北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Java图形程序设计

伍淳华
北京邮电大学网络空间安全学院

学习内容

- 采用Swing编写窗口程序;
- 如何在窗口中采用多种字体显示文本;
- 如何显示图像;

Swing概述

- 两种基本GUI程序设计类库
  - AWT(Abstract Window Toolkit)抽象窗口工具箱

    -对等体方法

    将处理用户界面元素的任务委派给每个目标平台的本地GUI工具箱，由本地GUI工具箱负责用户界面元素的创建和动作。

    “一次编写，随处使用”。

    “一次编写，到处调试”。

Swing概述

- 两种基本GUI程序设计类库
  - SWING

    -1996，Netscape创建了一种IFC(Internet Foundation Class)的GUI库，它将菜单、按钮等用户界面元素绘制在空白窗口上，而对等体只需创建和绘制窗口。

    在所有平台上的外观和动作都一样。

    -Sun和Netscape合作完善了这种方式，创建了一个名为Swing的用户界面库。

Swing概述

- 两种基本GUI程序设计类库
  - SWING vs. AWT
    - SWING显示用户界面的元素的速度比AWT慢一些；
    - SWING拥有一个丰富、便捷的用户界面元素集合；
    - SWING对低层平台的依赖很少，因此与平台相关的bug很少；
    - SWING给予不同平台的用户一致的感观效果；

Swing概述

- 两种基本GUI程序设计类库
  - SWING vs. AWT
    - SWING显示用户界面的元素的速度比AWT慢一些；
    - SWING拥有一个丰富、便捷的用户界面元素集合；
    - SWING对低层平台的依赖很少，因此与平台相关的bug很少；
    - SWING给予不同平台的用户一致的感观效果；
  - SWING没有完全替代AWT，而是基于AWT架构之上，其提供了能力更加强大的用户界面组件，但还需要使用基本的AWT事件处理。

GUI基本组成

- Java的图形用户界面的最基本组成成分是组件，组件是一个可以以图形化的方式显示在屏幕上并能与用户进行交互的对象，例如一个按钮，一个标签等。
- 组件不能独立地显示出来，必须将组件放在一定的容器中才可以显示出来。
- 容器（Container）实际上是Component的子类，因此容器本身也是一个组件，具有组件的所有性质，另外还具有容纳其他组件和容器的功能

创建框架

- 框架(frame)
  - 顶层窗口被称为框架
  - Swing用JFrame类来表示框架，该 类扩展于AWT的frame。
  - JFrame是极少数几个不绘制在画布上的Swing组件之一。其修饰部件（按钮、标题栏、图标等）由用户的窗口系统绘制，而不是由Swing绘制。

创建框架

- JFrame

```
java.long.Object
    |
    +----java.awt.Component
            |
        +----java.awt.Container
                |
            +----java.awt.Window
                        |
                    +----java.awt.Frame
                                |
                            +----javax.swing.JFrame
```

创建框架


```
import javax.swing.*;
import java.awt.*;
public class SimpleFrameTest {
   public static void main(String[] args) {
        EventQuene.involkLater(() ->
        {
       SimpleFrame  sFrame = new SimpleFrame();
       sFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       sFrame.setVisible(true);
       })
    }
}
```

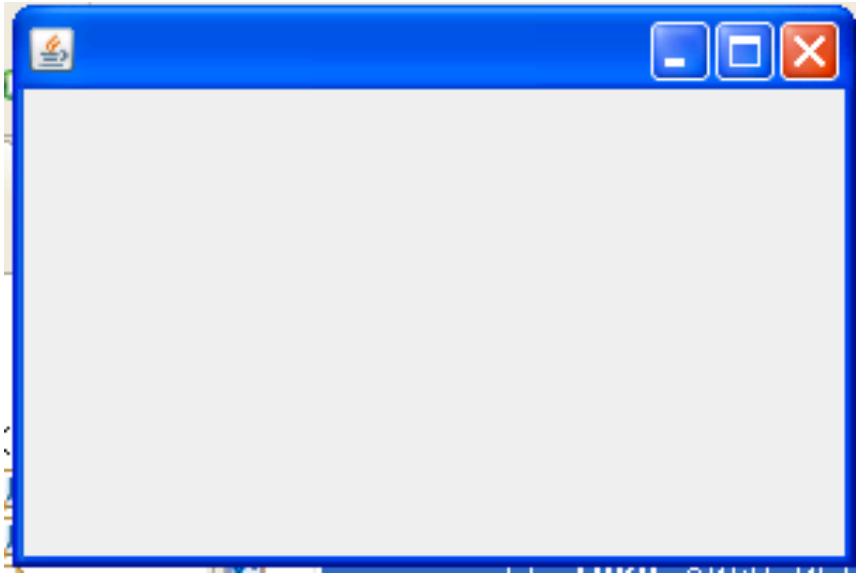
//所有的swing组件必须由事件分派线程进行配置，将鼠标点击和按键控制转移到用户接口组件。

创建框架


```
class SimpleFrame extends JFrame{
        public SimpleFrame(){
        setSize(DEFAULT_WIDTH,DEFAULT_HEIGHT);
    }
    public static int DEFAULT_WIDTH = 300;
    public static int DEFAULT_HEIGHT = 200;
}
```

创建框架

- 框架(frame)



创建框架

- 框架(frame)
  - 默认情况下,框架的大小为0*0象素;
  - 默认情况下,用户关闭窗口只是将框架隐藏了起来,程序并没有终止;
  - 构造一个框架并不自动显示,框架起初并不可见;

框架设置

- 设置合适的框架大小

  -获得用户系统的基于像素的屏幕分辨率信息,然后利用这些信息计算最佳的窗口大小。

  -获得用户系统的屏幕分辨率信息

```
Toolkit tk = Toolkit.getDefaultToolkit();
Dimension ds = tk.getScreenSize();
int width = ds.width;
int hight = ds.height;
```

框架设置

- 例：将一个可关闭框架设置为：
  - 其大小是整个屏幕的四分之一；
  - 位于屏幕的中央。

框架设置

- void setLocation(x,y);

  将框架放置在左上角水平x像素，垂直y像素的位置；坐标(0,0）位于屏幕的左上角；

- void setTitle(String s);

  设置框架的标题；

- void setIconImage(Image c);

