Arithmetic

Standard arithmetic, such as addition and multiplication, can be used in expressions with either variables or constants in order to carry out a more complex computation.

Both types of **numbers**, ints and doubles, have arithmetic operators: addition, subtraction, multiplication, and division all use the same operator symbol for the numeric types. The modulus operator gives the remainder for integer division.

Operator symbol	Function	Type(s)	Example expression
+	Addition	int, double	number + 1
-	Subtraction	int, double	number – 26
*	Multiplication	int, double	number * 5
/	Division	double	7.0 / 3.0 (= 2.333)
/	Division	int	7 / 3 (= 2)
%	Modulus	Integer	7 % 3 (= 1)

If there is more than one operator in an expression, then they are either evaluated in the order indicated by parentheses, but if there are no parentheses, multiplication, division, and modulus are evaluated first and finally addition and subtraction.

Other arithmetic operations are not given by single character operator such as these, but are given by functions in the Math package. For example, Math.pow computes exponents.

For **Strings**, there is an operator to catenate two strings together. For example, in the following:

String1 = “The Cat”
String2 = “in the Hat”
String3 = String1 + “ “ + String2 (Note that the middle string is a single space.)

The value of String3 is “The Cat in the Hat”.

Note that strings can have numbers in them, for example:

String1 = “Result = “
String2 = “35”
String3 = String1 + String 2

The value of String3 is “Result = 35”.

Boolean operators will be described in the section on conditional statements.

Arithmetic combined with Assignment

It is quite common to perform assignments in which the right hand side is an expression using the same variable.

Examples:

```
int num = 0;
num = num + 2;
num = num * 5;
```

In these cases, a combined operator can be used; the following statements would be equivalent to those above.

```
int num = 0;
num += 2;
num *= 5;
```

In the case that you are adding or subtracting the number 1, an even shorter abbreviation is used:

```
int num = 0;
num = num + 1;
```

is equivalent to:

```
int num = 0;
num ++;
```