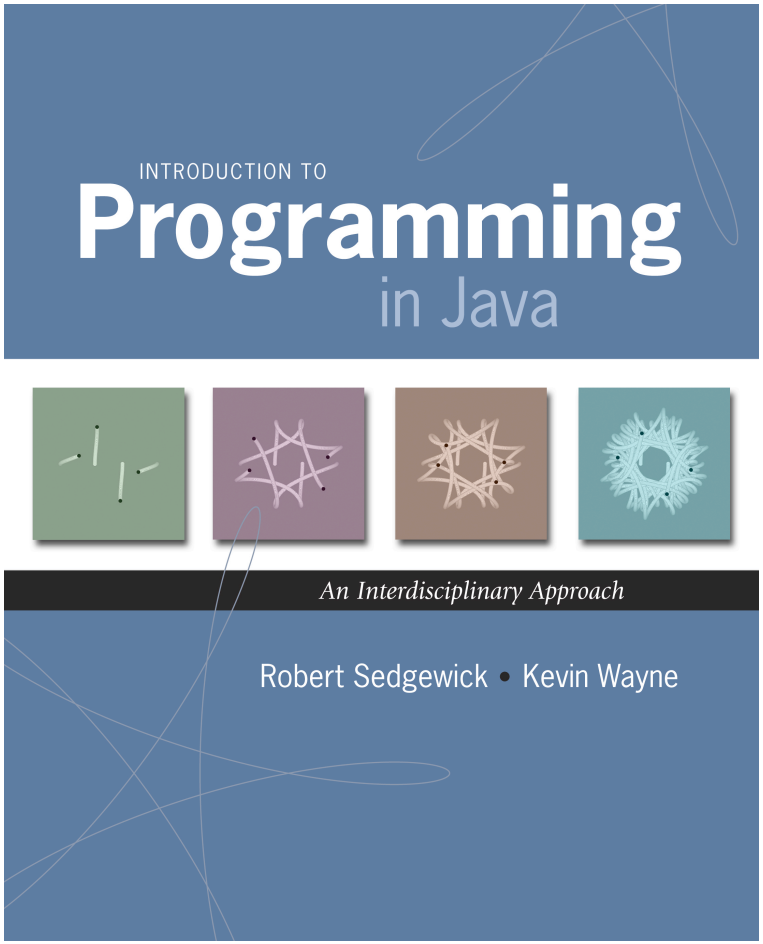
1.1 Your First Program

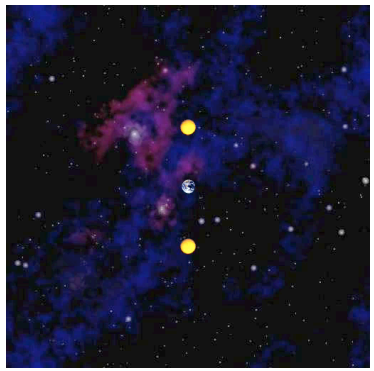


Why Programming?

Why programming? Need to tell computer what you want it to do.

Naive ideal. Natural language instructions.

"Please simulate the motion of these heavenly bodies, subject to Newton's laws of motion and gravity."

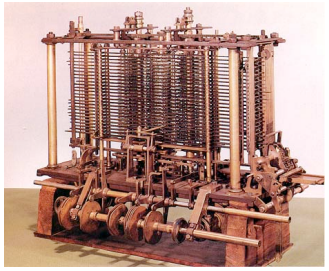


Prepackaged solutions (apps)? Great, when what they do is what you want.

Programming. Enables you to make a computer do anything you want.

well, almost anything
[stay tuned]

Ada Lovelace



Analytic Engine

Languages

Machine languages. Tedious and error-prone.

Natural languages. Ambiguous; can be difficult to parse.

High-level programming languages. Acceptable tradeoff.

> *"Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do."* – *Donald Knuth*

Why Program?

Why program?

- A natural, satisfying and creative experience.
- Enables accomplishments not otherwise possible.
- Opens new world of intellectual endeavor.

First challenge. Learn a programming language.

Next question. Which one?

Naive ideal. A single programming language.

Our Choice: Java

Java features.

- Widely used.
- Widely available.
- Embraces full set of modern abstractions.
- Variety of automatic checks for mistakes in programs.

Java economy.

$100 billion,
5 million developers

- Mars rover.
- Cell phones.
- Blu-ray Disc.
- Web servers.
- Medical devices.
- Supercomputing.
- ...

James Gosling
http://java.net/jag

Why Java?

Java features.

- Widely used.
- Widely available.
- Embraces full set of modern abstractions.
- Variety of automatic checks for mistakes in programs.

Facts of life.

- No language is perfect.
- We need to choose some language.

Our approach.

- Minimal subset of Java.
- Develop general programming skills that are applicable to many languages

It's not about the language!

> *"There are only two kinds of programming languages: those people always [gripe] about and those nobody uses."*
> *– Bjarne Stroustrup*

A Rich Subset of the Java Language

Built-In Types	
`int`	`double`
`long`	`String`
`char`	`boolean`

System
`System.out.println()`
`System.out.print()`
`System.out.printf()`

Math Library	
`Math.sin()`	`Math.cos()`
`Math.log()`	`Math.exp()`
`Math.sqrt()`	`Math.pow()`
`Math.min()`	`Math.max()`
`Math.abs()`	`Math.PI`

Flow Control	
`if`	`else`
`for`	`while`

Parsing
`Integer.parseInt()`
`Double.parseDouble()`

Primitive Numeric Types		
`+`	`-`	`*`
`/`	`%`	`++`
`--`	`>`	`<`
`<=`	`>=`	`==`
`!=`		

