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.

Ada Lovelace

Analytic Engine

well, almost anything [stay tuned]
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.
• Mars rover.
• Cell phones.
• Blu-ray Disc.
• Web servers.
• Medical devices.
• Supercomputing.
• ...

James Gosling
http://java.net/jag

$100 billion, 5 million developers
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>double</td>
</tr>
<tr>
<td>long</td>
<td>String</td>
</tr>
<tr>
<td>char</td>
<td>boolean</td>
</tr>
</tbody>
</table>

#### Flow Control

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>if</td>
<td>else</td>
</tr>
<tr>
<td>for</td>
<td>while</td>
</tr>
</tbody>
</table>

#### Boolean

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td></td>
</tr>
</tbody>
</table>

#### Punctuation

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td>}</td>
</tr>
<tr>
<td>(</td>
<td>)</td>
</tr>
<tr>
<td>,</td>
<td>;</td>
</tr>
</tbody>
</table>

#### System

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>System.out.println()</td>
</tr>
<tr>
<td>System.out.print()</td>
</tr>
<tr>
<td>System.out.printf()</td>
</tr>
</tbody>
</table>

#### Parsing

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer.parseInt()</td>
</tr>
<tr>
<td>Double.parseDouble()</td>
</tr>
</tbody>
</table>

#### Math Library

<table>
<thead>
<tr>
<th>Method</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math.sin()</td>
<td>Math.cos()</td>
</tr>
<tr>
<td>Math.log()</td>
<td>Math.exp()</td>
</tr>
<tr>
<td>Math.sqrt()</td>
<td>Math.pow()</td>
</tr>
<tr>
<td>Math.min()</td>
<td>Math.max()</td>
</tr>
<tr>
<td>Math.abs()</td>
<td>Math.PI</td>
</tr>
</tbody>
</table>

#### Primitive Numeric Types

<table>
<thead>
<tr>
<th>Operator</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>*</td>
<td>++</td>
</tr>
<tr>
<td>/</td>
<td>%</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>&lt;=</td>
<td>&gt;=</td>
</tr>
<tr>
<td>!=</td>
<td></td>
</tr>
</tbody>
</table>

#### String

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
</tr>
<tr>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>length()</td>
</tr>
<tr>
<td>compareTo()</td>
</tr>
<tr>
<td>charAt()</td>
</tr>
<tr>
<td>matches()</td>
</tr>
</tbody>
</table>

#### Arrays

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
</tr>
<tr>
<td>a.length</td>
</tr>
</tbody>
</table>

#### Objects

<table>
<thead>
<tr>
<th>Method</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>static</td>
</tr>
<tr>
<td>public</td>
<td>private</td>
</tr>
<tr>
<td>final</td>
<td>toString()</td>
</tr>
<tr>
<td>new</td>
<td>main()</td>
</tr>
</tbody>
</table>
Program Development

Edit → Compile → Run → Edit
• **Create** the program by typing it into a text editor, and save it as HelloWorld.java.

```java
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

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.

%
javac HelloWorld.java
%
java HelloWorld
Hello, World
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 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.
Three versions of the same program.

```java
// java HelloWorld
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.
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
**Data type.** A set of values and operations defined on those values.

<table>
<thead>
<tr>
<th>type</th>
<th>set of values</th>
<th>literal values</th>
<th>operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>characters</td>
<td>'A', '@'</td>
<td>compare</td>
</tr>
<tr>
<td>String</td>
<td>sequences of characters</td>
<td>&quot;Hello World&quot; &quot;CS is fun&quot;</td>
<td>concatenate</td>
</tr>
<tr>
<td>int</td>
<td>integers</td>
<td>17, 12345</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td>double</td>
<td>floating-point numbers</td>
<td>3.1415 6.022e23</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td>boolean</td>
<td>truth values</td>
<td>true false</td>
<td>and, or, not</td>
</tr>
</tbody>
</table>
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. Table of variable values after each statement.

