Events

Events

- An ***event*** is some external stimulus that your program can respond to.
- Common events include:
  - Mouse motion / clicking.
  - Keyboard buttons pressed.
  - Timers expiring.
  - Network data available.

Events

An ***event*** is some external stimulus that your program can respond to.

Common events include:

- Mouse motion / clicking.
- Keyboard buttons pressed.
- Timers expiring.
- Network data available.

Responding to Mouse Events

- To respond to events, your program must
  - indicate that it wants to receive events, and
  - write methods to handle those events.
- Call the `addMouseListeners()` method to tell your program receive mouse events.
  - This is typically done in `run`.
- Write appropriate methods to process the mouse events.

Methods for Handling Events

- Define any or all of the following mouse event handlers to respond to the mouse:

```
public void mouseMoved(MouseEvent e)
public void mouseDragged(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
public void mouseClicked(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
```

- Notice that these are `public void` and not `private void`. This is important!
- You must also `import java.awt.event.*;` for the `MouseEvent` class.

A Virtual Hole Puncher

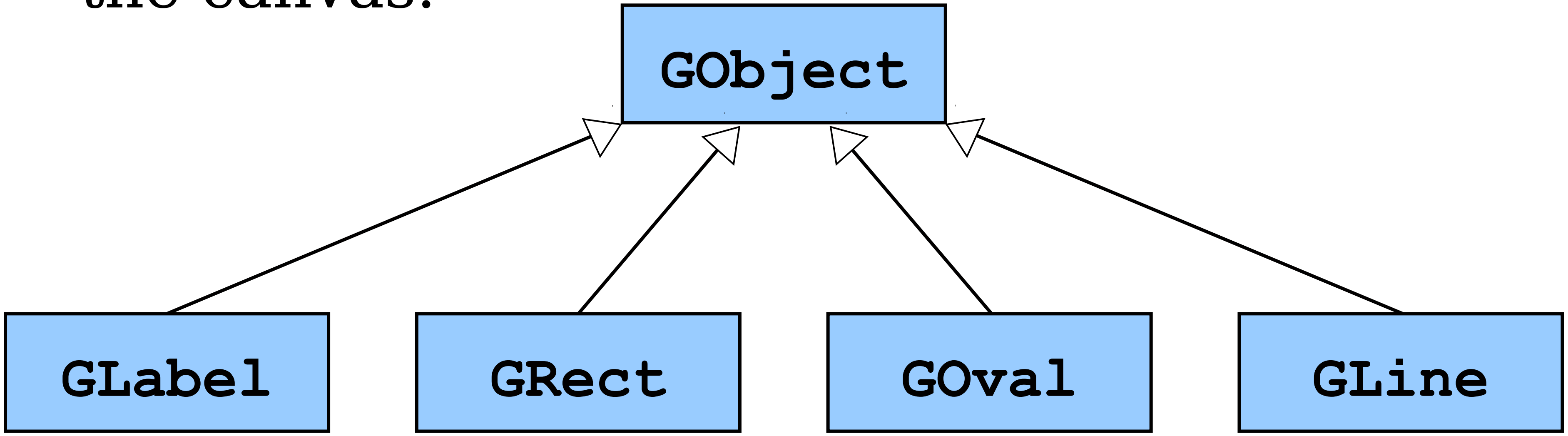
Interacting with the Canvas

Accessing the Canvas

- It is possible to determine what, if anything, is at the canvas at a particular point.
- The method

```
GObject getElementAt(double x, double y);
```

returns which object is at the given location on the canvas.



Accessing the Canvas

- It is possible to determine what, if anything, is at the canvas at a particular point.
- The method

  ```
  GObject getElementAt(double x, double y)
  ```

  returns which object is at the given location on the canvas.
- The return type is `GObject`, since we don't know what specific type (`GRect`, `GOval`, etc.) is really there.
- If no object is present, the special value **`null`** is returned.

A Debris Sweeper

Time-Out for Announcements!

Announcements

- Assignment 3 (**Problem-Solving in Java**) is due Monday at 3:15PM.
    - LaIR will be open on Sunday night if you have any questions!
- Having trouble remembering all the methods you can call? Check out the ACM Docs link on the CS106A website!
- Practice midterm next Wednesday from 7PM – 10PM in Cemex Auditorium. More details next week.

Looking for a diverse community within engineering and science at Stanford? Join one of Stanford Engineering's diversity societies!

