内容提要

- Java Applet概述
- Applet生命周期与API
- Applet属性与参数
- Java图形环境和图形对象
- 颜色和字体
- 使用Graphics类绘图
- 使用Graphics2D类绘图

Java Applet概述

- **Java Applet**
  - 一种嵌入HTML文档中的Java程序，具有动态交互与执行的功能，能进行GUI设计与编程，由web浏览器解释执行
  - 与Application相比，Applet具有明显的优点
    - web浏览器提供了运行Applet所需要的许多功能
    - Applet是在运行时通过网络从服务器端下载的，因而便于软件的发布和及时更新
  - Applet也有其局限性
    - 不能在客户机上读写当地文件
    - 也不能连接除它所在的服务器以外的其它机器

Java Applet概述

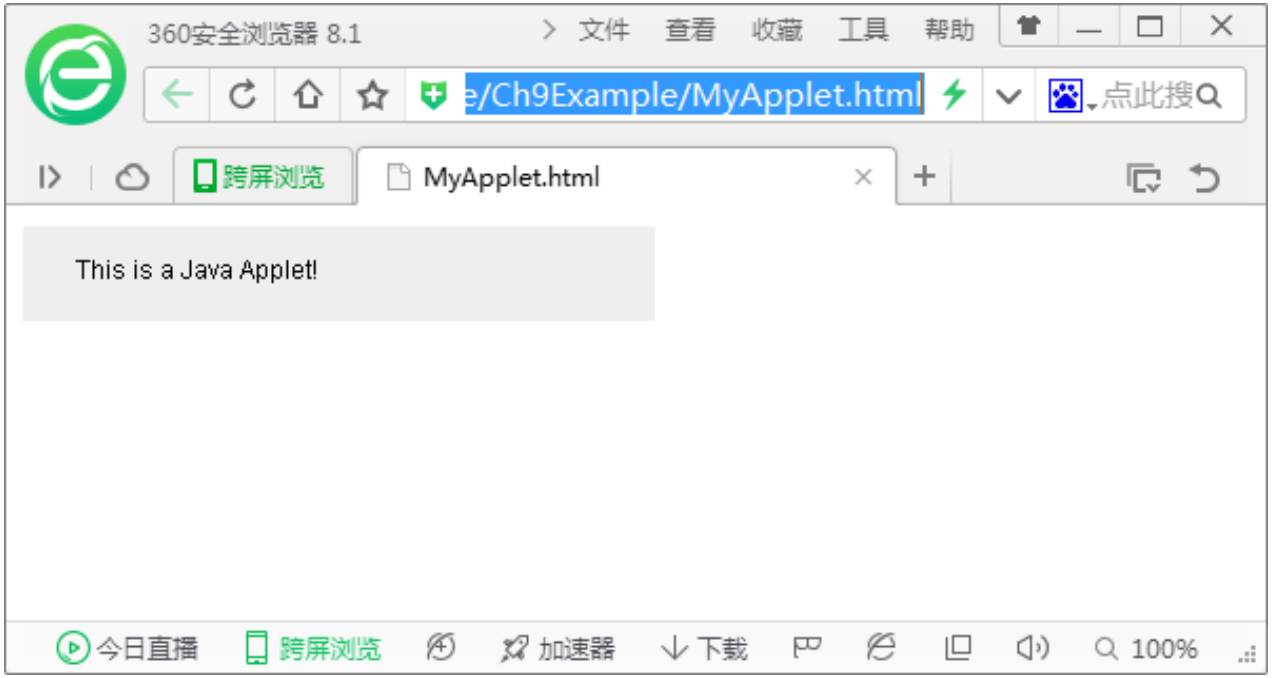
【例9-1】在浏览器中加载一个Applet，功能是显示一个字符串“This is a Java Applet！”

```
import java.awt.Graphics;
import javax.swing.JApplet;
public class MyApplet extends JApplet{
    public void paint(Graphics g) { //重载JApplet类的paint方法
        //参数是Graphics类的对象，是由浏览器传递过来的
        super.paint(g);  //调用其超类JApplet类的paint方法
        g.drawString("This is a Java Applet!",25,25);
    }
}
```

- 编译MyApplet.java产生字节码文件MyApplet.class。接下来就需要编写一个HTML文件MyApplet.html来嵌入MyApplet.class

```
<html>
<applet code="MyApplet.class" width="300" height="45">
</applet>
</html>
```

- 将MyApplet.html文件和MyApplet.class文件放在同一个目录下。现在，在浏览器中打开这个HTML文件，当浏览器遇到Applet标记时，就会自动载入指定的class文件，就会实现在屏幕上绘制一串字符的效果