Boolean	
`true`	`false`
`||`	`&&`
`!`	

Punctuation	
`{`	`}`
`(`	`)`
`,`	`;`

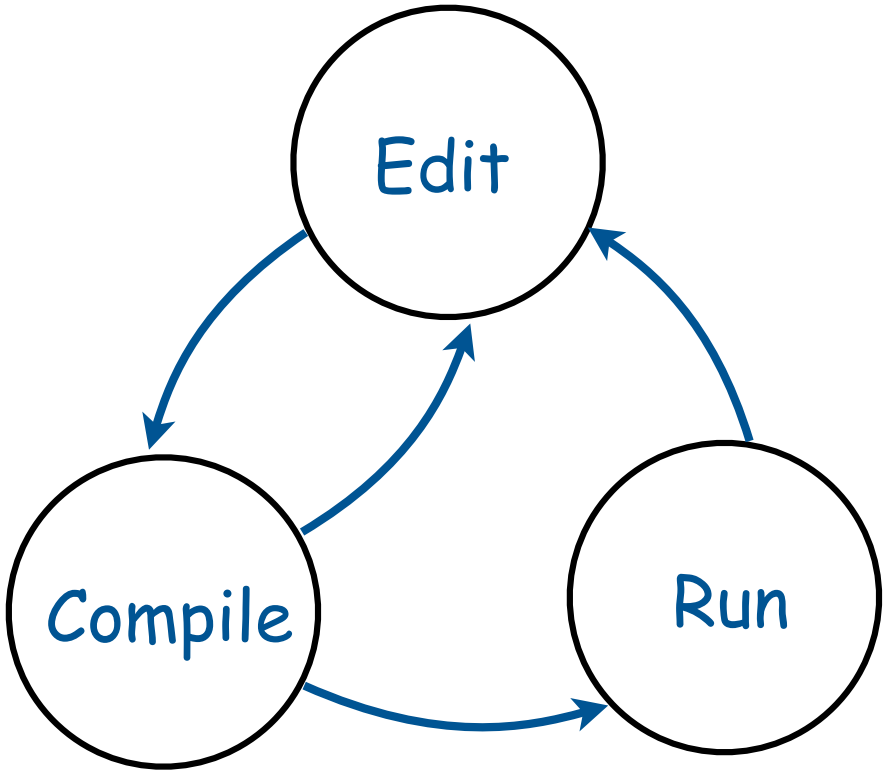
Assignment
`=`

String	
`+`	`""`
`length()`	`compareTo()`
`charAt()`	`matches()`

Arrays
`a[i]`
`new`
`a.length`

Objects	
`class`	`static`
`public`	`private`
`final`	`toString()`
`new`	`main()`

Program Development



Programming in Java.

