Lecture 2: Intro to Java

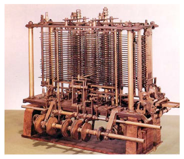
Why Programming?

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 need.

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

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



Ada Lovelace (left), Analytic Engine (right).

Languages

Machine languages. Tedious and error-prone.

Natural languages. Ambiguous and hard for computer to parse.

- Kids Make Nutritious Snacks.
- Milk Drinkers Turn to Powder.
- Red Tape Holds Up New Bridge.
- Police Squad Helps Dog Bite Victim.
- Tuna Biting Off Coast of Washington.
- Local High School Dropouts Cut in Half.

Reference: Rich Pattis

High-level programming languages. Acceptable tradeoff.

Why Java?

Java features.

- Widely used.
- Widely available.
- Embraces full set of modern abstractions.
- Variety of automatic checks for mistakes in programs.

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.

Caveat. No perfect language.

Our approach.
- Minimal subset of Java.
- Develop general programming skills that are applicable to: C, C++, C#, Python, Matlab, Fortran, Perl, …

A Rich Subset of the Java Language

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

Primitive Numeric Types		
+	-	*
/	%	++
--	>	<
<=	>=	==
<<	>>	|
&	^	!=

Parsing
Integer.parseInt()
Double.parseDouble()

Boolean	
true	false
||	&&
!	==

Punctuation	
{	}
(	)
,	;

Flow Control	
if	else
for	while
do	

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

Arrays
a[i]
new
a.length

Objects	
class	static
public	private
toString()	equals()
new	main()

1.1 Hello Java

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 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 prompt →

```
% 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`

command prompt →

```
% javac HelloWorld.java

% java HelloWorld
Hello, World
```

Programming in Java

Programming in Java (a slightly more realistic view)

1. Create the program by typing it into a text editor, and save it as `MyFancyProgram.java`
2. Compile it by typing at the command line:
   `javac MyFancyProgram.java`
3. Compiler says: That's not a legal program.
4. Back to step 1 to fix your errors of *syntax*. Repeat until none left.
5. Execute it by typing at the command line:
   `java MyFancyProgram`
6. Result is bizzarely (or subtley) wrong.
7. Back to step 1 to fix your errors of *semantics*. Repeat, etc.
8. After many attempts you finally get the result you wanted!

Hello, World

A few remarks.

- Name of class must match name of file.
- Comments between `/*` and `*/` are ignored by compiler.
- Whitespace and indentation is for human readability.
- Syntax coloration auto-generated by editor.

```
/*******************************************
 * Prints "Hello, World"
 * Everyone's first program.
 *******************************************/

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

HelloWorld.java

1.2 Primitive Types of Data

"Primitive" Data Types

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

Data Type	Description	Examples	Common Operations
`char`	character	`'A'` `'@'`	compare
`String`	sequence of characters	`"Hello World"` `"CS is fun"`	concatenate, compare
`int`	integer	`17` `12345`	add, subtract, multiply, remainder
`double`	floating point number	`3.1415` `2.17`	add, subtract, multiply, divide
`boolean`	truth value	`true` `false`	and, or, not, xor

Text

Text: the `String` data type.

- A sequence of Unicode characters.
- Each character is stored internally as a sequence of 16 bits:

0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	character '1' in Unicode

- Not technically a primitive type, but special language support.

Ex. 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;
      String ruler5 = ruler4 + "5 " + ruler4;
      System.out.println(ruler5);
   }
}
```

```
1
1 2 1
1 2 1 3 1 2 1
```

string concatenation

Ruler

```
% javac Ruler.java

% java Ruler
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
```

Integers

Integers: the `int` data type.

- Set of values: integers between `-2147483648` and `2147483647`. (-2^{31}, $2^{31} - 1$)
- Stored internally as a sequence of 32 bits:

`0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0`

14_{10}

- Useful when expressing algorithms.

Integers

```
public class IntOps {
   public static void main(String[] args) {
      int a = Integer.parseInt(args[0]);
      int b = Integer.parseInt(args[1]);
      int prod = a * b;
      int quot = a / b;
      int rem  = a % b;
      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 = 122166
1234 / 99 = 12
1234 % 99 = 46
```

`1234 = 12*99 + 46`

Initializing Variables

Q. What happens if I forget to initialize the variable `a` or `b`?

- Java compiler does not allow this.
- Caveat: in other languages, variable initialized to arbitrary value.

Q. What is default value for Registrar's room assignment variables?

Real Numbers

Real numbers: the `double` data type.

- Stored internally as a sequence of 64 bits using scientific notation.
- Useful in scientific applications.
- 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) {

      // read coefficients from command line
      double b = Double.parseDouble(args[0]);
      double c = Double.parseDouble(args[1]);

      // calculate roots
      double sqroot = Math.sqrt(b*b - 4.0*c);
      double root1 = (-b + sqroot) / 2.0;
      double root2 = (-b - sqroot) / 2.0;

      // print them out
      System.out.println(root1);
      System.out.println(root2);
   }
}
```