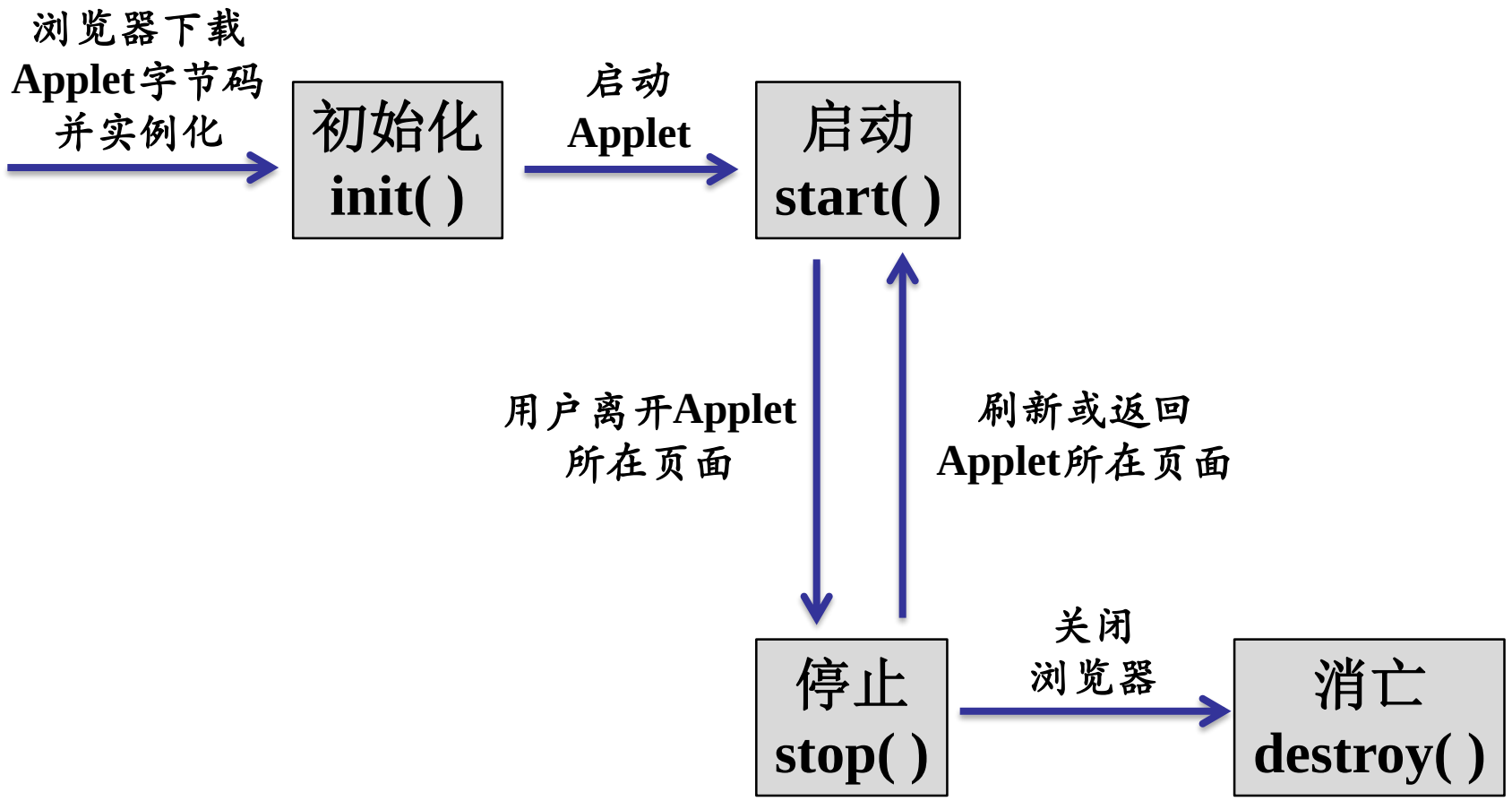
Applet生命周期与API

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Panel
              └java.applet.Applet
                  └javax.swing.JApplet
```

Applet生命周期与API

方法	调用时机和用途
void init()	当浏览器（即Applet容器）载入某个Applet时，容器会自动创建这个Applet类的一个实例，并调用它的init方法
void start()	init方法执行结束之后，自动调用这个方法。另外，当浏览器用户在访问另一个网址之后重新返回applet所在的HTML页时，将再次调用start方法
void paint (Graphics g)	start方法启动后调用此方法，另外每次需要重绘applet时也将调用该方法。程序通常不直接调用paint
void repaint()	在响应用户和Applet的交互时经常要用到。通常只是调用，而不重写这个方法。对于轻量级组件，它调用组件的paint方法，对重量级组件它调用组件的update方法，由update调用paint
void stop()	用户离开Applet所在的HTML页时调用该方法,它执行挂起Applet所需的所有任务，例如停止动画和线程
void destroy()	用户关闭浏览器窗口，Applet将从内存中移走的时候调用该方法

Applet生命周期与API



Applet生命周期与API

- JApplet类
  - JApplet类属于新的Swing组件，是一种顶层容器,默认的布局管理器为BorderLayout
  - 一个applet程序需要继承
    - javax.swing包中的JApplet类或AWT组件中的Applet类
  - 提供了applet在浏览器中运行需要具备的特定方法
    - 浏览器载入时，要依次运行init、start、paint方法
    - 离开浏览器页面时，执行stop
    - 退出浏览器时，执行destroy
  - 提供了所有这些方法的默认实现，在编写Applet时，只要继承这个JApplet，然后重写特定的方法来增加特殊功能

Applet生命周期与API

【例9-2】在控制台和Applet查看器中观察Applet的生命周期

```
import java.awt.Graphics;
import javax.swing.JApplet;
public class AppletLife extends JApplet {
   private String str;
   public void init(){
       str="Hello World!";
       System.out.println("Call init()...");  }
   public void start(){ System.out.println("Call start()..."); }
   public void stop(){ System.out.println("Call stop()..."); }
   public void destroy(){ System.out.println("Call destroy()..."); }
   public void paint(Graphics g) {  g.drawString(str,25,25);  }
}
```

控制台

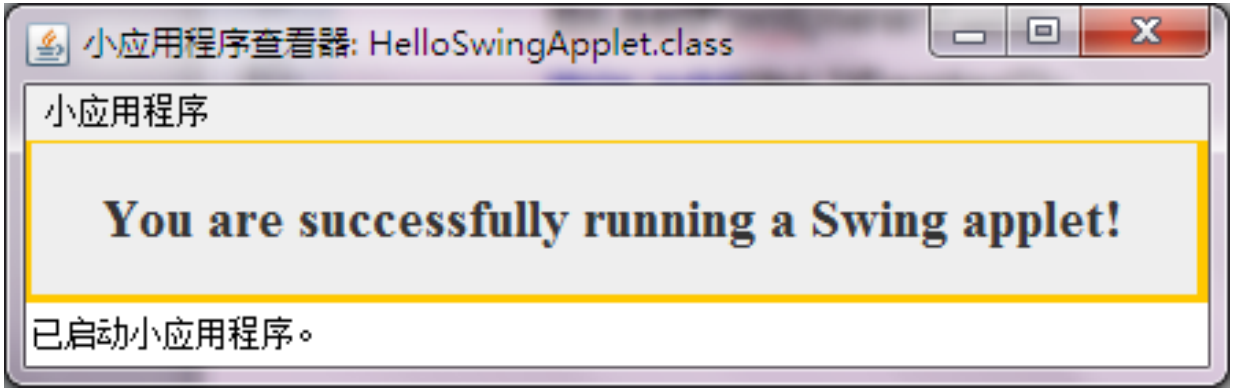
<已终止> AppletLife [Java Applet]

