CSE 8B Today

Creating simple GUIs

Javadoc

MORE memory models

The Arrays class

PSA3 next week...

SEATING ASSIGNMENT WILL BE SENT BEFORE TUESDAY EXAM (probably Monday)
Simple GUIs

In the book you will see a lot of subclasses (extends JFrame). We haven’t learned this yet so don’t worry about it. In our code this week we will just create JFrames and add other components to them. Also, don’t worry too much about this diagram (yet). We will revisit it in about 2-3 weeks, at which point it will all make sense.
Simple GUIs: Focus on the “green boxes”

- **JFrame**: A top-level window/container to put other graphical components in
- **JLabel**: A component that you can add text to (among other things)
- **JButton**: A component that the user can press
- **FlowLayout, GridLayout, BorderLayout**: Helper classes that govern where components appear in the JFrame (or the JPanel)
- **JPanel**: A component that can store other components (to help you arrange them neatly)

A very rough guide to creating a simple GUI:

1. Make a JFrame
2. Set its properties
3. Add a Layout Manager
4. Create and add components
5. Make the JFrame visible

You don’t always have to do this in this order. Some of the steps can have substeps (e.g., creating a JPanel to organize other components)
public static void main(String[] args) {
    JFrame myFrame = new JFrame("This is my window");
    myFrame.setSize(300, 400);
    myFrame.setVisible(true);
}
public static void main(String[] args) {
    JFrame myFrame = new JFrame("This is my window");
    myFrame.setSize(300, 400);
    //myFrame.setVisible(true);
}

PLACE ELEMENTS IN THE WINDOW BEFORE making it visible?
public static void main(String[] args) {
    JFrame myFrame = new JFrame("This is my window");
    myFrame.setSize(300, 400);

    FlowLayout flow = new FlowLayout(FlowLayout.LEFT);
    myFrame.setLayout(flow);
    for (int i = 0; i < 20; i++) {
        JLabel label = new JLabel("CSE 8B");
        myFrame.add(label);
    }

    myFrame.setVisible(true);
}
What is the difference?

FlowLayout flow = new FlowLayout( FlowLayout.LEFT );
myFrame.setLayout( flow );

vs.

myFrame.setLayout( new FlowLayout( FlowLayout.LEFT ) );

What is FlowLayout.LEFT?

Which is a true statement about the two snippets of code above?
A. The first one creates a FlowLayout object, but the second one does not
B. The first one works, but the second one causes an error
C. They invoke a different version of JFrame’s setLayout method
D. They both pass a reference to a new FlowLayout object into JFrame’s setLayout method
E. None of the above
public static void main(String[] args) {
    JFrame myFrame = new JFrame("This is my window");
    myFrame.setSize(300, 400);

    FlowLayout flow = new FlowLayout(FlowLayout.LEFT);
    GridLayout grid = new GridLayout(5, 6);
    myFrame.setLayout(grid);
    for (int i = 0; i < 20; i++) {
        JLabel label = new JLabel("CSE 8B");
        myFrame.add(label);
    }

    myFrame.setVisible(true);
}
Exam 1: Tuesday next week (in class)

• Will be short: about 1 page (front and back) and 20 minutes

• Questions similar to clicker questions, worksheets, and reading quiz

• Covers material through today (emphasis on PSA 1 and 2 topics)

• We’ll provide you with a list of methods (similar to green boxes in the book)

• Best way to study: redo clicker questions, worksheets and PSAs.
Draw the memory model for this solution

Complete the following method to replace all instances of one character with another in a string (and the return the resulting string). Your method should be case sensitive.

e.g., replaceChar( "One for the money, YO!", 'O', 'i' ) \rightarrow "ine for the money, Yi!"

```java
public String replaceChar( String s, char gone, char here ) {
    char[] chars = s.toCharArray();
    for ( int i = 0; i < chars.length; i++ ) {
        if ( chars[i] == gone )
            chars[i] = here;
    }
    return String.valueOf( chars );
}
```

Extra: Can you replace both lower and upper case instances and preserve case?
Complete the following method to replace all instances of one character with another in a string (and the return the resulting string). Your method should be case sensitive.

e.g., replaceChar( "One for the money, YO!", 'O', 'i' ) → "ine for the money, Yi!"

```java
class ReplaceChar {
    public static String replaceChar( String s, char gone, char here ) {
        char[] chars = s.toCharArray();
        for (char c : chars) {
            if (c == gone) {
                c = here;
            }
        }
        return String.valueOf( chars );
    }
}
```
public boolean anyGreaterThanN( int[] myArray, int num )
{
    for ( int x : myArray )
    {
        x = 1;
        if ( x > num )
        {
            myArray = new int[5];
            for ( int i = 0; i < 5; i++ )
            {
                myArray[i] = 1;
            }
            return true;
        }
    }
    return false;
}

public static void main( String[] args )
{
    ArrayPlay ap = new ArrayPlay();
    int[] myA = {2, 4, 6, 3, 15};
    ap.anyGreaterThanN(myA, 5);
}

What is the value of myA at the end of main?
E. I don’t know
The built-in Arrays class has useful methods for manipulating arrays. It is found in the package java.util. You can find its javadoc documentation here: http://docs.oracle.com/javase/6/docs/api/java/util/Arrays.html

```java
static void sort (Object[] a) {
    Sorts the specified array of objects into ascending order, according to the natural ordering of its elements.
}
```

> String[] stringA = {"Apple", "banana", "Zebra", "mouse"}
> java.util.Arrays.sort( stringA )

What is the value of the array that `stringA` refers to after the call to `sort`?
A. {“Apple”, “banana”, “mouse”, “Zebra”}
B. {“banana”, “mouse”, “Apple”, “Zebra”}
C. {“Apple”, “Zebra”, “banana”, “mouse”}
D. {“Apple”, “banana”, “Zebra”, “mouse”}
E. I don’t know