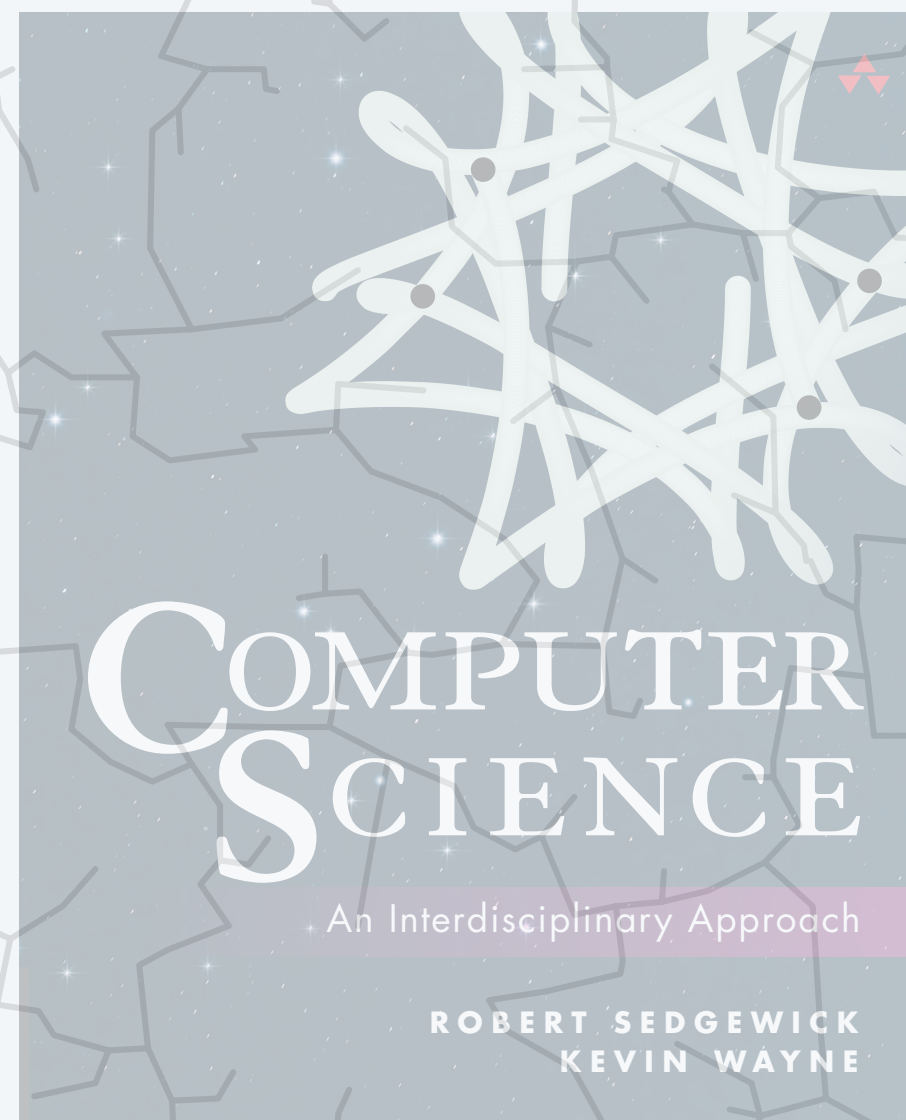


<https://introcs.cs.princeton.edu>

1.1-2 INTRO TO JAVA

- *why programming?*
- *your first program*
- *programming terminology*
- *String, int, and double*

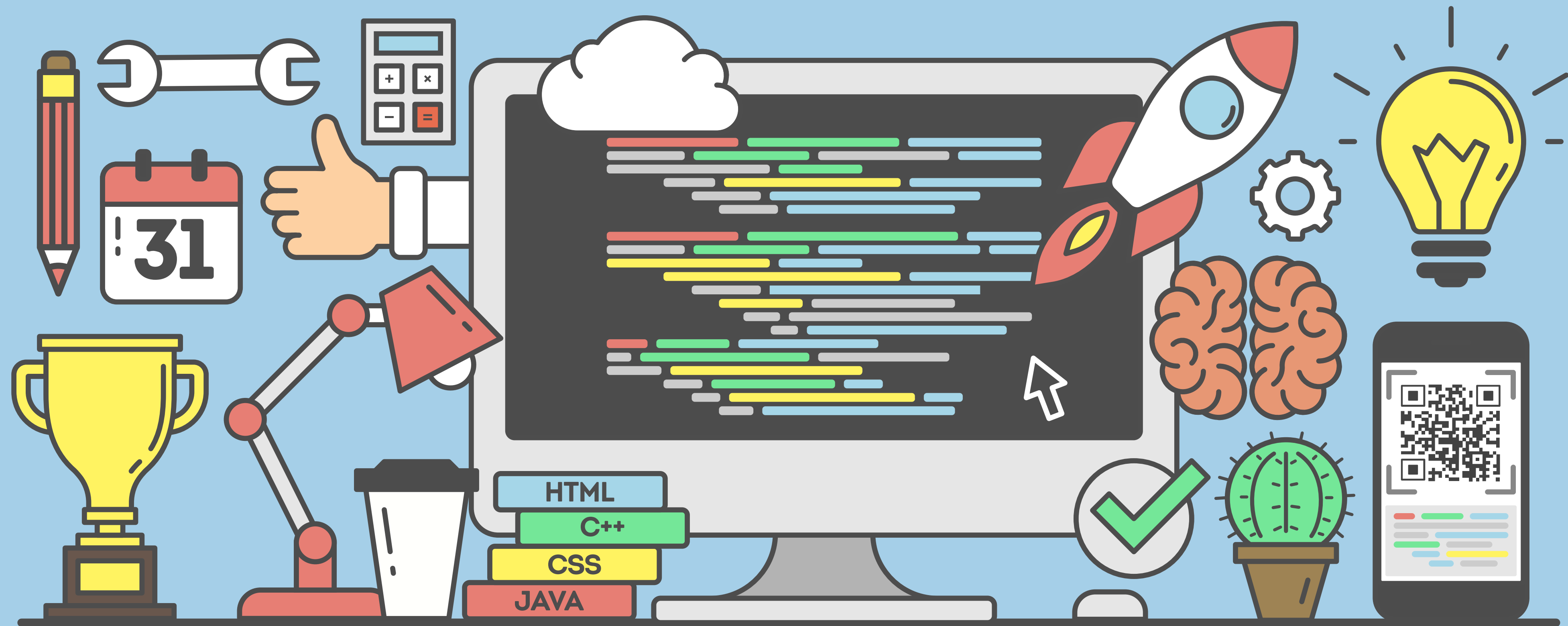


<https://introcs.cs.princeton.edu>

1.1-2 INTRO TO JAVA

- ▶ *why programming?*
- ▶ *your first program*
- ▶ *programming terminology*
- ▶ *String, int, and double*

PROGRAMMING



"Time Enough for Love" (1973) by Robert A. Heinlein

A human being should be able to

change a diaper,

plan an invasion,

butcher a hog,

conn a ship,

design a building,

write a sonnet,

balance accounts,

build a wall,

set a bone,

comfort the dying,

take orders,

give orders,

cooperate,

act alone,

solve equations,

analyze a new problem,

pitch manure,

*a natural, satisfying, and creative endeavor
(leading to accomplishments not otherwise possible)*

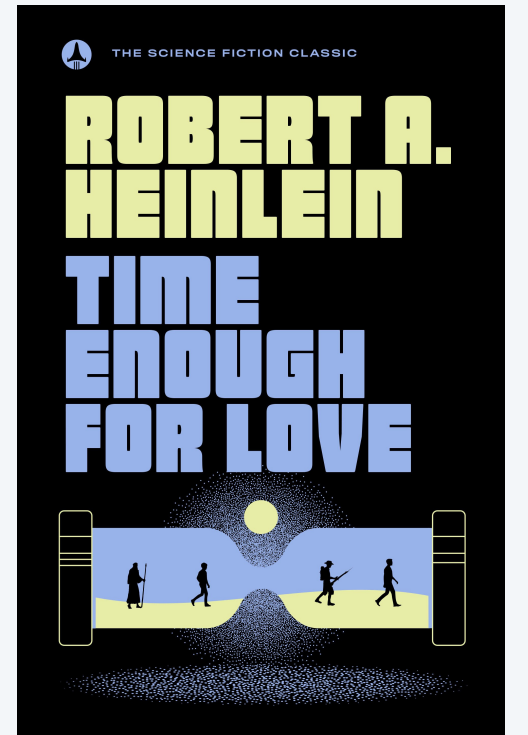


program a computer,

cook a tasty meal,

fight efficiently,

die gallantly.