```
Call init()...
Call start()...
Call stop()...
Call start()...
Call stop()...
Call start()...
Call stop()...
Call destroy()...
```

Applet生命周期与API

【例9-3】在Applet中显示标签

```
import java.awt.*;     import javax.swing.*;
public class HelloSwingApplet extends JApplet {
   public void init(){
         JLabel lbl=new JLabel("You are successfully running
                                                  a Swing applet!");
         lbl.setHorizontalAlignment(JLabel.CENTER);
         lbl.setBorder(BorderFactory.createMatteBorder(1, 2, 3, 4,
                                                  Color.ORANGE));
         lbl.setFont(new Font("Times New Roman",Font.BOLD,20));
         this.add(lbl,"Center");
   }
}
```



Applet属性与参数

属性	值	描述
Code(必须)	*URL*	规定 Java applet 的文件名
codebase	*URL*	规定 code 属性中指定的 applet 的基准 URL
archive	*URL*	规定档案文件的位置
name	*unique_name*	规定 applet 的名称，默认为类名
height	*pixels*	定义 applet 的高度
width	*pixels*	定义 applet 的宽度
hspace	*pixels*	定义围绕 applet 的水平间隔
vspace	*pixels*	定义围绕 applet 的垂直间隔
align	•*left / right / top / bottom / middle* •*baseline / texttop* •*absmiddle / absbottom*	定义 applet 相对于周围元素的对齐方式
alt	*text*	规定 applet 的替换文本

```
<applet code="MyApplet.class" width="300" height="45">
</applet>
```

Applet属性与参数

- Java Application中，通过命令行向main()方法传递参数
- 在Applet中
  - 在HTML中通过使用<PARAM>标记定义参数

    <PARM NAME = *appletParamter1* VALUE = *value*>

    <PARM NAME = *appletParamter2* VALUE = *value*>

    ...
  - 而在Applet主类的init()方法中，可以通过使用getParameter()方法获取参数

    getParameter("*appletParamter1*")

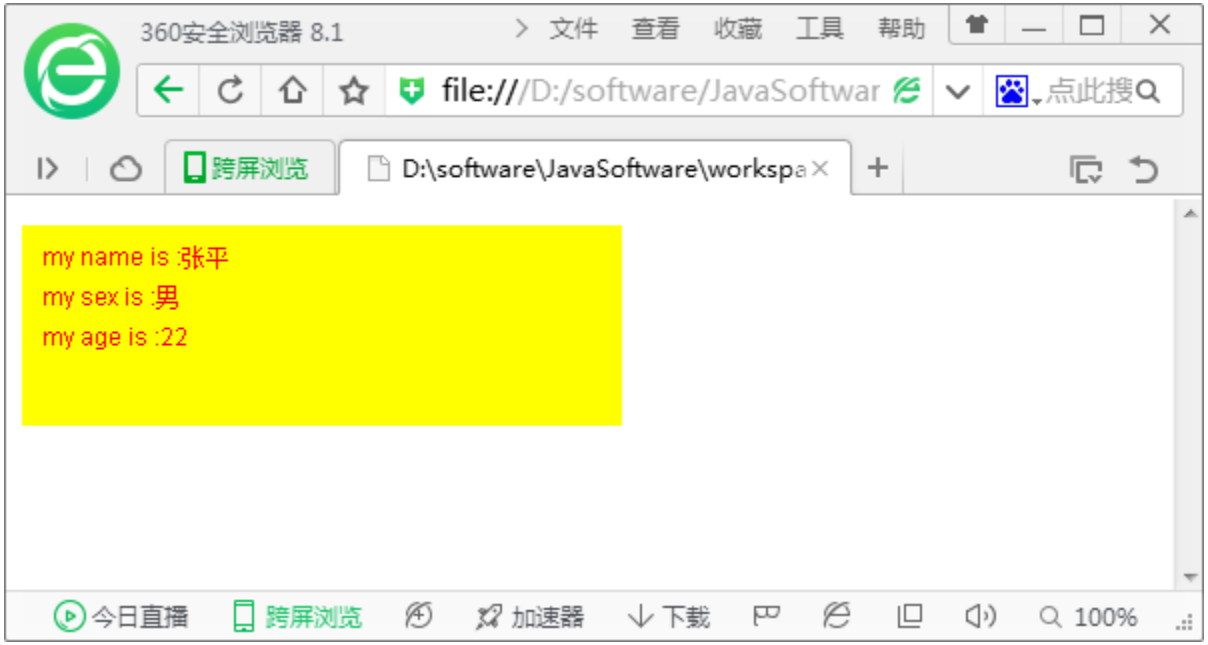
    getParameter("*appletParamter2*")

    ...

Applet属性与参数

【例9-4】向Applet传递参数

