“Variables” store state in objects and methods

- A variable is a name for a location in memory
- A field (also called a data member or an attribute) is one sort of variable
  - Fields store values through the life of an object
  - They are accessible throughout the class
- Methods can include shorter-lived variables called local variables
  - They exist only as long as the method is being executed
  - They are only accessible from within the method
A variable’s declaration gives its type

- A variable must be declared by specifying the variable's name and the type of information that it will hold
  
  data type        variable name       initial value

  int count = 15;

- It is best to always provide an initial value, except for fields initialized in a constructor

- A variable’s value can be changed over time

- When a variable is referenced anywhere in your program, its current value is used
Assignment gives a variable a new value

- The assignment operator is the \( = \) sign
  
  \[
  \text{total} = 55; \\
  \]

- The expression on the right is evaluated and the result is stored in the variable on the left

- The value that was in \texttt{total} is overwritten

- You can assign only a value that is consistent with the receiving variable's declared type
Besides objects, we also use primitive data

- There are exactly eight **primitive data types** in Java
- One represents true or false values:
  - `boolean`
- One of them represents single character values:
  - `char`
- Four of them represent integers of different sizes:
  - `byte, short, int, long`
- Two of them represent floating point numbers:
  - `float, double`
Boolean values represent true or false

- The reserved words `true` and `false` are the only valid values for a boolean type:

  ```java
  boolean done = false;
  ```

- A boolean also can be used to represent any two states, such as a light bulb being on or off

- You can save the value returned by a predicate for use later
Characters can also be stored in variables

- A `char` variable stores a single character from the **Unicode character set**
- A **character set** is an ordered list of characters, and each character corresponds to a unique number
- The Unicode character set uses 16 bits per character, allowing for 65,536 unique characters
- It is an international character set, containing symbols and characters from many world languages
- ASCII (an older US character set) is a subset of Unicode
- Character literals are delimited by single quotes:
  
  `'a'    'X'    '7'    '$'    ','    '\n'`
Primitive numeric types differ in their ranges

The difference between the various numeric primitive types is their size, and therefore the range of values they can store:

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8 bits</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>16 bits</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>64 bits</td>
<td>&lt; -9 x 10^{18}</td>
<td>&gt; 9 x 10^{18}</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>+/- 3.4 x 10^{38} with 7 significant digits</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>+/- 1.7 x 10^{308} with 15 significant digits</td>
<td></td>
</tr>
</tbody>
</table>
Data can sometimes be converted to another type

- Sometimes it is convenient to convert data from one type to another, and there are several ways this can happen.
- For now, we’ll only mention the first one, which is also the most powerful.
- A cast is an explicit data conversion requested by the programmer.
  - To cast, the type is put in parentheses in front of the value being converted.
  - If not used with care, it can cause loss of data.
- For example, from last week’s example:

  ```java
  int nextChar = ...;
  if ( Character.isWhitespace( (char)nextChar ) ) ...
  ```
Strings represent sequences of text characters

- Every character string is an object in Java, defined by the `String` class.
- Every string literal is delimited by double quotation marks, and represents a `String` object.
- The plus operator (+) is used to `concatenate`, or append, one string to the end of another (producing a new string as a result).
- It can also be used to append a number (or any other data value) to a string.
- A string literal cannot be broken across two lines in a program.
The String class provides many useful methods

- Many of the methods return a value, such as an integer or a new String object
- Here are some commonly used methods:

  ```java
  int length()
  boolean equals(String another)
  boolean startsWith(String prefix)
  boolean endsWith(String suffix)
  String substring(int start, int end)
  int indexOf(int Ch)
  String toLowerCase()
  ```

- See more in the Java API documentation for the String class
When should a variable be a field?

- Deciding where to place a variable’s declaration is important for code readability
- Work on developing skill at making such decisions
- **General rule**: make each declaration as **limited in scope** as possible (that is, visible in the smallest amount of your code as is reasonable)
- For now: if it is only used in one method, it should be a **local variable**
- If it is used by more than one method and it seems like a natural attribute of the class you are defining, **make it a field**