)
21     {
22        clicked=true;                          // 設定按鈕已被按下
23        Graphics g=getGraphics();              // 取得視窗的繪圖區
24        paint(g);
25     }
26     public void paint(Graphics g)
27     {
28        if(clicked)                            // 如果按鈕被按下
29           g.drawRect(100,50,70,55);           // 繪出長方形
30     }
31  }
```

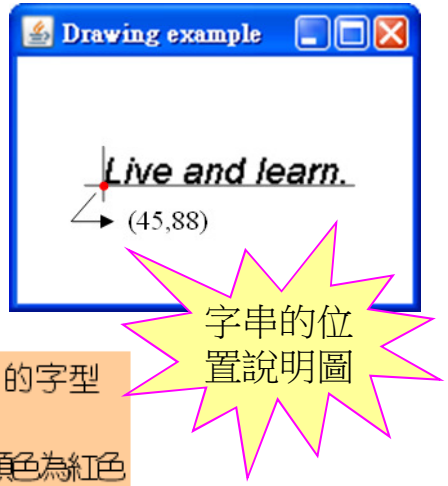
在paint() method裡測試Draw按鈕是否有被按下，如果沒有，就先擋住繪圖程式碼的執行

在視窗中繪圖

- app20_4是另一個簡單的範例




```
01  // app20_4, 簡單的繪圖
02  import java.awt.*;
03  import java.awt.event.*;
04  public class app20_4 extends Frame  // 設定 app20_4 繼承自 Frame 類別
05  {
06     static app20_4 frm=new app20_4();              // 建立視窗物件 frm
07
08     public static void main(String args[])
09     {
10        frm.setTitle("Drawing example");
11        frm.setSize(200,150);
12        frm.setVisible(true);
13     }
14     public void paint(Graphics g)
15     {
16        g.setFont(new Font("Arial",Font.ITALIC,18)); // 設定使用的字型
17        g.drawString("Live and learn.",45,88);       // 繪出字串
18        g.setColor(Color.red);                       // 設定繪圖顏色為紅色
19        g.drawRect(30,65,145,30);                    // 繪出長方形
20     }
21  }
```



設定顏色

- 下表列出Color類別常用的建構元與method：

表 20.2.1　java.awt.Color 的建構元與 method

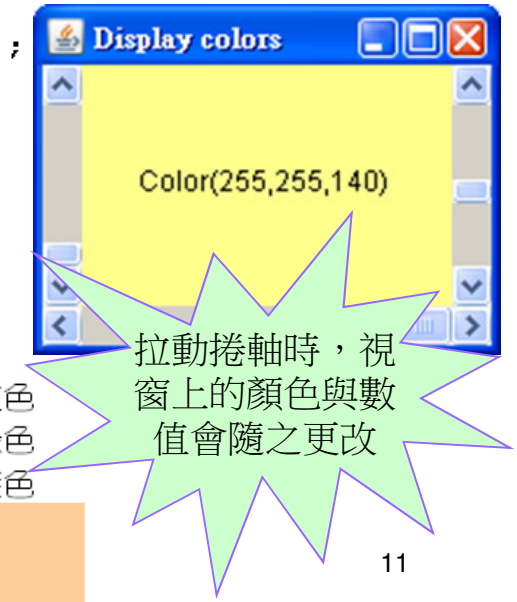
建構元	主要功能
Color(float r, float g, float b)	設定紅色(red)、綠色(green)、與藍色(blue)的值，這三者的值必須介於 0~1 之間的浮點數
Color(int r, int g, int b)	同上，但數值是介於 0~255 之間的整數

method	主要功能
Color brighter()	取得比目前顏色稍亮一點的顏色
Color darker()	取得比目前顏色稍暗一點的顏色
boolean equals(Object obj)	測試顏色是否相等
int getBlue()	取得 Color 裡藍色的值
int getGreen()	取得 Color 裡綠色的值
int getRed()	取得 Color 裡紅色的值

使用Color類別 (1/2)

- app20_5是Color類別使用的範例

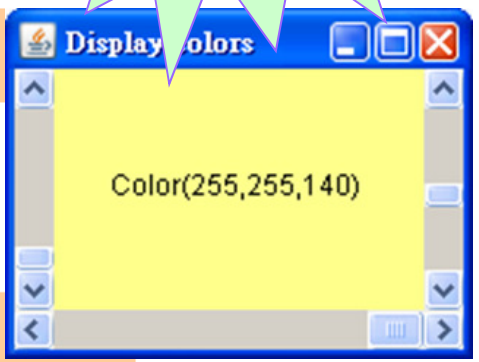
```
01  // app20_5, RGB color 色階的應用
02  import java.awt.*;
03  import java.awt.event.*;
04  public class app20_5 extends Frame implements AdjustmentListener
05  {
06     static app20_5 frm=new app20_5();
07     static Scrollbar scr1=new Scrollbar(Scrollbar.VERTICAL);
08     static Scrollbar scr2=new Scrollbar(Scrollbar.HORIZONTAL);
09     static Scrollbar scr3=new Scrollbar(Scrollbar.VERTICAL);
10