```
<html>
    <applet code="AppletParam.class" width="300" height="100">
      <Param Name=myName value="张平">
      <Param Name=mySex value="男">
      <Param Name=myAge value="22">
    </applet>
</html>
```

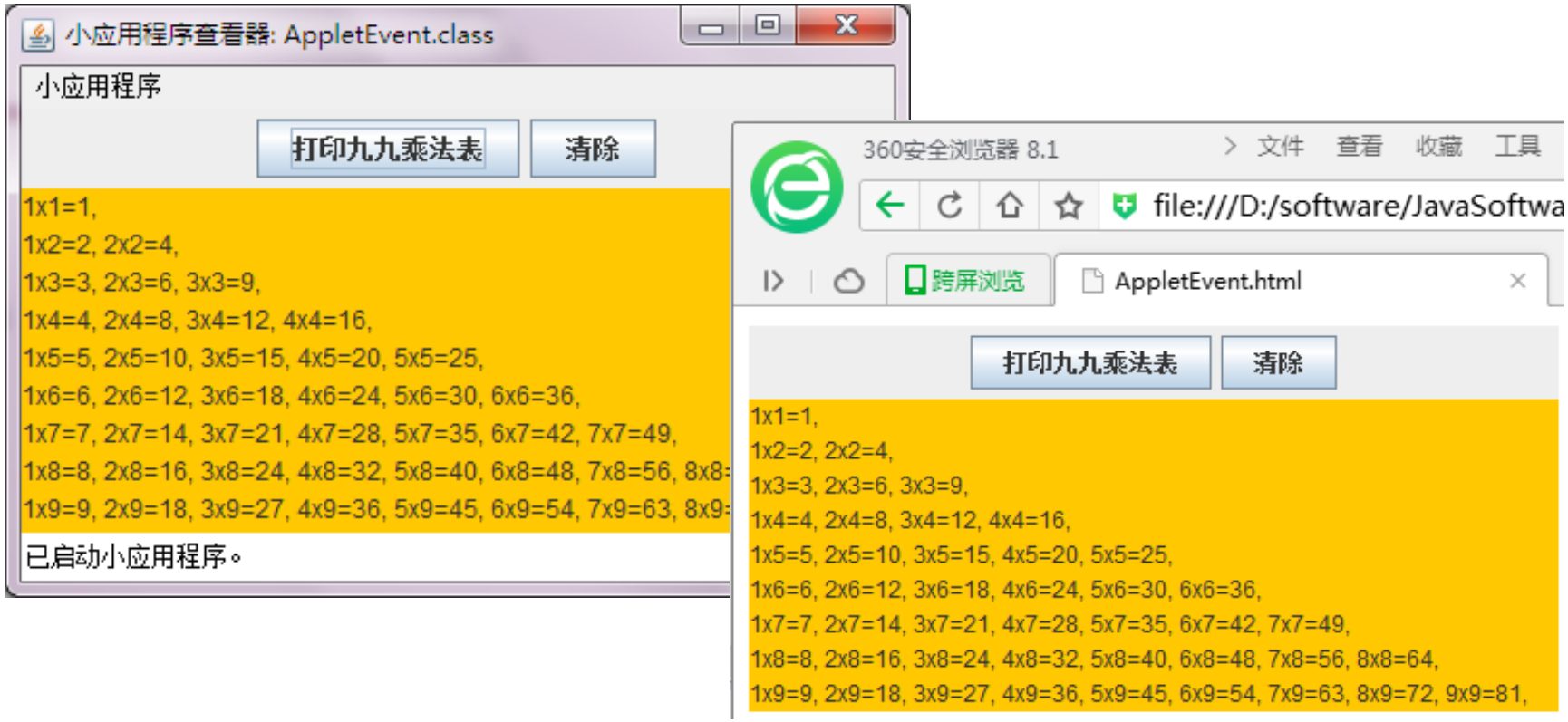


```
import java.awt.*;      import javax.swing.JApplet;
public class AppletParam extends JApplet {
   private String name;
   private String sex;
   private int age;
   public void init(){
         // 获取参数
         name=this.getParameter("myName");
         sex=this.getParameter("mySex");
         age=Integer.parseInt(this.getParameter("myAge"));
   }
   public void paint(Graphics g){
         g.setColor(Color.YELLOW);
         g.fillRect(0, 0, this.getWidth(), this.getHeight());
         g.setColor(Color.RED);
         g.drawString("my name is :"+ name, 10, 20);
         g.drawString("my sex is :"+ sex, 10, 40);
         g.drawString("my age is :"+ age, 10, 60);
   }
}
```

Applet中GUI设计及事件处理

- GUI设计与Application类似，基于Swing组件，但JApplet类是其顶层容器
- 事件处理机制与Application相同，采用监听方式

【例9-5】Applet中的事件处理



```
import java.awt.*;   import java.awt.event.*;  import javax.swing.*;

public class AppletEvent extends JApplet implements ActionListener {
   private JButton btnPrint,btnClear;
   private JTextArea ta;
   private JPanel panel;
   public void init(){
         btnPrint=new JButton("打印九九乘法表");
         btnPrint.addActionListener(this);  // 添加动作事件监听器
         btnClear=new JButton("清除");
         btnClear.addActionListener(this); // 添加动作事件监听器
         panel=new JPanel();
         panel.add(btnPrint);              panel.add(btnClear);
         ta=new JTextArea();
         ta.setBackground(Color.orange);

         this.add(panel,"North");          this.add(ta, "Center");
   }
```

```
    // 动作事件处理方法
    public void actionPerformed(ActionEvent e){
        JButton btn=(JButton) e.getSource();
        if(btn==btnPrint){
            String str="";
            for(int i=1;i<=9;i++){
                for(int j=1;j<=i;j++)
                    str+=j+"x"+i+"="+i*j+", ";
                str+="\n";
            }
            ta.setText(str);
        }
        else if(btn==btnClear)
            ta.setText(null);