Telling a computer what to do

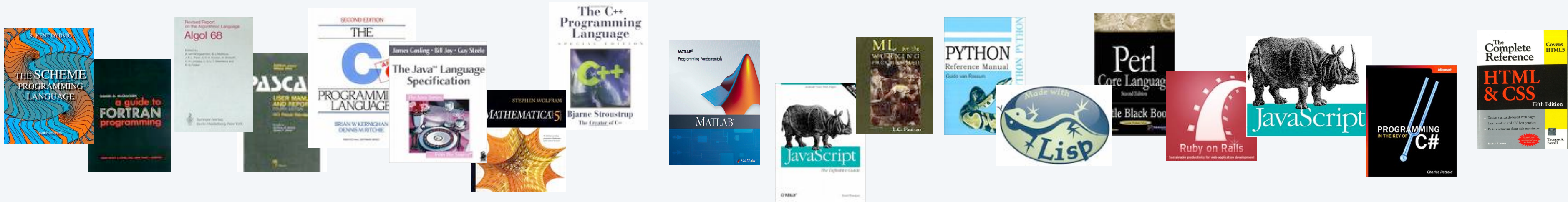
Machine languages. Easy for **computers**; error-prone for people.

```
00111111001000011001101110010111011100111011110100111111001100011110111111001111011110
011110111101100011000100000001001110010111001111110011011010001010011100000110000101001000
111101110111011111000011100010010100001001110000011010100110100001010110001001110010001...
```

Natural languages. Easy for **people**; error-prone for computers. *← rapid progress in past year (but not as robust as desired)*



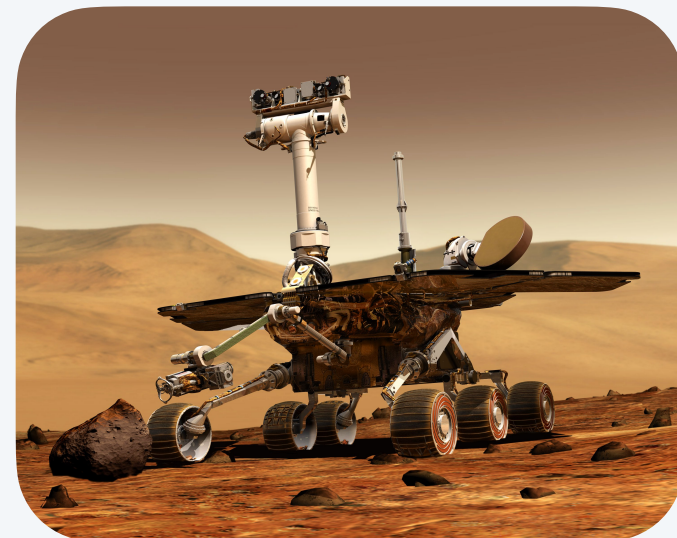
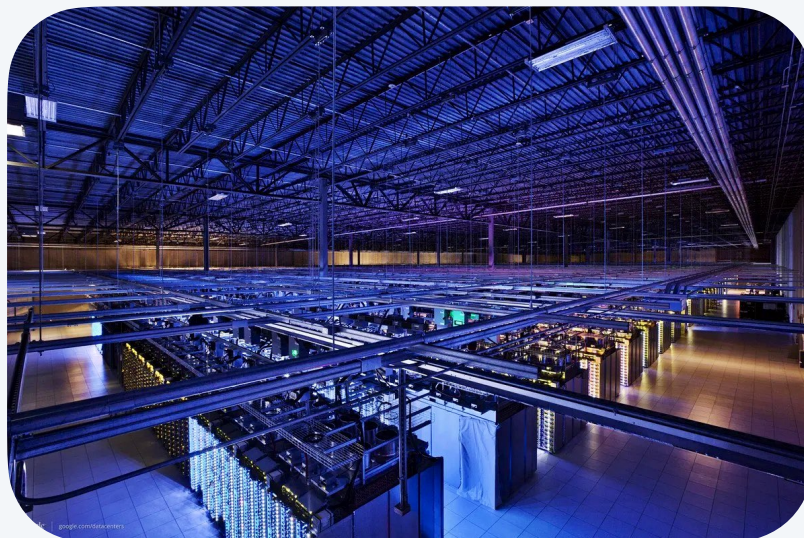
High-level programming languages. Enables **people and computers** to communicate effectively.



Java features.

- Embraces full set of modern abstractions.
- Freely available for macOS, Windows, and Linux.
- Variety of automatic checks for mistakes in programs.
- Widely used: millions of developers; billions of devices. ← *among top 3 languages for past two decades*

Ex. Android phones/TVs, web servers, Mars rover, medical devices, internet of things, ...



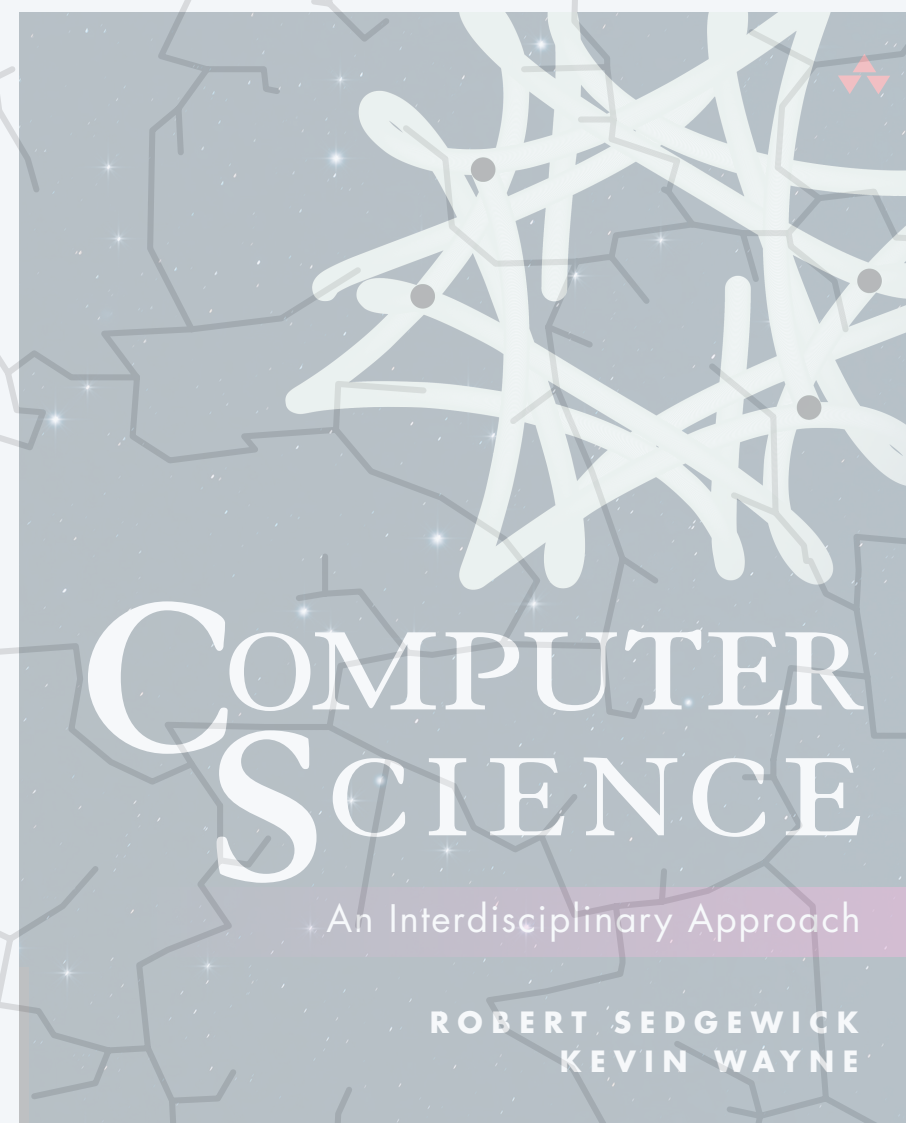
Reality. Use many programming languages; depends on domain.

A rich subset of the Java language



Your programs will primarily consist of these plus identifiers (names) that you make up. ← *seems like a lot, but typical English vocabulary is 20K words!*

data types	arithmetic		boolean		Math library		objects / methods			strings	
int	+	-	true	false	Math.min()	Math.max()	public	private		+	
double	*	/	&&		Math.sqrt()	Math.abs()	class	new		length()	
boolean	++	--	!	^	Math.log()	Math.exp()	static	final		charAt()	
char	%				Math.sin()	Math.cos()	void	main()		compareTo()	
String			type conversion		Math.PI	Math.E					toString()
				Integer.parseInt()				comments			
				Double.parseDouble()				/*	*/	//	
punctuation		comparisons		arrays		flow control		System methods			our I/O libraries
{ }	<	>	[]		if	else	System.out.print()			StdIn/In	
()	<=	>=	length		while	for	System.out.println()			StdOut/Out	
. ,	==	!=			do	return	System.out.printf()			StdPicture/Picture	
' "			assignment		break	continue				StdDraw/Draw	
;			=							StdAudio	