11     public static void main(String args[])
12     {
13        BorderLayout br=new BorderLayout(5,5);
14        frm.setTitle("Display colors");
15        frm.setSize(200,150);
16        frm.add(scr1,br.WEST);              // scr1 捲軸，用來控制紅色
17        frm.add(scr2,br.SOUTH);             // scr2 捲軸，用來控制綠色
18        frm.add(scr3,br.EAST);              // scr3 捲軸，用來控制藍色
19        scr1.setValues(255,45,0,300);       // 設定 scr1 的相關數值
20        scr2.setValues(255,45,0,300);       // 設定 scr2 的相關數值
21        scr3.setValues(140,45,0,300);       // 設定 scr3 的相關數值
```



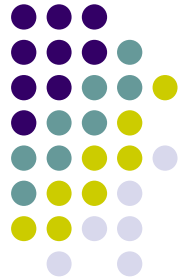
使用Color類別 (2/2)

```
22          scr1.addAdjustmentListener(frm);
23          scr2.addAdjustmentListener(frm);
24          scr3.addAdjustmentListener(frm);
25          frm.setVisible(true);
26      }
27      public void adjustmentValueChanged(AdjustmentEvent e)
28      {
29          Graphics g=getGraphics();
30          paint(g);
31      }
32
33      public void paint(Graphics g)
34      {
35          int red=scr1.getValue();          // 取得捲軸 scr1 的值
36          int green=scr2.getValue();        // 取得捲軸 scr2 的值
37          int blue=scr3.getValue();         // 取得捲軸 scr3 的值
38          String str="Color("+red+","+green+","+blue+")";
39          g.setColor(new Color(red,green,blue));     // 設定繪圖顏色
40          g.fillRect(0,0,getWidth(),getHeight());
41          g.setColor(Color.black);          // 設定繪圖顏色為黑色
42          g.drawString(str,45,80);          // 於(45,80)處寫上字串
43      }
44  }
```

拉動捲軸時，視窗上的顏色與數值會隨之更改



darker() 與brighter() method (1/2)

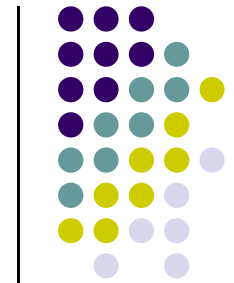


- app20_6是darker() 與brighter() 的使用範例

```
01  // app20_6, brighter()與 darker()的使用
02  import java.awt.*;
03  import java.awt.event.*;
04  class app20_6 extends Frame implements ActionListener
05  {
06     static app20_6 frm=new app20_6();
07     static Button btn1=new Button("Brighter");    // Brighter 按鈕
08     static Button btn2=new Button("Darker");      // Darker 按鈕
09     Color co=new Color(255,255,255);              // 設定顏色的初值
10
11     public static void main(String args[])
12     {
13        frm.setTitle("Brighter & Darker");