<table>
<thead>
<tr>
<th>int a, b;</th>
<th>a</th>
<th>b</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>a = 1234;</td>
<td>1234</td>
<td>undefined</td>
<td>undefined</td>
</tr>
<tr>
<td>b = 99;</td>
<td>1234</td>
<td>99</td>
<td>undefined</td>
</tr>
<tr>
<td>int t = a;</td>
<td>1234</td>
<td>99</td>
<td>1234</td>
</tr>
<tr>
<td>a = b;</td>
<td>99</td>
<td>99</td>
<td>1234</td>
</tr>
<tr>
<td>b = t;</td>
<td>99</td>
<td>1234</td>
<td>1234</td>
</tr>
</tbody>
</table>
String data type. Useful for program input and output.

<table>
<thead>
<tr>
<th>values</th>
<th>sequences of characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>typical literals</td>
<td>&quot;Hello, &quot; &quot;1 &quot; &quot; * &quot;</td>
</tr>
<tr>
<td>operation</td>
<td>concatenate</td>
</tr>
<tr>
<td>operator</td>
<td>+</td>
</tr>
</tbody>
</table>

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
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);
    }
}

% 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

<table>
<thead>
<tr>
<th>Typical literals</th>
<th>Integers between $-2^{31}$ and $+2^{31} - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234 99 -99 0 1000000</td>
<td></td>
</tr>
</tbody>
</table>

### Operations

<table>
<thead>
<tr>
<th>Operators</th>
<th>Add</th>
<th>Subtract</th>
<th>Multiply</th>
<th>Divide</th>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>-</td>
<td>*</td>
<td>/</td>
<td>%</td>
<td></td>
</tr>
</tbody>
</table>

### Expressions

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 + 3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5 - 3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5 * 3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>5 / 3</td>
<td>1</td>
<td>no fractional part</td>
</tr>
<tr>
<td>5 % 3</td>
<td>2</td>
<td>remainder</td>
</tr>
<tr>
<td>1 / 0</td>
<td></td>
<td>run-time error</td>
</tr>
<tr>
<td>3 * 5 - 2</td>
<td>13</td>
<td>* has precedence</td>
</tr>
<tr>
<td>3 + 5 / 2</td>
<td>5</td>
<td>/ has precedence</td>
</tr>
<tr>
<td>3 - 5 - 2</td>
<td>-4</td>
<td>left associative</td>
</tr>
<tr>
<td>(3 - 5) - 2</td>
<td>-4</td>
<td>better style</td>
</tr>
</tbody>
</table>

**int data type**
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);
  }
}

% 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.

<table>
<thead>
<tr>
<th>Values</th>
<th>Approximations to real numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical literals</td>
<td>3.14159  6.022e23  -3.0  2.0  1.4142135623730951</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operations</th>
<th>+</th>
<th>-</th>
<th>*</th>
<th>/</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>3.141 + .03</code></td>
<td>3.171</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>3.141 - .03</code></td>
<td>3.111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>6.02e23/2</code></td>
<td>3.01E+23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>5.0 / 3.0</code></td>
<td>1.666666666666666700</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>10.0 % 3.141</code></td>
<td>0.577</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>1.0 / 0.0</code></td>
<td>Infinity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Math.sqrt(2.0)</code></td>
<td>1.4142135623730951</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Math.sqrt(-1.0)</code></td>
<td>NaN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*There is a largest double and a smallest double.*

Examples of double operations:

- `Math.sqrt(2.0)` returns a double value of 1.4142135623730951.
- `Math.sqrt(-1.0)` returns the special value NaN, which stands for "not a number."
Excerpts from Java’s Math Library

```java
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)
}
```

The Math class also defined for `int`, `long`, and `float`

Inverse functions `asin()`, `acos()`, and `atan()` also available.

In radians. Use `toDegrees()` and `toRadians()` to convert.
Ex. Solve quadratic equation \( x^2 + bx + c = 0 \).

