Applicazioni grafiche e finestre Frame

Per visualizzare un frame:

1. Costruire un oggetto della classe `JFrame`:
   ```
   JFrame frame = new JFrame();
   ```
2. Stabilire le dimensioni del frame:
   ```
   frame.setSize(300, 400);
   ```
3. Eventualmente, definire il titolo del frame:
   ```
   frame.setTitle("An Empty Frame");
   ```
4. Fissare la "default close operation":
   ```
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   ```
5. Rendere il frame visibile:
   ```
   frame.setVisible(true);
   ```

Un frame

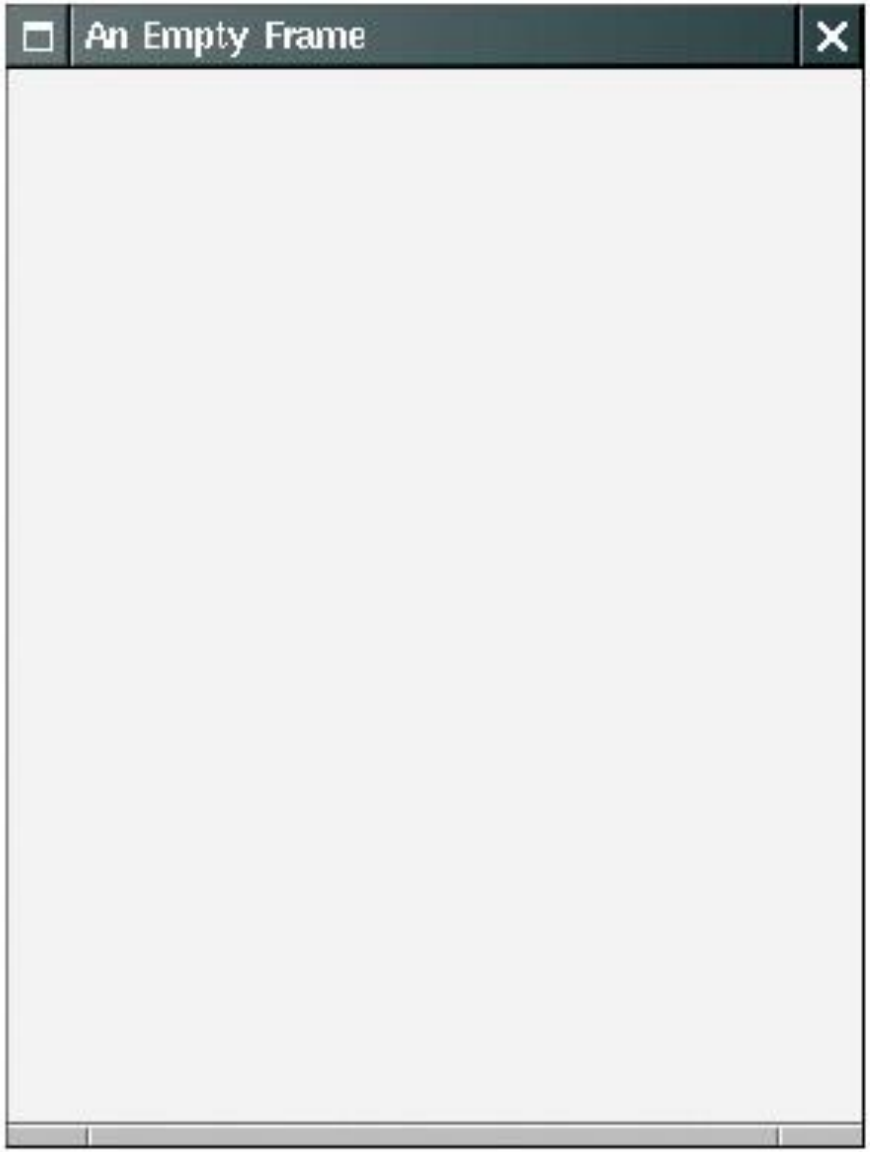

Figure 21 A Frame Window

ch02/emptyframe/EmptyFrameViewer.java

```
01: import javax.swing.JFrame;
02:
03: public class EmptyFrameViewer
04: {
05:    public static void main(String[] args)
06:    {
07:       JFrame frame = new JFrame();
08:
09:       frame.setSize(300, 400);
10:       frame.setTitle("An Empty Frame");
11:       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12:
13:       frame.setVisible(true);
14:    }
15: }
```

Disegnare in un frame

- Per visualizzare un disegno in un frame, definire innanzi tutto una classe che estende la classe `JComponent`.
- Sistemare le istruzioni di disegno dentro il metodo `paintComponent`. Tale metodo viene chiamato ogni volta che il componente necessita di essere ridisegnato.

