Announcements

- Signed Honor Code is Due Now

- Homework 0 Results
  - Current office hours will stand
  - Mix of 5th and 6th edition textbooks
  - Your goals and the class goals are well matched
Questions?

- Last Friday:
  - Designing Programs & Primitive Types
    - Writing Algorithms as Pseudocode
    - Variables & Primitive Types
    - Operators
  - Lab 0: Installing Java & Eclipse
Today in COMP 110

- Lecture
  - Strings
  - Console Input/Output
  - Documentation & Style

- Lab 1: Pseudocode ⇔ Code
Strings

- In programming, a string is a sequence of characters, e.g.
  - “Hello World!”
  - “Enter a whole number from 1 to 99.”

- In Java, the String class handles strings
  - Not a primitive type
  - Each character in the string is char type
String Declaration & Output

- **Code Example**
  - String animal = "tiger";
  - System.out.println(animal);

- **Output**
  - tiger
String Concatenation Operator

- + (binary)

Example:
- String animal = "tiger";
  String sentence = "My favorite animal is the " + animal;
  System.out.println(sentence);

Output:
- My favorite animal is the tiger
String Concatenation Operator

- + (binary)

Example:
- String animal = "tiger";
  String sentences = "My favorite animal is the " + animal + ". What is yours?";
  System.out.println(sentences);

Output:
- My favorite animal is the tiger. What is yours?
String Concatenation Operator

- Concatenating Strings with other types
  - Produces String typed value
  - Example
    - ```java
    int n1 = 7, n2 = 13;
    String s1 = "I like the number " + n1;
    String s2 = n2 + " can be unlucky";
    ```
String Concatenation Operator

- Concatenating Strings with other types
  - Gotcha Example
    - ```java
      int yearHundreds = 20, yearOnes = 12;
      String sYear1 = yearHundreds + yearOnes + " is the current year";
      String sYear2 = "The current year is " +
      yearHundreds + yearOnes;
      System.out.println(sYear1);
      System.out.println(sYear2);
    ```
  - Gotcha Output
    - 32 is the current year
    - The current year is 2012
  - Due to left-to-right order of operations on binary +
    - Fix: ```java
      String sYear1 = "" + yearHundreds + yearOnes + 
      " is the current year";
    ```
String (Class Type)

- Class types have methods

```java
String className = "COMP110";
int len = className.length();
```

Value: 7
Sample String Methods

<table>
<thead>
<tr>
<th>Invocation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>myString.length()</td>
<td>Returns the number of characters in myString</td>
</tr>
<tr>
<td>myString.equals(&quot;a string&quot;)</td>
<td>Returns true if myString and &quot;a string&quot; represent the same string</td>
</tr>
<tr>
<td>myString.toLowerCase()</td>
<td>Returns the lower case version of myString</td>
</tr>
<tr>
<td>myString.trim()</td>
<td>Removes whitespace from the beginning and end of myString</td>
</tr>
</tbody>
</table>

- You will explore these in the Lab tomorrow
- There are many more String methods
String Indices

- String output = "UNC is Great!";
- int len = output.length(); // 13

```
0 1 2 3 4 5 6 7 8 9 10 11 12
U N C i s G r e a t !
```

- char ch = output.charAt(7); // 'G'

Substrings

String output = "UNC is Great!";
int len = output.length(); // 13

String sound = output.substring(7, 9);
// Gr
Special Characters

- Double-quotes begin and end a string

- How do you put a double-quote inside a string?
  - "You can \"quote\" something like that."

- If " stores a double-quote, then how do you store a backslash?
  - "Insert a \ \ like that."
## Escape Characters

<table>
<thead>
<tr>
<th>Escape Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>Double-quote, required to store &quot; in a string value</td>
</tr>
<tr>
<td>'</td>
<td>Single-quote, required to store ' in a character value</td>
</tr>
<tr>
<td>\</td>
<td>Backslash</td>
</tr>
<tr>
<td>\n</td>
<td>New Line</td>
</tr>
<tr>
<td>\r</td>
<td>Carriage Return</td>
</tr>
<tr>
<td>\t</td>
<td>Tab</td>
</tr>
</tbody>
</table>
ASCII & Unicode

- **ASCII**: American Standard Code for Information Interchange
  - 1-byte characters (technically 7-bits, with 8-bit supersets like ISO-8859-1)
    - $2^7 = 128$ possibilities or $2^8 = 256$ possibilities
  - Includes characters typically found on English language keyboards