14        frm.setLayout(new FlowLayout());
15        frm.setSize(200,150);
16        frm.add(btn1);
17        frm.add(btn2);
```

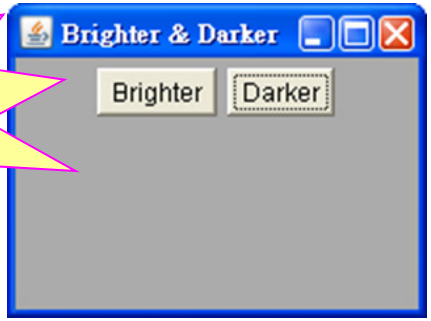
Brighter與Darker
按鈕可用來控制
視窗底色的明暗

darker() 與brighter() method (2/2)



```
18          btn1.addActionListener(frm);
19          btn2.addActionListener(frm);
20          frm.setVisible(true);
21      }
22      public void actionPerformed(ActionEvent e)
23      {
24          Button btn=(Button) e.getSource();          // 取得被按下的按鈕
25          if(btn==btn1)
26              co=co.brighter();   // 如果是按下 brighter，則將顏色變亮一點
27          else if(btn==btn2)
28              co=co.darker();     // 如果是按下 Darker，則將顏色變暗一點
29          Graphics g=getGraphics();
30          paint(g);
31      }
32      public void paint(Graphics g)
33      {
34          g.setColor(co);                             // 將繪圖顏色設為 co
35          g.fillRect(0,0,getWidth(),getHeight());     // 填滿顏色
36      }
37  }
```

Brighter與Darker按鈕可用來控制視窗底色的明暗



設定字型

- 下表列出Font類別常用的建構元與method：

表 20.2.2　java.awt.Font 的建構元與 method

建構元	主要功能
Font(String name, int style, int size)	設定字型的名稱、樣式與大小

method	主要功能
String getFontName()	取得字型的名稱
int getSize()	取得字型的大小
int getStyle()	取得字型的樣式(粗體、斜體或一般)
boolean isBold()	測試字體是否為粗體
boolean isItalic()	測試字體是否為斜體
boolean isPlain()	測試字體是否為一般字體

系統提供的字型 (1/2)

- app20_7是個可用來列出系統所提供之所有字型的範例

```
01  // app20_7, 列出系統所提供的字型
02  import java.awt.*;
03  import java.awt.event.*;
04  class app20_7 extends Frame implements ItemListener
05  {
06     static app20_7 frm=new app20_7();
07     static List lst=new List();      // 建立一個 List 物件
08     String str="Arial";              // 設定字串的初值
09
10     public static void main(String args[])
11     {
12        BorderLayout br=new BorderLayout(5,5);
13        frm.add(lst,br.NORTH);
14        frm.setBackground(Color.yellow);
15        frm.setTitle("Font List");
16        lst.addItemListener(frm);
17        frm.setSize(250,150);
18        GraphicsEnvironment ge;
19        ge=GraphicsEnvironment.getLocalGraphicsEnvironment();
20        String fnt[]=ge.getAvailableFontFamilyNames();
```

建立GraphicsEnvironment的物件ge，並以它來取得系統裡所提供的字型

