Introduction to Interactive Programming
by Lynn Andrea Stein
A Rethinking CS101 Project

Java.awt Quick Reference

AWT Components

An awt component is a visible gui entity. The root of the component hierarchy is the class java.awt.Component.

The class java.awt.Component is abstract. Its methods include:

- public void paint(Graphics g), an event-handler method supplying detailed instructions as to how to paint the Component.
- public void repaint(), a user-invoked method requesting a paint.

Specific widgets extending Component include

- Button, which has a label and can respond to being pressed.
- Label, a non-editable piece of text.
- TextField, a single line (potentially editable) text box.
- TextArea, a multi-line (potentially editable) text box.

- Checkbox, which can be checked or unchecked. If a Checkbox is part of a CheckboxGroup, only one Checkbox in the CheckboxGroup may be checked at any time.
- Choice, a popup menu, which contains a set of items. One of these items may be seleted.
- List, a Component with multiple Strings, some of which are selectable.

Most of the activity of these widgets is accomplished through the use of specialized event handlers, as described in the chapter on Event Delegation.

Two other Components deserve special mention:

- Canvas, which does nothing by itself, but is often extended.
- Container, an abstract Component capable of holding other Components inside it.

There are several varieties of Container, including

- Panel, an instantiable Container.
- Frame, a top level (outermost) Container

Component

This abstract class is the root of the visible AWT classes. All of the other classes extend it and inherit its methods. However, few subclasses rely on the full generality of Component and most of these methods are unused in most of Component's subclasses. If you want to exploit the behavior of Component, it is common to extend Canvas, the generic instantiable Component.

java.awt.Component

- abstract
- extends Object
- To cause the Component to be (re-)displayed on the screen, call its repaint() method:
  - public void repaint();
  - public void repaint(long time);
  - public void repaint(int x, int y, int width, int height);
  - public void repaint(long time, int x, int y, int width, int height);
- To give instructions for how the Component ought to look when it is time for it to appear, override its paint(Graphics g) method:
  - public void paint(Graphics g);
- Every Component that is not a Window is inside another, called its parent:
  - public Container getParent();
- If you want to know how big the Component is...
  - public Dimension getSize();
- Component event types:

    - public synchronized void addComponentListener(ComponentListener l);
    - public synchronized void addFocusListener(FocusListener l);
    - public synchronized void addKeyListener(KeyListener l);
    - public synchronized void addMouseListener(MouseListener l);
    - public synchronized void addMouseMotionListener(MouseMotionListener l);
    - public synchronized void removeComponentListener();
    - public synchronized void removeFocusListener();
    - public synchronized void removeKeyListener();
    - public synchronized void removeMouseListener();
    - public synchronized void removeMouseMotionListener();
- Override these to specify a different size from the default for your Component
    - public Dimension getMaximumSize();
    - public Dimension getMinimumSize();
    - public Dimension getPreferredSize();
- Used for double-buffering:
    - public Graphics getGraphics();

There are many, many other methods available in java.awt.Component. However, the vast majority of these (and even several of the ones listed here) are not relevant to the material covered in this book. Check the on-line Java API documentation for details.

Canvas

java.awt.Canvas

A Canvas is an instantiable Component. It has no additional behavior beyond that inherited from Component. It is often extended and customized, particularly by overriding its paint() method or supplying specialized event listeners.

- extends Component
- public Canvas();
- Canonical usage:
    - subclass Canvas, create instance of subclass, add this (subclassed) Canvas to Container: class SpecialCanvas extends Canvas{ ... }
    - common to override paint()
    - common to addMouse(Motion)Listener

Widgets and their Event Types

Button, Checkbox, Choice, List, TextArea and TextField, are each types of GUI widgets. Each is a subclass of java.awt.Component and a member of the package java.awt.

