Praktische Softwaretechnologie

Baher S. Salama and Károly Bósa
(Karoly.Bosa@jku.at)

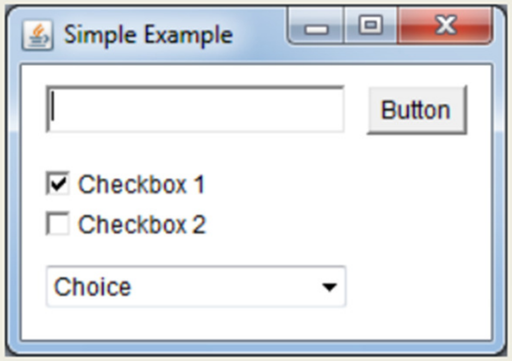
Research Institute for Symbolic Computation (RISC)

What is AWT?

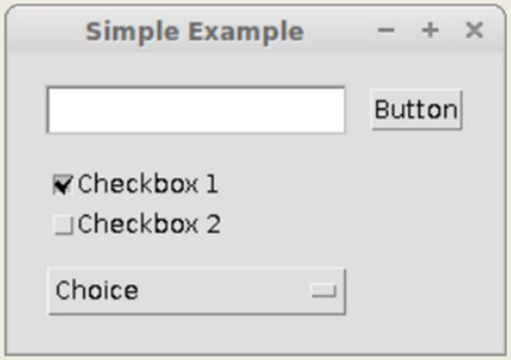
- Stands for "Abstract Window Toolkit"
- A platform-independent set of Java libraries for building GUIs
- Standard library package: `java.awt`
- Creates a layer of abstraction over native OS windowing system
- Provides a common, platform independent API which allows the programmer to write the GUI once and run it on every supported platform.
- An implementation of AWT builds the GUI using native OS windows and widgets

AWT vs. Swing

- AWT Uses heavy-weight components
- Components are displayed using the widgets of the host OS
- Same program will look different on different platforms
- Example:



Windows

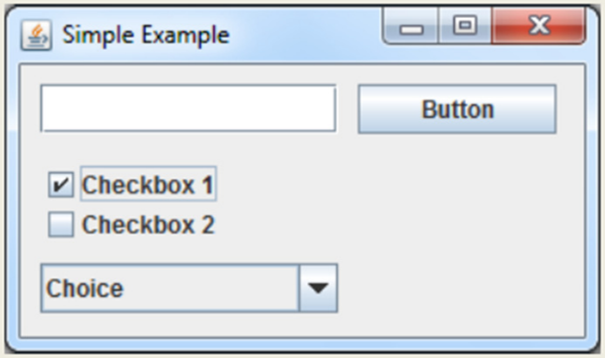


Linux (Motif)

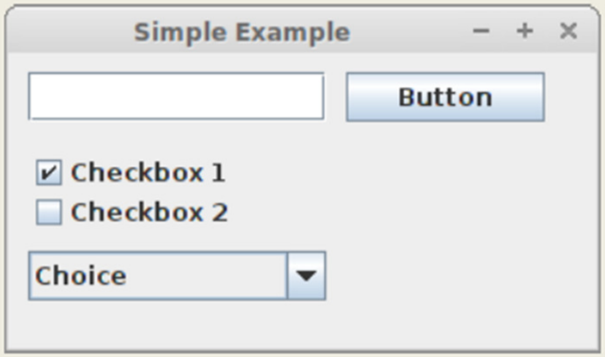
- Two windows look different from each other but widgets in each look similar to those of the native OS.

AWT vs. Swing

- Java Swing uses lightweight components
- Lightweight components are not associated with widgets of the native OS
- Displays almost the same on every platform
- Example:



Windows

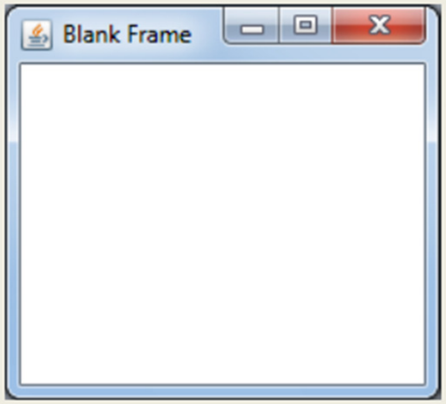


Linux

- Two windows look similar to each other but widgets look different from those of the native OS
- This presentation is concerned only with AWT

Frame

- A window with a title bar and a border



- Represented in AWT by class `java.awt.Frame`
- Code:

```
import java.awt.*;
public class BlankFrame {
    public static void main(String[] args) {
        Frame f = new Frame("Blank Frame");
        f.setSize(220,200);
        f.setVisible(true);
    }
}
```

Some Important Methods of Frame

Constructors:

`Frame()`

Constructs a new invisible frame with no title.