系統提供的字型 (2/2)

```
21          for(int i=2;i<fnt.length-2;i++)
22             lst.add(fnt[i]);
23          frm.setVisible(true);
24       }
25       public void itemStateChanged(ItemEvent e)
26       {
27          str=lst.getSelectedItem();    // 取得選擇表單裡被選取的項目
28          Graphics g=getGraphics();
29          update(g);                    // 清除背景顏色，再呼叫 paint()
30       }
31       public void paint(Graphics g)
32       {
33          g.setFont(new Font(str,Font.PLAIN,20));    // 設定字型
34          g.setColor(Color.black);                   // 設定顏色
35          g.drawString(str,50,110);     // 用指定的顏色與字型寫上字串
36       }
37    }
```

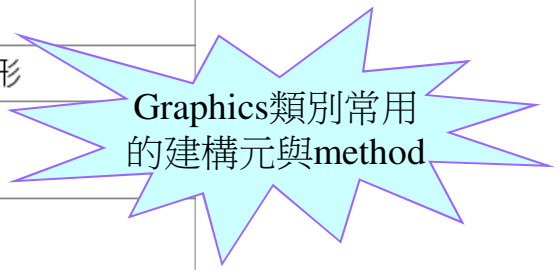
建構元與method (1/2)

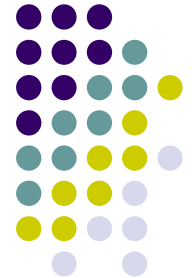
- Graphics類別是從java.lang.Object類別衍生而來

表 20.3.1　java.awt.Graphics 的建構元與 method

建構元	主要功能
protected Graphics()	建立一個新的 Graphics 物件

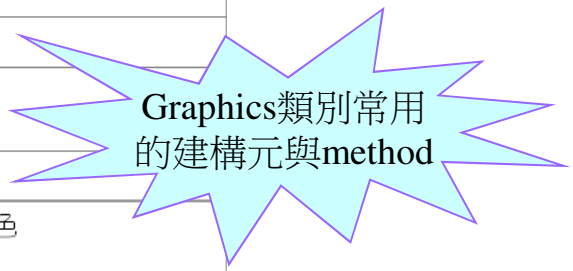
method	主要功能
abstract void clearRect(int x, int y, int w, int h)	清除所指定之長方形的繪圖區域，並填上背景顏色
abstract void clipRect(int x, int y, int w, int h)	只顯示長方形區域內的圖形
abstract void drawArc(int x, int y, int w, int h, int startAngle, int arcAngle)	繪出弧形
void drawChars(char[] data, int offset, int length, int x, int y)	在繪圖區寫上字元
abstract boolean drawImage(Image img, int x, int y, ImageObserver observer)	顯示指定的圖檔，圖形的左上角放在座標(x,y)之處
abstract boolean drawImage(Image img, int x,int y, int width, int height, ImageObserver observer)	顯示指定的圖檔，圖形的左上角放在座標(x,y)之處，並自動調整圖形的大小，使其寬度為 width，高為 height 個點素
abstract void drawLine(int x1, int y1, int x2, int y2)	繪出線段





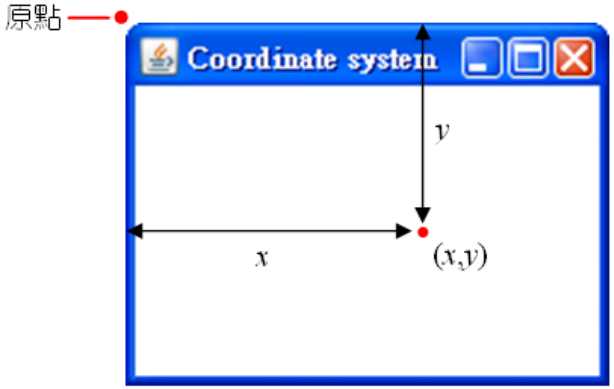
建構元與method (2/2)

method	主要功能
abstract void drawOval(int x, int y, int w, int h)	繪出橢圓形
abstract void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)	繪出多邊形
abstract void drawPolyline(int[] xPoints, int[] yPoints, int nPoints)	用線段將所給予的點連接起來
void drawRect(int x, int y, int w, int h)	繪出長方形
abstract void drawRoundRect(int x, int y, int w, int h, int arcW, int arcH)	繪出圓角長方形
abstract void drawString(String str, int x, int y)	在繪圖區寫上字串 str
abstract void fillArc(int x, int y, int w, int h, int startAngle, int arcAngle)	繪出弧形，並填滿顏色
