



كلية الهندسة المعلوماتية

برمجة 3

Java Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع السادس

GUI

الفصل الثاني 2022-2023

العنوان	رقم الفقرة
استخدام مربعات الحوار: المدخلات والمخرجات الأساسية مع مربعات الحوار	3.9
إنشاء رسومات بسيطة عرض الخطوط ورسما على الشاشة	4.14
رسم المستطيلات والأشكال البيضاوية استخدام الأشكال لتمثيل البيانات	5.10
الألوان والأشكال المعبأة رسم قوس قذح ورسومات عشوائية	6.13
رسم الأقواس رسم الحلزونات بالأقواس	7.13
استخدام الكائنات مع الرسومات تخزين الأشكال ككائنات	8.18
عرض النص والصور باستخدام الملصقات توفير معلومات الحالة	9.8
الرسم باستخدام تعدد الأشكال: تحديد أوجه التشابه بين الأشكال التمرين	10.8
توسيع الواجهة: استخدام مكونات واجهة المستخدم الرسومية ومعالجة الأحداث	14.17

كل برنامج رسومي يستخدم نافذة إطار window frame أو أكثر ولكل نافذة إطار شريط عنوان titel bar وحدود border لكي تظهر الإطار نستخدم الصنف JFream من الحزمة javax.swing ويجب:

1- إنشاء كائن من JFream وفق `JFrame appli = new JFrame("First");`

2- تجديد مقاس الإطار من الطريقة `setSize` . `appli.setSize(300, 300);`

3- إضافة الرسمة أو ماتم تجميعها ونرغب بعرضه إلى الإطار `appli.add( panel );`

4- جعل الإطار مرئي تستخدم الطريقة `show` لجعل مدير عرض النافذة `window manager` يعرضها افتراضيا هي `false`.

`appli.setVisible( true );`

5- عند تنفيذ البرنامج يتم إظهار الإطار وينتهي تنفيذ `main` ولكن يظل البرنامج يعمل والإطار ظاهر على الشاشة ويمكن تحريكه وتغيير حجمه و ... ، وعند إغلاق نافذة الإطار بالضغط على أيقونة الإغلاق من شريط العنوانه يظل البرنامج يعمل ولا يحدث شيء سوى إختفاء الإطار، ومن أجل إنهاء البرنامج يجب استخدام `System.exit(0)` والتي يجب أن تكون بنهاية `main` ولكن تخلق مشكلة جديدة وهي ظهور النافذة للحظة وجيزة وينتهي فوراً والرغبة هي إنهاء البرنامج عند الضغط المستخدم على أيقونة الغلق في

شريط العنوان وهنا نجد اسهل طريقة استخدام المنهج : `appli.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`

أو معالجة حدث النقر على أيقونة الغلق من أجل إنهاء البرنامج إضافة على إغلاق النافذة

- class Graphics من الحزمة (java.awt)، والتي توفر طرقاً مختلفة لرسم النص والأشكال على الشاشة.
- الصنف JPanel من الحزمة (javax.swing)، والتي توفر مساحة يمكن الرسم عليها.

```
public class DrawPanel extends JPanel
```

`extends` الكلمة الأساسية للإشارة إلى أن الصنف DrawPanel هو نوع محسن من JPanel وارث له.

• الكلمة الأساسية `extends` تمثل ما يسمى بعلاقة الوراثة التي يبدأ فيها صنفنا الجديد DrawPanel بالأعضاء الحاليين (البيانات والأساليب) من فئة JPanel .

• كل لوحة JPanel بما في ذلك DrawPanel، لديها طريقة `paintComponent`.

• ينادي النظام تلقائياً في كل مرة يحتاج فيها إلى عرض DrawPanel المنهج `paintComponent()` ويجب التصريح عنها `public void paintComponent(Graphics g)`، خلاف ذلك، لن يسمح النظام بمناداتها والعبارة الأولى فيها عندما تكتبها (تحملها تحملاً زائداً) هي `super.paintComponent(g)`;

• يتم استدعاء هذه الطريقة عندما يتم عرض JPanel لأول مرة على الشاشة، وعندما يتم تغطيتها `hidden` ثم الكشف عنها بواسطة نافذة أخرى على الشاشة، وعندما يتم تغيير حجم النافذة التي تظهر فيها.

- ✓ رسم الأقواس في Java يشبه رسم الأشكال البيضاوية - القوس هو مجرد جزء من الشكل البيضاوي.
- ✓ منهج الرسومات `fillArc` من الصنف `Graphics` يرسم قوساً ممتلئاً.
- ✓ يتطلب المنهج `fillArc` ستة معاملات.
  - الأربعة الأولى تمثل المستطيل المحيط الذي سيرسم فيه القوس.
  - المعامل الخامس هو زاوية البداية ، مع عدم وجود معامل تشير إلى الصفر على محور `x`.
  - والسادس يحدد مقدار المسح ، أو مقدار القوس المراد تغطيته.
  - يتم قياس زاوية البداية والمسح بالدرجات.
  - المسح الموجب يرسم القوس بعكس اتجاه عقارب الساعة والسالب معها.
- ✓ تتطلب طريقة `drawArc` نفس المعلومات مثل `fillArc`، ولكنها ترسم حافة القوس بدلاً من تعبئتها.
- ✓ منهج `setBackground` يغير لون الخلفية لمكون واجهة المستخدم الرسومية.

```
1 // Fig. 7.25: DrawRainbow.java
2 // Demonstrates using colors in an array.
3 import java.awt.Color;
4 import java.awt.Graphics;
5 import javax.swing.JPanel;
6
7 public class DrawRainbow extends JPanel
8 {
9     // define indigo and violet
10    private final static Color VIOLET = new Color( 128, 0, 128 );
11    private final static Color INDIGO = new Color( 75, 0, 130 );
12
13    // colors to use in the rainbow, starting from the innermost
14    // The two white entries result in an empty arc in the center
15    private Color[] colors =
16        { Color.WHITE, Color.WHITE, VIOLET, INDIGO, Color.BLUE,
17          Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
18
19    // constructor
20    public DrawRainbow()
21    {
22        setBackground( Color.WHITE ); // set the background to white
23    } // end DrawRainbow constructor
24
```

**Fig. 7.25** | Drawing a rainbow using arcs and an array of colors. (Part I of 2.)

```
25 // draws a rainbow using concentric arcs
26 public void paintComponent( Graphics g )
27 {
28     super.paintComponent( g );
29
30     int radius = 20; // radius of an arc
31
32     // draw the rainbow near the bottom-center
33     int centerX = getWidth() / 2;
34     int centerY = getHeight() - 10;
35
36     // draws filled arcs starting with the outermost
37     for ( int counter = colors.length; counter > 0; counter-- )
38     {
39         // set the color for the current arc
40         g.setColor( colors[ counter - 1 ] );
41
42         // fill the arc from 0 to 180 degrees
43         g.fillArc( centerX - counter * radius,
44                 centerY - counter * radius,
45                 counter * radius * 2, counter * radius * 2, 0, 180 );
46     } // end for
47 } // end method paintComponent
48 } // end class DrawRainbow
```