<https://introcs.cs.princeton.edu>

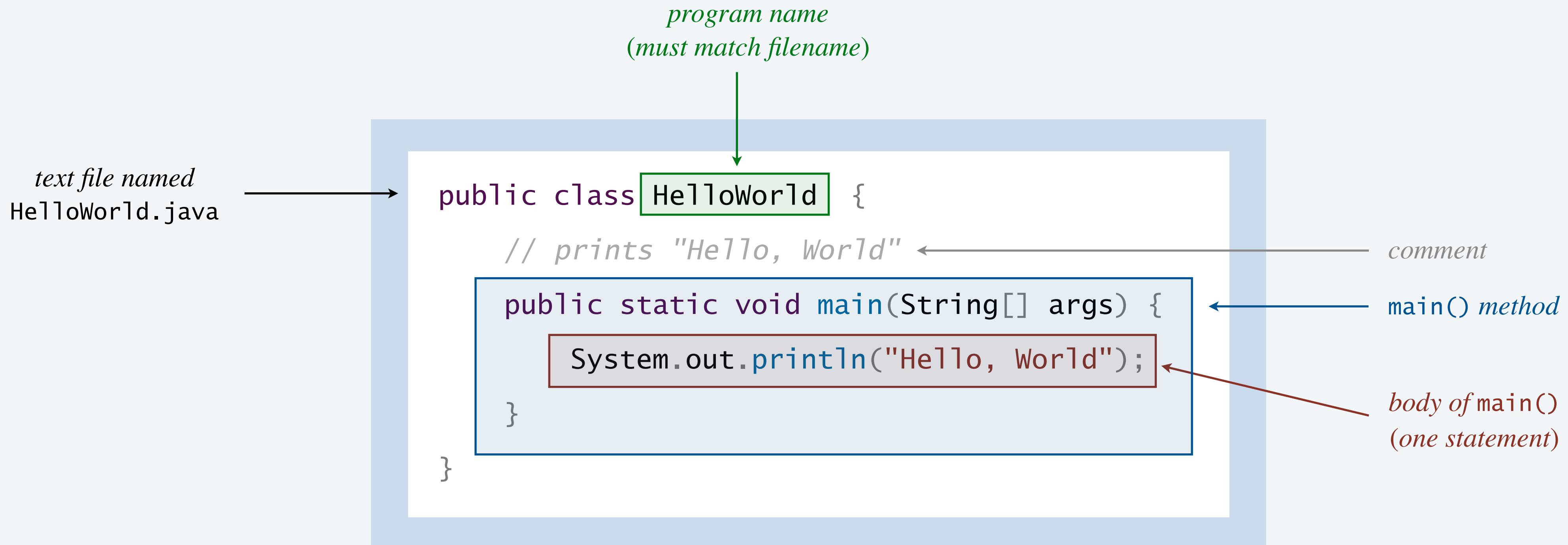
1.1-2 INTRO TO JAVA

- *why programming?*
- *your first program*
- *programming terminology*
- *String, int, and double*



```
< Hello, World! />
```

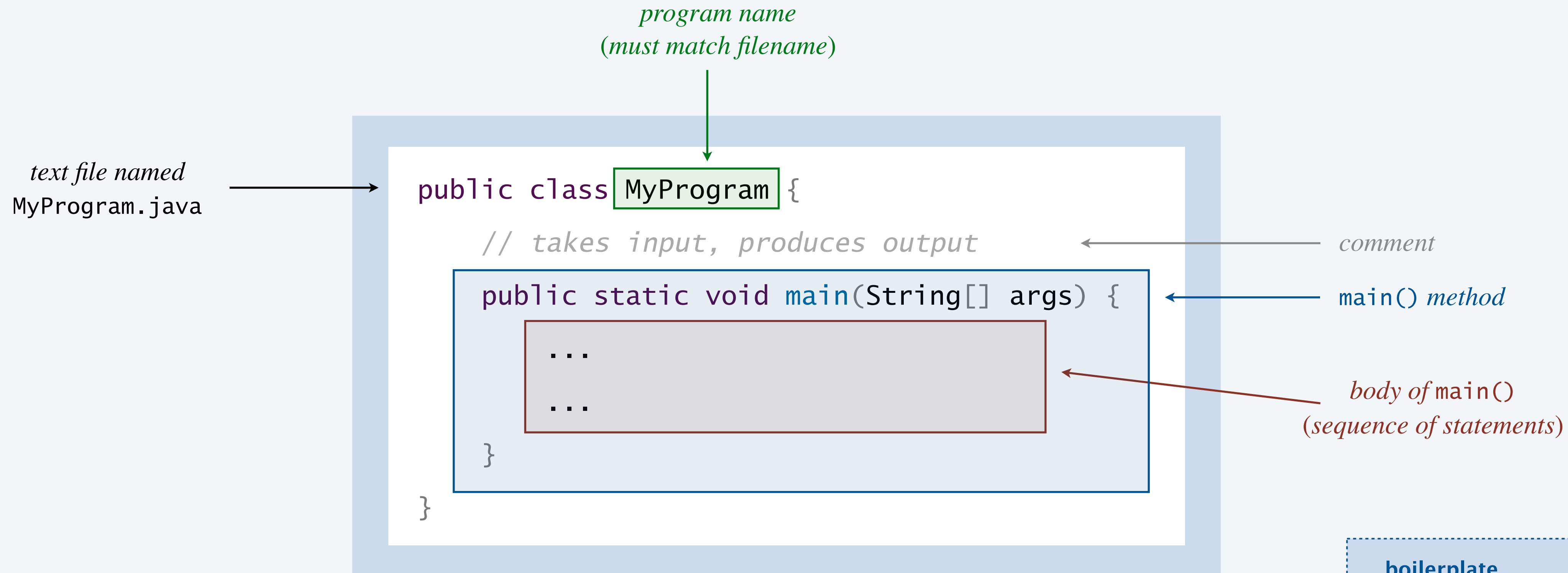
Anatomy of your first Java program



terminal window →

```
~/cos126/hello> javac HelloWorld.java ← compile the program  
  
~/cos126/hello> java HelloWorld ← execute (run) the program  
Hello, World  
    ↑  
  program output
```


Anatomy of your first few Java programs



```
~/cos126/hello> javac MyProgram.java  
~/cos126/hello> java MyProgram  
[program output]
```

boilerplate

public

class

static

void

main

String[]

*don't worry,
we'll learn
their meaning
in due time*

Hello World with textual input

Command-line arguments. A mechanism to receive textual input from the user.

```
public class HelloWorldWithArgument {
```

```
    // prints a command-line argument
```

```
    public static void main(String[] args) {
```

```
        System.out.print("Hello, World ");
```

```
        System.out.println(args[0]);
```

```
    }
```

```
}
```

command-line argument

*print, but without
the newline*

```
~/cos126/hello> javac HelloWorldWithArgument.java
```

```
~/cos126/hello> java HelloWorldWithArgument Kevin
```

```
Hello, World Kevin
```

```
~/cos126/hello> java HelloWorldWithArgument 🐅🐅🐅
```

```
Hello, World 🐅🐅🐅
```

*command-line
argument*

Hello World with audio output



Standard audio. Our course library for playing sound.

```
public class HelloWorldWithAudio {  
    // prints and speaks "Hello, World"  
    public static void main(String[] args) {  
        System.out.println("Hello, World");  
        StdAudio.play("HelloWorld.wav");  
    }  
}
```

an audio file

```
~/cos126/hello> javac-introcs HelloWorldWithAudio.java
```

```
~/cos126/hello> java-introcs HelloWorldWithAudio
```

```
Hello, World
```

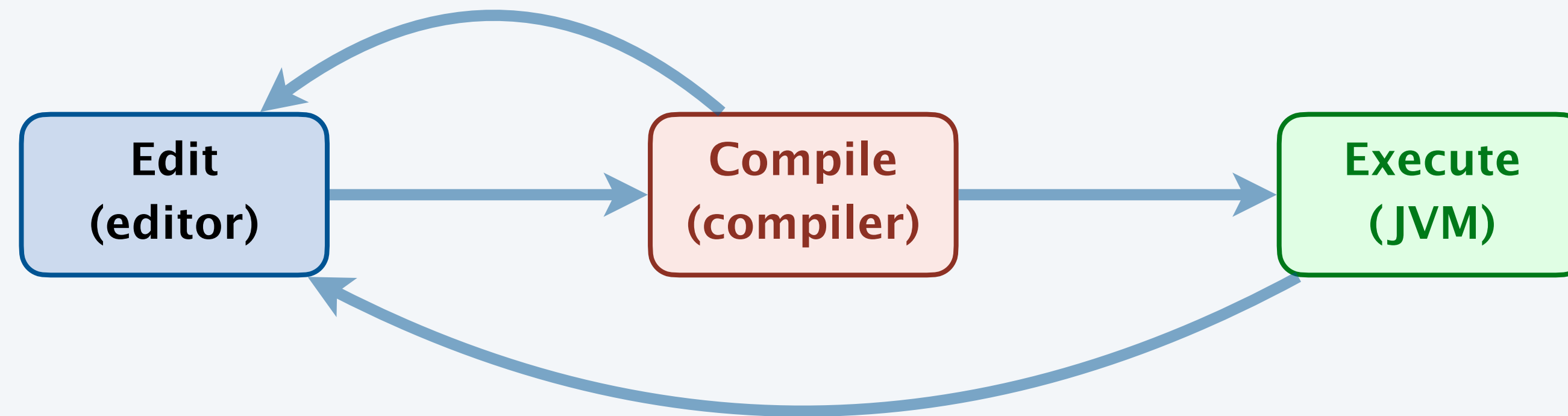
```
🔊 [speaks "Hello, World"]
```

*the javac-introcs and java-introcs commands
tell Java where to find our course libraries*

Program development in Java

Developing a Java program involves three steps:

- **Edit:** write your program.
- **Compile:** create a “machine-language” version of your program.
- **Execute:** run your program, taking input and producing output.



analogous to other creative processes
(compose–rehearse–play)

Almost always requires cyclic refinement:

- Not a legal Java program \leftarrow *compile-time error* \Rightarrow need to re-edit.
- A legal Java program that does the wrong thing \Rightarrow need to re-edit.

\nwarrow
*run-time error or
produces incorrect output*

Coding style

Coding style. Indentation, whitespace, naming conventions, comments, ...

