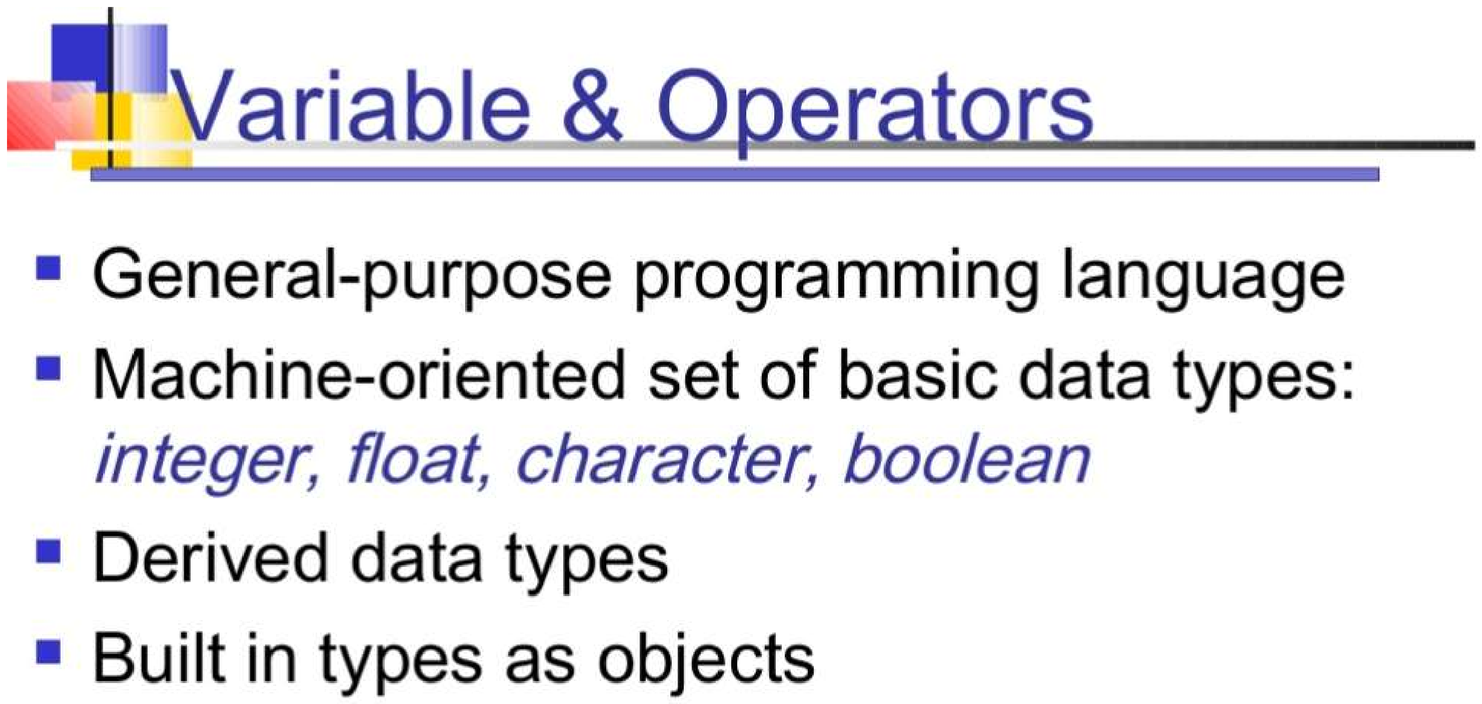
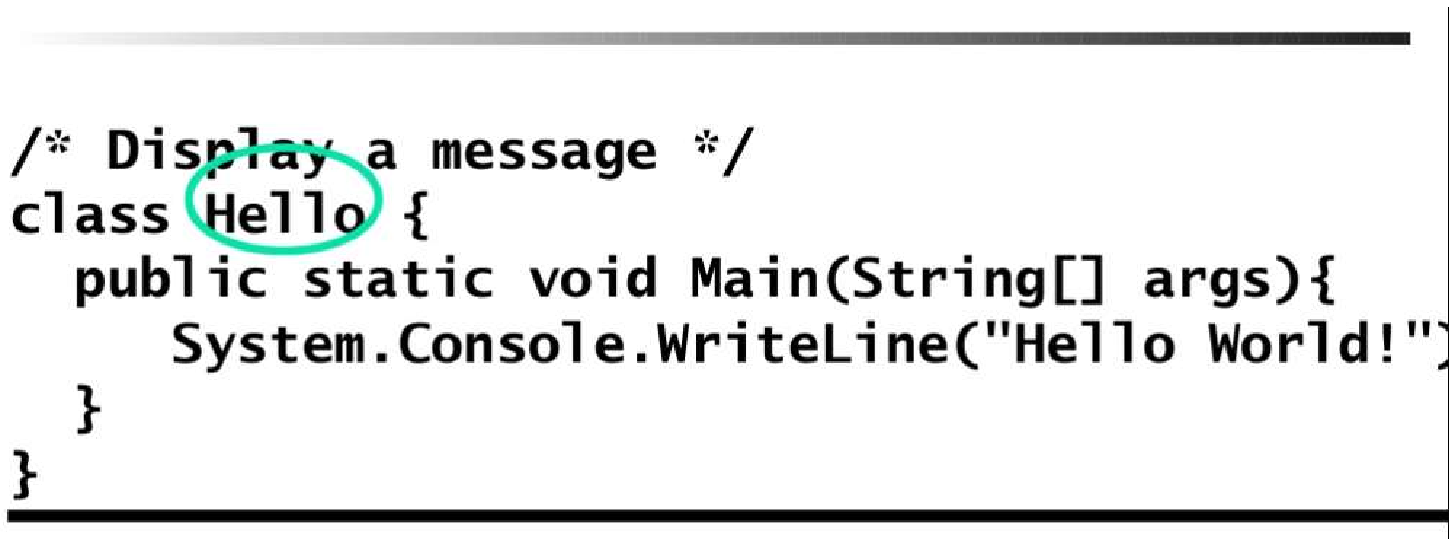
Operators

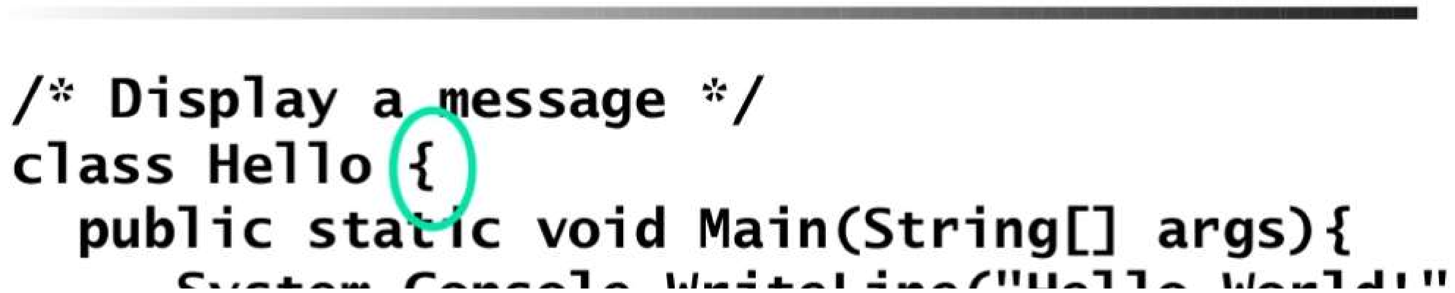


Variable & Operators

- General-purpose programming language
- Machine-oriented set of basic data types: *integer, float, character, boolean*
- Derived data types
- Built in types as objects



```
/* Display a message */
class Hello {
  public static void Main(String[] args){
     System.Console.WriteLine("Hello World!")
  }
}
```



```
/* Display a message */
class Hello {
  public static void Main(String[] args){
     System.Console.WriteLine("Hello World!"
```

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now

```
}
```

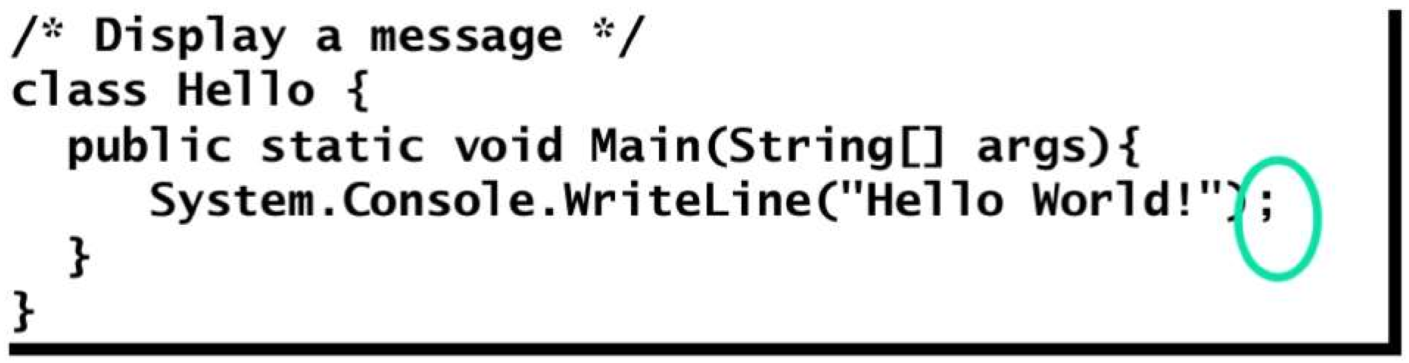
- C# program consists of a named class.
- The body of the class is surrounded by braces

```
/* Display a message */
class Hello {
  public static void Main(String[] args){
     System.Console.WriteLine("Hello
World!");
  }
}
```

- (Almost) every C# program must have one and only one `Main()` function.
- The body of the function is surrounded by braces

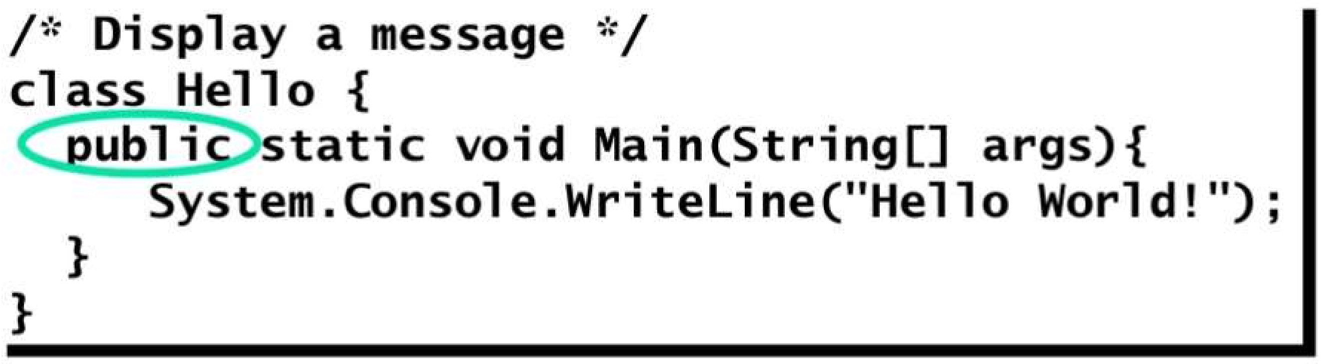
```
/* Display a message */
class Hello {
  public static void Main(String[] args){
     System.Console.WriteLine("Hello World
  }
}
```

