Java程序设计
—图形用户界面设计
JavaFX基础

李懿
医学技术学院
浙江中医药大学
2017年2月28日

JavaFX vs Swing and AWT

- AWT(Abstract Windows Toolkit)
  - 适用于简单的界面设计
  - 平台相关
  - 十分古老
- Swing
  - 更强壮、通用、灵活
  - 平台无关，即在所有的平台上运行效果都一样
  - 只能用于开发桌面应用
  - 目前已不再继续更新
- JavaFX
  - 最新的界面开发工具(GUI Platform)
  - 可用于开发富网络应用(Rich Internet Application(RIA))
  - 可在桌面、网页、智能设备等运行
  - 支持多点触控
  - 内建2D,3D,动画,视频,音频等支持，既可作为独立应用程序，也可从浏览器中运行

认识JavaFX程序

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class MyJavaFX extends Application{
  @Override//重写Application中的start方法
  public void start( Stage primaryStage ){
        //创建一个场景scene，在其中放入一个按钮
        Button btOK = new Button( "OK" );
        Scene scene = new Scene( btOK, 200, 250 );
        //设置舞台标题
        primaryStage.setTitle( "MyJavaFX" );
        //放置舞台主场景
        primaryStage.setScene( scene );
        //显示舞台
        primaryStage.show();
    }

    public static void main( String[] args ){
        Application.launch( args );
    }
}
```

关于程序的解释

1. Application中的launch用于执行一个独立的JavaFX应用
2. 如果从命令行开始运行，整个main方法可不写，此时，JVM会自动搜索源程序中的launch方法，并调用该方法运行程序
3. 重写Application类中的start方法，JVM构建应用对象时调用无参构造方法构造对象，并调用其start方法。一般来说，start方法在场景中放置UI控制器，设置场景中的各关键节点
4. Button是一个场景中的对象
5. Scene对象构造方法 Scene(node,width,height)，声明了场景的高度和宽度，以及将节点node放置在场景中
6. Stage对象是一个窗口，当程序运行时，JVM会自动创建一个Stage对象，称为primary stage(主舞台)
7. 设置Stage的场景并显示场景，这里利用了戏剧舞台的类比，stage是支持各场景的平台，node为在各场景中进行演出的演员

认识JavaFX程序

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class MyJavaFX extends Application{
	@Override
	public void start( Stage primaryStage ){
		Scene scene = new Scene( new Button("OK"), 200, 250 );
		primaryStage.setTitle( "MyJavaFX" );
		primaryStage.setScene( scene );
		primaryStage.show();

		Stage stage = new Stage();
		stage.setTitle( "Second Stage" );
		stage.setScene( new Scene( new Button( "New Stage" ), 100, 100 ) );
		stage.show();
	}
}
```

JavaFX构成部件

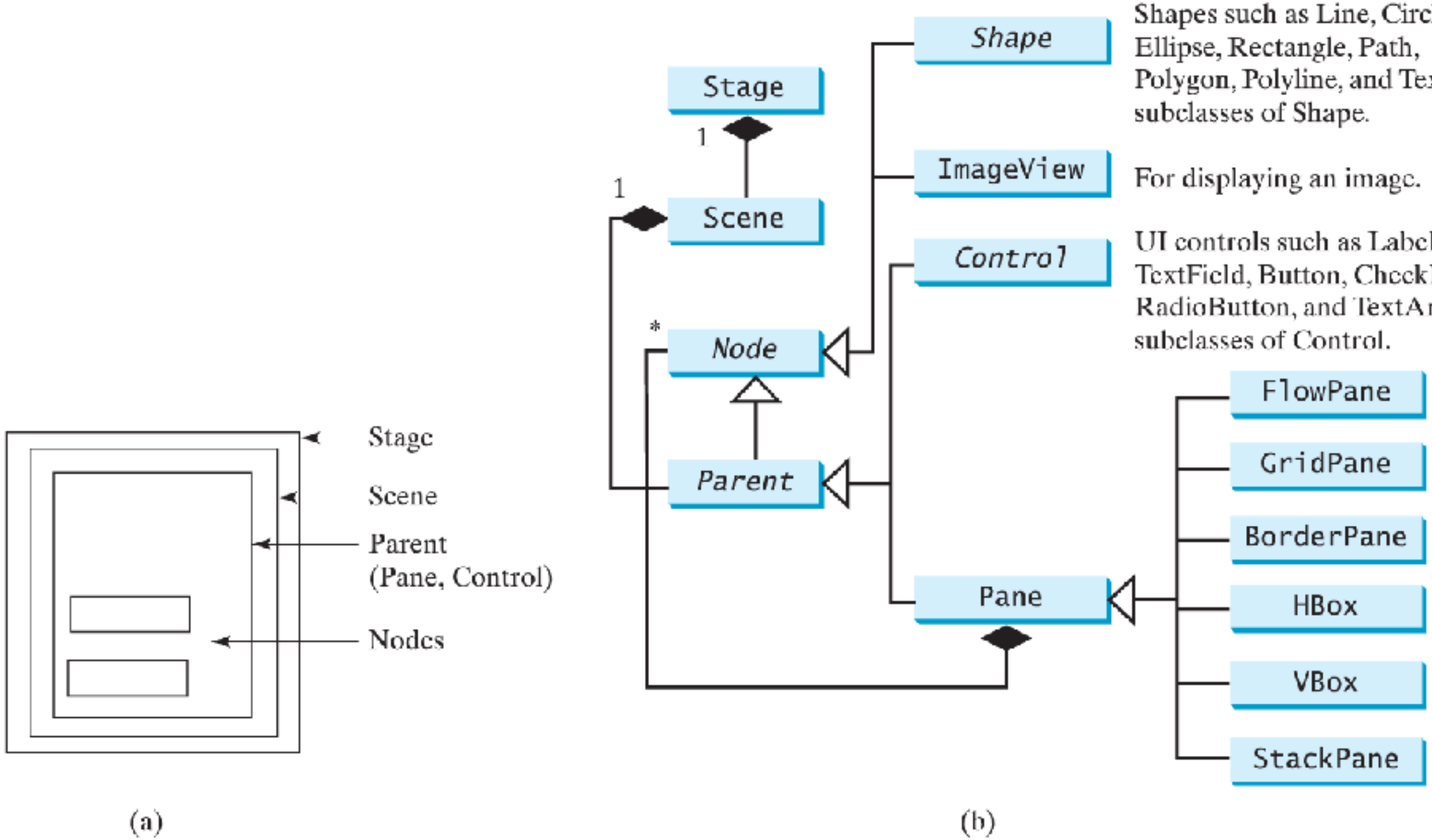


FIGURE 14.3 (a) Panes are used to hold nodes. (b) Nodes can be shapes, image views, UI controls, and panes.

JavaFX构成部件

- Pane(容器类)：自动布局放在其中的各结点类(Node)，包括位置和大小，然后将容器放置在场景(Scene)中
- Node(结点类)：可视化部件，可以是形状(Shape)、图片窗口(Imageview)、界面控制器(UI Control)，或者是一个容器
- UI Control(界面控制器)：包括如label, button, check box, radio button, text field, text area等
- 场景(Scene)要放在舞台(Stage)中显示
- 场景(Scene)可以包含控制器或容器，但不能是简单的形状或是图片窗口
- 创建场景(Scene)对象可调用构造方法Scene(Parent,width,height)或Scene(Parent)
- 容器类(Pane)中含有所有结点类(Node)的子类，所有Node的子类都有默认无参的构造方法供调用构建对象

JavaFX构成部件

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class ButtonInPane extends Application{
      @Override
      public void start( Stage primaryStage ){
            StackPane pane = new StackPane();
            pane.getChildren().add( new Button( "OK" ) );

            Scene scene = new Scene( pane, 200, 50 );
            primaryStage.setTitle( "Button in a Pane" );
            primaryStage.setScene( scene );
            primaryStage.show();
   }
}
```

Pane中显示Button

JavaFX构成部件

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;

public class MyJavaFX extends Application{
      @Override
      public void start( Stage primaryStage ){
//创建一个圆对象并设置其属性
Circle circle = new Circle();
circle.setCenterX( 100 );
circle.setCenterY( 100 );
circle.setRadius( 50 );
circle.setStroke( Color.BLACK );
circle.setFill( Color.WHITE );

//创建一个容器放置圆
Pane pane = new Pane();
pane.getChildren().add( circle );

//创建场景并将其放置在舞台上
Scene scene = new Scene( pane,
200, 200 );

primaryStage.setTitle( "ShowCircle" );
            primaryStage.setScene( scene );
            primaryStage.show();
      }
}
```

场景中画圆

JavaFX构成部件

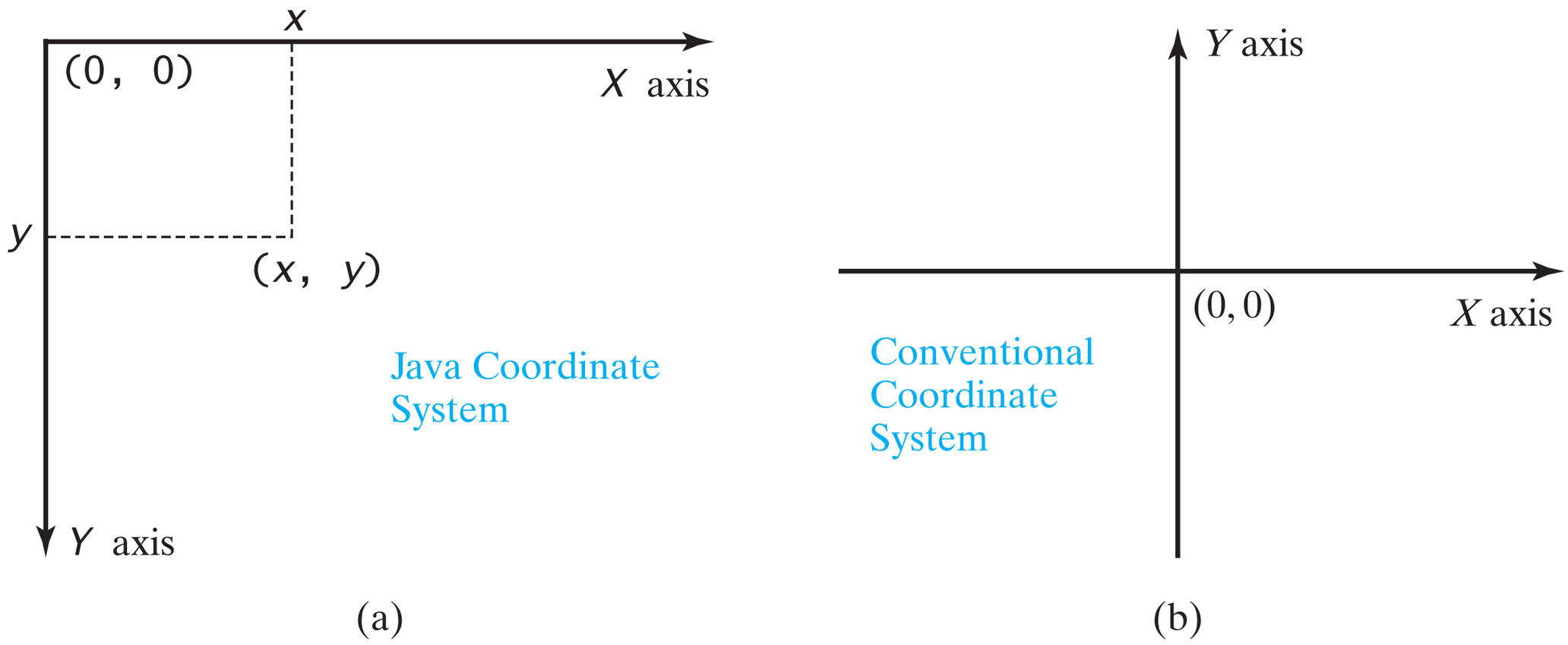


FIGURE 14.6 The Java coordinate system is measured in pixels, with `(0, 0)` at its upper-left corner.

