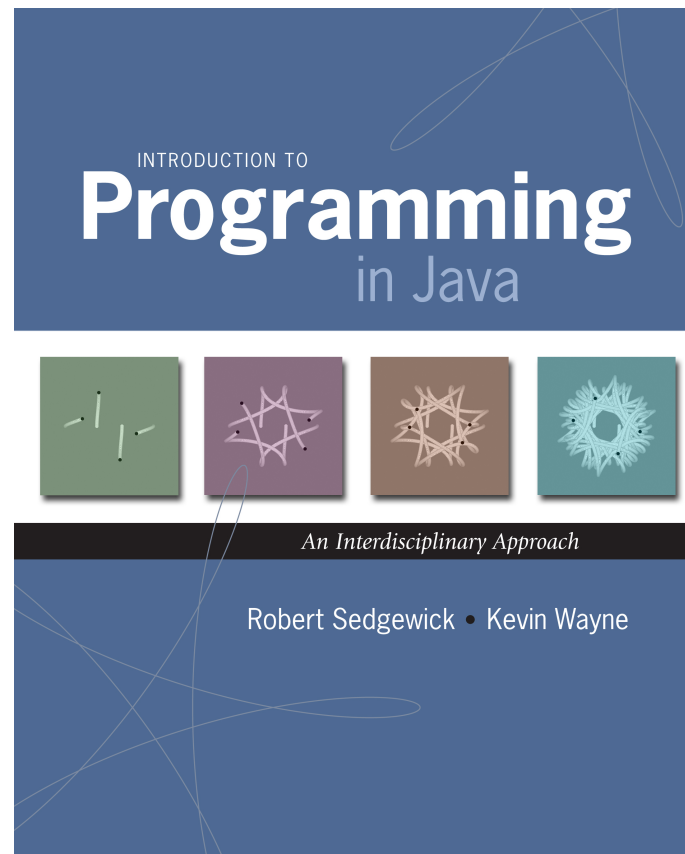


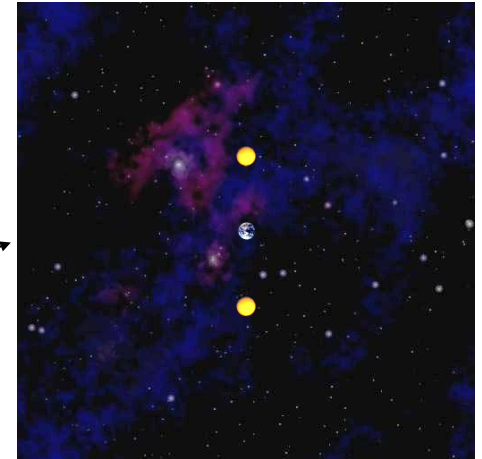
1.1 Your First Program



Why Programming?

Why programming? Need to tell computer what to do.

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



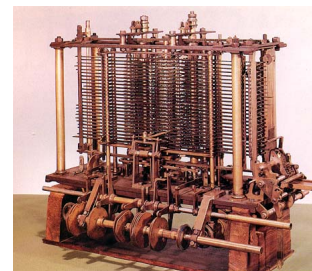
Prepackaged software solutions. Great, they do exactly what you want.



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



Ada Lovelace



Analytic Engine

well, almost anything
[stay tuned]

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



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.

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

↖
\$100 billion,
5 million developers



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	
<code>int</code>	<code>double</code>
<code>long</code>	<code>String</code>
<code>char</code>	<code>boolean</code>

System
<code>System.out.println()</code>
<code>System.out.print()</code>
<code>System.out.printf()</code>

Math Library	
<code>Math.sin()</code>	<code>Math.cos()</code>
<code>Math.log()</code>	<code>Math.exp()</code>
<code>Math.sqrt()</code>	<code>Math.pow()</code>
<code>Math.min()</code>	<code>Math.max()</code>
<code>Math.abs()</code>	<code>Math.PI</code>

Flow Control	
<code>if</code>	<code>else</code>
<code>for</code>	<code>while</code>

Parsing
<code>Integer.parseInt()</code>
<code>Double.parseDouble()</code>

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

Boolean	
<code>true</code>	<code>false</code>
<code> </code>	<code>&&</code>
<code>!</code>	

Punctuation	
<code>{</code>	<code>}</code>
<code>(</code>	<code>)</code>
<code>,</code>	<code>;</code>

