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,
program a computer,
cook a tasty meal,
fight efficiently, and
die gallantly.
Specialization is for insects.

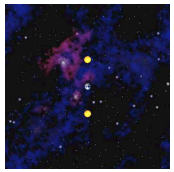
Robert A. Heinlein
Time Enough for Love (1973)

Why Programming?

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

Naive ideal. Natural language instructions.

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

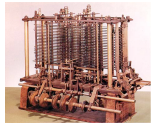


Prepackaged solutions (apps)? Great, when what they do is what you want.

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

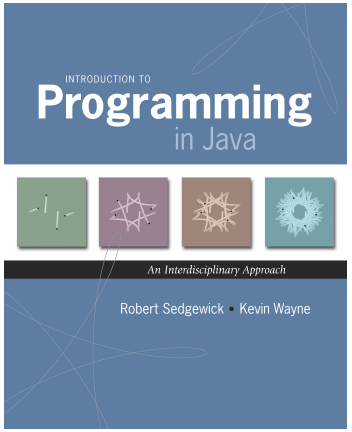
well, almost anything [stay tuned]

Ada Lovelace



Analytic Engine

1.1 Your First Program



Languages

Machine languages. Tedious and error-prone.

Natural languages. Ambiguous; can be difficult 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. ← $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

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

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

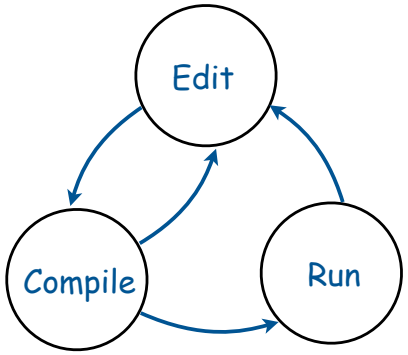
Assignment
`=`

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

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

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

Program Development



Program development in Java (bare-bones)

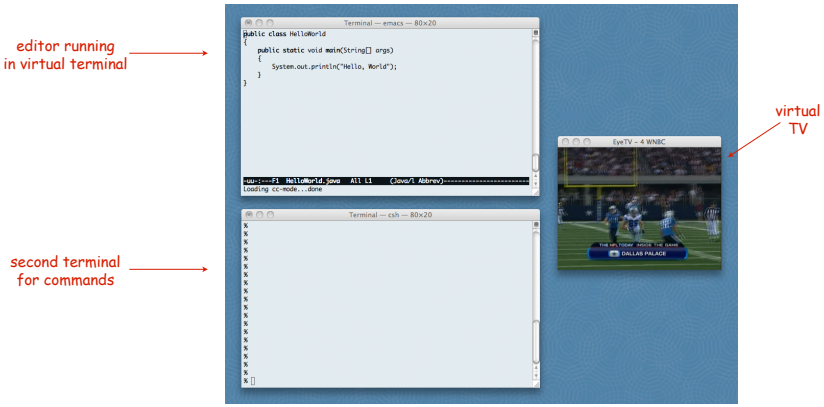
1. Edit your program.
   - Use a text editor.
   - Result: a text file such as `HelloWorld.java`.

2. Compile it to create an executable file.
   - Use the Java compiler
   - Result: a Java bytecode file file such as `HelloWorld.class`
   - Mistake? Go back to 1. to fix and recompile.

3. Run your program.
   - Use the Java runtime.
   - Result: your program's output.
   - Mistake? Go back to 1. to fix, recompile, and execute

Program development in Java (using virtual terminals).

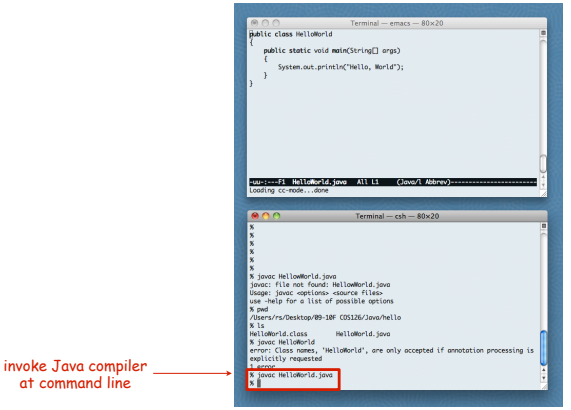
1. Edit your program using any text editor.
2. Compile it to create an executable file.
3. Run your program.



Program development in Java (using virtual terminals).

1. Edit your program.
2. Compile it by typing `javac HelloWorld.java` at the command line.
3. Run your program.

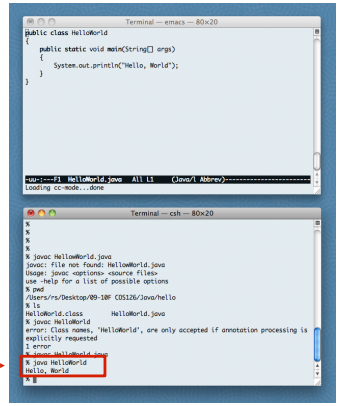
creates `HelloWorld.class`



Program Development (virtual terminals)

Program development in Java (using virtual terminals).