    }
}
```

Java图形环境和图形对象

- Java图形处理分为两个层次
  - 处理原始图形，图形直接以点、线和面的形式画到界面上，这一层较原始
  - 提供大量组件（AWT/Swing），实现可定制的图形用户界面，这一层较高级
- 坐标
  - GUI组件的左上角坐标默认为（0，0）
  - 从左上角到右下角，水平坐标x和垂直坐标y增加
  - 坐标的单位是象素

Java图形环境和图形对象

- Graphics对象
  - 专门管理图形环境。Graphics类是一个抽象类
  - 抽象类Graphics提供了一个与平台无关的绘图接口
  - 各平台上实现的Java系统将创建Graphics类的一个子类，来实现绘图功能（其中包括线、圆和椭圆、矩形和多边形、图像以及各种字体的文本等），但是这个子类对程序员是透明的
  - 在执行paint方法时，系统会传递一个指向特定平台的Graphics子类的图形对象g

颜色和字体

- Java中有关颜色的类是Color类，它在java.awt包中，声明了用于操作颜色的方法和常量

名称	描述
final static Color GREEN	常量 绿色
final static Color RED	常量 红色
Color(int r, int g, int b) Color(int r, int g, int b, int a)	指定红色、蓝色、绿色、alpha分量（0～255），创建一种颜色。其中，当alpha分量为255时，表示完全不透明，为0时，表示完全透明
int getRed()	返回某颜色对象的红色分量值(0～255)
Graphics：void setColor(Color c)	Graphics类的方法，用于设置组件的颜色
Graphics：Color getColor()	Graphics类的方法，用于获得组件的颜色

颜色和字体

- Font类——有关字体控制，在java.awt包中

名称	描述
final static int PLAIN	一个代表普通字体风格的常量
final static int BOLD	一个代表**加粗**字体风格的常量
final static int ITALIC	一个代表*斜体*字体风格的常量
Font(String name,int style,int size)	用指定的字体、风格和大小创建一个Font对象
int getStyle()	返回一个表示当前字体风格的整数值
int getSize()	返回一个表示当前字体大小的整数值
String getName()	返回一个表示当前字体名称的字符串
Boolean isPlain()	测试一个字体是否是普通字体风格
Graphics：Font getFont()	获得当前字体
Graphics：setFont(Font f)	设置当前字体为f指定的字体、风格和大小

颜色和字体

【例9-6】获取系统可用字体，并显示在文本域内

```
import java.awt.Color;
import java.awt.Font;
import java.awt.GraphicsEnvironment;
import javax.swing.*;

public class SystemFont extends JFrame {
   private JTextArea ta;
   public SystemFont(){
      super("系统可用字体");
      this.setSize(520,320);
      this.setLocationRelativeTo(null); //把窗口置于显示屏的中央
      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     ta=new JTextArea();
      ta.setFont(new Font("黑体",Font.BOLD,20)); //设置默认字体
       ta.setBackground(new Color(90, 130, 180, 255));
      ta.setForeground(Color.YELLOW);

      JScrollPane sp = new JScrollPane(ta);
      this.add(sp,"Center");
```

```
    //获取系统可用的字体名
    GraphicsEnvironment
            ge=GraphicsEnvironment.getLocalGraphicsEnvironment();

        Font[ ] fonts=ge.getAllFonts();
        String str="";
        for (int j=0;j<fonts.length;j++){
            str+=(j+1) + ": "+ fonts[j].getFontName()+ "\n";
        }
        ta.setText(str);
    }

    public static void main(String[ ] args) {
            SystemFont   frm=new SystemFont();
            frm.setVisible(true);
    }

