More Data Types

✴ in addition to the data types `double` and `int` we will discuss the data types:
  - `char` for storing characters
  - `boolean` for storing boolean values
  - `String` for storing text strings

The Char Data Type

✴ variables with a data type of `char` are used to store/represent individual characters
  - character EX: ‘a’, ‘!’, ‘7’
  - note single quotes ‘’ indicate a character literal
  - ‘z’ is a character literal, “z” is a string literal

Variable Declaration

✴ syntax:

    `dataType variableName;`

✴ EX:

    `char letterGrade;`

  - assigns the label `letterGrade` to a memory location
  - tells Java that a single character will be stored at that location
Char Assignment

**EX:**

```java
char c1, c2;
c1 = 'a';
c2 = c1;
```

Char Output

**EX:**

```java
char c = '*';
Output.showValue("c is ",c);
```

Char Input

**EX:**

```java
char grade;
g grade = Input.readChar("Enter the letter grade");
```

**readChar**

**EX:**

```java
char grade;
g grade = Input.readChar("Enter the letter grade");
```
Relational Ops

- char's can be compared using the same relational operators as numbers
  - ==, !=, <, <=, and >= work as expected
  - EX: 'a' < 'z' and '5' < '6' are both true

The Boolean Data Type

- variables with a data type of boolean are used to store/represent the values true or false
- boolean literals: true, false
- true and false are Java keywords
- boolean variables can store only these two values

Variable Declaration

- syntax:
  ```
  dataType variableName;
  ```
- EX:
  ```
  boolean isBig;
  ```
- assigns the label isBig to a memory location
- tells Java that true or false will be stored at that location

Boolean Assignment

- EX:
  ```
  boolean isRaining, b, isBig;
  isRaining = false;  // boolean literal
  b = isRaining;      // boolean variable
  isBig = num > max; // boolean expression
  ```
Boolean Input/Output

- the Input and Output classes do not provide methods for direct boolean input and output
- can use an if-else for output:

```java
boolean b = ...;
if (b) // b is a boolean expression
    Output.showMessage("b is true");
else
    Output.showMessage("b is false");
```

The String “Data Type”

- variables with a data type of String (note capital S) are used to store/represent text strings
  - string EX: “hello world!”, “a”, “123”
  - note double quotes "" indicate a string literal
- String is not exactly a data type; we’ll learn the difference in about a week

Variable Declaration

- syntax:
  
  ```java
dataType variableName;
```

EX: 

```java
String name;
```
- assigns the label name to a memory location
- tells Java that a text string will be stored at that location

String Assignment

EX:

```java
String name, pres, empty;
name = "Obama"; // string literal
pres = name; // String variable
empty = ""; // empty string
```
String Output
- we have been using the Output class `showMessage` method to output string literals
  - it can also output String variables
- syntax:
  ```java
  Output.showMessage(stringVariableName);
  ```
- EX:
  ```java
  String str = "Hello World!";
  Output.showMessage(str);
  ```
  outputs: Hello World!

String Input
- the Input class provides the `readString` method for getting an input text string
- works like the other input methods, except it returns a String
- syntax:
  ```java
  variableName = Input.readString("prompt");
  ```
- EX:
  ```java
  String city;
  city = Input.readString("Enter the city");
  ```
  prompts user with “Enter the city”
  - after this statement executes city will have the user-input text string value

readString
- EX:
  ```java
  String city;
  city = Input.readString("Enter the city");
  ```
  prompts user with “Enter the city”
  - after this statement executes city will have the user-input text string value

The Newline Character
- the newline character ‘\n’ (backslash followed by n) can be inserted into a text string to act like a carriage return
- EX:
  ```java
  Output.showMessage("Barack\n\nObama");
  ```
**String Concatenation**

Two strings can be **concatenated**, or joined together to form a single string, using the concatenation operator `+` (plus sign).

**EX:**

```java
String alpha = "abc" + "def";
Output.showMessage(alpha);
```

- Outputs: `abcdef`

**String Concatenation**

Concatenating a `String` and a number or character will convert the values to strings and concatenate them.

**EX:**

```java
String answer = "The answer is " + 42;
Output.showMessage(answer);
```

- Outputs: `The answer is 42`

**Concatenating Lines**

Can be used to help build complex or multi-line strings.

**Sentence EX:**

```java
int w = 2, h = 3, area = 6;
String answer = 
  "Rectangle with width " + w + " and height " + h + " has area " + area;
Output.showMessage(answer);
```

- Outputs: `Rectangle with width 2 and height 3 has area 6`

**Concatenating Lines**

**Multi-line EX:**

```java
int w = 2, h = 3, area = 6;
String answer = "width: " + w + 
  " Height: " + h + 
  "Area: " + area;
Output.showMessage(answer);
```

- Outputs: [Image]
CS200 Data Types

<table>
<thead>
<tr>
<th>data type</th>
<th>stores</th>
<th>EX literals</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>true or false</td>
<td>true, false</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>char</td>
<td>characters</td>
<td>'a', '?', '6'</td>
<td>readChar</td>
<td>showValue</td>
</tr>
<tr>
<td>double</td>
<td>real numbers</td>
<td>3.14, -5.8, 0.0</td>
<td>readDouble</td>
<td>showValue</td>
</tr>
<tr>
<td>int</td>
<td>integers</td>
<td>-4, 6789, 0</td>
<td>readInt</td>
<td>showValue</td>
</tr>
<tr>
<td>String</td>
<td>text strings</td>
<td>&quot;hi!&quot;, &quot;a&quot;, &quot;678&quot;</td>
<td>readString</td>
<td>showMessage</td>
</tr>
</tbody>
</table>

Module 14 Vocabulary

- char
- character literal
- boolean
- boolean literal
- String
- empty string

Questions?

Email Kevin at sahrk@sou.edu