1.1 Your First Program

Why Programming?

Why program?
- Need to tell computer what to do.
- Prepackaged software solutions:
  - Great, they do exactly what you want.
- Programming:
  - Enables you to make a computer do anything you want.

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

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.

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

It’s not about the language!
Programming in Java

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

```java
import java.util.Scanner;

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`.

Dr. Java

http://drjava.org

Dr. Java

command-line

javac HelloWorld.java

Dr. Java

command-line argument
Java is:
- Object oriented.
- Statically typed.
- Architecture neutral.
- Multi-threaded.
- Garbage-collecting.
- Robust.
- Small.
- Simple.
- Fast.
- Secure.
- Complex.

Java is not:
- Not new.
- Designed for toasters.
- Not done yet.
- Not as useful as C, C++, FORTRAN.
- Slow.
- Unsafe.
- Huge.

1.1 Extra Slides

Don’t believe anything on this slide! Make up your own mind.

1.2 Built-in Types of Data

Don’t believe anything on this slide! Make up your own mind.

Java Bytecode

<table>
<thead>
<tr>
<th>Java Bytecode</th>
<th>Language Details</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Basic Definitions

**Variable**: A name that refers to a value of declared type.
**Literal**: Programming language representation of a value.
**Assignment statement**: Associates a value with a variable.
Trace

Trace. Table of variable values after each statement.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>99</td>
<td>1234</td>
</tr>
<tr>
<td>1234</td>
<td>99</td>
<td>1234</td>
</tr>
<tr>
<td>99</td>
<td>1234</td>
<td>1234</td>
</tr>
<tr>
<td>1234</td>
<td>99</td>
<td>1234</td>
</tr>
</tbody>
</table>

Text

String data type. Useful for program input and output.

<table>
<thead>
<tr>
<th>values</th>
<th>sequences of characters</th>
<th>Concat. Meaning of characters depend on context</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Hello.&quot;</td>
<td>&quot;1 2 1&quot;</td>
<td>&quot;1234&quot; + &quot; 2 &quot; + &quot;1&quot;</td>
</tr>
<tr>
<td>&quot;1 2 1&quot;</td>
<td>&quot;1234 + &quot;99&quot;</td>
<td>&quot;1234 + 99&quot;</td>
</tr>
<tr>
<td>&quot;1234 + &quot;99&quot;</td>
<td>&quot;1234 + 99&quot;</td>
<td></td>
</tr>
</tbody>
</table>

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);
    }
}

Integers

Integers

... -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, ...

Values | Typical literals | Operators |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>+, -, *, /, %</td>
</tr>
</tbody>
</table>

Expressions | Value | Operators |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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>/</td>
</tr>
<tr>
<td>5 % 3</td>
<td>2</td>
<td>%</td>
</tr>
<tr>
<td>5 + 5 / 2</td>
<td>5</td>
<td>+, /</td>
</tr>
<tr>
<td>5 - 5 / 2</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>(3 + 5) + 2</td>
<td>10</td>
<td>+, ()</td>
</tr>
<tr>
<td>3 - (5 - 2)</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
**Integer Operations**

```java
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);
    }
}
```

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

- `javac IntOps.java`
- `java IntOps 1234 99`

**Floating-Point Numbers**

**Floating-Point Numbers (continued)**

<table>
<thead>
<tr>
<th>double data type</th>
<th>Usefulness in scientific applications</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>float</code></td>
<td>Small range, low precision</td>
</tr>
<tr>
<td><code>double</code></td>
<td>Large range, high precision</td>
</tr>
</tbody>
</table>

**Excerpts from Java's Math Library**

```java
public class Math {
    public static void main(String[] args) {
        double b = Double.parseDouble(args[0]);
        double c = Double.parseDouble(args[1]);
        double discriminant = b * b - 4.0 * c;
        if (discriminant > 0.0) {
            double root1 = (-b + Math.sqrt(discriminant)) / 2.0;
            double root2 = (-b - Math.sqrt(discriminant)) / 2.0;
            System.out.println(root1);
            System.out.println(root2);
        } else {
            System.out.println("No real roots");
        }
    }
}
```

**Ex.** Solve quadratic equation \(x^2 + bx + c = 0\).

```
x = \frac{-b \pm \sqrt{b^2 - 4c}}{2}
```

Testing

Some valid and invalid inputs.

```java
public class Testing {
    public static void main(String[] args) {
        try {
            int[] data = {1, 2, 3};
            System.out.println(Arrays.toString(data));
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(e.getMessage());
        }
        try {
            int[] data = {1, 2, 3};
            System.out.println(Arrays.toString(data[4]));
        } catch (IndexOutOfBoundsException e) {
            System.out.println(e.getMessage());
        }
    }
}
```
Booleans

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

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

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

Comparisons

**Comparisons.** Take two operands of one type (e.g., int) and produce a result 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>

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.

```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 |= (year % 400 == 0);
        System.out.println(isLeapYear);
    }
}
```

```
% java LeapYear 2004
true
% java LeapYear 1900
false
% java LeapYear 2000
true
```

Type Conversion

**Type conversion.** Convert value from one data type to another.

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

```java
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 $N - 1$.

```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);  // double to int
        System.out.println("random integer is "+n);
    }
}
```

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.
- Ex 1: in 1996, Ariane 5 rocket exploded after takeoff because of bad type conversion.
- Ex 2: $i = 0$ in Matlab redefines $\sqrt{-1}$.