Lecture 2: Intro to Java

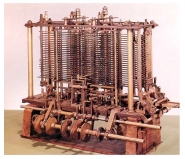
Why Programming?

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.

Ada Lovelace



Analytic Engine

Languages

> 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

Machine languages. Tedious and error-prone.

Natural languages. Ambiguous and hard for computer to parse.

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.

Caveat. No perfect language.

Our approach.

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

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	
{	}
(	)
,	;

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

Arrays
a[i]
new
a.length

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

1.1 Your First Program

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`

(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`

(or click the Run button in DrJava)

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 Java program.
 *****************************************/

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

HelloWorld.java

1.2 Built-In Types of Data

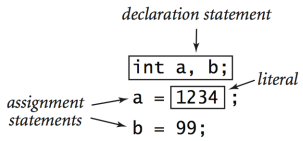
Built-In Data Types

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

Type	Description	Literals	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

Basics

Definitions.



Trace.

```
int a, b;
a = 1234;
b = 99;
int t = a;
a = b;
b = t;
```

a	b	t
undefined	*undefined*	
1234	*undefined*	
1234	99	
1234	99	1234
99	99	1234
99	1234	1234

Text

Text: the `String` data type.

- Values: sequences of Unicode characters.
- Operations: string concatenation.
- Useful for program input and output.

expression	*value*
`"Hi," + "Bob"`	`"Hi, Bob"`
`"1 " + "2 " + "1 "`	`"1 2 1"`
`"1234" + " + " + "99"`	`"1234 + 99"`
`"1234" + "99"`	`"123499"`

Typical `String` *expressions*

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

```
% 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. -2^{31} $2^{31} - 1$

- Values: integers between `-2147483648` and `2147483647`.
- Operations: add, subtract, multiply, divide, remainder.
- Useful for expressing algorithms.

expression	*value*	*comment*
`5 + 3`	8	
`5 - 3`	2	
`5 * 3`	15	
`5 / 3`	1	
`5 % 3`	2	
`1 / 0`		runtime 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

Typical `int` *expressions*

Integer Operations

```
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);
   }
}
```

command-line arguments

```
% javac IntOps.java
% java IntOps 1234 99
1234 * 99 = 122166
1234 / 99 = 12
1234 % 99 = 46
```

Java automatically converts rem to a String

1234 = 12*99 + 46

Floating Point Numbers

Floating point numbers: the `double` data type.

- Values: real numbers represented according to IEEE 754 standard.
- Operations: add, subtract, multiply, divide.
- Useful in scientific applications.

like scientific notation

expression	*value*
`3.141 + .03`	`3.171`
`6.02e23 - 2`	`6.02e23`
`6.02e23 / 2`	`3.01e23`
`5.0 / 3.0`	`1.6666666666666667`
`5.0 % 2.0`	`2.5`
`1.0 / 0.0`	`Infinity`
`Math.sqrt(2.0)`	`1.4142135623730951`
`Math.sqrt(-1.0)`	`NaN`

Typical `double` *expressions*

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 and Comparisons

Booleans: the `boolean` data type.

- Values: `true` or `false`.
- Operations: and, or, not.
- Useful to control logic and flow of a program.

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 < 13	3 < 2
<=	less or equal	2 <= 2	3 <= 2
>	greater	13 > 2	2 > 13
>=	greater or equal	3 >= 2	2 >= 3

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

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.

Be aware.

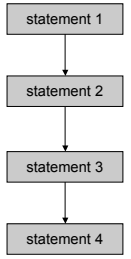
- Declare type of values.
- Convert between types when necessary.
- In 1996, Ariane 5 rocket exploded after takeoff because of bad type conversion.

1.3 Conditionals and Loops

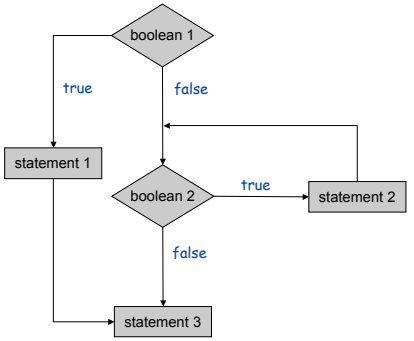
Control Flow

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

If-Else Statement

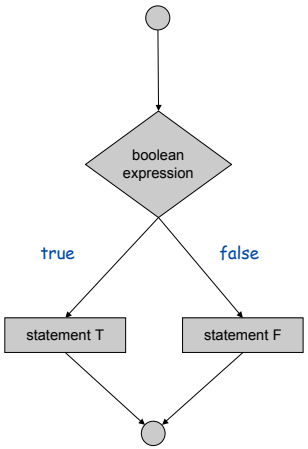
The `if-else` statement. A common branching structure.

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

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

can be any sequence of statements

if-else syntax



if-else flow chart

If-Else: Leap Year

If-else. Take different action 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]);
      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);
   }
}
```

read in 3 integers from command-line

swap b and c

swap a and b

swap b and 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.