Assignment
<code>=</code>

String	
<code>+</code>	<code>""</code>
<code>length()</code>	<code>compareTo()</code>
<code>charAt()</code>	<code>matches()</code>

Arrays
<code>a[i]</code>
<code>new</code>
<code>a.length</code>

Objects	
<code>class</code>	<code>static</code>
<code>public</code>	<code>private</code>
<code>final</code>	<code>toString()</code>
<code>new</code>	<code>main()</code>

Hello, World



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

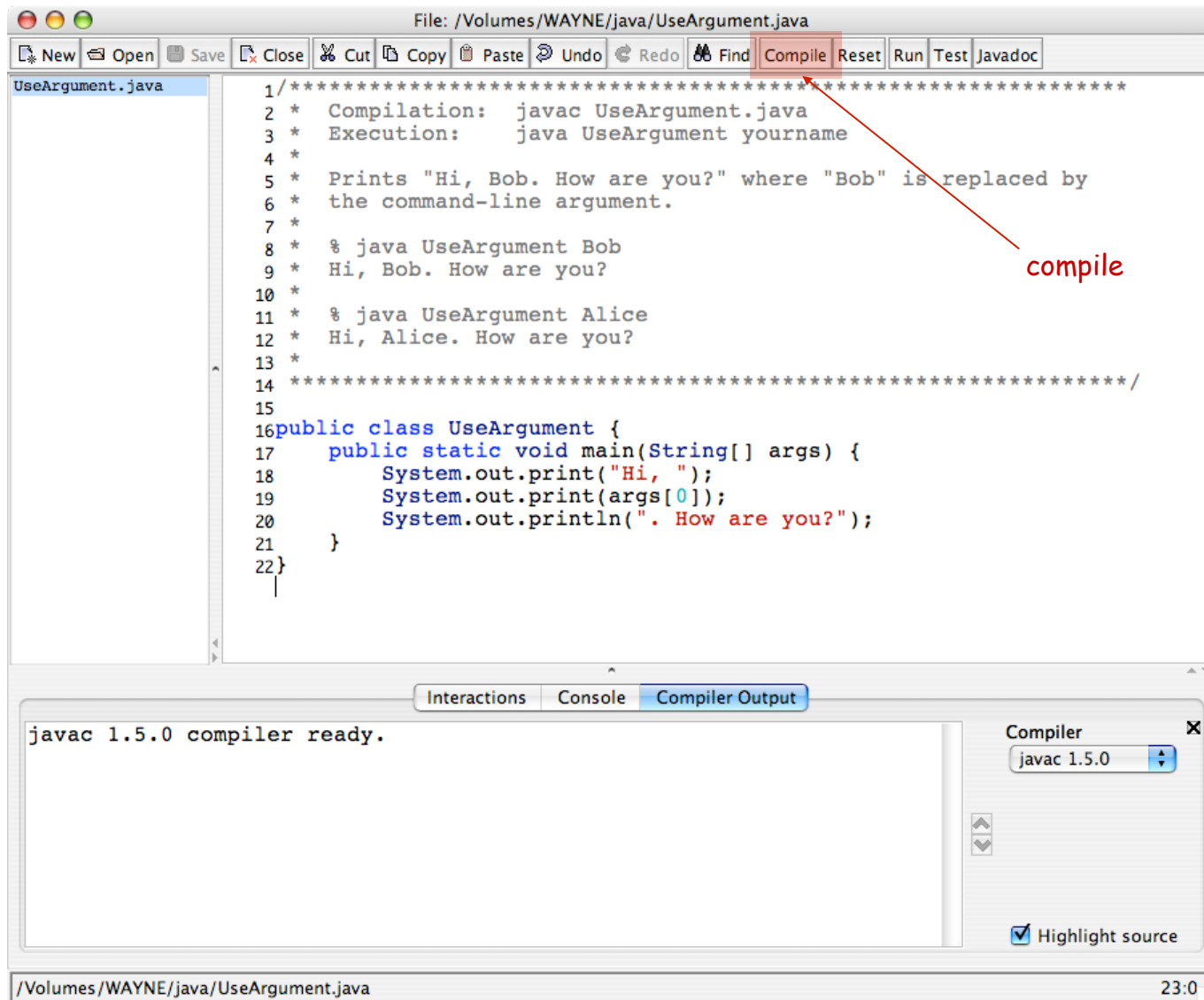
% java HelloWorld
Hello, World
```

Dr. Java



<http://drjava.org>

Dr. Java



Dr. Java

The screenshot shows the Dr. Java IDE interface. The top window displays the source code for `UseArgument.java`. The code includes a multi-line comment explaining the program's purpose and usage, followed by a `public class UseArgument` with a `main` method. The `main` method prints "Hi, " followed by the first command-line argument.