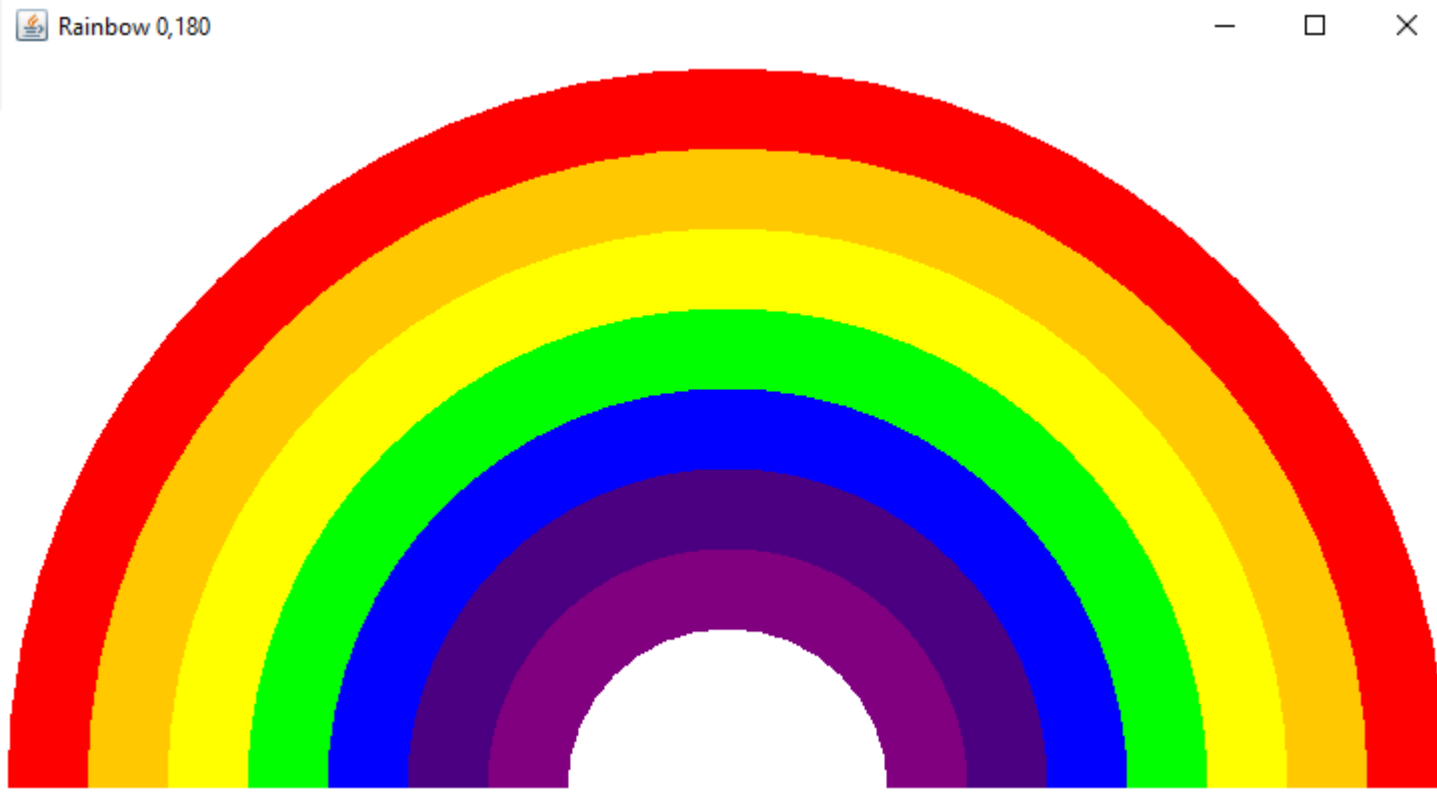
**Fig. 7.25** | Drawing a rainbow using arcs and an array of colors. (Part 2 of 2.)



```
1 // Fig. 7.26: DrawRainbowTest.java
2 // Test application to display a rainbow.
3 import javax.swing.JFrame;
4
5 public class DrawRainbowTest
6 {
7     public static void main( String[] args )
8     {
9         DrawRainbow panel = new DrawRainbow();
10        JFrame application = new JFrame();
11
12        application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
13        application.add( panel );
14        application.setSize( 400, 250 );
15        application.setVisible( true );
16    } // end main
17 } // end class DrawRainbowTest
```

**Fig. 7.26** | Creating JFrame to display a rainbow. (Part 1 of 2.)





**Fig. 7.26** | Creating JFrame to display a rainbow. (Part 2 of 2.)

# Create a special color

```
final Color VIOLET = new Color( 128, 0, 128 );
final Color INDIGO = new Color( 75, 0, 130 );
```

اتعريف لونين جديدين

- أن ألوان قوس قزح هي الأحمر والبرتقالي والأصفر والأخضر والأزرق والنيلى والبنفسجي.
- تحتوي Java فقط على ثوابت سابقة التعريف للألوان الخمسة الأولى.

- الاسطر تهيئة مصفوفة مع ألوان قوس قزح، بدءاً من الأقواس الأعمق أولاً يبدأ المصفوفة بعنوان Color.WHITE، والتي، كما سترى قريباً، هي لرسم الأقواس الفارغة في مركز قوس قزح.

```
private Color colors[] = { Color.WHITE, Color.WHITE, VIOLET, INDIGO,
Color.BLUE, Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
```

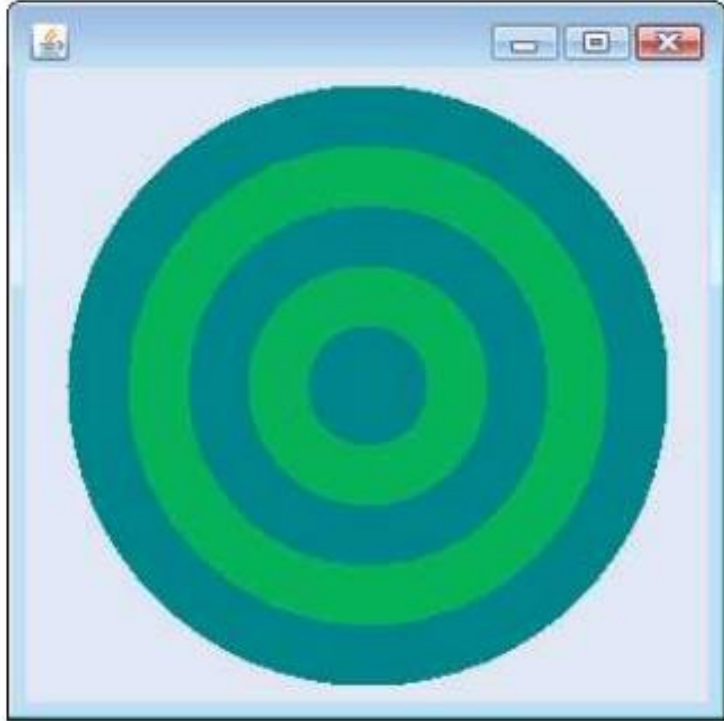
- يمكن تهيئة متغيرات الحالة عند إعلانها. يحتوي المنشئ على عبارة واحدة تستدعي طريقة setBackground (الذي تم توارثه من الفئة JPanel) بالمتغير Color.WHITE. تأخذ مجموعة setBackground وسيطة لون واحدة وتجعل الخلفية للمكون من ذلك اللون.
  - الطريقة paintComponent تعلن عن متغير يعبر عن نصف قطر الدائرة التي سترسم، أي يحدد سمك كل قوس. تحدد المتغيرات المحلية centerX و centerY موقع نقطة الوسط على قاعدة قوس قزح. (منتصف X وعلى بعد 10 من كامل h).
  - تستخدم الحلقة counter للتحكم والعد العكسي من نهاية المصفوفة، مع رسم أكبر قوس أولاً ووضع كل قوس أصغر متتابع أعلى القوس السابق. \* يعين السطر 40 اللون لرسم القوس الحالي من المصفوفة. السبب في أن لدينا مداخل Color.WHITE في بداية المصفوفة هي إنشاء قوس فارغ في المركز. خلاف ذلك، فإن مركز قوس قزح يكون مجرد دائرة نصف دائرة البنفسجي الصلبة.
- [ملاحظة: يمكنك تغيير ألوان فردية وعدد الإدخالات في المصفوفة لإنشاء تصميمات جديدة.]