  设置框架的图标；

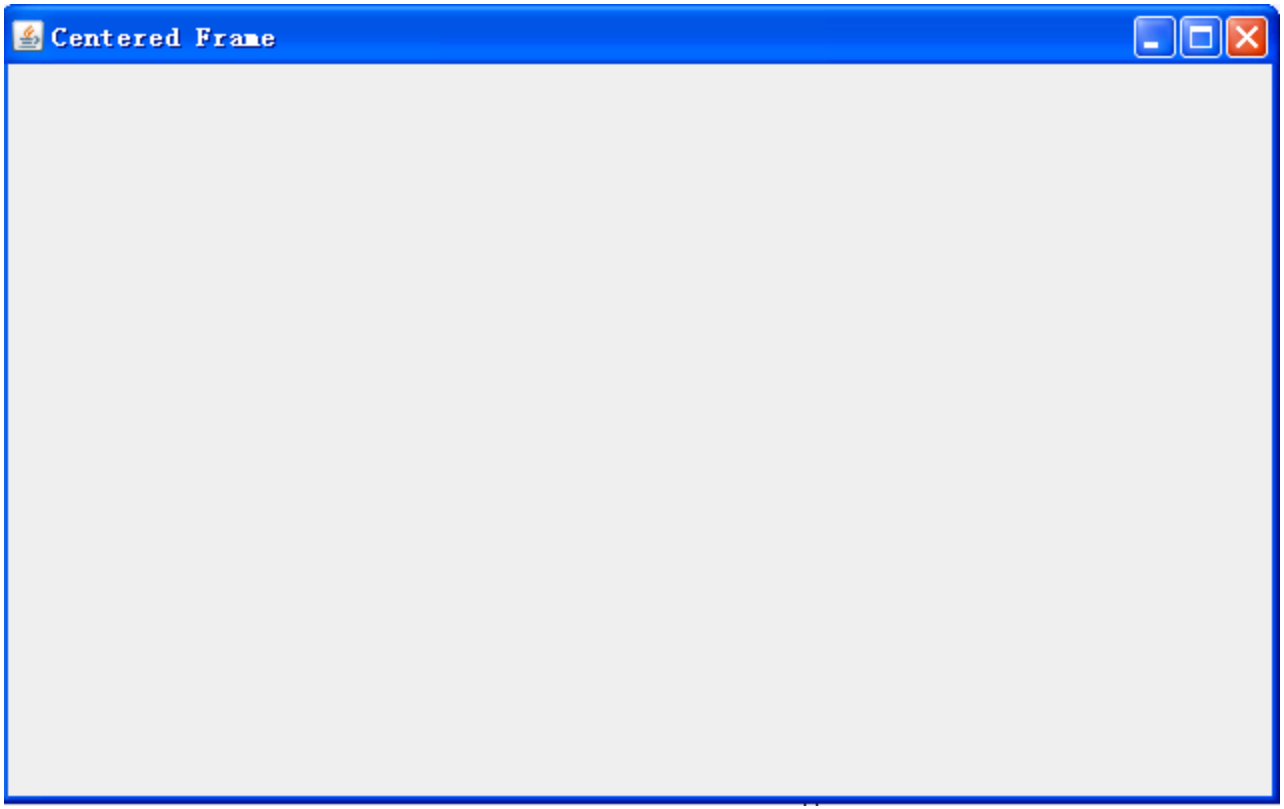
框架设置


```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class CenteredFrameTest {
   public static void main(String[]args){
        EventQuene.involkLater(() ->
        {
            CenteredFrame cf = new CenteredFrame();
          cf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          cf.setVisible(true);
        }
   }
}
```

框架设置


```
class CenteredFrame extends JFrame{
   public CenteredFrame(){
        Toolkit tk = Toolkit.getDefaultToolkit();
        Dimension ds = tk.getScreenSize();
        int width = ds.width;
        int hight = ds.height;

        setSize(width/2,hight/2);
        setLocation(width/4,hight/4);

        Image img = tk.getImage("icon.gif");
        setIconImage(img);
        setTitle("Centered Frame");
   }
}
```