- Create the program by typing it into a text editor, and save it as HelloWorld.java.

```
/*************************************************
 * Prints "Hello, World"
 * Everyone's first Java program.
 *************************************************/

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

HelloWorld.java

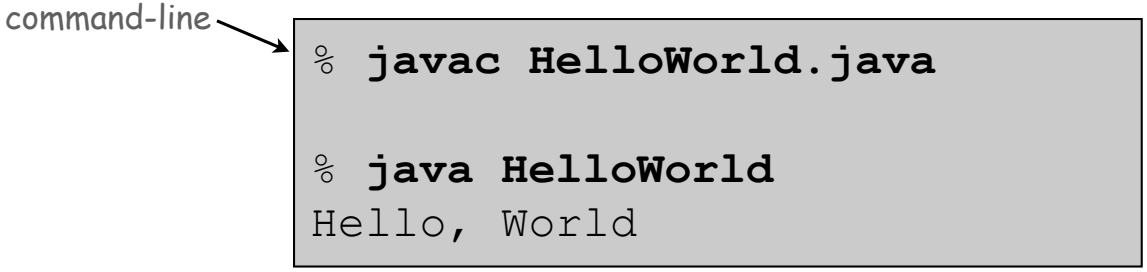
Programming in Java.

- Create the program by typing it into a text editor, and save it as HelloWorld.java.
- Compile it by typing at the command-line:
  javac HelloWorld.java.

- This creates a Java bytecode file named: HelloWorld.class.

Programming in Java.

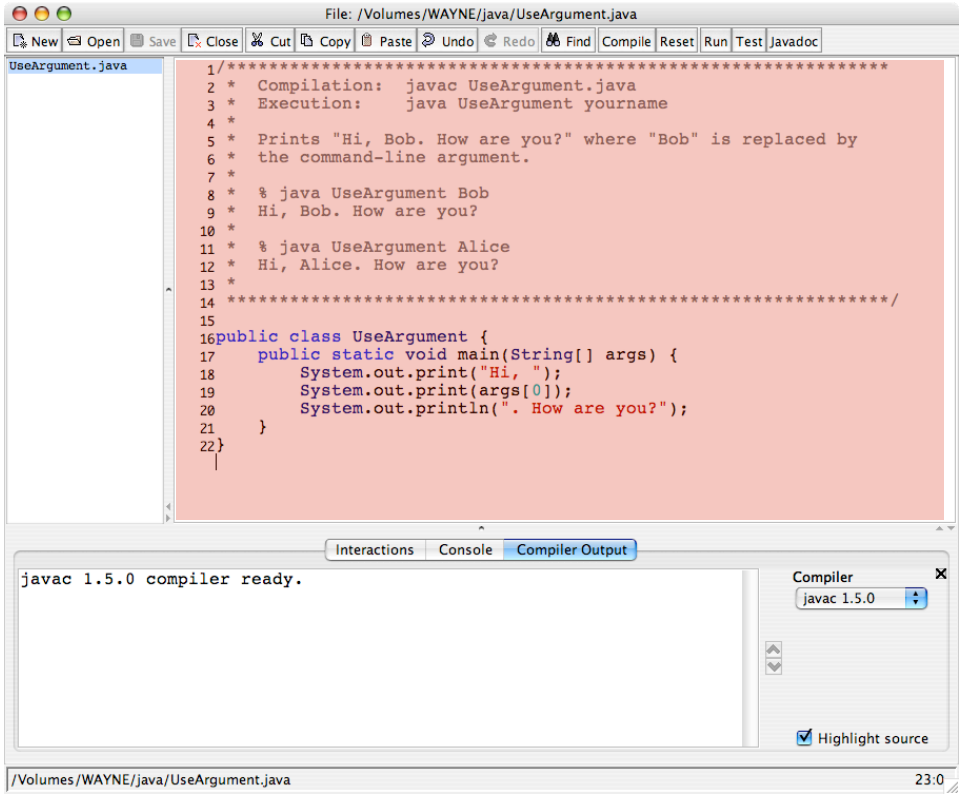
- Create the program by typing it into a text editor, and save it as HelloWorld.java.
- Compile it by typing at the command-line:
  javac HelloWorld.java.
- Execute it by typing at the command-line:
  java HelloWorld.



Program Development (using DrJava)

Program development in Java (using DrJava).

1. Edit your program using the built-in text editor.
2. Compile it to create an executable file.
3. Run your program.

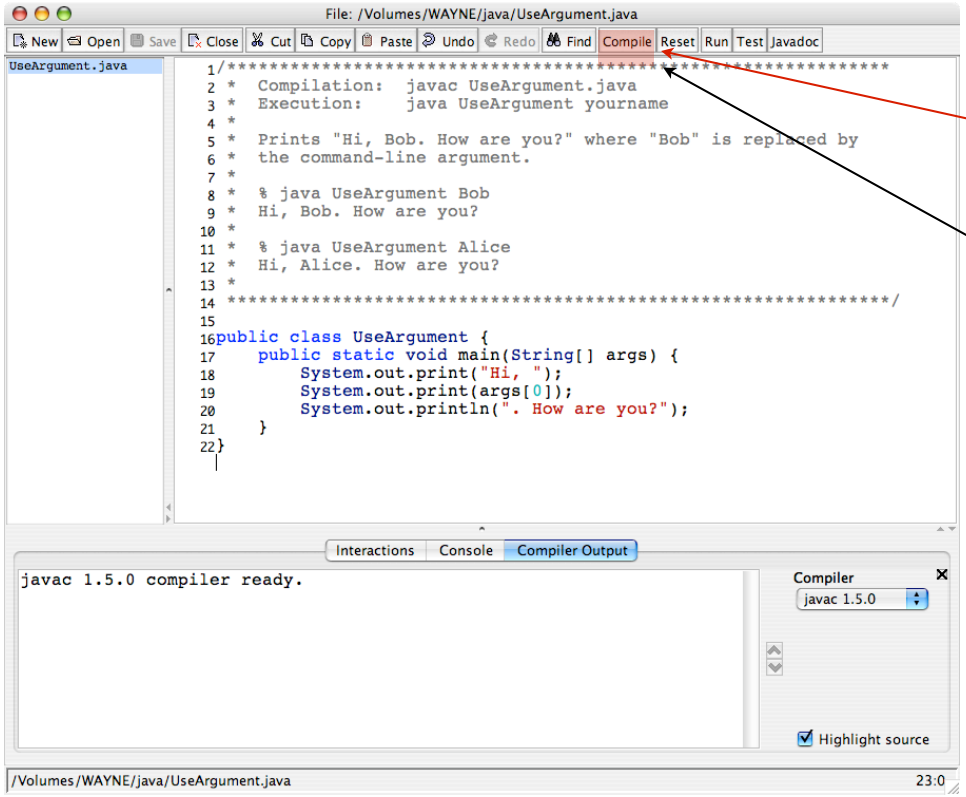


text editor

Program Development (using DrJava)

Program development in Java (using DrJava).

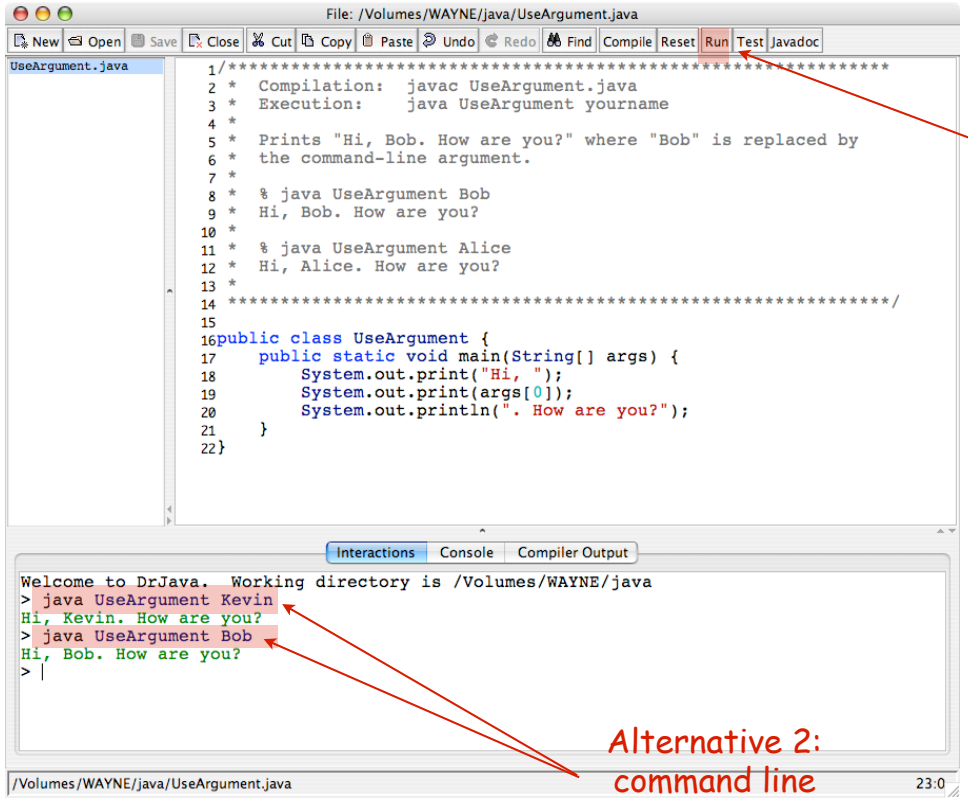
1. Edit your program.
2. Compile it by clicking the "compile" button.
3. Run your program.



Program Development (using DrJava)

Program development in Java (using DrJava).

1. Edit your program.
2. Compile it to create an executable file.
3. Run your program by clicking the "run" button or using the command line.



Alternative 1: run button (OK if no args)

Alternative 2: command line (to provide args)

both use `UseArgument.class`

Note: Program Style

Three versions of the same program.

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