```
1/*****
2 *  Compilation:  javac UseArgument.java
3 *  Execution:    java UseArgument yourname
4 *
5 *  Prints "Hi, Bob. How are you?" where "Bob" is replaced by
6 *  the command-line argument.
7 *
8 *  % java UseArgument Bob
9 *  Hi, Bob. How are you?
10 *
11 *  % java UseArgument Alice
12 *  Hi, Alice. How are you?
13 *
14 *****/
15
16 public class UseArgument {
17     public static void main(String[] args) {
18         System.out.print("Hi, ");
19         System.out.print(args[0]);
20         System.out.println(". How are you?");
21     }
22 }
```

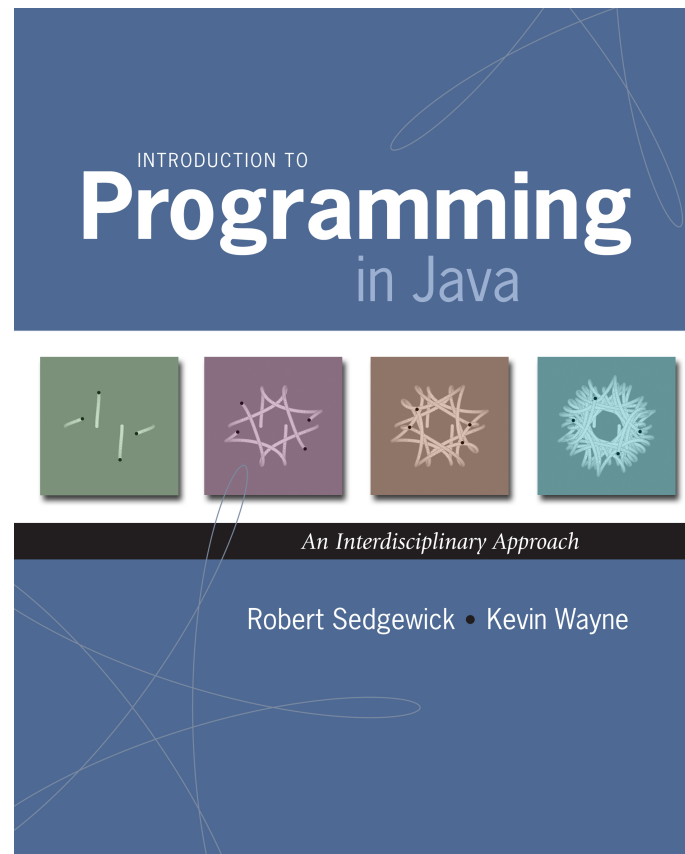
The bottom window shows the 'Interactions' tab, which contains the output of the program. It displays the welcome message and the results of two `java` commands. Red arrows point from the text 'command-line argument' to the arguments 'Kevin' and 'Bob' in the commands.

```
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

The status bar at the bottom shows the file path `/Volumes/WAYNE/java/UseArgument.java` and the time `23:0`.

1.2 Built-in Types of Data



Built-in Data Types

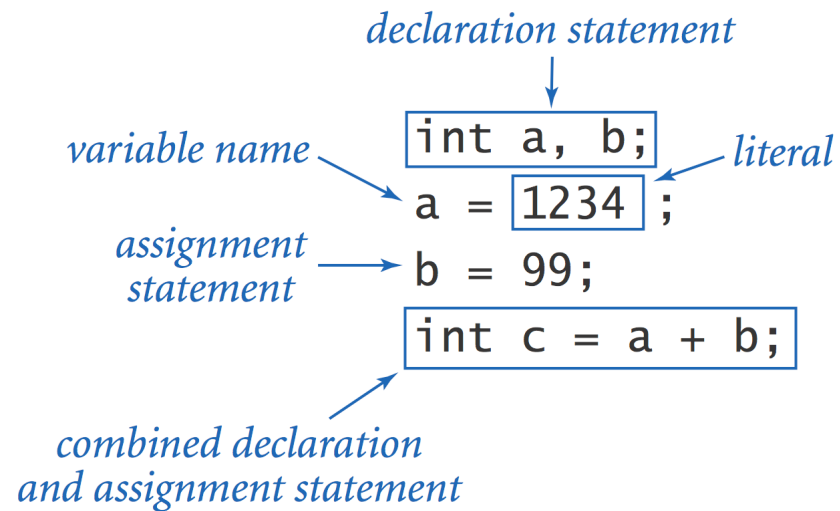
Data type. A set of values and operations defined on those values.