- **Unicode**
  - 2-byte characters
    - 16 bits $\rightarrow 2^{16} = 65536$ possibilities
  - Superset of ASCII
| Dec | Hx | Oct | Char      | Dec | Hx | Oct | Html | Char      | Dec | Hx | Oct | Html | Char      | Dec | Hx | Oct | Html | Char      |
|-----|----|-----|-----------|-----|----|-----|------|-----------|-----|----|-----|------|-----------|-----|----|-----|------|-----------|-----|----|-----|------|-----------|-----|----|-----|------|-----------|
| 0   | 0  | 000 | NULL      | 32  | 20 | 040 | #32; | Space     | 64  | 40 | 100 | #64; | B         | 96  | 60 | 140 | #96; | `        |
| 1   | 1  | 001 | SOH       | 33  | 21 | 041 | #33; | !         | 65  | 41 | 101 | #65; | A         | 97  | 61 | 141 | #97; | a        |
| 2   | 2  | 002 | STX       | 34  | 22 | 042 | #34; | "         | 66  | 42 | 102 | #66; | B         | 98  | 62 | 142 | #98; | b        |
| 3   | 3  | 003 | ETX       | 35  | 23 | 043 | #35; | #         | 67  | 43 | 103 | #67; | C         | 99  | 63 | 143 | #99; | c        |
| 4   | 4  | 004 | EOT       | 36  | 24 | 044 | #36; | $         | 68  | 44 | 104 | #68; | D         | 100 | 64 | 144 | #100; | d        |
| 5   | 5  | 005 | ENQ       | 37  | 25 | 045 | #37; | %         | 69  | 45 | 105 | #69; | E         | 101 | 65 | 145 | #101; | e        |
| 6   | 6  | 006 | ACK       | 38  | 26 | 046 | #38; | &         | 70  | 46 | 106 | #70; | F         | 102 | 66 | 146 | #102; | f        |
| 7   | 7  | 007 | BEL       | 39  | 27 | 047 | #39; | '         | 71  | 47 | 107 | #71; | G         | 103 | 67 | 147 | #103; | g        |
| 8   | 8  | 010 | BS        | 40  | 28 | 050 | #40; | (         | 72  | 48 | 108 | #72; | H         | 104 | 68 | 150 | #104; | h        |
| 9   | 9  | 011 | TAB       | 41  | 29 | 051 | #41; | )         | 73  | 49 | 111 | #73; | I         | 105 | 69 | 151 | #105; | i        |
| 10  | A  | 012 | LF        | 42  | 2A | 052 | #42; | `         | 74  | 4A | 112 | #74; | J         | 106 | 6A | 152 | #106; | j        |
| 11  | B  | 013 | VT        | 43  | 2B | 053 | #43; | +         | 75  | 4B | 113 | #75; | K         | 107 | 6B | 153 | #107; | k        |
| 12  | C  | 014 | FF        | 44  | 2C | 054 | #44; | ,         | 76  | 4C | 114 | #76; | L         | 108 | 6C | 154 | #108; | l        |
| 13  | D  | 015 | CR        | 45  | 2D | 055 | #45; | -         | 77  | 4D | 115 | #77; | M         | 109 | 6D | 155 | #109; | m        |
| 14  | E  | 016 | SO        | 46  | 2E | 056 | #46; | .         | 78  | 4E | 116 | #78; | N         | 110 | 6E | 156 | #110; | n        |
| 15  | F  | 017 | ST        | 47  | 2F | 057 | #47; | /         | 79  | 4F | 117 | #79; | O         | 111 | 6F | 157 | #111; | o        |
| 16  | 10 | 020 | DLE       | 48  | 30 | 060 | #48; | 0         | 80  | 50 | 120 | #80; | P         | 112 | 70 | 160 | #112; | p        |
| 17  | 11 | 021 | DC1       | 49  | 31 | 061 | #49; | 1         | 81  | 51 | 121 | #81; | Q         | 113 | 71 | 161 | #113; | q        |
| 18  | 12 | 022 | DC2       | 50  | 32 | 062 | #50; | 2         | 82  | 52 | 122 | #82; | R         | 114 | 72 | 162 | #114; | r        |
| 19  | 13 | 023 | DC3       | 51  | 33 | 063 | #51; | 3         | 83  | 53 | 123 | #83; | S         | 115 | 73 | 163 | #115; | s        |
| 20  | 14 | 024 | DC4       | 52  | 34 | 064 | #52; | 4         | 84  | 54 | 124 | #84; | T         | 116 | 74 | 164 | #116; | t        |
| 21  | 15 | 025 | NAK       | 53  | 35 | 065 | #53; | 5         | 85  | 55 | 125 | #85; | U         | 117 | 75 | 165 | #117; | u        |
| 22  | 16 | 026 | SYN       | 54  | 36 | 066 | #54; | 6         | 86  | 56 | 126 | #86; | V         | 118 | 76 | 166 | #118; | v        |
| 23  | 17 | 027 | ETB       | 55  | 37 | 067 | #55; | 7         | 87  | 57 | 127 | #87; | W         | 119 | 77 | 167 | #119; | w        |
| 24  | 18 | 030 | CAN       | 56  | 38 | 070 | #56; | 8         | 88  | 58 | 130 | #88; | X         | 120 | 78 | 170 | #120; | x        |
| 25  | 19 | 031 | EM        | 57  | 39 | 071 | #57; | 9         | 89  | 59 | 131 | #89; | Y         | 121 | 79 | 171 | #121; | y        |
| 26  | 1A | 032 | SUB       | 58  | 3A | 072 | #58; | :         | 90  | 5A | 132 | #90; | Z         | 122 | 7A | 172 | #122; | z        |
| 27  | 1B | 033 | ESC       | 59  | 3B | 073 | #59; | ;         | 91  | 5B | 133 | #91; | [         | 123 | 7B | 173 | #123; | {        |
| 28  | 1C | 034 | FS        | 60  | 3C | 074 | #60; | <         | 92  | 5C | 134 | #92; | \        | 124 | 7C | 174 | #124; | |        |
| 29  | 1D | 035 | GS        | 61  | 3D | 075 | #61; | =         | 93  | 5D | 135 | #93; | ]         | 125 | 7D | 175 | #125; | )        |
| 30  | 1E | 036 | RS        | 62  | 3E | 076 | #62; | >         | 94  | 5E | 136 | #94; | ^        | 126 | 7E | 176 | #126; | ~        |
| 31  | 1F | 037 | US        | 63  | 3F | 077 | #63; | ?         | 95  | 5F | 137 | #95; | `         | 127 | 7F | 177 | #127; | DEL      |