American Indian Science and Engineering Society

Meetings
Every Monday,
12:00 Noon,
at NACC

Society of Black Scientists and Engineers

Meetings
Every Tuesday,
12:00 Noon,
at BCSC

Society of Women Engineers

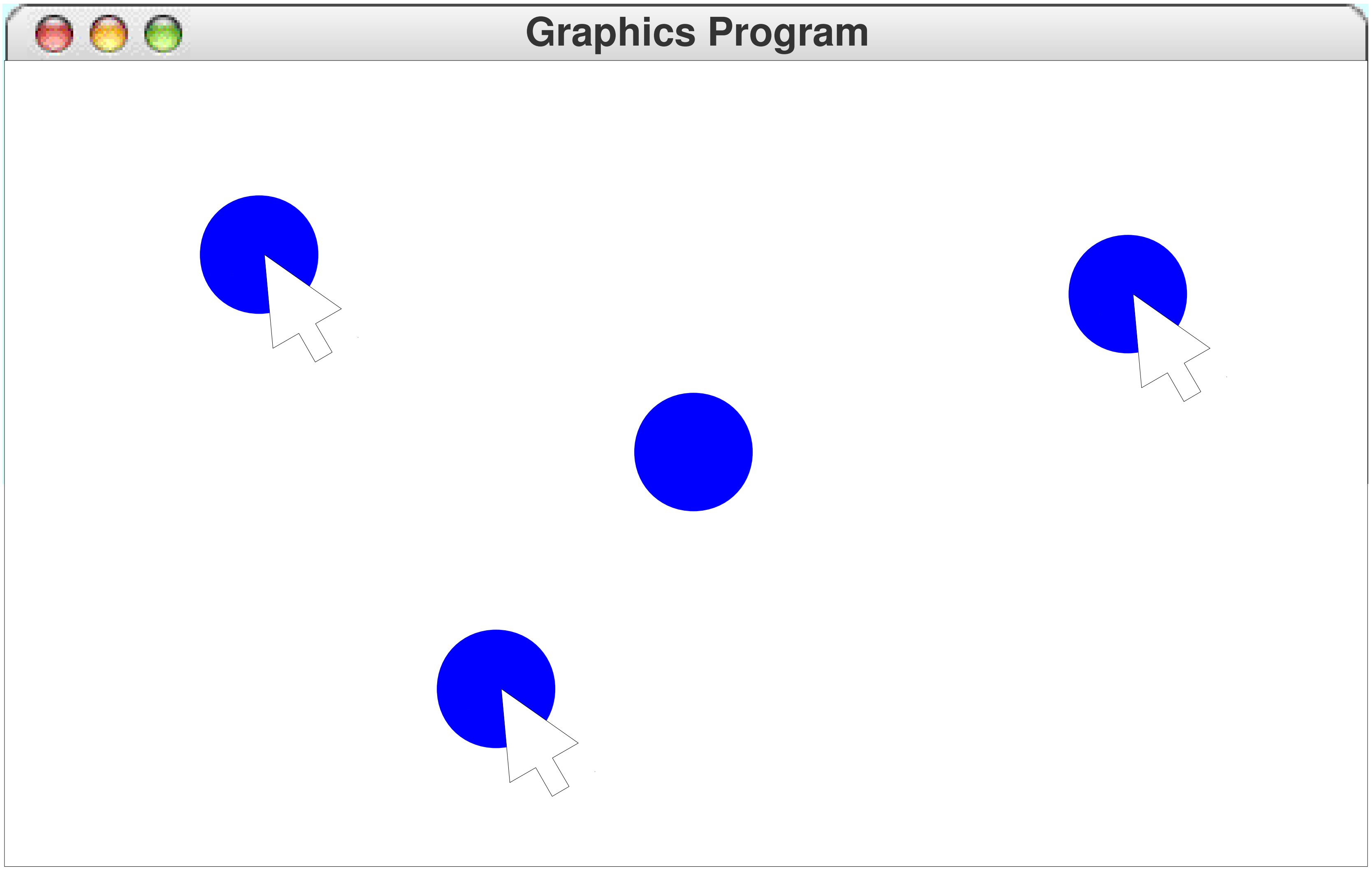
Meetings
Every Wednesday,
12:00 Noon,
at MERL
(Building 660,
Second Floor)

Society of Latino Engineers

Meetings
Every Friday,
12:00 Noon,
at BCSC

Back to CS106A!

A Friendly Circle



Let's Code it Up!

A Problem of Scoping

- The `mouseMoved` handler has no way of referring to the existing circle because it is a local variable in a different method.
- How do we make it possible for the listener to know about the circle?

Instance Variables

- An ***instance variable*** (or ***field***) is a variable that can be read or written by any method in a class.
- Define instance variables outside of any method using the syntax

```
private type name;
```

Fixing our Tracker

The Importance of Style

- It is considered *extremely* poor style to use instance variables unnecessarily:

Do not use instance variables where local variables, parameters, and return values suffice.

- Use local variables for temporary information.
- Use parameters to communicate data into a method.
- Use return values to communicate data out of a method.
- In the next assignment, we will ask you to justify each instance variable you use in your writeup. Feel free to ask your SL for advice about whether it's necessary for certain variables to be instance variables!

Event-Driven Programming

- An ***event-driven program*** is a program that primarily reacts to user events.
- If the event handlers need to remember information, they typically use instance variables to keep track of it.
- It's not uncommon to see instance variables read or written to purely by event handlers as “scratch space.”

Click-and-Drag