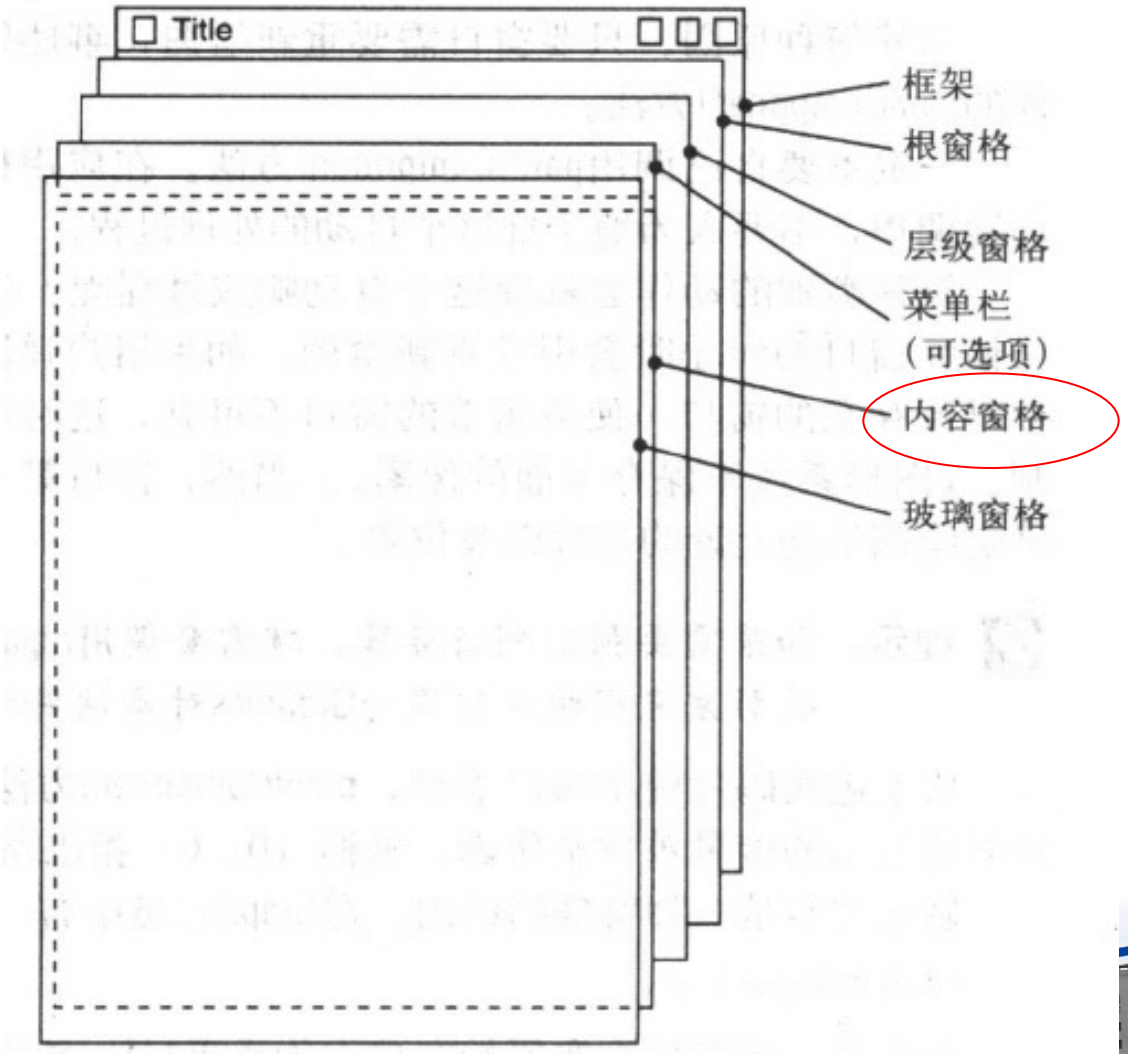
框架设置



在组件中显示信息

- 如何在框架中显示文本信息？
  - 可以直接在框架中绘制信息，但这不是一种良好的编程习惯；
  - 框架在JAVA中是一个容器，用来放置其它组件，如：放置一个按钮、一个文本框等；
  - 通常在一个组件上绘制信息，并将该组件添加到框架中；

- 如何在框架中添加一个组件？



- Swing程序员关心的是内容窗格,可使用下面的代码将组件添加到内容窗格中:

```
Container contentPane = frame.getContentPane();
Component c=new …;
contentPane.add(c);
```

- 在JDK 5.0后,可直接调用JFrame.add方法将组件添加到框架中,该方法实际上调用了内容窗格中的add方法

```
frame.add(c);
```

- 为了能够在框架中显示信息，需要进行
  - 定义一个扩展于JComponent的组件类，并将该组件加到框架中；
  - 在扩展的组件类中,覆盖paintComponent方法

- paintComponent方法
  - 该方法定义在JComponent类中,这个类是所有非窗口Swing组件的超类;
  - 该方法有一个Graphics类型的参数，通过该对象，可以进行绘制图案、图像和文本;
  - 程序员不用手工调用paintComponent方法，当应用程序需要绘图时(如窗口第一次出现、窗口大小调整等)，事件处理器就会通告组件，使得paintComponent自动调用。

- 绘制文本

  通过Graphics类中的drawString方法可在面板上绘制文本

```
g.drawString(text,x,y);
```

```
class NotHelloWorldComponent extends JComponent{
    public void paintComponent(Graphics g){
        ...
    g.drawString("Not a HelloWorld Program", MESSAGE_X, MESSAGE_Y);
    }
    public static final  int MESSAGE_X = 75;
    public static final  int MESSAGE_Y = 100;
}
```

- 绘制文本

  组件需要确定自己的大小，覆盖getPreferredSize方法，返回一个有首选宽度和高度的Dimension类对象；

```
class NotHelloWorldComponent extends JComponent{
        private static final int DEFAULT_WIDTH = 300;
        private static final int DEFAULT_HEIGHT = 200;
        ….
     public Dimension getPreferredSize(){
     return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
     }
}
```

- 绘制文本

框架中嵌入组件时，如果只想使用其首选大小，可以调用pack方法，而不用调用setSize方法：

```
class NotHelloWorldFrame extends JFrame{
    public HelloWorldFrame(){
        add(new NotHelloWorldComponent());
            pack();
    }
}
```

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

public class NotHelloWorldFrameTest {
   public static void main(String[] args){
        EventQuene.involkLater(() ->
           {
                JFrame f = new NotHelloWorldFrame();
                f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                f.setVisible(true);
                f.setTitle( “Not HelloWorld” );
            } );
        }
   }
```

```
class NotHelloWorldFrame extends JFrame{
        public HelloWorldFrame() {
              add(new NotHelloWorldComponent());
                 pack();
        }
   }
```

```
class NotHelloWorldComponent extends JComponent{
    public void paintComponent(Graphics g){
      g.drawString("Not a HelloWorld Program", MESSAGE_X, MESSAGE_Y);
    }
    public Dimension getPreferredSize(){
      return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
    }

    private static final int DEFAULT_WIDTH = 300;
    private static final int DEFAULT_HEIGHT = 200;

    public static final  int MESSAGE_X = 75;
    public static final  int MESSAGE_Y = 100;
}
```

面板与信息显示

2D图形

- **图形绘制主要有两个类**
  - **Graphics(JDK 1.0):绘制图形的能力有限;**
  - **Graphics2D(JDK 1.2)：Graphics 的子类，绘制图形的能力强大;**
- **如果使用的是支持Java 2D的版本，paintComponent方法可自动地获得一个Graphics2D类对象，只需进行一次类型转换**

```
public void paintComponent(Graphics g)
{
    Graphics2D g2 = (Graphics2D)g;
    ……
}
```

2D图形

- **常用的几何图形类**
  - **Line2D**
  - **Rectangle2D**
  - **Ellipse2D**

  ….

  这些几何图形类实现了Shape接口
- **绘制图形**
  - **首先创建一个实现了Shape接口的类对象；**
  - **调用Graphics2D类中的draw方法；**

```
Rectangle2D rect = …;
g2.draw(rect);
```

2D图形

- **Java2D图形类采用的是浮点坐标；**
- **每个图形类有两个版本：一个是为节省空间设置的float类型的坐标；另一个是double类型的坐标；**
  - 例： Rectangle2D类有两个子类
    Rectangle2D.Float
    Rectangle2D.Double

```
Rectangle2D.Float fRect = new Rectangle2D.Float(10.0f,25.0f,22.5f,20.0f);
Rectangle2D.Double dRect = new Rectangle2D.Double(10.0,25.0,22.5,20.0)
```

2D图形

- **Rectangle2D.Float和 Rectangle2D.Double均继承于Rectangle2D,并且子类只覆盖了Rectangle2D超类中的方法，故没有必要记住图形类型，可直接使用Rectangle2D变量保存矩形的引用：**