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now



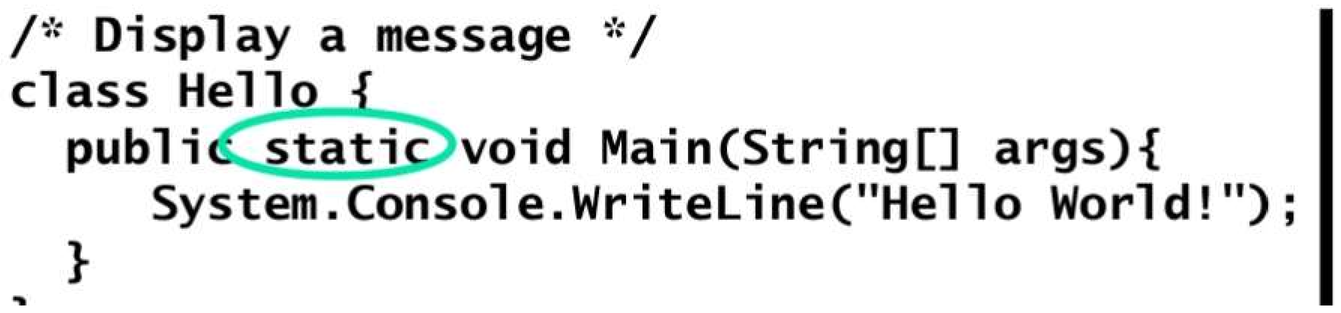
```
/* Display a message */
class Hello {
  public static void Main(String[] args){
     System.Console.WriteLine("Hello World!");
  }
}
```

- A semicolon is a statement *terminator*.



```
/* Display a message */
class Hello {
  public static void Main(String[] args){
     System.Console.WriteLine("Hello World!");
  }
}
```

- **public** indicates that this function can be called by objects outside of the class



```
/* Display a message */
class Hello {
  public static void Main(String[] args){
     System.Console.WriteLine("Hello World!");
  }
```

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

Take survey Not now

- **static** indicates that this function remains in memory throughout the execution of the application

```
/* Display a message */
class Hello {
  public static void Main(String[] args){
     System.Console.WriteLine("Hello World!")
  }
}
```

- **void** indicates that this function does not return a value to the object that calls it

```
/* Display a message */
class Hello {
  public static void Main(String[] args){
     System.Console.WriteLine("Hello World!");
  }
}
```

- **args** can be used in the *Main* function to pass parameters from the operating system command line

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

Take survey Not now

```
/* Display a message */
class Hello {
  public static void Main(String[] args){
     System.Console.WriteLine("Hello World!");
  }
}
```

- Comments are the most important part of your program
- Criteria for good comments

Rules...

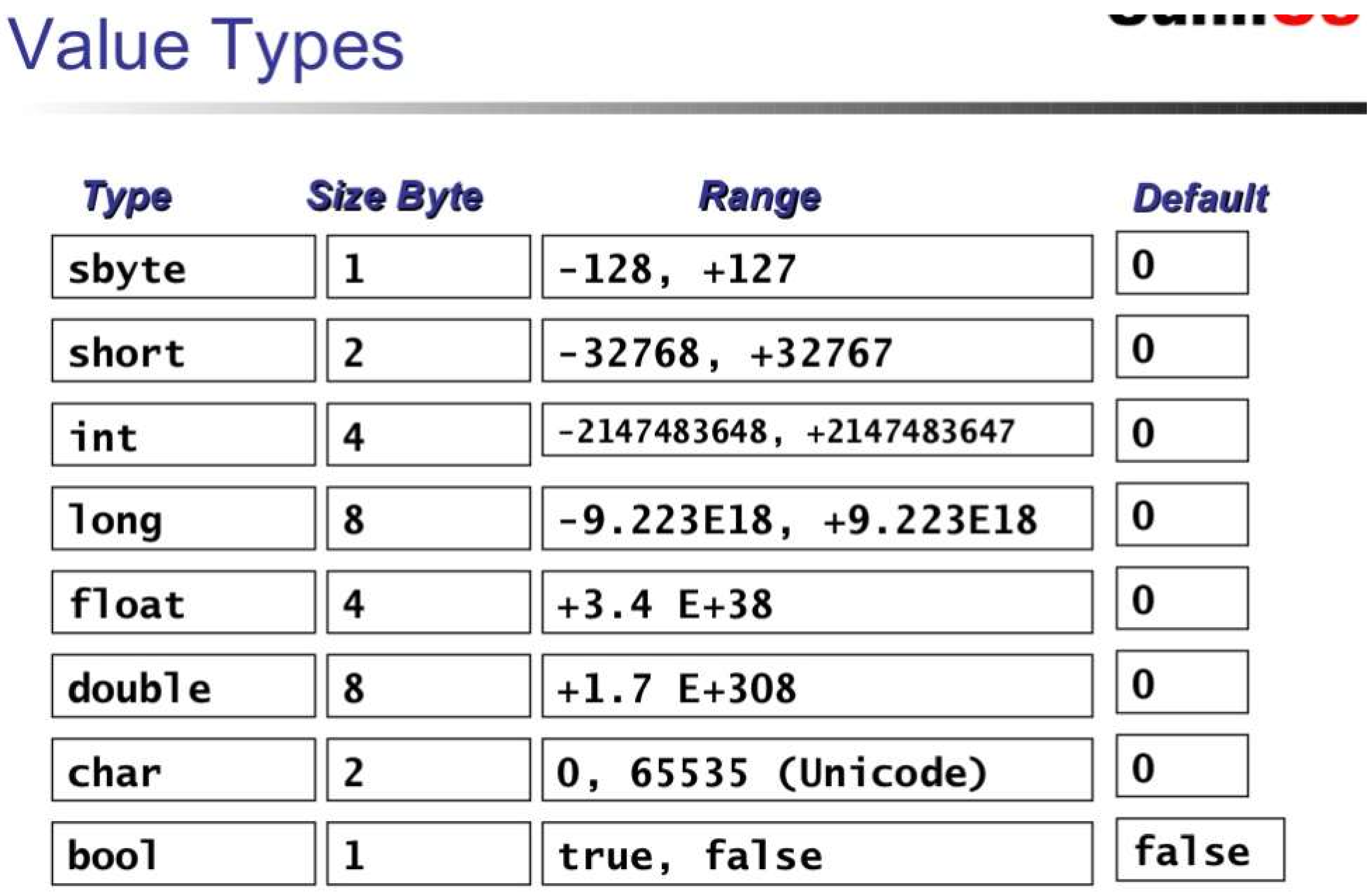
- The first character must be any non-digit from the Unicode standard
  - String **FirstName**;
- Subsequent characters may include digits
  - **int** total123
- Case is significant i. e. C# is case sensitive
  - **int** count =0 ; Count = 1 are two different variables
- Avoid using underscore and $ for the first character

- User-defined identifiers can not duplicate keywords
- Examples

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

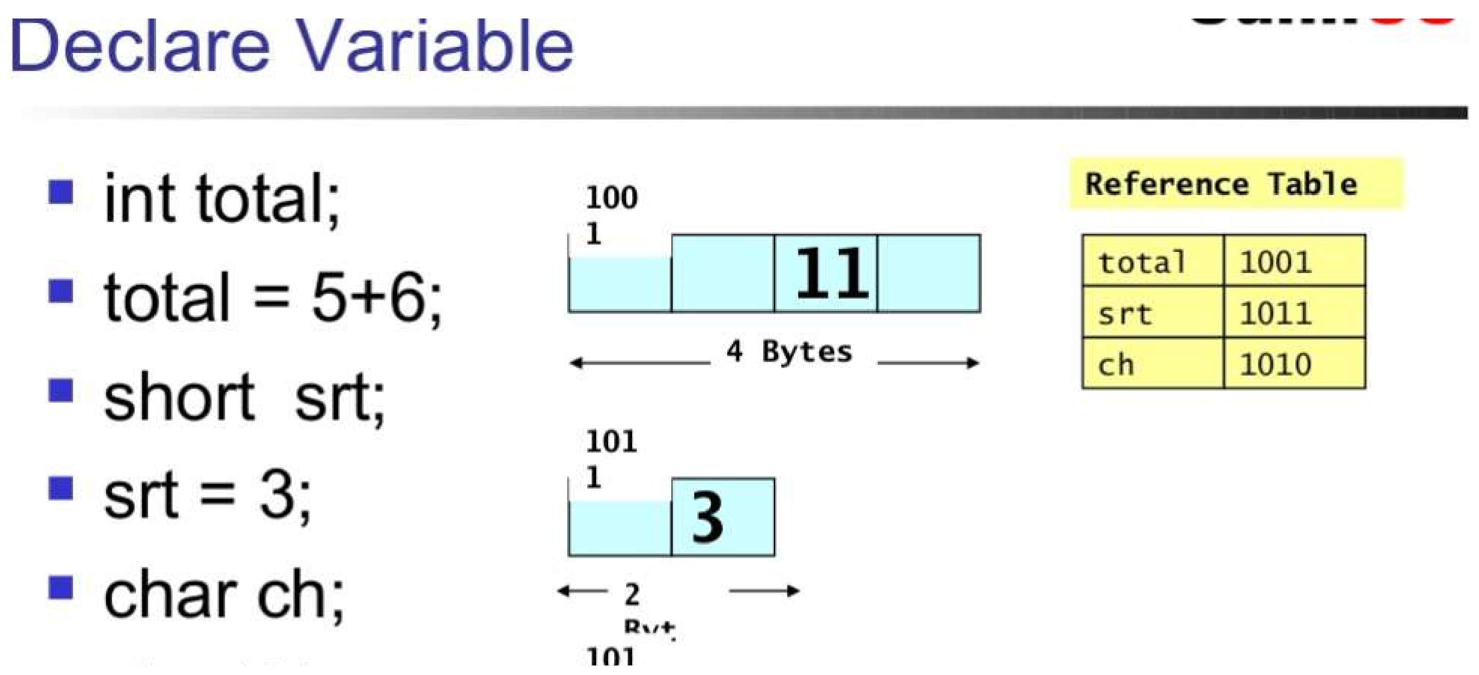
Take survey Not now

- **TotalCost**



Value Types

Type	Size Byte	Range	Default
sbyte	1	-128, +127	0
short	2	-32768, +32767	0
int	4	-2147483648, +2147483647	0
long	8	-9.223E18, +9.223E18	0
float	4	+3.4 E+38	0
double	8	+1.7 E+308	0
char	2	0, 65535 (Unicode)	0
bool	1	true, false	false