```
/*************************************************************************
 *  Compilation:  javac HelloWorld.java
 *  Execution:    java HelloWorld
 *
 *  Prints "Hello, World". By tradition, this is everyone's first program.
 *
 *  % java HelloWorld
 *  Hello, World
 *
 *************************************************************************/

public class HelloWorld {

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

Fonts, color, comments, and extra space are not relevant to Java.


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

Note: Program Style

Emphasizing consistent style can

- Make it easier to spot errors.
- Make it easier for others to read and use code.
- Enable development environment to provide useful visual cues.

Bottom line for COS 126:

- Let the Doctor indent for you.
- Correct any style problems automatically discovered when you submit.
- Follow your preceptor/grader's advice on style.

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

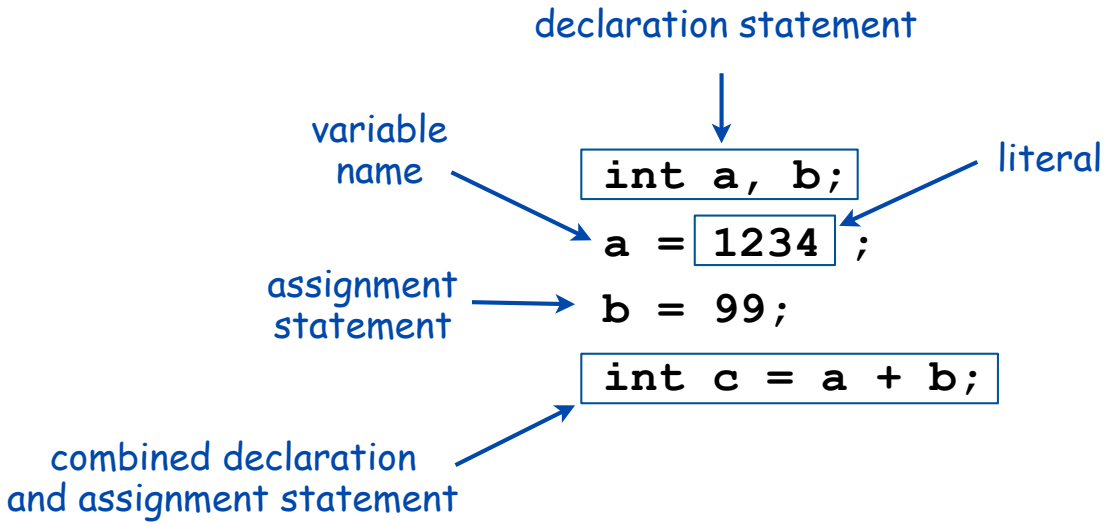
Basic Definitions

Variable. A name that refers to a value.

Literal. Programming-language representation of a value.

Assignment statement. Associates a value with a variable.

Program. Sequence of statements.



Trace

Trace. Table of variable values after each statement.

	a	b	t
`int a, b;`	undefined	undefined	undefined
`a = 1234;`	**1234**	undefined	undefined
`b = 99;`	1234	**99**	undefined
`int t = a;`	1234	99	**1234**
`a = b;`	**99**	99	1234
`b = t;`	99	**1234**	1234

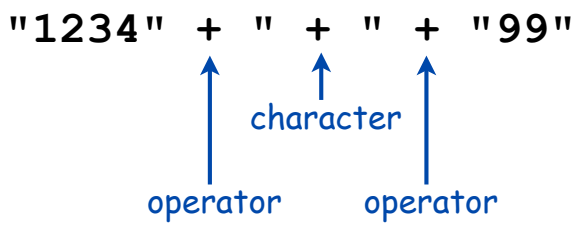
Text

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

values	sequences of characters
typical literals	`"Hello, " "1 " " * "`
operation	concatenate
operator	+

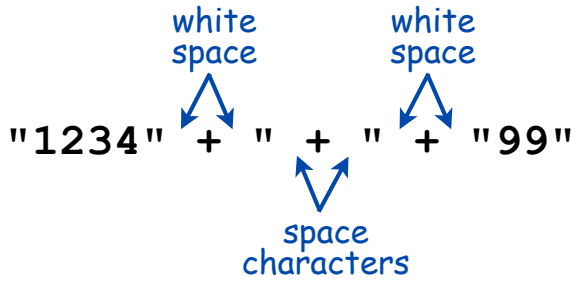
`String` data type

Important note: meaning of characters depends on context!



expression	value
`"Hi, " + "Bob"`	`"Hi, Bob"`
`"1" + " 2 " + "1"`	`"1 2 1"`
`"1234" + " + " + "99"`	`"1234 + 99"`
`"1234" + "99"`	`"123499"`