```
Rectangle2D fRect = new Rectangle2D.Float(10.0f,25.0f,22.5f,20.0f);
Rectangle2D dRect = new Rectangle2D.Double(10.0,25.0,22.5,20.0);
```

- **对Rectangle2D的论述适用于其它所有的2D图形类,如Point2D也有两个子类Point2D.Float和Point2D.Double,子类也仅实现了父类的方法,可直接使用父类变量保存子类对象的引用。**

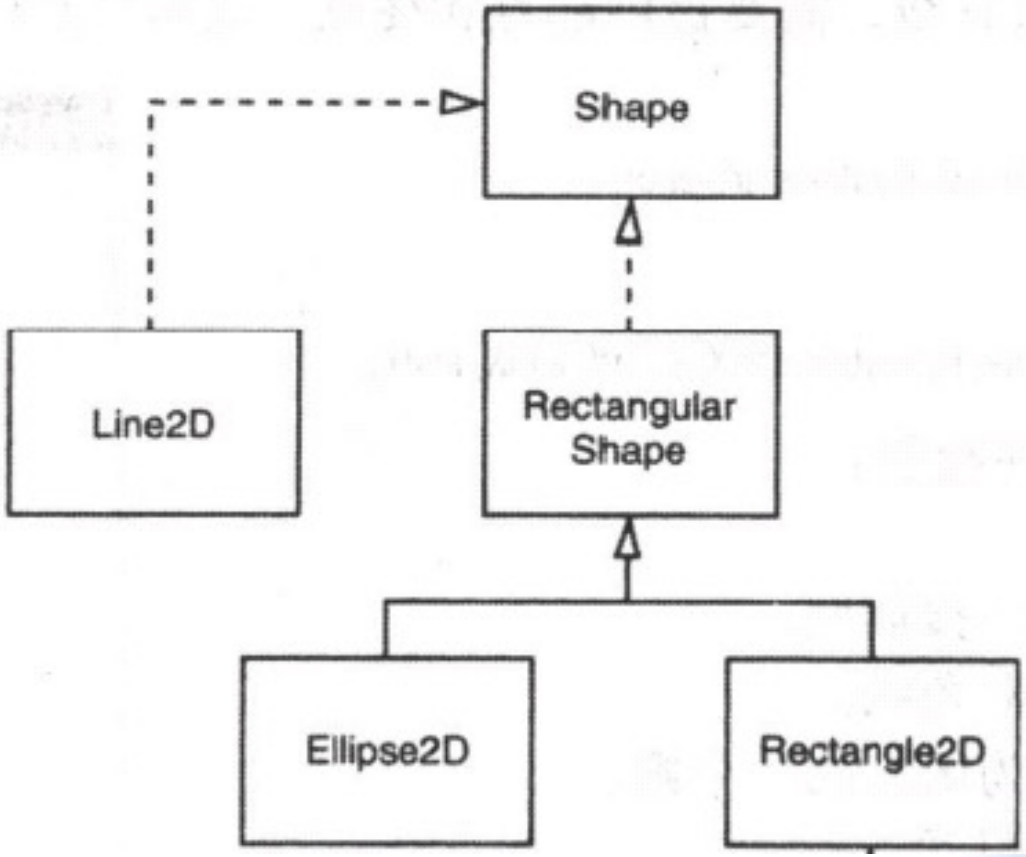
2D图形

- **常用2D图形类**
  - abstract RectangularShape
    几何框架类图形的父类;

```
public double getWidth( );
public double getHeight( );
public  double getCenterX( );
public double getCenterY( );
```

2D图形

- **常用2D图形类**



2D图形

- **常用2D图形类**
  - Rectangle2D.Double

    Rectangle2D.Double(double x,double y,double w,double h)

    x,y---矩形左上角的坐标(x,y);

    w ---矩形宽度

    h ---矩形高度

    Rectangle2D.Double( )

    x=y=w=h=0

2D图形

- **常用2D图形类**
  - Rectangle2D.Double

如果已知两个矩形的两个对角点，也可以依此构造一个矩形，假设此两点的坐标分别为p(px,py),q(qx,qy):

```
Rectangle2D rect = new Rectangle2D.Double( );
rect.setFrameFromDiagonal(px,py,qx,qy);
```

或 rect.setFrameFromDiagonal(p,q); (p,q是用Point2D对象表示的两个对角点)

```
public void setFrameFromDiagonal(double x1,double y1,double x2,double y2);
public void setFrameFromDiagonal(Point2D p1, Point2D p2);
```

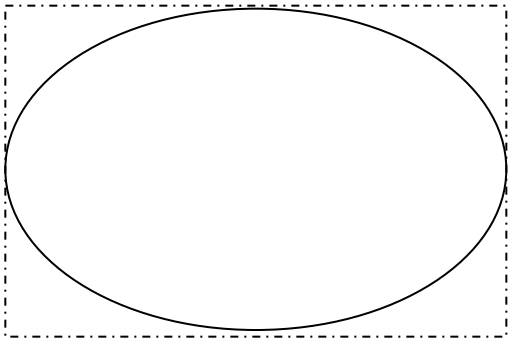
- **常用2D图形类**
  - Ellipse2D.Double

    Ellipsee2D.Double(double x,double y,double w,double h)

    x,y---椭圆所在矩形外框左上角的坐标(x,y);

    w ---椭圆所在矩形外框矩形宽度

    h ---椭圆所在矩形外框矩形高度



Ellipse2D.Double( )

椭圆所在矩形外框为x=y=w=h=0

- **常用2D图形类**
  - Ellipse2D.Double

通过椭圆中心点c(centerX,centerY)、宽和高来构造一个椭圆：

```
Ellipse2D ellipse=new Ellipse2D.Double(certerX-
width/2,centerY-height/2,width,height);
```

通过椭圆中心点及外矩形框中四个顶点中的一个来构造椭圆：

```
Ellipse2D ellipse = new Ellipse2D.Double( );
ellipse.setFrameFromCenter(centerX,centerY,cornerX,cornerY);
```

通过外矩形框来构造一个椭圆：

```
Ellipse2D ellipse = new Ellipse2D.Double( );
ellipse.setFrame(rect);
```

2D图形

- **常用2D图形类**
  - Line2D.Double

    Line2D.Double( );

    Line2D.Double(double x1, double y1, double x2, double y2);

    Line2D.Double(Point2D p1, Poiont2D p2;)

2D图形

- **常用2D图形类**
  - Point2D.Double

    Point2D.Double( );

    Point2D.Double(double x, double y);

2D图形

- **绘制图形**
  - **一个矩形**
  - **这个矩形的内接椭圆**
  - **矩形的一条对角线**
  - **以矩形中心为圆心的一个圆**

2D图形


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

public class DrawTest {
   public static void main(String[]args){
    EventQuene.involkLater(() ->
      {
            JFrame f = new DrawFrame();
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            f.setVisible(true);
            f.setTitle( “Draw Test” );
       } );
   }
}
```