Command Line Arguments

Command line arguments. A simple method for processing a small amount of user input.

```
% java Quadratic -3.0 2.0
2.0
1.0                             command line arguments

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

% java Quadratic 1.0 1.0
NaN
NaN  ← not a number

% 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 and Comparisons

Booleans: the `boolean` data type.

- Set of values: `true` or `false`.
- Internally represented as one bit.
- Useful to control logic and flow of a program.

false	true
0	1

Logical Operators

a	b	a && b	a || b
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

a	! a
false	true
true	false

Comparison Operators

op	Description	true	false
==	equal	2 == 2	2 == 3
!=	not equal	2 != 3	2 != 2
<	less	2 < 3	3 < 2
<=	less or equal	2 <= 3	3 <= 2
>	greater	3 > 2	2 > 3
>=	greater or equal	3 >= 3	2 >= 3

Booleans and Comparisons

Q. Is a year a leap year?

A. Yes if divisible by 400, or 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: no loss of precision; or with strings.
- Explicit: cast; or method.

Ex. Generate a pseudo-random number between 0 and N-1.

- `Math.random()` returns a `double` between 0.0 and 1.0.

```
public class RandomInt {
   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      double r = Math.random(); ↖ String to int (method)
      int n = (int) (r * N);
                 ↑       ↑
      double to int (cast)   int to double (automatic)
      System.out.println("random integer is: " + n);
   }                                             ↑
}                                     int to String
                                      (automatic)
```

```
% java RandomInt 6
3
% java RandomInt 6
0
% java RandomInt 10000
3184
```

Summary

A data type is a set of values and operations on those values.

`String`	text processing.
`double`, `int`	mathematical calculation.
`boolean`	basis for decision making.

Be aware.

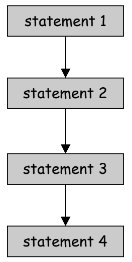
- Declare type of values.
- Convert between types when necessary.
- In 1996, Ariane 5 rocket exploded after takeoff because of bad type conversion.

2.3 Flow Control

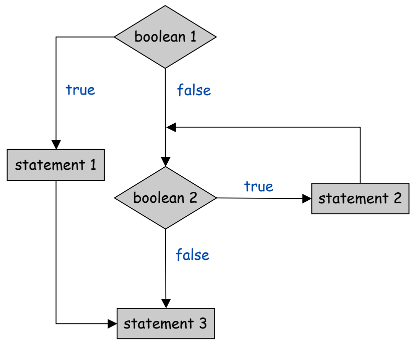
Flow-Of-Control

Flow-of-control.

- Sequence of statements that are actually executed in a program.
- Conditionals and loops: enable us to harness power of the computer.



straight-line flow-of-control



flow-of-control with conditionals and loops

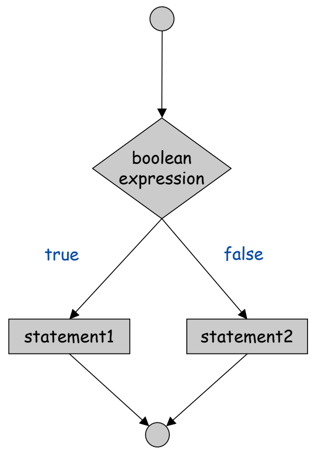
If-Else

The `if-else` statement is a common branching structure.

- Check `boolean` condition.
- If `true`, execute some statements.
- Otherwise, execute other statements.

```
if (boolean expression)
   statement1
                          can be a block
else                      of statements
   statement2
```

if-else syntax



if-else flow chart

If-Else: Leap Year

If-else example: print informative text.

- Different operation is performed depending on value of variable.
  - if `isLeapYear` is `true`, then print `"is a"`
  - otherwise, print `"isn't a "`

```
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.

```
public class Sort3 {
   public static void main(String[] args) {

      int a = Integer.parseInt(args[0]);      read in
      int b = Integer.parseInt(args[1]);      3 integers
      int c = Integer.parseInt(args[2]);
                                               swap a and b
      if (b < a) { int t = b; b = a; a = t; }
      if (c < b) { int t = c; c = b; b = t; }
      if (b < a) { int t = b; b = a; a = t; }

      System.out.println(a + " " + b + " " + c);
   }
}
```

```
% java Sort3 9 8 7
7 8 9

% java Sort3 2 1 7
1 2 7
```

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

Puzzle 2. Sort 6 integers with 12.