```
public class RectangleComponent extends JComponent
{
   public void paintComponent(Graphics g)
   {
      Drawing instructions go here
   }
}
```

Classi `Graphics` e `Graphics2D`

- `Graphics` è una classe che consente di manipolare le caratteristiche grafiche (ad esempio il colore corrente)
- `Graphics2D` è una classe con metodi per disegnare forme
- Usare una conversione (cast) per recuperare l'oggetto `Graphics2D` dal parametro `Graphics`:

```
public class RectangleComponent extends JComponent
{
   public void paintComponent(Graphics g)
   {
      // Recover Graphics2D
      Graphics2D g2 = (Graphics2D) g;
      . . .
   }
}
```

Continua

Classes `Graphics` and `Graphics2D` (cont.)

- Chiamare il metodo `draw` della classe `Graphics2D` per disegnare forme, come rettangoli, ellissi, segmenti lineari, poligoni ed archi:

```
public class RectangleComponent extends JComponent
{
   public void paintComponent(Graphics g)
   {
      . . .
      Rectangle box = new Rectangle(5, 10, 20, 30);
      g2.draw(box);
      . . .
   }
}
```

Disegnare rettangoli

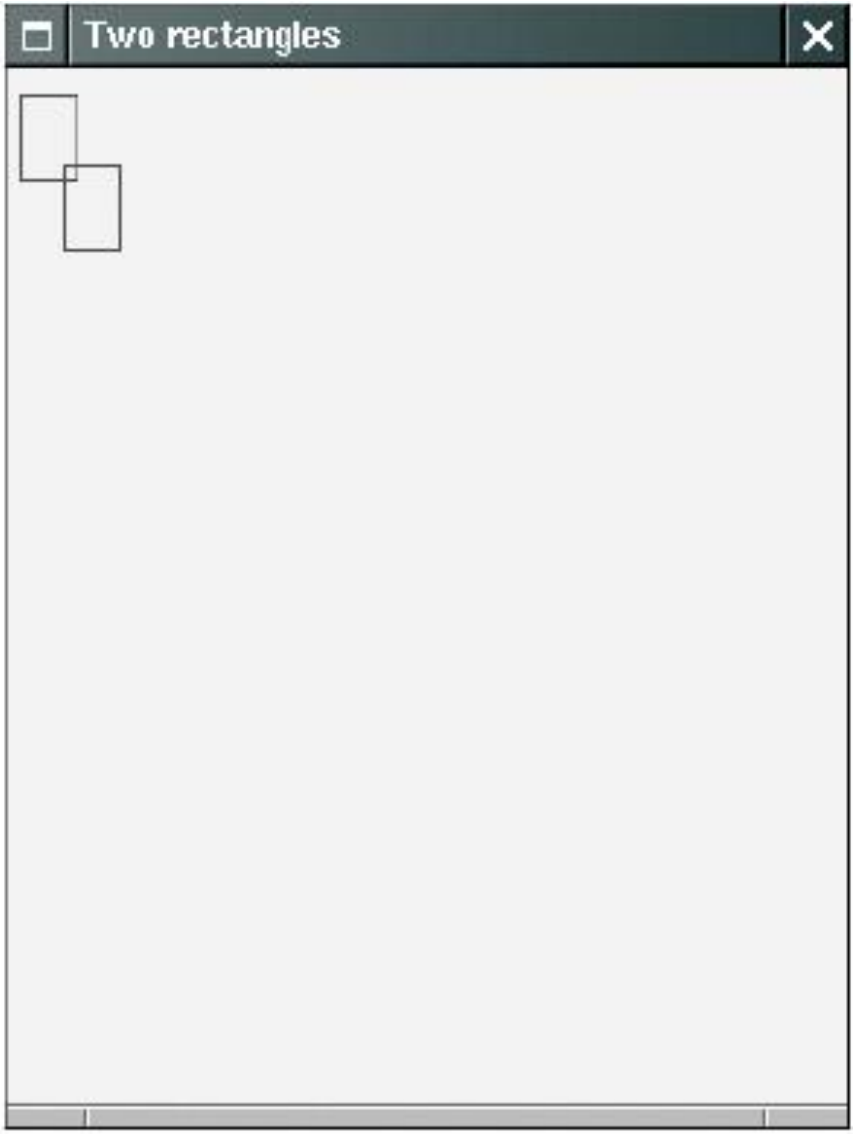

Figure 22
Drawing Rectangles

ch02/rectangles/RectangleComponent.java

