



كلية الهندسة المعلوماتية

برمجة 3

Java Programming

أ.د. علي عمران سليمان

محاضرات الأسبوع الثامن

GUI 2

الفصل الأول 2024-2025

Outline



- 12.1 Introduction
- 12.2 Java's Nimbus Look-and-Feel
- 12.3 Simple GUI-Based Input/Output with JOptionPane
- 12.4 Overview of Swing Components
- 12.5 Displaying Text and Images in a Window
- 12.6 Text Fields and an Introduction to Event Handling with Nested Classes
- 12.7 Common GUI Event Types and Listener Interfaces
- 12.8 How Event Handling Works
- 12.9 JButton
- 12.10 Buttons That Maintain State
- 12.10.1 JCheckBox
- 12.10.2 JRadioButton

12.11 JComboBox; Using an Anonymous Inner Class for Event Handling

12.12 JList

12.13 Multiple-Selection Lists

12.14 Mouse Event Handling

12.15 Adapter Classes

12.16 JPanel Subclass for Drawing with the Mouse

12.17 Key Event Handling

12.18 Introduction to Layout Managers

12.18.1 FlowLayout

12.18.2 BorderLayout

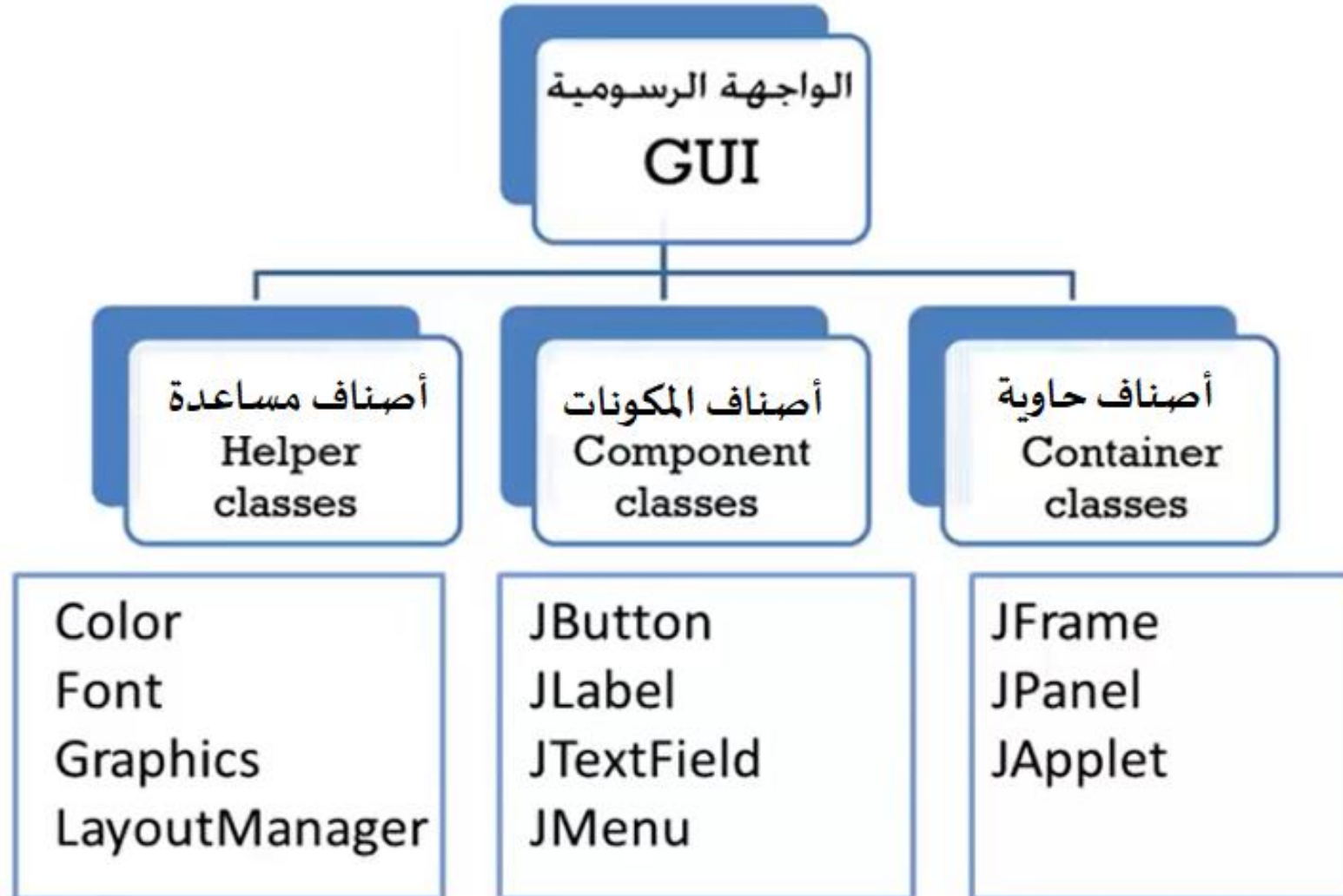
12.18.3 GridLayout

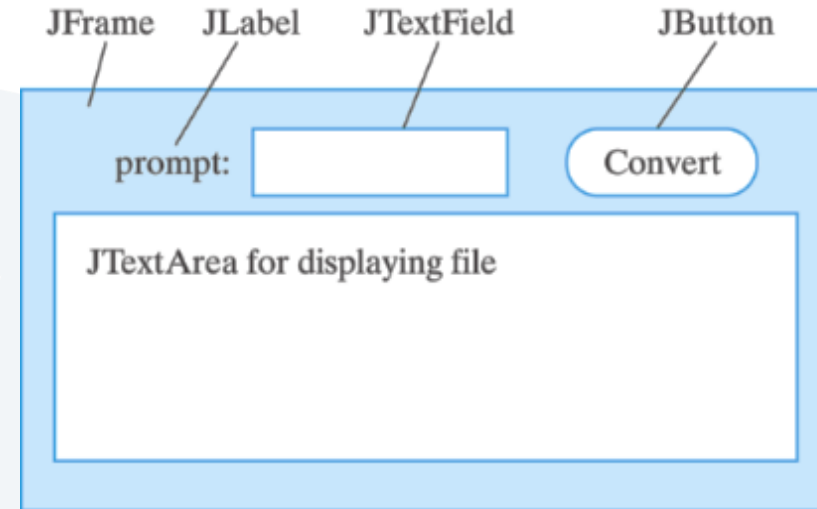
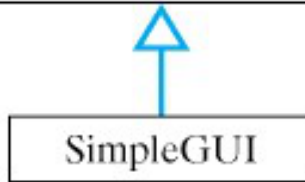
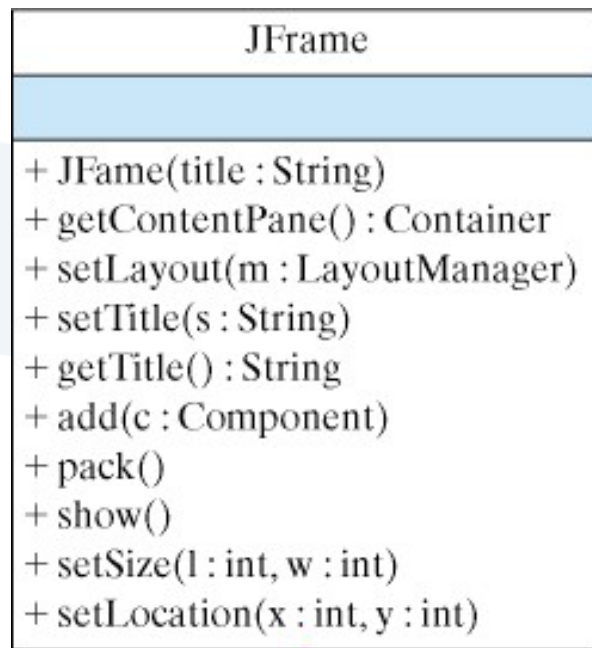
12.19 Using Panels to Manage More Complex Layouts

12.20 JTextArea

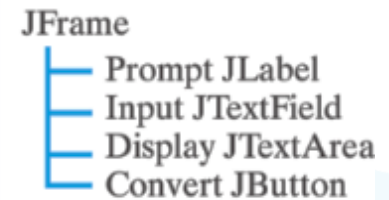
References - Deitel & Deitel, Java How to Program, Pearson; 10th Ed(2015)

