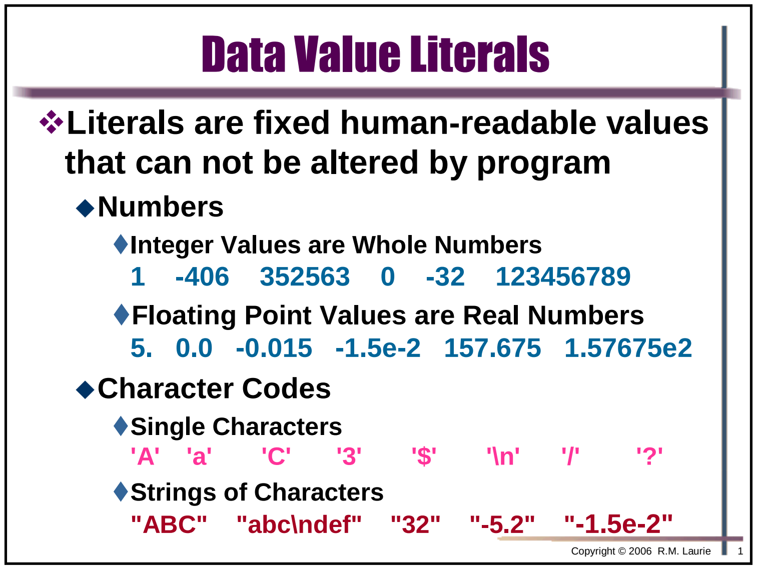
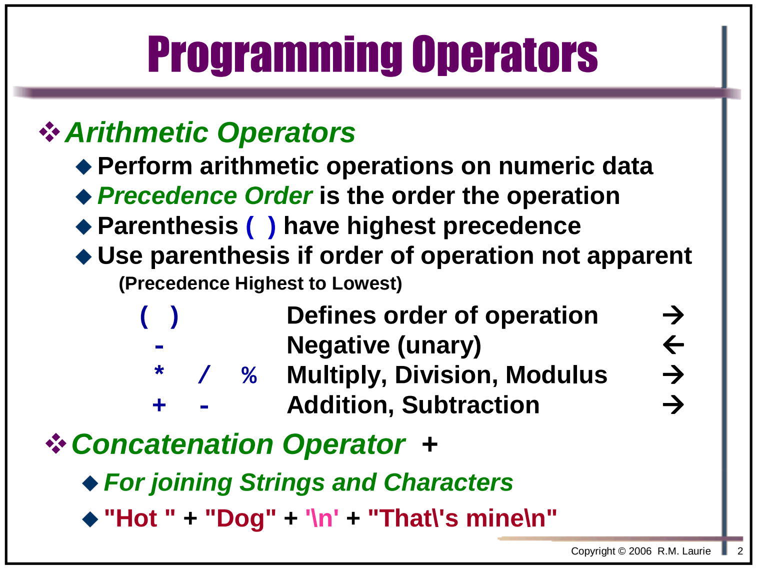


Data Value Literals

- Literals are fixed human-readable values that can not be altered by program
  - Numbers
    - Integer Values are Whole Numbers
      1   -406   352563   0   -32   123456789
    - Floating Point Values are Real Numbers
      5.   0.0   -0.015   -1.5e-2   157.675   1.57675e2
  - Character Codes
    - Single Characters
      'A'   'a'   'C'   '3'   '$'   '\n'   '/'   '?'
    - Strings of Characters
      "ABC"   "abc\ndef"   "32"   "-5.2"   "-1.5e-2"



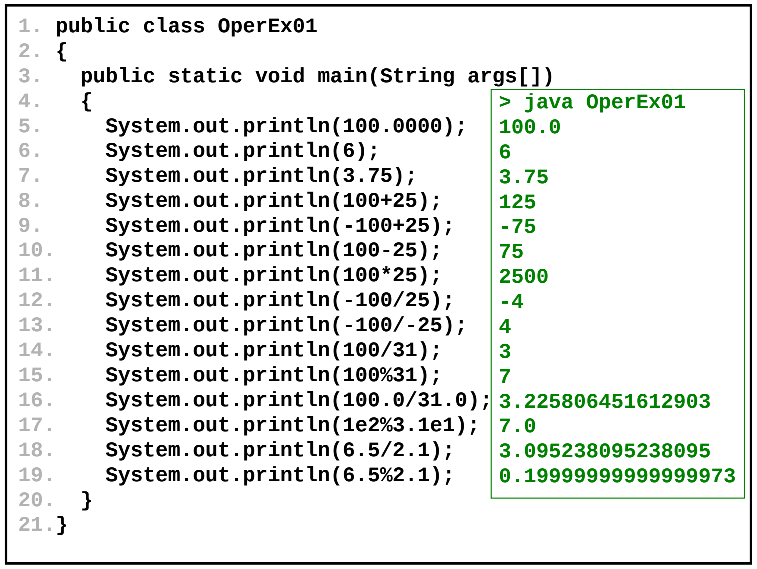
Programming Operators

- *Arithmetic Operators*
  - Perform arithmetic operations on numeric data
  - *Precedence Order* is the order the operation
  - Parenthesis ( ) have highest precedence
  - Use parenthesis if order of operation not apparent

(Precedence Highest to Lowest)