Goal. Make it easier for programmers (including you!) to read and understand the code.

textbook

Program 1.1.1 Hello, World

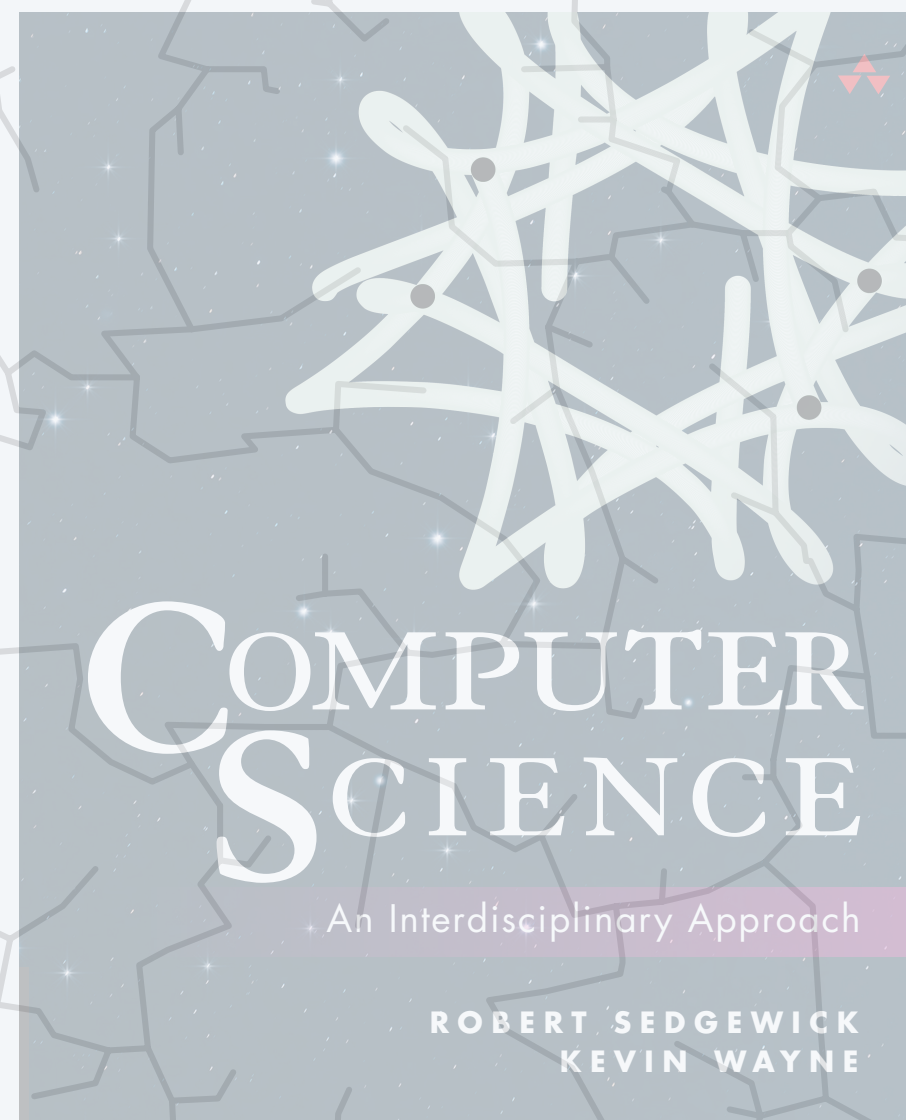
```
public class HelloWorld
{
    public static void main(String[] args)
    {
        // Prints "Hello, World" in the terminal window.
        System.out.println("Hello, World");
    }
}
```

IntelliJ

```
1  /*****
2   * Prints "Hello, World". By tradition, this is everyone's first program.
3   *
4   * These first 6 lines of text are comments. They are not part of the program;
5   * they serve to remind us about its properties.
6   *****/
7
8  public class HelloWorld {
9      public static void main(String[] args) {
10
11          // Prints "Hello, World" in the terminal window.
12          System.out.println("Hello, World");
13      }
14  }
```

Java compiler

```
public class HelloWorld { public static void main ( String [ ] args { System . out . println ( "Hello, World" ) ; } }
```



<https://introcs.cs.princeton.edu>

1.1-2 INTRO TO JAVA

- *why programming?*
- *your first program*
- *programming terminology*
- *String, int, and double*

Built-in data types

A **data type (type)** is a set of values and a set of operations on those values.

type	set of values	example values	examples of operations
<i>int</i>	<i>integers</i>	17 -12345	<i>add, subtract, multiply, divide, compare, equality</i>
<i>double</i>	<i>floating-point numbers</i>	2.5 -0.125	<i>add, subtract, multiply, divide, compare, equality</i>
<i>boolean</i>	<i>truth values</i>	true false	<i>and, or, not, equality</i>
<i>String</i>	<i>sequences of characters</i>	"Hello, World" "COS 126 is fun!"	<i>concatenate</i>

Java’s built-in data types
(that we use regularly in this course)

Programming terminology

Program. Sequence of statements. *← for now*

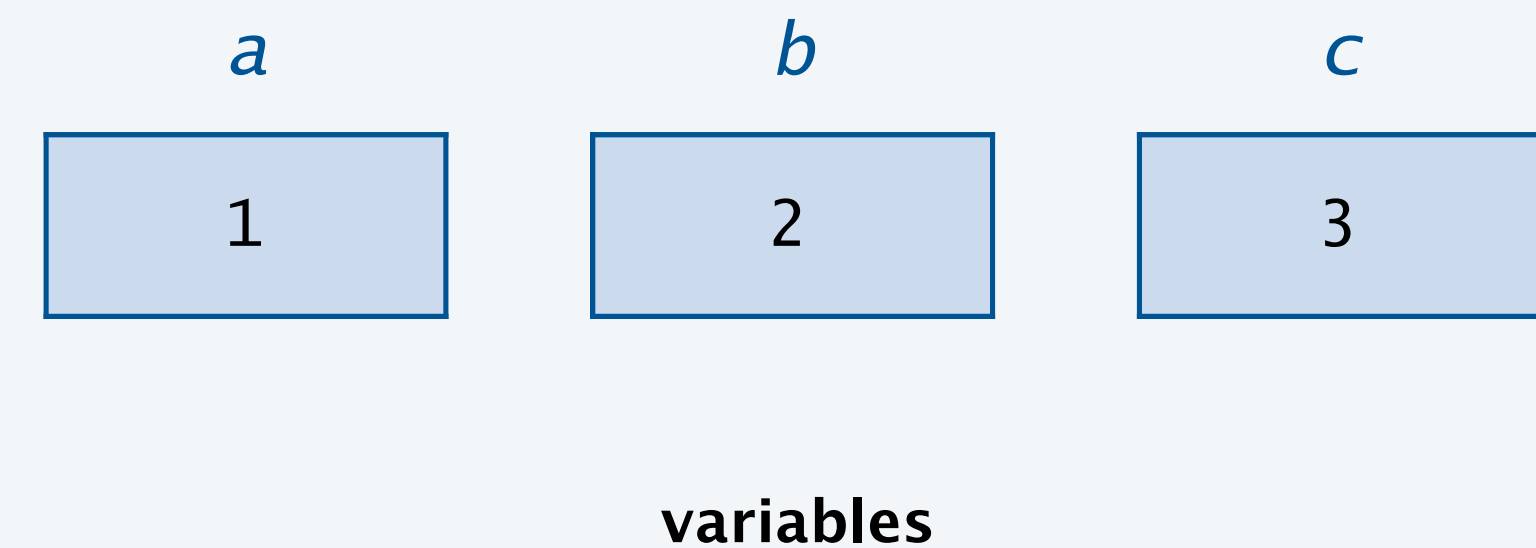
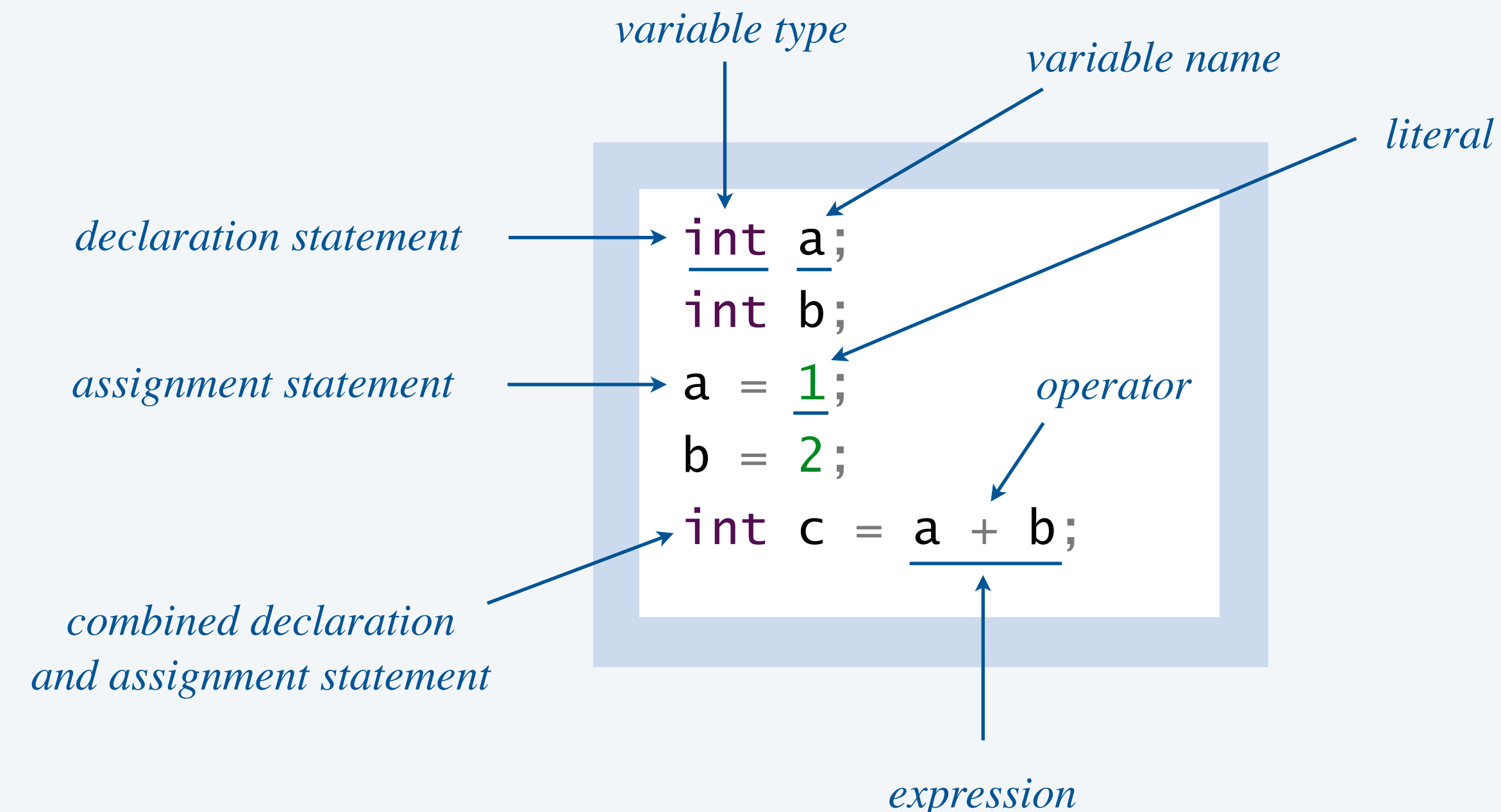
Declaration statement. Associates a variable with a name and data type.