# Rainbow 20, 18

# Rainbow 20, -18



A bulls-eye with two alternating,  
random colors.



ليكن المطلوب رسم الشكل السابق

# Draw many shapes 1



# DrawRainbow21 1

```
// Fig. 5.26: Shapes2.java
// Demonstrates drawing different Shapes2.
import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JPanel;
import java.security.SecureRandom;
//import java.util.Random; // program uses class Random
public class DrawRainbow21 extends JPanel
{
    // Define indigo and violet
    final Color VIOLET = new Color( 128, 0, 128 );
    final Color INDIGO = new Color( 75, 0, 130 );
    int x1,y1,w,h;
    // colors to use in the rainbow, starting from the innermost
    // The two white entries result in an empty arc in the center
    private Color colors[] =
        { Color.WHITE, Color.WHITE, VIOLET, INDIGO, Color.BLUE,
          Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
}
```

# Draw many shapes 2



# DrawRainbow21 2

```
// constructor
private int choice; // user's choice of which shape to draw

// constructor sets the user's choice

public DrawRainbow21(int userChoice)
{choice = userChoice;
  setBackground( Color.WHITE ); // set the background to white
} // end DrawRainbow21 constructor

// draws a rainbow using concentric circles
public void paintComponent( Graphics g )
{
  super.paintComponent( g );
}

// draws a cascade of Shapes21 starting from the top-left corner///
  SecureRandom randomNumbers = new SecureRandom();
// SecureRandom number generator
```

# Draw many shapes 3



# DrawRainbow21 3

```
// pick the shape based on the user's choice
switch ( choice )
{case 1: // draw DrawRainbow2
    int radius = 40;// radius of an arch
    // draw the rainbow near the bottom-center
    int centerX = getWidth() / 2;
    int centerY = getHeight() - 10;

    // draws filled arcs starting with the outermost
    for ( int counter = colors.length; counter > 0; counter-- )
    { // set the color for the current arc
      g.setColor( colors[ counter - 1 ] );

      // fill the arc from 0 to 180 degrees
      g.fillArc( centerX - counter * radius,
                centerY - counter * radius,
                counter * radius * 2, counter * radius * 2, 45, 90 );
    }
}
```



# Draw many shapes 4



# DrawRainbow21 4

```
// set the color for the current arc
    /*g.setColor( colors[ counter - 1 ] );
    g.fillOval( 10 + counter * 10, 10 + counter * 10,
    240 - counter * 20, 240 - counter * 20 );*/
    } break;
case 2: // set the color for the current arc
    for ( int counter = 8; counter > 0; counter-- )
    { g.setColor( colors[ counter - 1 ] );
        g.fillOval(130-counter * 10,130-counter *10,counter * 20,counter * 20 );
        //g.fillOval(30+counter *10,30+counter *10,200-counter *20,200-counter *20);
        /*g.fillOval(10+counter*10,10+counter*10,240-counter*20,240-counter*20); */
        x1=130 - counter *10; y1=130 - counter * 10; w= counter * 20; h=counter * 20;
        System.out.println("x1=" +x1+ " y1= "+y1+ " w= "+w+" h= "+h);
    } break;
case 3: // stores each random integer generated shapesType,shapesColor,(x1,y1)
begin ,widthn ,height
```

# Draw many shapes 5



# DrawRainbow21 5

```
for ( int counter = 10; counter > 0; counter-- )
{   int shapesType, shapesColor, x1, y1, x2, y2;
    shapesType = 1 + randomNumbers.nextInt( 2 );
    x1 = 1 + randomNumbers.nextInt( 200 ); y1 = 1 + randomNumbers.nextInt( 200 );
    w = 1 + randomNumbers.nextInt( 188 ); h = 1 + randomNumbers.nextInt( 188 );
    System.out.println("x1=" +x1+ " y1=" +y1+ " w=" +w+" h=" +h);
    //if((x1+x2)>400) x2=380;  if((y1+y2)>400) y2=370;
    if(shapesType==1)
{   // draw rectangles and set the color for the current rectangles
    shapesColor = 2 + randomNumbers.nextInt( 6 );
    g.setColor( colors[ shapesColor ] ); g.fillRect( x1, y1, w, h ); }
else { // draw ovals and set the color for the current ovals
    shapesColor = 2 + randomNumbers.nextInt( 6 );
    g.setColor( colors[ shapesColor ] ); g.fillOval( x1, y1, w, h );} //break;
    } } // end for
} // end method paintComponent
} // end class DrawRainbow21
```

# Draw many shapes 6



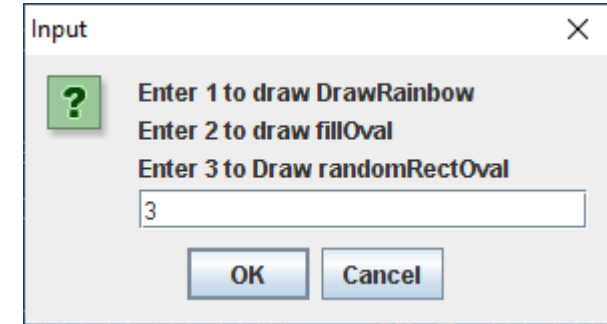
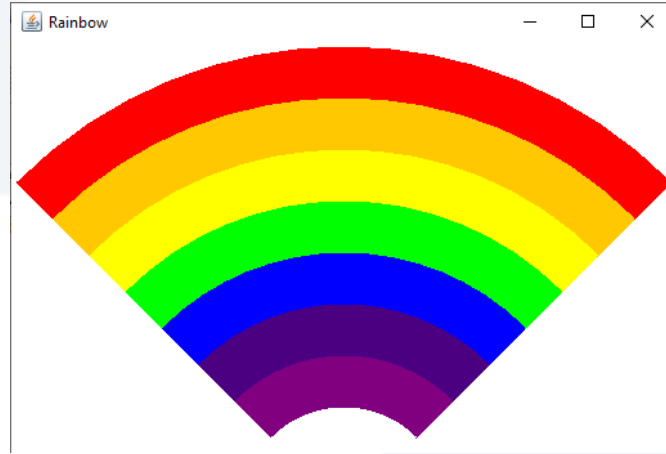
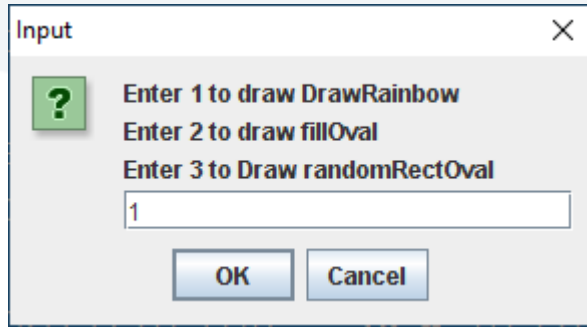
## DrawRainbow21Test

```
public class DrawRainbow21Test
{
    public static void main( String[] args )
    {
        // obtain user's choice
        String input = JOptionPane.showInputDialog(
            "Enter 1 to draw DrawRainbow \n" +
            "Enter 2 to draw fillOval\n"+
            "Enter 3 to Draw randomRectOval\n");
        int choice = Integer.parseInt( input ); // convert input to int
        // create the panel with the user's input
        DrawRainbow21 panel = new DrawRainbow21( choice );
        JFrame application = new JFrame(); // creates a new JFrame
        application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        application.setSize( 400, 400 ); // set the desired size
        application.add( panel ); // add the panel to the frame
        application.setVisible( true ); // show the frame
    } // end main
} // end class DrawRainbow1Test
```