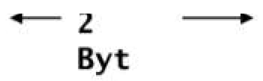
Declare Variable

- int total;
- total = 5+6;
- short srt;
- srt = 3;
- char ch;

Reference Table

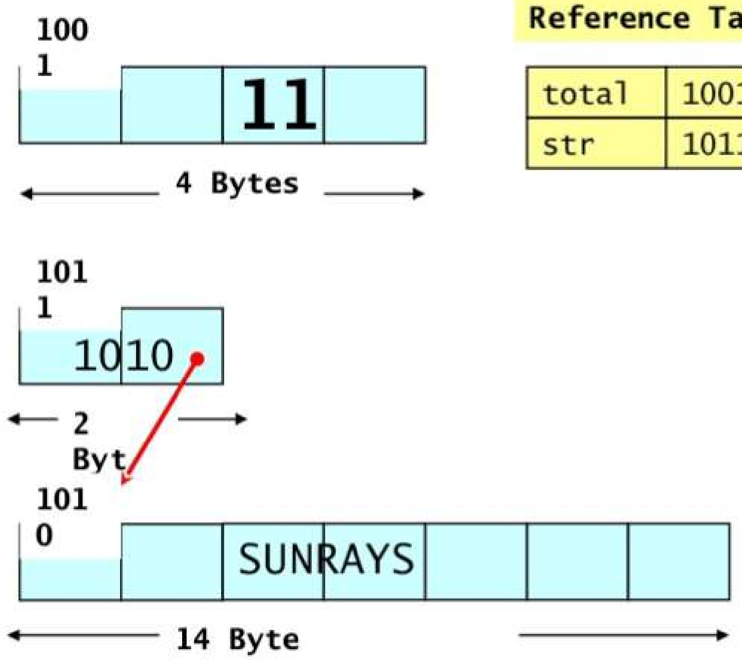
total	1001
srt	1011
ch	1010

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?
Take survey Not now



Declare Variable

- int total;
- total = 5+6;
- String str;
- str = “sunRays”
- Or
- str = new String(“sunRays”)

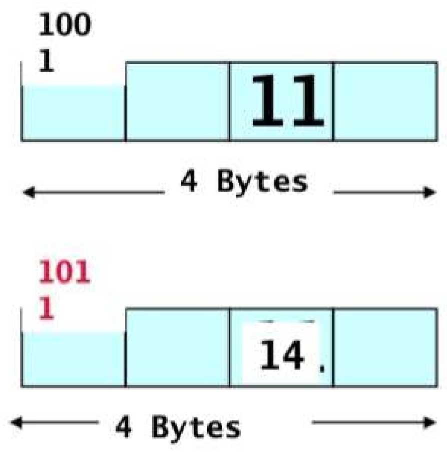


Reference Table

total	1001
str	1011

Declare Variable

- int total;
- total = 5+6;
- int newTotal;
- newTotal = total
- newTotal = newTotal+3

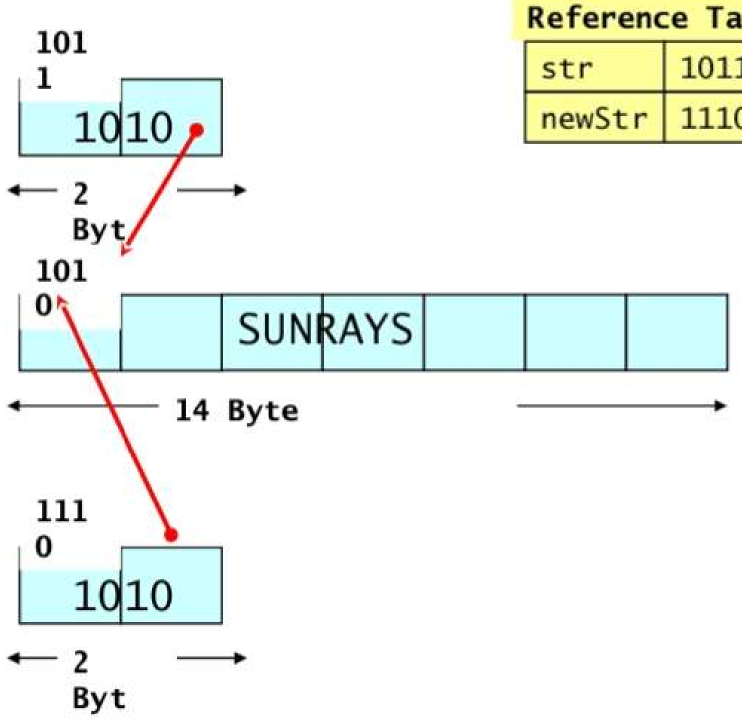


Reference Table

total	1001
newTotal	1011

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now

- String str;
- str = “sunRays”
- Or
- str = new String(“sunRays”)
- String newStr;
- newStr = str;



Reference Table	
str	1011
newStr	1110

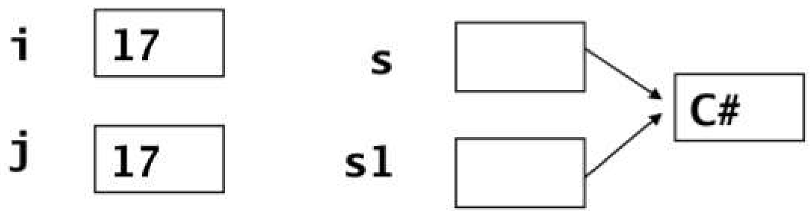
- Value Types
  - byte, short, int, long, float, double, boolean, char
- Reference Data Types
  - String
  - Object
  - Arrays

	Value Types	Reference Types
Variable Contains	Value	Reference

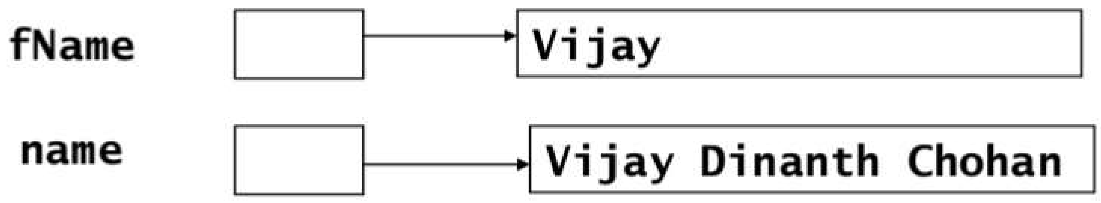
We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?
Take survey Not now

Assignment Copies	Values	Reference

Example	int i = 17;
	int j = i;
		Sting s = “C#”;
		String s1=s;



- String is immutable
- String fName = “Vijay”;
- String name = fName + “Dinanath Chohan”;



- StringBuffer is immutable

- Operators are *tokens* that trigger some computation when applied to variables and other objects.
- *Arithmetic*, *logical*, and *bit-level* operators.

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?
Take survey Not now

()	/	<	^
++	%	>	|
--	+	<=	&&
!	-	>=	||
~	<<	==	?:
is	>>	!=	=
*		&	op=

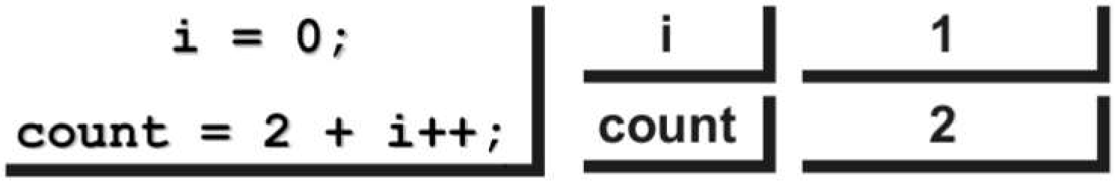
- Operators have the precedence. Higher precedence operator will be evaluated before than lower precedence operator.
- Eg. data = a * b + c
- since * (multiply) has higher precedence than + (plus) so a & b will be multiplied first then result will added to c.
- Or (a*b) +c

() Group expression

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now

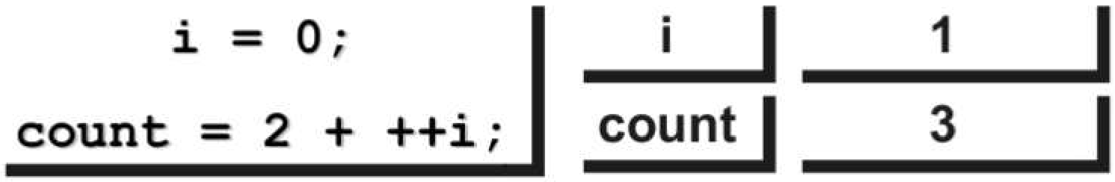
+	Unary plus
-	Unary minus

~	Bitwise complement
!	Logical negation
++	Pre- or Post-increment
--	Pre- or Post-decrement



```
i = 0;
count = 2 + i++;
```

