CSSE 220 Day 14

Event Based Programming

Check out *EventBasedProgramming* from SVN

Partner preference Survey

- There is a survey on ANGEL for you to indicate your partner preference for the next pair programming project.
- Please complete it by 1 PM on Wednesday if you want a say in who you work with.

Get Your Game On

- Share designs for the Game interface

Leftovers From Session 13

Example

- Charges: Look at completed code in new repository

Notation: In Code

interface, not class

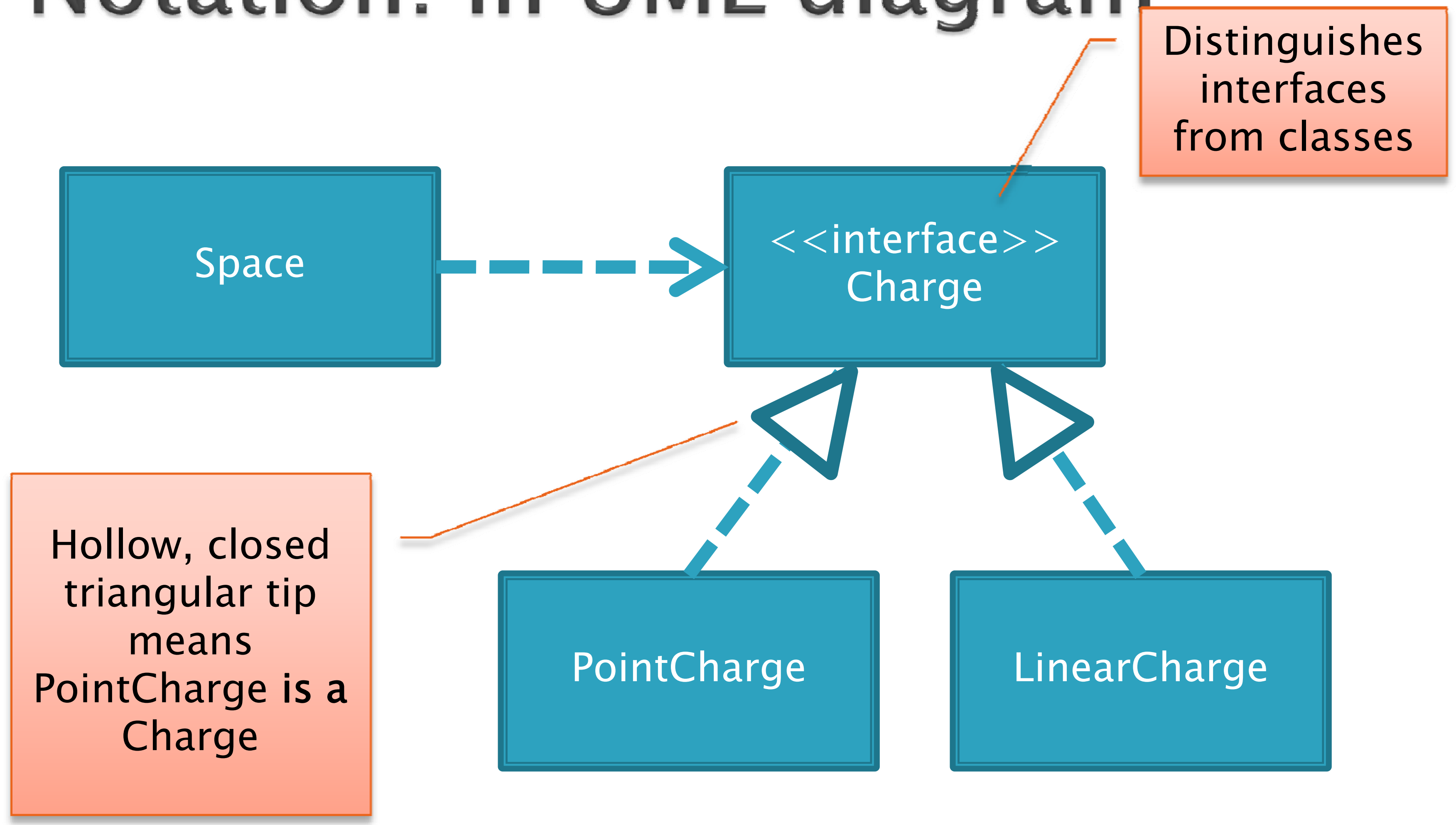
```
    public interface Charge {
        /**
         *   regular javadocs here
         */
        Vector forceAt(int x, int y);

        /**
         *   regular javadocs here
         */
        void drawOn(Graphics2D g);
    }

public class PointCharge implements Charge {

}
```

No "public", automatically are so

No method body, just a semi-colon

PointCharge promises to implement all the methods declared in the Charge interface

Notation: In UML diagram



Distinguishes interfaces from classes
Space
<<interface>> Charge
Hollow, closed triangular tip means PointCharge is a Charge
PointCharge
LinearCharge

Session 13 Q10

How does all this help reuse?