场景中画圆

属性绑定

- 属性绑定(Property binding)：令目标对象的属性与源对象的属性相绑定
  - 如果源对象的属性值发生变化，目标对象的属性也会相应变化
  - 例：将圆心位置与窗口大小和位置绑定

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;

public class MyJavaFX extends Application{
    @Override
    public void start( Stage primaryStage ){
        //创建一个容器放置圆
        Pane pane = new Pane();
        //创建一个圆对象并设置其属性
        Circle circle = new Circle();

        circle.centerXProperty().bind( pane.
        widthProperty().divide( 2 ) );
        circle.centerYProperty().bind( pane.
        heightProperty().divide( 2 ) );
        circle.setRadius( 50 );
        circle.setStroke( Color.BLACK );
        circle.setFill( Color.WHITE );
        pane.getChildren().add( circle );
        //创建场景并将其放置在舞台上
        Scene scene = new Scene( pane,
200, 200 );

        primaryStage.setTitle( "ShowCircle
        Centered" );
        primaryStage.setScene( scene );
        primaryStage.show();
    }
}
```

属性绑定

- target.bind(source)
  - bind方法定义在javafx.beans.property.Property接口中
  - 其中binding属性是javafx.beans.property.Property的一个实例
  - 源对象为javafx.beans.value.ObservableValue接口的一个实例
- JavaFX中分别为基本数据类型和字符串定义了绑定属性
  - double/float/long/int/boolean -- DoubleProperty/FloatProperty/LongProperty/IntegerProperty/BooleanProperty
  - String -- StringProperty
  - 这些都是ObservableValue的子类
  - 每一个JavaFX类的绑定属性都有相应的get/set方法，如Circle中有getCenterX()和setCenterX(double)方法
  - 绑定属性命名为属性名后加Property
    - centerXProperty()和centerYProperty()
  - 双向绑定bindBidirectional(Property)

```
public class Circle{
   private DoubleProperty centerX;
   public double getCenterX(){...} //值的get方法
   public void setCenterX(double value){...} //值的set方法
   public DoubleProperty centerXProperty(){...} //属性的get方法
}
```

```
circle.centerXProperty().bind(pane.widthProperty()
.divide(2))
||
centerX.bind(width.divide(2))
```

Image和ImageView类

- Image类表示一幅图像
- ImageView表示可用于显示图像的工具类
- javafx.scene.image.Image
- javafx.scene.image.ImageView
  - Image image = new Image("images/us.gif");
  - ImageView imageView = new ImageView(image);
  - 或者将两者合二为一：ImageView imageView = new ImageView("image/us.gif");

Node类属性和方法

- Node类定义了所有的结点类的共同属性和方法
- JavaFX风格属性与网页中HTML元素通过CSS表明风格类似，因此风格属性也称JavaFX CSS
- 所有的属性以“-fx-”作为前缀，以";"作分割，用":"指明属性值
  - 如circle.setStyle("-fx-stroke: black; -fx-fill: red;");用于设置圆的画笔和填充属性
  - 与circle.setStroke(Color.BLACK);和circle.setFill(Color.RED);等同
- 如果采用了错误的JavaFX CSS，程序能够编译，但是错误的风格设置会被忽略
- Color和Font类

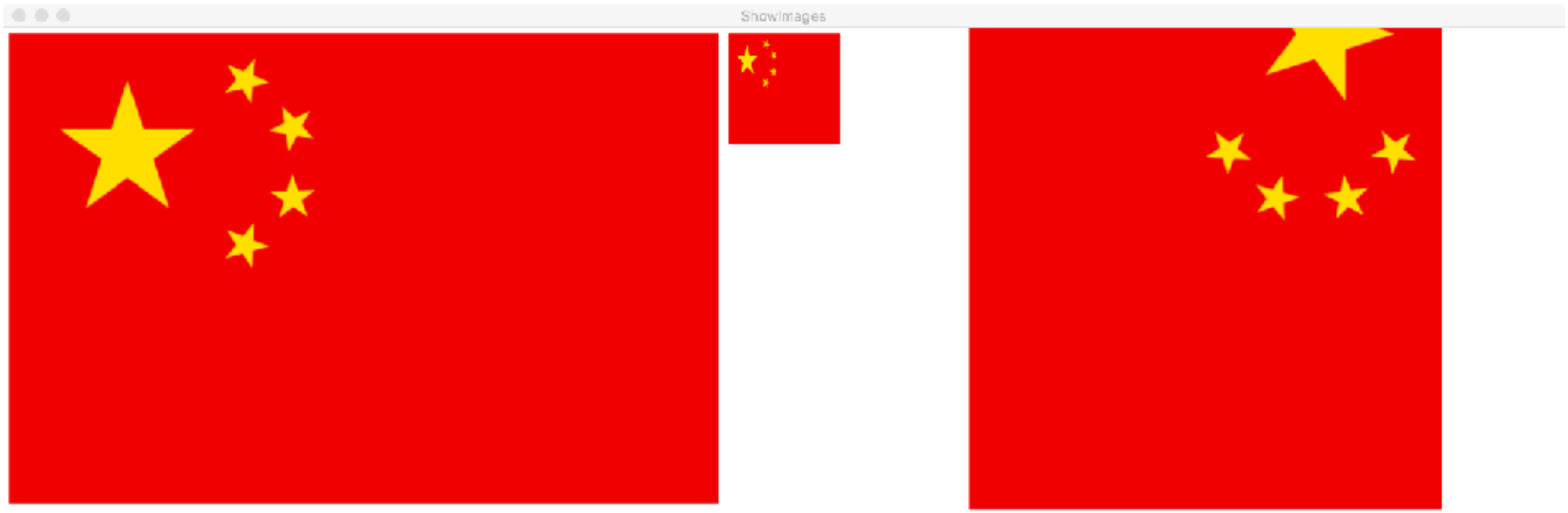
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;
import javafx.scene.layout.StackPane;

public class MyJavaFX extends Application{
	@Override
	public void start( Stage primaryStage ){
		StackPane pane = new StackPane();
		Button btOK = new Button( "OK" );
		btOK.setStyle( "-fx-border-color:
blue;" );
		pane.getChildren().add( btOK );

		pane.setRotate( 45 );
		pane.setStyle( "-fx-border-color: red; -
fx-background-color: lightgray;" );

		Scene scene = new Scene( pane, 200, 250
);
		primaryStage.setScene( scene );
		primaryStage.show();
	}
}
```

显示旋转的按钮

Image和ImageView类

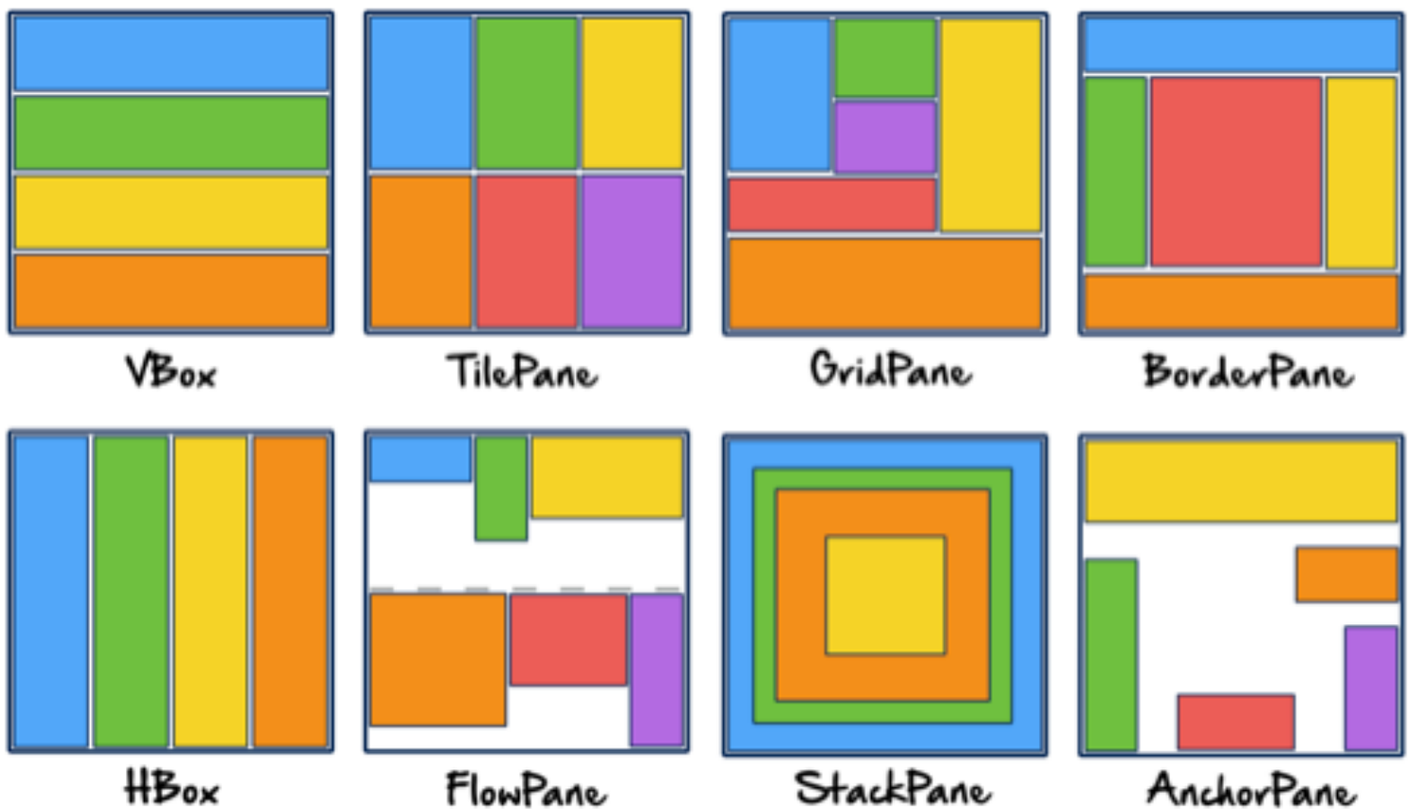
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

public class ShowImage extends
Application{
     public void start( Stage
primaryStage ){
          Pane pane = new HBox( 20 );
          pane.setPadding( new Insets( 5, 5,
5, 5 ) );
          Image image = new
Image( "images/cn.png" );
          pane.getChildren().add( new
ImageView( image ) );
          ImageView imageView2 = new
ImageView( image );
          imageView2.setFitHeight( 100 );
          imageView2.setFitWidth( 100 );

pane.getChildren().add( imageView2 );

          ImageView imageView3 = new
ImageView( image );
          imageView3.setRotate( 90 );

pane.getChildren().add( imageView3 );