2D图形


```
class DrawFrame extends JFrame{
   public DrawFrame(){
         add(new DrawComponent());
            pack();
     }
}
```

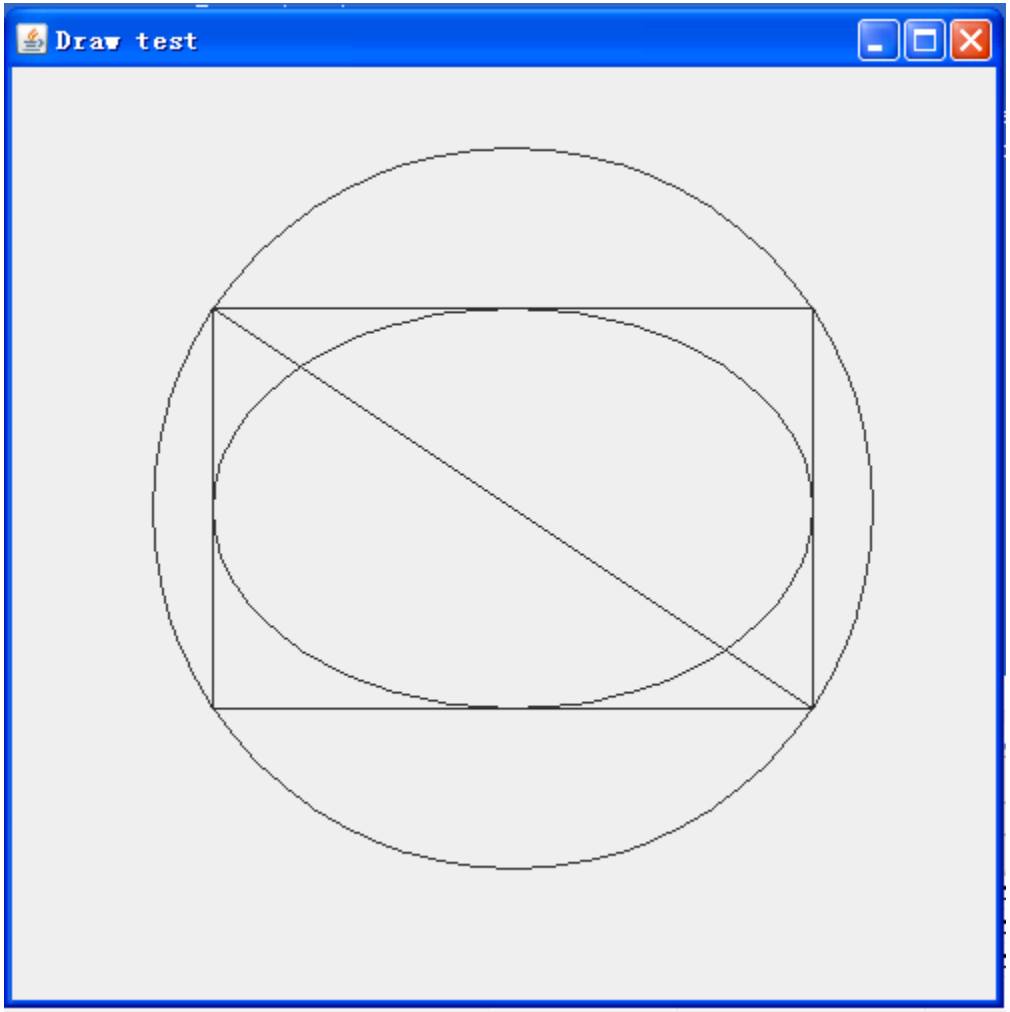
2D图形


```
class DrawComponent extends JComponent{
   public void paintComponent(Graphics g){
     Graphics2D g2 = (Graphics2D)g;

     int leftX = 100;
     int leftY = 120;
     int width = 300;
     int height = 200;

     Rectangle2D  rec = new
                    Rectangle2D.Double(leftX,leftY,width,height);
     g2.draw(rec);

     Ellipse2D eps = new Ellipse2D.Double();
     eps.setFrame(rec);
     g2.draw(eps);
```

2D图形


```
        g2.draw(new
                    Line2D.Double(leftX, leftY, leftX+width, leftY+height));
        double x = eps.getCenterX();
        double y = eps.getCenterY();
        double radius = 180;

        Ellipse2D circle = new Ellipse2D.Double();
        circle.setFrameFromCenter(x, y, x+radius, y+radius);
        g2.draw(circle);
        }
        public Dimension getPreferredSize(){
          return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
    }
  private static final int DEFAULT_WIDTH = 400;
  private static final int DEFAULT_HEIGHT = 400;
}
```

2D图形



颜色

- **Java.awt.Color类用于定义颜色**
  - **类中提供了13个预定义的常量，分别表示13种标准颜色**

BLACK GREEN RED BLUE
WHITE LIGHT_GRAY CYAN MAGENTA
YELLOW DARK_GRAY ORANGE GRAY
PINK

Color.RED
Color.GREEN

颜色

- **Java.awt.Color类用于定义颜色**
  - **利用三原色（红、绿和蓝）来创建一个Color对象，三种颜色都是用(0~255)之间的数值来表示；**

public Color(int red, int green, int blue);

颜色

- **Graphics2D类的setPaint方法可以为图形环境上的所有后续的绘制操作选择颜色。绘制颜色的过程：**
  - **选择颜色**
  - **绘制图形**

```
g2.setPaint(Color.RED);
g2.drawString("Warning!",100,100);
```

```
void setPaint(Paint p);
```

----Color类实现了Paint接口，

颜色

- **设置组件的背景颜色和前景颜色**

  Component 类中的两个方法

  - **public void setBackground(Color c);**
  - **public void setForeground(Color c);**

```
MyComponent p = new MyComponent( );
p.setBackground(Color.PINK);
```

颜色

- **填充图形**

  可选用一种颜色，填充闭合图形的内部，调用Graphics2D的fill方法；

  – **public void fill(Shape s);**

```
Rectangle2D rect = …;
g2.setpaint(Color.RED);
g2.fill(rect);
```

颜色

- **例：用红色填充一个矩形，并用暗绿色填充该矩形的内接椭圆。**

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;
public class FillTest {
   public static void main(String[] args){
         EventQuene.involkLater(() ->
          {
               JFrame f = new FillFrame();
               f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
               f.setVisible(true);
               f.setTitle( “Fill Window” );
          }); }
}
```

颜色