`Frame(String title)`

Constructs a new invisible frame with the given title.

Instance methods:

`void setTitle(String title)`

Sets the title of the frame to the given title.

`void setBounds(int x, int y, int width, int height)`

Sets the location of the frame to (x, y) and its size to (width, height)

`void setVisible(boolean b)`

If b is true, the frame is displayed on the screen. Otherwise, it is hidden.

`void setBackground(Color c)`

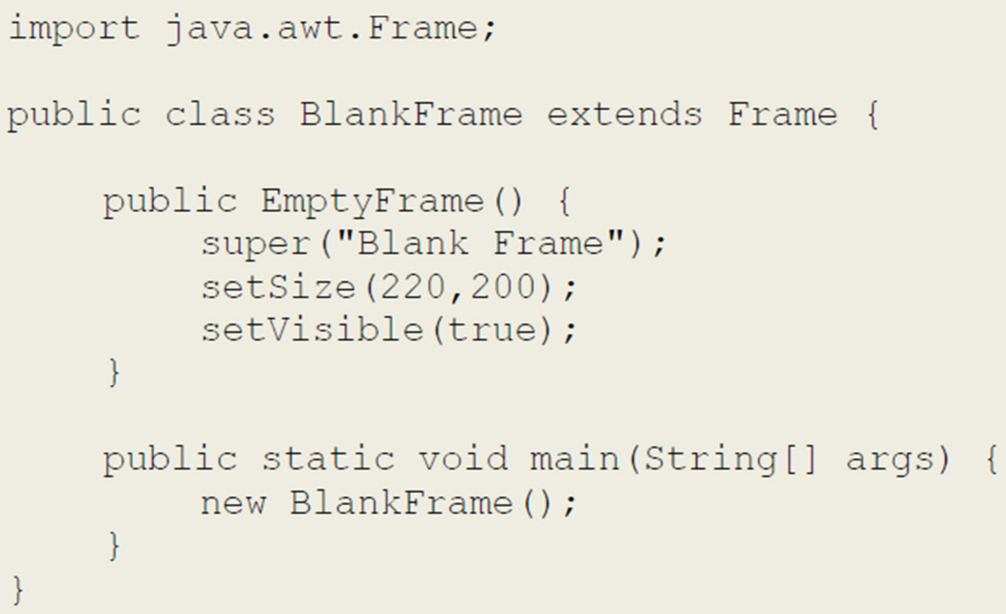
Sets the background color of the frame to c, which is an instance of class `java.awt.Color`.

`Component add(Component c)`

Adds a component to the frame. (More on components later)

Extending Class Frame

- More conventional way for designing a new frame
- Instead of creating an instance of Frame and initializing it externally, create a subclass of Frame and write the implementation inside it. Finally create an instance (or more) of the subclass.
- The following code creates the same empty frame shown previously



```
import java.awt.Frame;

public class BlankFrame extends Frame {

    public EmptyFrame() {
        super("Blank Frame");
        setSize(220,200);
        setVisible(true);
    }

    public static void main(String[] args) {
        new BlankFrame();
    }
}
```

Components

- All windows created so far were empty
- Hardly of any practical use
- For a GUI to be useful, it needs to contain usable widgets or components
- Examples of components include:
  - Labels
  - Buttons
  - Text fields
  - Check boxes
  - ...
- Represented in AWT by the **abstract** class `java.awt.Component`
- Provides a common set of methods shared by all components
- Every AWT component has to be a subclass of `Component`

Component Methods

Constructor:

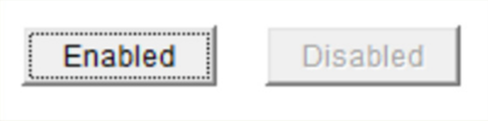
```
protected Component()
```

Invoked by a subclass of component when a new concrete component is constructed.

Instance methods:

```
setEnabled(boolean b)
```

This method can be used to enable or disable a component.



```
setLocation(int x, int y)
```

Sets the (x,y) coordinates of the component within the current Frame (or Container)

```
setSize(int width, int height)
```

Sets the width and height dimensions of the component.

Component Methods

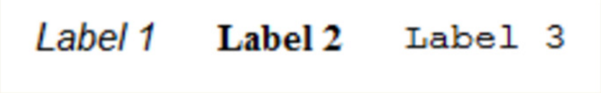
Instance methods (continued):

`setFont(Font f)`

Sets the font of any text associated with this component. The parameter is an instance of class `java.awt.Font`.