- د.علي سليمان، بني معطيات بلغة JAVA، جامعة تشرين 2013-2014





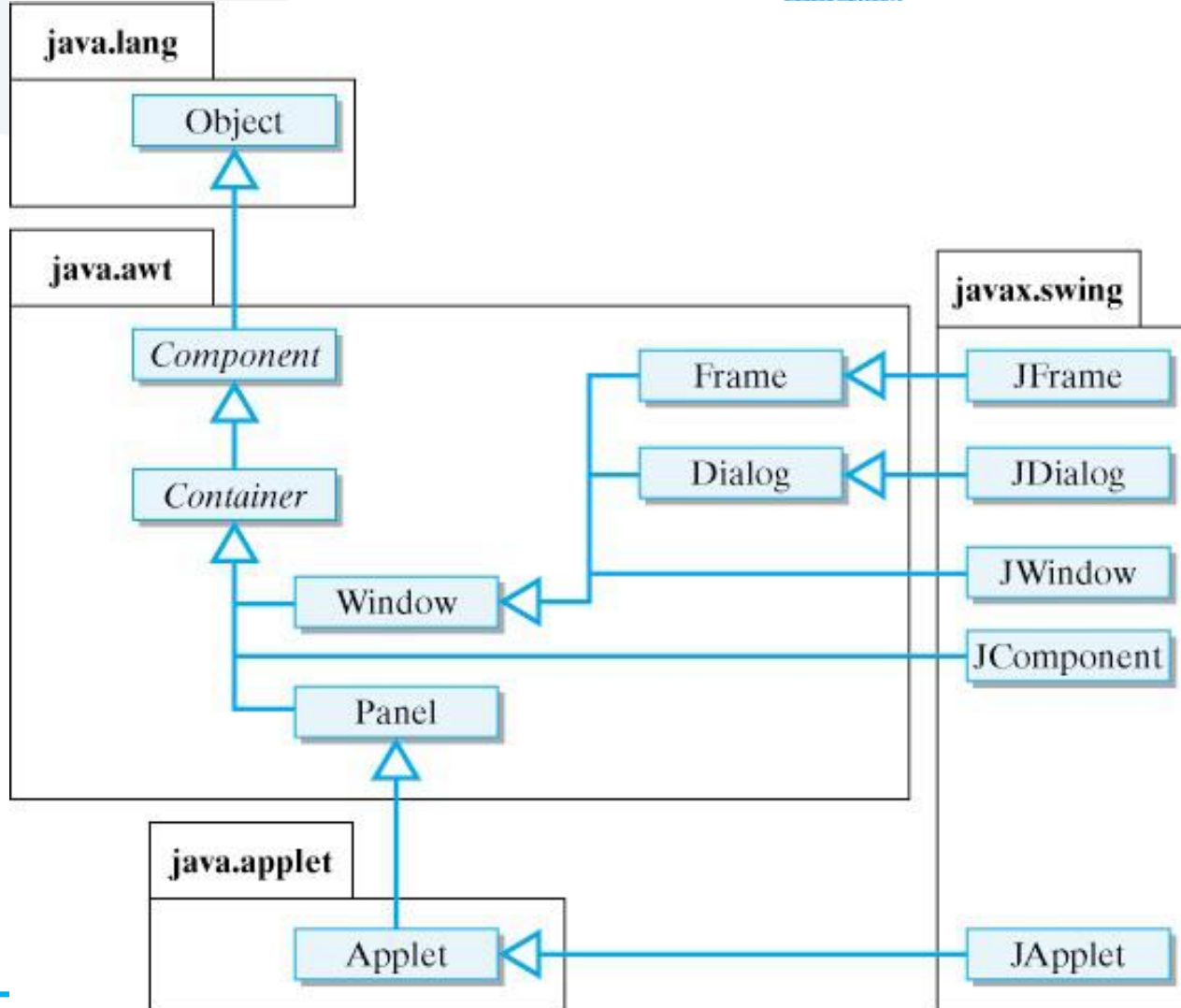
Containment Hierarchy



يوضح UML السابق أساليب JFrame الموروثة من Frame من Window من Container. هنا سيتم بناء صنف ليكن SimpleGUI يرث JFrame. حيث JFrame() هو الباني، و getContentPane() لتعيد لنا حاوي نضع عليه المكونات، و setSize() و setLocation() لتعيين حجم SimpleGUI وموقعه. يمكن أن نستخدم أسلوب setTitle() لتعيين عنوانه بدل من اسناده من الباني، ونستخدم أسلوب setVisible() لجعله يظهر على وحدة التحكم، pack() ويعاد ضبط حجم إطار JFrame إلى أصغر حجم ممكن.

