Interactors and GUIs

Announcements

- Assignment 5 due right now.
- Assignment 6 (**NameSurfer**) out, due Wednesday, March 7 at 3:15PM.
  - Play around with **`HashMap`**s, interactors, and file processing.
  - See some interesting trends in US history.
- YEAH Hours Tonight, 7-8PM in 420-041.

NameSurfer Demo

Iterating Over a **HashMap**

- Because a **HashMap** doesn't have an order associated with it, the techniques we've used to iterate over **String**s, arrays, and **ArrayList**s won't work on it.
- Instead, we can use a **for each loop**:

```
for (KeyType key: map.keySet()) {
        /* … use key … */
}
```

- Keys will be returned in no particular order.

The “For Each” Loop

- For **`String`**s, arrays, and **`ArrayList`**s:

```
for (ElemType elem: collection) {

                    …

}
```

- Elements will be returned in sequence.
- Almost always easier to use than a standard **`for`** loop, but you don't get access to the indices as you iterate.

Combo Boxes

- A **combo box** is a drop-down list from which the user can make a selection.
- Create the combo box using

```
new JComboBox()
```

- Add each item by calling **`addItem`**.
- Set a default by calling **`setSelectedItem`**.
- Call **`setEditable(false)`** to disable editing.
- Call **`addActionListener(this)`** (plus optionally **`setActionCommand`**) to respond to events.

Checkboxes

- Java also supports checkboxes.
- You can create the checkbox using

```
new JCheckBox(title, is-initially-checked)
```

- To receive notifications from the checkbox, you must call

```
checkbox.addActionListener(this)
```

- You can retrieve whether the box is checked with

```
checkbox.isSelected()
```

The Conditional Operator

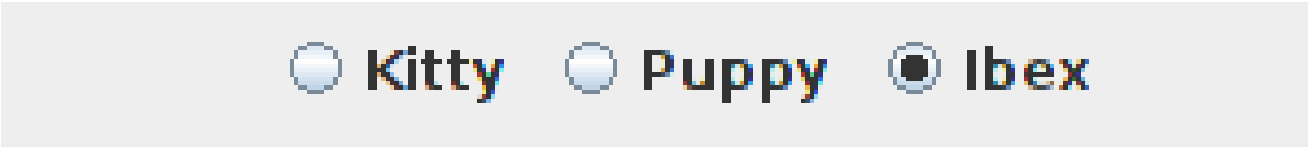
- There is one surprisingly useful Java operator we haven't covered yet: the **ternary conditional operator**.
- Syntax:

condition `?` ***if-true*** `:` ***if-false***

- First, evaluates ***condition***.
- If ***condition*** is true, the expression evaluates to ***if-true***.
- Otherwise, the expression evaluates to ***if-false***.

Radio Buttons

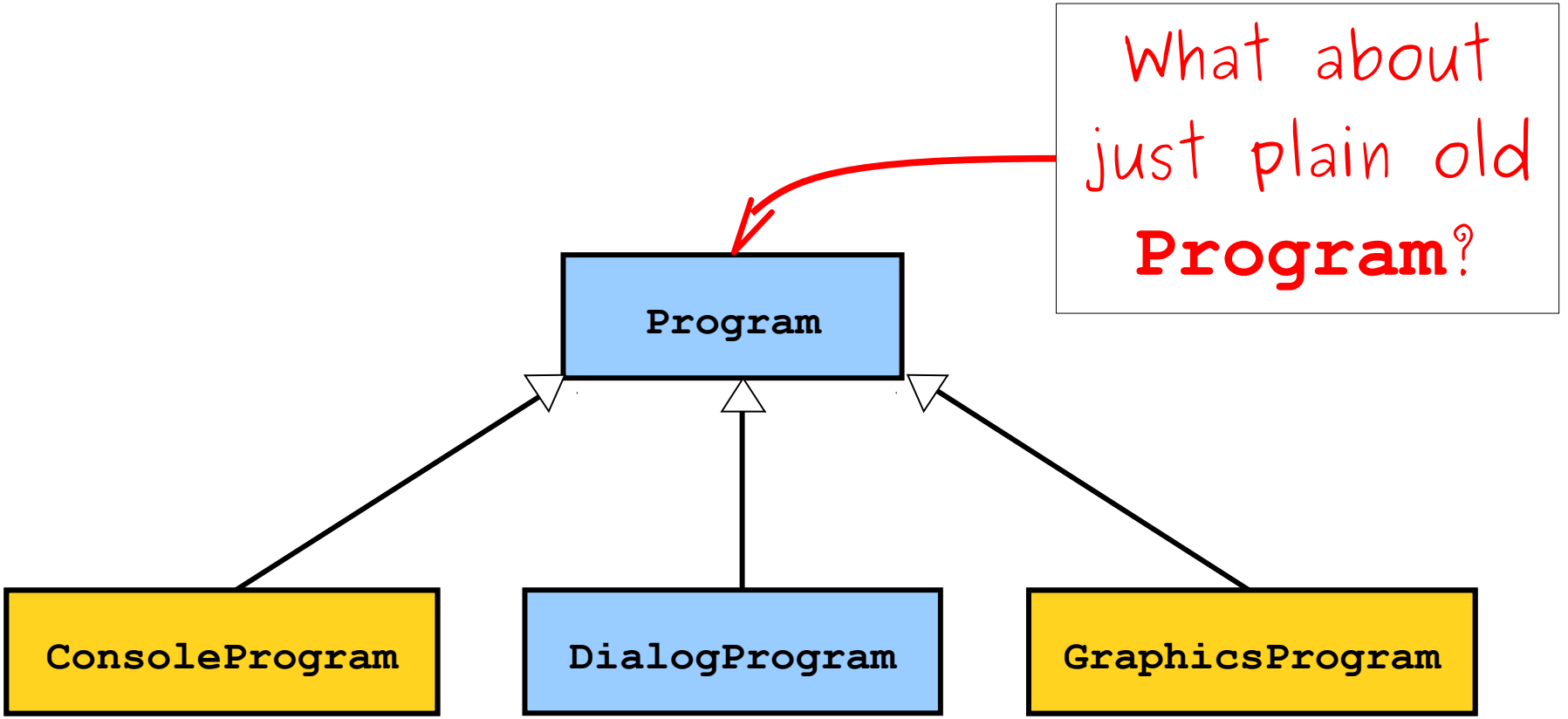
- **Radio buttons** represent a group of mutually exclusive options.



