Now on gradesource:

- Integrity Agreement: If you don’t sign it you won’t get MAGIC NUMBER
- Participation and reading quizzes, week 2
- Slip days USED
Seating Layout – Check your group!
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(flow);
    for (int i = 0; i < 20; i++)
    {
        JLabel label = new JLabel("CSE 8B");
        label.setFont(new Font("Serif", Font.ITALIC, i*3));
        myFrame.add(label);
    }

    myFrame.setVisible(true);
}
What are those “Constant” values?

“Typical pre-defined” values are defined as **static variables** in the Class:

```java
public class Color implements Paint, Serializable{  // Color.java

	...

	/** Constant for the color white: R=255, G=255, B=255. */
	public static final Color white = new Color(0xffffff, false);

	public static final Color WHITE = white;

	/** Constant for the color light gray: R=192, G=192, B=192. */
	public static final Color lightGray = new Color(0xc0c0c0, false);
	...

public class Font implements Serializable{  // Font.java

	/**  * Constant indicating a "plain" font.  */
	public static final int PLAIN = 0;

	/**  * Constant indicating a "bold" font.  */
	public static final int BOLD = 1;

	/**  * Constant indicating an "italic" font.  */
	public static final int ITALIC = 2;

	...
```
Sort and Search methods in Arrays

(from docs.oracle.com)

```java
public static void sort(byte[] a)
```

Sorts the specified array into ascending numerical order.

**Implementation note:** The sorting algorithm is a Dual-Pivot Quicksort by Vladimir Yaroslavskiy, Jon Bentley, and Joshua Bloch. This algorithm offers $O(n \log(n))$ performance on many data sets that cause other quicksorts to degrade to quadratic performance, and is typically faster than traditional (one-pivot) Quicksort implementations.

**Parameters:**
- `a` - the array to be sorted
binarySearch

public static int binarySearch(int[] a, int key)

Searches the specified array of ints for the specified value using the binary search algorithm. The array must be sorted (as by the sort(int[]) method) prior to making this call. If it is not sorted, the results are undefined. If the array contains multiple elements with the specified value, there is no guarantee which one will be found.

Parameters:

- `a` - the array to be searched
- `key` - the value to be searched for

Returns:

- index of the search key, if it is contained in the array;
- otherwise, `-(insertion point) - 1`. The insertion point is defined as the point at which the key would be inserted into the array: the index of the first element greater than the key, or `a.length` if all elements in the array are less than the specified key. Note that this guarantees that the return value will be >= 0 if and only if the key is found.
A note about style...
PSA3: carefully read a few notes and hints

PART 1: COUNT

getWordsFromFile: read all words from a file:


setUniqueAndCounts: This method will need to create two new arrays: one for counts (int[]) and one for uniqueWords (String[]).

Don’t forget:

import ...

public String[] getWordsFromFile( String filename ) throws IOException

PART 2: DISPLAY