هو تسلسل هرمي للفئات يوضح العلاقات بين بعض فئات Swing و AWT ذات المستوى الأعلى. على سبيل المثال، فئة javax.swing.JFrame، التي تمثل نافذة ذات مستوى أعلى، هي فئة فرعية من java.awt.Frame، و javax.swing.JPanel هي فئة فرعية من java.awt.Panel. يمكننا أن نرى من هذا الشكل أن JFrameisFrame و WindowisWindow و JFrameisWindow و WindowisContainer. إنشاء علاقات الفئة الفرعية/الفئة العليا هذه في تعريفات الفئات الخاصة بها باستخدام الوراثة extends على النحو التالي:

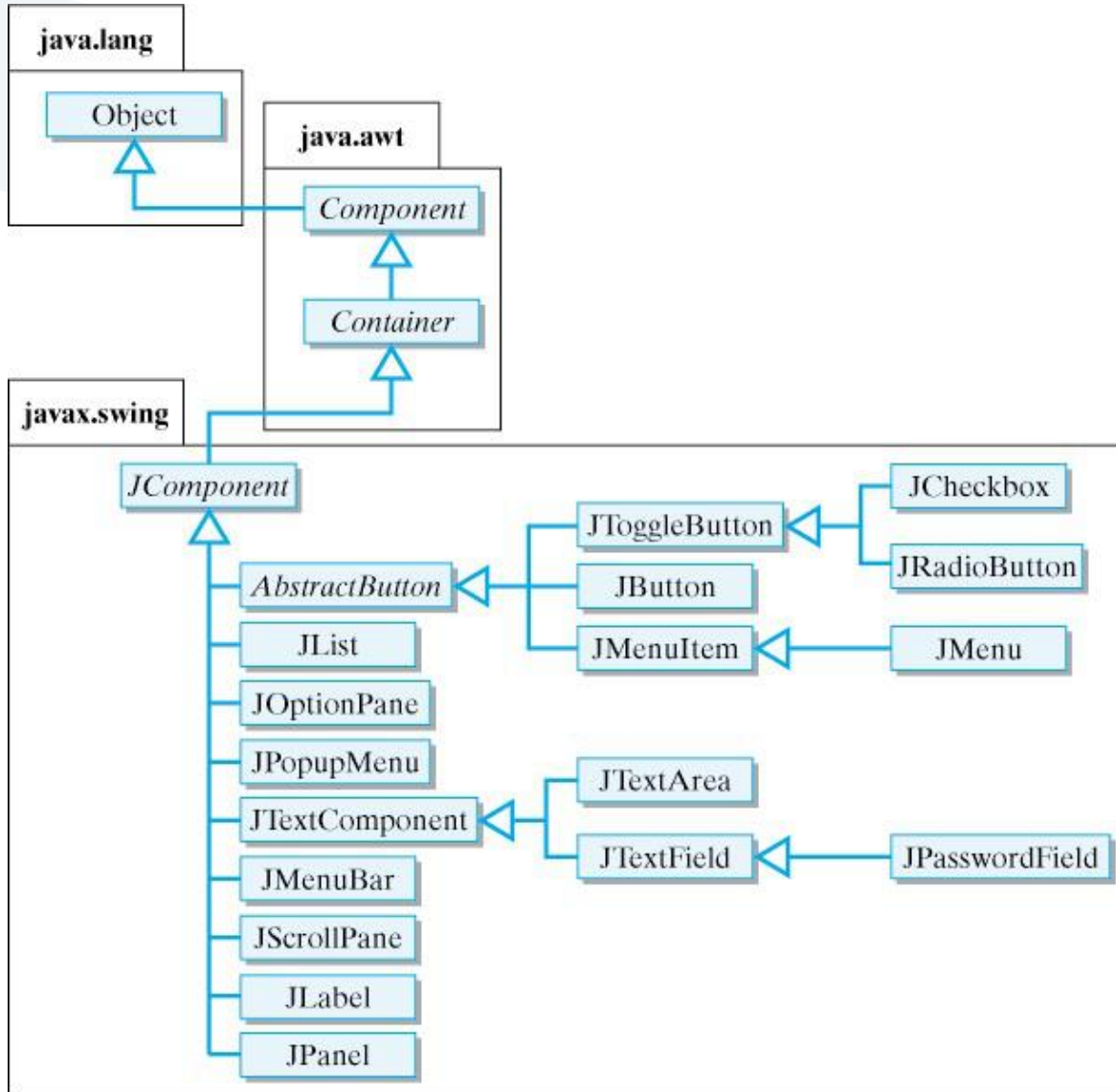
تذكرة



التسلسل الهرمي لبعض فئات
 java.lang الحاوية على المكتبة
 java.awt والوارث لها :
 javax.swing و java.applet
 ونجد معظم الأصناف المدرجة.