i	1
count	2

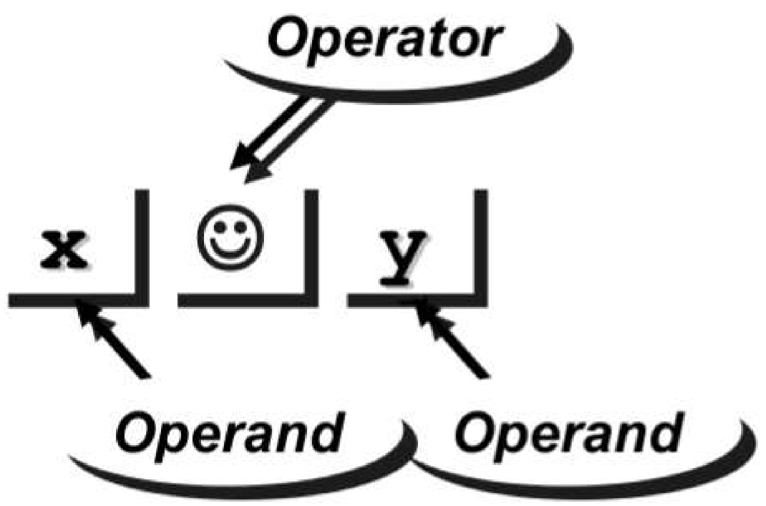


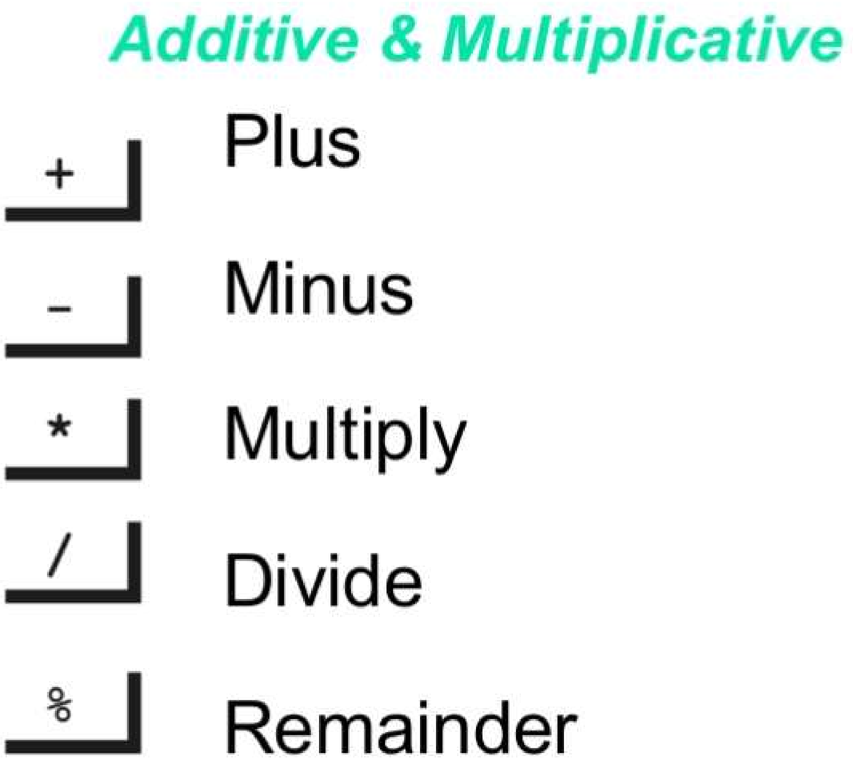
```
i = 0;
count = 2 + ++i;
```

i	1
count	3

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

Take survey Not now





Additive & Multiplicative

+	Plus
-	Minus
*	Multiply
/	Divide
%	Remainder

= Assignment

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now

- The left-hand operand of an assignment must be an LVALUE

= Assignment

- An LVALUE is an expression that refers to a region of memory
  - Names of variables are LVALUES
  - Names of functions and arrays are not LVALUES

```
class ExampleAssignment {
    public static void Main(String[] args) {
            int result, val_1, val_2;
            result = (val_1 = 1) + (val_2 = 2);
            System.Console.WriteLine("val_1 = "+val_1);
            System.Console.WriteLine("val_2 = "+val_2);
            System.Console.WriteLine("result = "+result);
      }
}
```

```
val_1 = 1
val_2 = 2
result = 3
```

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

Take survey Not now

- Variable stores human data like numbers and alphabets.
- Data type will decide what values will be stored in variables.
- You can say data type will define the structure of your data.

Variables and Data Types

- Decimal values will be stored in float and double data type.
- Non-decimals values will be stored in int, long, byte, and short data types.
- Character will be stored in char data type.
- True/False will be stored in boolean data type.

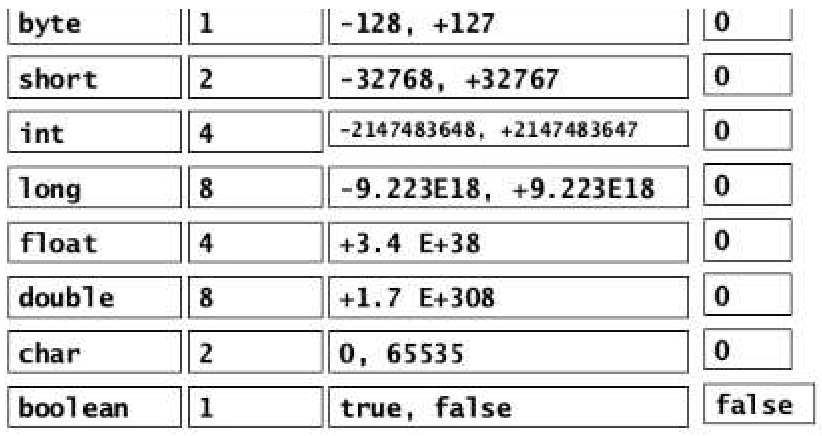
Data Types

Data types are divided into two categories.

- Primitive Data Types
  - `byte, short, int, long, float, double, boolean, char.`
  - It occupies number of bytes as per data type.
  - It stores values.
- Reference Data Types
  - It stores memory address of a value.
  - It occupies 2 bytes to store a reference (memory address).
  - Strings, Objects, Arrays are reference data types.

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

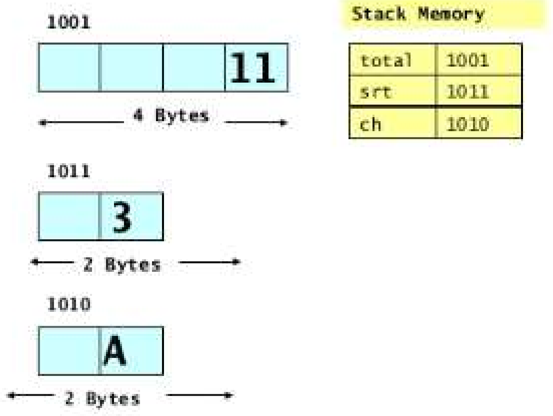
Take survey Not now



byte	1	-128, +127	0
short	2	-32768, +32767	0
int	4	-2147483648, +2147483647	0
long	8	-9.223E18, +9.223E18	0
float	4	+3.4 E+38	0
double	8	+1.7 E+308	0
char	2	0, 65535	0
boolean	1	true, false	false

Declare Variable

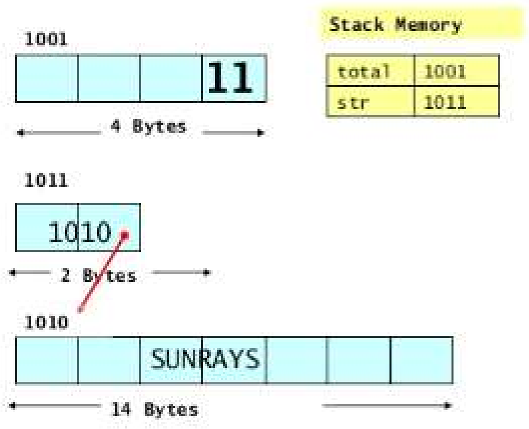
- int total;
- total = 5+6;
- short srt;
- srt = 3;
- char ch;
- ch ='A' ;



total	1001
srt	1011
ch	1010

Declare Object

- int total;
- total = 5+6;
- String str;
- str = "sunRays";
  - or
- str = new String("sunRays");



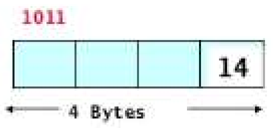
total	1001
str	1011

Declare Variable – Primitive Data

- int total;

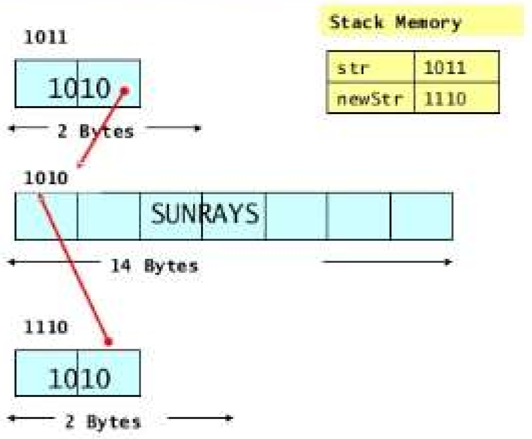
We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now

- newTotal = total ;
- newTotal = newTotal+3 ;



Declare Object - Copy reference

- String str;
- str = "sunRays"
- Or
- str = new String("sunRays")
- String newStr;
- newStr = str;



Java Identifier

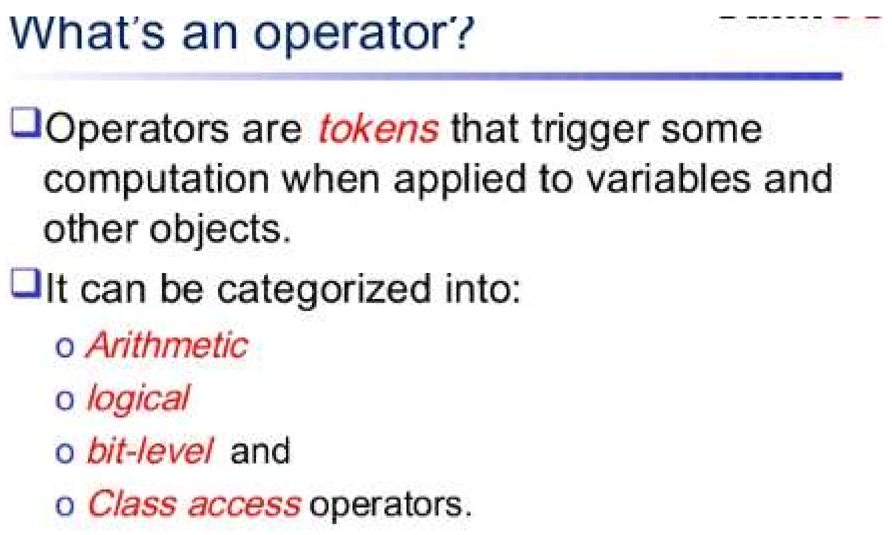
- It is a name of:
  - Variable
  - Method
  - Class
  - Interface
  - Package
- Used to identify a variable, method and class in its scope.

Java Identifier Rules

- Name of an Identifier follows certain rules. Here are key rules:
  - The first character must be a non-digit character from the Unicode standard `String` **`firstName`**;

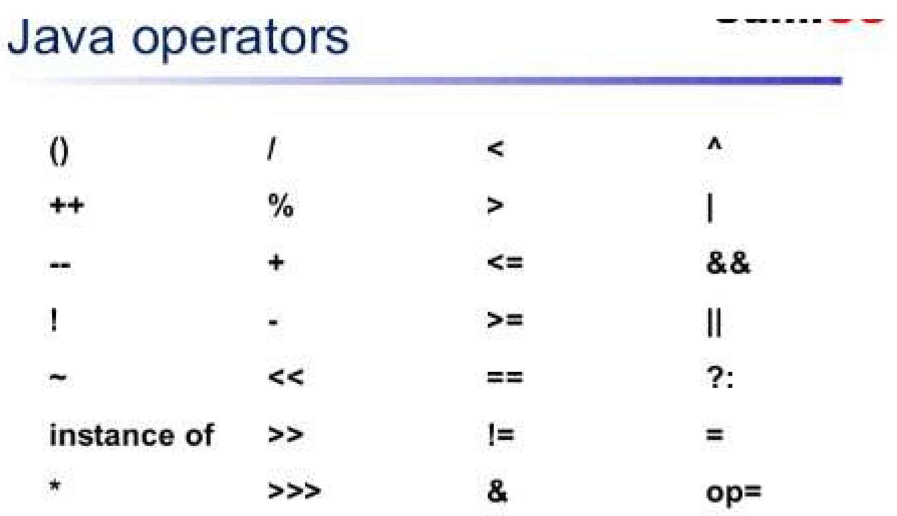
We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now

- o Avoid using underscore (_) and $ for the first character.
- o User-defined identifiers can not duplicate Java keywords.