- Can pass an **instance** of a class where an interface type is expected
  - But only *if the class* ***implements*** *the interface*
- We can pass **LinearCharge**s to **Space**'s **addCharge(Charge c)** method without changing **Space**!
- We can pass any any object from a class that implements **ActionListener** to a **JButton**'s **addActionListener** method !
- **Use interface types** for fields, method parameters, and return types whenever possible

Q11

Why is this OK?

-
  ```
  Charge c = new PointCharge(…);
  Vector v1 = c.forceAt(…);
  c = new LinearCharge(…);
  Vector v2 = c.forceAt(…);
  ```
- The type of the **actual object** determines the method used.

Q12

An important Inteface (we saw this in the Fraction class)

- java.util.Comparable
  - Says that there is a "less than" ordering relation between objects of the class that implements Comparable.

```
public class Fraction implements Comparable<Fraction>{

. . .


  @Override
  public int compareTo(Fraction other){
     return this.numerator*other.denominator –
            this.denominator*other.numerator;
  }
```

Implementing this interface allows us to call `Arrays.sort()`, etc. with an array of Fractions

Packages and Folders

- Use Windows Explorer (MY Documents\...) to examine the folder structure of the OnToInterfaces packages
- In particular note
  - …JavaWorkspace\OnToInterfaces\src\edu\roseHulman\csse220\charges

Polymorphism (more later …)

- Origin:
  - Poly → many
  - Morph → shape
- Classes implementing an interface give **many differently "shaped" objects for the interface type**
- Late Binding: choosing the right method based on the actual type of the implicit parameter **at run time.**
  - **a.k.a dynamic binding**

Q13,14

Let's Get GUI: (recap and extension) Graphical User Interfaces in Java

- We say what to draw
- Java windowing library:
  - Draws it
  - Gets user input
  - Calls back to us with events
- We handle events

Hmm, donuts

New Quiz: Q1

Handling Events

- Many kinds of events:
  - Mouse pressed, mouse released, mouse moved, mouse clicked, button clicked, key pressed, menu item selected, …
- We create event listener objects
  - that implement the right **interface**
  - that handle the event as we wish
- We register our listener with an event source
  - Sources: buttons, menu items, graphics area, …

Q2

Using Inner Classes

- Classes can be defined **inside** other classes or methods
- Used for “smallish” helper classes

- Example: **`Ellipse2D.Double`**

Outer class

Inner class

- Often used for **`ActionListener`**s…

Q3

Anonymous Classes

- Sometimes very small helper classes are only used once
    - This is a job for an anonymous class!

- Anonymous → no name
- A special case of inner classes

- Used for the simplest `ActionListener`s…

Inner Classes and Scope

- Inner classes can access any variables in surrounding scope

- Caveats:
  - Local variables must be `final`
  - Can only use instance fields of surrounding scope if we’re inside an instance method