# Drawing Random Rect and Oval

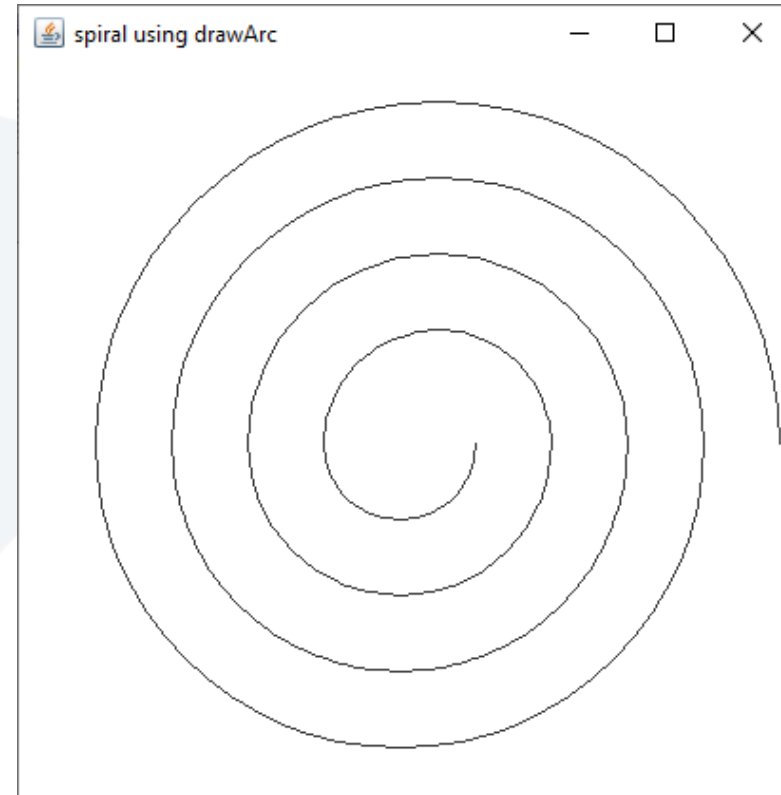
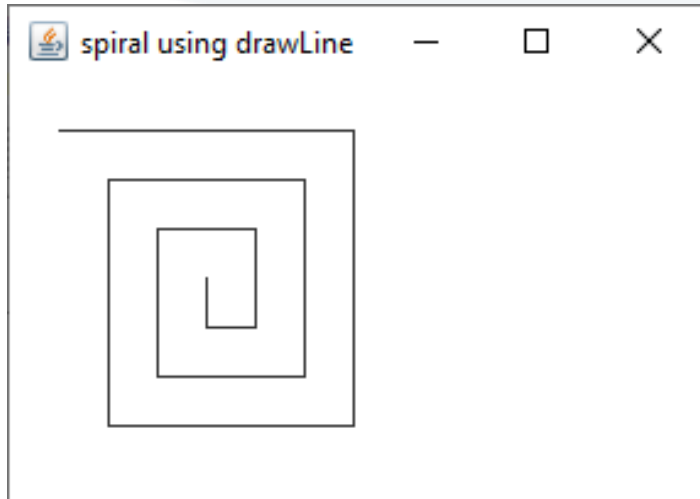


## Draw many shapes 7



# Drawing spiral

## GUI and Graphics Case Study Exercise



ليكن المطلوب رسم الشكل السابق

- ✓ يخزن المثال التالي معلومات حول الأشكال المعروضة حتى نتمكن من إعادة إنتاجها في كل مرة يستدعي فيها النظام `paintComponent`.
- ✓ سننشئ فئات شكل "ذكية" يمكنها رسم نفسها باستخدام كائن رسومات من نمط `Graphics`.
- ✓ يوضح الشكل 8.21 صنف `MyLine`، الذي يحتوي على كل هذه القدرات.
- ✓ يتكرر المنهج `paintComponent` في الصنف `DrawPanel` من خلال رسم مجموعة من الكائنات `MyLine`.
- في كل استدعاء للكائن من الصنف `MyLine` يتم تكرار الاستدعاء لمنهج الرسم `draw` ويمرره كائن الرسومات `Graphics` من أجل الرسم على اللوحة.

```
import java.awt.Color;
import java.awt.Graphics;
public class MyLine
{
    private int x1; // x-coordinate of first endpoint
    private int y1; // y-coordinate of first endpoint
    private int x2; // x-coordinate of second endpoint
    private int y2; // y-coordinate of second endpoint
    private Color myColor; // color of this shape
    // constructor with input values
    public MyLine( int x1, int y1, int x2, int y2, Color color )
    {
        this.x1 = x1; // set x-coordinate of first endpoint
        this.y1 = y1; // set y-coordinate of first endpoint
        this.x2 = x2; // set x-coordinate of second endpoint
        this.y2 = y2; // set y-coordinate of second endpoint
        myColor = color; // set the color
    }
    // end MyLine constructor // Actually draws the line
    public void draw( Graphics g )
    {
        g.setColor( myColor ); g.drawLine( x1, y1, x2, y2 );
    }
    // end method draw
}
// end class MyLine
```



## DrawPanel2.java **الصف**



uses class MyLine to  
draw random lines 1

```
// Fig. 8.19: DrawPanel.java
// Program that uses class MyLine
// to draw random lines.
import java.awt.Color;
import java.awt.Graphics;
import java.util.Random;
import javax.swing.JPanel;
public class DrawPanel2 extends JPanel
{
    private Random randomNumbers = new Random();
    private MyLine[] lines; // array on lines
    // constructor, creates a panel with random shapes
    public DrawPanel2()
    {
        setBackground( Color.WHITE );
        lines = new MyLine[ 5 + randomNumbers.nextInt( 5 ) ]; // create lines
        for ( int count = 0; count < lines.length; count++ )
        { // generate random coordinates
            int x1 = randomNumbers.nextInt( 300 );
            int y1 = randomNumbers.nextInt( 300 );
            int x2 = randomNumbers.nextInt( 300 );
            int y2 = randomNumbers.nextInt( 300 );
```