Variable. A place to store a data-type value (and, later, refer to it by name).

Assignment statement. Stores a data-type value in a variable.

Literal. Programming-language representation of a data-type value.

Expression. A combination of variable names, literals, operators, etc. that evaluates to a value.



Assignment statements

Q. How does an assignment statement work?

A. Java evaluates the **expression on the RHS** and assigns that value to the **variable on the LHS**.

↑
*expression type must be
compatible with variable type*

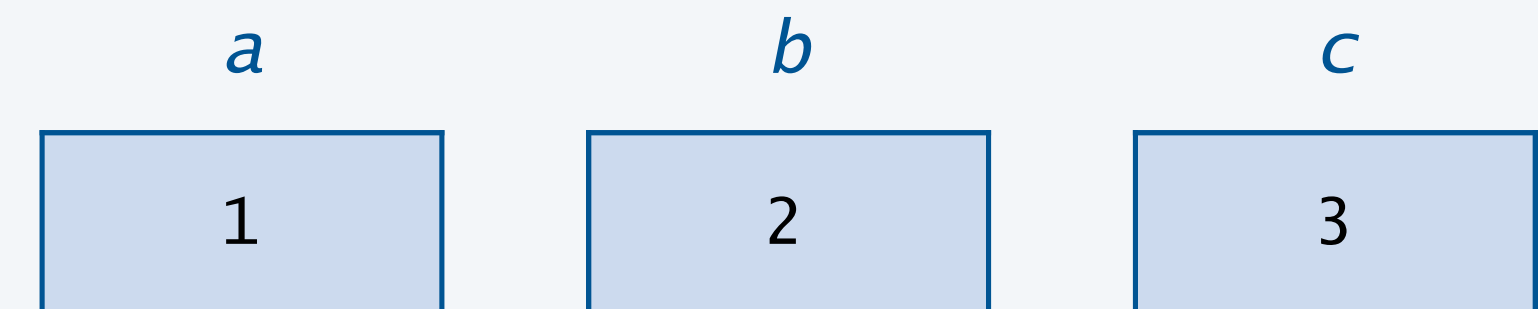
*assignment operator =
means assignment
(not math equality!)*

*variable name evaluates to
value stored in variable*

```
int a;  
int b;  
a = 1;  
b = 2;  
int c = a + b;
```

*variable
of type int*

*expression
of type int*



variables

Valid and invalid assignment statements

Q. Which of these independent code fragments are valid?

statements	compiles?	remark
<code>int a = 1; 123 = a;</code>	😞	<i>LHS is not a variable (= does not mean math equality)</i>
<code>double a = 2.5; int b = a;</code>	😞	<i>RHS type is incompatible with LHS type</i>
<code>String s = 123;</code>	😞	<i>RHS type is incompatible with LHS type</i>
<code>int b = 2; int a = 3 * b;</code>	😍	<i>RHS can be an expression</i>
<code>int a = 3; a = 2 * a;</code>	😍	<i>a variable can be reassigned (that's why it's called a variable!)</i>
<code>int a = 2 * a;</code>	😞	<i>a variable must be assigned a value before it can be used in an expression</i>



Q. What does this code fragment do?

A. Let's **trace** the variables during execution of the code. *← table of variable values*

```
int a = 100;
int b = 126;
int temp = a;
a = b;
b = temp;
```

*this idiom exchanges
the values stored in the
variables a and b*

	a	b	temp
start of code fragment	undeclared	undeclared	undeclared
int a = 100;	100	undeclared	undeclared
int b = 126;	100	126	undeclared
int temp = a;	100	126	100
a = b;	126	126	100
b = temp;	126	100	100
	126	100	100

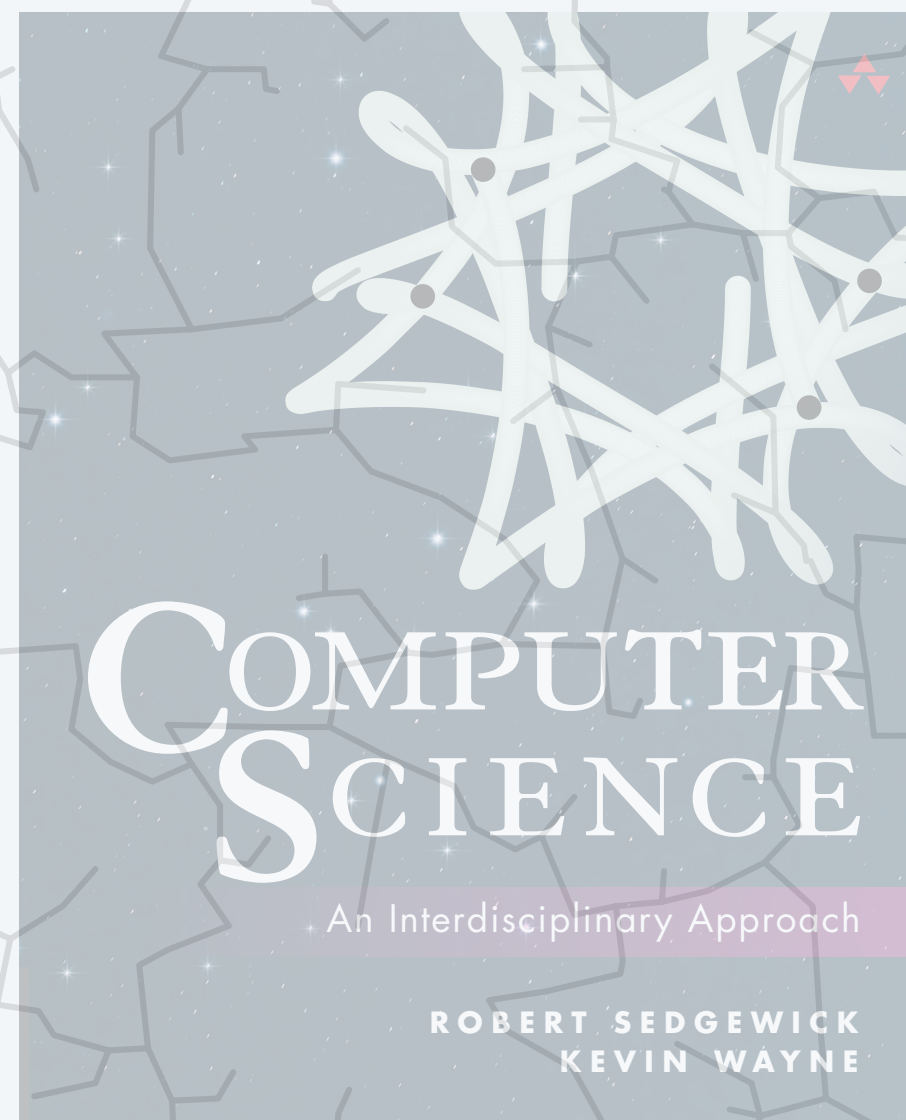
trace of variables
(after each statement)



What are the values stored in the variables *a* and *b* after the code fragment is executed?

- A. 100 and 126.
- B. 126 and 100.
- C. 226 and 126.
- D. -26 and -26.
- E. Compile-time error.

```
int a = 100;  
int b = 126;  
a = a + b;  
b = a - b;  
a = a - b;
```



<https://introcs.cs.princeton.edu>

1.1-2 INTRO TO JAVA

- *why programming?*
- *your first program*
- *programming terminology*
- ***String, int, and double***

The *String* data type

Typical usage. Program input and output; text processing.

values	<i>sequences of characters</i>			
example literals	"Hi "	"1234"	"Nĩ hǎo"	"💩💩💩"
operation	<i>concatenation</i>			
operator	+			

expression	value	remark
"My " + "Precious"	"My Precious"	<i>spaces within a string literal matter</i>
"1234" + "99"	"123499"	<i>strings are not integers</i>
"A" + "B" + "C"	"ABC"	<i>can concatenate several strings together, in one expression</i>
"ᠠᠯᠤᠰ " + "ᠠᠨᠠᠭᠤᠨᠠᠨᠢᠨᠠᠨᠢ"	"ᠠᠯᠤᠰ ᠠᠨᠠᠭᠤᠨᠠᠨᠢᠨᠠᠨᠢ"	<i>Unicode supported</i>

Command-line arguments are strings

Command-line arguments. The variables `args[0]`, `args[1]`, `args[2]`, ... are of type *String*. Java initializes them automatically to corresponding values.