```
class FillFrame extends JFrame{
    public FillFrame(){
        add(new FillComponent());
            pack();
    }
}
```

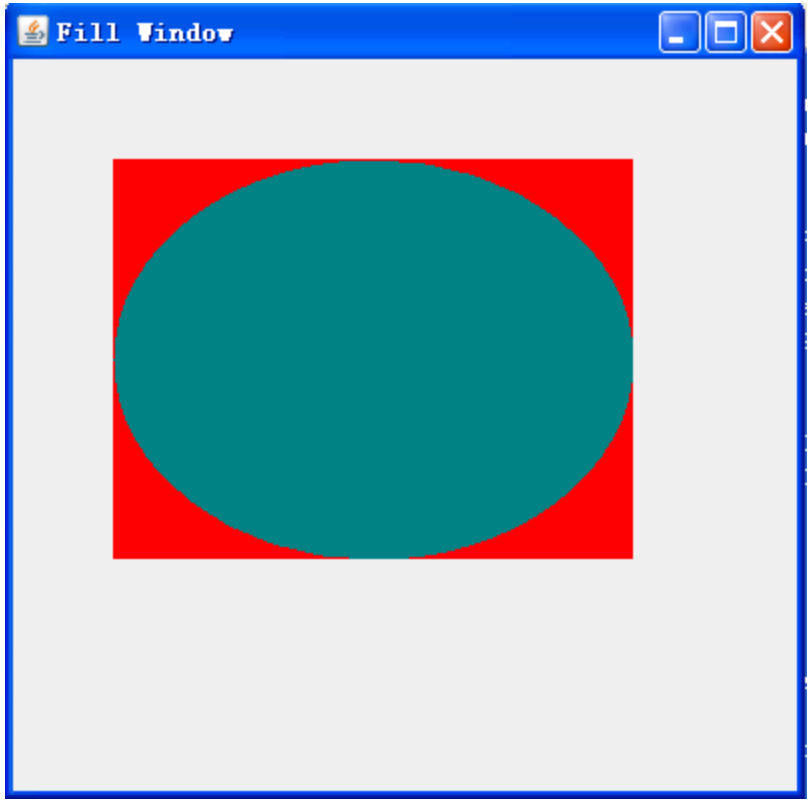
颜色


```
class FillComponent extends JComponent{
   public void paintComponent(Graphics g){
      Graphics2D g2 = (Graphics2D)g;
      double leftX = 50;
      double leftY = 50;
      double width = 260;
      double height = 200;
     Rectangle2D rec = new Rectangle2D.Double(leftX, leftY, width,
  height);
      g2.setPaint(Color.RED);
      g2.fill(rec);
    Ellipse2D ese = new Ellipse2D.Double();
    ese.setFrame(rec);
    g2.setPaint(new Color(0,128,128));
    g2.fill(ese);
}
```

颜色


```
    public Dimension getPreferredSize(){
        return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
    }

    private static final int DEFAULT_WIDTH = 300;
    private static final int DEFAULT_HEIGHT = 200;
}
```

颜色



设置字体

- **java.awt.Font**
  - **Font(Stirng name, int style, int size)**

    -name:字体名;

    -style:字体风格

    Font.PLAIN :常规

    Font.BOLD :粗体

    Font.ITALIC:斜体

    Font.BOLD+Font.ITALIC:粗体+斜体

    -size:字体大小,以点为单位,每英寸包含72个点

Eg: Font serif14 = new Font(“Serif”,Font.BOLD,14);

设置字体

- **java.awt.Font**
  - **字体名是指一台机器上所允许使用的字体的名字;不同的机器所支持的字体有所不同;**
  - **列出机器上所允许的字体名**
    - **调用GraphicsEnvironment类中的getAvailableFontFamilyNames方法,该方法返回一个字符型数组,包含了所有可用的字体名;**
    - **GraphicsEnvironment类描述了用户系统的图形环境,可调用静态的getLocalGraphicsEnvironment方法得到该类的一个对象;**

设置字体


```
import java.awt.*;
public class ListFonts {
    public static void main(String[]args){
        String[] fontsName =
   GraphicsEnvironment.getLocalGraphicsEnvironment().getAva
   ilableFontFamilyNames();
         For(String s:fontsName)
              System.out.println(s);
     }
}
```

设置字体

Algerian

Arial

Arial Black

Arial Narrow

Arial Unicode MS

Baskerville Old Face

Batang

BatangChe

……..

设置字体

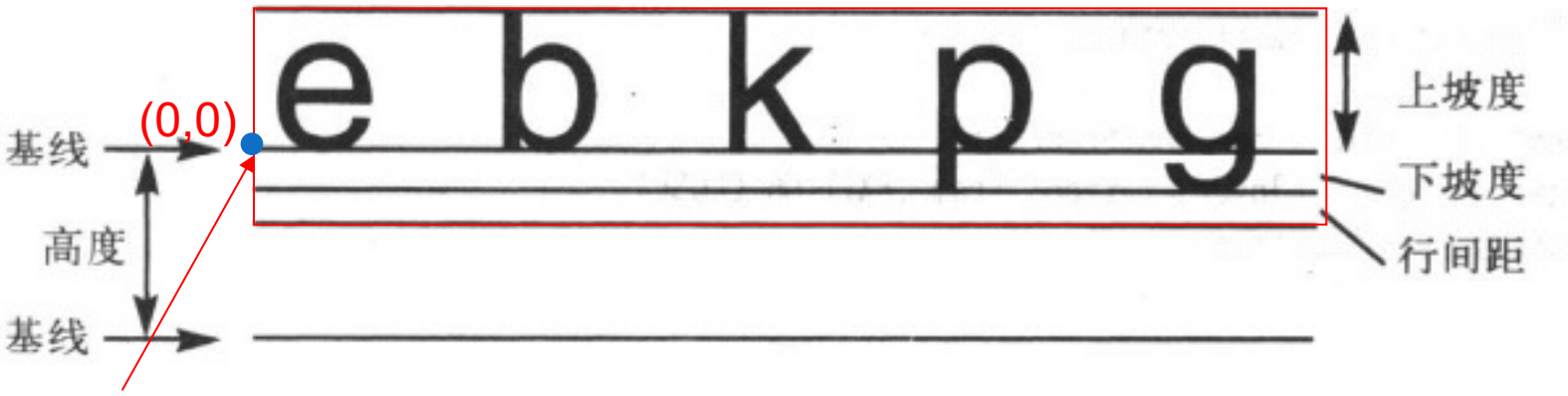
- **java.awt.Font**
  - **JAVA AWT定义了五个逻辑(logical)字体名:**
    - **SansSerif**
    - **Serif**
    - **Monospaced**
    - **Dialog**
    - **DialogInput**

      这五种字体会被映射到客户机上的实际字体。不同的机器映射方式可能不同。

设置字体

- **设置字符串位置**
  - **获得描述字符串位置的矩形**

