1.00/1.001
Introduction to Computers and Engineering Problem Solving

Quiz II Review

April 11 2012

Quiz II

- Friday April 13
- 3:05-4:25pm (80 min)
- Room: 50-340 (Walker)
- Open book/notes, No computer
- Style/length/difficulty: similar to past quizzes

What we have learned so far

- Everything from quiz I
- Recursion
- Inheritance
  - Subclasses
  - Abstract classes/methods
  - Interfaces

What we have learned so far

- Swing
  - Layout Managers
  - Events
  - Model-View-Controller
  - Graphics
  - Transformations

Recursion

- Divide and conquer or divide and combine problem solving approach

1. Define the base case
2. Divide big problem into smaller problems
3. Recursively solve the smaller problems
4. Combine the solutions to the smaller problems

Recursion

- Fibonacci Sequence: $F_n = F_{n-1} + F_{n-2}$ $F_0 = 0,\ F_1 = 1$
- Formula: Fib(n)=Fib(n-1)+Fib(n-2)

Image removed due to copyright restrictions. See: http://www.codeproject.com/KB/cpp/Recursion_Prmr_CPP_01/10-Binary_Recursion.gif

Finding max of array

Assume we can only find max of 2 numbers at a time. Suppose we want to find the max of a set of numbers, say 8 of them.

35 74 32 92 53 28 50 62

Our recursive max method calls itself:

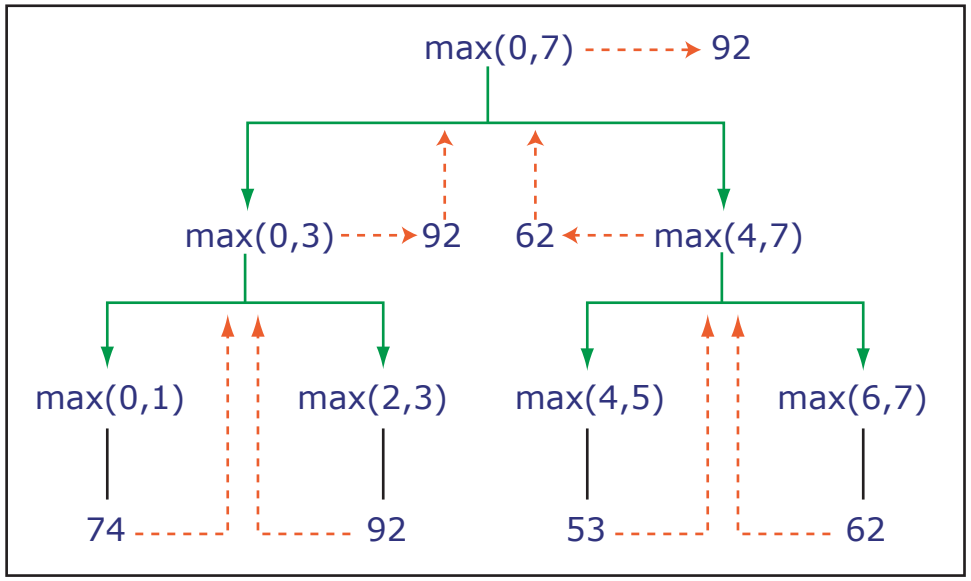


Image by MIT OpenCourseWare.

Code for maximum method

```
public class MaxRecurse {
    public static void main(String[] args) {
        int[] a= {35, 74, 32, 92, 53, 28, 50, 62};
        System.out.println("Max: " + max(0, 7, a));
    }

    public static int combine(int a, int b) {
        if (a >= b) return a;
          else return b;
    }

    public static int max( int i, int j, int[] arr) {
        if ( (j - i) <= 1) {          // Small enough
            if (arr[j] >= arr[i])
                return arr[j];
            else
                return arr[i]; }
        else                          // Divide and combine
            return (combine(max(i, (i+j)/2, arr),
                            max((i+j)/2+1, j, arr)));
    }
}
```

Inheritance: Access

- Private:
  - Access only by class's methods
- Protected
  - Access by:
    - Class's methods
    - Methods of inheriting classes, called subclasses or derived classes
    - Classes in same package
- Package (No modifier):
  - Access by methods of classes in same package
- Public:
  - Access to all classes everywhere

Inheritance: Access

http://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

Inheritance: Abstract

- May have data members like any class
- May have some implemented (concrete) methods
- May have some unimplemented (**`abstract`**) methods
  - Name says what method does
  - No information on how method works

Inheritance: Abstract

- Cannot instantiate (create object with new) abstract class
  - Why? because some methods may be abstract
- Concrete subclasses must implement all abstract methods (Override)
- Use abstract classes for organization, to provide some default behavior

Inheritance: Interfaces

- Interface lists methods that implementing class must include
  - *Like a checklist for classes*
- Set of method declarations
  - NO implemented methods
  - NO instance data members (must be **`final static`**)
- Defines a list of possible behaviors