DrawPanel2.java **الصف**



uses class MyLine to  
draw random lines 2

```
        // generate a random color
        Color color = new Color( randomNumbers.nextInt( 256 ),
            randomNumbers.nextInt( 256 ), randomNumbers.nextInt( 256 ) );
        // add the line to the list of lines to be displayed
        lines[ count ] = new MyLine( x1, y1, x2, y2, color );
    }        // end for
}        // end DrawPanel constructor
// for each shape array, draw the individual shapes
public void paintComponent( Graphics g )
{
    super.paintComponent( g );
    // draw the lines
    for ( MyLine line : lines )
        line.draw( g );
}        // end method paintComponent
}        // end class DrawPanel2
```

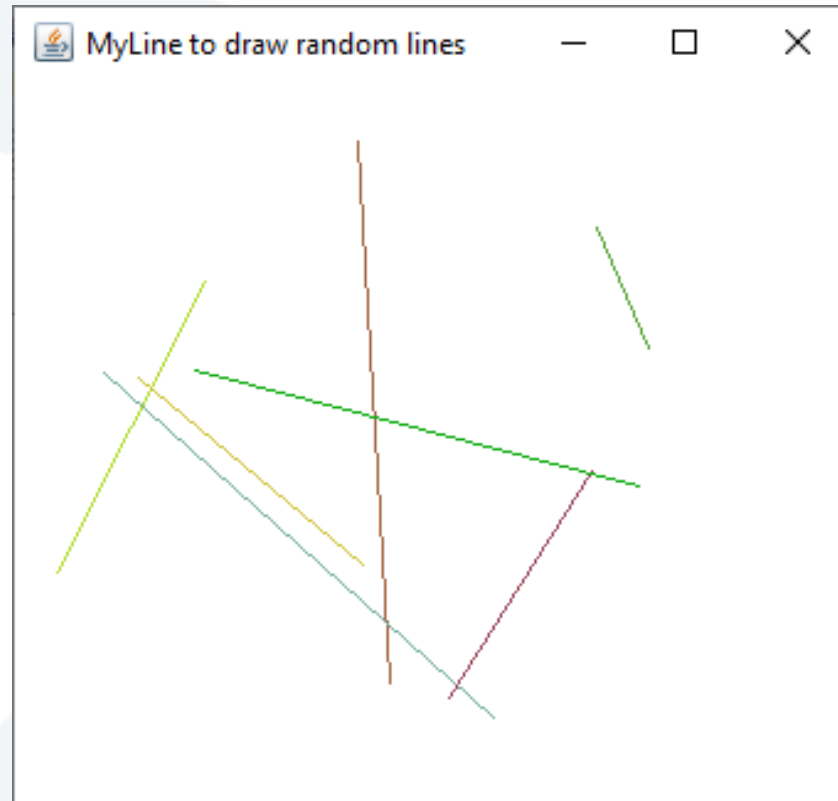
## الصنف DrawTest2.java



Creating a JFrame to display a DrawPanel

```
// Fig. 8.20: TestDraw.java
// Creating a JFrame to display a DrawPanel.
import javax.swing.JFrame;
public class DrawTest2
{
    public static void main( String[] args )
    {
        DrawPanel2 panel = new DrawPanel2();
        JFrame application = new JFrame("MyLine to draw random lines");
        application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        application.add( panel );
        application.setSize( 300, 300 );
        application.setVisible( true );
    } // end main
} // end class TestDraw
```

uses class MyLine to  
draw random lines 3



رسم مستقسمات عشوائية اللون والطول والعدد لايقبل عن 5 ولايزيد عن 10

Demonstrates the use  
of labels



## 9.12 GUI & Graphics

✓ **Labels** هي طريقة مناسبة لتحديد مكونات واجهة المستخدم الرسومية على الشاشة وإبقاء المستخدم على اطلاع بالحالة الحالية للبرنامج.

✓ يمكن لـ **JLabel** من الحزمة `javax.swing` عرض نص أو صورة أو كليهما.

✓ يوضح المثال في الشكل 9.13 العديد من ميزات `JLabel`، بما في ذلك تسمية النص العادي وتسمية الصورة والتسمية مع كل من النص والصورة.

# JLabel with text and with images



## LabelDemo 1

```
package ch12GUI;
//Fig 9.13: LabelDemo.java
//Demonstrates the use of labels.
import java.awt.BorderLayout; //contains constants that specify where we can place GUI components
import javax.swing.ImageIcon; //represents an image that can be displayed on a JLabel.
import javax.swing.JLabel; // to create an object can display text, an image or both.
import javax.swing.JFrame; // represents the window that will contain all the labels.
public class LabelDemo
{public static void main( String[] args )
{ // Create a label with plain text
JLabel northLabel = new JLabel( "North" );
// create an icon from an image so we can put it on a JLabel
// ImageIcon can load images in GIF, JPEG and PNG image formats.
ImageIcon labelIcon = new ImageIcon( "GUItip.gif" );
// create a label with an Icon instead of text
JLabel centerLabel = new JLabel( labelIcon );
// create another label with an Icon
JLabel southLabel = new JLabel( labelIcon );
```

LabelDemo **الصف**

# JLabel with text and with images



## LabelDemo 2

```
// set the label to display text (as well as an icon)
southLabel.setText( "South" );
// create a frame to hold the labels
JFrame application = new JFrame("LabelDemo");
application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

// add the labels to the frame; the second argument specifies
// where on the frame to add the label
application.add( northLabel, BorderLayout.NORTH );
application.add( centerLabel, BorderLayout.CENTER );
application.add( southLabel, BorderLayout.SOUTH );

application.setSize( 300, 300 ); // set the size of the frame
application.setVisible( true ); // show the frame
} // end main
} // end class LabelDemo
```

