Questions?
Today in COMP 110

- Review of String methods
- Keyboard and Screen Input/Output
- Introduction to Java Swing
Run Code in Eclipse

- See May13.java, StringsAndChars.java, and TypeCasting.java on the Course Website for more details
Keyboard and Screen I/O: Outline

- Screen Output
- Keyboard Input
Screen Output

- We've seen several examples of screen output already.
- `System.out` is an object that is part of Java.
- `println()` is one of the methods available to the `System.out` object.
The concatenation operator (+) is useful when everything does not fit on one line.

```java
System.out.println("Lucky number = "+13+
"Secret number = "+number);
```

Do not break the line except immediately before or after the concatenation operator (+).
Alternatively, use `print()`
```java
System.out.print("One, two, ");
System.out.print(" buckle my shoe.");
System.out.println(" Three, four,");
System.out.println(" shut the door.");
```
ending with a `println()`.
Java 5.0 and 6.0 have reasonable facilities for handling keyboard input.

These facilities are provided by the `Scanner` class in the `java.util` package.

- A package is a library of classes.
Near the beginning of your program, insert:

```java
import java.util.Scanner;
```

Create an object of the `Scanner` class:

```java
Scanner keyboard = new Scanner(System.in);
```

Read data (an `int` or a `double`, for example):

```java
int n1 = keyboard.nextInt();
double d1 = keyboard.nextDouble();
```
Some `Scanner` Class Methods

- `Scanner_Object_Name.next()`
  Returns the `String` value consisting of the next keyboard characters up to, but not including, the first delimiter character. The default delimiters are whitespace characters.

- `Scanner_Object_Name.nextLine()`
  Reads the rest of the current keyboard input line and returns the characters read as a value of type `String`. Note that the line terminator `\n` is read and discarded; it is not included in the string returned.

- `Scanner_Object_Name.nextInt()`
  Returns the next keyboard input as a value of type `int`.

- `Scanner_Object_Name.nextDouble()`
  Returns the next keyboard input as a value of type `double`.

- `Scanner_Object_Name.nextFloat()`
  Returns the next keyboard input as a value of type `float`.

See p. 90
The `nextLine()` method reads
- The remainder of the current line,
- Even if it is empty.

Make sure to read *Gotcha* on p. 89
The Empty String

- A string can have any number of characters, including zero.

- The string with zero characters is called the empty string.

- The empty string is useful and can be created in many ways including
  ```java
  String s3 = "";
  ```
Java Swing

- Java makes it easy to build Graphical User Interfaces (GUIs).
- Need to import `javax.swing.*` into your program
- Read Sections 1.4 and 2.5 for more info (Graphics Supplement)
You will be using `JOptionPane` in your lab on Friday. `JOptionPane` can be used to construct windows that interact with the user. The `JOptionPane` class is imported by `import javax.swing.JOptionPane;` The `JOptionPane` class produces windows for obtaining input or displaying output.
Use `showInputDialog()` for input.

Only string values can be input.

To convert an input value from a string to an integer use the `parseInt()` method from the `Integer` class, use

```
appleCount = Integer.parseInt(appleString);
```
Output is displayed using the `showMessageDialog` method.

```java
JOptionPane.showMessageDialog(null, "The total number of fruits = " + totalFruitCount);`
Syntax

- Input
  
  String_Variable = JOptionPane.showInputDialog(String_Expression);

- Output
  
  JOptionPane.showMessageDialog(null, String_Expression);

- `System.exit(0)` ends the program.
If the input is not in the correct format, the program will crash.
If you omit the last line (System.exit (0)), the program will not end, even when the ok button in the output window is clicked.
Always label any output.
JOptionPane.showInputDialog can be used to input any of the numeric types.

Figure 2.8 Methods for converting strings to numbers

<table>
<thead>
<tr>
<th>Result Type</th>
<th>Method for Converting</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>Byte.parseByte(String_To_Convert)</td>
</tr>
<tr>
<td>short</td>
<td>Short.parseShort(String_To_Convert)</td>
</tr>
<tr>
<td>int</td>
<td>Integer.parseInt(String_To_Convert)</td>
</tr>
<tr>
<td>long</td>
<td>Long.parseLong(String_To_Convert)</td>
</tr>
<tr>
<td>float</td>
<td>Float.parseFloat(String_To_Convert)</td>
</tr>
<tr>
<td>double</td>
<td>Double.parseDouble(String_To_Convert)</td>
</tr>
</tbody>
</table>
To output multiple lines using the method `JOptionPane.showMessageDialog`, insert the new line character '\n' into the string used as the second argument.

```java
OptionPane.showMessageDialog(null,
"The number of apples\n" +
"plus the number of oranges\n" +
"is equal to " + totalFruit);
```
Multi-Line Output Windows

- Figure 2.9 A dialog window containing multiline output

![Message Window Example]

The number of apples plus the number of oranges is equal to 12
Next up: Lab

- Lab 2
- Programming help for Program 1