Swing components

JFrame



التسلسل الهرمي لبعض فئات `java.lang`
الحاوية على المكتبة `java.awt` والوارث لها:
`JComponent` و `javax.swing` تحديداً أصناف

- شجرة `AbstractButton`
- شجرة `JTextComponent`

نستخدم ثلاث حزم لبرمجة واجهة المستخدم وهي (1996) java.awt و (1997) swing و (2014) javaFx

- قدمت لغة الجافا حزمة تسمى (Abstract Window Toolkit) awt مخصصة لبرمجة الواجهات المستخدم لأول مرة وكانت كل الأصناف موجودة فيها والتي تقوم بضبط اعدادات البرنامج تلقائياً حسب المنصة التي يتم تشغيل البرنامج عليها، وهي تنفع في بناء واجهات مستخدم بسيطة، وخصصت بشكل أساسي لأجهزة الكمبيوتر، مع دعم محدود جداً للويب/الجوال.
- تم استبدال حزمة awt بحزمة أكثر تميز وكفاءة هي حزمة swing وعرفت light components weight أي المكونات الخفيفة لأنها تعتمد على انشاء الكائنات من دون الإعتماد على منصة التشغيل على خلاف awt والتي تعرف heavy components weight المكونات الثقيلة لأنها تعتمد وتتعامل مع منصة التشغيل، swing خصصت لبناء واجهات مستخدم محترفة ومتميزة.
- ومن أجل التفريق بين اصناف حزمة awt واصناف swing يتم اضافة السابقة ل قبل اسم كل صنف من اصناف فئة swing .
- طريقة تصميم وترتيب ال GUI Application Program Interface تعتبر من افضل الأمثلة على استخدام الوراثة والأصناف والواجهات Interfaces.
- ظهرت مؤخراً حزمة جديدة تعتبر الأفضل من حيث التميز والكفاءة إنما لاتزال قيد التطوير javaFx موجه لمنصات (سطح مكتب، ويب، جوال) وتتميز بمجموعة حديثة وواسعة من مكونات واجهة المستخدم ولاتزال تحت الدعم والتطوير، على خلاف swing حيث كانت مكتبة غنية بالمكونات المتقدمة ومركّزه على سطح المكتب، مع دعم محدود للويب/الجوال.

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

1- إنشاء كائن من JFream وفق `new JFrame("First");` حيث First هي عنوان الإطار ويمكن أن تترك فارغة ونستخدم `appli.setTitel("First");`

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

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

4- جعل الإطار مرئي نستخدم 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، ويمكن استخدام معالجة الحدث النقر على أيقونة الغلق من أجل إنهاء البرنامج إضافة على إغلاق النافذة أو إنهاء البرنامج باستخدام `System.exit(0)` والتي يجب أن تكون بنهاية main ولكن تخلق مشكلة جديدة وهي ظهور النافذة للحظة وجيزة وينتهي فوراً والرغبة هي إنهاء البرنامج عندما يرغب المستخدم مثل ضغط المستخدم على أيقونة الغلق في شريط العنوان وهنا نجد اسهل وسيلة استخدام الطريقة : `appli.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);` ولها عدة حالات مثل `DO_NOTHING_ON_CLOSE` وهنا لن يعمل شيء عند النقر على X.

GUI and Graphics

Creating Simple Drawings

Java's Coordinate System

• class Graphics من الحزمة (java.awt)، والتي توفر طرقًا مختلفة لرسم النص والأشكال على الشاشة.

• الصنف JPanel من الحزمة (javax.swing)، والتي توفر مساحة يمكن الرسم عليها.

```
public class DrawPanel extends JPanel
```

• كما هو معروف **extends** كلمة أساسية للإشارة إلى أن الصنف DrawPanel هو نوع محسن من JPanel أي وراث له.

• هنا الصنف الجديد DrawPanel يبدأ بالأعضاء الحاليين (البيانات والأساليب) من فئة JPanel.

• كل لوحة **JPanel** بما في ذلك DrawPanel، لديها طريقة **paintComponent()**.

• ينادي النظام تلقائيًا في كل مرة يحتاج فيها إلى عرض DrawPanel الطريقة **paintComponent()** ويجب التصريح عنها **public void paintComponent(Graphics g)**، خلاف ذلك، لن يسمح النظام بمناداتها ويجب أن تكون العبارة الأولى فيها عندما تكتب (أو تحمل تحميلًا زائدًا) هي **super.paintComponent(g);**

• يتم استدعاء هذه الطريقة في الحالات الآتية: عندما يتم عرض **JPanel** لأول مرة على الشاشة، وعندما يتم تغطيتها hidden أو الكشف عنها بواسطة نافذة أخرى على الشاشة، وعندما يتم تغيير حجم النافذة التي تظهر فيها.