abstract void fillOval(int x, int y, int w, int h)	繪出橢圓形，並填滿顏色
abstract void fillPolygon(int[] xPoints, int[] yPoints, int nPoints)	繪出多邊形，並填滿顏色
abstract void fillRect(int x, int y, int w, int h)	繪出長方形，並填滿顏色
abstract void fillRoundRect(int x, int y, int w, int h, int arcW, int arcH)	繪出圓角長方形，並填滿顏色
abstract Color getColor()	取得繪圖的顏色
abstract Font getFont()	取得繪圖的字型
abstract void setClip(int x, int y, int w, int h)	設定剪裁的區域為長方形
abstract void setColor(Color c)	設定繪圖的顏色為 c
abstract void setFont(Font font)	設定繪圖區所用的顏色為 c
abstract void translate(int x, int y)	將原點定位在點(x,y)



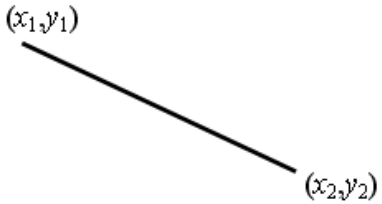
座標系統的原點

- method只要是牽涉到座標
  - 均是以視窗的左上角為原點
  - 向右為正x方向
  - 向下為正y方向
  - 繪圖區域內的每一個點均可由座標 (x,y) 來表示：



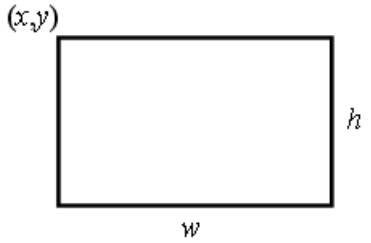
繪圖method引數的說明 (1/2)

- 下面列出部份method的引數於二維平面中所代表的意義：



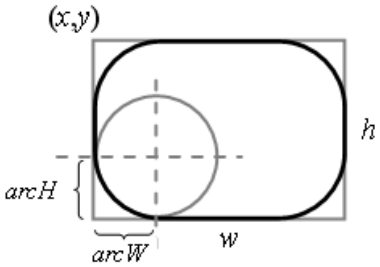
drawLine(int x_1, int y_1, int x_2, int y_2)

drawRect()，給予頂點與寬高來繪製長方形



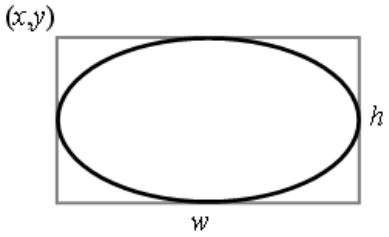
drawRect(int x, int y, int w, int h)

drawRect()，給予頂點與寬高來繪製長方形



drawRoundRect()，給予頂點、寬高與兩個圓角距離繪製圓角長方形

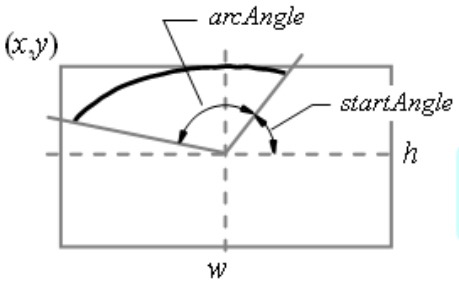
drawRoundRect(int x, int y, int w, int h,int $arcW$, int $arcH$)



drawOval(int x, int y, int w, int h)

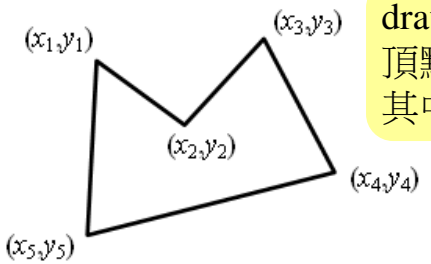
drawOval()，給予頂點與寬高來繪製橢圓形

繪圖method引數的說明 (2/2)



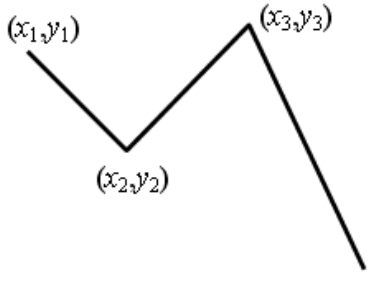
drawArc()，給予頂點、起始角度與弧形角度來繪弧

drawArc(int *x*, int *y*, int *startAngle*, int *arcAngle*)



drawPolygon()，用線段連接頂點，形成封閉的多邊形，其中nPoints為所給予的點數

drawPolylgon(int *xPoints*[], int *yPoints*[], int *nPoints*)



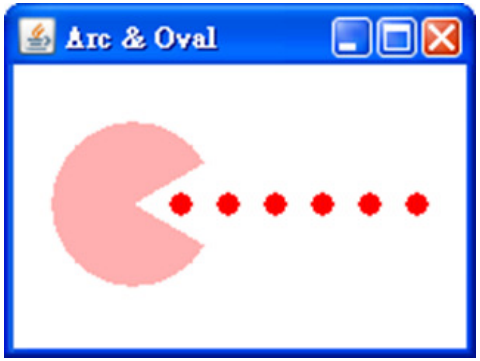
drawPolyline()，用線段連接每一個頂點，nPoints為所給予的點數

drawPolyline(int *xPoints*[], int *yPoints*[], int *nPoints*)

基本幾何形狀的繪圖

- app20_8是圓弧和圓形的繪圖練習

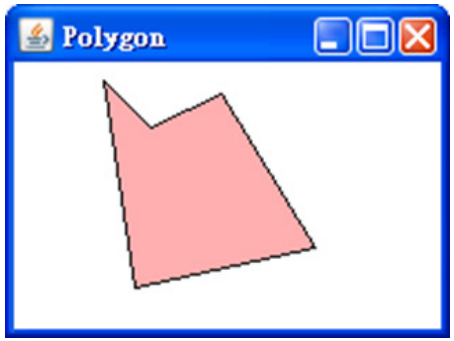
```
01  // app20_8, 填滿圓弧與圓形
02  import java.awt.*;
03  public class app20_8 extends Frame
04  {
05     static app20_8 frm=new app20_8();
06
07     public static void main(String args[])
08     {
09        frm.setTitle("Arc & Oval");
10        frm.setSize(200,150);
11        frm.setVisible(true);
12     }
13
14     public void paint(Graphics g)
15     {