}
```

使用Graphics类绘图

- 使用Graphics类提供了用于绘制文本、线条、矩形、多边形、椭圆、弧等多种图形的方法

名称	描述
void drawString(String str, int x, int y)	绘制字符串，左上角的坐标是（x,y)
void drawLine(int x1, int y1, int x2, int y2)	在(x1,y1)与(x2,y2)两点之间绘制一条线段
void drawRect(int x, int y, int width, int height)	用指定的width和height绘制一个矩形，该矩形的左上角坐标为(x,y)
void fillRect(int x, int y, int width, int height)	用指定的width和height绘制一个实心矩形，该矩形的左上角坐标为(x,y)

使用Graphics类绘图

名称	描述
public void clearRect(int x, int y, int width, int height)	用指定的width和height，以当前背景色绘制一个实心矩形。该矩形的左上角坐标为（x,y)
public void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)	用指定的width和height绘制一个圆角矩形，圆角是一个椭圆的1/4弧，此椭圆由arcWidth、arcHeight确定两轴长。其外切矩形左上角坐标为（x,y)
public void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)	用当前色绘制实心圆角矩形，各参数含义同drawRoundRect。
public void draw3DRect(int x, int y, int width, int height, boolean b)	用指定的width和height绘制三维矩形，该矩形左上角坐标是(x,y)，b为true时，该矩形为突出的，b为false时，该矩形为凹陷的。
public void fill3DRect(int x, int y, int width, int height, boolean b)	用当前色绘制实心三维矩形，各参数含义同draw3DRect。

使用Graphics类绘图

名称	描述
void drawPolygon(int[] xPoints, int [] yPoints, int nPoints)	用xPoints，yPoints数组指定的点的坐标依次相连绘制多边形，共选用前nPoints个点
void fillPolygon(int[] xPoints, int [] yPoints, int nPoints)	绘制实心多边形，各参数含义同drawPolygon
void drawOval(int x, int y, int width, int height)	用指定的width和height，以当前色绘制一个椭圆，外切矩形的左上角坐标是(x,y)
void fillOval(int x, int y, int width, int height)	绘制实心椭圆，各参数含义同drawOval
void drawArc(int x, int y,int width, int height, int startAngle, int arcAngle)	绘制指定width和height的椭圆，外切矩形左上角坐标是(x,y)，但只截取从startAngle开始，并扫过arcAngle度数的弧线
void fillArc(int x, int y,int width, int height, int startAngle, int arcAngle)	绘制一条实心弧线（即扇形），各参数含义同drawArc

使用Graphics类绘图

【例9-7】用各种颜色绘制文字及各种图形

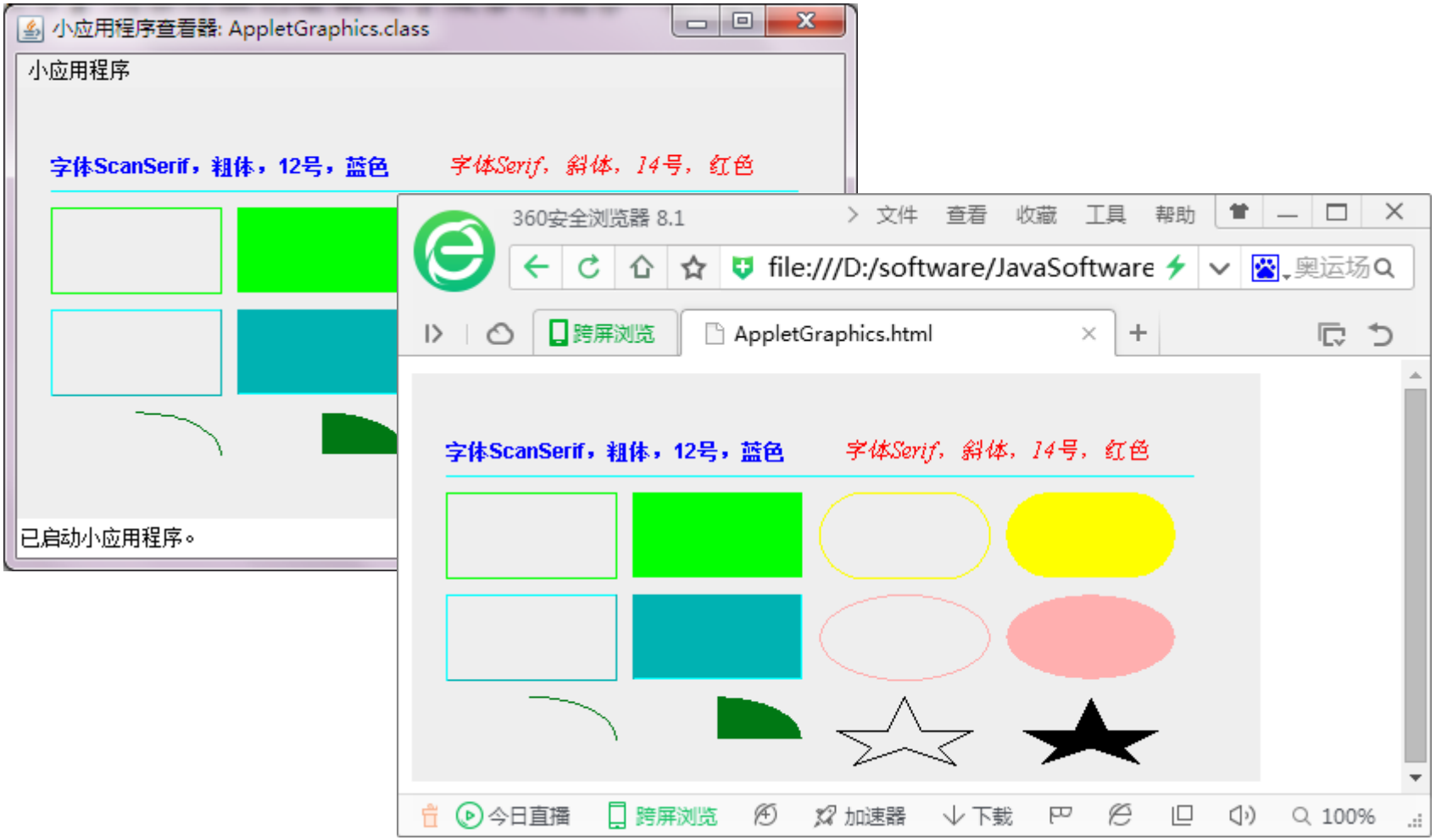
```
import java.awt.*;
import javax.swing.*;
public class GraphicsTester  extends JFrame {
   public GraphicsTester ()
   {  super("演示字体、颜色、绘图" );    //调用基类构造方法
      setVisible( true );                //显示窗口
      setSize( 480, 250 );               //设置窗口大小
   }
   public void paint( Graphics g )  {
       super.paint( g );  // call superclass's paint method
       g.setFont( new Font( "SansSerif", Font.BOLD, 12 ) );
       g.setColor(Color.blue);                          //设置颜色
       g.drawString("字体ScanSerif，粗体，12号，蓝色",20,50);

      g.setFont( new Font( "Serif", Font.ITALIC, 14 ) );
      g.setColor(new Color(255,0,0));
      g.drawString( " 字体Serif，斜体，14号，红色", 250, 50 );

      g.drawLine(20,60,460,60);                    //绘制直线
```

```
g.setColor(Color.green);
g.drawRect(20,70,100,50);                    //绘制空心矩形
g.fillRect(130,70,100,50);                   //绘制实心矩形

g.setColor(Color.yellow);
g.drawRoundRect(240,70,100,50,50,50);  //绘制空心圆角矩形
g.fillRoundRect(350,70,100,50,50,50);     //绘制实心圆角矩形

g.setColor(Color.cyan);
g.draw3DRect(20,130,100,50,true);       //绘制突起效果空心矩形
g.fill3DRect(130,130,100,50,false);       //绘制凹陷效果实心矩形
g.drawLine(20,60,460,60);                    //绘制直线

g.setColor(Color.pink);
g.drawOval(240,130,100,50);                 //绘制空心椭圆
g.fillOval(350,130,100,50);                  //绘制实心椭圆
```

```
        g.setColor(new Color(0,120,20));
        g.drawArc(20,190,100,50,0,90);          //绘制一段圆弧
        g.fillArc(130,190,100,50,0,90);         //绘制扇形

        g.setColor(Color.black);
        int xValues[]={250,280,290,300,330,310,320,290,260,270};
        int yValues[]={210,210,190,210,210,220,230,220,230,220};
        g.drawPolygon(xValues,yValues,10);      //绘制空心多边形

        int xValues2[]={360,390,400,410,440,420,430,400,370,380};
        g.fillPolygon(xValues2,yValues,10);     //绘制实心多边形
    }

public static void main( String args[ ] )  {
        GraphicsTester app = new GraphicsTester ();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE );
    }
}
```

使用Graphics类绘图

【例9-7'】用各种颜色绘制文字及各种图形—在Applet中实现