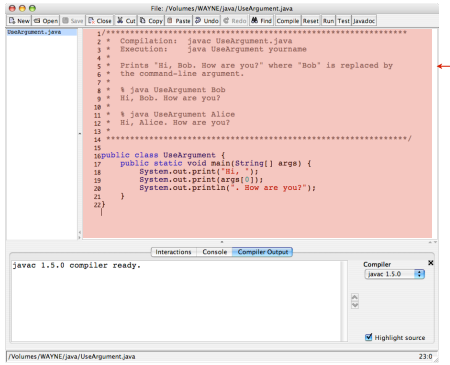
1. Edit your program.
2. Compile it to create an executable file.
3. Run your program by typing `java HelloWorld` at the command line.



Program Development (using DrJava)

Program development in Java (using DrJava).

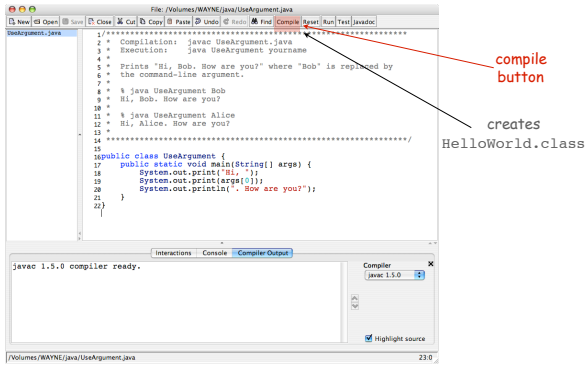
1. Edit your program using the built-in text editor.
2. Compile it to create an executable file.
3. Run your program.



Program Development (using DrJava)

Program development in Java (using DrJava).

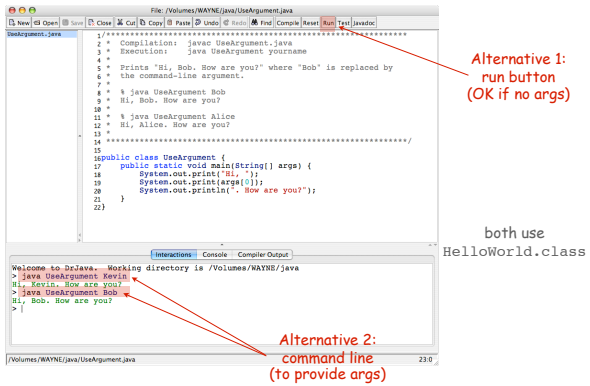
1. Edit your program.
2. Compile it by clicking the "compile" button.
3. Run your program.



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.