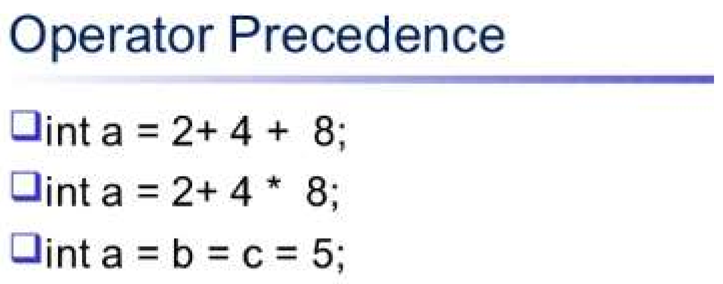
What's an operator?

- ❑Operators are *tokens* that trigger some computation when applied to variables and other objects.
- ❑It can be categorized into:
  - o *Arithmetic*
  - o *logical*
  - o *bit-level* and
  - o *Class access* operators.



Java operators

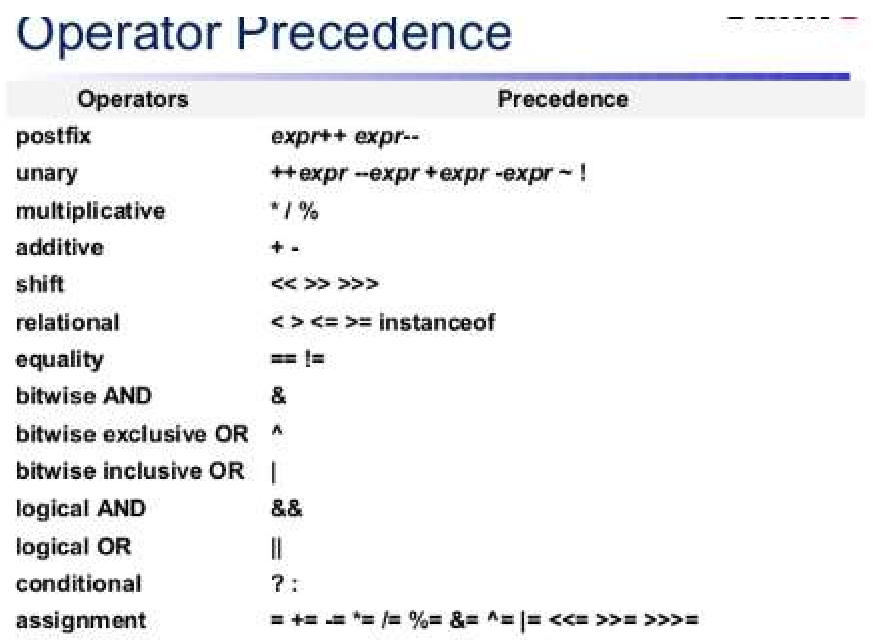
()	/	<	^
++	%	>	|
--	+	<=	&&
!	-	>=	||
~	<<	==	?:
instance of	>>	!=	=
*	>>>	&	op=



Operator Precedence

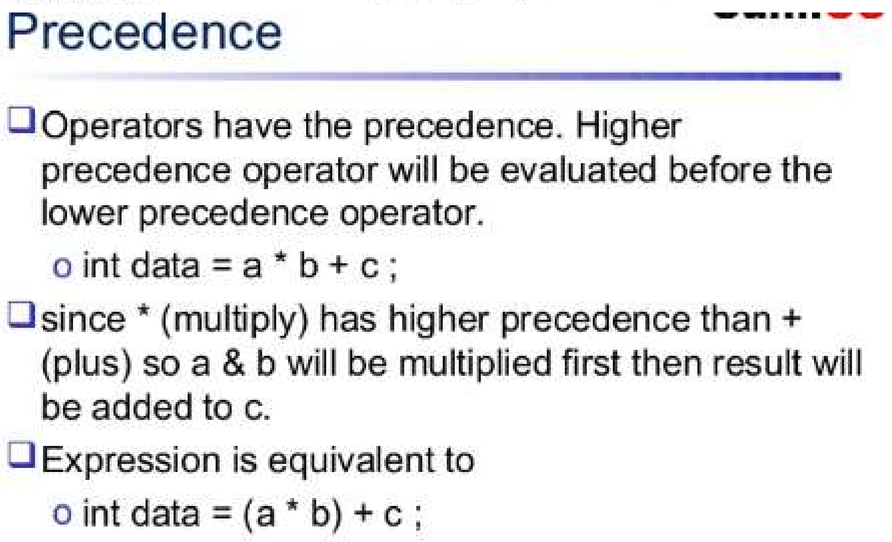
- ❑int a = 2+ 4 + 8;
- ❑int a = 2+ 4 * 8;
- ❑int a = b = c = 5;

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now



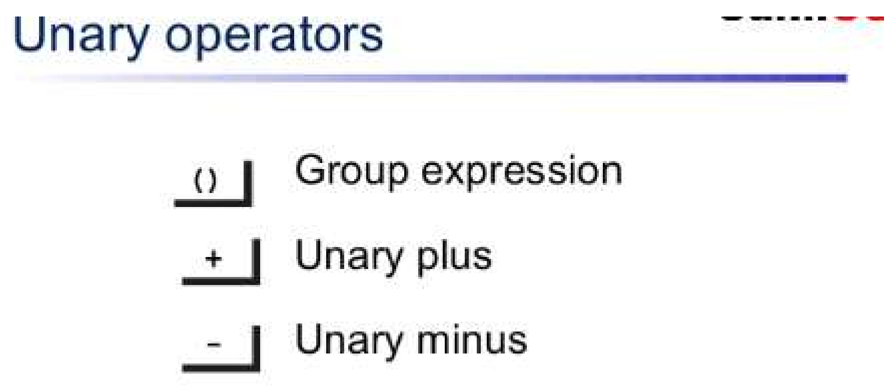
Operator Precedence

Operators	Precedence
postfix	*expr++ expr--*
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
conditional	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=



Precedence

- Operators have the precedence. Higher precedence operator will be evaluated before the lower precedence operator.
  - int data = a * b + c ;
- since * (multiply) has higher precedence than + (plus) so a & b will be multiplied first then result will be added to c.
- Expression is equivalent to
  - int data = (a * b) + c ;

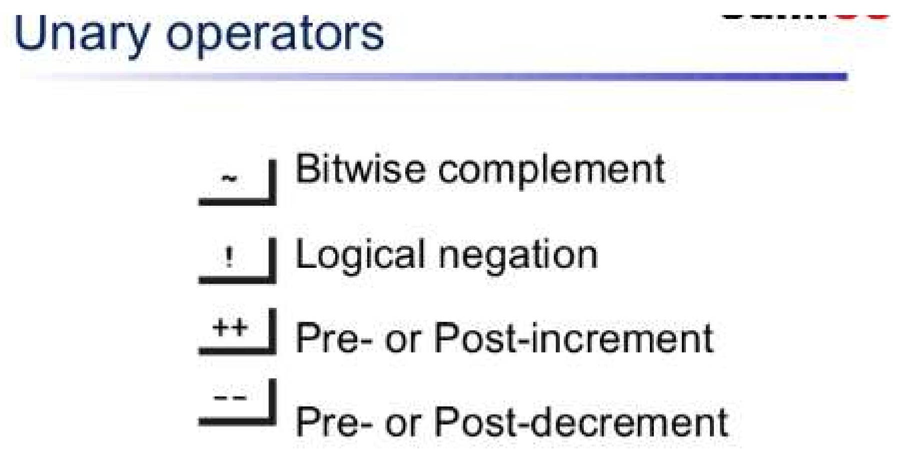


Unary operators

( )	Group expression
+	Unary plus
-	Unary minus

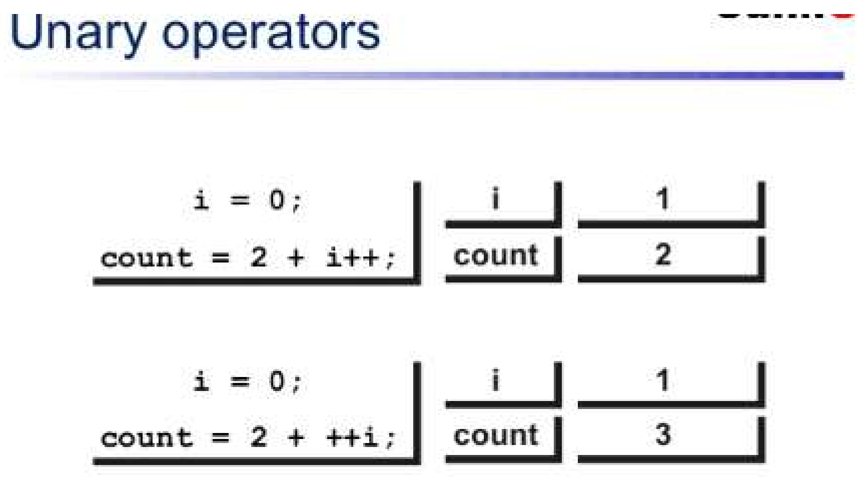
We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

Take survey Not now



Unary operators

`~`	Bitwise complement
`!`	Logical negation
`++`	Pre- or Post-increment
`--`	Pre- or Post-decrement