*we'll revisit notation
in Section 1.4 (arrays)*

```
public class CommandLineArguments {  
    public static void main(String[] args) {  
        String a = args[0];  
        String b = args[1];  
        String c = args[2];  
        String result = a + "-" + b + "-" + c;  
        System.out.println(result);  
    }  
}
```

```
~/cos126/datatypes> java CommandLineArguments A B C  
A-B-C
```

args[0]

```
~/cos126/datatypes> java CommandLineArguments do re mi  
do-re-mi
```

*arguments delimited
by whitespace*

```
~/cos126/datatypes> java CommandLineArguments  
Exception in thread "main"  
java.lang.ArrayIndexOutOfBoundsException:  
Index 0 out of bounds for length 0 at  
CommandLineArguments.main(CommandLineArguments.java:3)
```

*line number
of error*

The *int* data type

Typical usage: math calculations involving integers; program control flow.

values	<i>integers between -2^{31} and $2^{31} - 1$</i>				
example literals	1234 99 0 1000000 -3				
operations	<i>add</i>	<i>subtract</i>	<i>multiply</i>	<i>divide</i>	<i>remainder</i>
operators	+	-	*	/	%

*only 2^{32} different int values
(not quite the same as integers)*

expression	value	remark
20 + 3	23	
20 - 3	17	
20 * 3	60	
20 / 3	6	drop fractional part
20 % 3	2	remainder
20 / 0	-	division-by-zero error
<u>2147483647</u> + 1	-2147483648	integer overflow
$2^{31} - 1$		

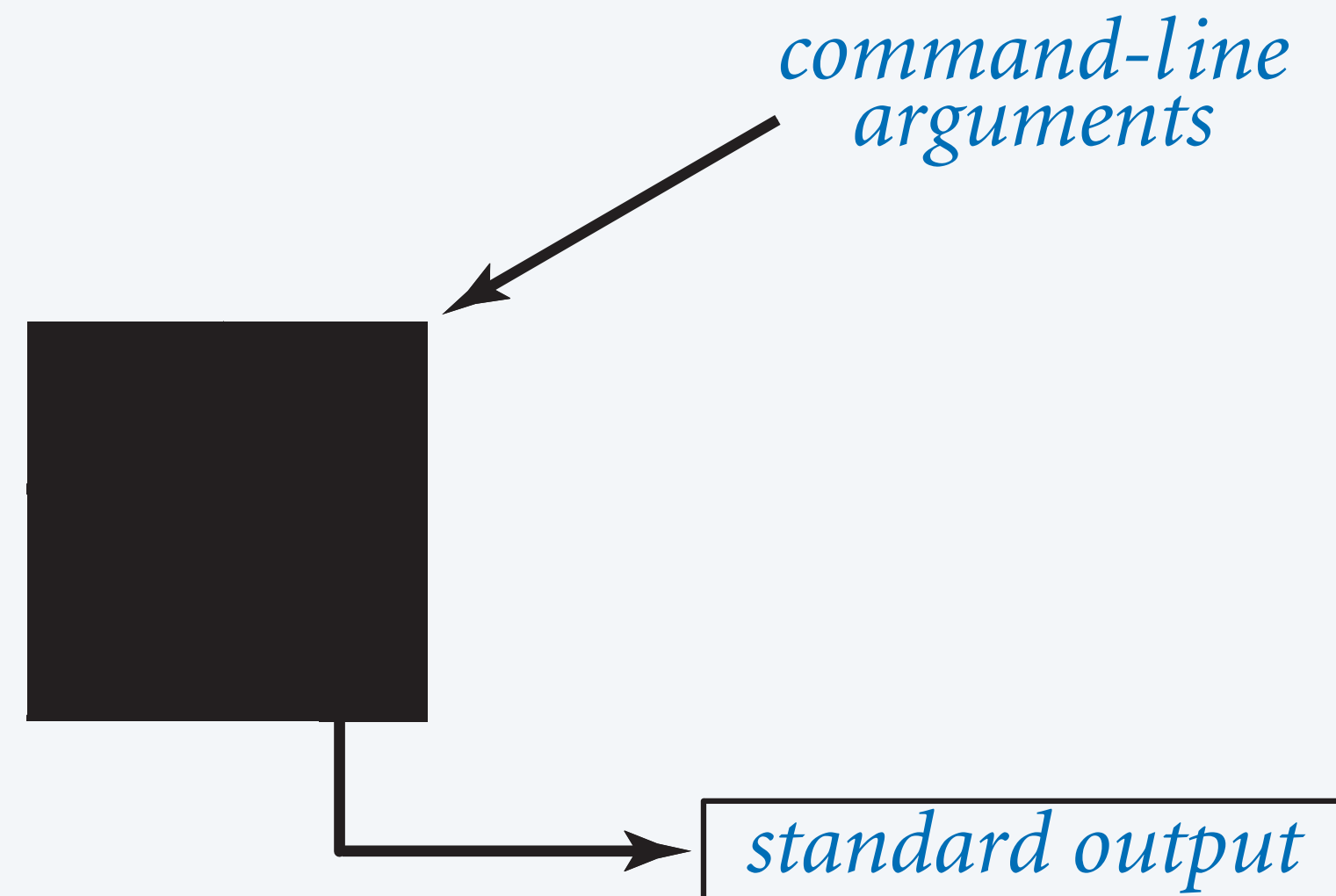
*applying an int operator
to two int operands
always results in an int
(or division-by-zero error)*

don't use int with very large integers

Input and output

Java I/O model. [for now]

- Read strings from the command line.
- Print strings to standard output.



Q. How to read integers from the command line?

A. The system method *Integer.parseInt()* converts from a *String* to an *int*.

Q. How to print integers to standard output?

A. When a *String* is concatenated with an *int*, Java converts the *int* to a *String*.

Input and output with integers

```
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);  
    }  
}
```

*converts from
String to int*

*converts from
int to String*

```
~/cos126/datatypes> java IntOps 20 3
```

20 + 3 = 23

20 * 3 = 60

20 / 3 = 6

20 % 3 = 2

← 20 = 6×3 + 2

```
~/cos126/datatypes> java IntOps 1234 10
```

1234 + 10 = 1244

1234 * 10 = 12340

1234 / 10 = 123

1234 % 10 = 4

← 1234 = 123×10 + 4

```
~/cos126/datatypes> java IntOps 1234 Hello
```

Exception in thread "main"

java.lang.NumberFormatException:

For input string: "Hello"

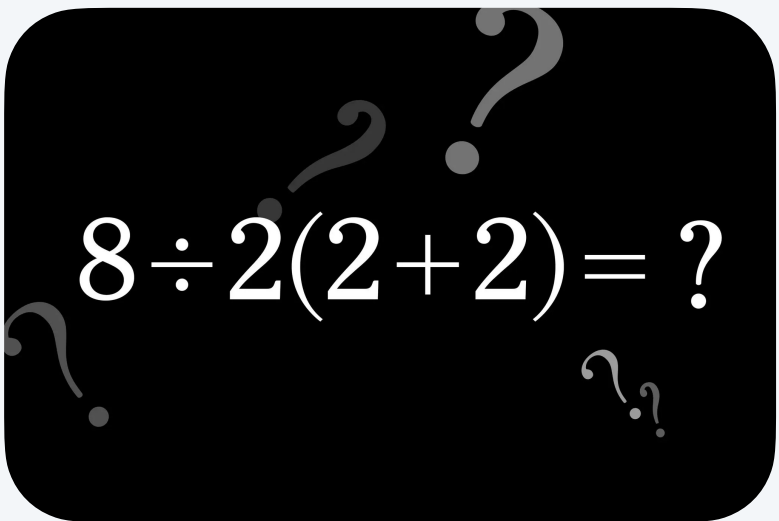
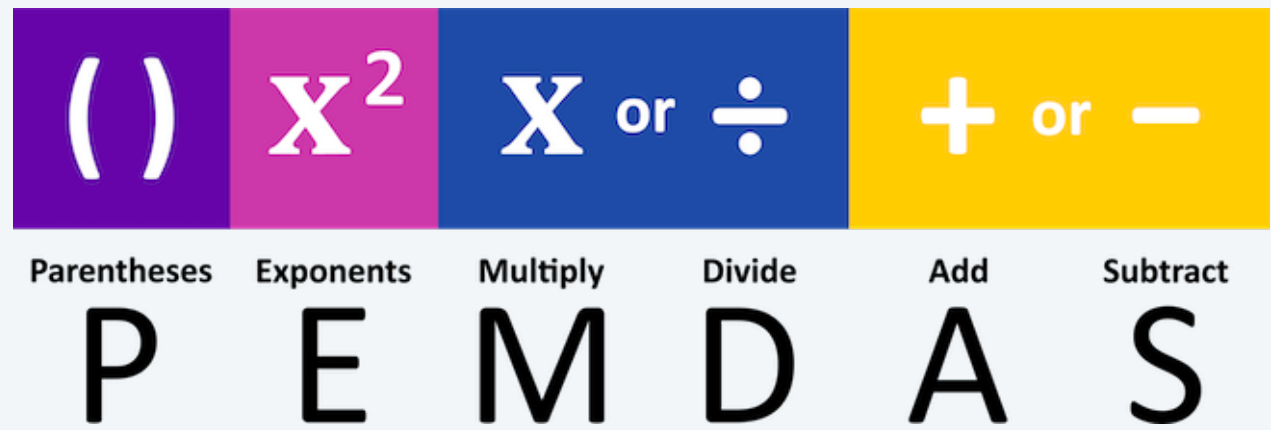
...

at IntOps.main(IntOps.java:4)

← *line number of
run-time error*

Order of operations

PEMDAS. Rules for evaluating an arithmetic expression.



internet meme

Operator precedence. Priority for grouping operands with operators in an expression.