Inheritance

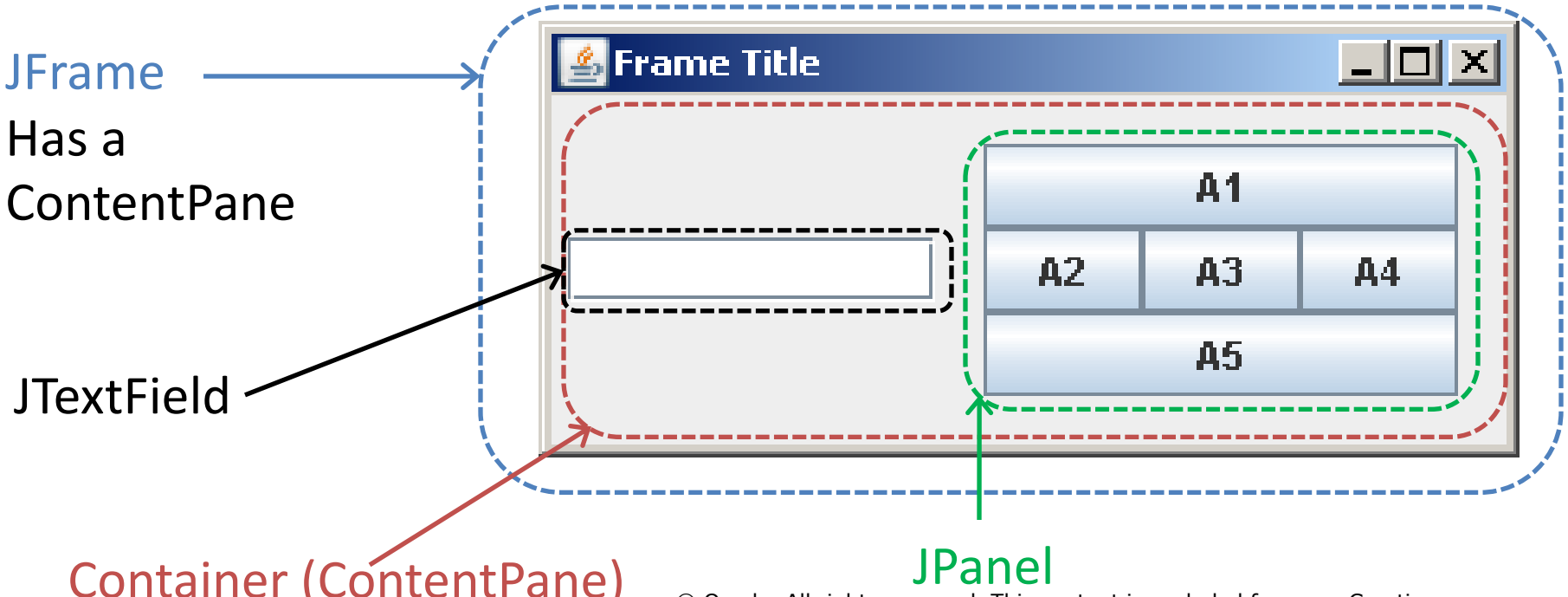
- Abstract Classes have
  - Static and instance data members
  - Concrete and/or abstract methods
  - Single inheritance (via **`extends`**)
  - Constructor
- Interfaces have
  - Static final data members (constant)
  - All methods abstract
  - "Multiple Inheritance" (via **`implements`**)
  - No constructor

`instanceof` operator checks if an object is an instance of a specified class or interface:

```
variablename instanceof Type
```

Swing

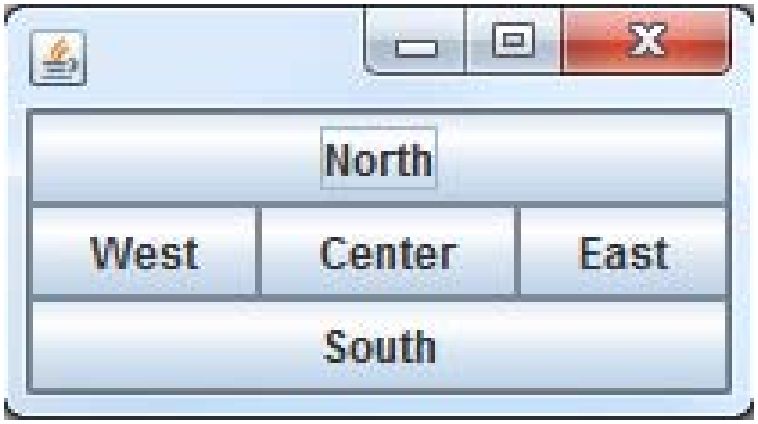
- Java's Graphical User Interface (GUI)
- Import javax.swing.* and java.awt.*