LabelDemo **الصف**



# Demonstrates the use of labels

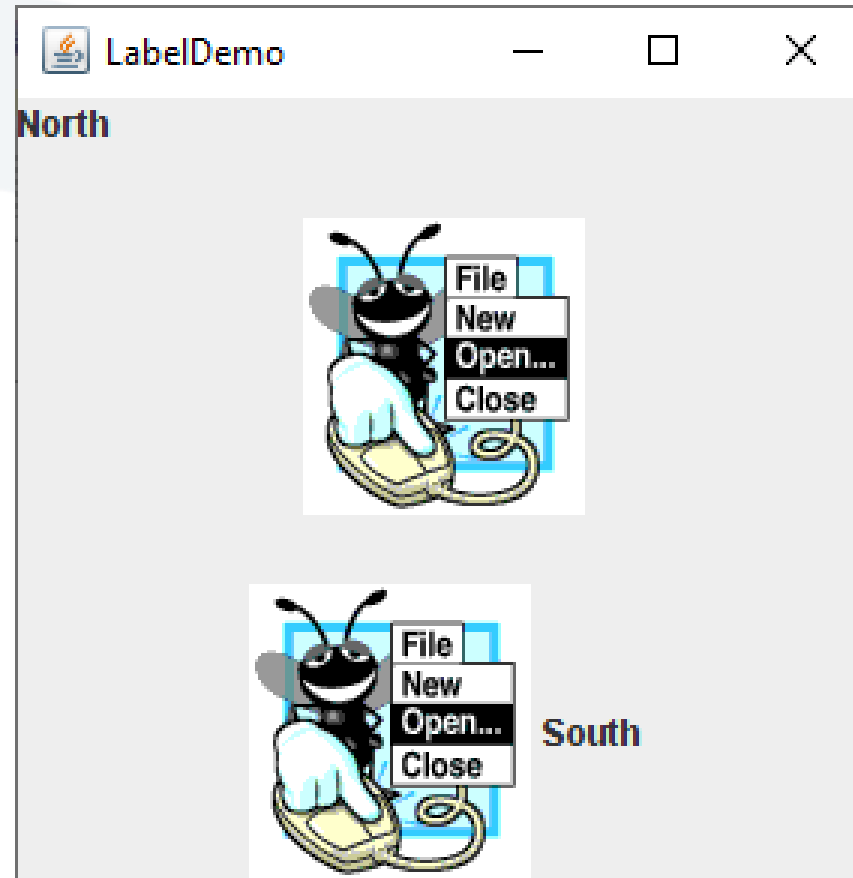


Fig. 9.13 | JLabel with text and with images

الصنف `TestDraw.java`

## JLabel with text and with images.



# 9.12 GUI & Graphics

- ✓ يتلقى باني الصنف ImageIcon سلسلة تحدد المسار إلى الصورة.
- ✓ في حال تلقى اسم الملف فقط، تفترض Java أنه موجود في نفس الدليل مثل الصنف .LabelDemo.
- ✓ يمكن للصنف ImageIcon تحميل الصور بتنسيقات صور GIF و JPEG و PNG.
- ✓ تستدعي الطريقة setText لإضافة تسمية أو لتغيير النص الذي تعرضه التسمية. يمكن استدعاء الأسلوب setText على أي JLabel لتغيير نصه.
- ✓ يعرض هذا JLabel كلاً من الرمز والنص.
- ✓ من خلال استدعاء نسخة محملة بشكل زائد من طريقة إضافة تأخذ بارامترين. الأول هي المكون الذي نريد إرفاقه ، والثاني هو المنطقة التي يجب وضعها فيه.
- ✓ يحتوي كل إطار JFrame على تخطيط مرتبط يساعد JFrame في وضع مكونات واجهة المستخدم الرسومية المرفقة به.
- ✓ يُعرف التخطيط الافتراضي لـ JFrame باسم BorderLayout وله خمس مناطق  
Five regions - North (top), South (bottom), East (right side), West (left side) and Center.

showInputDialog  
showMessageDialog





- ✓ // display result in a JOptionPane message dialog

```
JOptionPane.showMessageDialog(null, "The sum is " + sum, "Sum of Two Integers",  
JOptionPane.PLAIN_MESSAGE);
```

- ✓ قد تتضمن طريقة العرض الساكنه showMessageDialog من الصنف JOptionPane أربع متغيرات.
- ✓ الأول يحدد مكان ظهور مربع الحوار إذا غاب أو كان null ستكون في الوسط.
- ✓ الثاني الرسالة التي يجب عرضها - في هذه الحالة ، نتيجة تسلسل السلسلة "المجموع هو" وقيمة المجموع.
- ✓ الثالث السلسلة التي يجب أن تظهر في شريط العنوان أعلى مربع الحوار وهنا **"Sum of Two Integers"**.
- ✓ الرابع الايقونه التي ستظهر على يسار مربع الحوار وإن كانت PLAIN\_MESSAGE لن تظهر ايقونه.
- ✓ توجد أربع ايقونات هي QUESTION\_MESSAGE ، INFORMATION\_MESSAGE ، WARNING\_MESSAGE ، ERROR\_MESSAGE والحالة PLAIN\_MESSAGE بدون ايقونه.

# showMessageDialog

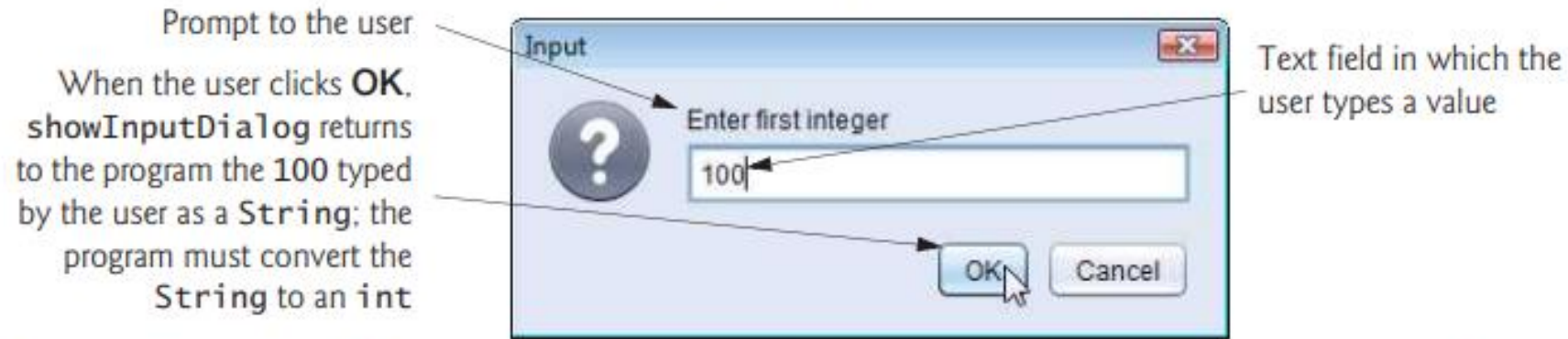
## JOptionPane static constants for message dialogs

Message dialog type	Icon	Description
ERROR_MESSAGE		Indicates an error.
INFORMATION_MESSAGE		Indicates an informational message.
WARNING_MESSAGE		Warns of a potential problem.
QUESTION_MESSAGE		Poses a question. This dialog normally requires a response, such as clicking a Yes or a No button.
PLAIN_MESSAGE	no icon	A dialog that contains a message, but no icon

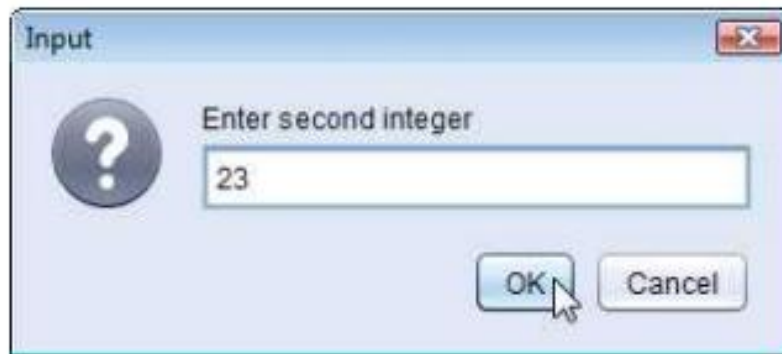
**Fig. 12.3** | JOptionPane static constants for message dialogs.