Demonstrates the use
of labels



9.12 GUI & Graphics

✓ **Labels** هي آلية مناسبة لتعريف 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 الصنف

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");

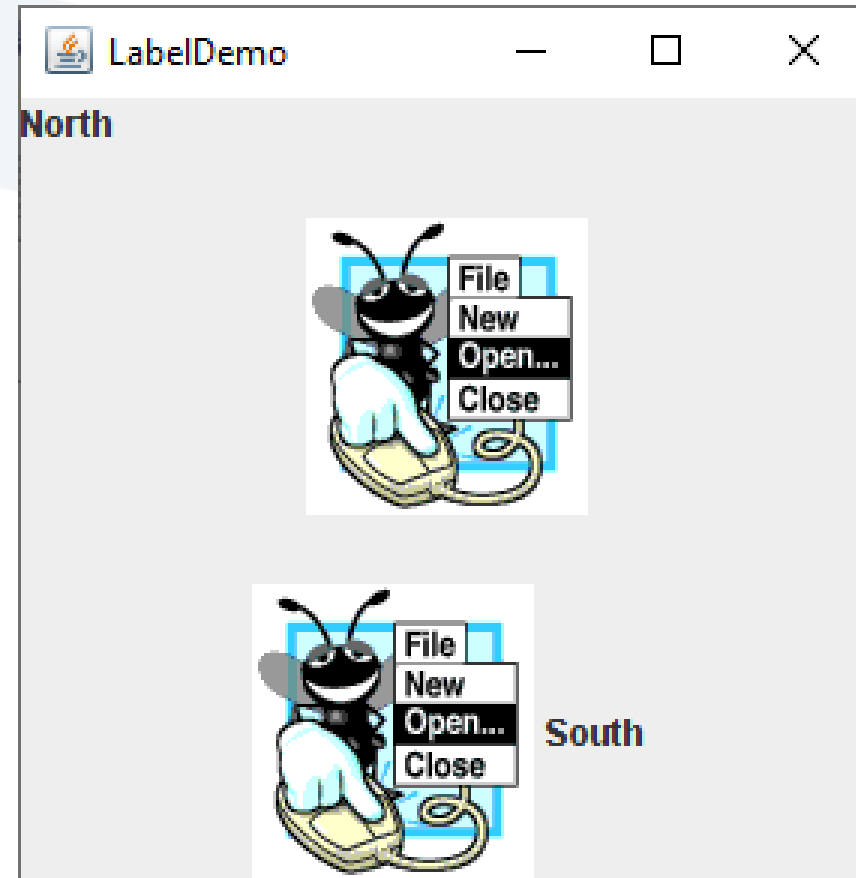
application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

// 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



[Back](#)

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



12.3 Simple GUI-Based Input / Output with JOptionPane

✓ // 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.

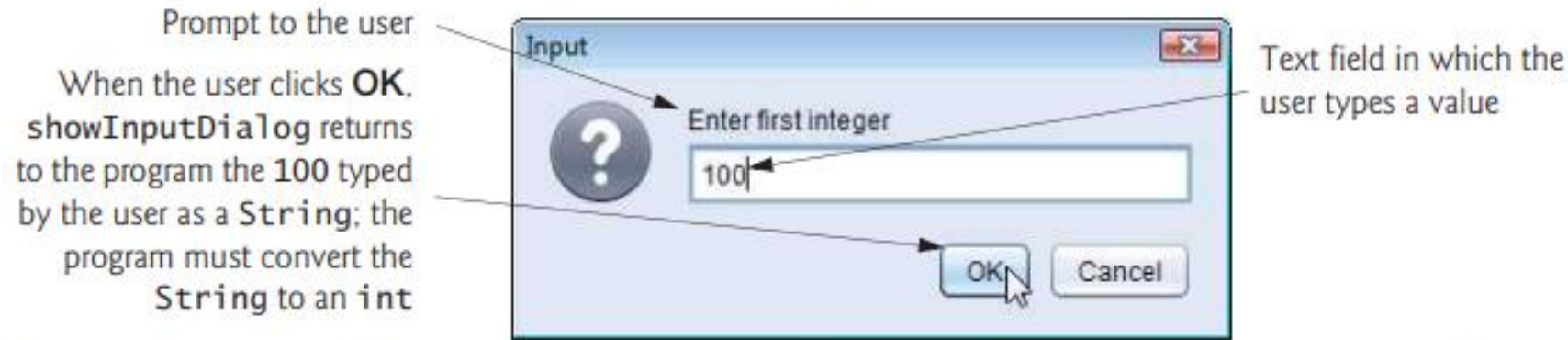
✓ توجد أربع ايقونات هي QUESTION_MESSAGE ، INFORMATION_MESSAGE ، WARNING_MESSAGE ، ERROR_MESSAGE والحالة PLAIN_MESSAGE بدون ايقونه.