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

```java
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. 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

\[ x^2 - 3x + 2 \]
\[ x^2 - x - 1 \]
\[ x^2 + x + 1 \]
**Booleans**

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

<table>
<thead>
<tr>
<th>values</th>
<th>true or false</th>
</tr>
</thead>
<tbody>
<tr>
<td>literals</td>
<td>true false</td>
</tr>
<tr>
<td>operations</td>
<td>and or not</td>
</tr>
<tr>
<td>operators</td>
<td>&amp;&amp;</td>
</tr>
</tbody>
</table>

**boolean data type**

| a | !a | a | b | a && b | a || b |
|---|----|---|---|-------|-------|
| true | false | false | false | false | false |
| false | true | false | true | false | true |
| true | false | true | false | false | true |
| true | true | true | true | true | true |

Truth-table definitions of boolean operations
Comparison operators.

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

<table>
<thead>
<tr>
<th>op</th>
<th>meaning</th>
<th>true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>equal</td>
<td>2 == 2</td>
<td>2 == 3</td>
</tr>
<tr>
<td>!=</td>
<td>not equal</td>
<td>3 != 2</td>
<td>2 != 2</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>2 &lt; 13</td>
<td>2 &lt; 2</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal</td>
<td>2 &lt;= 2</td>
<td>3 &lt;= 2</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>13 &gt; 2</td>
<td>2 &gt; 13</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal</td>
<td>3 &gt;= 2</td>
<td>2 &gt;= 3</td>
</tr>
</tbody>
</table>

Comparison examples:

- Non-negative discriminant? \(( b \times b - 4.0 \times a \times c ) \geq 0.0\)
- Beginning of a century? \((\text{year} \mod 100) == 0\)
- Legal month? \((\text{month} \geq 1) \&\& (\text{month} \leq 12)\)
Q. Is a given year a leap year?
A. Yes if either (i) divisible by 400 or (ii) divisible by 4 but not 100.

```java
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).

<table>
<thead>
<tr>
<th>expression</th>
<th>type</th>
<th>value</th>
<th>conversion type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;1234&quot; + 99</td>
<td>String</td>
<td>&quot;123499&quot;</td>
<td>automatic</td>
</tr>
<tr>
<td>Integer.parseInt(&quot;123&quot;)</td>
<td>int</td>
<td>123</td>
<td>explicit</td>
</tr>
<tr>
<td>(int) 2.71828</td>
<td>int</td>
<td>2</td>
<td>cast</td>
</tr>
<tr>
<td>Math.round(2.71828)</td>
<td>long</td>
<td>3</td>
<td>explicit</td>
</tr>
<tr>
<td>(int) Math.round(2.71828)</td>
<td>int</td>
<td>3</td>
<td>cast</td>
</tr>
<tr>
<td>(int) Math.round(3.14159)</td>
<td>int</td>
<td>3</td>
<td>cast</td>
</tr>
<tr>
<td>11 * 0.3</td>
<td>double</td>
<td>3.3</td>
<td>automatic</td>
</tr>
<tr>
<td>(int) 11 * 0.3</td>
<td>double</td>
<td>3.3</td>
<td>cast, automatic</td>
</tr>
<tr>
<td>11 * (int) 0.3</td>
<td>int</td>
<td>0</td>
<td>cast</td>
</tr>
<tr>
<td>(int) (11 * 0.3)</td>
<td>int</td>
<td>3</td>
<td>cast, automatic</td>
</tr>
</tbody>
</table>

Pay attention to the type of your data. Type conversion can give counterintuitive results but gets easier to understand with practice.
Ex. Generate a pseudo-random number between 0 and \( N-1 \).

```java
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);
    }
}
```

% java RandomInt 6
random integer is 3

% java RandomInt 6
random integer is 0

% java RandomInt 10000
random integer is 3184
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.

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
The if statement. A common branching structure.

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

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

The **if statement**. A common branching structure.

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

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

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

Diagram:

- if statement:
  - **boolean expression**
  - `if (x > y)`
  - `int t = x;`  
  - `x = y;`
  - `y = t;`

- Decision diagrams:
  - `x < 0`? 
  - yes: `x = -x;`
  - no
  - `x > y`? 
  - yes: `max = x;`
  - no: `max = y;`
If Statement

Ex. Take different action depending on value of variable.

```java
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. Take different action depending on value of variable.

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

```java
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");
    }
}
Oblivious Sorting

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

```java
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);
   }
}
```

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

Puzzle 2. Sort 6 integers with 12.

% java Sort3 9 8 7
7 8 9

% java Sort3 2 1 7
1 2 7
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:

```c
#include <stdio.h>

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 Rößler 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. It is well as 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.