          Scene scene = new Scene( pane );

primaryStage.setTitle( "ShowImages" );
          primaryStage.setScene( scene );
          primaryStage.show();
     }
}
```

Image和ImageView类



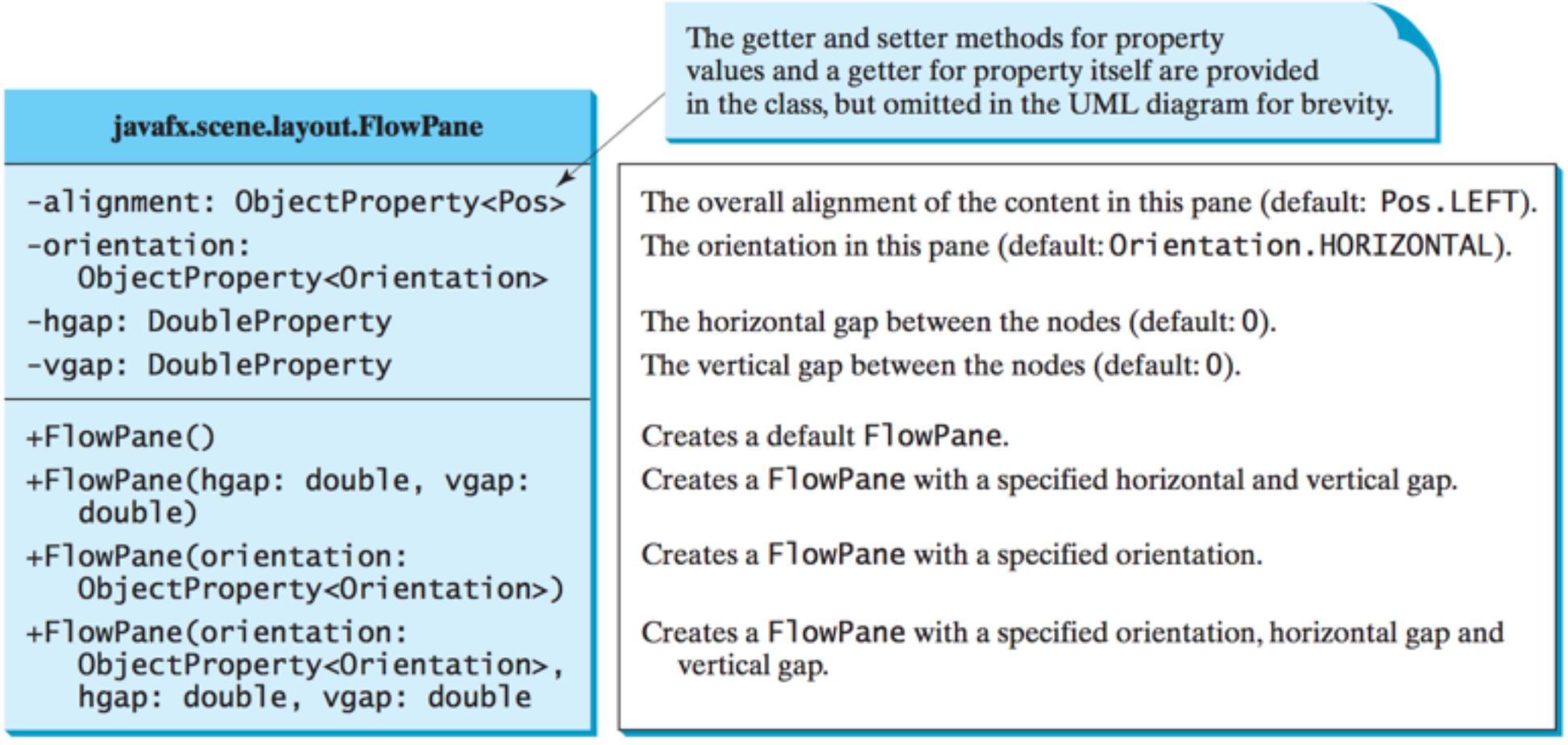
布局容器

类	描述
Pane	所有布局容器类的基类，其getChildren()方法可用于返回容器里一系列子结点
StackPane	结点依次放置在其它结点上方，所有结点位于容器中央
FlowPane	以水平方向一行一行或垂直方向一列一列的形式放置结点
GridPane	以二维网格形式放置结点
BorderPane	分别在上、下、左、右和中部放置结点
HBox	将所有结点放在一行
VBox	将所有结点放在一列



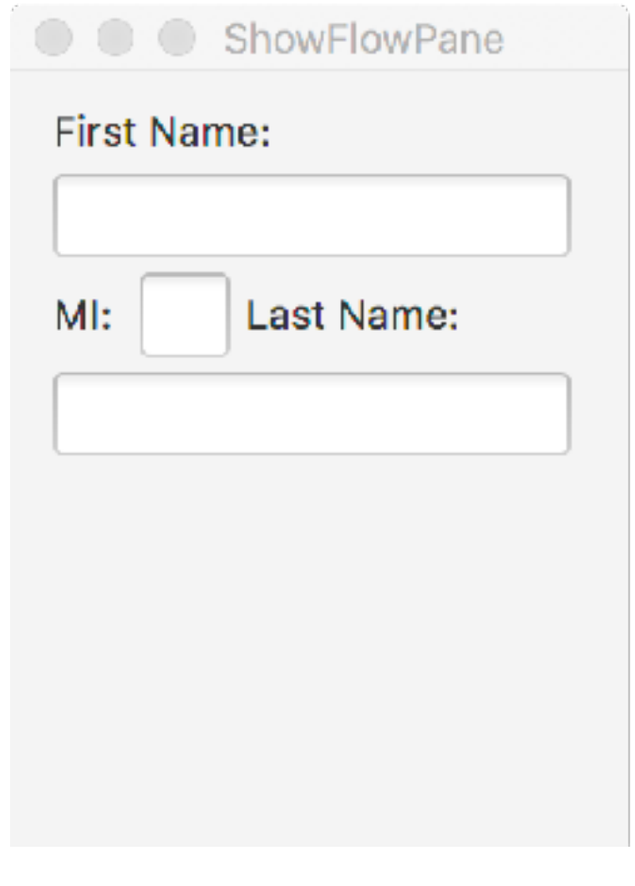
布局容器

- **FlowPane**
  - 将所有的结点对象按添加顺序，按照行顺序从左往右从上往下进行填充，填充一行/列满，则填充下一行/列
  - Orientation.HORIZONTAL, Orientation.VERTICAL声明填充方向，按行或列



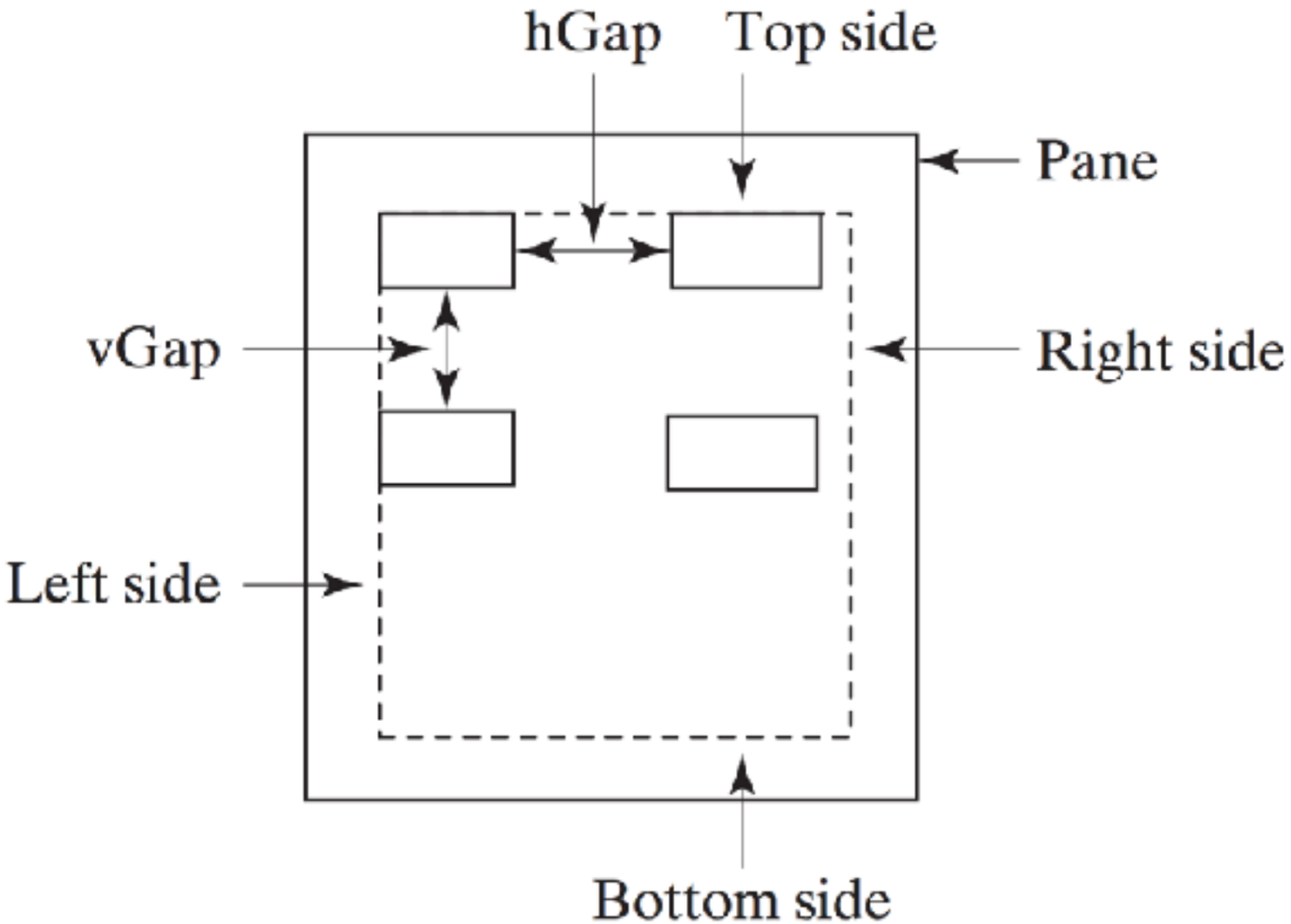
布局容器

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class ShowFlowPane extends Application{
	@Override
	public void start( Stage primaryStage ){
		FlowPane pane = new FlowPane();
		pane.setPadding( new Insets( 11, 12, 13, 14 ) );
		pane.setHgap( 5 );
		pane.setVgap( 5 );

		pane.getChildren().addAll( new Label( "First Name:" ), new TextField(), new Label( "MI: " ) );
		TextField tfMi = new TextField();
		tfMi.setPrefColumnCount( 1 );//设置文本框的宽度为1列宽
		pane.getChildren().addAll( tfMi, new Label( "Last Name:" ), new TextField() );

		Scene scene = new Scene( pane, 200, 250 );
		primaryStage.setTitle( "ShowFlowPane" );
		primaryStage.setScene( scene );
		primaryStage.show();
	}
}
```



布局容器

- FlowPane的padding属性：指的是空间大小布局，Insets(11,12,13,14)指的是创建一个Insets，其边界尺寸分别以像素计为top(11)，right(12)，bottom(13)，left(14)
- FlowPane的Hgap和Vgap分别指的是行与行之间，列与列之间的间隔大小
- tfMi.setPrefColumnCount(1)将对应的文本框对象的宽度设定为1列宽
- 在一个Pane中，各结点对象只允许添加一次



布局容器

- **GridPane**
  - 将所有结点对象放置在一二维矩阵型布局中，各位置以行、列标示

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.layout.GridPane

```
-alignment: ObjectProperty<Pos>
-gridLinesVisible:
   BooleanProperty
-hgap: DoubleProperty
-vgap: DoubleProperty
```

```
+GridPane()
+add(child: Node, columnIndex:
   int, rowIndex: int): void
+addColumn(columnIndex: int,
   children: Node...): void
+addRow(rowIndex: int,
   children: Node...): void
+getColumnIndex(child: Node):
   int
+setColumnIndex(child: Node,
   columnIndex: int): void
+getRowIndex(child:Node): int
+setRowIndex(child: Node,
   rowIndex: int): void
+setHalighnment(child: Node,
   value: HPos): void
+setValighnment(child: Node,
   value: VPos): void
```

The overall alignment of the content in this pane (default: `Pos.LEFT`).
Is the grid line visible? (default: `false`)

The horizontal gap between the nodes (default: 0).
The vertical gap between the nodes (default: 0).