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

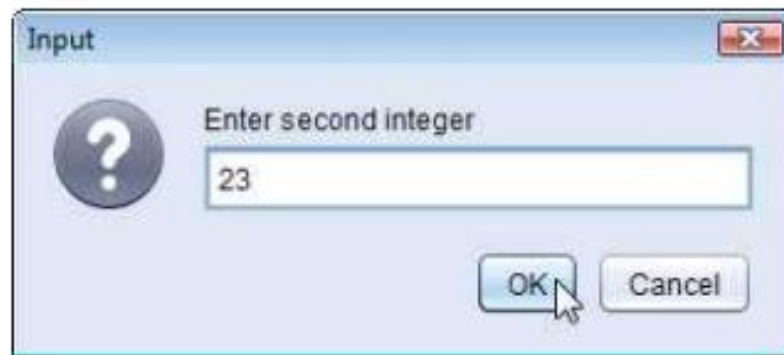
Fig. 12.3 | JOptionPane static constants for message dialogs.

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)

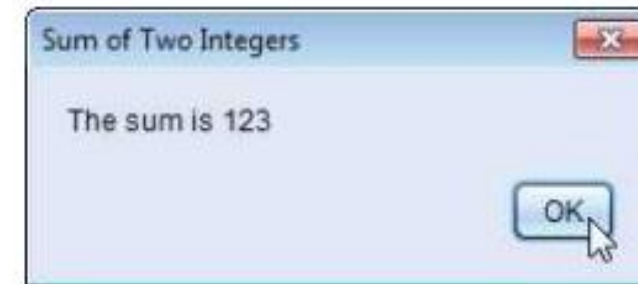


Fig. 12.2 | Addition program that uses `JOptionPane` for input and output. (Part 2 of 2.)

Some basic Swing GUI components

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

Fig. 12.4 | Some basic Swing GUI components.

12.5 Displaying Text and Images Using Labels

- ✓ 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.