E.g: `label.setFont(new Font("SansSerif", Font.BOLD, 15))`

sets the font of the label to 15pt Sans-Serif bold.



Label 1 **Label 2** `Label 3`

`setBounds(int x, int y, int width, int height)`

Sets the location and size of the component at the same time.

`setBackground(Color c)`

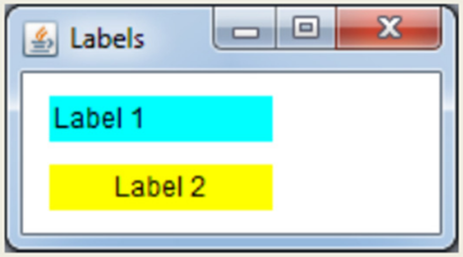
Sets the background color of the component.

- **There are also getter methods for most of these properties.**
- **Every subclass of component could have additional, more specialized methods.**

Label

- Area that displays a single line of text
- Usually used to label other components (e.g. text fields)
- Implemented by AWT class `java.awt.Label`.
- Since it is a subclass of `Component`, it inherits all component methods.
- Text in the label can be left, center, or right justified.

```
import java.awt.*;
public class Labels extends Frame {
    public Labels() {
        super("Labels");
        setSize(150,120);
        setLayout(null);
        Label l = new Label("Label 1");
        l.setBackground(Color.CYAN);
        l.setBounds(20, 40, 100, 20);
        add(l);
        l = new Label("Label 2", Label.CENTER);
        l.setBackground(Color.YELLOW);
        l.setBounds(20, 70, 100, 20);
        add(l);
        setVisible(true);
    }
}
```



Label Methods

Constructors:

`Label()`

Creates a left-justified empty Label object.

`Label(String label)`

Creates a Label object with `label` as initial text.

`Label(String label, int alignment)`

Creates a Label object with `label` as initial text and of the given alignment. The alignment is expressed using the constants `Label.LEFT`, `Label.CENTER` and `Label.RIGHT`

Instance Methods:

`void setText(String text)`

Changes the text of the label to the given string `text`.

`void setAlignment(int alignment)`

Changes the alignment of the label to the given `alignment`.

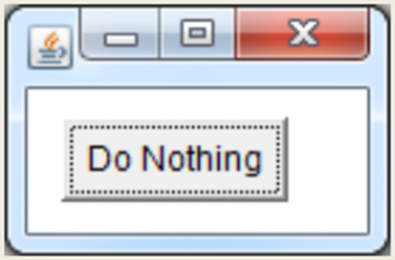
`String getText()`

`int getAlignment()`

These methods return the current text and alignment of the label, respectively.

Button

- One of the most frequently used objects.
- Implemented by AWT class `java.awt.Button`.
- When clicked by the user, it sends a signal to the program.
- Signals are implemented using Events (which will be explained shortly)
- A label can be written on the button to explain its purpose.
- Example:



Code:

```
import java.awt.*;
...
        Button b = new Button("Do Nothing");
        b.setBounds(20, 40, 80, 30);
        add(b);
...
```

Button Methods

Constructors:

```
    Button()
```

Creates a new Button with no label.

```
    Button(String label)
```

Creates a new Button with the given `label` as initial label.

Instance methods:

```
    void setLabel(String label)
    String getLabel()
```

Set and return the label on the button, respectively.

```
    void addActionListener(ActionListener l)
```

Adds an ActionListener to the button. The action listener will get notified with an Event whenever the button is pressed.

```
    void setActionCommand(String command)
```

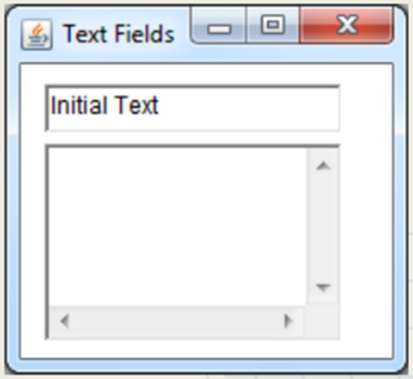
Sets the name of the command associated with this button. The command is used in the events fired by the button.

The last 2 methods are explained in more details in the Events section.