`String` concatenation examples



Example: 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

`int` data type. Useful for calculations, expressing algorithms.

there is a largest `int` and a smallest `int`

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

`int` data type

expression	value	comment
`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

examples of `int` operations

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

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

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

1234 = 12*99 + 46

Floating-Point Numbers

`double` data type. Useful in scientific applications.

there is a largest `double` and a smallest `double`

values	approximations to real numbers				
typical literals	`3.14159`	`6.022e23`	`-3.0`	`2.0`	`1.414213562373095`
operations	add	subtract	multiply	divide	remainder
operators	`+`	`-`	`*`	`/`	`%`

`double` data type

expression	value	
`3.141 + .03`	`3.171`	
`3.141 - .03`	`3.111`	
`6.02e23/2`	`3.01E+23`	
`5.0 / 3.0`	`1.6666666666666700`	
`10.0 % 3.141`	`0.577`	
`1.0 / 0.0`	`Infinity`	special value
`Math.sqrt(2.0)`	`1.4142135623731000`	
`Math.sqrt(-1.0)`	`NaN`	special value "not a number"

examples of `double` operations

Excerpts from Java's Math Library

public class Math

`double abs(double a)`	absolute value of a
`double max(double a, double b)`	maximum of a and b
`double min(double a, double b)`	minimum of a and b

also defined for `int`, `long`, and `float`

`double sin(double theta)`	sine function
`double cos(double theta)`	cosine function
`double tan(double theta)`	tangent function

inverse functions `asin()`, `acos()`, and `atan()` also available

In radians. Use `toDegrees()` and `toRadians()` to convert.

`double exp(double a)`	exponential (e^a)
`double log(double a)`	natural log ($\log_e a$, or ln a)
`double pow(double a, double b)`	raise a to the bth power (a^b)
`long round(double a)`	found to the nearest integer
`double random()`	random number in [0. 1)
`double sqrt(double a)`	square root of a
`double E`	value of e (constant)
`double PI`	value of p (constant)

Quadratic Equation

Ex. Solve quadratic equation $x^2 + bx + c = 0$.

$$\text{roots} = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$$

```
public class Quadratic
{
   public static void main(String[] args)
   {
      // Parse coefficients from command-line.
      double b = Double.parseDouble(args[0]);
      double c = Double.parseDouble(args[1]);

      // Calculate roots.
      double discriminant = b*b - 4.0*c;
      double d = Math.sqrt(discriminant);
      double root1 = (-b + d) / 2.0;
      double root2 = (-b - d) / 2.0;

      // Print them out.
      System.out.println(root1);
      System.out.println(root2);
   }
}
```

Testing

Testing. Some valid and invalid inputs.

```
% java Quadratic -3.0 2.0
2.0
1.0

% java Quadratic -1.0 -1.0
1.618033988749895
-0.618033988749949

% java Quadratic 1.0 1.0
NaN
NaN

% java Quadratic 1.0 hello
java.lang.NumberFormatException: hello

% java Quadratic 1.0
java.lang.ArrayIndexOutOfBoundsException
```

command-line arguments

golden ratio

"not a number"

$x^2 - 3x + 2$

$x^2 - x - 1$

$x^2 + x + 1$

Booleans

`boolean` data type. Useful to control logic and flow of a program.

values	true or false		
literals	`true`	`false`	
operations	and	or	not
operators	`&&`	`||`	`!`

`boolean` data type

`a`	`!a`
`true`	`false`
`false`	`true`

`a`	`b`	`a && b`	`a || b`
`false`	`false`	`false`	`false`
`false`	`true`	`false`	`true`
`true`	`false`	`false`	`true`
`true`	`true`	`true`	`true`

Truth-table definitions of `boolean` operations

Comparison Operators

Comparison operators.

- Two operands of the same type.
- Result: a value of type `boolean`.

op	meaning	true	false
`==`	equal	`2 == 2`	`2 == 3`
`!=`	not equal	`3 != 2`	`2 != 2`
`<`	less than	`2 < 13`	`2 < 2`
`<=`	less than or equal	`2 <= 2`	`3 <= 2`
`>`	greater than	`13 > 2`	`2 > 13`
`>=`	greater than or equal	`3 >= 2`	`2 >= 3`

comparison operators

non-negative discriminant?	`( b * b - 4.0 * a * c ) >= 0.0`
beginning of a century?	`( year % 100 ) == 0`
legal month?	`( month >= 1 ) && ( month <= 12 )`

comparison examples

Leap Year

Q. Is a given year a leap year?

A. Yes if either (i) divisible by 400 or (ii) divisible by 4 but not 100.

```
public class LeapYear
{
   public static void main(String[] args)
   {
      int year = Integer.parseInt(args[0]);
      boolean isLeapYear;

      // divisible by 4 but not 100
      isLeapYear = (year % 4 == 0) && (year % 100 != 0);

      // or divisible by 400
      isLeapYear = isLeapYear || (year % 400 == 0);


      System.out.println(isLeapYear);
   }
}
```

```
% java LeapYear 2004
true

% java LeapYear 1900
false

% java LeapYear 2000
true
```

Type Conversion

Type conversion. Convert from one type of data to another.