12.5 Displaying Text and Images Using Labels

```
package ch12GUI;

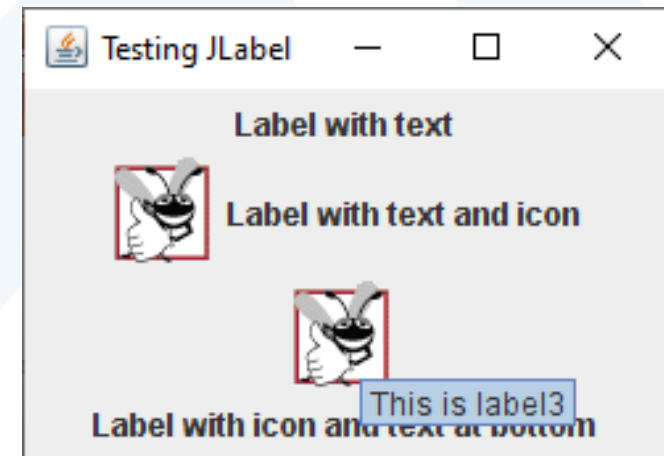
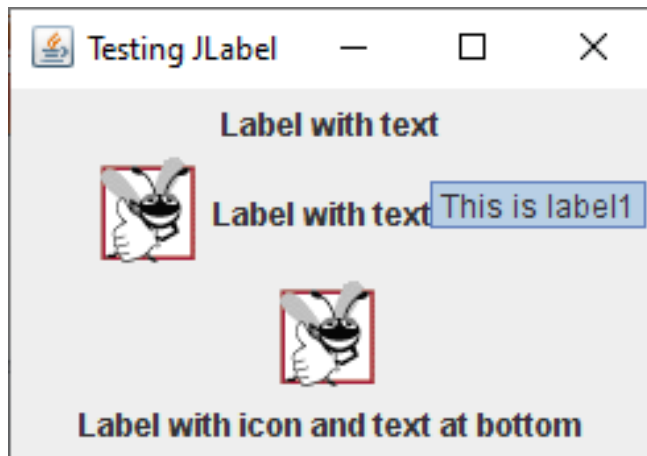
//Fig. 12.6: LabelFrame.java      JLabels with text and icons.
import java.awt.FlowLayout; // specifies how components are arranged
import javax.swing.JFrame;   // provides basic window features
import javax.swing.JLabel;   // displays text and/or images
import javax.swing.SwingConstants; // common constants 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 constructed 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")` الطريقة `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 لطريقة `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` تعمل التقنيات الموضحة هنا على تمكين التطبيق من تحميل ملفات الصور من المواقع المرتبطة بموقع ملف الصنف.
- تحميل واجهة موارد الصور (`package javax.swing`) `SwingConstants` تعلن عن مجموعة من الثوابت الصحيحة الشائعة (مثل `SwingConstants.LEFT` و `SwingConstants.CENTER` و `SwingConstants.RIGHT`) التي يتم استخدامها مع العديد من مكونات `Swing`. افتراضيًا، يظهر النص على يمين الصورة عندما تحتوي التسمية على نص وصورة معًا. يمكن ضبط المحاذاة الأفقية والرأسية لـ `Label` باستخدام الأساليب `setHorizontalAlignment` و `setVerticalAlignment`، 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

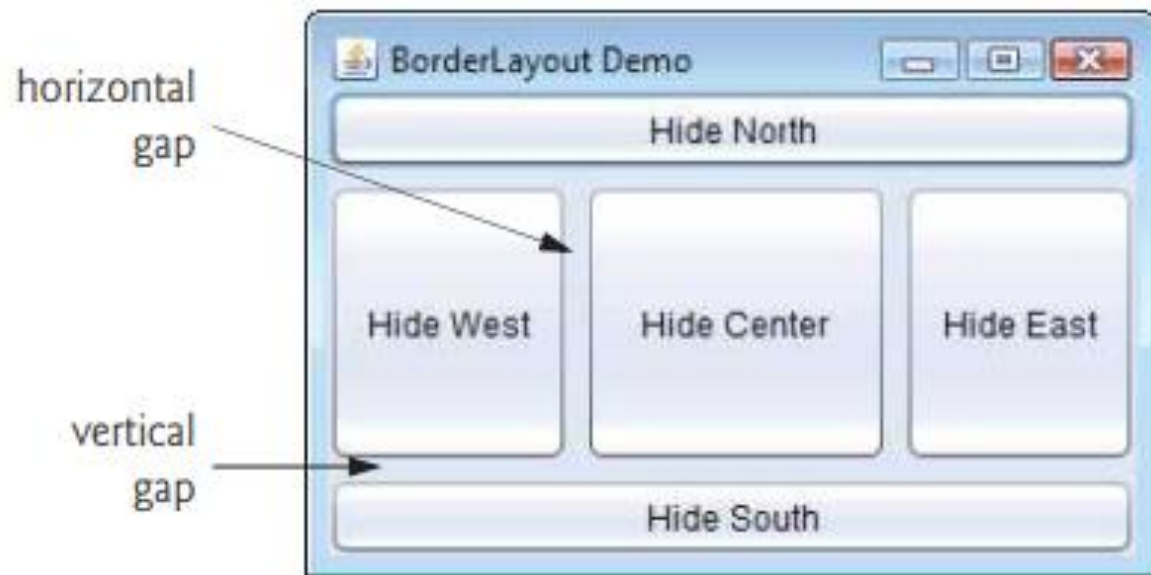
```
package Rectangle7_10_2023;  
import java.io.*;  
public class Excepy {  
    public static void main(String args[]) throws IOException  
    { // American Standard Code for Information Interchange  
        int b; b=System.in.read();  
        //for(int i='a'; i<='z';i++)  
        System.out.println("ASCII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

Layout Managers 1

- **Absolute positioning**
 - By setting a Container's layout to null, `setLayout(null)`
 - `setBounds(x, y, w, h)`
- **Layout managers**
 - Available for arranging GUI components
 - Processes layout details
 - Programmer can concentrate on basic “look and feel”
 - Interface `LayoutManager`
- **Visual programming in an IDE**

Layout Managers 2

Layout manager	Description
FlowLayout	Default for <code>java.awt.Applet</code> , <code>java.awt.Panel</code> and <code>javax.swing.JPanel</code> . Places components sequentially (left to right) in the order they were added. It is also possible to specify the order of the components by using the <code>Container</code> method <code>add</code> , which takes a <code>Component</code> and an integer index position as arguments.
BorderLayout	Default for the content panes of <code>JFrames</code> (and other windows) and <code>JApplets</code> . Arranges the components into five areas: NORTH, SOUTH, EAST, WEST and CENTER.
GridLayout	Arranges the components into rows and columns.



Layout Managers

- **FlowLayout**
 - Most basic layout manager
 - GUI components placed in container from left to right
- **BorderLayout**
 - Arranges components into five regions
 - NORTH (top of container) SOUTH (bottom of container)
 - EAST (right side) WEST (left side)
 - CENTER (center of container)
- **GridLayout**
 - Divides container into grid of specified row and columns
 - Components are added starting at top-left cell
 - Proceed left-to-right until row is full

انتهت محاضرات الأسبوع الثامن