16        g.setColor(Color.pink);              // 設定繪圖顏色為粉紅
17        g.fillArc(20,50,70,70,30,300);       // 填滿圓弧
18        g.setColor(Color.red);               // 設定繪圖顏色為紅色
19        for(int x=70;x<=170;x=x+20)
20           g.fillOval(x,80,10,10);           // 繪出六個小圓
21     }
22  }
```

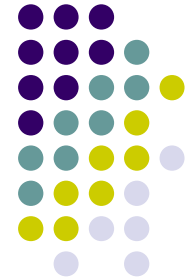


多邊形的繪圖

- 下面的範例備齊所有的頂點後繪製多邊形：

```
01  // app20_9, 繪製多邊形
02  import java.awt.*;
03  class app20_9 extends Frame
04  {
05     static app20_9 frm=new app20_9();
06
07     public static void main(String args[])
08     {
09        frm.setTitle("Polygon");
10        frm.setSize(200,150);
11        frm.setVisible(true);
12     }
13
14     public void paint(Graphics g)
15     {
16        int x[]={44,65,97,139,58};    // 儲存所有頂點 x 座標的一維陣列
17        int y[]={34,55,40,109,127};   // 儲存所有頂點 y 座標的一維陣列
18        g.setColor(Color.pink);       // 繪圖顏色設為粉紅色
19        g.fillPolygon(x,y,5);         // 繪出多邊形，並填滿粉紅色
20        g.setColor(Color.black);      // 繪圖顏色設為黑色
21        g.drawPolygon(x,y,5);         // 用黑色繪出多邊形
22     }
```





會變色的小圓形

app20_10利用呼叫亂數來改變視窗內54個小圓的顏色


```
01  // app20_10, 以不同顏色的小圓形鋪滿視窗
02  import java.awt.*;
03  class app20_10 extends Frame
04  {
05     static app20_10 frm=new app20_10();
06
07     public static void main(String args[])
08     {
09        frm.setTitle("Random Color");
10        frm.setSize(200,150);
11        frm.setVisible(true);
12     }
13
14     public void paint(Graphics g)
15     {
16        for(int x=10;x<=180;x=x+20)
17           for(int y=27;y<=140;y=y+20)
18           {
19              int red=(int)(Math.random()*255);          // 紅色的亂數
20              int green=(int)(Math.random()*255);        // 綠色的亂數
21              int blue=(int)(Math.random()*255);         // 藍色的亂數
22              g.setColor(new Color(red,green,blue));     // 設定顏色
23              g.fillOval(x,y,15,15);              // 用指定的顏色填滿小圓
24           }
25     }
26  }
```

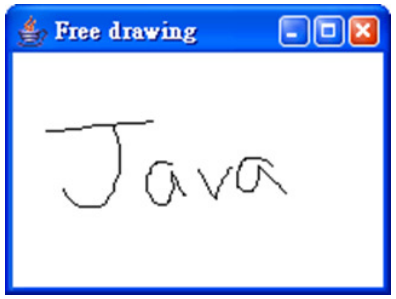
剪裁繪圖區

app20_11是setClip() method的使用範例

```
01  // app20_11, 剪裁繪圖區
02  import java.awt.*;
03  public class app20_11 extends Frame
04  {
05     static app20_11 frm=new app20_11();
06