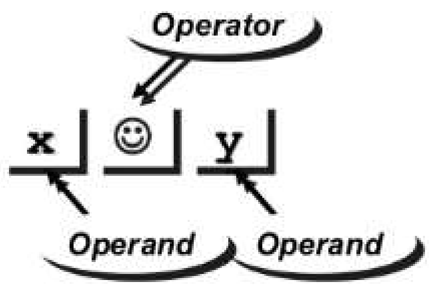


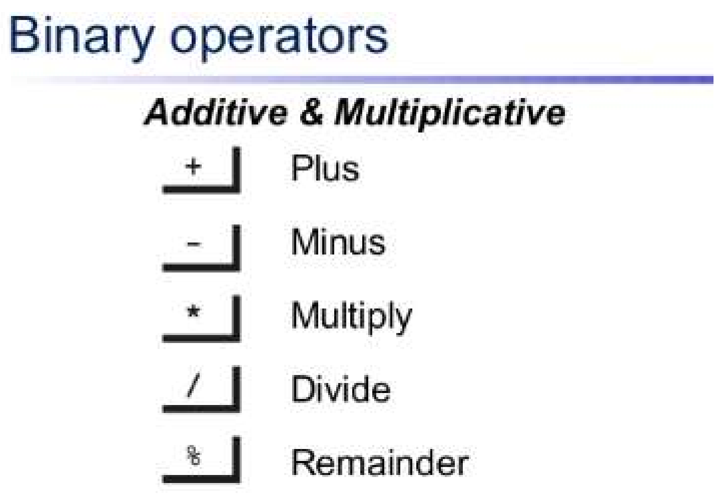
Unary operators

`i = 0;`	i	1
`count = 2 + i++;`	count	2

`i = 0;`	i	1
`count = 2 + ++i;`	count	3

Binary operators





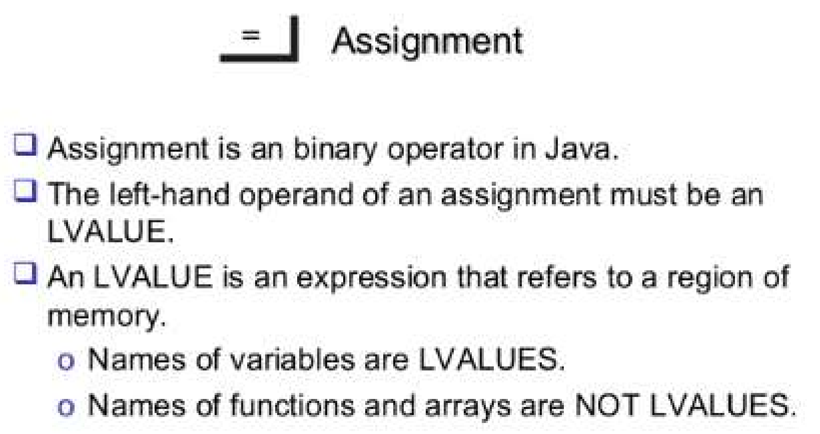
Binary operators

Additive & Multiplicative

`+`	Plus
`-`	Minus
`*`	Multiply
`/`	Divide
`%`	Remainder

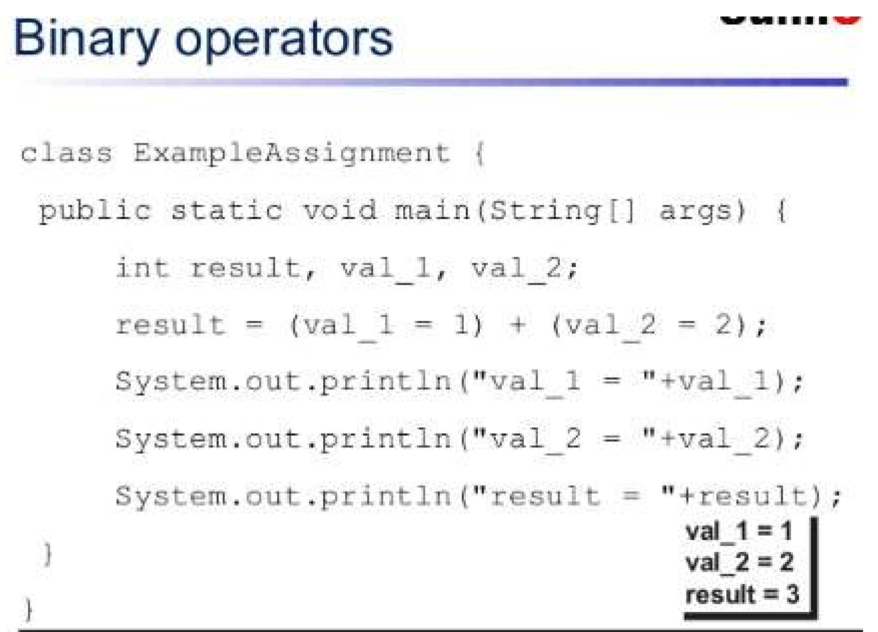
We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

Take survey Not now



= Assignment

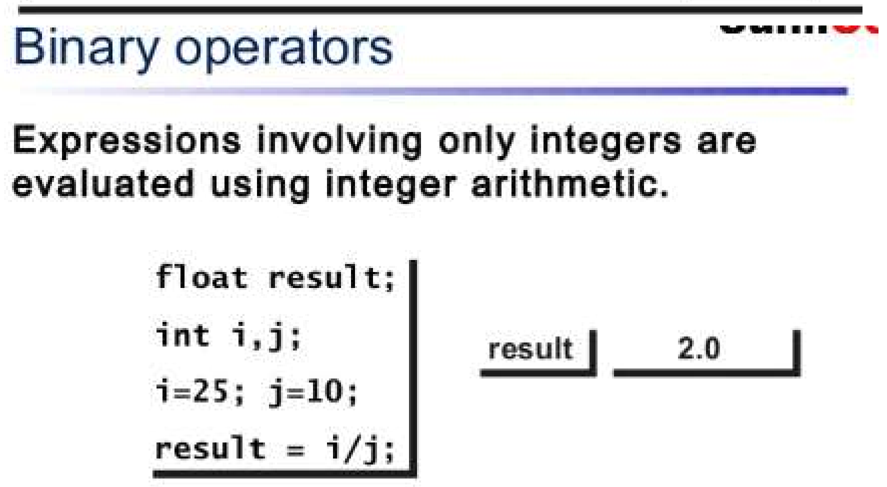
- Assignment is an binary operator in Java.
- The left-hand operand of an assignment must be an LVALUE.
- An LVALUE is an expression that refers to a region of memory.
  - Names of variables are LVALUES.
  - Names of functions and arrays are NOT LVALUES.



Binary operators

```
class ExampleAssignment {
 public static void main(String[] args) {
     int result, val_1, val_2;
     result = (val_1 = 1) + (val_2 = 2);
     System.out.println("val_1 = "+val_1);
     System.out.println("val_2 = "+val_2);
     System.out.println("result = "+result);
 }
}
```

```
val_1 = 1
val_2 = 2
result = 3
```



Binary operators

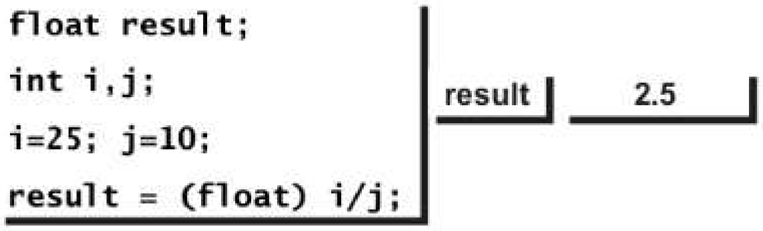
Expressions involving only integers are evaluated using integer arithmetic.

```
float result;
int i,j;
i=25; j=10;
result = i/j;
```

result 2.0

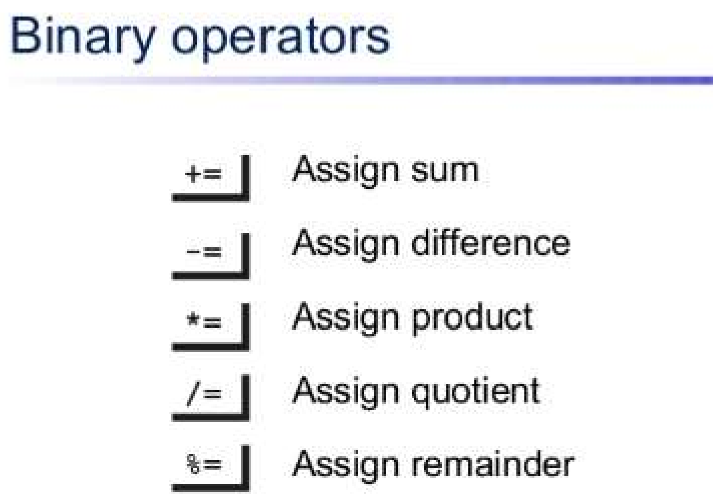
Binary operators

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now



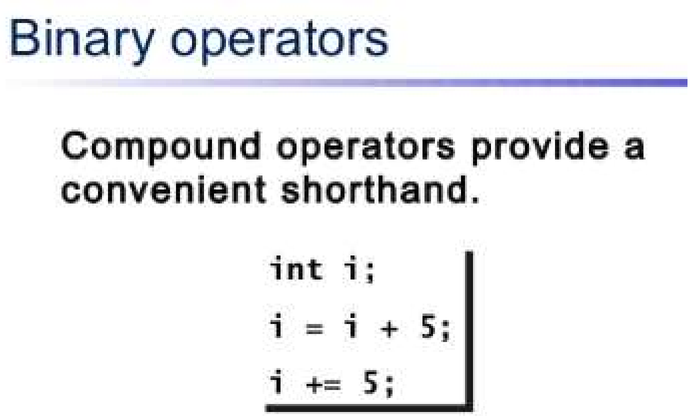
```
float result;
int i,j;
i=25; j=10;
result = (float) i/j;
```

result	2.5



Binary operators

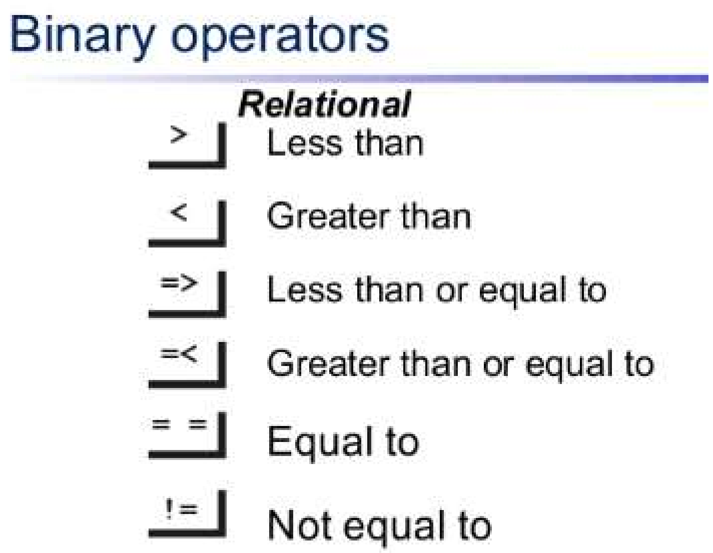
+=	Assign sum
-=	Assign difference
*=	Assign product
/=	Assign quotient
%=	Assign remainder



Binary operators

Compound operators provide a convenient shorthand.

```
int i;
i = i + 5;
i += 5;
```



Binary operators

Relational

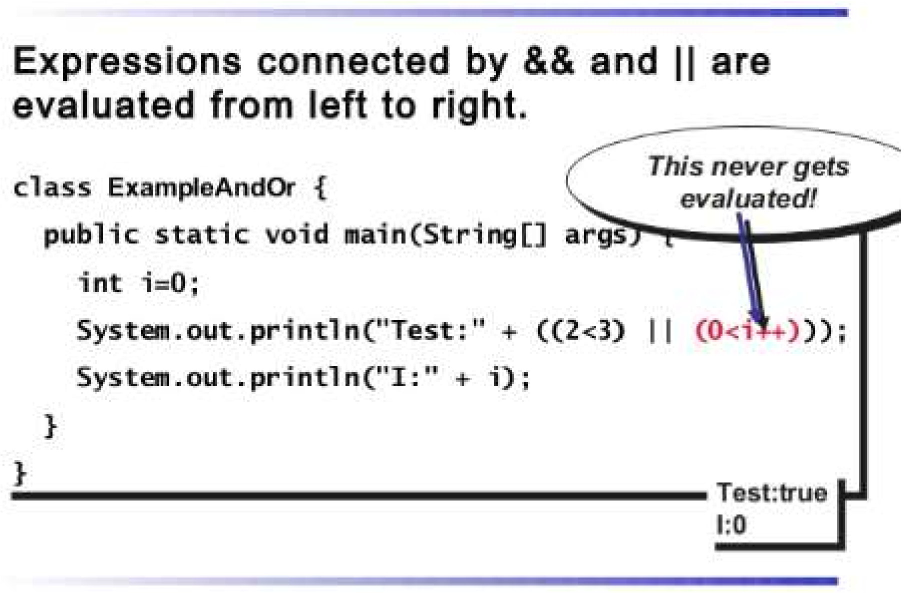
>	Less than
<	Greater than
=>	Less than or equal to
=<	Greater than or equal to
= =	Equal to
!=	Not equal to

Logical

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

Take survey Not now

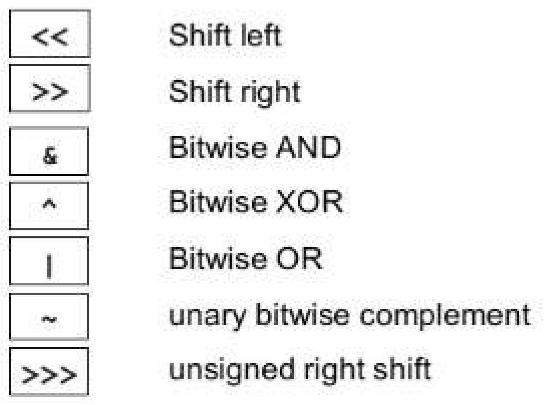
Expressions connected by && and || are evaluated from left to right.



