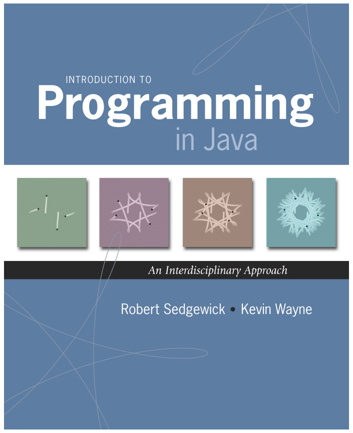
1.1 Your First Program



INTRODUCTION TO

Programming in Java

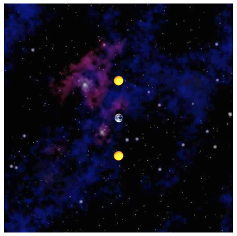
An Interdisciplinary Approach

Robert Sedgewick • Kevin Wayne

*Introduction to Programming in Java: An Interdisciplinary Approach · Robert Sedgewick and Kevin Wayne · Copyright © 2008 · * **

Why Programming?

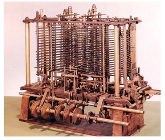
Idealized computer. "Please simulate the motion of a system of N heavenly bodies, subject to Newton's laws of motion and gravity."



Prepackaged software solutions. Great, if it does exactly what you want.

Computer programming. Art of making a computer do what you want.

Ada Lovelace



Analytic Engine

Languages

Machine languages. Tedious and error-prone.

Natural languages. Ambiguous and hard for computer to parse.

Kids Make Nutritious Snacks.
Red Tape Holds Up New Bridge.
Police Squad Helps Dog Bite Victim.
Local High School Dropouts Cut in Half.

[real newspaper headlines, compiled by Rich Pattis]

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*

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 perfect language.
- We need to choose some language.

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

Our approach.

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

It's not about the language!

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
toString()	equals()
new	main()

Create, Compile, Execute

Programming in Java

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

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`

command-line

```
% javac HelloWorld.java
```

(or click the Compile button in DrJava)

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

Programming in 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`
- Execute it by typing at the command-line:
  `java HelloWorld`

command-line

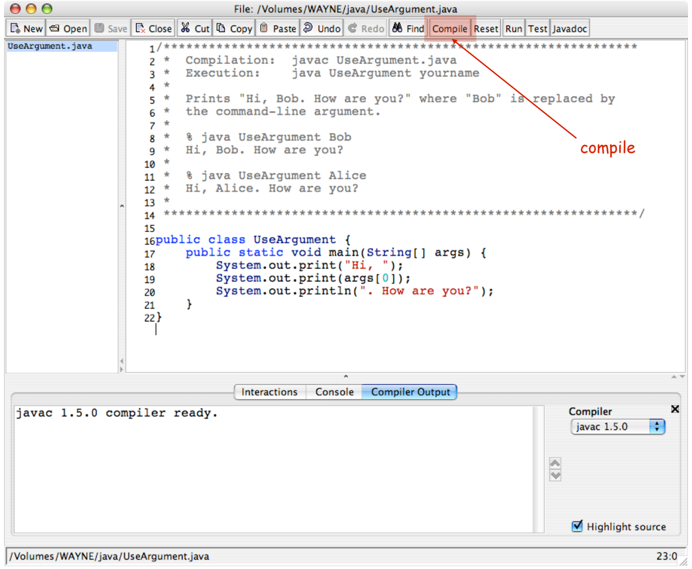
```
% javac HelloWorld.java

% java HelloWorld
Hello, World
```

(or click the Run button in DrJava)

Dr. Java

http://drjava.org

Dr. Java



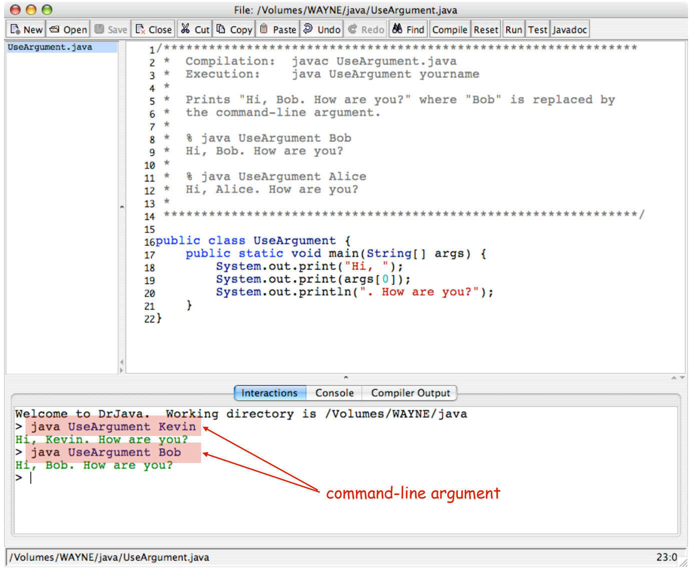
File: /Volumes/WAYNE/java/UseArgument.java

```
/****************************************************************
 *  Compilation:  javac UseArgument.java
 *  Execution:    java UseArgument yourname
 *
 *  Prints "Hi, Bob. How are you?" where "Bob" is replaced by
 *  the command-line argument.
 *
 *  % java UseArgument Bob
 *  Hi, Bob. How are you?
 *
 *  % java UseArgument Alice
 *  Hi, Alice. How are you?
 *
 ****************************************************************/

public class UseArgument {
    public static void main(String[] args) {
        System.out.print("Hi, ");
        System.out.print(args[0]);
        System.out.println(". How are you?");
    }
}
```

```
javac 1.5.0 compiler ready.
```