Component Name	Description	Main Event Generated
java.awt.Button	Clickable button with label. Clicking on this component generates an ActionEvent.	java.awt.event.ActionEvent
java.awt.Checkbox	Label with on/off mark. Clicking this item causes its state (checked/unchecked) to change. If a Checkbox is part of a CheckboxGroup, at most one Checkbox in the group can be selected.	java.awt.event.ItemEvent
java.awt.Choice	Popup with a list of labels from which a single item can be selected	java.awt.event.ItemEvent
java.awt.Label	A non-editable text item.	none
java.awt.List	List of labels, each of which may be selected or not. Clicking an item toggles (flips) its state.	java.awt.event.ItemEvent
java.awt.TextArea	A multi-line text box.	java.awt.event.TextEvent
java.awt.TextField	Box into which a single line of text may be typed. Hitting the return key causes an ActionEvent.	java.awt.event.ActionEvent

The major methods of each widget type are listed in separate sidebars, below.

Basic Widgets

java.awt.Label

- constructors
  - public Label();
  - public Label(String text);
  - public Label(String text, int alignment);
- alignment management: Symbolic constants and getter/setter for Label text alignement
  - public static final int CENTER, LEFT, RIGHT
  - public int getAlignment();
  - public synchronized void setAlignment(int alignment);
- text management: What should the Label say?
  - public String getText();
  - public synchronized void setText(String text);
- Canonical usage:
  - create Label, add Label to Container:
    Label l = new Label(*text*); *container*.add(l);

java.awt.Button

- constructors
  - public Button();
  - public Button(String label);
- label management: What text should appear next to the Button?
  - public String getLabel();
  - public synchronized void setLabel(String label);
- ActionListener management: Who needs to know when this Button is pressed?
  - public synchronized void addActionListener(ActionListener l);
  - public synchronized void removeActionListener(ActionListener);
- Canonical usage:
  - create Button, addActionListener to Button, add Button to Container:
    Button b = new Button(*label*); cb.addActionListener(*listener*); *container*.add(b);

Item Selectable Widgets

These widgets each contain multiple items, one or more of which may be selected at any time. Each implements an interface specifying certain behavior. The methods of this interface are not repeated for each of the implementing classes below.

java.awt.ItemSelectable (interface)

- Listener management: Who needs to know when one of the items is selected or deselected?
  - public void addItemListener(ItemListener l);
  - public void removeItemListener(ItemListener l);
- public Object[] getSelectedObjects; returns null if none currently selected.

java.awt.Choice (a.k.a. dropdown list)

Has a set of indexed String items. Generates ItemEvents.

- implements ItemSelectable
- constructor
  - public Choice();
- item management:
  - public synchronized void add(String item);
  - public synchronized void addItem(String item);
  - public synchronized void insert(String item, int index);
  - public synchronized void remove(String item);
  - public synchronized void remove(int index);
  - public synchrnoized void removeAll();
  - public String getItem(int index);

        - public int getItemCount(); returns how many there are currently.
    - item selection management: Which item is currently selected?
        - public synchronized void select(int index);
        - public synchronized void select(String item);
        - public int getSelectedIndex();
        - public synchronized String getSelectedItem();
    - Canonical usage:
        - create Choice, add items to Choice (one by one), addItemListener to Choice, add Choice to Container:
          Choice c = new Choice(); /* *repeatedly* */ c.add(*label*); c.addItemListener(*listener*); container.add(c);

java.awt.Checkbox

Has a label, a state (clicked or not), and possibly a CheckboxGroup. Generates ItemEvents.

- implements ItemSelectable
- constructor
    - public Checkbox();
    - public Checkbox(String label);
    - public Checkbox(String label, boolean state);
    - public Checkbox(String label, boolean state, CheckboxGroup group);
    - public Checkbox(String label, CheckboxGroup group, boolean state);
- label management: What text should appear next to the Checkbox?
    - public String getLabel();
    - public synchronized void setLabel(String label);
- state management: True is checked, false is unchecked
    - public boolean getState();
    - public void setState(boolean state);
- group management: Is this checkbox part of a group of mutually exclusive alternatives?
    - public CheckboxGroup getCheckboxGroup();
    - public void setCheckboxGroup(CheckboxGroup group);
- Canonical usage:
    - create Checkbox, addItemListener to Checkbox, add Checkbox to Container:
      Checkbox cb = new Checkbox(*label*); cb.addItemListener(*listener*); container.add(cb);
    - OR create Checkbox in CheckboxGroup, addItemListener to Checkbox, add Checkbox to Container:
      Checkbox cb = new Checkbox(*label*, *group*); cb.addItemListener(*listener*); container.add(cb);

java.awt.CheckboxGroup

- *Not* a Component!
  - extends Object implements Serializable
- constructor
  - public CheckboxGroup();
- Given a group, you can get the currently selected Checkbox:
  - public Checkbox getSelectedCheckbox();