07     public static void main(String args[])
08     {
09        frm.setTitle("setClip()");
10        frm.setSize(200,150);
11        frm.setVisible(true);
12     }
13
14     public void paint(Graphics g)
15     {
16        g.setClip(30,45,140,80);                    // 限定繪圖的顯示區域
17        for(int x=10;x<=180;x=x+20)
18           for(int y=27;y<=140;y=y+20)
19           {
20              int red=(int)(Math.random()*255);
21              int green=(int)(Math.random()*255);
22              int blue=(int)(Math.random()*255);
23              g.setColor(new Color(red,green,blue));
24              g.fillOval(x,y,15,15);
25           }
26     }
27  }
```

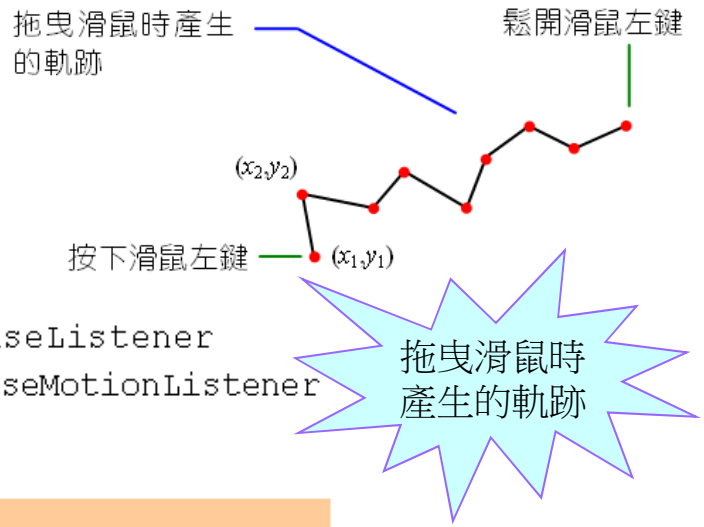
setClip() 的作用相當於剪裁繪圖區，把不要的部份剪裁掉，只留下要顯示的部份

拖曳滑鼠繪圖 (1/2)



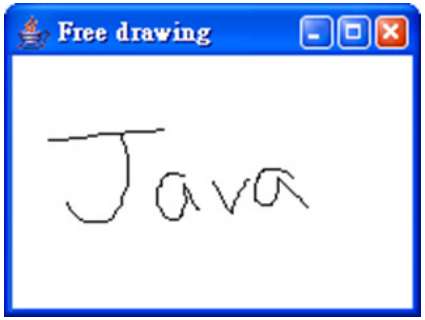
- app20_12是拖曳滑鼠繪圖的範例

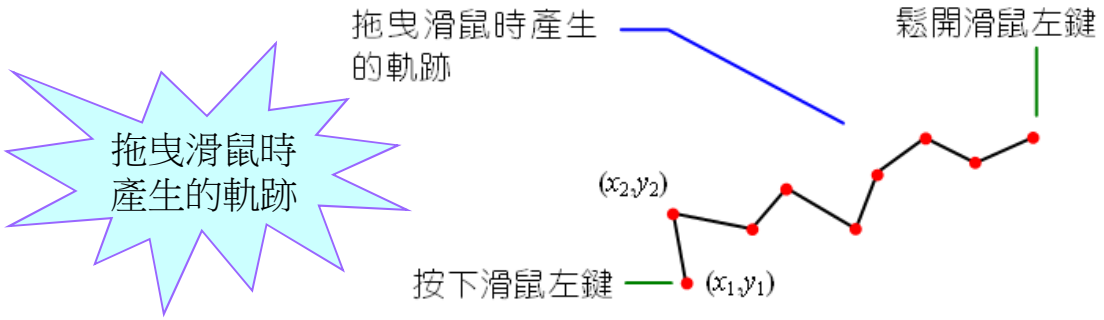
```
01  // app20_12, 拖曳滑鼠繪圖
02  import java.awt.*;
03  import java.awt.event.*;
04  public class app20_12 extends Frame implements MouseMotionListener,MouseListener
05  {
06     static app20_12 frm=new app20_12();
07     int x1,x2,y1,y2;
08     public static void main(String args[])
09     {
10        frm.setTitle("Free drawing");
11        frm.setSize(200,150);
12        frm.addMouseListener(frm);          // 設定 MouseListener
13        frm.addMouseMotionListener(frm);    // 設定 MouseMotionListener
14        frm.setVisible(true);
15     }
16     public void mousePressed(MouseEvent e)
17     {
18        x1=e.getX();     // 取得滑鼠按下時的 x 座標 (繪圖起始點的 x 座標)
19        y1=e.getY();     // 取得滑鼠按下時的 y 座標 (繪圖起始點的 y 座標)
20     }
```



拖曳滑鼠繪圖 (2/2)

```
21      public void mouseDragged(MouseEvent e)
22      {
23         Graphics g=getGraphics();
24         x2=e.getX();                          // 取得拖曳滑鼠時的 x 座標
25         y2=e.getY();                          // 取得拖曳滑鼠時的 y 座標
26         g.drawLine(x1,y1,x2,y2);              // 繪出(x1,y1)到(x2,y2)的連線
27         x1=x2;                                // 更新繪圖起始點的 x 座標
28         y1=y2;                                // 更新繪圖起始點的 y 座標
29      }
30      public void mouseMoved(MouseEvent e){}
31      public void mouseReleased(MouseEvent e){}