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");
    }
}
```

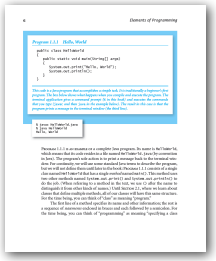
Fonts, color, comments, and extra space are not relevant to Java.

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


Note: Program Style

Different styles are appropriate in different contexts.

- DrJava
- Booksite
- Book
- COS 126 assignment



Enforcing consistent style can

- Stifle creativity.
- Confuse style rules with language rules.

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: Life is easiest if you use DrJava style.

99% of program development

Debugging. Cyclic process of editing, compiling, and fixing mistakes (bugs).
You will make many mistakes as you write programs. It's normal.

> *As soon as we started programming, we found out to our surprise that it wasn't as easy to get programs right as we had thought. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs.* – Maurice Wilkes

Program Development Environment. Software to support cycle of editing to fix mistakes, compiling programs, running programs, and examining output.

Examples: Terminal/editor, DrJava.

Naive ideal. "Please compile, execute, and debug my program".

Bad news. Even a computer can't find all the mistakes in your program.

profound idea
[stay tuned]

TEQ on Program Development

[easy if you did Exercise 1.1.2]

How do you cope with the following error messages?

A.
```
% javac HelloWorld.java
% java HelloWorld.java
Main method not public.
```

B.
```
% javac HelloWorld.java
HelloWorld.java:3: invalid method declaration; return type required
    public static main(String[] args)
                  ^
```

Program Development Environments: A Short History

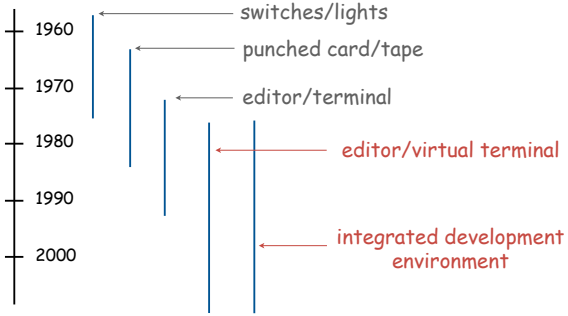
Historical context is important in computer science

- We regularly use old software.
- We regularly emulate old hardware.
- We depend upon old concepts and designs.

First requirement in any computer system: program development

Widely-used methods:

- switches/lights
- punched cards
- terminal
- editor/virtual terminal
- IDE



Switches and Lights

Use switches to enter binary program code, lights to read results

PDP-8, circa 1970

Punched Cards/Line Printer

Use punched cards for program code, line printer for output

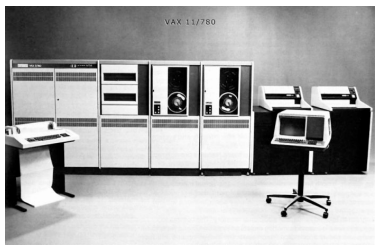
IBM System 360, circa 1975

Timesharing Terminal

Use terminal for editing program, reading output, and controlling computer

VT-100 terminal

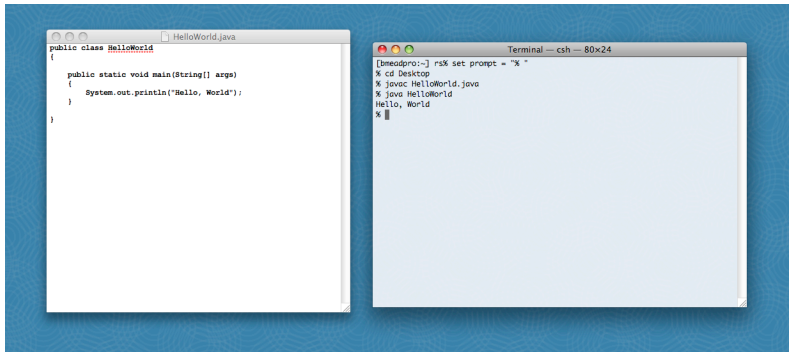
VAX 11/780 circa 1977



Timesharing: allowed many people to simultaneously use a single machine.

Editor and Virtual Terminal on a Personal Computer

Use an editor to create and make changes to the program text.
Use a virtual terminal to invoke the compiler and run the executable code.



Pros:

- Works with any language.
- Useful for other tasks.
- Used by professionals.

Cons:

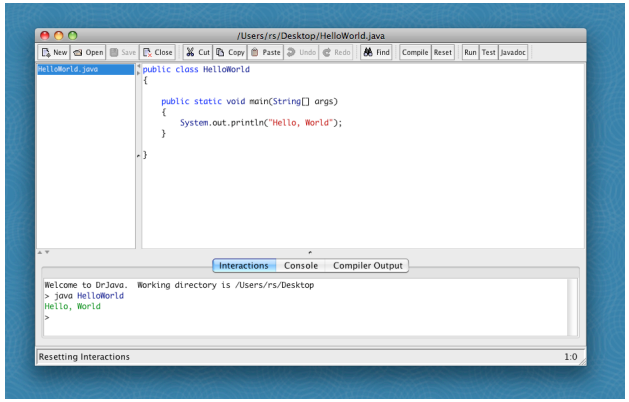
- Good enough for long programs?
- Dealing with two applications.

Integrated Development Environment

Use a customized application for all program development tasks.

Ex. DrJava

http://drjava.org



Pros:

- Easy-to-use language-specific tools.
- System-independent (in principle).
- Used by professionals.

Cons:

- Overkill for short programs?
- Large application to learn and maintain.
- Skills may not transfer to other languages.

Lessons from Short History

First requirement in any computer system: program development

Programming is primarily a process of finding and fixing mistakes.

Program development environment must support cycle of editing to fix errors, compiling program, running program, and examining output.

Two approaches that have served for decades:

- editor and virtual terminal
- integrated development environment

Macbook Air 2008

Xerox Alto 1978

1.2 Built-in Types of Data

Built-in Data Types

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

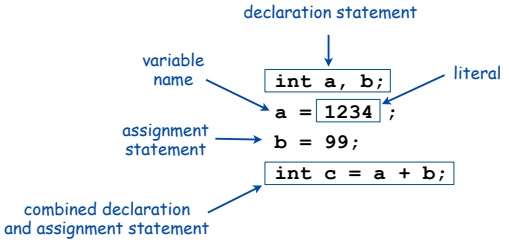
type	set of values	literal values	operations
`char`	characters	`'A'` `'@'`	compare
`String`	sequences of characters	`"Hello World"` `"CS is fun"`	concatenate
`int`	integers	`17` `12345`	add, subtract, multiply, divide
`double`	floating-point numbers	`3.1415` `6.022e23`	add, subtract, multiply, divide
`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.



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

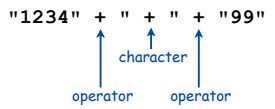
Text

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

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

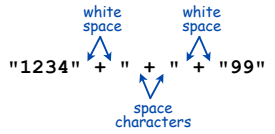
`String` data type

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.

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

```
% 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

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

also defined for int, long, and float

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

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

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
-0.6180339887498949

% 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	flase	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

TEQ on Type Conversion

[not difficult if you read Exercise 1.2.6]

What is the type and value of each of the following expression?

A. `( 7 / 2 ) * 2.0`

B. `( 7 / 2.0 ) * 2`

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.