Text Widgets

These three widget types provide varying kinds of text display and editing. TextField is by far the simplest, especially as it relies on ActionEvents triggered only when editing is "complete", e.g., when the user hits return. TextEvents allow finer-grained access to the user's editing.

java.awt.TextComponent

Parent class for TextArea, TextField; less commonly used directly.

- TextListener management: Who should listen to random text changes. Note: it is more common to use an ActionListener with a TextField
  - protected transient TextListener textListener;
  - public void addTextListener(TextListener l);
  - public void removeTextListener(TextListener l);
- Manipulating selected (highlighted) text:
  - public synchronized String getSelectedText();
  - public synchronized int getSelectionStart();
  - public synchronized int getSelectionEnd();
  - public synchronized void select(int startIndex, int endIndex);
  - public synchronized void selectAll();
  - public synchronized void setSelectionStart(int index);
  - public synchronized void setSelectionEnd(int index);
- Basic text manipulation:
  - public synchronized String getText();
  - public synchronized void setText(String text);
- Where is insertion point?
  - public int getCaretPosition();
  - public void setCaretPosition(int index);
- Can user edit text?
  - public boolean isEditable();
  - public synchronized void setEditable(boolean state);

java.awt.TextField

A single line of text, with facility for hiding (e.g., as password). Primary event type is ActionEvent, *not* TextEvent.

- extends TextComponent
- constructors
  - public TextField();
  - public TextField(String text);
  - public TextField(int columns);
  - public TextField(String text, int columns);
- Size in columns, i.e., how wide can this line of text be. Note also interacts with component size.
  - public int getColumns();
  - public void setColumns(int columns);
  - public Dimension getMinimumSize(int columns);
  - public Dimension getPreferredSize(int columns);
- If echoChar is set, text typed into the TextField will appear as echoChar. This is useful if the information typed is secret, e.g., a password.
  - public void echoCharIsSet();
  - public char getEchoChar();
  - public void setEchoChar(char echoChar);
- ActionListener is TextField's main event handler. It is triggered when the return (or enter) key is pressed.
  - public synchronized void addActionListener();
  - public synchronized void removeActionListener();
- Canonical usage:
  - create TextField with default size, addActionListener to TextField, setEditable, add TextField to Container:
    TextField tf = new TextField(*columns*); tf.addActionListener(*listener*); tf.setEditable(true); container.add(tf);

java.awt.TextArea

A full scrollable block of text. Inherits much of its behavior from TextComponent.

- extends TextComponent
- constructors
  - public TextArea();
  - public TextArea (String text);
  - public TextArea (int rows, int columns);
  - public TextArea (String text, int rows, int columns);

    - public TextArea (String text, int rows, int columns, int scrollbars);
- Size in columns and rows, i.e., how wide and high can this block of text appear. Note also interacts with component size.
    - public int getRows();
    - public void setRows(int rows);
    - public int getColumns();
    - public void setColumns(int columns);
    - public Dimension getMinimumSize(int rows, int columns);
    - public Dimension getPreferredSize(int rows, int columns);
    -
- Scrollbar appearance management:
    - public static final int SCROLLBARS_BOTH, SCROLLBARS_HORIZONTAL_ONLY, SCROLLBARS_NONE, SCROLLBARS_VERTICAL_ONLY;
    - public int getScrollbarVisibility();
- Text management (beyond TextComponent's methods):
    - public synchronized void append(String text);
    - public synchronized void insert(String text, int index);
    - public synchronized void replaceRange(String text, int startIndex, int endIndex);

Container

This abstract class is the root of the parent (container) AWT classes. All of the other container classes extend it and inherit its methods. Only classes extending Container can be a parent to another Component.