Expressions connected by && and || are evaluated from left to right.

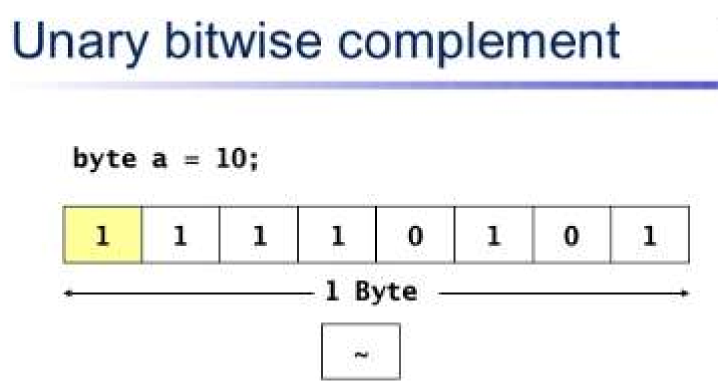
```
class ExampleAndOr {
  public static void main(String[] args) {
    int i=0;
    System.out.println("Test:" + ((2<3) || (0<i++)));
    System.out.println("I:" + i);
  }
}
```

```
Test:true
I:0
```



<<	Shift left
>>	Shift right
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
~	unary bitwise complement
>>>	unsigned right shift

These operators are less commonly used.

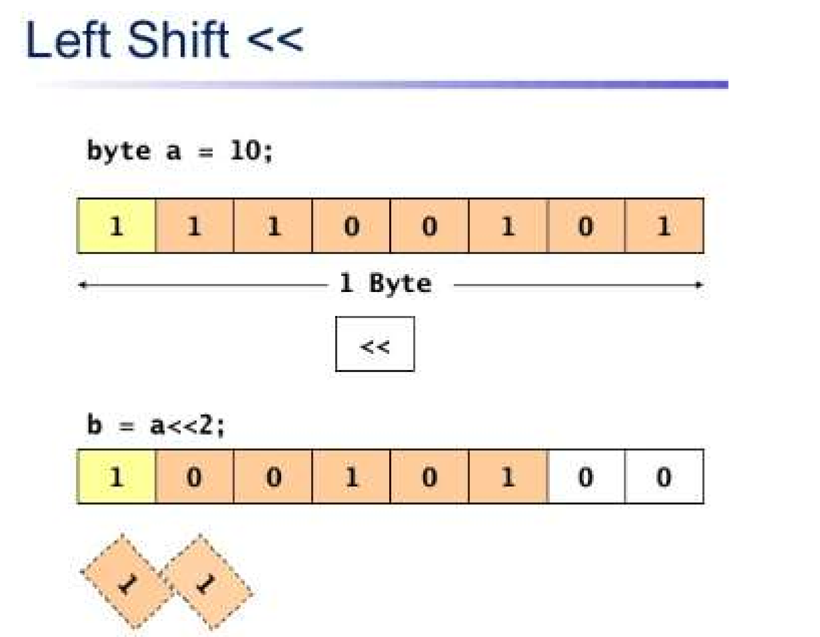


Unary bitwise complement

```
byte a = 10;
```

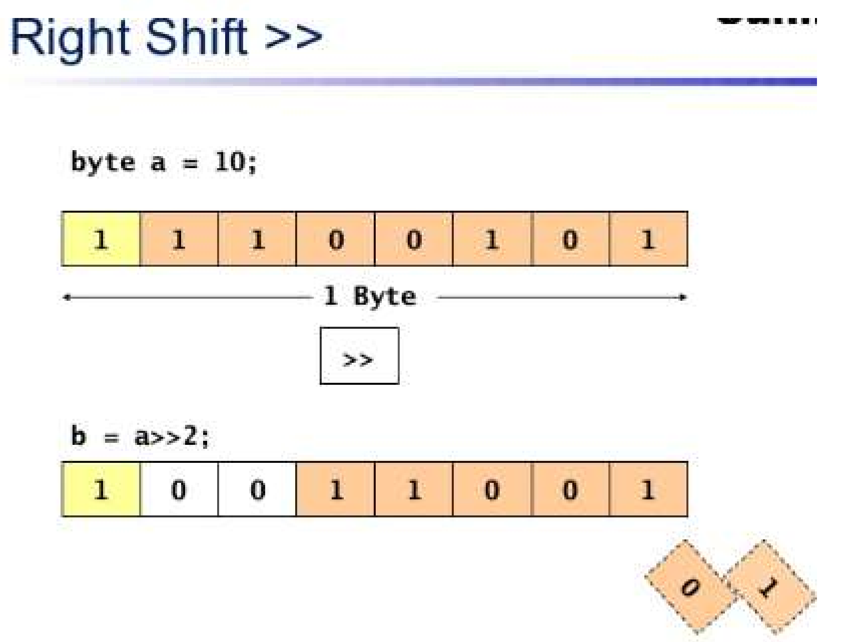
1	1	1	1	0	1	0	1

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now


Left Shift <<

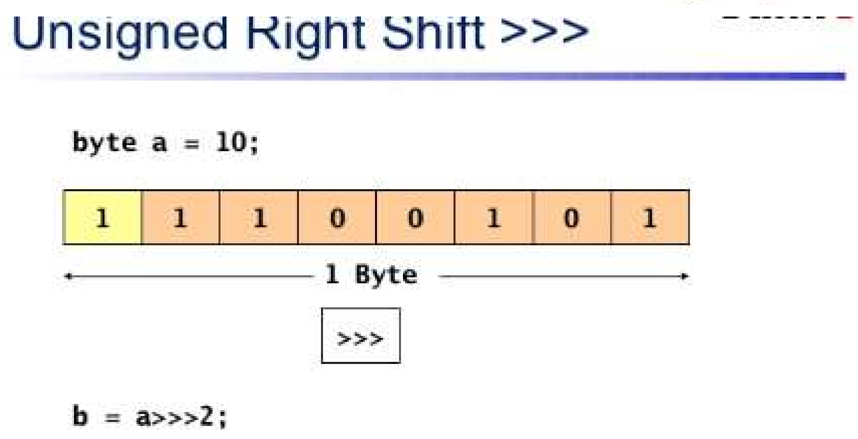
```
byte a = 10;
```

```
b = a<<2;
```


Right Shift >>

```
byte a = 10;
```

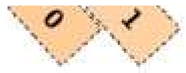
```
b = a>>2;
```


Unsigned Right Shift >>>

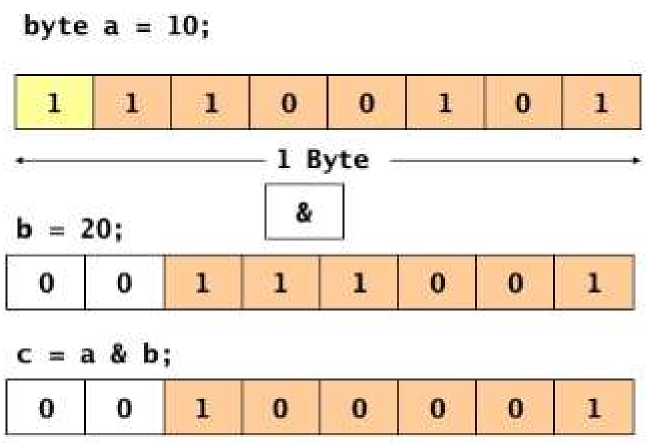
```
byte a = 10;
```

```
b = a>>>2;
```

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now



And bitwise &



byte a = 10;

1	1	1	0	0	1	0	1

1 Byte

&

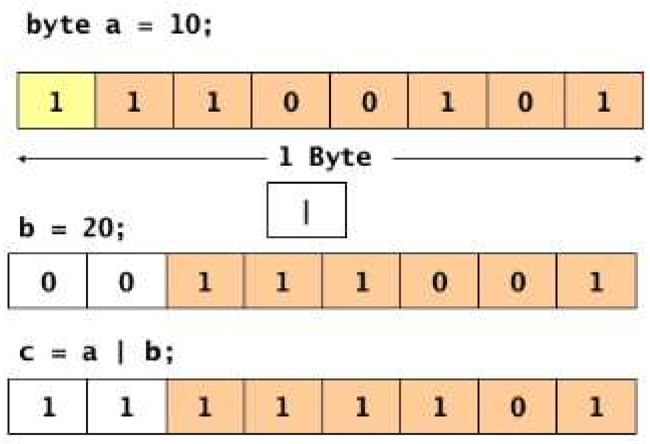
b = 20;

0	0	1	1	1	0	0	1

c = a & b;

0	0	1	0	0	0	0	1

OR bitwise |



byte a = 10;

1	1	1	0	0	1	0	1

1 Byte

|

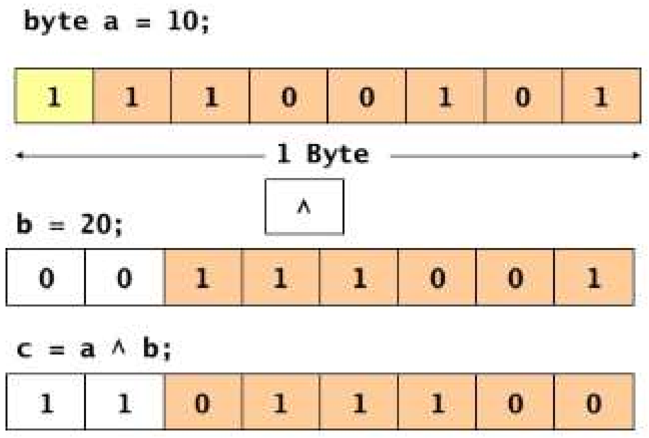
b = 20;