Creates a `GridPane`.
Adds a node to the specified column and row.

Adds multiple nodes to the specified column.

Adds multiple nodes to the specified row.

Returns the column index for the specified node.

Sets a node to a new column. This method repositions the node.

Returns the row index for the specified node.
Sets a node to a new row. This method repositions the node.

Sets the horizontal alignment for the child in the cell.

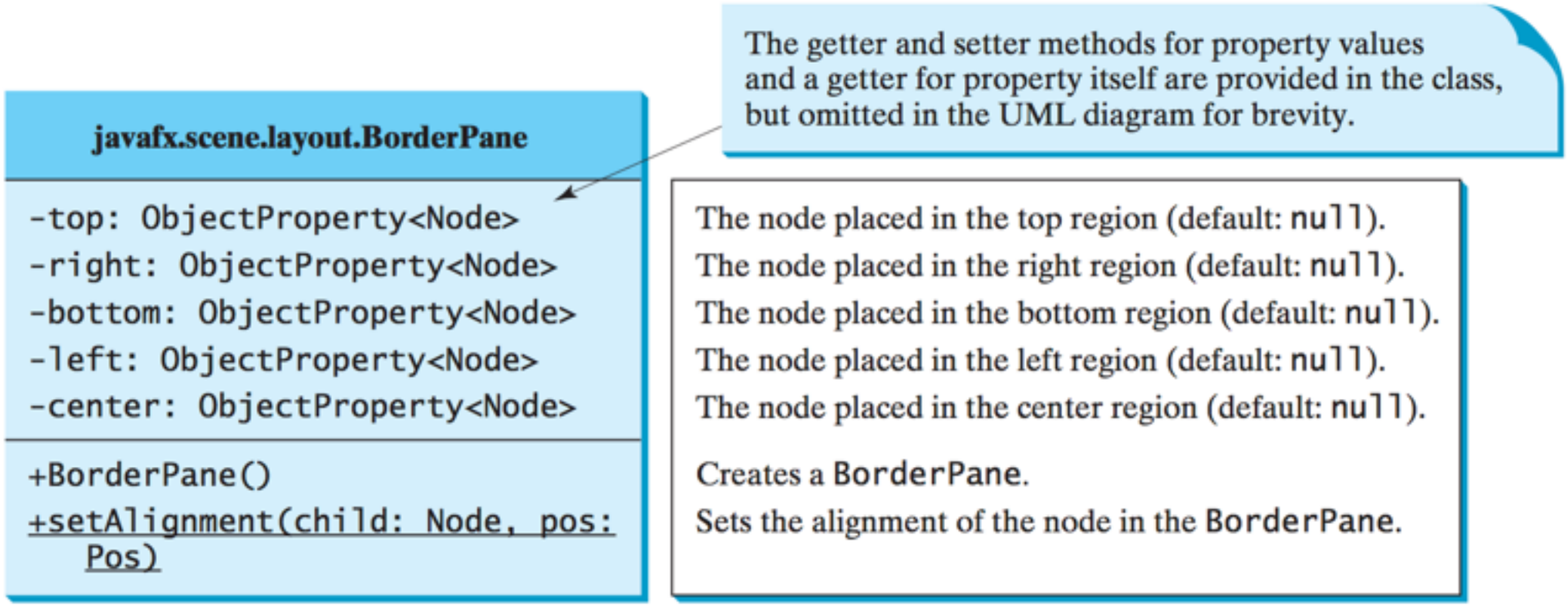
Sets the vertical alignment for the child in the cell.

布局容器

```
import javafx.application.Application;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class ShowGridPane extends Application{
    @Override
    public void start( Stage primaryStage ){
        GridPane pane = new GridPane();
        pane.setAlignment( Pos.CENTER );
        pane.setPadding( new Insets( 11.5, 12.5, 13.5,
14.5 ) );
        pane.setVgap( 5.5 );
        pane.setHgap( 5.5 );
        pane.add( new Label( "First Name: " ), 0,
0 );
        pane.add( new TextField(), 1, 0 );
        pane.add( new Label( "MI: " ), 0, 1 );
        pane.add( new TextField(), 1, 1 );
        pane.add( new Label( "Last Name: " ), 0, 2 );
        pane.add( new TextField(), 1, 2 );
        Button btAdd = new Button( "Add Name" );
        pane.add( btAdd, 1, 3 );
        GridPane.setHalignment( btAdd,
HPos.RIGHT );//设置按钮右对齐
        Scene scene = new Scene( pane );
        primaryStage.setTitle( "ShowGridPane" );
        primaryStage.setScene( scene );
        primaryStage.show();
    }
}
```

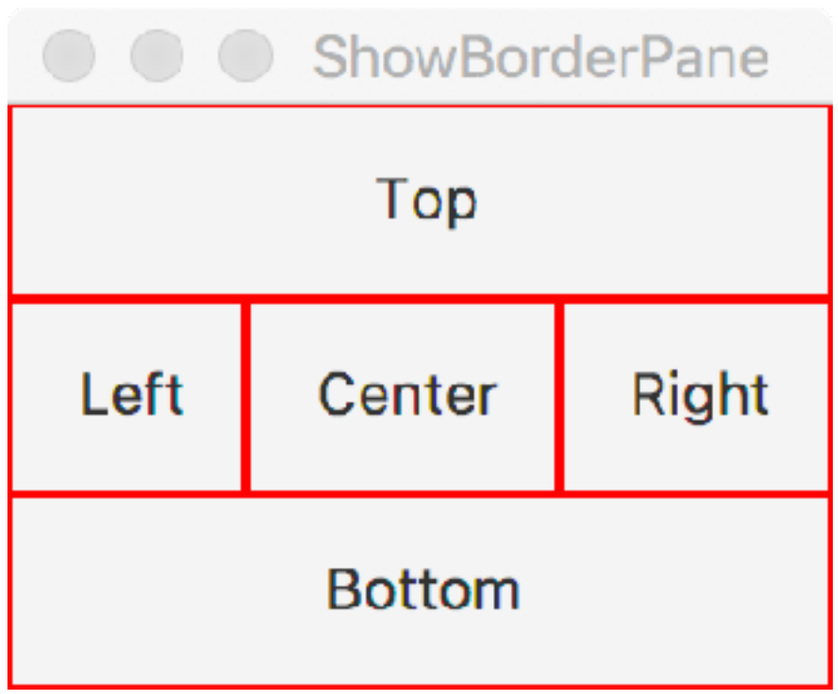
布局容器

- **BorderPane**
  - 所有的结点放置在五个区域：上(top)、下(bottom)、左(left)、右(right)和中间(center)
  - 对应的方法分别为 setTop(node)，setBottom(node)，setLeft(node)，setRight(node), setCenter(node)