Container has four important subclasses:

- java.awt.Panel is a generic instantiable Container. It provides no additional functionality, but is often used directly or extended to create a Container instance.
- java.applet.Applet is a specialized Panel that can be used inside an applet viewer or web browser. See the appendix on Applets for further information.
- java.awt.Window is a top level Container, i.e., a Container that does not itself need to be Contained. However, Window contains no platform-specific niceties (such as resizability), so it is rarely used directly.
- java.awt.Frame is a subclass of Window that is commonly used in its place.

java.awt.Container

- abstract
- extends Component *and so inherits all of its methods*
- protected Container();

- Contained Component management. Position is dictated by this Container's LayoutManager. In this book, we stick to the default LayoutManager.
    - public void add(Component c);
    - public void add(String name, Component c);
    - public void add(Component c, int index);
    - public Component getComponent(int index);
    - public Component getComponentAt(int x, int y);
    - public Component getComponentAt(Point p);
    - public Component getComponentCount();
    - public Component[] getComponents();
    - public void remove(int index);
    - public void remove(Component c);
    - public void removeAll();
    - public void removeContainerListener(ContainerListener l);
- Special event handler:
    - public void addContainerListener(ContainerListener l);
    - public void removeContainerListener(ContainerListener l);

There are many, many other methods available in java.awt.Container as well. Check the on-line Java API documentation for details.

Panel and Frame

A Frame is a top-level Window. A Panel is a generic Container. Every component must be inside a Container except a top-level (Window) Container such as a Frame.

java.awt.Frame

- extends Window
- implements MenuContainer
- constructors
    - public Frame();
    - public Frame(String title);
- The title is displayed on the Frame's titlebar:
    - public String getTitle();
    - public synchronized void setTitle(String title);
- Make the Frame as small as it can be while still holding all of its contained Components
    - public void pack(); *inherited from Window*
- Make the Frame visible:
    - public void show(); *inherited from Window*

    - public boolean isShowing(); *inherited from Window*
- Is the user allowed to resize the Frame?
    - public boolean isResizable();
    - public synchronized void setResizable(boolean resizable);
- What to do when you're done with the Frame and its contained Components:
    - public synchronized dispose();
- Special event handler (includes window closing events)
    - public synchronized void addWindowListener(WindowListener l);
    - public synchronized void removeWindowListener(WindowListener l);
- Canonical usage:
    - create Frame, create and add Components, add WindowListener, pack Frame, show Frame
      Frame f = new Frame();
      Component c = new *ComponentSubclass()*; f.add(c); */* repeat this line */*
      f.addWindowListener(*listener*); f.pack(); f.show();
    - OR subclass Frame, create instance of subclass

java.awt.Panel

- extends Container
- constructors
    - public Panel();
    - public Panel(LayoutManager lm);
- Canonical usage:
    - create Panel, create and add Components, add Panel to Container
      Panel p = new Panel();
      Component c = new *ComponentSubclass()*; p.add(c); */* repeat this line */*
      container.add(p);
    - OR subclass Panel, create instance of subclass

Dimension, Point, and Rectangle

A dimension represents length and width; a point represents x and y coordinates. A rectangle is represented in terms of its upper lefthand corner and its height and width, i.e., combining a Point and a Dimension.

java.awt.Dimension

- implements Serializable
- constructors:
    - public Dimension();
    - public Dimension(Dimension d);

        - public Dimension(int width, int height);
    - publicly accesible fields (!!)
        - public int height;
        - public int width;
    - A nicer way to access fields:
        - public Dimension getSize();
        - public void setSize(Dimension d);
        - public void setSize(int width, int height);

java.awt.Point

- implements Serializable
- constructors:
    - public Point();
    - public Point(Point p);
    - public Point(int width, int height);
- publicly accesible fields (!!)
    - public int x;
    - public int y;
- A nicer way to access fields:
    - public Point getLocation();
    - public void setLocation(Point p);
    - public void setLocation(int width, int height);
    - public void translate(int x, int y);

java.awt.Rectangle

- extends java.awt.geom.Rectangle2D
- implements Serializable
- constructors:
    - public Rectangle();
    - public Rectangle(Dimension d);
    - public Rectangle(int width, int height);
    - public Rectangle(int x, int y, int width, int height);
    - public Rectangle(Point p);
    - public Rectangle(Point p, Dimension d);
    - public Rectangle(Rectangle r);
- publicly accesible fields (!!)
    - public int height;