- Automatic (done by Java when no loss of precision; or with strings).
- Explicitly defined by function call.
- Cast (write desired type within parens).

expression	type	value	
`"1234" + 99`	`String`	`"123499"`	automatic
`Integer.parseInt("123")`	`int`	`123`	explicit
`(int) 2.71828`	`int`	`2`	cast
`Math.round(2.71828)`	`long`	`3`	explicit
`(int) Math.round(2.71828)`	`int`	`3`	cast
`(int) Math.round(3.14159)`	`int`	`3`	cast
`11 * 0.3`	`double`	`3.3`	automatic
`(int) 11 * 0.3`	`double`	`3.3`	cast, automatic
`11 * (int) 0.3`	`int`	`0`	cast
`(int) (11 * 0.3)`	`int`	`3`	cast, automatic

Pay attention to the type of your data. ← type conversion can give counterintuitive results but gets easier to understand with practice

Type Conversion Example: Random Integer

Ex. Generate a pseudo-random number between `0` and `N-1`.

```
public class RandomInt
{
   public static void main(String[] args)
   {
      int N = Integer.parseInt(args[0]);
      double r = Math.random();
      int n = (int) (r * N);

      System.out.println("random integer is " + n);
   }
}
```

`String` to `int` (method)

`double` between `0.0` and `1.0`

`double` to `int` (cast)

`int` to `double` (automatic)

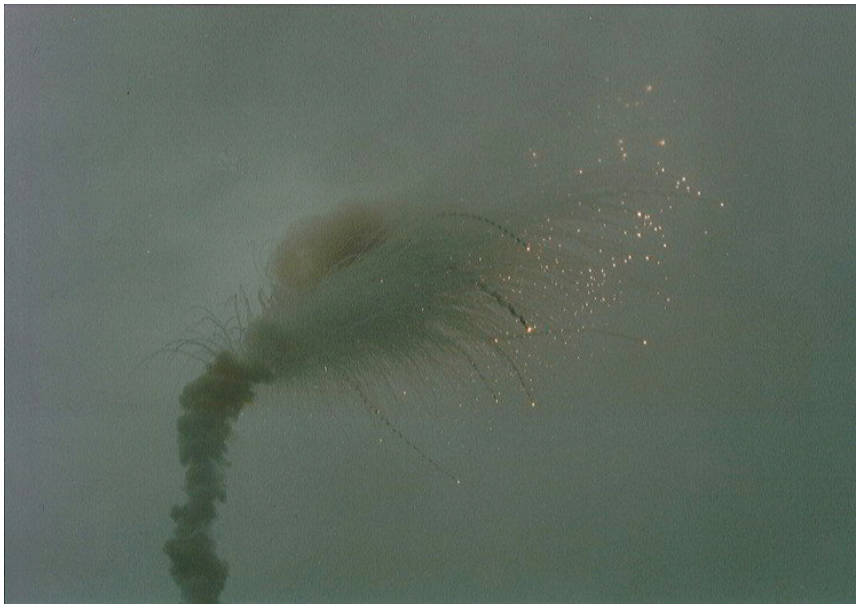
`int` to `String` (automatic)

```
% java RandomInt 6
random integer is 3

% java RandomInt 6
random integer is 0

% java RandomInt 10000
random integer is 3184
```

Summary

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

- `String` text processing, input and output.
- `double`, `int` mathematical calculation.
- `boolean` decision making.



Example of bad type conversion

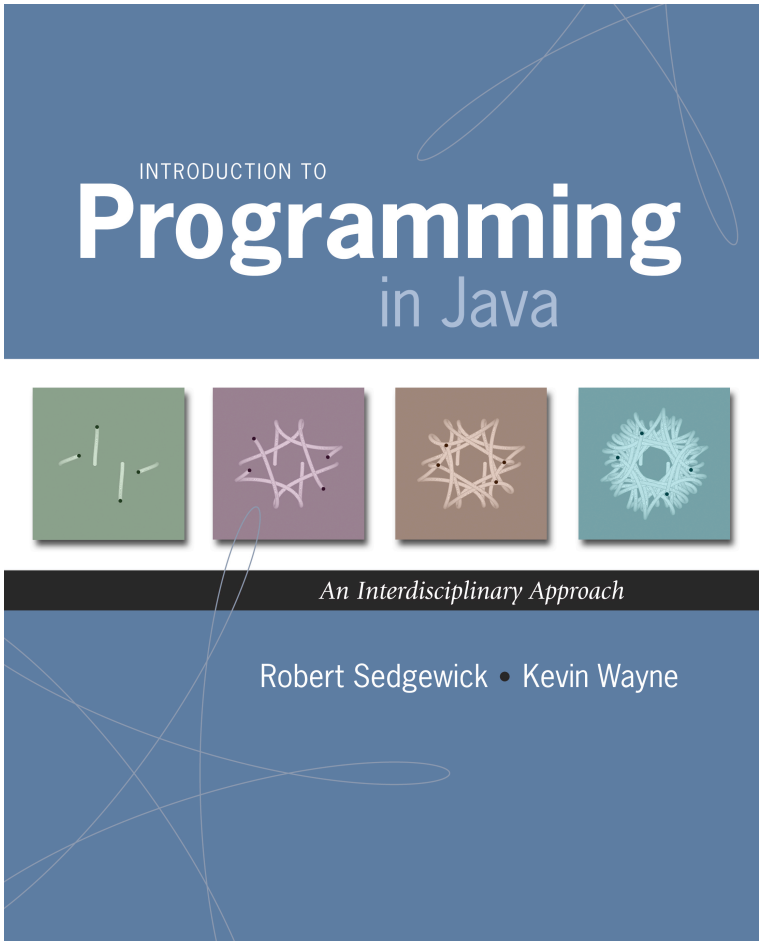
Be aware. In Java you must:

- Declare type of values.
- Convert between types when necessary.

Why do we need types?

- Type conversion must be done at some level.
- Compiler can help do it correctly.
- Example: In 1996, Ariane 5 rocket exploded after takeoff because of bad type conversion.