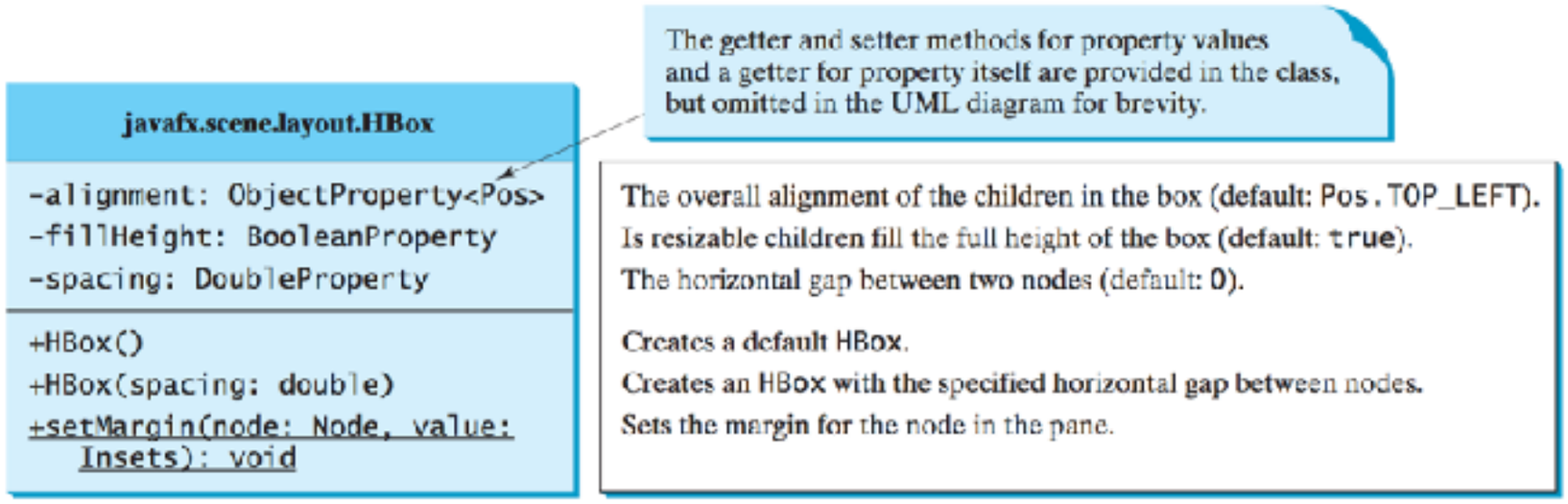
布局容器

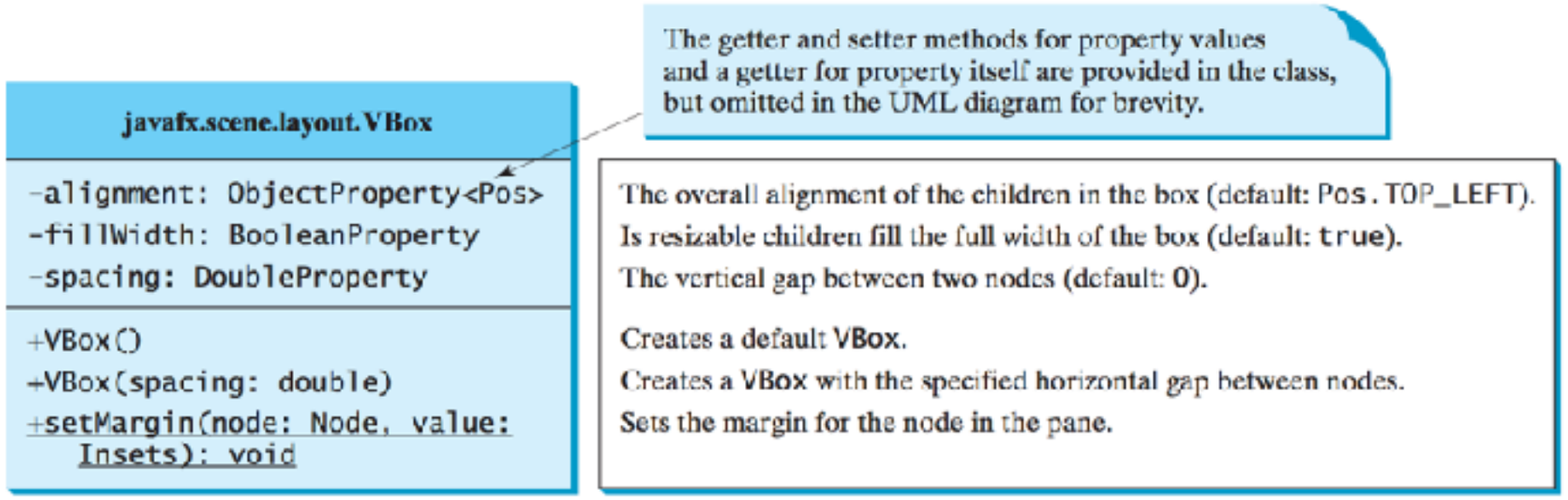
```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

class CustomPane extends StackPane{
	public CustomPane( String title ){
		getChildren().add( new Label( title ) );
		setStyle( "-fx-border-color: red" );
		setPadding( new Insets( 11.5, 12.5, 13.5,
14.5 ) );
	}
}

public class ShowBorderPane extends Application{
	@Override
	public void start( Stage primaryStage ){
		BorderPane pane = new BorderPane();

		pane.setTop( new CustomPane( "Top" ) );
		pane.setBottom( new
CustomPane( "Bottom" ) );
		pane.setRight( new CustomPane( "Right" ) );
		pane.setLeft( new CustomPane( "Left" ) );
		pane.setCenter( new CustomPane( "Center" )
);
		Scene scene = new Scene( pane );
		primaryStage.setTitle( "ShowBorderPane" );
		primaryStage.setScene( scene );
		primaryStage.show();
	}
}
```



布局容器

- **HBox和VBox**
  - HBox: 所有的结点放在同一行
  - VBox: 所有的结点放在同一列
  - 注意与FlowPane区分



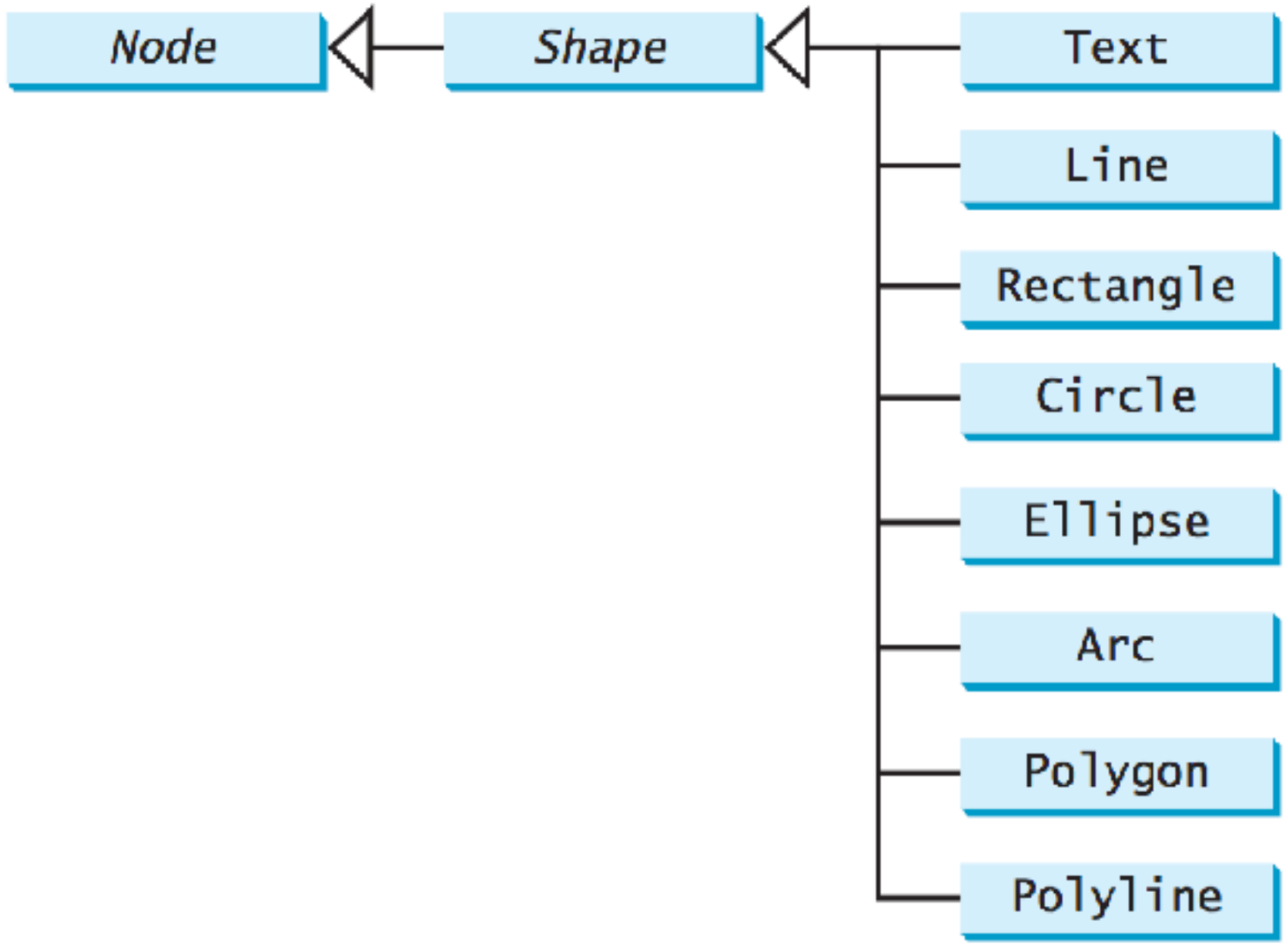


布局容器

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

public class ShowHBoxVBox extends Application{
	@Override
	public void start( Stage primaryStage ){
		BorderPane pane = new BorderPane();

		pane.setTop( getHBox() );
		pane.setLeft( getVBox() );

		Scene scene = new Scene( pane );
		primaryStage.setTitle( "ShowHBoxVBox" );
		primaryStage.setScene( scene );
		primaryStage.show();
	}

	public HBox getHBox(){
		HBox hBox = new HBox( 15 );
		hBox.setPadding( new Insets( 15, 15, 15, 15 ) );
		hBox.setStyle( "-fx-background-color: gold" );
		hBox.getChildren().add( new Button( "Computer
Science" ) );
		hBox.getChildren().add( new Button( "Chemistry" ) );

		ImageView imageView = new ImageView( new
Image( "images/cn.png" ) );
		hBox.getChildren().add( imageView );
		return hBox;
	}

	public VBox getVBox(){
		VBox vBox = new VBox( 15 );
		vBox.setPadding( new Insets( 15, 5, 5, 5 ) );
		vBox.getChildren().add( new Label( "Courses" ) );

		Label[] courses = { new Label( "CS1501"), new
Label( "CS1502" ), new Label( "ME1501" ), new Label( "ME1502" ) };

		for( Label course : courses ){
			vBox.setMargin( course, new Insets( 0, 0, 0,
15 ) );
			vBox.getChildren().add( course );
		}