        - public int width;
        - public int x;
        - public int y;
    - A nicer way to access fields:
        - public Dimension getSize();
        - public void setSize(int width, int height);
        - public void setSize(Dimension d);
        - public double getHeight();
        - public double getWidth();
        - public Point getLocation();
        - public void setLocation(int x, int y);
        - public void setLocation(Point p);
        - public double getX();
        - public double getY();
        - public Rectangle getBounds();
        - public void setBounds(int x, int y, int width, int height);
        - public void setBounds(Rectangle r);
    - Geometric predicates:
        - public boolean contains(Point p);
        - public boolean contains(Rectangle r);
        - public boolean intersects(Rectangle r);
        - public boolean isEmpty();
    - Geometric computations:
        - public Rectangle intersection(Rectangle r);
        - public Rectangle union(Rectangle r);

Graphics

A Graphics is the "screen" object on which all primitive drawing takes place. Graphics support a huge number of methods. You will almost always use the Graphics passed into a paint method when it is invoked by Java.

java.awt.Graphics

- abstract
- extends Object
- constructor
    - protected Graphics();
- Make pictures on this Graphics:

    - public abstract void clearRect(int x, int y, int width, int height);
    - public void draw3DRect(int x, int y, int width, int height, boolean raised);
    - public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle);
    - public abstract boolean drawLine(int startX, int startY, int endX, int endY);
    - public abstract void drawOval(int x, int y, int width, int height);
    - public abstract void drawPolygon(int[] xCoords, int[] yCoords, int numCoords);
    - public void drawPolygon(Polygon p);
    - public abstract void drawPolyline(int[] xCoords, int[] yCoords, int numCoords);
    - public void drawRect(int x, int y, int width, int height);
    - public abstract void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight);
    - public abstract void drawString(String string, int x, int y);
    - public void fill3DRect(int x, int y, int width, int height, boolean raised);
    - public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle);
    - public abstract void fillOval(int x, int y, int width, int height);
    - public abstract void fillPolygon(int[] xCoords, int[] yCoords, int numCoords);
    - public void fillPolygon(Polygon p);
    - public abstract void fillRect(int x, int y, int width, int height);
    - public abstract void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight);
- A Graphics draws in one color at a time. These methods access and change the currently active Color:
    - public abstract Color getColor();
    - public abstract void setColor(Color color);
    - public abstract void setXORMode(Color color);
- A Graphics displays text in one Font at a time. These methods access and change the currently active Font:
    - public abstract Font getFont();
    - public FontMetrics getFontMetrics();
    - public abstract FontMetrics getFontMetrics(Font font);
    - public abstract void setFont(Font font);
- Copy whatever is on this Graphics to a new Graphics.
    - public abstract Graphics create();
    - public Graphics create(int x, int y, int width, int height);
- Get rid of a Graphics you no longer need (*only* if you've created it!)
    - public abstract void dispose();
- You can manipulate java.awt.Images; see the online documentation for Java for details.
    - public abstract boolean drawImage(Image image, int x, int y, ImageObserver observer);

- public abstract boolean drawImage(Image image, int x, int y, int width, int height, ImageObserver observer);
- public abstract boolean drawImage(Image image, int x, int y, Color background, ImageObserver observer);
- public abstract boolean drawImage(Image image, int x, int y, int width, int height, Color background, ImageObserver observer);
- public abstract boolean drawImage(Image image, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver observer);
- public abstract boolean drawImage(Image image, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, Color background, ImageObserver observer);

AWT Events

There are many different kinds of events in the package java.awt.event. Each is a subclass of java.awt.event.AWTEvent. It is unlikely that you would ever need to create an awt event. Instead, you are likely to write Listeners that handle these Events.

java.awt.AWTEvent

The most important method of the class java.awt.AWTEvent is

- public Object getSource();

which returns the Object to which the event occurred. Because all other awt events extend AWTEvent directly or indirectly, they, too, have getSource() methods. Their getSource() methods will generally return a Component (or an instance of one of its subclasses).

