Variables, Types & Operators
What is a variable?

- A named location in memory that holds a value.
- In Java, those values must be of a declared datatype.
- Furthermore, every defined variable has the following:
  - a name
  - a memory location
  - a value
  - a ‘data type’
What is a ‘data type’?

- A ‘data type’ (or simply type) is way of expressing some constraints around the possible values inside a variable.

- It has two purposes:
  - states the classification of a variable, such as real, integer or boolean
  - expresses the range of possible values that the variable of that classification can hold.

- Examples
  - **short** can hold whole numbers with a minimum value of -32,768 and a maximum value of 32,767.
  - **boolean** represents truth values, either **true** or **false**.
# Java's `primitive` types

<table>
<thead>
<tr>
<th>Type</th>
<th>What is it?</th>
<th>Range of values</th>
<th>Amount of memory allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8-bit, signed, two's compliment integer</td>
<td>-128 .. 127</td>
<td>1 byte</td>
</tr>
<tr>
<td>short</td>
<td>16-bit, signed, two's compliment integer</td>
<td>-32,768 .. 32,767</td>
<td>2 bytes</td>
</tr>
<tr>
<td>int</td>
<td>32-bit, signed, two's compliment integer</td>
<td>-2,147,483,648 .. 2,147,483,647</td>
<td>4 bytes</td>
</tr>
<tr>
<td>long</td>
<td>64-bit, signed, two's compliment integer</td>
<td>-9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807</td>
<td>8 bytes</td>
</tr>
<tr>
<td>float</td>
<td>Single precision (32-bit) IEEE 754 floating point</td>
<td><a href="http://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3">see: http://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3</a></td>
<td>4 bytes</td>
</tr>
<tr>
<td>double</td>
<td>Double precision (64-bit) IEEE 754 floating point</td>
<td><a href="http://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3">see: http://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3</a></td>
<td>8 bytes</td>
</tr>
<tr>
<td>char</td>
<td>A single 16-bit Unicode character</td>
<td>'u0000' .. 'uFFFF' (Decimal equivalent: 0 .. 65,535)</td>
<td>2 bytes</td>
</tr>
<tr>
<td>boolean</td>
<td>Simple flag to track true/false conditions</td>
<td>true, false</td>
<td>Not precisely defined, but a boolean value represents one bit of information.</td>
</tr>
</tbody>
</table>
Java has static typing

- This is very different than Python.

- Two very important properties you must realize….
  - You must declare a variable before you begin to use it.
  - You must specify the type of that variable upon declaration,
  - The type of that variable will never change.

- Moreover, you cannot do this!

```
1    int i = 123;
2    i = "This is a string";
```

- Sometimes it might look like we are changing the type of a variable when we cast or coerce, but in those cases we are copying the value of the variable and assigning it to a new type.
Coercion & Casting

- It is possible to convert values contained in one type into a variable of another type with 'coercion' and 'casting'.

- In these cases, we are making a copy of the value and putting it in a new variable of a new type.

- Example of coercion

```java
1 // 'Implicit' or 'widening' coercion
2 byte b = 10; // a byte is only 8 bits
3 int i = b; // an int is 32 bits! More space!
```

- Example of casting

```java
1 // 'Explicit', or 'narrowing' coercion, casting necessary
2 int i = 1024; // an int is 32 bits! More space!
3 byte b = (int) i; // a byte is only 8 bits
```

- See CoercionAndCasting.java
What are operators?

- Operators are used to apply operations to specific types.

```java
1 | int i = 5 + 5; // i == 10
2 | String s = "5" + "5"; // s == 55
```

- All operands of an operator must be of the same type. So in a sense, operators have types.

- If two operands are of different types, Java will try and coerce one of the operands for you!

- It take the ‘narrower’ type and ‘coerce’ it into the ‘wider’ type.

- Remember also that operators have precedence, PEMDAS.
Precedence & Coercion