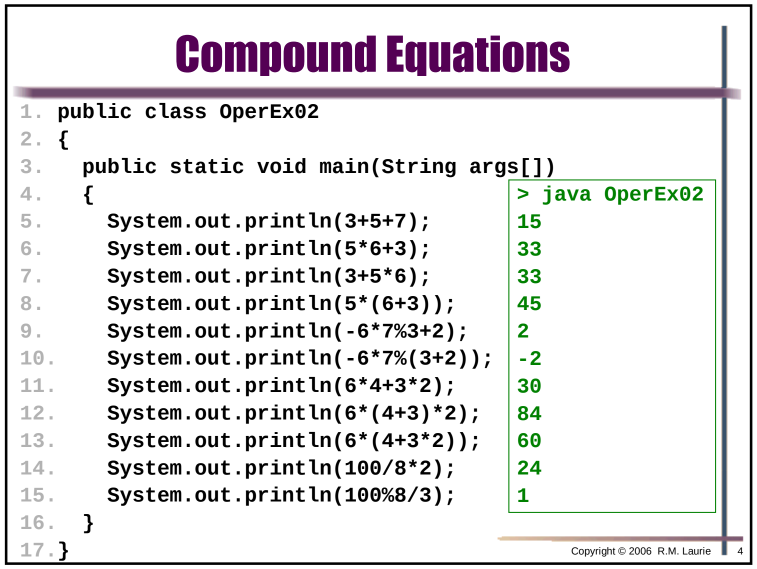
( )	Defines order of operation	→
-	Negative (unary)	←
* / %	Multiply, Division, Modulus	→
+ -	Addition, Subtraction	→

- *Concatenation Operator* +
  - *For joining Strings and Characters*
  - "Hot " + "Dog" + '\n' + "That\'s mine\n"



```
public class OperEx01
{
  public static void main(String args[])
  {
    System.out.println(100.0000);
    System.out.println(6);
    System.out.println(3.75);
    System.out.println(100+25);
    System.out.println(-100+25);
    System.out.println(100-25);
    System.out.println(100*25);
    System.out.println(-100/25);
    System.out.println(-100/-25);
    System.out.println(100/31);
    System.out.println(100%31);
    System.out.println(100.0/31.0);
    System.out.println(1e2%3.1e1);
    System.out.println(6.5/2.1);
    System.out.println(6.5%2.1);
  }
}
```

```
> java OperEx01
100.0
6
3.75
125
-75
75
2500
-4
4
3
7
3.225806451612903
7.0
3.095238095238095
0.19999999999999973
```



Compound Equations

```
public class OperEx02
{
  public static void main(String args[])
  {
    System.out.println(3+5+7);
    System.out.println(5*6+3);
    System.out.println(3+5*6);
    System.out.println(5*(6+3));
    System.out.println(-6*7%3+2);
    System.out.println(-6*7%(3+2));
    System.out.println(6*4+3*2);
    System.out.println(6*(4+3)*2);
    System.out.println(6*(4+3*2));
    System.out.println(100/8*2);
    System.out.println(100%8/3);
  }
}
```

```
> java OperEx02
15
33
33
45
2
-2
30
84
60
24
1
```

Mixed Mode Expressions

❖ *Integer Expression*
- If all numbers are integers then result is integer

❖ *Real (Floating Point) Expression*
- If any number is floating point (real) then result is floating point (real) number

❖ *String Expression*
- If any value on either side of the + operator is a string then the operator is concatenation
- You can force an arithmetic operation by enclosing the Integer or Real Expressions with Parenthesis

```
public class OperEx03
{
  public static void main(String args[])
  {
    System.out.println(20/3);
    System.out.println(20./3);
    System.out.println(3.+9/6);
    System.out.println(3+9/6.);
    System.out.println("ABC"+'D'+"EF");
    System.out.println("ABC"+'\t'+"EF");
    System.out.println("ABC"+'\"'+"EF");
    System.out.println("Product = " + 7*5);
    System.out.println("Quotient = " + 7/5.);
    System.out.println("Remainder = " + (7%5));
    System.out.println("Sum = " + 7+5);
    System.out.println("Sum = " + (7+5));
    System.out.println("Difference = " + (7-5));
    System.out.println("23 + 42 = " + 23+42);
    System.out.println("23 + 42 = " + (23+42));
  }
}
```

```
6
6.666666666667
4.0
4.5
ABCDEF
ABC    EF
ABC"EF
Product = 35
Quotient = 1.4
Remainder = 2
Sum = 75
Sum = 12
Difference = 2
23 + 42 = 2342
23 + 42 = 65
```

Character Values

❖ *ASCII*: 8-bit, Latin characters (C++ but Not Java)
- Both uppercase and lowercase letters
- Digits 0 to 9 and keyboard symbols $,#.!;@*

❖ *Unicode*: 16-bit, All Language Glyphs, Java!
- 65,536 different glyphs for all languages

❖ *Escape Characters* can be contained in string

`\"`	Double quote.
`\'`	Single quote.
`\\`	Backslash.
`\n`	New line. Go to the beginning of the next line.
`\r`	Carriage return. Go to beginning of current line.
`\t`	Tab. White space up to the next tab stop.

Escape Character Example

❖ How do you print characters with special meaning? For example, how do you print the following string?

`The word "hard"`

Would this do it?

`System.out.println( "The word "hard"" );`

No, it would give a compiler error - it sees the string `The word` between the first set of double quotes and is confused by what comes after

❖ Use the backslash character, `\"`, to escape the special meaning of the internal double quotes:

`System.out.println( "The word \"hard\"" );`