```
FontRenderContext context = g2.getFontRenderContext( );
Rectangle2D bounds = f.getStringBounds(message,context);
```



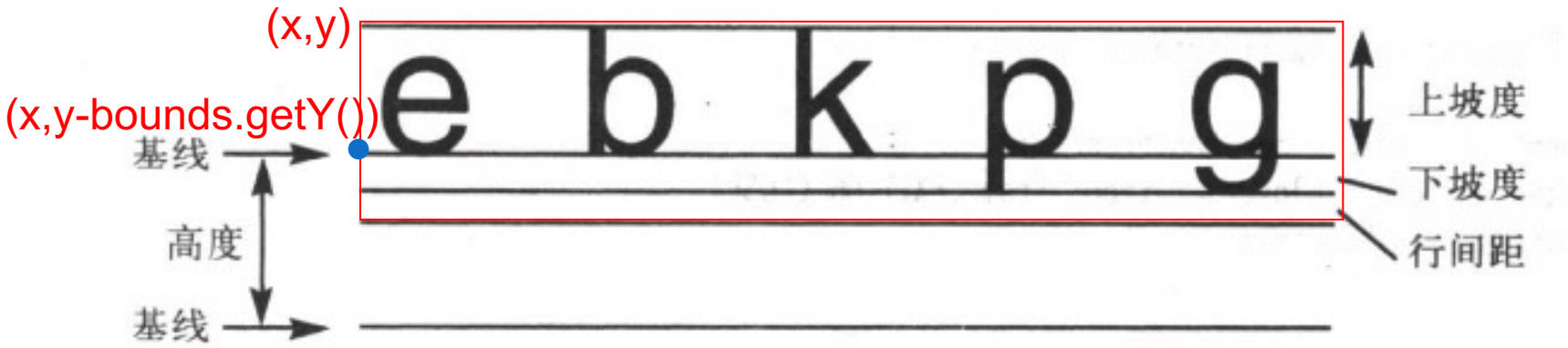
-通过该矩形的getWidth( )和getHeight( )可以得到字符串所占的宽度和高度

-该矩形的坐标原点在字符串的基线开始的地方，故通过getY( )得到的是上坡度的负值；

-drawString(s, x, y)方法中参数标识的点就是基线开始的点在屏幕中的坐标位置

设置字体

- **设置字符串位置**
  - **设置字符串的位置，即是求得基准线起始点在屏幕中的坐标值，该值以屏幕左上角(0,0)为基准**



设置字体

- **将字符串写在面板的中央**

```
FontRenderContext frc = g2.getFontRenderContext();
Rectangle2D rec = f.getStringBounds(s,frc);

double x = (this.getWidth()-rec.getWidth())/2;
double y = (this.getHeight()-rec.getHeight())/2;
double ascent = -rec.getY();
g2.drawString(s, (int)x, (int)(y+ascent));
```

设置字体

- **例：将字符串写在面板的中央**

```
import java.awt.*;
import javax.swing.*;
import java.awt.geom.*;
import java.awt.font.*;
public class FontTest {
  public static void main(String[]args){
     EventQuene.involkLater(() ->
        {
              JFrame f = new FontFrame();
              f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
              f.setVisible(true);
              f.setTitle( “Font Window” );
          });}
}
```

设置字体

- **例：将字符串写在面板的中央**

```
class FontFrame extends JFrame{
   public FontFrame(){
                add(new FontComponent());
                pack();
    }

}
```

设置字体

- **例：将字符串写在面板的中央**

```
class FontComponent extends JComponent{
    public void paintComponent(Graphics g){
        Graphics2D g2 = (Graphics2D)g;
        String s = new String("Hello World, angle!");
        Font f = new Font("Serif",Font.BOLD+Font.ITALIC,30);
        g2.setFont(f);

        FontRenderContext frc = g2.getFontRenderContext();
        Rectangle2D rec = f.getStringBounds(s,frc);
        double x = (this.getWidth()-rec.getWidth())/2;
        double y = (this.getHeight()-rec.getHeight())/2;
        double ascent = -rec.getY();
        g2.drawString(s, (int)x, (int)(y+ascent));
```

设置字体

- **例：将字符串写在面板的中央**

```
g2.setPaint(Color.GRAY);
g2.draw(new Line2D.Double(x,y+ascent, x+rec.getWidth(), y+ascent));
g2.draw(new Rectangle2D.Double(x,y, rec.getWidth(),
rec.getHeight()));
}
    public Dimension getPreferredSize(){
       return new Dimensi                    _HEIGHT);
    }

        private static f                      300;
        private static f                      200;
}
```

图像

- **读取图像文件**
  - **表示图像文件的类**

    java.awt.Image;

    java.awt.image.BufferedImage;
  - **读图像**

    Image image = new ImageIcon(filename).getImage();

图像

- **显示图像**

java.awt.Graphics

boolean drawImage(Image ima, int x, int y, imageObserver observer)

boolean drawImage(Image ima, int x, int y, int width, int height, ImageObserver observer)

例:

```
public void paintComponent(Graphics g){
    …;
    g.drawImage(image,x,y,null);
}
```

图像

- **例：平铺显示图像**

```
import javax.swing.*;
import java.io.*;
import java.awt.geom.*;
import java.awt.*;
import javax.imageio.*;
public class ImageTest {
   public static void main(String[]args){
       EventQuene.involkLater(() ->
        {
              JFrame f = new ImageFrame();
              f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
              f.setVisible(true);
              f.setTitle( “Image Window” );
          });          }
```

图像

- **例：平铺显示图像**