Other event objects with fields worth noting are summarized in the following table:

Event Class	Notable Event Methods
ActionEvent	public String getActionCommand(); *can be used to disambiguate source.* public int getModifiers(); *indicates alt/ctrl/shift/meta keys pressed*

MouseEvent	public Component getComponent(); *same as getSource(), but typed correctly*
	public Point getPoint();
	public int getX();
	public int getY();
	public int getClickCount();
	public boolean isAltDown();
	public boolean isControlDown();
	public boolean isMetaDown();
	public boolean isShiftDown();
	public int getModifiers();

ItemEvent	public Object getItem(); *returns selected item* public ItemSelectable getItemSelectable(); *same as getSource(), but typed correctly* public int getStateChange(); *returns ItemEvent.SELECTED or DESELECTED*
WindowEvent	public Window getWindow(); *same as getSource(), but typed correctly*
ComponentEvent	public Component getComponent(); *same as getSource(), but typed correctly*
ContainerEvent	public Component getChild(); *who was added or removed* public Container getContainer(); *who it was added to/removed from. same as getSource(), but typed correctly*

ActionEvent and ActionListener

java.awt.event.ActionEvent and java.awt.event.ActionListener

Although ActionEvent does have some methods, it is most common simply to register the occurence of an ActionEvent, especially if the ActionListener is only listening to the ActionEvents of a single Component. The ActionEvent's getSource() method can always be used to disambiguate the source of ActionEvents if necessary.

The interface java.awt.event.ActionListener has a single method:

- public abstract void actionPerformed(ActionEvent e);

To handle the action events generated by a Button or TextField, you will need to write a class that implements java.awt.event.ActionListener and its actionPerformed method.

AWT Listeners and Adapters

An Adapter provides trivial implementations of its corresponding Listener's methods. Generally, you should extend the Adapter class (if available) and override any methods you wish to handle. If you will be overriding all of the methods, you may wish to implement the Listener interface directly. You must implement the interface directly in the cases where no adapter is available.

All events are public abstract void.

Event Class	Listener Interface	Adapter Class	Listener/Adapter methods
ActionEvent	ActionListener	--	actionPerformed(ActionEvent e);

MouseEvent	MouseListener	MouseAdapter	mouseClicked(MouseEvent e);
			mouseEntered(MouseEvent e);
			mouseExited(MouseEvent e);
			mousePressed(MouseEvent e);
			mouseReleased(MouseEvent e);

	MouseMotionListener	MouseMotionAdapter	mouseDragged(MouseEvent e); mouseMoved(MouseEvent e);

ItemEvent	ItemListener	--	itemStateChanged(ItemEvent e);

WindowEvent	WindowListener	WindowAdapter	windowActivated(WindowEvent e);
			Window gains focus, etc.
			windowClosed(WindowEvent e);
			successfully closed Window
			windowClosing(WindowEvent e);
			user requestedWindow close
			windowDeactivated(WindowEvent e);
			Window loses focus, etc.
			windowDeiconified(WindowEvent e);
			windowIconified(WindowEvent e);
			windowOpened(WindowEvent e);

ComponentEvent	ComponentListener	ComponentAdapter	componentHidden(ComponentEvent e);
			componentMoved(ComponentEvent e);
			componentResized(ComponentEvent e);
			componentShown(ComponentEvent e);

ContainerEvent	ContainerListener	ContainerAdapter	componentAdded(ContainerEvent e) componentRemoved(ContainerEvent e)
FocusEvent	FocusListener	FocusAdapter	focusGained(FocusEvent e); focusLost(FocusEvent e);
TextEvent	TextListener	--	textValueChanged(TextEvent e);

KeyEvent	KeyListener	KeyAdapter	keyPressed(KeyEvent e);
			keyReleased(KeyEvent e);
			keyTyped(KeyEvent e);

This chapter is excerpted from a draft of *Introduction to Interactive Programming In Java*, a forthcoming textbook. It is a part of the course materials developed as a part of Lynn Andrea Stein's Rethinking CS101 Project at the Computers and Cognition Laboratory of the Franklin W. Olin College of Engineering and formerly at the MIT AI Lab and the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology.

Questions or comments:
<webmaster@cs101.org>