```
import java.awt.*;
import javax.swing.JApplet;

public class AppletGraphics extends JApplet {
    public void paint( Graphics g ) {
        super.paint( g ); // call superclass's paint method
        g.setFont( new Font( "SansSerif", Font.BOLD, 12 ) );
        g.setColor(Color.blue);                    //设置颜色
        g.drawString("字体ScanSerif，粗体，12号，蓝色",20,50);

       g.setFont( new Font( "Serif", Font.ITALIC, 14 ) );
       g.setColor(new Color(255,0,0));
       g.drawString( " 字体Serif，斜体，14号，红色", 250, 50 );

       ......// 代码同例9-6

       int xValues2[]={360,390,400,410,440,420,430,400,370,380};
       g.fillPolygon(xValues2,yValues,10);        //绘制实心多边形
    }
}
```

使用Graphics类显示图像

- 使用Graphics类的drawImage方法可以在swing组件(包括JApplet)上显示图像
  - boolean **drawImage**(Image img, int x, int y, ImageObserver observer) 图像按原始大小在指定位置显示
    - **img**：已加载的要显示的图像
    - **x、y**: 要显示的图像的矩形的左上角所处的位置
    - **observer**:多加载图像时的图像观察器
  - boolean **drawImage**(Image img, int x, int y, int width, int height, ImageObserver observer) 图像进行缩放后指定矩形区域内显示
    - **width、height**：矩形区域的大小

使用Graphics类显示图像

- **Application中获取图像Image：利用javax.swing.ImageIcon类**
  - ImageIcon icon=new ImageIcon("img/UPC.jpg");
  - Image img=icon.getImage();
- **Applet中获取图像Image：利用Applet的getImage方法**
  - Image **getImage**(URL url) **url 必须指定绝对 URL**
  - Image **getImage**(URL url, String name) **name 是相对于 url 参数的图象位置**
  - **例如**
    - **URL imgURL=new URL(getDocumentBase(),"img/UPC.jpg")**
    - **Image img=getImage(imgURL);**
  - **获取java.net.URL**
    - **Applet代码所在的URL：getCodeBase()方法**
    - **Applet嵌入的HTML文档所在的URL：getDocumentBase()方法**

使用Graphics类显示图像

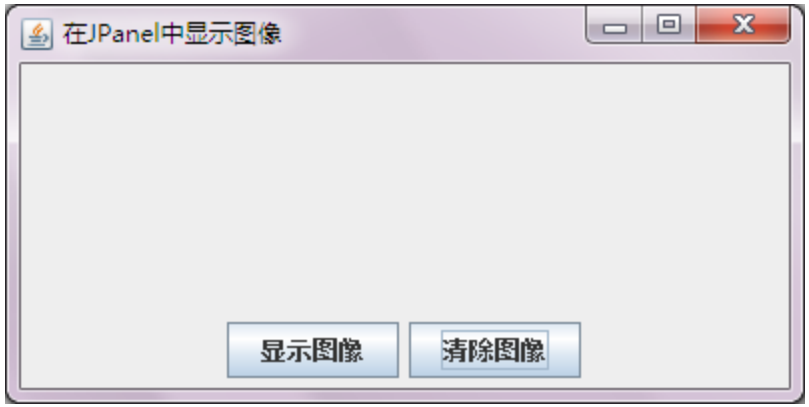
【例9-8】在JFrame中显示一幅图像


```
import java.awt.*;
import javax.swing.*;

public class ShowImage extends JFrame {
   private Image img;
   public ShowImage(){
          super("显示图像");
          this.setSize(400,200);
          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          ImageIcon icon=new ImageIcon("img/UPC.jpg");
          img=icon.getImage();
   }
```

```
public void paint(Graphics g){
        super.paint(g); //窗体重新绘制
        Insets a=this.getInsets(); //确定此容器的 insets，它指示容器边框的大小。例如，Frame 对象有一个顶端 inset，它对应于窗体的标题栏的高度。

        // 缩放到指定大小绘制 (注意,JFrame的左上角的绘图坐标是(0,0),客户区左上角坐标为(a.left,a.top)，而不是(0,0))
        //g.drawImage(img, a.left, a.top,this.getWidth()-a.right-a.left,this.getHeight()-a.top-a.bottom, this);
        g.drawImage(img,  a.left,  a.top,
                            this.getContentPane().getWidth(),
                            this.getContentPane().getHeight(),  this);
        //g.drawImage(img, a.left, a.top, this); // 按图像大小绘制
        g.setColor(Color.green);
        g.drawRect(a.left, a.top, this.getWidth()-a.left-a.right-1,
                                        this.getHeight()-a.top-a.bottom-1);
}

public static void main(String[] args) {
        ShowImage frm=new  ShowImage();
        frm.setVisible(true);
}
```

使用Graphics类显示图像

【例9-9】在JPanel中绘制或清除一幅图像




```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ShowImageInPanel extends JFrame
                                    implements ActionListener {
   private Image img;
   private JPanel  panelTop, panelBottom;
   private JButton  btnShow, btnClear;
   private boolean  showFlag=false;
```