```
01: import java.awt.Graphics;
02: import java.awt.Graphics2D;
03: import java.awt.Rectangle;
04: import javax.swing.JComponent;
05:
06: /**
07:    A component that draws two rectangles.
08: */
09: public class RectangleComponent extends JComponent
10: {
11:    public void paintComponent(Graphics g)
12:    {
13:       // Recover Graphics2D
14:       Graphics2D g2 = (Graphics2D) g;
15:
16:       // Construct a rectangle and draw it
17:       Rectangle box = new Rectangle(5, 10, 20, 30);
18:       g2.draw(box);
19:
```

Continued

ch02/rectangles/RectangleComponent.java (cont.)

```
20:         // Move rectangle 15 units to the right and 25 units down
21:         box.translate(15, 25);
22:
23:         // Draw moved rectangle
24:         g2.draw(box);
25:     }
26: }
```

Usare un componente

1. Costuire un frame.
2. Costruire un oggetto della propria classe componente:
   ```
   RectangleComponent component = new RectangleComponent();
   ```
3. Aggiungere il componente al frame:
   ```
   frame.add(component);
   ```
4. Rendere il frame visibile

ch02/rectangles/RectangleViewer.java

```
01: import javax.swing.JFrame;
02:
03: public class RectangleViewer
04: {
05:    public static void main(String[] args)
06:    {
07:       JFrame frame = new JFrame();
08:
09:       frame.setSize(300, 400);
10:       frame.setTitle("Two rectangles");
11:       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12:
13:       RectangleComponent component = new RectangleComponent();
14:       frame.add(component);
15:
16:       frame.setVisible(true);
17:    }
18: }
```

Ellissi

- `Ellipse2D.Double` descrive un ellisse
- Non si userà la classe `.Float`
- Questa classe è una classe interna – non ha importanza per noi eccetto per l'istruzione `import`:

```
import java.awt.geom.Ellipse2D; // no .Double
```

- Bisogna istanziare *e disegnare* la forma