Text Field and Text Area

- Input areas where the user can type-in text input
- Implemented by classes `TextField` and `TextArea`, respectively.
- TextField is single-line, TextArea is a scrollable multiple-line area
- The two classes share a large number of instance methods
- both are subclasses of the abstract class `TextComponent`, which contains the methods common to both classes.
- Example:

```
import java.awt.*;
...
        TextField tf = new TextField("Initial Text");
        tf.setBounds(20,40,150,25);
        add(tf);
        TextArea ta = new TextArea();
        ta.setBounds(20,70,150,100);
        add(ta);
...
```



Initial Text

TextComponent Methods

Instance methods:

```
public void setText(String Text)
public String getText()
```

Set and return the text written in the text field.

```
public String getSelectedText()
```

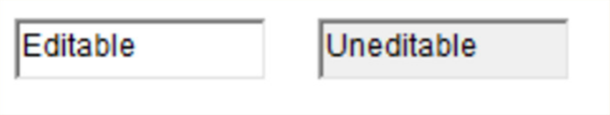
Returns a string with the text currently highlighted in the text field.

```
public void setCaretPosition(int position)
public int getCaretPosition()
```

Set and return the position of the caret. The position is 0-based.

```
public void setEditable(boolean state)
```

If `state` is `false`, the text field is read-only and cannot be modified.



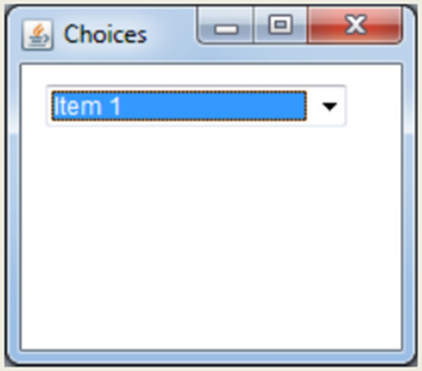
Editable

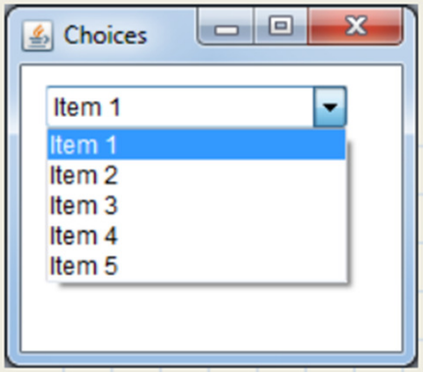
Uneditable

Dropdown Menu

- Allows the user to pick one choice from a list of choices
- Implemented in AWT by class `java.awt.Choice`.
- Instance method `add()` is used to add choices to the menu
- Example:

```
import java.awt.*;
...
        Choice ch = new Choice();
        ch.setBounds(20,40,150,25);
        for (int i = 1;i <= 5;i++)
            ch.add("Item " + i);
        add(ch);
...
```





Choice Methods

Constructor:

`Choice()`

Only constructor. Creates a new empty instance of Choice.

Instance methods:

`void add(String item)`

Adds a new `item` at the end of the list of choices.

`void remove(int position)`
`void remove(String item)`

Remove an item by position or by name. The `position` is 0-based.

`String getItem(int index)`

Returns the string of the item at the given 0-based `index`.

`String getSelectedItem()`
`int getSelectedIndex()`

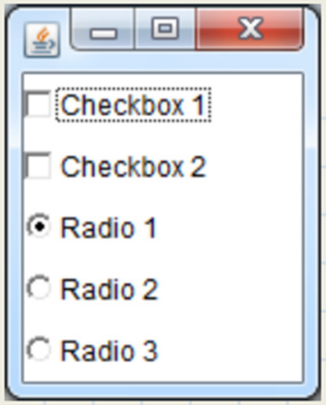
Return the name and position, respectively, of the selected item.

Checkboxes and Radiobuttons

Karoly.Bosa@jku.at

- Checkboxes allow the user to choose 0 or more from a set of choices
- Radio buttons allow the user to choose exactly one from a set of choices
- Both are implemented with class: `java.awt.Checkbox`
- To create a checkbox, the following constructor should be used:
    - **`Checkbox(String label)`**: creates a checkbox with the given label
- To create radio buttons, first an instance of `CheckboxGroup` is created.
- Each radio button in the group receives a reference to the `CheckboxGroup` instance in the constructor.

```
import java.awt.*;
...
        add(new Checkbox("Checkbox 1"));
        add(new Checkbox("Checkbox 2"));
        CheckboxGroup cbg = new CheckboxGroup();
        add(new Checkbox("Radio 1", cbg,true));
        add(new Checkbox("Radio 2", cbg,false));
        add(new Checkbox("Radio 3", cbg,false));
...
```



□ Checkbox 1

□ Checkbox 2

● Radio 1

○ Radio 2

○ Radio 3