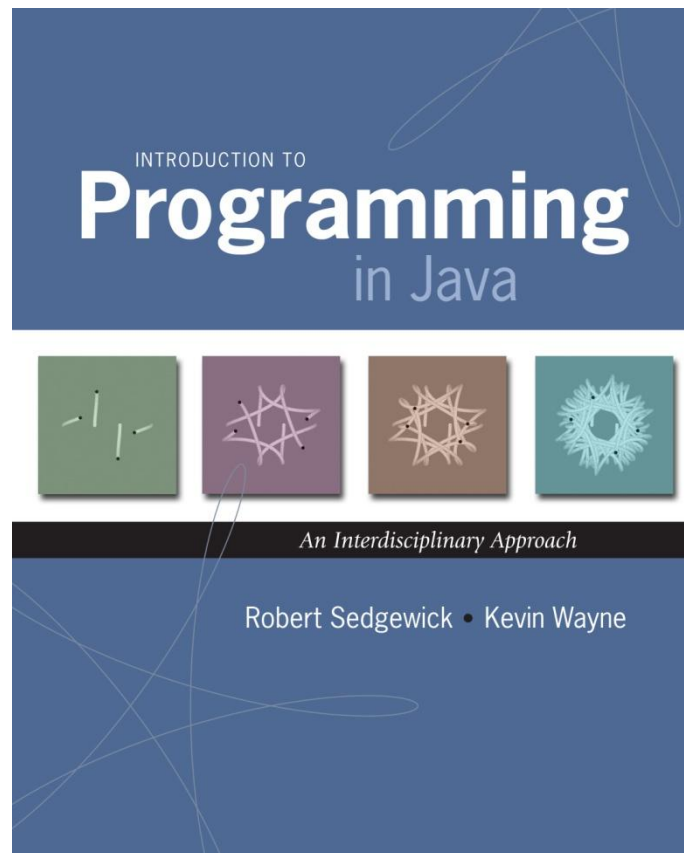


# 1.2 Built-in Types of Data



# Built-in Data Types

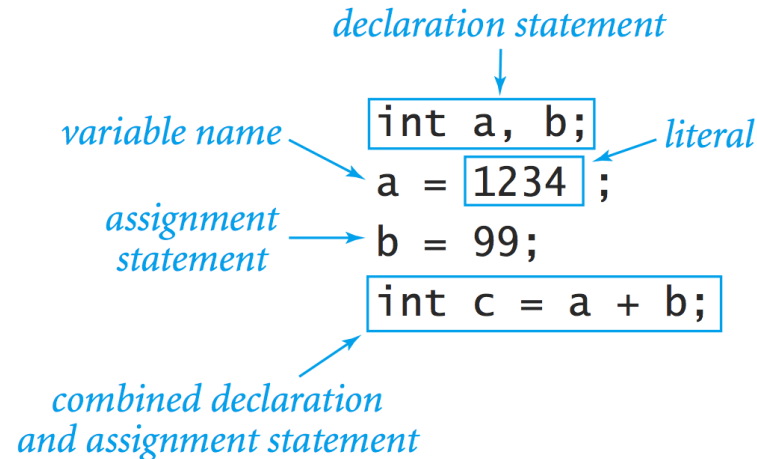
**Data type.** A set of values and operations defined on those values.

type	set of values	literal values	operations
char	characters	'A' '@'	compare
String	sequences of characters	"Hello World" "CS is fun"	concatenate
int	integers	17 12345	add, subtract, multiply, divide
double	floating point numbers	3.1415 6.022e23	add, subtract, multiply, divide
boolean	truth values	true false	and, or, not

# Basic Definitions

**Variable.** A name that refers to a value.

**Assignment statement.** Associates a value with a variable.



# Trace

**Trace.** Table of variable values after each statement.

	<u>a</u>	<u>b</u>	<u>t</u>
<code>int a, b;</code>	<i>undefined</i>	<i>undefined</i>	
<code>a = 1234;</code>	1234	<i>undefined</i>	
<code>b = 99;</code>	1234	99	
<code>int t = a;</code>	1234	99	1234
<code>a = b;</code>	99	99	1234
<code>b = t;</code>	99	1234	1234

Text

---

# Text

**String data type.** Useful for program input and output.

<i>values</i>	sequences of characters
<i>typical literals</i>	"Hello," "1 " " * "
<i>operation</i>	concatenate
<i>operator</i>	+

<i>expression</i>	<i>value</i>
"Hi, " + "Bob"	"Hi, Bob"
"1" + " 2 " + "1"	"1 2 1"
"1234" + " " + " " + "99"	"1234 + 99"
"1234" + "99"	"123499"

# Subdivisions of a Ruler

```
public class Ruler {  
    public static void main(String[] args) {  
        String ruler1 = "1";  
        String ruler2 = ruler1 + " 2 " + ruler1;  
        String ruler3 = ruler2 + " 3 " + ruler2;  
        String ruler4 = ruler3 + " 4 " + ruler3;  
        System.out.println(ruler4);  
    }  
}
```

"1"  
"1 2 1"  
"1 2 1 3 1 2 1"

string concatenation

```
% java Ruler  
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
```



# Integers

---

# Integers

`int` data type. Useful for expressing algorithms.

*values*  
*typical literals*  
*operations*  
*operators*

integers between  $-2^{31}$  and  $+2^{31}-1$

1234 99 -99 0 1000000

add	subtract	multiply	divide	remainder
+	-	*	/	%

<i>expression</i>	<i>value</i>	<i>comment</i>
5 + 3	8	
5 - 3	2	
5 * 3	15	
5 / 3	1	no fractional part
5 % 3	2	remainder
1 / 0		run-time error
3 * 5 - 2	13	* has precedence
3 + 5 / 2	5	/ has precedence
3 - 5 - 2	-4	left associative
( 3 - 5 ) - 2	-4	better style

# Integer Operations

```
public class IntOps {  
    public static void main(String[] args) {  
        int a = Integer.parseInt(args[0]);  
        int b = Integer.parseInt(args[1]);  
        int sum = a + b;  
        int prod = a * b;  
        int quot = a / b;  
        int rem = a % b;  
        System.out.println(a + " + " + b + " = " + sum);  
        System.out.println(a + " * " + b + " = " + prod);  
        System.out.println(a + " / " + b + " = " + quot);  
        System.out.println(a + " % " + b + " = " + rem);  
    }  
}
```

command-line  
arguments

```
% javac IntOps.java  
% java IntOps 1234 99  
1234 + 99 = 1333  
1234 * 99 = 122166  
1234 / 99 = 12  
1234 % 99 = 46
```

Java automatically converts  
a, b, and rem to type String

$$1234 = 12 * 99 + 46$$