```
Ellipse2D.Double ellipse = new Ellipse2D.Double(x, y,
   width, height); g2.draw(ellipse);
```

Un ellisse

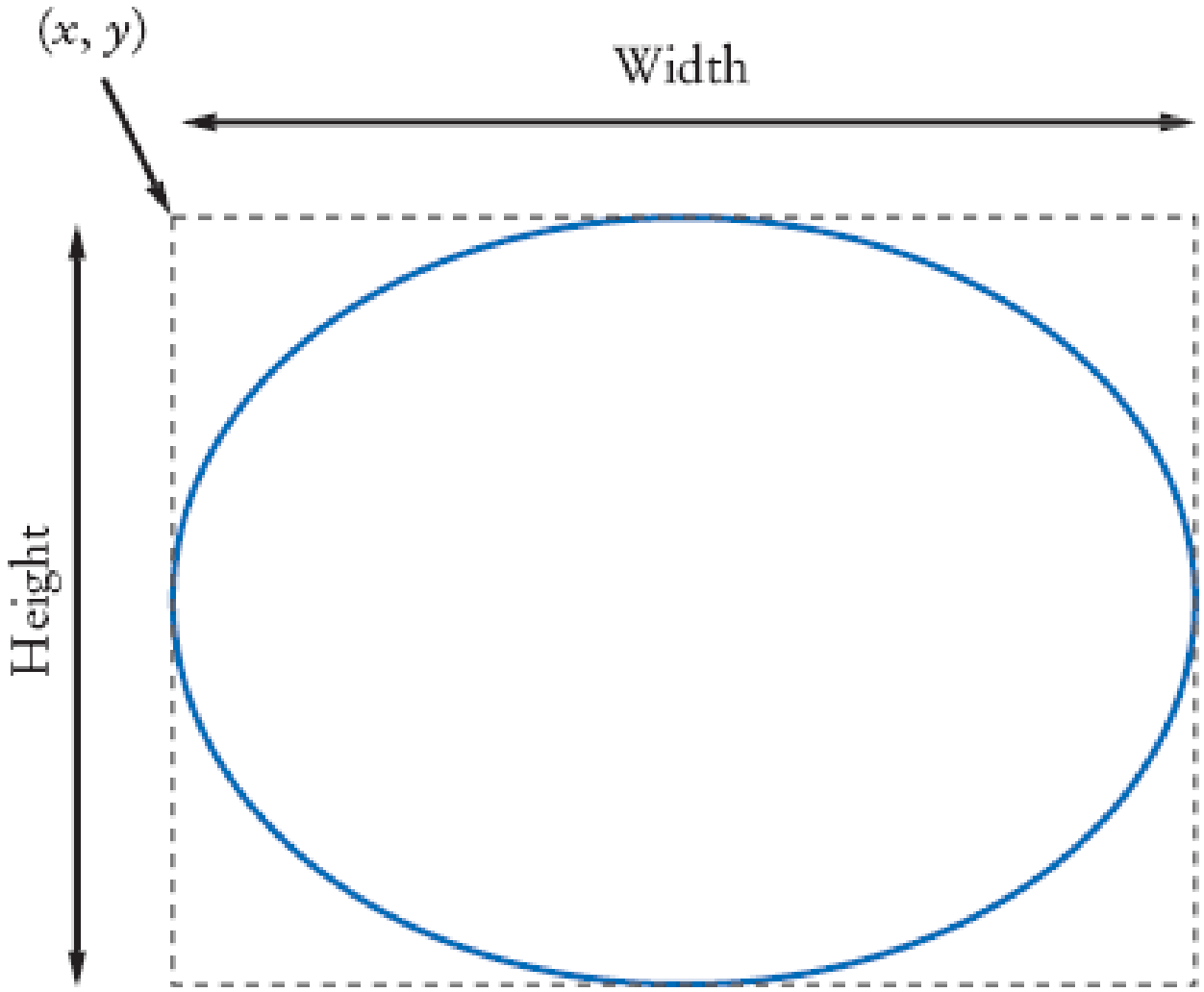

Figure 23 An Ellipse and Its Bounding Box

Disegnare linee

Per disegnare una linea:

```
Line2D.Double segment = new Line2D.Double(x1, y1, x2, y2);
g2.draw(segment);
```

o,

```
Point2D.Double from = new Point2D.Double(x1, y1);
Point2D.Double to = new Point2D.Double(x2, y2);
Line2D.Double segment = new Line2D.Double(from, to);
g2.draw(segment);
```

Disegnare testo

```
g2.drawString("Message", 50, 100);
```

Figure 24 Basepoint and Baseline

Colori

- Colori standard: `Color.BLUE`, `Color.RED`, `Color.PINK etc.`
- Si può specificare il rosso, il verde ed il blu con valori tra `0` e `255`

```
Color magenta = new Color(255, 0, 255);
```

- Il colore viene stabilito attraverso il contesto grafico

```
g2.setColor(magenta);
```

- Il colori è usato per disegnare e riempire forme

```
g2.fill(rectangle); // filled with current color
```

Colori predefiniti e valori RGB (Red Green Blue)

Color	RGB Value
Color.BLACK	0, 0, 0
Color.BLUE	0, 0, 255
Color.CYAN	0, 255, 255
Color.GRAY	128, 128, 128
Color.DARKGRAY	64, 64, 64
Color.LIGHTGRAY	192, 192, 192
Color.GREEN	0, 255, 0
Color.MAGENTA	255, 0, 255
Color.ORANGE	255, 200, 0
Color.PINK	255, 175, 175
Color.RED	255, 0, 0
Color.WHITE	255, 255, 255
Color.YELLOW	255, 255, 0