		return vBox;
	}
```

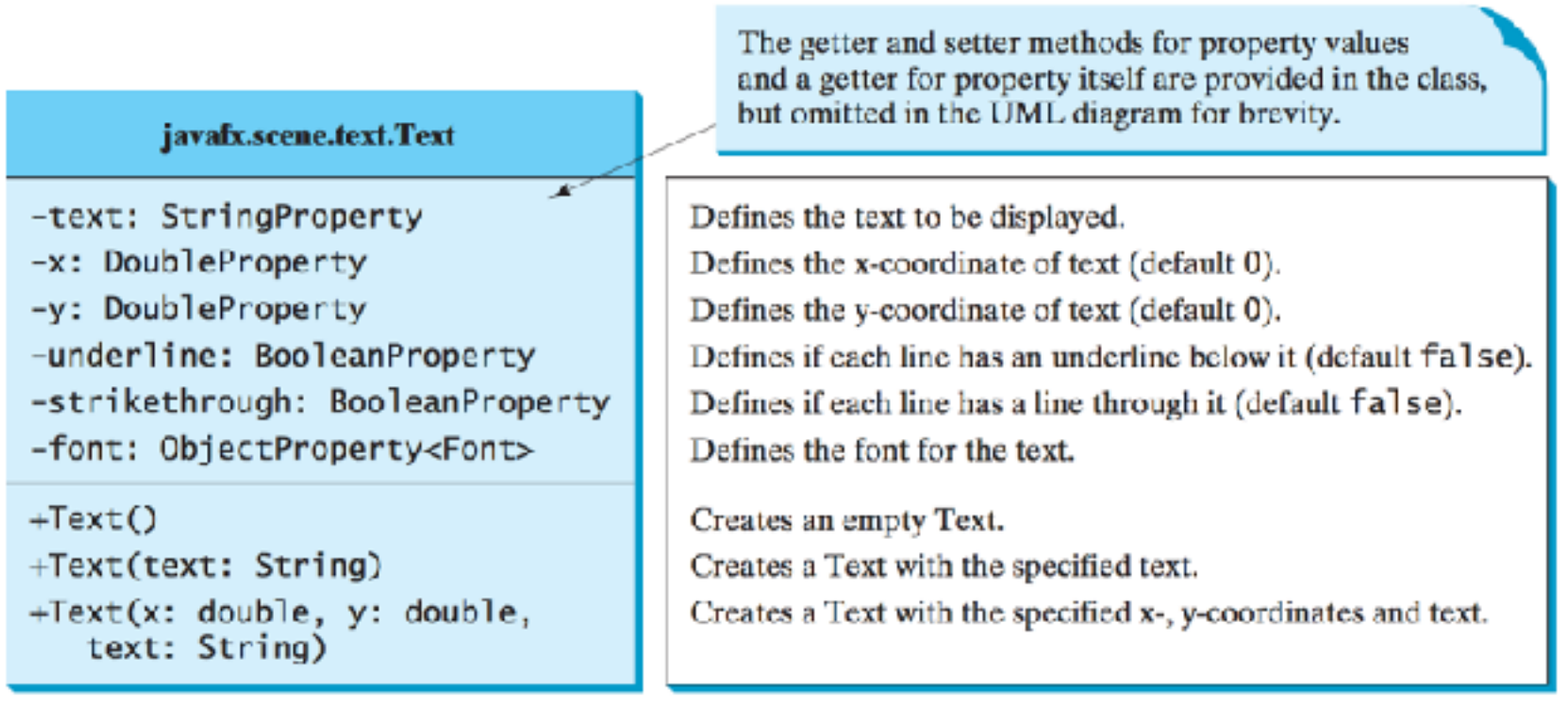
形状

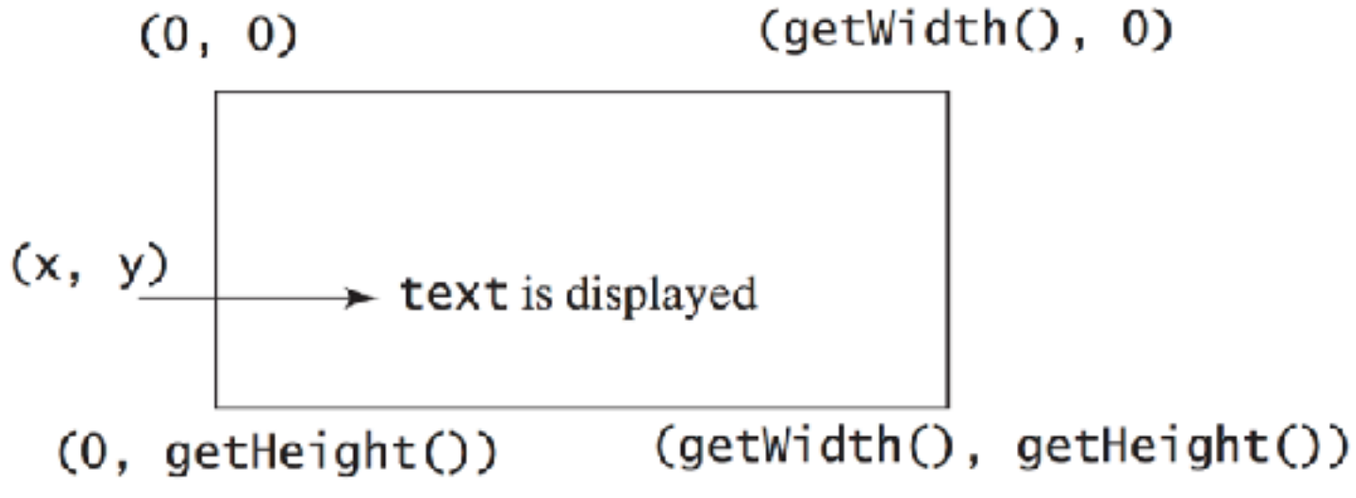
- **Shapes**
  - JavaFX提供了一系列形状相关的类用于绘制基本的二维图形，比如直线、圆、矩形、椭圆、多边形、折线、文字等
  - Shape类为一抽象类，定义了所有形状所共有的各类属性，如fill，stroke，strokeWidth等



形状--文字

- **Text**
  - 在位置(x,y)处显示字符串



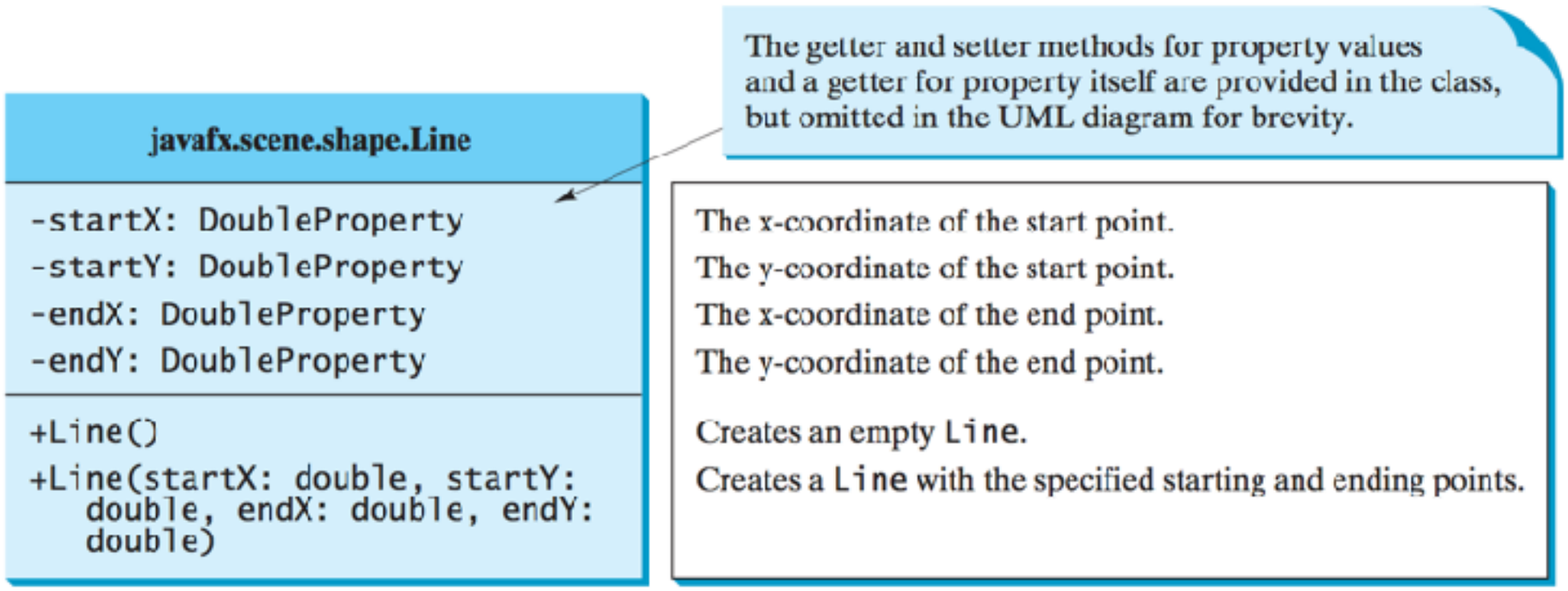


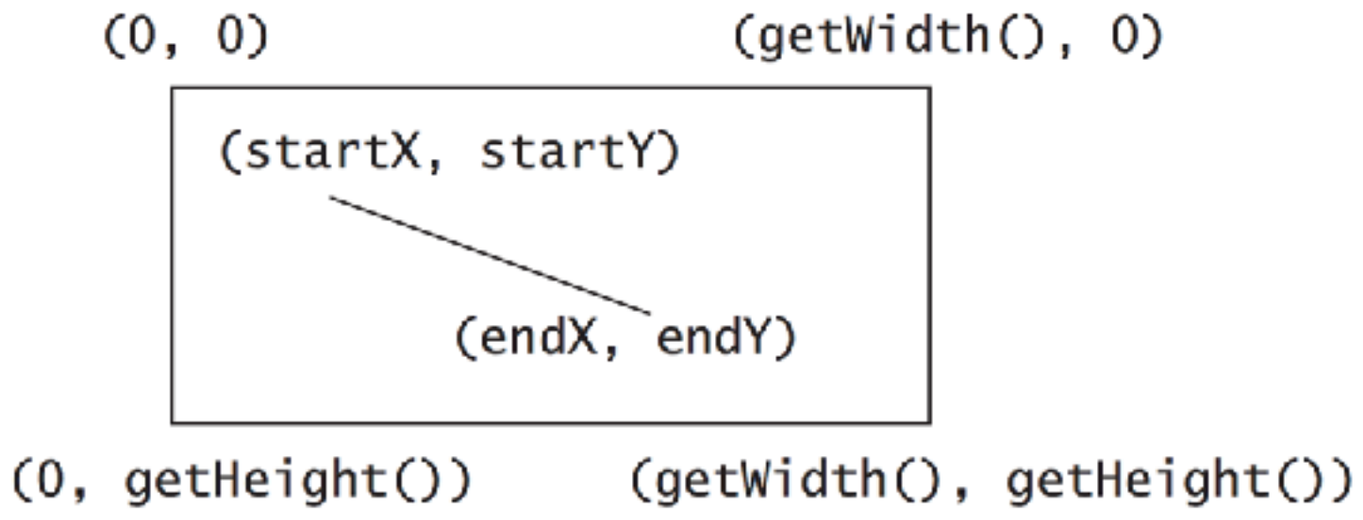
形状--文字

```
iimport javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.FontPosture;

public class ShowText extends Application{
	@Override
	public void start( Stage primaryStage ){
		Pane pane = new Pane();
		pane.setPadding( new Insets( 5, 5, 5,
5 ) );
		Text text1 = new Text( 20, 20,
"Programming is fun" );
		text1.setFont( Font.font( "Courier",
FontWeight.BOLD, FontPosture.ITALIC, 15 ) );
		pane.getChildren().add( text1 );

		Text text2 = new Text( 60, 60,
"Programming is fun\nDisplay Text" );
		pane.getChildren().add( text2 );

		Text text3 = new Text( 10, 100,
"Programming is fun\nDisplay Text" );
		text3.setFill( Color.RED );
		text3.setUnderline( true );
		text3.setStrikethrough( true );
		pane.getChildren().add( text3 );

		Scene scene = new Scene( pane );
		primaryStage.setTitle( "ShowText" );
		primaryStage.setScene( scene );
		primaryStage.show();
	}
}
```

形状--线条

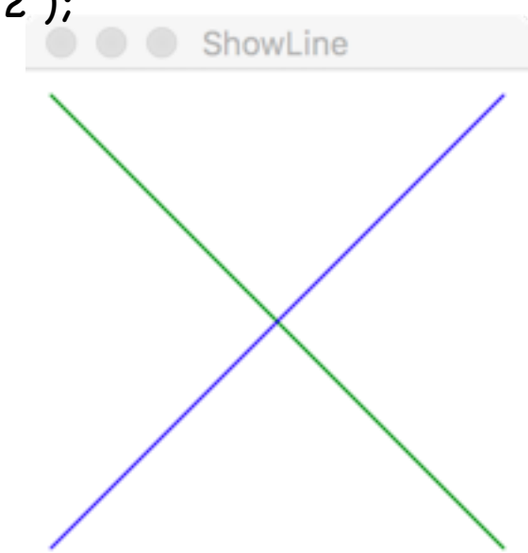
- **Line**
  - 根据(startX,startY,endX,endY)四参数绘制线条





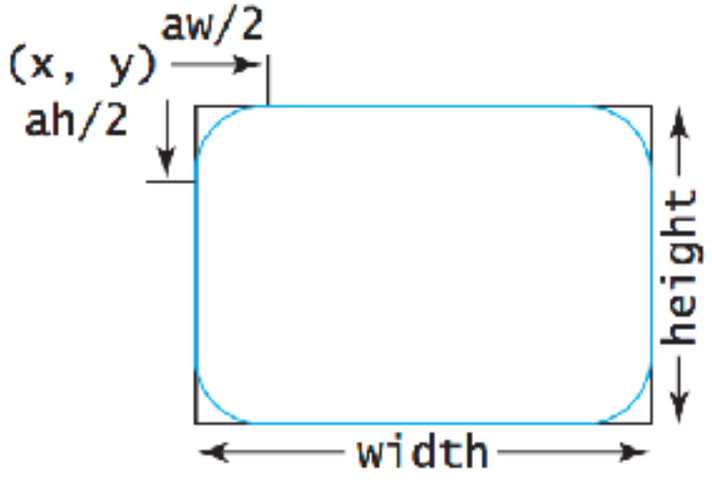
形状--线条

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Line;

public class ShowLine extends Application{
	@Override
	public void start( Stage primaryStage ){
		Scene scene = new Scene( new LinePane(),
200, 200 );
		primaryStage.setTitle( "ShowLine" );
		primaryStage.setScene( scene );
		primaryStage.show();
	}
}

class LinePane extends Pane{
	public LinePane(){
		Line line1 = new Line( 10, 10, 10, 10 );
line1.endXProperty().bind( widthProperty().subtract( 1
0 ) );

line1.endYProperty().bind( heightProperty().subtract(
10 ) );
		line1.setStrokeWidth( 1 );
		line1.setStroke( Color.GREEN );
		getChildren().add( line1 );

		Line line2 = new Line( 10, 10, 10, 10 );
line2.startXProperty().bind( widthProperty().subtract(
10 ) );

line2.endYProperty().bind( heightProperty().subtract(
10 ) );
		line2.setStrokeWidth( 1 );
		line2.setStroke( Color.BLUE );
		getChildren().add( line2 );
	}
}
```



形状--矩形

- **Rectangle**
  - 矩形绘制需给定六参数，x，y，width，height，arcWidth，arcHeight
  - 其中，矩形的左上角顶点为(x,y)，arcWidth为左上角弧的水平直径，arcHeight为左上角弧的垂直方向直径



javafx.scene.shape.Rectangle

```
-x: DoubleProperty
-y:DoubleProperty
-width: DoubleProperty
-height: DoubleProperty
-arcWidth: DoubleProperty

-arcHeight: DoubleProperty
```

```
+Rectangle()
+Rectanlge(x: double, y:
   double, width: double,
   height: double)
```

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

The x-coordinate of the upper-left corner of the rectangle (default 0).