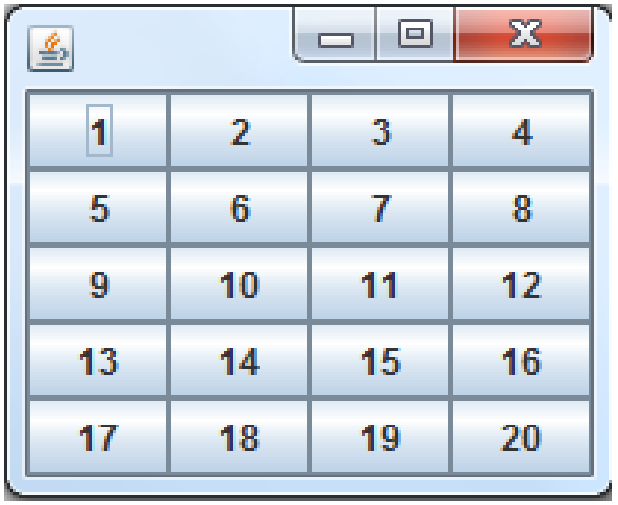
Contains a JTextField and a JPanel, organized in FlowLayout.

Contains 5 JButtons organized in BorderLayout

Swing: Layouts



BorderLayout



GridLayout(5, 4)

Grid Layout with 5 rows, 4 cols

- Default Layout
  - `BorderLayout` for Jframe's contentpane
  - `FlowLayout` for JPanel

Swing: Events

Event sources

Events are triggered by **`JComponent`**s.
Example: a **`JButton`** triggers an **`ActionEvent`** when the user clicks it

Event listeners

An object implementing a listener interface can listen to events.
Each listener interface has (a) method(s) that react to events.
Example: an object implementing the **`ActionListener`** interface has an **`ActionPerformed`** method that reacts to **`ActionEvents`** triggered by **`JButtons`**.

Source-listener relationships

Event listeners are registered at event sources
Example: **`aJButton.addActionListener(aListenerObject)`**

Swing: Events

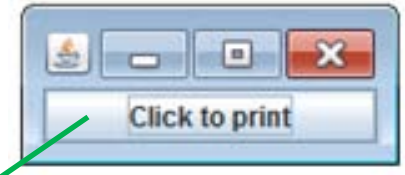
- Listener object is anything that is of type ActionListener!

```
public class InnerTest extends JPanel  {

   public class InnerButtonListener implements ActionListener{
     public void actionPerformed(ActionEvent e) { /*commands*/ }
   }

   public InnerTest(){
        ... // More commands not shown
        Jbutton b1 = new Jbutton("Button 1")
        b1.addActionListener(new InnerButtonListener());
   }
}
```

Swing: Events

- Anonymous Inner Class

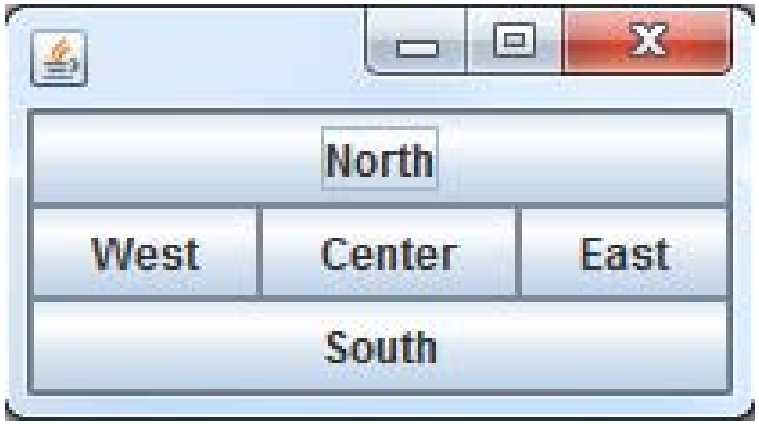
```
public class PrinterPanel extends JPanel{

    JButton b;

    public PrinterPanel(){
        b = new JButton("Click to Print")
        add(b);

        b.addActionListener(
         new ActionListener(){
           public void actionPerformed(ActionEvent e){
              System.out.println("Swing");
           }
        }
);
    }
}
```



Action Event

Layout/Event Exercise

- Construct below Jframe
- When any button is clicked, the button's text is printed



Swing: Model View Controller

- Model: computational
  - Only knows how to compute the solution
  - Doesn't know how to draw
  - Doesn't know about events, or the GUI at all
- View: purely display of results
  - Only knows how to draw
  - Doesn't know how to compute the solution
  - Doesn't know about events
- Controller: manages events
  - Manages startup (construction), object creation, events, repaints, label refreshes, exit, …
  - Doesn't know how to draw
  - Doesn't know how to compute

1.00 / 1.001 / 1.002 Introduction to Computers and Engineering Problem Solving
Spring 2012