



Store information needed by the program

- Must have a TYPE
 - int - can only store a number without a fractional part
 - float, double - can store any number, with or without a fractional part (double has more precision)
 - char - individual character (letter,digit,punctuation etc.)
 - String - text data
 - boolean - store value true or false
 - More later on values of true/false
- Must have a VALUE, the value can change as the program runs
 - if you don't provide an initial value, the compiler will supply one for you. That initial value is zero for the primitive types, null for strings.

Literal values

- Every primitive type (and Strings) can be initialized using a literal of the type. Here are examples of literal values:

Type	Literal value	Example initialization
String	"Hello World"	String greeting = "Hello World";
char	'A'	char letter = 'A';
boolean	true	boolean flag = true;
int	1024	int kilo = 1024;
double	3.154159	double pi = 3.14159;
• Note the difference between a char literal and a String literal. A char literal is in single quotes while a String literal is in double quotes. A String may consist of only one character but a char cannot consist of multiple characters.		
• String s = "A"; // LEGAL		
• char c = 'AB'; // ILLEGAL		

Naming Variables

- Must have a NAME
 - the name must start with an alphabetic (or "_")
 - the name may include alphabetics, digits, and underscores
 - the name should indicate the purpose of the variable in the overall design and purpose of the program
 - lousy variable names - larry, moe, curly, ah67, gh_78
 - reasonable variable names - width, fileName, cursorPosition, count, response
- You should not have a program that consists entirely of single letter variables names. It is too hard to read. When code is being explained in class I may however use single letter variable names, not because it is good programming style but because I don't have a lot of room on the slides.
- Java strongly encourages variables to be named all lower case except as noted below:
- If your variable name is a multi-word name then make the first word of the name lowercase and subsequent words start with a capitol.
- Examples of multi-word variable names:
 - fileName
 - remoteUserAddress
 - phoneBook

Assignment

- the `=` operator assigns
the value on the right to the variable on the left
WARNING! Not to be confused with the `==` operator which compares (asks "are they equal?").
- it is extremely counter-intuitive because it assigns
from the right to the left and we read from the left to the right.
- the left hand side MUST be a variable
- the right hand side can be a variable or an expression

```
int x = 0, y = 3, z = 4;
```

```
z = 10;           // z is given the value of 10
x = y + 5;        // x is given the value of y + 5 (i.e. 8)
x = x + 1;        // takes the current value of x and increments it by one
```

at the end of these 3 statements, what are the values of `x`, `y`, and `z`?

```
10 = z;           // WRONG , can't have number on left hand side of =
3 + y = y;        // WRONG, can't have expression (3 + y) on left hand side
```

Named Constants

A magic number is a numeric constant that appears in your code without explanation.

e.g.

```
answer = rate * 1.67;    // what is the significance of 1.67?
```

instead make a named constant using keyword **final**

```
// this is the scheduled rate increase
```

```
final float RATE_INCREASE = 1.67;
```

```
answer = rate * RATE_INCREASE ;
```

The value of a named constant cannot be assigned another value.

Java strongly encourages use of ALL CAPS for constants. Multi word constant names can have an UNDERSCORE between words

Conversions

Implicit Conversion

float op float is float

float op int is float

int op float is float

int op int is int

e.g.

int op float is float

5 + 13.5 is 18.5

int op int is int

5 + 8 is 13

3.0 / 4 is 0.75

3 / 4 is 0

3/4 * 10 is 7.5 or 0 ????

Practice

If a number has a decimal point, it's a double.

No decimal point, it's an int.

```
int i,j;
```

```
double x,y;
```

```
String numstr = "456973";
```

```
String word = "catnip";
```

```
String str;
```

- `i = 5 + 9;`

- `y = 5.9 + 8;`

- `i = 5.9 + 8;`

- `y = 5.0 + 4.0;`

- `y = 10 / 4;`

- `y = 10 / 4.0;`

- `i = 10 / 4;`

- `y = 5 / 10;`

- `i = 5 / 10;`

- `y = 5.0 / 10;`

- `y = y % 10;`

- `y + i = y;`

- `i = numstr.length();`

- `y = numstr + 10;`

- `str = word.substring(1);`

- `str = word + numstr.substring(2,2);`

- `str = y.length() + i.length();`

Operator Precedence

- $*$ / $\%$ are performed before $+$ and $-$
- operations performed left to right

$3 + 5 * 5$ is ?

$8 \% 3 + 9$ is ?

$2 \% 6 * 10$ is ?

$5 + 4 + 3 / 3$ is ?

$(5 + 4 + 3) / 3$ is ?

Shortcuts

<code>x = x + 5;</code>	same as	<code>x += 5;</code>
<code>x = x * 10;</code>	same as	<code>x *= 10;</code>
<code>x = x - 9;</code>	same as	<code>x -= 9;</code>
<code>x = x / 10;</code>	same as	<code>x /= 10;</code>
<code>x = x % 3;</code>	same as	<code>x %= 3;</code>
<code>x = x + 1;</code>	same as same as	<code>x ++; // post incr</code> <code>++ x; // pre incr</code> <code>x +=1;</code>