Operator associativity. Rule for when two operators in an expression have same priority.

expression	equivalent to	value	remark
$3 * 5 - 2$	$(3 * 5) - 2$	13	$*$ has higher precedence than $-$
$3 + 5 / 2$	$3 + (5 / 2)$	5	$/$ has higher precedence than $+$
$3 - 5 - 2$	$(3 - 5) - 2$	-4	left-to-right associative
$(3 - 5) - 2$	itself	-4	better style
$8 / 2 * (2 + 2)$	$(8 / 2) * (2 + 2)$	16	left-to-right associative (multiplication and division have same precedence)



What value does the following expression evaluate to?

```
1 + 2 + "ABC" + 3 + 4
```

- A. "12ABC34"
- B. "3ABC7"
- C. "3ABC34"
- D. "12ABC7"
- E. Compile-time error.

The `double` data type

Typical usage: scientific calculations involving real numbers.

values	<i>IEEE floating-point numbers</i>				
example literals	18.25	-2.0	1.4142135623730951	6.022E23	
operations	<i>add</i>	<i>subtract</i>	<i>multiply</i>	<i>divide</i>	<i>remainder</i>
operators	+	-	*	/	%

*only 2⁶⁴ different double values
(not quite the same as real numbers)*

*6.022 × 10²³
(scientific notation)*

expression	value	remark
1.5 + 0.25	1.75	
1.5 - 0.25	1.25	
1.5 * 2.0	3.0	
5.0 / 3.0	1.6666666666666667	not exactly $\frac{5}{3}$
-1.0 / 0.0	-Infinity	not an error
0.0 / 0.0	NaN	“not a number”

*applying a double operator
to two double operands
always results in a double
(never results in an error)*

*only binary fractional values
can be represented exactly, such as*
 $\frac{1}{4} + \frac{1}{16} + \frac{1}{128} = 0.3203125$
(but not $\frac{5}{3}$, $\frac{1}{10}$, or π)

Excerpts from Java's *Math* library

Math library function

description

static double abs(double a)
static double max(double a, double b)
static double min(double a, double b)

absolute value of a
maximum of a and b
minimum of a and b

← *also defined for int*

static double sin(double theta)
static double cos(double theta)
static double tan(double theta)

sine (sin θ)
cosine (cos θ)
tangent (tan θ)

static double exp(double a)
static double log(double a)
static double sqrt(double a)
static double pow(double a, double b)

exponential (e^a)
natural logarithm ($\log_e a$)
positive square root (\sqrt{a})
power (a^b)

static long round(double a)
static double random()

round to the nearest integer
pseudorandom number in [0, 1)

static double E
static double PI

value of e (constant)
value of π (constant)

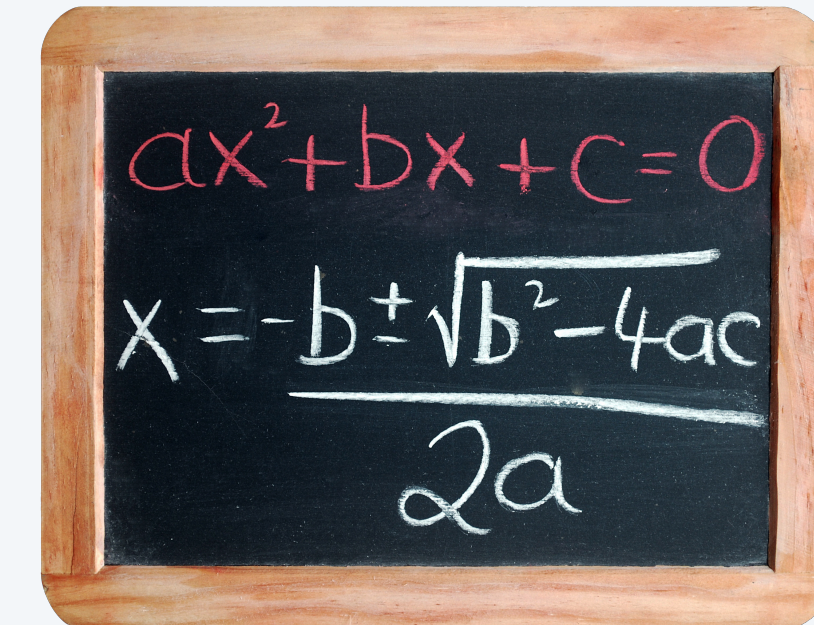


You can discard your calculator now (please).

expression	value
Math.max(1.0, 2.5)	2.5
Math.cos(0.0)	1.0
Math.sqrt(2.0)	1.4142135623730951
Math.random()	0.7707780210347349
Math.PI	3.141592653589793

Quadratic equation

Goal. Print the solutions to the equation $ax^2 + bx + c = 0$, assuming $a \neq 0$.


$$ax^2 + bx + c = 0$$
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
public class Quadratic {  
    public static void main(String[] args) {  
  
        // Parse coefficients from command-line.  
        double a = Double.parseDouble(args[0]);  
        double b = Double.parseDouble(args[1]);  
        double c = Double.parseDouble(args[2]);  
  
        // Calculate roots of ax^2 + bx + c = 0.  
        double discriminant = b*b - 4.0*a*c;  
        double d = Math.sqrt(discriminant);  
        double root1 = (-b + d) / (2.0*a);  
        double root2 = (-b - d) / (2.0*a);  
  
        // Print the two roots.  
        System.out.println(root1);  
        System.out.println(root2);  
    }  
}
```

```
~/cos126/datatypes> java Quadratic 1.0 -3.0 2.0  
2.0  
1.0
```

$$x^2 - 3x + 2$$

```
~/cos126/datatypes> java Quadratic 1.0 -1.0 -1.0  
1.618033988749895  
-0.6180339887498949
```

$$\longleftarrow \frac{1 \pm \sqrt{5}}{2}$$

$$x^2 - x - 1$$