The y-coordinate of the upper-left corner of the rectangle (default 0).

The width of the rectangle (default: 0).

The height of the rectangle (default: 0).

The `arcWidth` of the rectangle (default: 0). `arcWidth` is the horizontal diameter of the arcs at the corner (see Figure 14.31a).

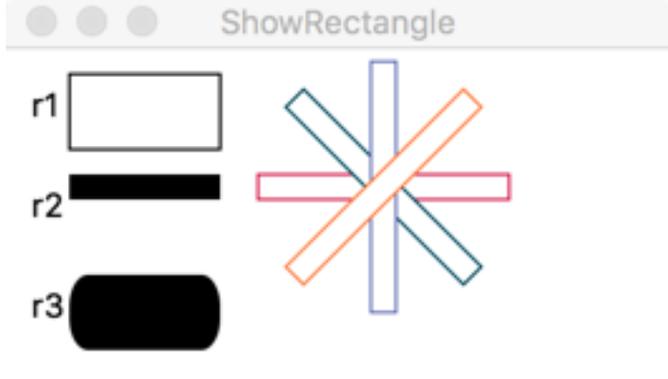
The `arcHeight` of the rectangle (default: 0). `arcHeight` is the vertical diameter of the arcs at the corner (see Figure 14.31a).

Creates an empty `Rectangle`.

Creates a `Rectangle` with the specified upper-left corner point, width, and height.

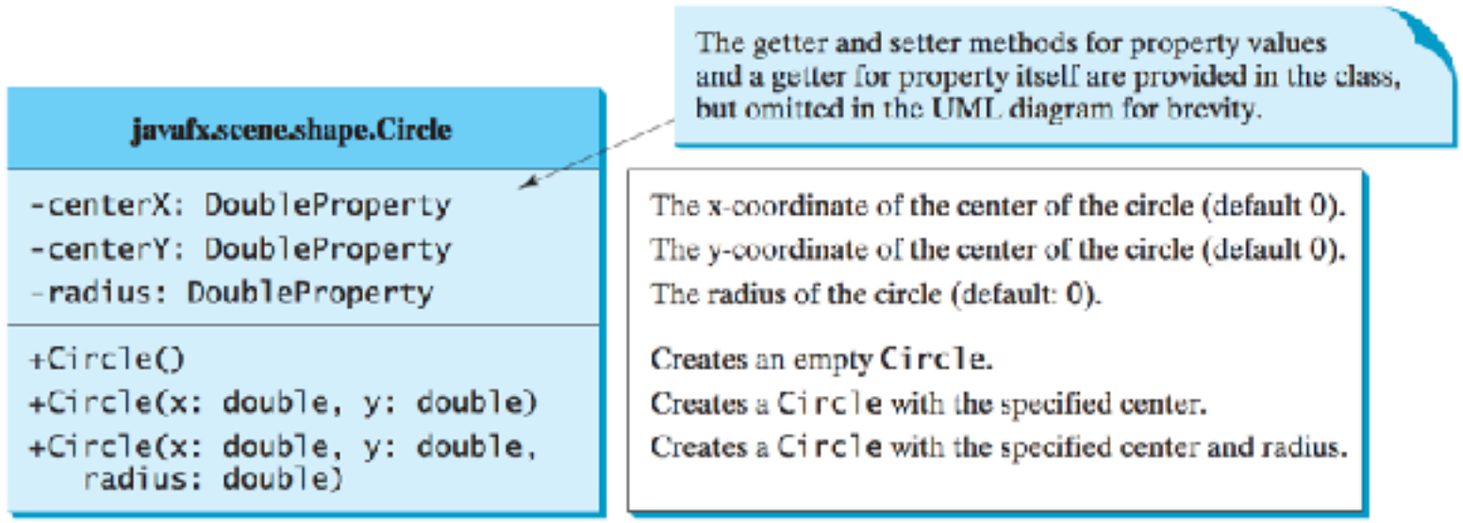
形状--矩形

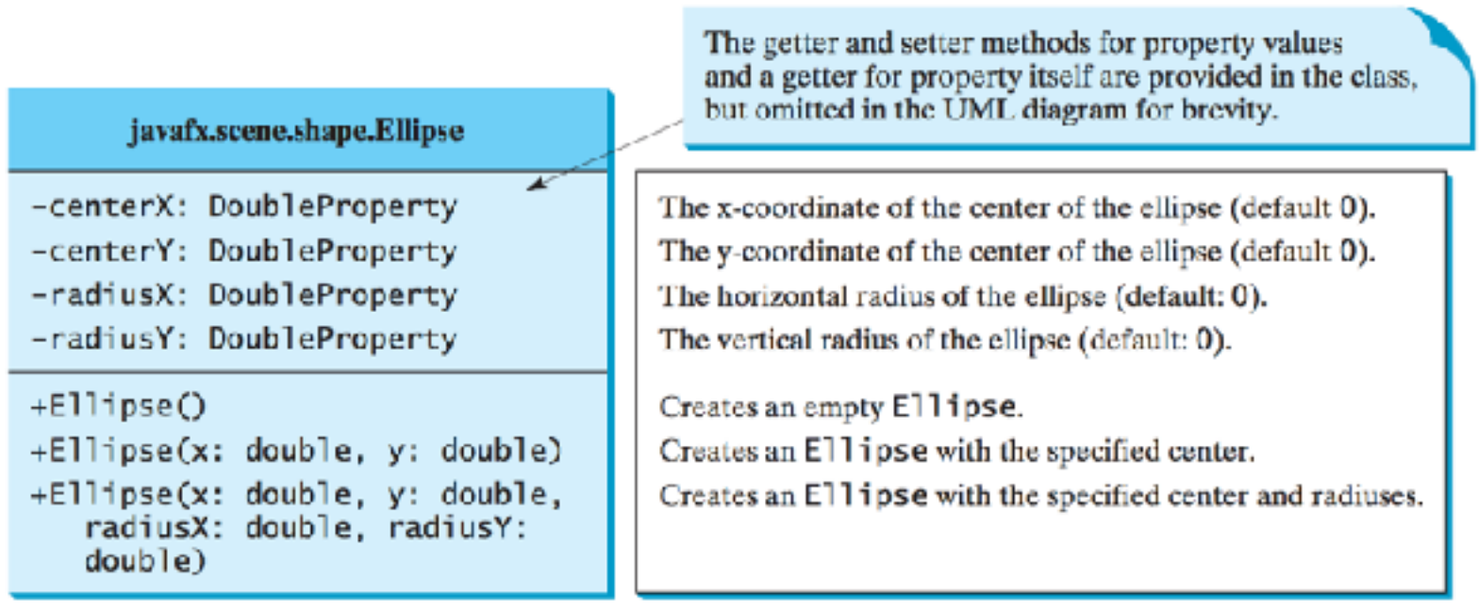
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import javafx.scene.shape.Rectangle;

public class ShowRectangle extends Application{
	@Override
	public void start( Stage primaryStage ){
		Pane pane = new Pane();
		Rectangle r1 = new Rectangle( 25, 10, 60, 30 );
		r1.setStroke( Color.BLACK );
		r1.setFill( Color.WHITE );
		pane.getChildren().add( new Text( 10, 27,
"r1" ) );
		pane.getChildren().add( r1 );

		Rectangle r2 = new Rectangle( 25, 50, 60, 10 );
		pane.getChildren().add( new Text( 10, 67,
"r2" ) );
		pane.getChildren().add( r2 );
		Rectangle r3 = new Rectangle( 25, 90, 60, 30 );
		r3.setArcWidth( 15 );
		r3.setArcHeight( 25 );
		pane.getChildren().add( new Text( 10, 107, "r3" )
);
		pane.getChildren().add( r3 );

		for( int i = 0; i < 4; i++ ){
			Rectangle r = new Rectangle( 100, 50,
100, 10 );
			r.setRotate( i * 360 / 8 );
			r.setStroke( Color.color( Math.random(),
Math.random(), Math.random() ) );
			r.setFill( Color.WHITE );
			pane.getChildren().add( r );
		}

		Scene scene = new Scene( pane, 250, 100 );
		primaryStage.setTitle( "ShowRectangle" );
		primaryStage.setScene( scene );
		primaryStage.show();
	}
}
```



形状--圆和椭圆

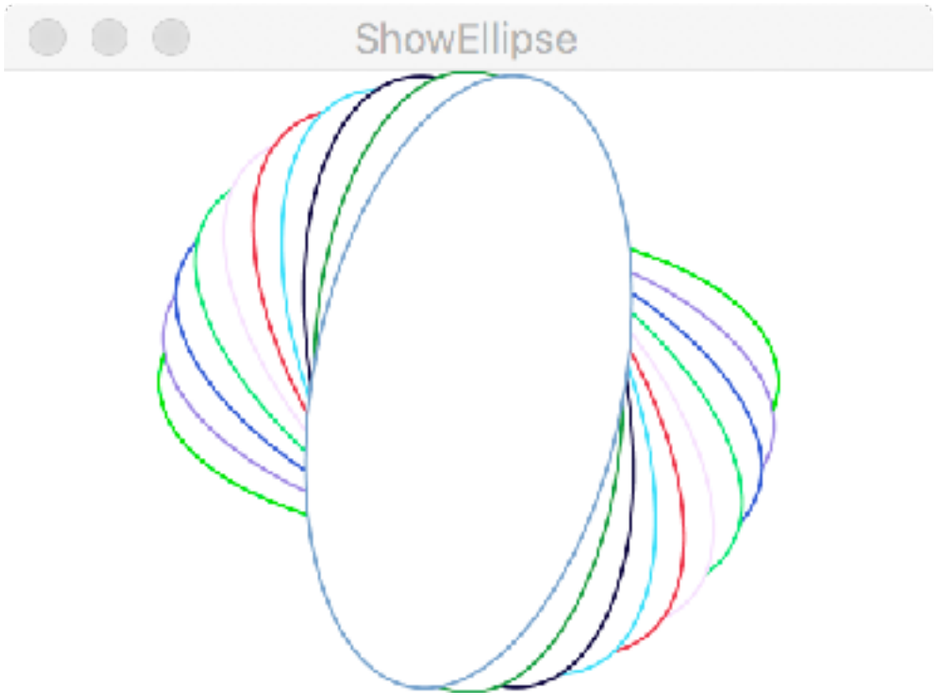
- **Circle**
  - 绘制圆，需用三参数：centerX, centerY, radius
- **Ellipse**
  - 绘制椭圆，需用四参数: centerX, centerY, radiusX, radiusY





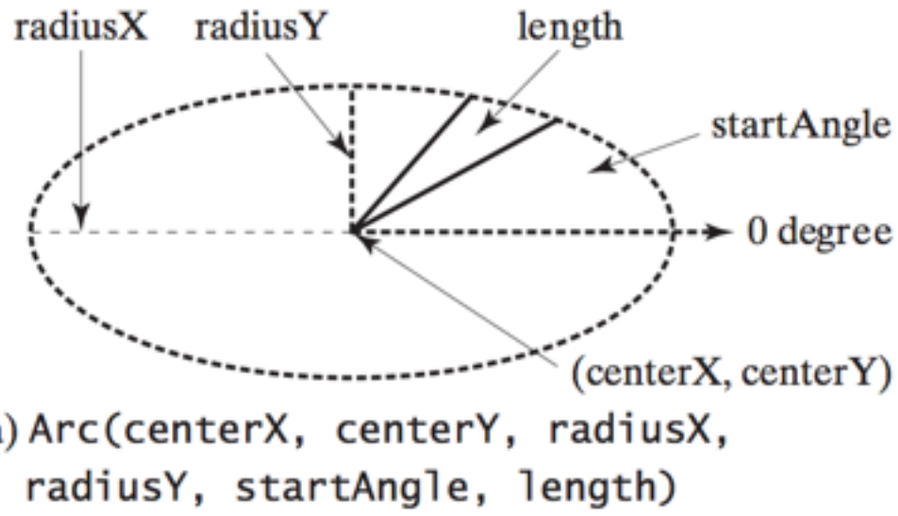
形状--矩形

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Ellipse;

public class ShowEllipse extends Application{
	@Override
	public void start( Stage primaryStage ){
		Pane pane = new Pane();

		for( int i = 0; i < 10; i++ ){
			Ellipse e1 = new Ellipse( 150, 100, 100, 50 );
			e1.setStroke( Color.color( Math.random(), Math.random(), Math.random() ) );
			e1.setFill( Color.WHITE );
			e1.setRotate( i * 180 / 16 );
			pane.getChildren().add( e1 );
		}

		Scene scene = new Scene( pane, 300, 200 );
		primaryStage.setTitle( "ShowEllipse" );
		primaryStage.setScene( scene );
		primaryStage.show();
	}
}
```