# program that uses JOptionPane for input and output

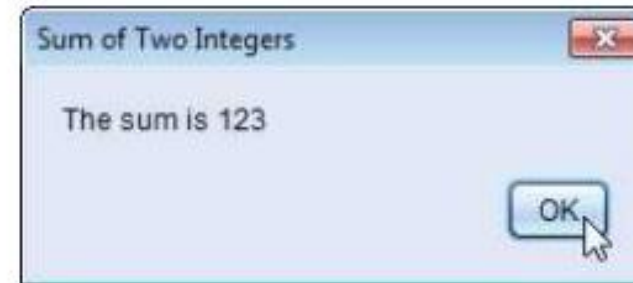
(a) Input dialog displayed by lines 10–11



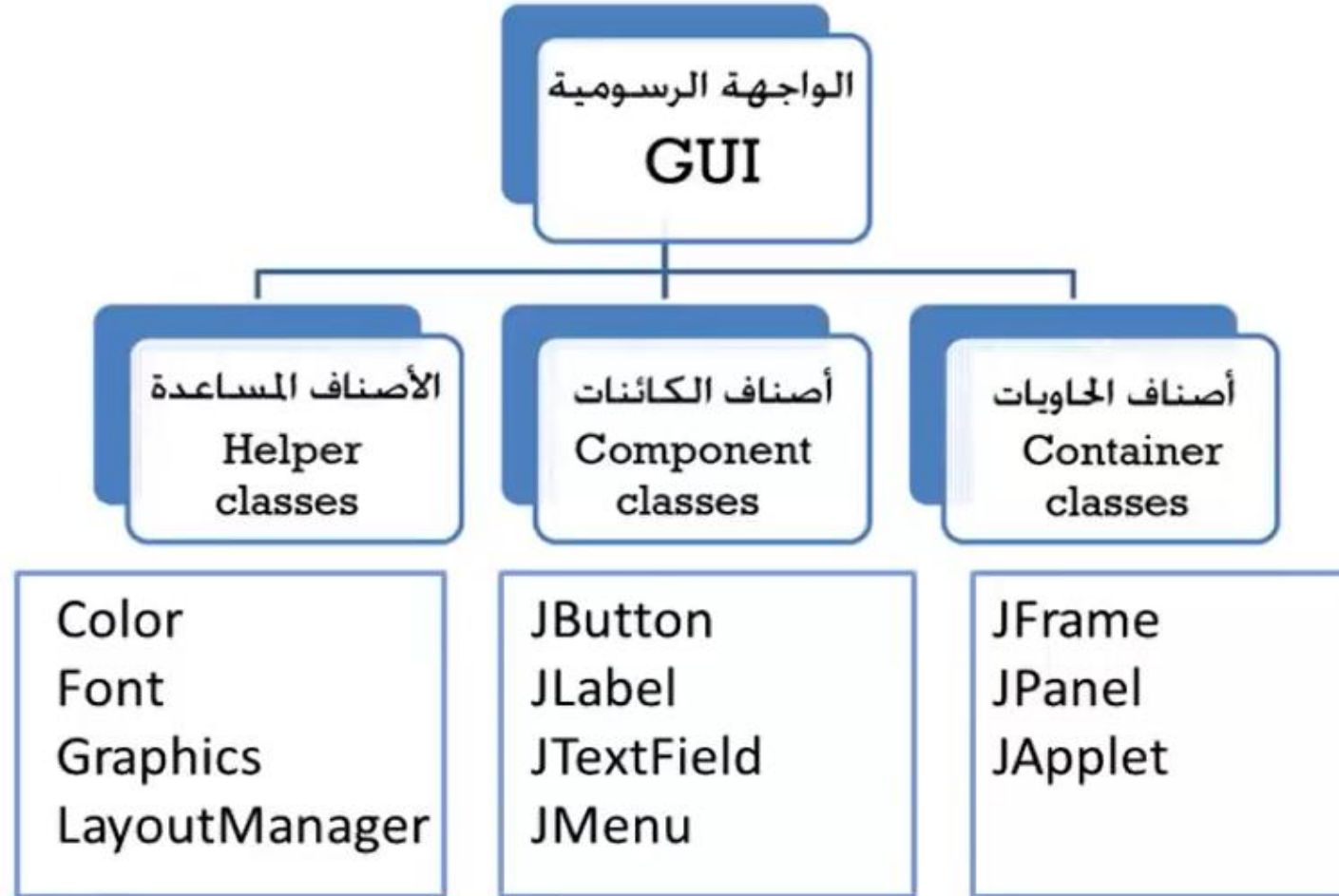
(b) Input dialog displayed by lines 12–13



(c) Message dialog displayed by lines 22–23—When the user clicks **OK**, the message dialog is dismissed (removed from the screen)



**Fig. 12.2** | Addition program that uses `JOptionPane` for input and output. (Part 2 of 2.)



Component	Description
JLabel	Displays uneditable text and/or icons.
JTextField	Typically receives input from the user.
JButton	Triggers an event when clicked with the mouse.
JCheckBox	Specifies an option that can be selected or not selected.
JComboBox	A drop-down list of items from which the user can make a selection.
JList	A list of items from which the user can make a selection by clicking on any one of them. Multiple elements can be selected.
JPanel	An area in which components can be placed and organized.

**Fig. 12.4** | Some basic Swing GUI components.



- ✓ JFrame هو صنف فرعي غير مباشرة من الصنف `java.awt.Window` الذي يوفر السمات والسلوكيات الأساسية للنافذة مثل شريط العنوان في الأعلى، وأزرار لتصغير النافذة وتكبيرها وإغلاقها.
- ✓ تتكون GUI النموذجية من العديد من المكونات. غالبًا ما يقدم مصمم GUI نصًا يوضح الغرض من كل منها.
- ✓ نص التسمية يتم إنشاؤه باستخدام `JLabel` - صنف فرعية من `JComponent`.
- ✓ نادرًا ما تغير التطبيقات محتويات التسميات بعد إنشائها.
- ✓ لكل عنصر GUI العديد من الميزات أكثر مما يمكننا تغطيته في الأمثلة الخاصة بنا.
- ✓ لمعرفة التفاصيل الكاملة، قم بزيارة صفحته في التوثيق عبر الإنترنت.
- ✓ [docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html](https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html).