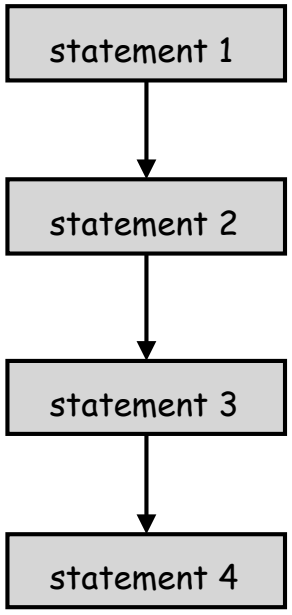
1.3 Conditionals and Loops



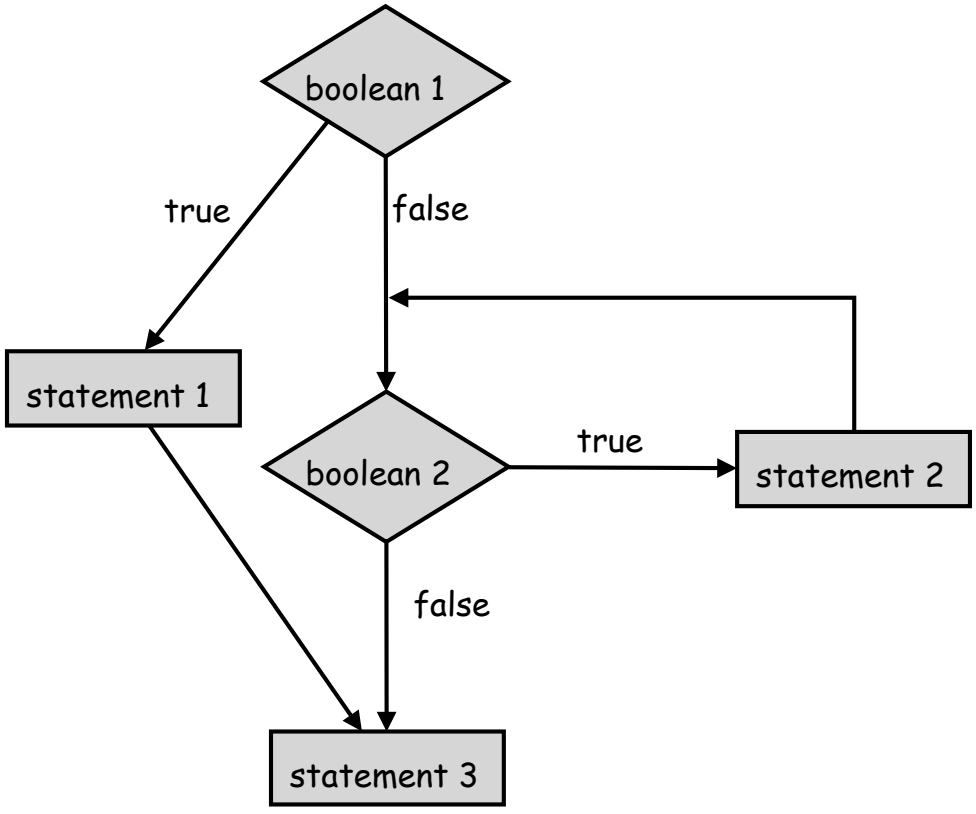
Control Flow

Control flow.

- Sequence of statements that are actually executed in a program.
- Conditionals and loops: enable us to choreograph control flow.



straight-line control flow



control flow with conditionals and loops

Conditionals

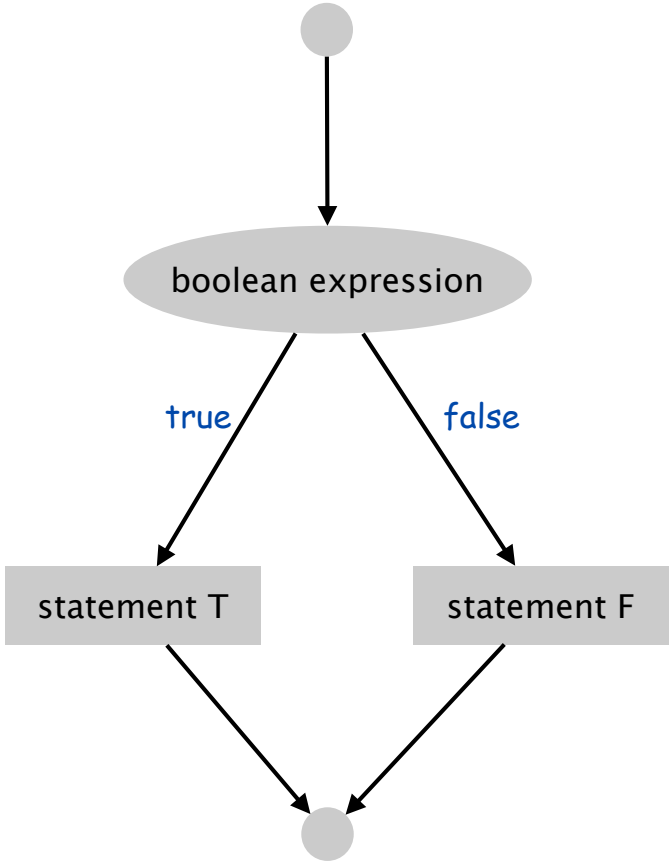
If Statement

The `if` statement. A common branching structure.

- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- If `false`, execute other statements.

```
if (boolean expression) {
    statement T;
}
else {
    statement F;
}
```

can be any sequence of statements

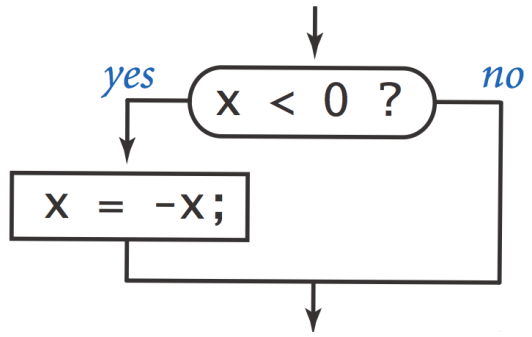


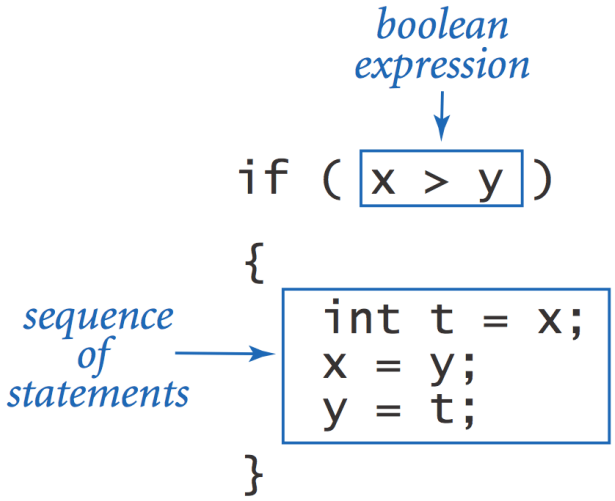
If Statement

The `if` statement. A common branching structure.

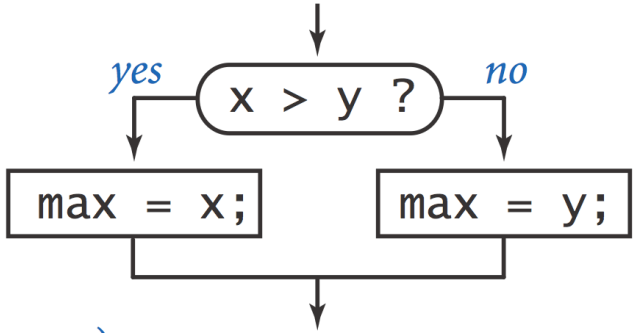
- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- If `false`, execute other statements.

```
if (x < 0) x = -x;
```






```
if (x > y) max = x;
else       max = y;
```



If Statement

Ex. Take different action depending on value of variable.

```
public class Flip {
   public static void main(String[] args) {
      if (Math.random() < 0.5) System.out.println("Heads");
      else                     System.out.println("Tails");
   }
}
```

```
% java Flip
Heads
% java Flip
Heads
% java Flip
Tails

% java Flip
Heads
```

If-Else: Leap Year revisited

If-else. Take different action depending on value of variable.

- If `isLeapYear` is `true`, then print `"is a"`.
- Otherwise, print `"isn't a "`.

```
System.out.print(year + " ");

if (isLeapYear) {
   System.out.print("is a");
}
else {
   System.out.print("isn't a");
}

System.out.println(" leap year");
```

If-Else: Leap Year revisited

```
public class LeapYear
{
   public static void main(String[] args)
   {
      int year = Integer.parseInt(args[0]);
      boolean isLeapYear;

      // divisible by 4 but not 100
      isLeapYear = (year % 4 == 0) && (year % 100 != 0);

      // or divisible by 400
      isLeapYear = isLeapYear || (year % 400 == 0);

      System.out.print(year + " ");

      if (isLeapYear) {
          System.out.print("is a");
      }
      else {
          System.out.print("isn't a");
      }

      System.out.println(" leap year");
   }
}
```

```
% java LeapYear 2004
2004 is a leap year

% java LeapYear 1900
1900 isn't a leap year

% java LeapYear 2000
2000 is a leap year
```

Oblivious Sorting

Sort. Read in 3 integers and rearrange them in ascending order.

```
public class Sort3 {
   public static void main(String[] args) {

      int a = Integer.parseInt(args[0]);
      int b = Integer.parseInt(args[1]);
      int c = Integer.parseInt(args[2]);

      if (b > c) { int t = b; b = c; c = t; }
      if (a > b) { int t = a; a = b; b = t; }
      if (b > c) { int t = b; b = c; c = t; }

      System.out.println(a + " " + b + " " + c);
   }
}
```

read in 3 integers from command-line

swap b and c

swap a and b

swap b and c

```
% java Sort3 9 8 7
7 8 9

% java Sort3 2 1 7
1 2 7
```

Puzzle 1. Sort 4 integers with 5 compare-exchanges.

Puzzle 2. Sort 6 integers with 12.

Contents

The latest additions are shown in **bold**.

The Hello World Collection

"Hello World" is the first program one usually writes when learning a new programming language. The first Hello World program appeared in chapter 1.1 of the first edition of Kernighan & Ritchie's original book about C, "The C Programming Language", in 1978 and read like this:

```
main() {
    printf("hello, world\n");
}
```

Since then, Hello World has been implemented in just about every programming language on the planet. This collection includes **441 Hello World programs** in many more-or-less well known programming languages, plus **64 human languages**.

The programs in this collection are intended to be as minimal as possible in the respective language. They are meant to demonstrate how to output Hello World as simply as possible, not to show off language features. For a collection of programs that tell more about what programming in the languages actually is like, have a look at the 99 Bottles of Beer collection.

The Collection was compiled by Wolfram Rösler with help from many people around the world. It was started on 3-Oct-1994, put on the Internet on 30-Dec-1999, exceeded 200 entries on 14-Jul-2005, 300 on 6-Dec-2006, and 400 on 27-Jul-2008. It is now probably the biggest collection of Hello World programs on the Internet, and the only one collecting human languages as well. It is administered as a bunch of text files which are compiled into this single HTML file by a bash script executed under the Cygwin environment, run on Windows.

Click here for a list of all contributors.
Click here for related links.

By the way, this site is the original Hello World Collection. Throughout the Web, you can find many copies of various versions of this file, not all of which give proper credit to those who compiled and contributed to it over the years.

Last update: Sep 25, 2010.

Index

The latest additions are shown in **bold**.