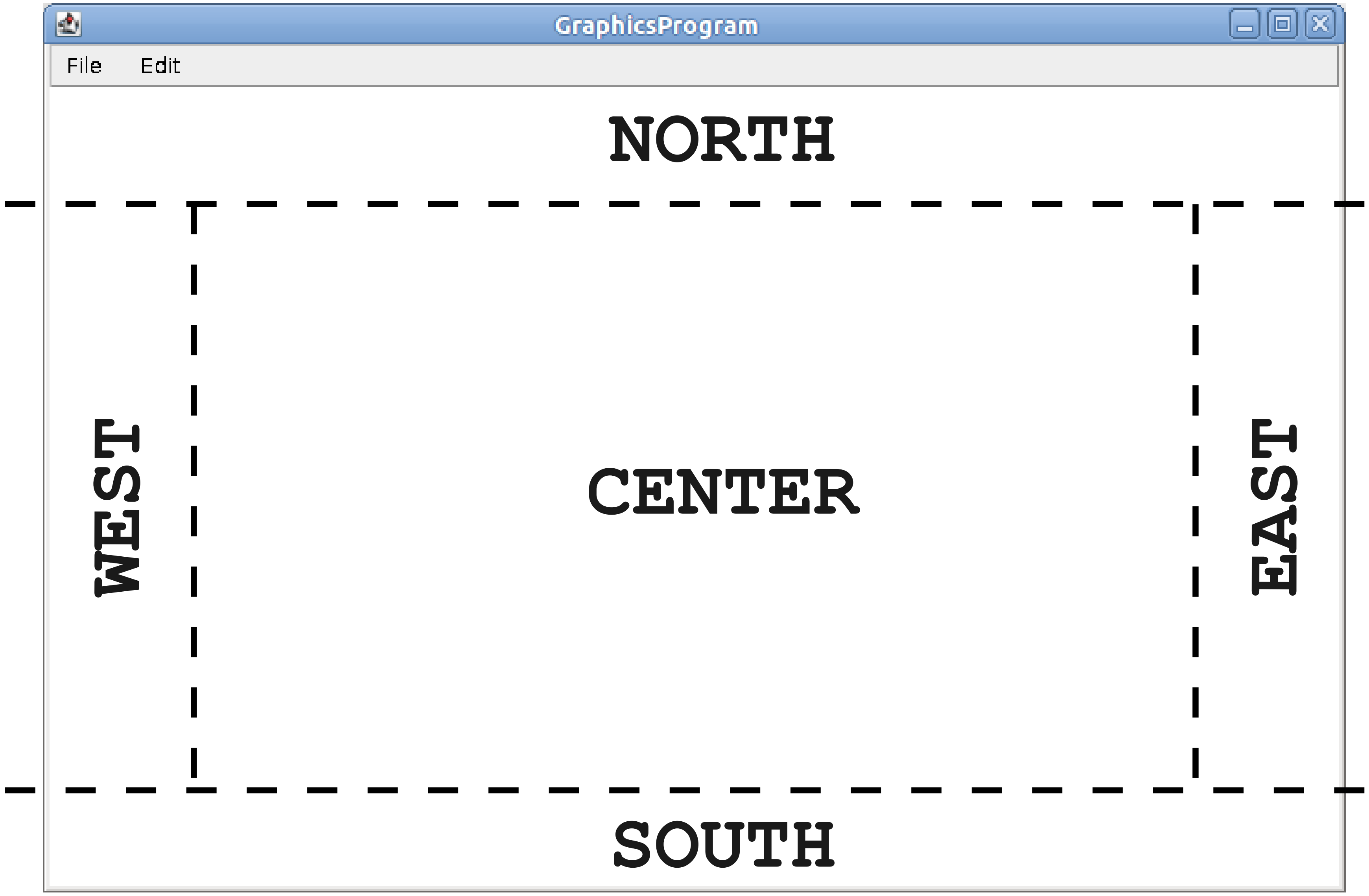
# Interactors

# Anatomy of a Window



## Introducing Interactors

- An **interactor** is a widget that can be added to a window.
- The user can then interact with the program through the interactors.

## Adding Interactors

- To use most interactors, you will need to

```
import acm.gui.*;
import javax.swing.*;
```

- You can add an interactor to the appropriate part of the window by calling

`add(`***interactor***`,` ***location***`);`

- ***location*** can be **`NORTH`**, **`SOUTH`**, **`EAST`**, or **`WEST`**.

## Structuring a Program

- Inside **`init`**:
  - Create interactors.
  - Add interactors to the program.
- Inside **`run`**:
  - Set up any graphics, state, etc.
  - Run the program.

# Text Input

- Three common text input controls:
- **`JTextField`**
  - Takes in any text as input.
- **`IntField`**
  - Only accepts **`int`** values; will prompt if you give bad data.
- **`DoubleField`**
  - Only accepts **`double`** values; will prompt if you give bad data.

# Slider Controls

- The **`JSlider`** control lets the user visually choose from a range of integers.
- Constructor:

```
new JSlider(min, max, initial)
```

- To construct a vertical slider bar:

```
new JSlider(SwingConstants.VERTICAL,
                min, max, initial)
```

# Responding to Commands

- As with mouse events, responding to interactor events requires two steps.
- Tell Java that you want to respond to commands by calling

```
addActionListeners();
```

- Respond to events by writing a method

```
public void actionPerformed(ActionEvent e)
```

## Determining the Cause

- You can tell where an **`ActionEvent`** came from in one of two ways:
- Calling **`e.getActionCommand()`**, which returns a string containing the name of the source.
    - Most common use case: the name of the **`JButton`** that was clicked.
- Calling **`e.getSource()`**, which returns a reference to the interactor that caused the event.

## Responding to Text

- If the user presses ENTER or RETURN in a text box, you will not automatically be notified of this.
- One way to get notification:

```
text.addActionListener(this);
```

- Can then use `e.getSource()` to find the text box.
- Once you've done the above, you can also

```
text.setActionCommand(command-string);
```

- Can then use `e.getActionCommand()` to find the text box.

# 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 **`addActionListeners(this)`** (plus optionally **`setActionCommand`**) to respond to events.

# 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.