## 12.5 Displaying Text and Images Using Labels

```
package ch12GUI;
//Fig. 12.6: LabelFrame.java      JLabels with text and icons.
import java.awt.FlowLayout; // specifies how components are arranged
import javax.swing.JFrame; // provides basic window features
import javax.swing.JLabel; // displays text and/or images
import javax.swing.SwingConstants; // common constants used with Swing
import javax.swing.Icon; // interface used to manipulate images
import javax.swing.ImageIcon; // loads images
public class LabelFrame extends JFrame
{private final JLabel label1; // JLabel with just text
private final JLabel label2; // JLabel constructed with text and icon
private final JLabel label3; // JLabel with added text and icon
// LabelFrame constructor adds JLabels to JFrame
public LabelFrame()
{    super("Testing JLabel");
    setLayout(new FlowLayout()); // set frame layout
```

## 12.5 Displaying Text and Images Using Labels

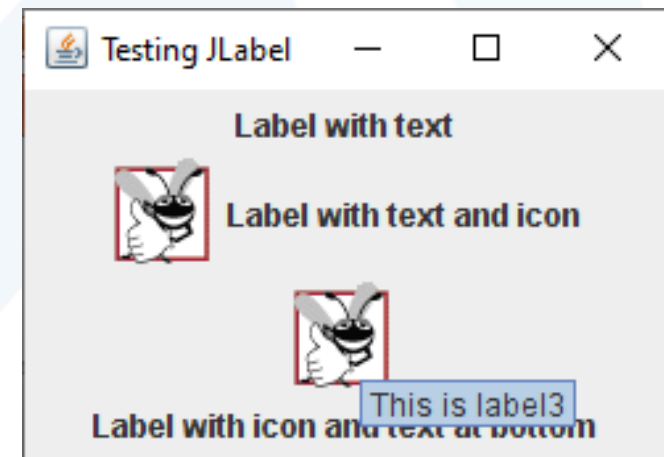
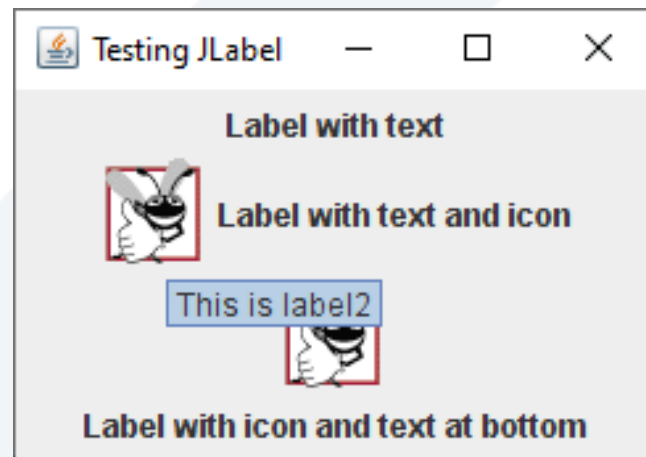
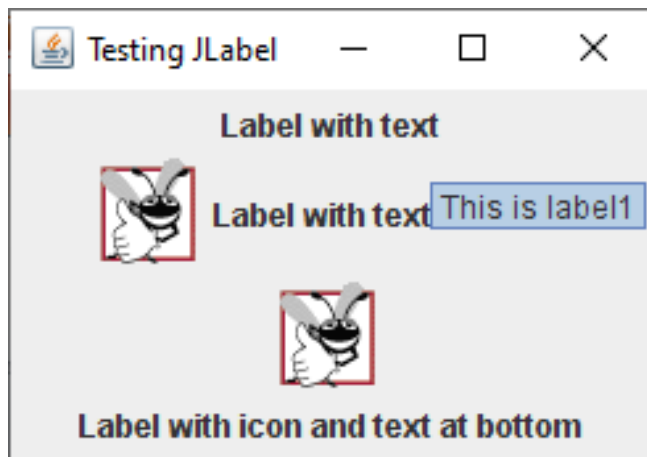
```
// JLabel constructor with a string argument
label1 = new JLabel("Label with text");
label1.setToolTipText("This is label1");
add(label1); // add label1 to JFrame
// JLabel constructor with string, Icon and alignment arguments
Icon bug = new ImageIcon(getClass().getResource("bug1.png"));
label2 = new JLabel("Label with text and icon", bug, SwingConstants.LEFT);
label2.setToolTipText("This is label2");
add(label2); // add label2 to JFrame
label3 = new JLabel(); // JLabel constructor no arguments
label3.setText("Label with icon and text at bottom");
label3.setIcon(bug); // add icon to JLabel
label3.setHorizontalTextPosition(SwingConstants.CENTER);
label3.setVerticalTextPosition(SwingConstants.BOTTOM);
label3.setToolTipText("This is label3");
add(label3); // add label3 to JFrame
}
} // end class LabelFrame
```

## 12.5 Displaying Text and Images Using Labels

```

package ch12GUI;
//Fig. 12.7: LabelTest.java    Testing LabelFrame.
import javax.swing.JFrame;
public class LabelTest
{public static void main(String[] args)
{  LabelFrame labelFrame = new LabelFrame();
  labelFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  labelFrame.setSize(260, 180);
  labelFrame.setVisible(true); }
} // end class LabelTest

```



- تحميل مصدر الصورة في السطر، يستدعي التعبير (`getClass().getResource("bug1.png")`) المنهج (`getClass()`) (الموروثة بشكل غير مباشر من الصنف `Object`) لاسترداد مرجع إلى كائن `Class` الذي يمثل تعريف صنف `LabelFrame`. يتم بعد ذلك استخدام هذا المرجع لاستدعاء `getResource` التابع لمنهج `Class`، والذي يُرجع موقع الصورة كعنوان `URL`. يستخدم منشئ `ImageIcon` عنوان `URL` لتحديد موقع الصورة، ثم يقوم بتحميلها.
- تقوم `JVM` بتحميل تصريحات الأصناف في الذاكرة باستخدام مُحمل الصنف. يعرف مُحمل الصنف مكان وجود كل صنف يقوم بتحميلها على القرص. تستخدم طريقة `getResource` مُحمل صنف كائن الصنف لتحديد موقع المورد، مثل ملف صورة. في هذا المثال، يتم تخزين ملف الصورة في نفس الموقع مثل ملف `LabelFrame.class`. تعمل التقنيات الموضحة هنا على تمكين التطبيق من تحميل ملفات الصور من المواقع المرتبطة بموقع ملف الصنف.
- تحميل واجهة موارد الصور (`SwingConstants` (package `javax.swing`) تعلن عن مجموعة من الثوابت الصحيحة الشائعة (مثل `SwingConstants.LEFT` و `SwingConstants.CENTER` و `SwingConstants.RIGHT`) التي يتم استخدامها مع العديد من مكونات `Swing`. افتراضياً، يظهر النص على يمين الصورة عندما تحتوي التسمية على نص وصورة معاً. يمكن ضبط المحاذاة الأفقية والرأسية لـ `Label` باستخدام الأساليب `setVerticalAlignment` و `setHorizontalAlignment`، على التوالي. يحدد السطر نص تلميح الأداة للتسمية 2، ويضيف السطر التسمية 2 إلى `JFrame`.

يتم قراءة البيانات من لوحة المفاتيح بصيغة أسكي إذا كانت القراءة بايت واحد وبصيغة سلسلة إذا كانت القراءة بعدة بايتات وعلى المبرمج تحويل من سلسلة نصية إلى أرقام عددية `int` باستخدام الدوال الخاصة بالسلاسل للتعامل معها كأعداد

```
package Rectangle7_10_2023;
import java.io.*;
public class Excepy {
    public static void main(String args[]) throws IOException
        { // American Standard Code for Information Interchange
    int b; b=System.in.read();
//for(int i='a'; i<='z';i++)
System.out.println("ASCII "+b);
    }
}
```

# انتهت محاضرة الأسبوع السادس