- Example:
  - Prompt user for what porridge tastes like

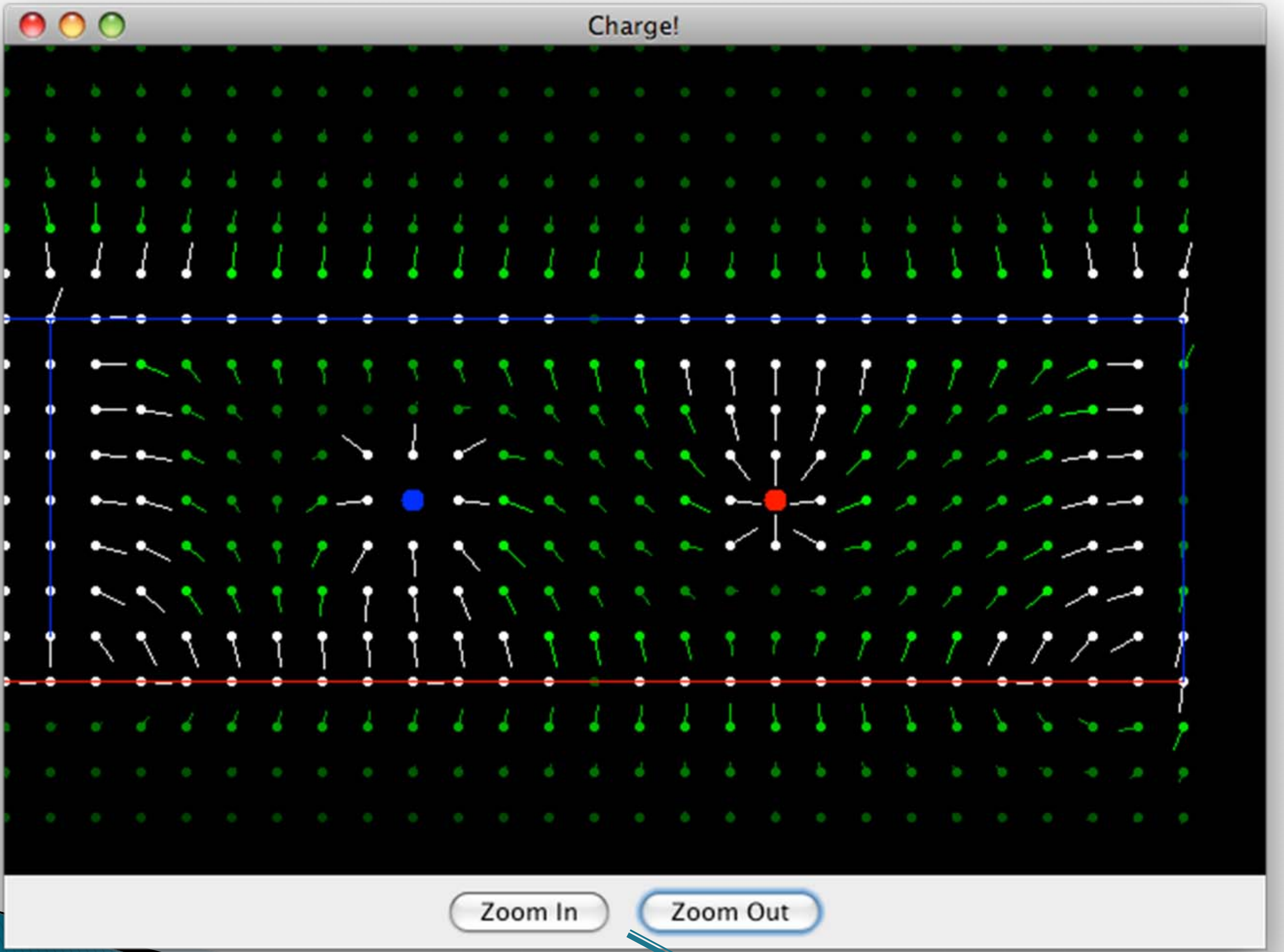
Time to Make the Buttons

- Layout in Java windows

Key Layout Ideas

- JFrame's add(Component c) method
  - Adds a new component to be drawn
  - Throws out the old one!
- JFrame also has method add(Component c, Object constraint)
  - Typical constraints:
    - BorderLayout.NORTH, BorderLayout.CENTER
  - Can add one thing to each "direction", plus center
- JPanel is a container (a thing!) that can display multiple components
- Default Frame layout is BorderLayout; default Panel Layout is FlowLayout.
- There are also GridLayout, CardLayout, etc.

Q4,5



So, how do we do this?

Repaint (and thin no more)

- With GUIs we're giving up control
  - To the user
  - To Java windowing library

- To update graphics:
  - We tell Java library that we need to be redrawn:
    - `space.repaint()`
  - Library calls `paintComponent()` when it's ready

- **Don't call `paintComponent()` yourself! It's just there for Java's call back.**

Q6

Mouse Listeners

```
public interface MouseListener {
   public void mouseClicked(MouseEvent e);
   public void mouseEntered(MouseEvent e);
   public void mouseExited(MouseEvent e);
   public void mousePressed(MouseEvent e);
   public void mouseReleased(MouseEvent e);
}
```

Q7

Possible Work Time

- BigRational
  HW 14