0	0	1	1	1	0	0	1

c = a | b;

1	1	1	1	1	1	0	1

XOR bitwise ^



byte a = 10;

1	1	1	0	0	1	0	1

1 Byte

^

b = 20;

0	0	1	1	1	0	0	1

c = a ^ b;

1	1	0	1	1	1	0	0

Ternary operators

Conditional

"if *a* then *x*, else *y*"

```
a?x:y
```

```
result = (x<y) ? x : y;
```

Multiple Assignments

- int a = b = c = 10;

Exercise

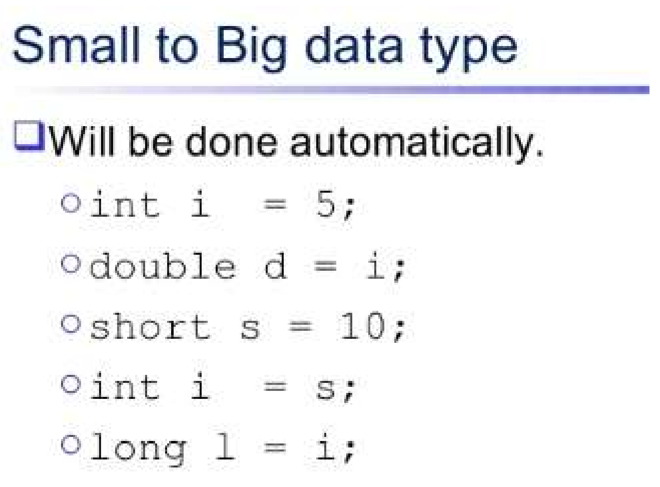
- What is the result of

```
int i = 0 ;
System.out.println(++i + ++i + ++i + ++i + ++i + ++i);
System.out.println("" + ++i + ++i + ++i + ++i + ++i + ++i);
```

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

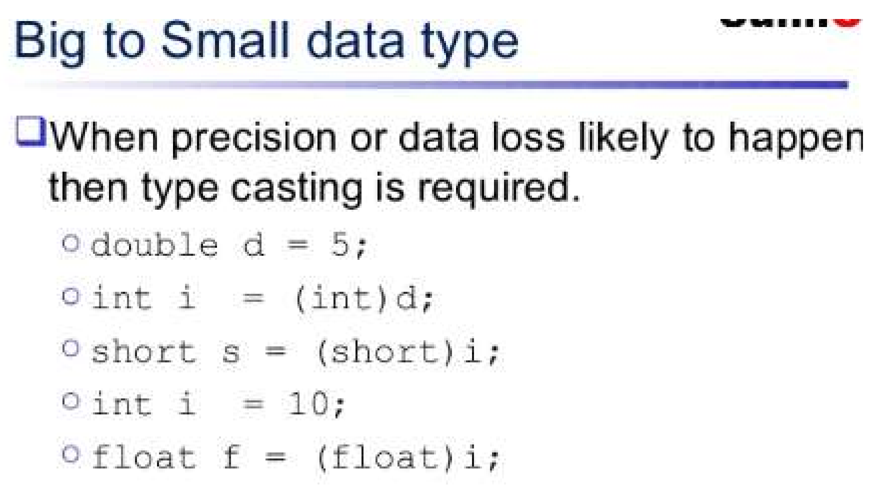
Take survey Not now

Type Conversion



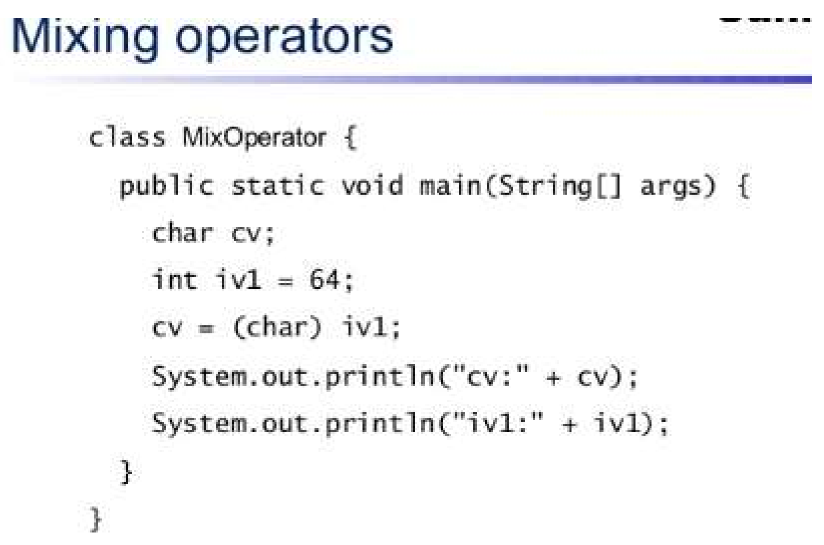
Small to Big data type

- Will be done automatically.
  - `int i   = 5;`
  - `double d = i;`
  - `short s = 10;`
  - `int i   = s;`
  - `long l = i;`



Big to Small data type

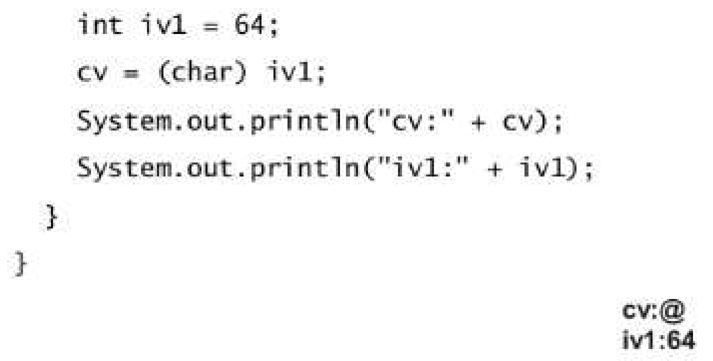
- When precision or data loss likely to happen then type casting is required.
  - `double d = 5;`
  - `int i   = (int)d;`
  - `short s = (short)i;`
  - `int i   = 10;`
  - `float f = (float)i;`



Mixing operators

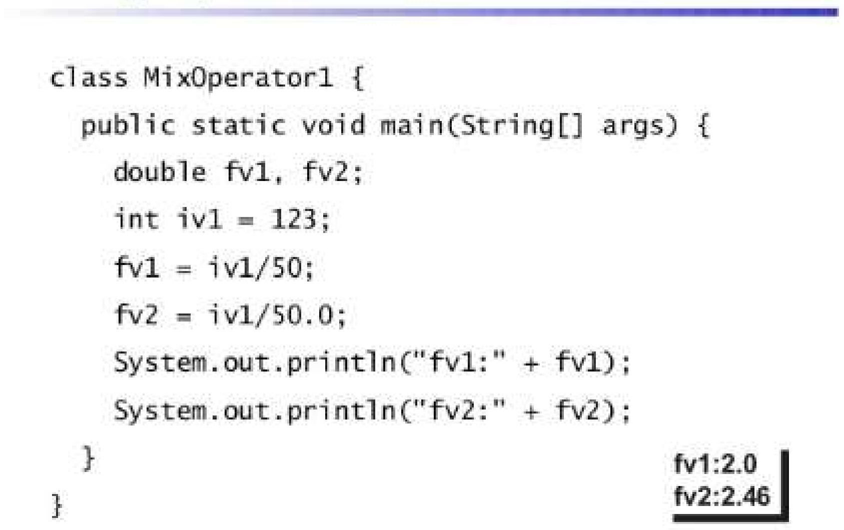
```
class MixOperator {
  public static void main(String[] args) {
    char cv;
    int iv1 = 64;
    cv = (char) iv1;
    System.out.println("cv:" + cv);
    System.out.println("iv1:" + iv1);
  }
}
```

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing? Take survey Not now



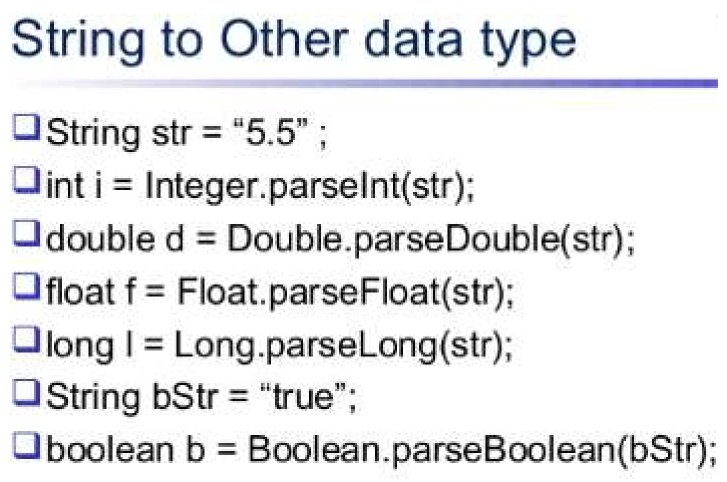
```
    int iv1 = 64;
    cv = (char) iv1;
    System.out.println("cv:" + cv);
    System.out.println("iv1:" + iv1);
  }
}
```

```
cv:@
iv1:64
```



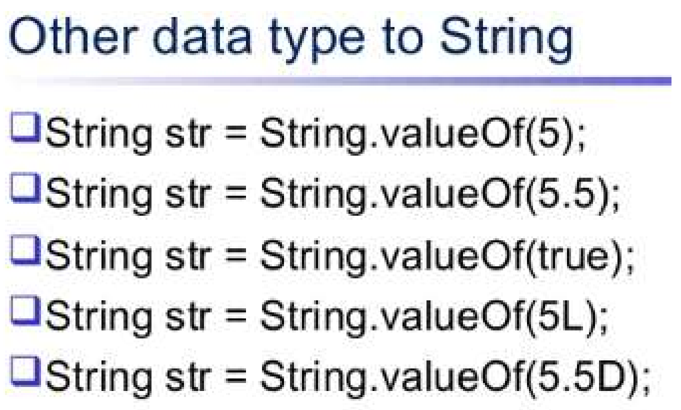
```
class MixOperator1 {
  public static void main(String[] args) {
    double fv1, fv2;
    int iv1 = 123;
    fv1 = iv1/50;
    fv2 = iv1/50.0;
    System.out.println("fv1:" + fv1);
    System.out.println("fv2:" + fv2);
  }
}
```

```
fv1:2.0
fv2:2.46
```



String to Other data type

- String str = “5.5” ;
- int i = Integer.parseInt(str);
- double d = Double.parseDouble(str);
- float f = Float.parseFloat(str);
- long l = Long.parseLong(str);
- String bStr = “true”;
- boolean b = Boolean.parseBoolean(bStr);



Other data type to String

- String str = String.valueOf(5);
- String str = String.valueOf(5.5);
- String str = String.valueOf(true);
- String str = String.valueOf(5L);
- String str = String.valueOf(5.5D);

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

Take survey Not now

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

Take survey Not now

We want to make Evernote better. Your feedback means the world to us. Can you take a minute to tell us how we're doing?

Take survey Not now