- To create a group of radio buttons:
  - Create a **`ButtonGroup`** to indicate that the buttons are all linked.
  - Create each **`JRadioButton`** and add it to the button group by calling ***group*`.add(`*button*`)`**.
  - Call ***button*`.addActionListener(this)`** on each radio button to receive notifications.

Changing the Layout

`acm.program` Hierarchy



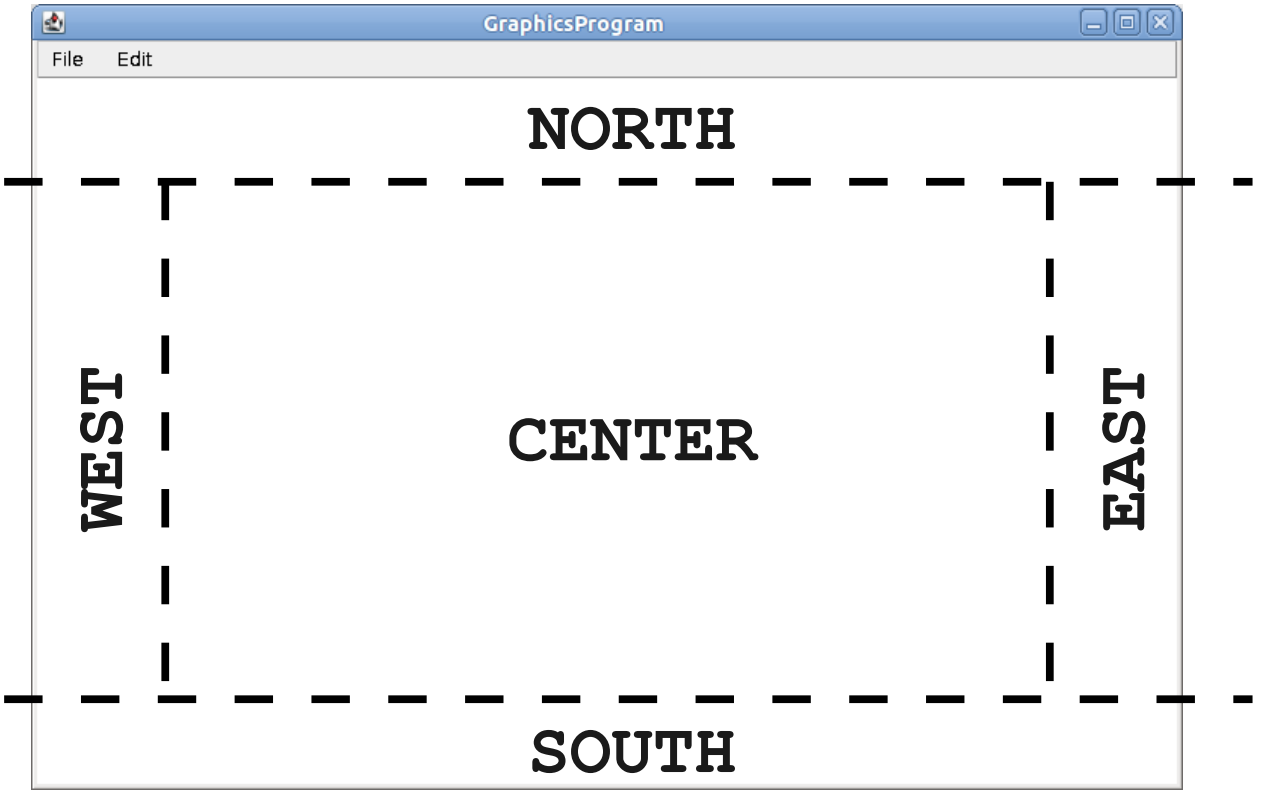
Graphic courtesy of Eric Roberts

Components and Layouts

- Each piece of a window is called a **component**.
  - The graphics part of a **`GraphicsProgram`**.
  - The console in a **`ConsoleProgram`**.
  - All interactors.
- When components are added to a window, a piece of code called the **layout manager** decides where everything should go.

BorderLayout

- The standard layout used by **Program** is the **BorderLayout**, which works as you've seen so far.



GridLayout

- You can organize components in a grid with a **GridLayout**.
- To install a grid layout, use

```
setLayout(new GridLayout(rows, cols));
```

- Components will be added across the columns of the first row from left-to-right, then the second row from left-to-right, etc.

FlowLayout

- The **FlowLayout** layout manager organizes components so that they flow horizontally, one after another.
- The components in the borders of a normal **Program** are organized in a **FlowLayout**.
- You can change the layout of the window to be a flow layout by calling

```
setLayout(new FlowLayout());
```

TableLayout

- **TableLayout** is similar to **GridLayout**, except that components don't resize to fill the grid.
- You can do some interesting layout tricks with **TableLayout**; consult the book (Page 388) for more details.