形状--圆弧

- **Arc**
  - 椭圆的一部分，参数为centerX, centerY, radiusX, radiusY, startAngle, length, 以及圆弧类型(ArcType.OPEN, ArcType.CHORD, ArcType.ROUND)
  - 其中，startAngle为起始角度，length为扫过的角度
  - 0度为水平向右，逆时针方向为正



(a) Arc(centerX, centerY, radiusX, radiusY, startAngle, length)

javafx.scene.shape.Arc

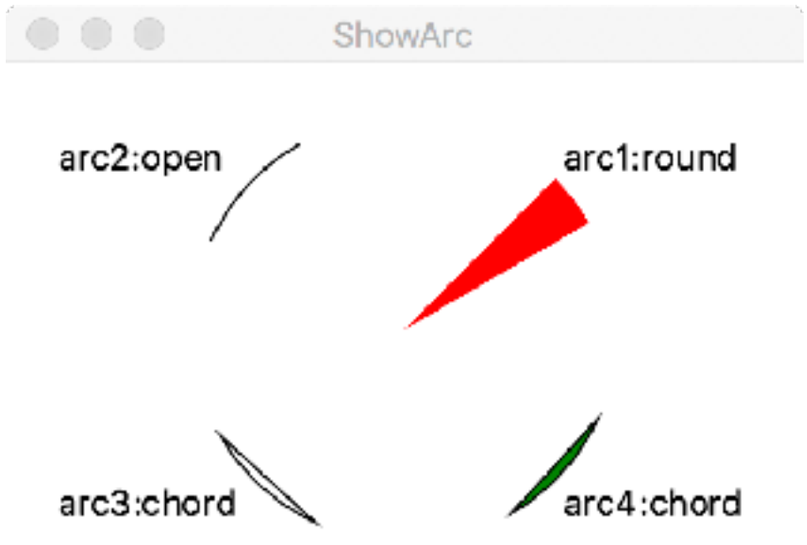
```
-centerX: DoubleProperty
-centerY: DoubleProperty
-radiusX: DoubleProperty
-radiusY: DoubleProperty
-startAngle: DoubleProperty
-length: DoubleProperty
-type: ObjectProperty<ArcType>
```

```
+Arc()
+Arc(x: double, y: double,
   radiusX: double, radiusY:
   double, startAngle: double,
   length: double)
```

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

The x-coordinate of the center of the ellipse (default 0).
The y-coordinate of the center of the ellipse (default 0).
The horizontal radius of the ellipse (default: 0).
The vertical radius of the ellipse (default: 0).
The start angle of the arc in degrees.
The angular extent of the arc in degrees.
The closure type of the arc (`ArcType.OPEN`, `ArcType.CHORD`, `ArcType.ROUND`).

Creates an empty `Arc`.
Creates an `Arc` with the specified arguments.

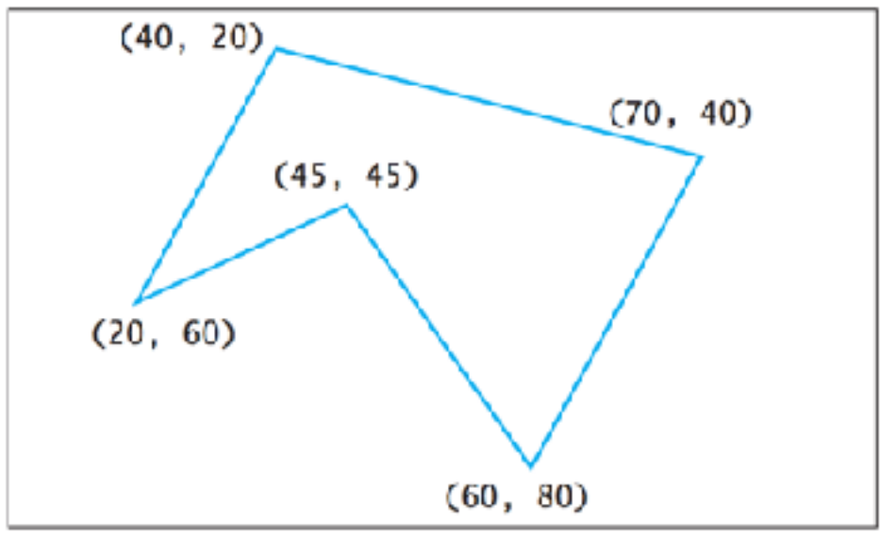


形状--圆弧

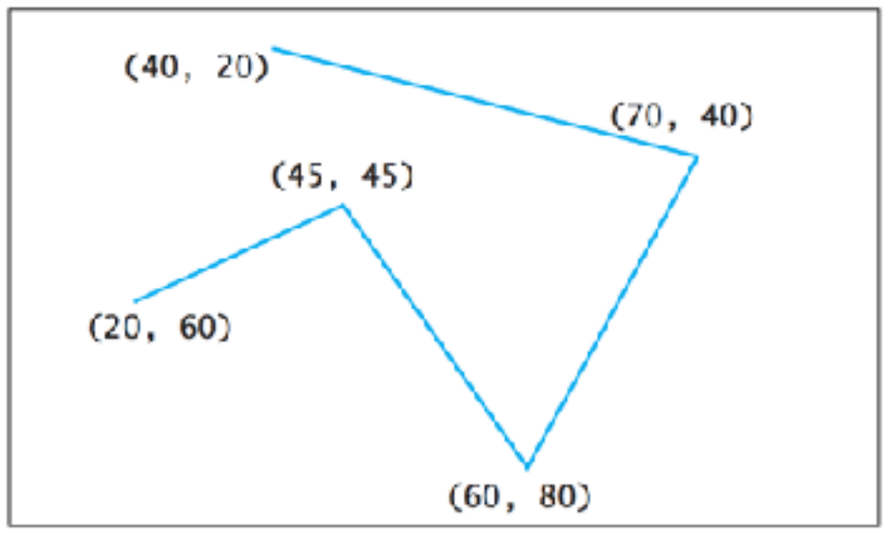
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Arc;
import javafx.scene.shape.ArcType;
import javafx.scene.text.Text;

public class ShowArc extends Application{
	@Override
	public void start( Stage primaryStage ){
		Pane pane = new Pane();

		Arc arc1 = new Arc( 150, 100, 80, 80, 30, 15 );
		arc1.setFill( Color.RED );
		arc1.setType( ArcType.ROUND );
		pane.getChildren().add( new Text( 210, 40,
"arc1:round" ) ) ;
		pane.getChildren().add( arc1 );

		Arc arc2 = new Arc( 150, 100, 80, 80, 30 + 90,
35 );
		arc2.setFill( Color.WHITE );
		arc2.setType( ArcType.OPEN );
		arc2.setStroke( Color.BLACK );
		pane.getChildren().add( new Text( 20, 40,
"arc2:open" ) );
		pane.getChildren().add( arc2 );

		Arc arc3 = new Arc( 150, 100, 80, 80, 30 +
180, 35 );
		arc3.setFill( Color.WHITE );
		arc3.setType( ArcType.CHORD );
		arc3.setStroke( Color.BLACK );
		pane.getChildren().add( new Text( 20, 170,
"arc3:chord" ) );
		pane.getChildren().add( arc3 );

		Arc arc4 = new Arc( 150, 100, 80, 80, 30 +
270, 35 );
		arc4.setFill( Color.GREEN );
		arc4.setType( ArcType.CHORD );
		arc4.setStroke( Color.BLACK );
		pane.getChildren().add( new Text( 210, 170,
"arc4:chord" ) );
		pane.getChildren().add( arc4 );

		Scene scene = new Scene( pane, 300, 200 );
		primaryStage.setTitle( "ShowArc" );
		primaryStage.setScene( scene );
		primaryStage.show();

	}
}
```

形状--多边形

- **Polygon**
  - 由一组点连接所形成的封闭多边形
- **Polyline**
  - 由一组点连接所形成的开放多边形



(a) Polygon



(b) Polyline

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.shape.Arc

```
+Polygon()
+Polygon(double... points)
+getPoints():
   ObservableList<Double>
```

Creates an empty `Polygon`.

Creates a `Polygon` with the given points.

Returns a list of double values as x-and y-coordinates of the points.

形状--多边形

```
import javafx.application.Application;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Polygon;

public class ShowPolygon extends Application{
	@Override
	public void start( Stage primaryStage ){
		Pane pane = new Pane();
		Polygon polygon = new Polygon();
		pane.getChildren().add( polygon );
		polygon.setFill( Color.WHITE );
		polygon.setStroke( Color.BLACK );
		ObservableList<Double> list =
polygon.getPoints();

		final double WIDTH = 200, HEIGHT = 200;
		double centerX = WIDTH / 2, centerY =
HEIGHT / 2;
		double radius = Math.min( WIDTH, HEIGHT ) *
0.4;

		for( int i = 0; i < 6; i++ ){
			list.add( centerX + radius * Math.cos( 2
* i * Math.PI / 6 ) );
			list.add( centerY + radius * Math.sin( 2 *
i * Math.PI / 6 ) );
		}

		Scene scene = new Scene( pane, WIDTH,
HEIGHT );
		primaryStage.setTitle( "ShowPolygon" );
		primaryStage.setScene( scene );
		primaryStage.show();
	}
}
```