32      public void mouseEntered(MouseEvent e){}
33      public void mouseExited(MouseEvent e){}
34      public void mouseClicked(MouseEvent e){}
35   }
```





用滑鼠移動幾何圖形 (1/2)

- 拖曳圓形的動作只是不斷的擦掉圓形與重繪圓形

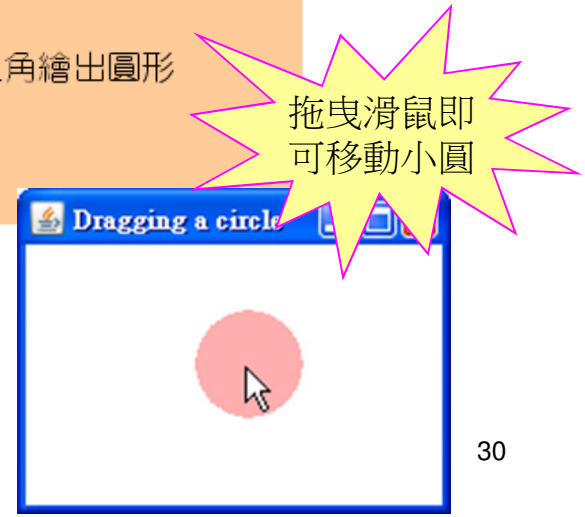
app20_12利用拖曳滑鼠來繪圖

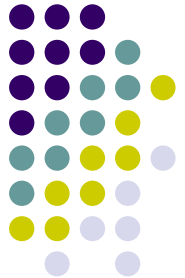
```
01  // app20_13, 利用滑鼠移動圓形
02  import java.awt.*;
03  import java.awt.event.*;
04  public class app20_13 extends Frame implements MouseMotionListener,MouseListener
05  {
06     static app20_13 frm=new app20_13();
07     int x=70,y=60,posX=70,posY=60,dx,dy;
08     public static void main(String args[])
09     {
10        frm.setTitle("Dragging a circle");
11        frm.setSize(200,150);
12        frm.addMouseListener(frm);
13        frm.addMouseMotionListener(frm);
14        frm.setVisible(true);
15     }
16     public void mousePressed(MouseEvent e)
17     {
18        dx=e.getX()-posX;          // 取得滑鼠按下之點與基準點 x 方向的距離
19        dy=e.getY()-posY;          // 取得滑鼠按下之點與基準點 y 方向的距離
20     }
```

拖曳滑鼠即可移動小圓

用滑鼠移動幾何圖形 (2/2)

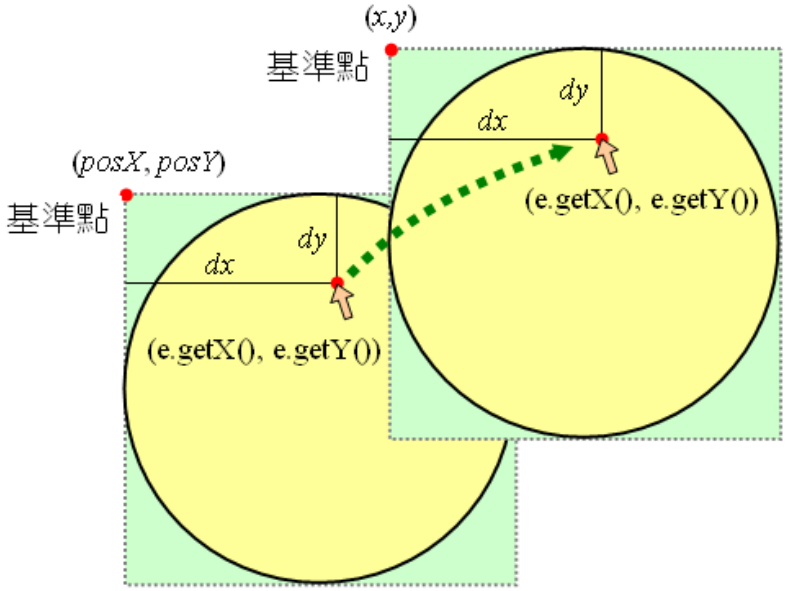
```
21      public void mouseDragged(MouseEvent e)
22      {
23         x=e.getX()-dx;            // 取得拖曳時的基準點 x 座標
24         y=e.getY()-dy;            // 取得拖曳時的基準點 y 座標
25         if(dx>0 && dx<50 && dy>0 && dy<50) // 如果指標落在正方形區域內
26         {
27            Graphics g=getGraphics();
28            update(g);             // 清空畫面為背景顏色，再呼叫 paint()
29         }
30      }
31      public void paint(Graphics g)
32      {
33         g.setColor(Color.pink);   // 設定繪圖顏色為粉紅
34         g.fillOval(x,y,50,50);    // 以基準點為圖形的左上角繪出圓形
35         posX=x;                   // 更新基準點的 x 座標
36         posY=y;                   // 更新基準點的 y 座標
37      }
38      public void mouseMoved(MouseEvent e){}
39      public void mouseReleased(MouseEvent e){}
40      public void mouseEntered(MouseEvent e){}
41      public void mouseExited(MouseEvent e){}
42      public void mouseClicked(MouseEvent e){}
43   }
```

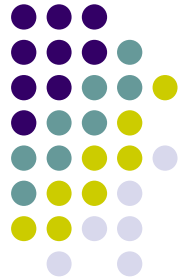




繪圖的基準點

- 繪圖的基準點
  - app20_13中，圓形的外接正方形左上角為繪圖的基準點
  - 當圓形被拖曳時，圓形是以這個點為基準重新繪製
  - 下圖是基準點座標的計算方式之說明





-The End-