/Volumes/WAYNE/java/UseArgument.java

Dr. Java

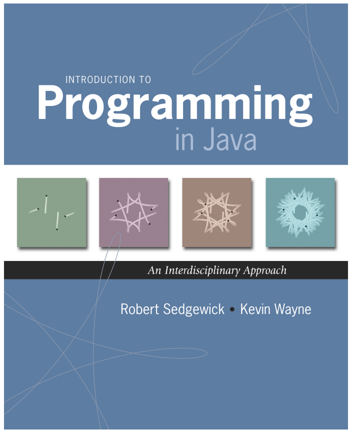


```
/**************************************************************
 *  Compilation:  javac UseArgument.java
 *  Execution:    java UseArgument yourname
 *
 *  Prints "Hi, Bob. How are you?" where "Bob" is replaced by
 *  the command-line argument.
 *
 *  % java UseArgument Bob
 *  Hi, Bob. How are you?
 *
 *  % java UseArgument Alice
 *  Hi, Alice. How are you?
 *
 **************************************************************/

public class UseArgument {
    public static void main(String[] args) {
        System.out.print("Hi, ");
        System.out.print(args[0]);
        System.out.println(". How are you?");
    }
}
```

```
Welcome to DrJava.  Working directory is /Volumes/WAYNE/java
> java UseArgument Kevin
Hi, Kevin. How are you?
> java UseArgument Bob
Hi, Bob. How are you?
> |
```

command-line argument

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

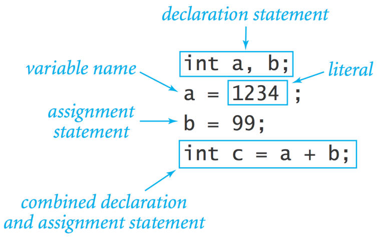
`int`	integers	`17`
		`12345`
			add, subtract, multiply, divide

`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.



```
int a, b;
a = 1234 ;
b = 99;
int c = a + b;
```

Trace

Trace. Table of variable values after each statement.

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

Text

Text

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

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

expression	*value*
`"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);
   }
}
```

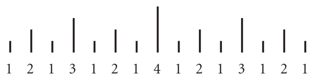
string concatenation

"1"

"1 2 1"

"1 2 1 3 1 2 1"

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

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

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
```

1234 = 12*99 + 46

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

Floating-Point Numbers

Floating-Point Numbers

double data type. Useful in scientific applications.

values	approximations to real numbers				
typical literals	3.14159	6.022e23	-3.0	2.0	1.4142135623730951
operations	add	subtract	multiply	divide	
operators	+	-	*	/	

expression	*value*
3.141 + .03	3.171
3.141 - .03	3.111
6.02e23 / 2	3.01e23
5.0 / 3.0	1.6666666666666667
10.0 % 3.141	0.577
1.0 / 0.0	Infinity
Math.sqrt(2.0)	1.4142135623730951
Math.sqrt(-1.0)	NaN

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*
Note 1: abs(), max(), and min() are defined also for int, long, and float.		
double	sin(double theta)	*sine function*
double	cos(double theta)	*cosine function*
double	tan(double theta)	*tangent function*
Note 2: Angles are expressed in radians. Use toDegrees() and toRadians() to convert. *Note 3: Use asin(), acos(), and atan() for inverse functions.*		
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)	*round 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* π *(constant)*

See booksite for other available functions.

Excerpts from Java's mathematics library

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                        command-line arguments
1.0

% java Quadratic -1.0 -1.0
1.618033988749895            golden ratio
-0.6180339887498949

% java Quadratic 1.0 1.0
NaN       not a number
NaN

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

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

$x^2 - 3x + 2$

$x^2 - x - 1$

$x^2 + x + 1$

Booleans

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	&&	||	!

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*

Comparisons

Comparisons. Take operands of one type and produce an operand 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

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

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

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

- Automatic: no loss of precision; or with strings.
- Explicit: cast; or method.

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

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.
`double`, `int`	mathematical calculation.
`boolean`	decision making.

Be aware.

- Declare type of values.
- Convert between types when necessary.
- In 1996, Ariane 5 rocket exploded after takeoff because of bad type conversion.