http://www.asciitable.com/
Unicode Escape Sequences

- Syntax `\uHHHHH`, where `H` is a hexadecimal digit (0-9, A-F)

- Examples
  - `char ch_A = '\u0041'; // A`
  - `char ch_a = '\u0061'; // a`
  - `char ch_pi = '\u03C0'; // π`
  - `String yum = "I like \u03C0";`
I/O

I/O: Input/Output
- Read in from a source
- Write out to a sink

Console I/O
- Read in from the keyboard
- Write out the window

Other I/O Sources/Sinks
- Files
- Network
- Devices
Console I/O

- Supported by two objects
  - Input: System.in
  - Output: System.out
Writing to Console Output

What is the difference?

- System.out.print("This is a string");
- System.out.println("This is a string");
Writing to Console Output

What is the difference?

- System.out.print("This is a string");
- System.out.println("This is a string");

The println method outputs the string and then advances to the next line of output, the print method does not.
Writing to Console Output

Equivalent Statement Groups

- System.out.println("This is a string");
- System.out.print("This is a string");
  System.out.println();
- System.out.print("This is a string");
  System.out.print("\r\n");  // On Windows
- System.out.print("This is a string");
  System.out.print("\n");  // On Linux/Unix/Mac OS X

\r\n versus \n
- Both tend to work on any modern system, for compatibility reasons
- println is recommended to be safely platform agnostic
For convenience, wrap System.in in a Scanner object

Scanner keyboard = new Scanner(System.in);
int myInt = keyboard.nextInt();
double myDouble = keyboard.nextDouble();
String word = keyboard.next();
String line = keyboard.nextLine();

Most next…() methods

- Discard preceding separators
- Read the value
- Stops before the following separator

Gotcha (p. 89 5th ed.): nextLine() reads through the \r\n, but then discards the \r\n.
Documentation & Style

- Meaningful Names
- Indenting
- Comments
- Named Constants
Meaningful Names

- Assigning names that are easy to understand reduces the amount of other documentation required
Indenting

- Block: A grouping construct of code, often bounded by `{ and }`

Indenting Style Rules

- Indent by one level within a block
  ```java
  { 
  System.out.println("Hello!");
  }
  ```

- Indent by one level a line continuation
  ```java
  String aBunchofStuff = "This is a bunch" + 
  "of text and a number: " + n1 + 
  ". Isn't it?";
  ```
Comments

- Provides documentation
- Compiler ignores them

Examples:

- // This is a single line comment
- // This is a pair of
  // single line comments
- /* This is a
  * multi-line comment */
Named Constants

- Syntax
  - public static final Type VARIABLE_NAME = value;

- Placement: Inside the class, outside the main method
- Style: All letters capitalized, words underscore-separated

- Example
  - public class SomeProgram
    {
      public static final int DAYS_PER_WEEK = 7;
      public static void main(String[] args)
      {
        // ...
    }
Questions?
Logistics

- Next: Lab 1

- Tomorrow:
  - Review of Chapters 1 & 2
  - Branching Statements

- Lab 2