

CS1007 lecture #19 notes

thu 4 apr 2002

- news
- GUIs
- recursion
- midterm review
- reading: ch 9, 11

cs1007-spring2002-sklar-lect19

1

news.

- midterm #2 TUE APRIL 9
- hw#5 will be posted this week and will be due TUE APRIL 16

cs1007-spring2002-sklar-lect19

2

GUIs (1).

- Graphical User Interface
- topics:
 - components
 - containers
 - layout managers
 - events
 - listeners

cs1007-spring2002-sklar-lect19

3

GUIs (2).

- *components*
- a component is a building block of any GUI
- here are some examples:
 - Label
 - TextField, TextArea
 - PushButton
 - CheckBox
 - RadioButton
 - ComboBox
 - List
 - PulldownMenu
 - ... and many more!!

cs1007-spring2002-sklar-lect19

4

GUIs (3).

- *containers*
- a container is a special component that can hold other components
- here are some examples:
 - Applet
 - Frame
 - Panel

GUIs (4).

- *layout managers*
- a layout manager describes where the components are laid out within a given container
- you need to “set” the layout manager for each container
- you can “nest” containers (and their layout managers)
- BorderLayout — simplest layout manager
- looks like this:

	north	
west	center	east
	south	

recursion (1).

- recursion is defining something in terms of itself
- there are many examples in nature
- and in mathematics
- and in computer graphics, e.g., the Koch snowflake (textbook, p.485)

power function.

- *power* is defined recursively: $x^y = \begin{cases} 1 & \text{if } y == 0, \\ x & \text{if } y == 1, \\ x * x^{y-1} & \text{otherwise,} \end{cases}$

here it is in a Java method.

```
public int power ( int x, int y ) {
    if ( y == 0 ) {
        return( 1 );
    }
    else if ( y == 1 ) {
        return( x );
    }
    else {
        return( x * power( x, y-1 ) );
    }
} // end of power() method
```

- Notice that `power()` calls itself!
- You can do this with any method *except* `main()`
- BUT beware of infinite loops!!
- You have to know when and how to stop the recursion — what is the *stopping* condition

let's walk through `power(2, 4)`.

call	x	y	return value
1	power(2,4)	2	4 * power(2,3)
2	power(2,3)	2	3 * power(2,2)
3	power(2,2)	2	2 * power(2,1)
4	power(2,1)	2	1 * 2

- the first is the *original call*
- followed by three *recursive calls*