```
~/cos126/datatypes> java Quadratic 1.0 1.0 1.0  
NaN  
NaN
```

$$\longleftarrow \frac{-1 \pm 3i}{2}$$

$$x^2 + x + 1$$

Floating-point catastrophe

Patriot missile.

- In February 1991, a Patriot missile failed to track and intercept an incoming Scud missile.
- Scud missile hit a U.S. Army barracks, killing 28 and wounding 260.
- Time measured in tenths of a second, but stored using binary floating-point. $\longleftarrow \frac{1}{10}$ *not exactly representable*
- After 100 hours of continuous use, system's internal clock had drifted by 1/3 second.



Scud Missile Hits a U.S. Barracks, Killing 27

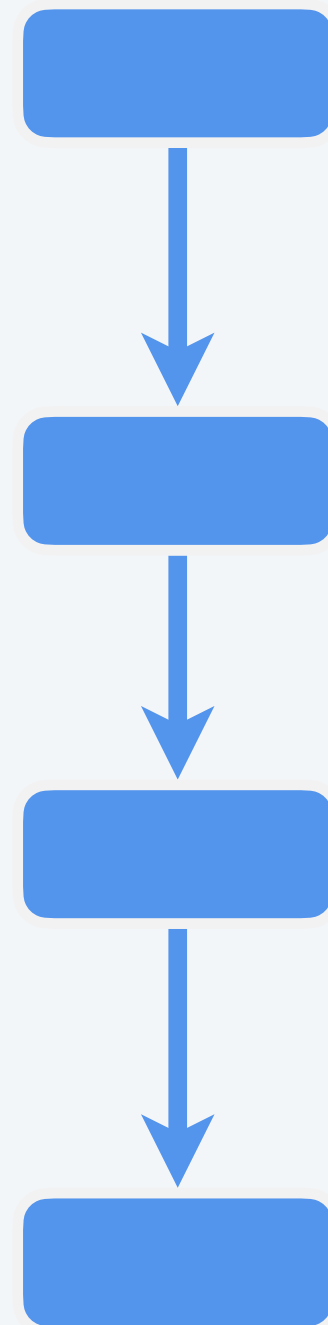


An Iraqi Scud missile that demolished a barracks yesterday near Dhahran, Saudi Arabia, killed 27 American soldiers and wounded 98, according to officials. A soldier told photographers to leave the scene.

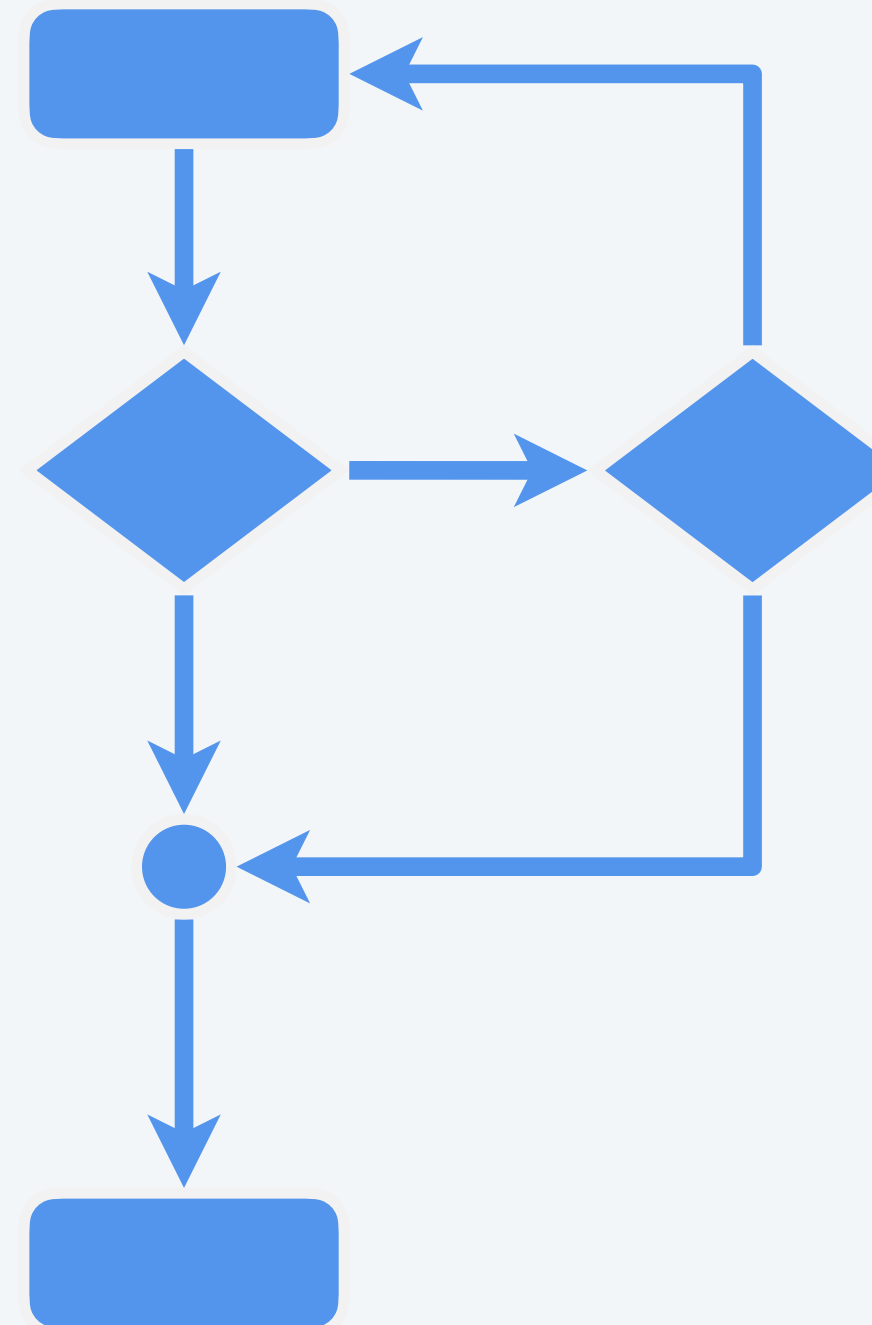
Overview

This lecture. Write programs with **declaration**, **assignment**, and **print** statements.

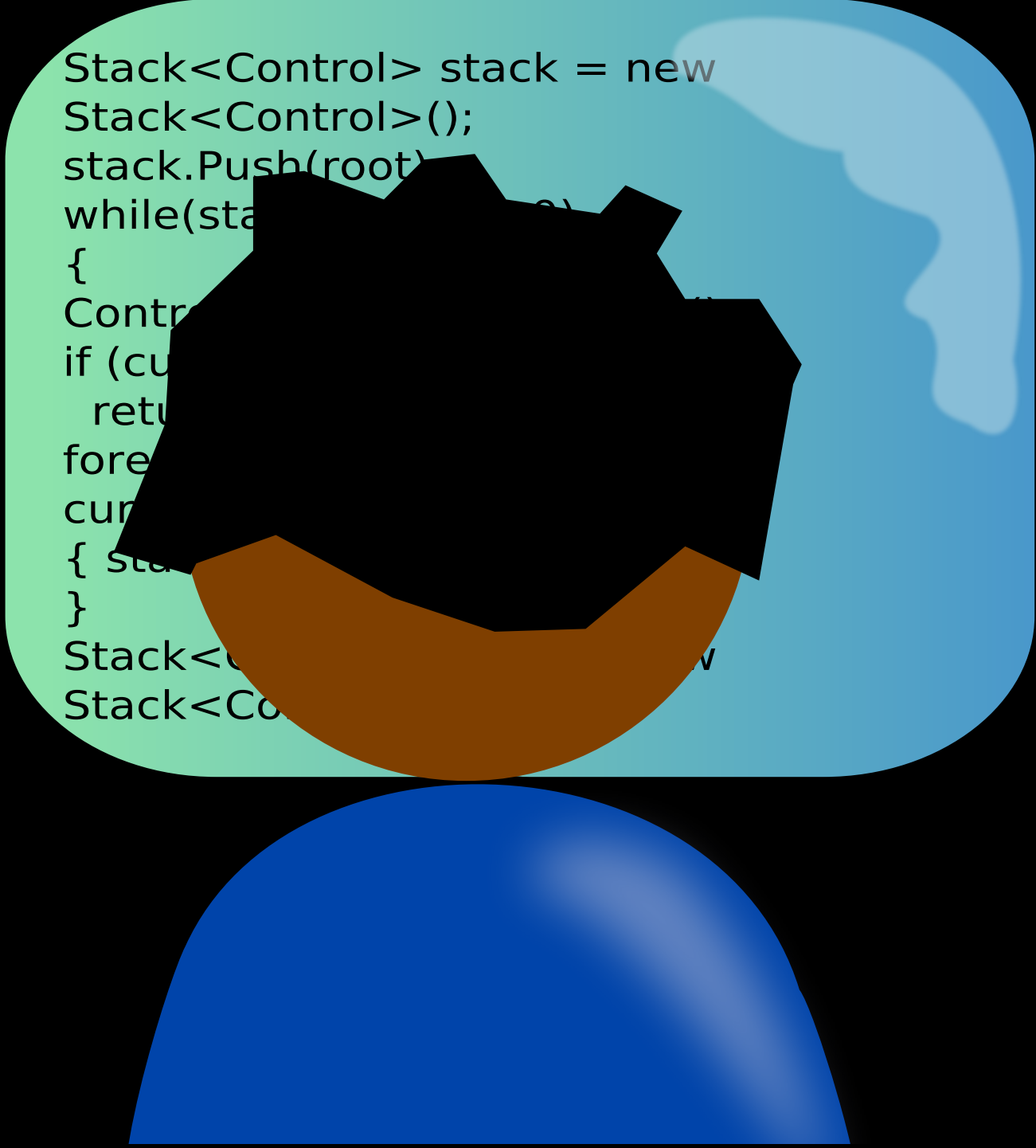
Next week. Write programs with **conditionals** and **loops**.



straight-line control flow



control flow with conditionals and loops



```
Stack<Control> stack = new  
Stack<Control>();  
stack.Push(root);  
while(stack.Count > 0)  
{  
    Control cur = stack.Pop();  
    if (cur != null)  
        return cur;  
    foreach (Control child in cur.Children)  
        stack.Push(child);  
}  
Stack<Control> stack = new  
Stack<Control>();
```

More questions?



attend office hours



ask on Ed



Credits

media	source	license
<i>Hello, World</i>	<u>Adobe Stock</u>	<u>education license</u>
<i>Programming</i>	<u>Adobe Stock</u>	<u>education license</u>
<i>Time Enough for Love</i>	<u>Robert A. Heinlein</u>	
<i>Ada Lovelace</i>	<u>Margaret Sarah Carpenter</u>	<u>public domain</u>
<i>Babbage's Analytic Engine</i>	<u>Science Museum, London</u>	<u>CC BY-SA 2.0</u>
<i>Java Logo</i>	<u>Sun Microsystems</u>	
<i>Android Phone</i>	<u>nicepng.com</u>	<u>public domain</u>
<i>Google Data Center</i>	<u>Google / Connie Zhou</u>	
<i>Mars Rover</i>	<u>NASA / JPL / Cornell</u>	<u>public domain</u>
<i>MRI Machine</i>	<u>Adobe Stock</u>	<u>education license</u>
<i>Internet of Things</i>	<u>Adobe Stock</u>	<u>education license</u>

Credits

media	source	license
<i>Jazz Musician Band</i>	<u>Adobe Stock</u>	<u>education license</u>
<i>Programmer</i>	<u>Jaime Botero</u>	<u>public domain</u>
<i>PEMDAS</i>	<u>Mometrix</u>	
<i>PEMDAS meme</i>	<u>New York Times</u>	
<i>Scientific Calculator</i>	<u>Wikimedia</u>	<u>CC BY-SA 3.0</u>
<i>Solving Quadratic Equations</i>	<u>Adobe Stock</u>	<u>education license</u>
<i>Patriot Missile Launcher</i>	<u>Raytheon</u>	
<i>Incorrectly Calculated Range Gate</i>	<u>GAO</u>	<u>public domain</u>
<i>Scud Missile Hits a U.S. Barracks</i>	<u>New York Times</u>	
<i>Students Asking Questions</i>	<u>Adobe Stock</u>	<u>education license</u>
<i>Question Marks</i>	<u>pikpng.com</u>	<u>non-commercial use</u>