```
class ImageFrame extends JFrame{
   public ImageFrame(){
                add(new ImageComponent());
                 pack();
         }
}
```

- **例：平铺显示图像**

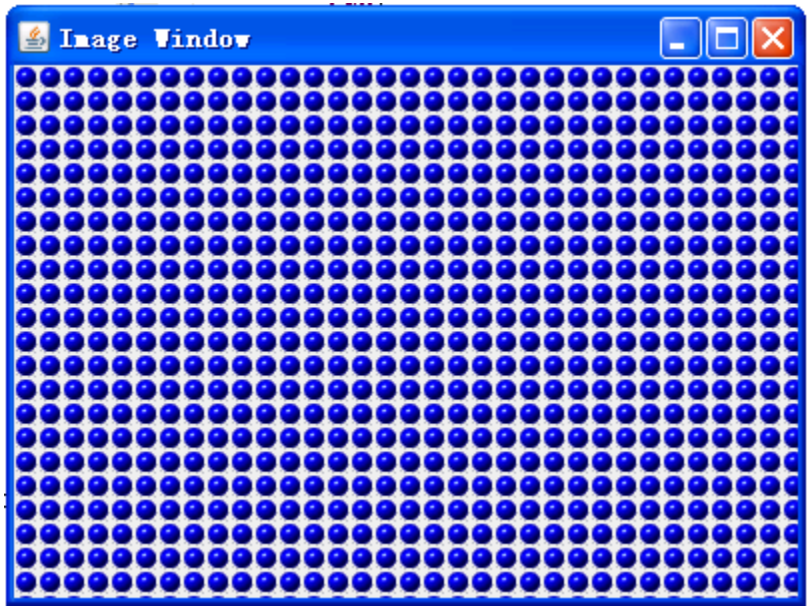
```
class ImageComponent extends JComponent{
   public ImageComponent(){
        ima = new ImageIcon( “blue-ball.gif” ).getImage();
   }
    public Dimension getPreferredSize(){
         return new Dimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);
     }
   public void paintComponent(Graphics g){
        Graphics2D g2 = (Graphics2D)g;
        if(ima == null)return;

            int imageWidth = ima.getWidth(this);
        int imageHeight = ima.getHeight(this);
```

- **例：平铺显示图像**

```
    g2.drawImage(ima,0,0,null);

    for(int i = 0; i*imageWidth < this.getWidth(); i++)
       for(int j = 0; j*imageHeight <this.getHeight(); j++){
                  if ( i+ j >0)
            g2.copyArea(0, 0, imageWidth, imageHeight,
                        i*imageWidth, j*imageHeight);
            }
    }
    private Image ima;
        private static final int DEFAULT_WIDTH = 300;
       private static final int DEFAULT_HEIGHT = 200;
}
```

图像

原图像



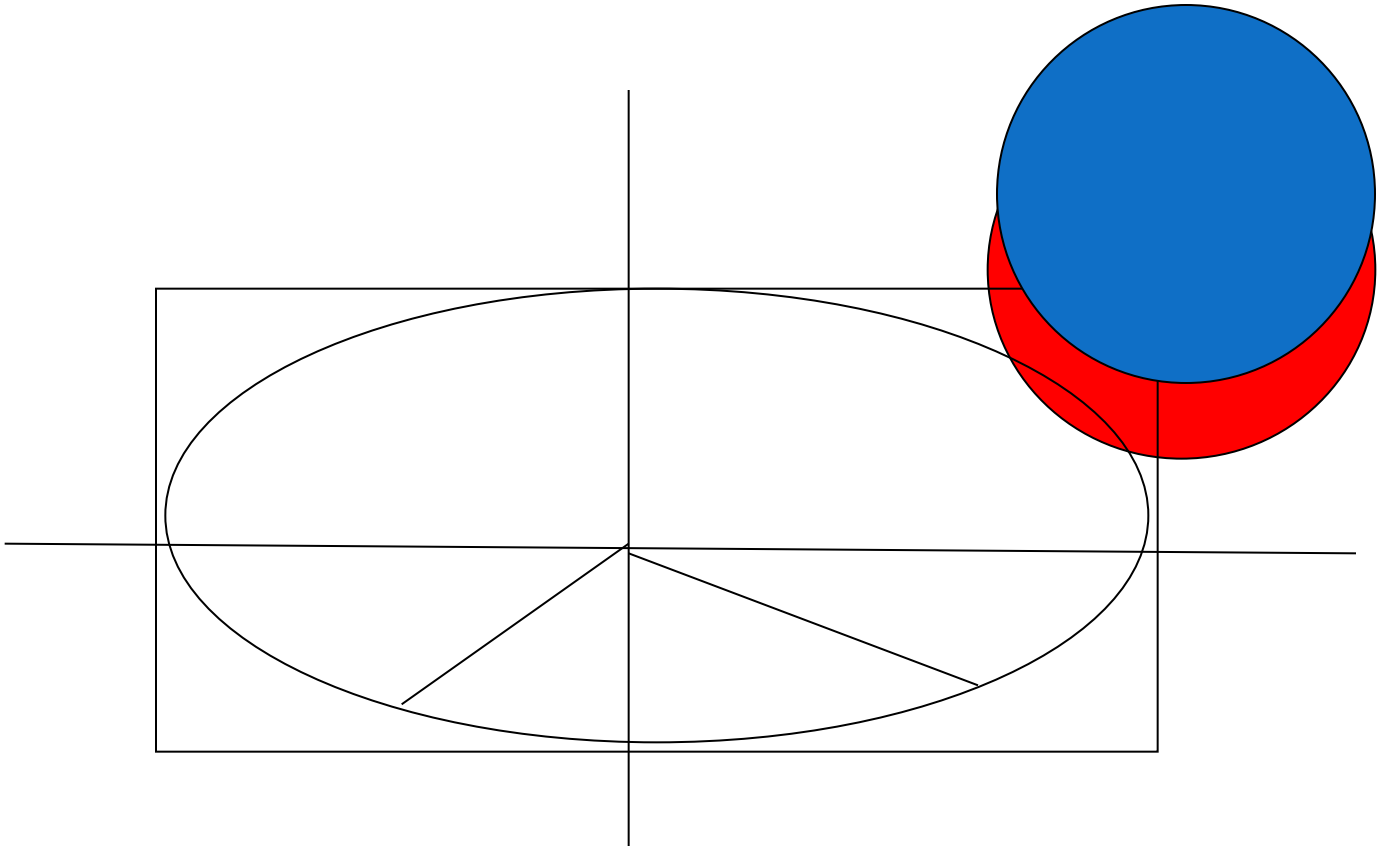
平铺后得到的图像

小结

- 框架的显示;
- 如何在窗体中绘制各类信息(文字、几何图形、图像）;
- 文本信息的显示;
- 图形的绘制;
- 颜色的设置;
- 图象的绘制;

作业

- 尝试在窗体上画一张笑脸.Arc,Area



Thank You !