```
public ShowImageInPanel(){
   super("在JPanel中显示图像");
   this.setSize(400,200);
   this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   ImageIcon icon=new ImageIcon("img/UPC.jpg");
   img=icon.getImage();

   panelTop=new JPanel();
   panelBottom=new JPanel();
   btnShow=new JButton("显示图像");
   btnClear=new JButton("清除图像");
   panelBottom.add(btnShow);
   panelBottom.add(btnClear);
   this.add(panelTop, "Center");
   this.add(panelBottom, "South");

   btnShow.addActionListener(this);
   btnClear.addActionListener(this);
}
```

```
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==(JButton)btnShow){
                showFlag=true;
                this.repaint(); //请求重新绘制窗体
        }else if(e.getSource()==(JButton)btnClear){
                showFlag=false;
                this.repaint(); //请求重新绘制窗体
        }
    }
    public void paint(Graphics g){
        super.paint(g); //窗体重新绘制（此时不会绘制我们的图像）
        if(showFlag) {  // 绘制图像
          Graphics g2=panelTop.getGraphics(); // 在JPanel上绘制图像
          g2.drawImage(img, 0, 0, panelTop.getWidth(),
                              panelTop.getHeight(), panelTop);
          g2.setColor(Color.green);
          g2.drawRect(0, 0,panelTop.getWidth()-1,panelTop.getHeight()-1);
        }
    }
    public static void main(String[] args) {
        ShowImageInPanel frm=new ShowImageInPanel();
        frm.setVisible(true);
    }
}
```

使用Graphics类显示图像

【例9-10】在JApplet中显示一幅图像


```
import java.awt.*;
import javax.swing.*;

public class ShowImageInApplet
                        extends JApplet {
   private Image img;
   public void init(){
      this.setSize(400,200);
      img=this.getImage(this.getDocumentBase(),"img/UPC.jpg");
   }
   public void paint(Graphics g){
          super.paint(g); //Applet重新绘制
          // 缩放到指定大小绘制图像
         g.drawImage(img, 0, 0,this.getWidth(),this.getHeight(), this);
          g.setColor(Color.green);
          g.drawRect(0, 0,this.getWidth()-1,this.getHeight()-1);
   }
}
```

使用Graphics2D类绘图

- **Java2D API**
  - 提供了高级的二维图形功能
  - 分布在**java.awt**、**java.awt.image**、**java.awt.color**、**java.awt.font**、**java.awt.geom**、**java.awt.print**和**java.awt.image.renderable**包中
  - 它能轻松使你完成以下功能：
    - 绘制任何宽度的直线
    - 用渐变颜色和纹理来填充图形
    - 平移、旋转、伸缩、切变二维图形，对图像进行模糊、锐化等操作

使用Graphics2D类绘图

- Graphics2D类
  - 是Graphics类的抽象子类
  - 要使用Java2D API，就必须建立该类的对象
  - 事实上，传递给paint方法的对象是Graphics2D的一个子类实例，被向上转型为Graphics类的实例。要访问Graphics2D功能，必须将传递给paint方法的Graphics引用强制转换为Graphics2D引用：

    Graphics2D g2d=(Graphics2D)g

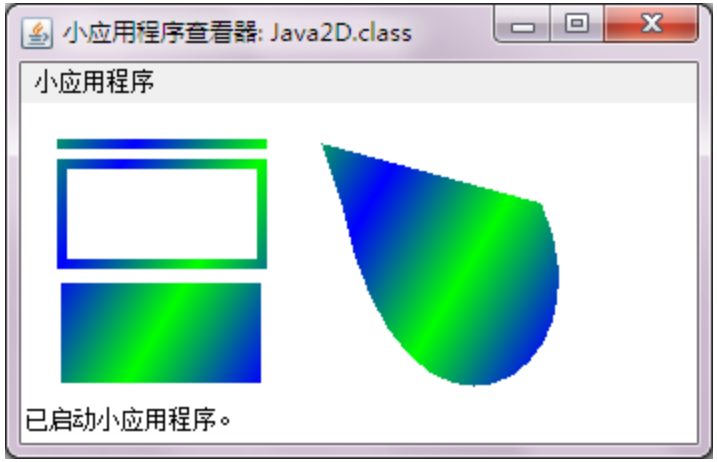
使用Graphics2D类绘图

【例9-11】使用Java2D使文字出现渐变色效果


```
import java.awt.*;
import javax.swing.*;
public class Graphics2DTester extends JApplet{
    public void paint(Graphics g) {
      super.paint(g);
      Graphics2D g2d=(Graphics2D)g;
      g2d.setFont(new Font("Georgia",Font.BOLD,30));
      g2d.setPaint(new GradientPaint(0,0,Color.red,180,45,Color.blue));
      g2d.drawString("This is a Java Applet!",25,50);
    }
}
```

使用Graphics2D类绘图

【例9-12】使用Java2D绘制指定线宽和渐变颜色的图形




```
import java.applet.Applet;
import java.awt.*;
import java.awt.geom.*; //Line2D等
public class Java2D extends Applet {
    public void paint(Graphics g){
        super.paint(g);
        Graphics2D g2d=(Graphics2D)g;
        //设置笔画宽度
        g2d.setStroke(new BasicStroke(5));
        //设置渐变颜色绘制或填充
        g2d.setPaint( new GradientPaint(0,0,Color.green,
                                        50,30,Color.blue,true) );
```

```
        //画线
        Line2D line = new Line2D.Float(20,20,120,20);
        g2d.draw(line);
        //画矩形
        Rectangle2D rect = new Rectangle2D.Float(20,30,100,50);
        g2d.draw(rect);
        rect = new Rectangle2D.Float(20,90,100,50);
        g2d.fill(rect);
        //画几何图形
        GeneralPath path = new GeneralPath();
        path.moveTo(150,20);
        path.lineTo(160,50);
        path.curveTo(190,200,300,140,260,50);
        //g2d.draw(path);
        g2d.fill(path);
    }
}
```

小　结

1. Applet的生命周期及其方法
2. Applet中的事件处理方法与Application相同
3. Java中使用Graphics类绘制图形和图像

习　题

1. P284习题9.1~9.3大题
2. 上机：
   a) 参考例9-5，在Applet上打印九九乘法表
   b) 在Applet的适当位置绘一个矩形，然后在该矩形内绘制你的头像，并在头像下方绘制你的学号和姓名（设置适当的字体）

谢谢大家！