type	set of values	literal values	operations
char	characters	'A' '@'	compare
String	sequences of characters	"Hello World" "126 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.

	a	b	t
int a, b;	<i>undefined</i>	<i>undefined</i>	
a = 1234;	1234	<i>undefined</i>	
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.

<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"

Caveat. Meaning of characters depends on context.

"1234" + " " + " " + "99"

operator character operator

"1234" + " " + " " + "99"

white space white space space characters

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.

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

<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
3 - (5 - 2)	0	unambiguous

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

$$1234 = 12 * 99 + 46$$

Floating-Point Numbers

Floating-Point Numbers

double data type. Useful in scientific applications.

<i>values</i>	real numbers (specified by IEEE 754 standard)			
<i>typical literals</i>	3.14159	6.022e23	-3.0	2.0 1.4142135623730951
<i>operations</i>	add	subtract	multiply	divide
<i>operators</i>	+	-	*	/

<i>expression</i>	<i>value</i>
3.141 + .03	3.171
3.141 - .03	3.111
6.02e23 / 2.0	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

Excerpts from Java's Math Library

```
public class Math
```

<code>double abs(double a)</code>	<i>absolute value of a</i>
<code>double max(double a, double b)</code>	<i>maximum of a and b</i>
<code>double min(double a, double b)</code>	<i>minimum of a and b</i>

Note 1: `abs()`, `max()`, and `min()` are defined also for `int`, `long`, and `float`.

<code>double sin(double theta)</code>	<i>sine function</i>
<code>double cos(double theta)</code>	<i>cosine function</i>
<code>double tan(double theta)</code>	<i>tangent function</i>

Note 2: Angles are expressed in radians. Use `toDegrees()` and `toRadians()` to convert.

Note 3: Use `asin()`, `acos()`, and `atan()` for inverse functions.

<code>double exp(double a)</code>	<i>exponential (e^a)</i>
<code>double log(double a)</code>	<i>natural log ($\log_e a$, or $\ln a$)</i>
<code>double pow(double a, double b)</code>	<i>raise a to the bth power (a^b)</i>

<code>long round(double a)</code>	<i>round to the nearest integer</i>
<code>double random()</code>	<i>random number in $[0, 1)$</i>
<code>double sqrt(double a)</code>	<i>square root of a</i>

<code>double E</code>	<i>value of e (constant)</i>
<code>double PI</code>	<i>value of π (constant)</i>

<http://java.sun.com/javase/6/docs/api/java/lang/Math.html>

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
```

← command-line arguments

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

```
% java Quadratic -1.0 -1.0
```

```
1.618033988749895
```

```
-0.6180339887498949
```

← golden ratio

$$x^2 - x - 1$$

```
% java Quadratic 1.0 1.0
```

```
NaN
```

```
NaN
```

← not a number

$$x^2 + x + 1$$

```
% java Quadratic 1.0 hello
```

```
java.lang.NumberFormatException: hello
```

```
% java Quadratic 1.0
```

```
java.lang.ArrayIndexOutOfBoundsException
```

Booleans

Booleans

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

<i>values</i>	true or false		
<i>literals</i>	true false		
<i>operations</i>	and	or	not
<i>operators</i>	&&		!

<i>a</i>	<i>!a</i>	<i>a</i>	<i>b</i>	<i>a && b</i>	<i>a b</i>
true	false	false	false	false	false
false	true	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`.

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

non-negative discriminant?

`(b*b - 4.0*a*c) >= 0.0`

beginning of a century?

`(year % 100) == 0`

legal month?

`(month >= 1) && (month <= 12)`

Typical comparison expressions

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.

<i>expression</i>	<i>expression type</i>	<i>expression value</i>
"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.

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.



example of bad type conversion